STEVEN J. RIZZI (admitted *pro hac vice*)
srizzi@kslaw.com
RAMY HANNA (*pro hac vice* forthcoming)
rhanna@kslaw.com
KING & SPALDING LLP
1185 Avenue of the Americas, 35th Floor
New York, NY 10036
Telephone: (212) 556-2100
Facsimile: (212) 556-2222

RYAN A. SCHMID (*pro hac vice* forthcoming)
rschmid@kslaw.com
KING & SPALDING LLP
1700 Pennsylvania Avenue, N.W.
Washington, D.C. 20006
Telephone: (202) 737-0500
Facsimile: (202) 626-3737

RAMON A. MIYAR (CA SBN 284990)
rmiyar@kslaw.com
KING & SPALDING LLP
50 California Street, Suite 3300
San Francisco, CA 94111
Telephone: (415) 318-1200
Facsimile: (415) 318-1300

Attorneys for Plaintiff
EXPRESS MOBILE, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

EXPRESS MOBILE, INC.

Plaintiff,

v.

BOOKING HOLDINGS, INC., BOOKING.COM B.V., PRICELINE.COM LLC, AGODA COMPANY PTE. LTD., and OPENTABLE, INC.,

Defendants.

Civil Action No. 3:20-CV-08491-RS

PLAINTIFF EXPRESS MOBILE, INC.'S FIRST AMENDED COMPLAINT

Jury Trial Demanded

FIRST AMENDED COMPLAINT

Plaintiff Express Mobile, Inc. ("Express Mobile" or "Plaintiff"), by and through its undersigned counsel, brings this action for patent infringement against defendants Booking.com B.V. ("Booking-BV"), priceline.com LLC ("Priceline"), Agoda Company Pte. Ltd. ("Agoda), and OpenTable, Inc. ("OpenTable") (Booking B.V., Agoda, Priceline, and OpenTable collectively, "Defendants") and alleges as follows:

NATURE OF THE ACTION

1. This is a civil action arising under 35 U.S.C. § 271 for Defendants' infringement of Express Mobile's United States Patent Nos. 6,546,397 ("the '397 patent"), 7,594,168 ("the '168 patent"), 9,063,755 ("the '755 patent"), 9,471,287 ("the '287 patent"), and 9,928,044 ("the '044 patent") (collectively the "Patents-In-Suit").

THE PARTIES

2. Plaintiff Express Mobile, Inc. is a Delaware corporation with a place of business at 38 Washington Street, Novato, CA 94947.

3. Defendants are each wholly owned subsidiaries of Booking Holdings, Inc., and collectively offer an assortment of online travel and restaurant searching and reservation products and services, including the Accused Instrumentalities (defined *infra*, ¶ 55), throughout the United States, including in this District. In particular, Defendants and other companies under the Booking Holdings, Inc. umbrella market and provide these products and services through six widely-used and recognized e-commerce brands: (1) "Booking.com," which includes the www.Booking.com website, Booking.com mobile application, and Pulse mobile application, (2) "Kayak.com," which includes the www.Kayak.com website and Kayak mobile application, (3) "Priceline.com," which includes the www.priceline.com website and Priceline mobile application, (4) "Rentalcars.com," which includes the www.Rentalcars.com website and Rentalcars mobile application, (5) "Agoda.com," which includes the www.agoda.com website and Agoda mobile application, and (6) "OpenTable.com," which includes the www.opentable.com website and OpenTable mobile application. *See* https://ir.bookingholdings.com/node/24796/html, pp. 1-4 ("We offer these

services through six primary consumer-facing brands: Booking.com, KAYAK, priceline, agoda, Rentalcars.com and OpenTable. While historically our brands operated on a largely independent basis and many of them focused on a particular service (e.g., accommodation reservations) or geography, we are increasing the collaboration, cooperation and interdependency among our brands in our efforts to provide consumers with the best and most comprehensive services. We also seek to maximize the benefits of our scale by sharing resources and technological innovations, co-developing new services and coordinating activities in key markets among our brands. For example, Booking.com, the world's leading brand for booking online accommodation reservations (based on room nights booked), offers rental car and other ground transportation services, flights, restaurant reservations, tours and activities reservations and other services, many of which are supported by our other brands. Similarly, hotel reservations available through Booking.com are also generally available through agoda and priceline."); *see also* https://www.bookingholdings.com/about/factsheet/; https://ir.bookingholdings.com/investor-relations.

4. Upon information and belief, defendant Booking-BV is a company incorporated under the laws of the Kingdom of the Netherlands with its principal place of business in Amsterdam. Booking-BV is a wholly owned subsidiary of Booking Holdings, Inc., a Delaware corporation. Booking-BV directly and/or indirectly develops, designs, manufactures, distributes, markets, offers to sell and/or sells infringing products and services related to Booking.com in the United States, including in the Northern District of California, and otherwise purposefully directs infringing activities to this District in connection with its products and services.

5. Upon information and belief, defendant Priceline is a Delaware limited liability company with a principal place of business located at 800 Connecticut Avenue, Norwalk, Connecticut 06854. Priceline directly and/or indirectly develops, designs, manufactures, distributes, markets, offers to sell and/or sells infringing products and services related to Priceline.com in the United States, including in the Northern District of California, and otherwise purposefully directs infringing activities to this District in connection with its products and

services.

6. Upon information and belief, defendant Agoda is a Singapore private limited liability company with a principal place of business located at 30 Cecil Street, Prudential Tower #19-08, Singapore 049712. Agoda directly and/or indirectly develops, designs, manufactures, distributes, markets, offers to sell and/or sells infringing products and services related to Agoda.com in the United States, including in the Northern District of California, and otherwise purposefully directs infringing activities to this District in connection with its products and services.

7. Upon information and belief, defendant OpenTable is a corporation organized under the laws of the State of Delaware with a principal place of business located at 1 Montgomery Street, Suite 700, San Francisco, California 94104. OpenTable directly and/or indirectly develops, designs, manufactures, distributes, markets, offers to sell and/or sells infringing products and services related to OpenTable.com in the United States, including in the Northern District of California, and otherwise purposefully directs infringing activities to this District in connection with its products and services.

JURISDICTION

8. This is a civil action for patent infringement arising under the patent laws of the United States, 35 U.S.C. § 1 *et seq.*, including specifically 35 U.S.C. § 271.

9. This Court has subject matter jurisdiction over the matters pleaded herein under 28 U.S.C. §§ 1331 and 1338(a).

10. This Court has personal jurisdiction over Defendants because they have, jointly or individually:

(1) purposefully availed themselves of the rights and benefits of the laws of this State and this Judicial District**;**

(2) transacted, conducted, and/or solicited business and engaged in a persistent course of conduct in the State of California (and in this District) directly or through intermediaries**;**

(3) each derived substantial revenue from the sales and/or use of one or more infringing products and services in the State of California (and in this District) (including, without limitation, the **(i)** WebDirect website building platform (the "WebDirect Platform"); **(ii)** the Booking-BV website building platform (the "Booking Platform") available for and through Booking.com, the https://join.booking.com/ website and related mobile application, and the https://partner.booking.com/en-us website and related mobile application; **(iii)** the Priceline Agoda Global Partner Services website building platform (the "YCS Platform") available for and through Agoda.com and Priceline.com at the https://ycs.agoda.com/en-us/kipp/public/home website and related mobile application; and **(iv)** the OpenTable restaurant reservation software platform (the "OpenTable Platform"), available for and through OpenTable.com at https://restaurant.opentable.com/ website and related mobile application;

(4) purposefully directed activities (directly and/or through intermediaries), such as distributing, offering for sale, selling, marketing, and/or advertising their products and services, at residents of the State of California (and residents in this District)**;**

(5) delivered their products and services into the stream of commerce with the expectation that the such products and services will be used and/or purchased by consumers in the State of California (and in this District)**;** and

(6) committed, contributed to, and/or induced acts of patent infringement in the State of California (and in this District).

11. In particular, Defendants have committed and continue to commit acts of infringement in violation of 35 U.S.C. § 271, and have made, used, marketed, distributed, offered for sale, sold, and/or imported infringing products and services in/into the State of California, including in this District, and engaged in infringing conduct within and directed at or from this District. For example, Defendants have purposefully and voluntarily placed their brands'

respective websites, website building platforms, and mobile applications into the stream of commerce with the expectation that such websites, website building platforms, and mobile applications will be used in this District. Defendants' websites, website building platforms and mobile applications have been and continue to be distributed to and used in this District. Defendants' acts cause and have caused injury to Express Mobile, including within this District.

<u>VENUE</u>

12. Venue is proper in this Judicial District under 28 U.S.C. §§ 1391 and 1400(b).

13. Venue is proper as to Booking-BV and Agoda under 28 U.S.C. § 1391(c)(3) because Booking-BV and Agoda are not residents of any judicial district of the United States and therefore may be sued in any judicial district.

14. Venue is proper as to OpenTable in this Judicial District under 28 U.S.C. §§ 1391 and 1400(b) based on the information and belief that (1) OpenTable has committed, contributed to, and/or induced acts of infringement, and/or has advertised, marketed, sold, and/or offered to sell products and services, including infringing products and services, in this Judicial District, as discussed, *supra* ¶¶ 3, 7, and 10-11, which are incorporated by reference herein; and (2) OpenTable maintains at least one regular and established place of business in this Judicial District via its office located at 1 Montgomery Street, Ste 700, San Francisco, California, 94104.

15. Venue is proper as to Priceline in this Judicial District based on information and belief that (1) Priceline has committed, contributed to, and/or induced acts of infringement, and/or has advertised, marketed, sold, and/or offered to sell products and services, including infringing products and services, in this Judicial District, as discussed, *supra* ¶¶ 3, 5, and 10-11, which are incorporated by reference herein, and (2) Defendants have voluntarily waived any objections to venue in the Northern District of California (*see* ECF No. 25; attached as Exhibit I).

<u>THE PATENTS-IN-SUIT</u>

16. On April 8, 2003, United States Patent No 6,546,397 entitled "Browser Based Web Site Generation Tool and Run Time Engine," was duly and legally issued to Steven H. Rempell after full and fair examination. Plaintiff is the lawful owner of all right, title, and interest in and to

the ‘397 patent, including the right to recover for infringement thereof. A copy of the ‘397 patent is attached as Exhibit A.

17. The claimed inventions of the ‘397 patent solve technical problems related to the creation and generation of websites. For example, the inventions enable the creation of websites through browser-based visual editing tools, for example, selectable settings that describe website elements, with one or more settings corresponding to commands. These features are implemented utilizing computer technology, including a virtual machine.

18. The claims of the ‘397 patent do not merely describe performing some known business practice on the Internet. Instead, the claims of the ‘397 patent recite inventive concepts that are rooted in computerized website creation technology and overcome problems specific to this realm.

19. The claimed inventions of the ‘397 patent do not merely apply routine or conventional technologies for website creation and generation. Instead, the claims describe a browser-based website creation system and method in which information representing user-selected settings for a website are stored in a database, and the stored information is retrieved to generate the website.

20. The claims in the ‘397 patent do not preempt all ways of creating and generating websites or web pages, all uses of website authoring tools, nor any other well-known prior art technology.

21. Each claim of the ‘397 patent thus recites a combination of elements sufficient to ensure that the claim amounts to significantly more than a patent- ineligible concept.

22. On September 22, 2009, United States Patent No 7,594,168 entitled “Browser Based Web Site Generation Tool and Run Time Engine,” was duly and legally issued to Steven H. Rempell after full and fair examination. Plaintiff is the lawful owner of all right, title, and interest in and to the ‘168 patent, including the right to recover for infringement thereof. A copy of the ‘168 patent is attached as Exhibit B.

23. The claimed inventions of the ‘168 patent solve technical problems related to the

creation and generation of websites. For example, the inventions utilize browser-based build tools and a user interface to enable the creation of websites. These inventions greatly improve the productivity of the designer utilizing an innovative implementation for styles. These features are implemented utilizing computer technology.

24. The claimed inventions of the '168 patent do not perform a known business practice on the Internet. Instead, the claims of the '168 patent recite inventive concepts rooted in computerized website creation technology, and overcome problems specifically arising in this realm.

25. The claimed inventions of the '168 patent do not merely apply routine or conventional technologies for website creation and generation. Instead, the inventions describe a browser-based website creation system including a server comprising a build engine configured to create and apply styles to, for example, a website with web pages comprised of objects.

26. The claims in the '168 patent do not preempt all ways of creating and generating websites or web pages, all uses of website authoring tools, nor any other well-known or prior art technology.

27. Each claim of the '168 patent thus recites a combination of elements sufficient to ensure that the claim amounts to significantly more than a patent-ineligible concept.

28. In Case No. 3:18-CV-04679-RS, an infringement action filed by Plaintiff in the Northern District of California, the defendant in that action, Code and Theory LLC, brought a Motion to Dismiss Plaintiff's Complaint, asserting that the '397 and '168 patents do not claim patent-eligible subject matter under 35 U.S.C. § 101 as a matter of law. (Case No. 3:18-CV-04679-RS D.I. 35.) Subsequent briefing included Plaintiff Express Mobile, Inc.'s Opposition to Defendant Code and Theory LLC's Motion to Dismiss Plaintiff's Complaint (Case No. 3:18-CV-04679-RS D.I. 40), and Motion to Dismiss Plaintiff's Complaint [sic] (Case No. 3:18-CV-04679-RS D.I. 41). Each of those filings is incorporated by reference into this Complaint.

29. In Case No. 3:18-CV-04688-RS, an infringement action filed by Plaintiff in the Northern District of California, the defendant in that action, Pantheon Systems, Inc., brought a

Motion to Dismiss Counts I and II of Plaintiff's First Amended Complaint asserting that the '397 and '168 patents were directed to the abstract idea of creating and displaying webpages based upon information from a user with no further inventive concept, and purportedly ineligible for patenting under 35 U.S.C. § 101. (Case No. 3:18-CV-04688-RS D.I. 26.) Subsequent briefing included Plaintiff's Answering Brief in Opposition of Defendant's Motion to Dismiss (Case No. 3:18-CV-04688-RS D.I. 32), and Reply in Support of Defendant's Motion to Dismiss Counts I and II of Plaintiff's First Amended Complaint (Case No. 3:18-CV-04688-RS D.I. 34). Each of those filings is incorporated by reference into this Complaint.

30. After a motion hearing and a consideration of the respective pleadings, the Hon. Richard Seeborg denied both motions with respect to both patents in a joint order, because "the patents purport to describe a novel technological approach to creating websites on the internet." (Case No. 3:18-CV-04679-RS D.I. 45; Case No. 3:18-CV-04688-RS D.I. 40; attached as Exhibit F.) In denying the motions, Judge Seeborg made several findings:

- "The patents here are directed at a purportedly revolutionary technological solution to a technological problem—how to create webpages for the internet in a manner that permits 'what you see is what you get' editing, and a number of other alleged improvements over the then-existing methodologies." *Id*. at 5.
- The claims of the '397 and '168 patents are "directed to a specific improvement to the way computers operate," and "it simply cannot be said on the present record that the claims are drawn so broadly as to be divorced from the potentially patent-eligible purported technological improvements described in the specification." *Id*. at 6.

31. In C.A. 2:17-00128, an infringement action filed by Plaintiff in the Eastern District of Texas, the defendant in that action, KTree Computer Solutions, brought a Motion for Judgement on the Pleadings, asserting that the '397 and '168 patents were invalid as claiming abstract subject matter under 35 U.S.C. § 101. (C.A. 2:17-00128 D.I. 9.) Subsequent briefing included Plaintiff's Response and related Declarations and Exhibits (C.A. 2:17-00128 D.I. 17, 22-24), KTree's Reply

(C.A. 2:17-00128 D.I. 25), and Plaintiff's Sur-Reply and related Declarations and Exhibits (C.A. 2:17-00128 D.I. 26-27). Each of those filings is incorporated by reference into this Complaint.

32. After consideration of the respective pleadings, Magistrate Judge Payne recommended denial of KTree's motion, without prejudice, holding that "the claims appear to address a problem particular to the internet: dynamically generating websites and displaying web pages based on stored user-selected settings" and further stating "the asserted claims do not bear all of the hallmarks of claims that have been invalidated on the pleadings by other courts in the past. For example, the claims are not merely do-it-on-a-computer claims." (C.A. 2:17-00128 D.I. 29, attached as Exhibit G.) No objection was filed to the Magistrate Judge's report and recommendation and the decision therefore became final.

33. In Case Nos. 1:18-CV-01173-RGA and 1:18-CV-01175-RGA, infringement actions filed by Plaintiff in the District of Delaware, the respective defendants in those actions, Dreamhost LLC and Hostway Services, Inc., brought Motions to Dismiss claims of the '397 and '168 patents on the basis of invalidity under 35 U.S.C. § 101. (Case No. 1:18-CV-01173-RGA D.I. 14; Case No. 1:18-CV-01175-RGA D.I. 14.) Subsequent briefing included Plaintiff's Responses and related Declarations and Exhibits (Case No. 1:18-CV-01173-RGA D.I. 18-21; Case No. 1:18-CV-01175-RGA D.I. 17-19), and defendants' Replies (Case No. 1:18-CV-01173-RGA D.I. 24; Case No. 1:18-CV-01175-RGA D.I. 23). Each of these filings is incorporated by reference.

34. After consideration of the respective pleadings, Judge Andrews denied both motions in a joint order, pointing to factual allegations of inventiveness identified by the Plaintiff, and an expert declaration explaining inventiveness of the claims, noting that such factual issues preclude a finding of invalidity on a motion to dismiss. (Case No. 1:18-CV-01173-RGA D.I. 43; Case No. 1:18-CV-01175-RGA D.I. 42; attached as Exhibit H.)

35. On June 23, 2015, United States Patent No 9,063,755 entitled "Systems and methods for presenting information on mobile devices," was duly and legally issued to Steven H. Rempell, David Chrobak and Ken Brown after full and fair examination. Plaintiff is the lawful owner of all right, title, and interest in and to the '755 patent, including the right to recover for

infringement thereof. A copy of the ‘755 patent is attached as Exhibit C.

36. The inventions of the ‘755 patent utilize inventive concepts to solve technical problems, such as those associated with methods and systems for displaying dynamic content on displays of devices, providing more efficient ways of generating code for more uniformly displaying dynamic content across different kinds of devices. For example, the inventions of the ‘755 patent allow a data-efficient and flexible association between a symbolic name and a UI object (e.g., a UI object for a widget), corresponding to a web component of a web service, that is defined for presentation on a display of a device. A device-independent application including the symbolic name is produced and provided to the device, together with a device-platform-dependent player.

37. The claimed inventions of the ‘755 patent allow the UI object to be efficiently displayed across different kinds of devices (e.g., PC, mobile or tablet; or different browsers, operating systems, and applications, including also for example both native and browser-based applications). In turn, a user can enter an input value to the UI object and obtain an output value based on a web service associated with the UI object, the input value and output value also being communicated through symbolic names to provide an additional level of efficiency. These inventive features are implemented utilizing computer technology and solve technical problems in the prior art.

38. The claims of the ‘755 patent do not recite merely the performance of a known business practice on the Internet. Instead, the claims of the ‘755 patent recite inventive concepts concerning the computerized, data-efficient generation of content (e.g., a UI object for providing dynamic content) on displays for different types of devices, such as PC, tablet, or mobile devices, or different browsers and applications. For example, the claims of the ‘755 patent utilize symbolic name associations and provide device-independent applications including those symbolic names, together with device-platform-dependent players, to devices. Further, input values and output values for the defined content are also communicated as symbolic names. Such features are specifically grounded in and overcome problems with data efficiency and flexibility specifically

arising in, the realm of computerized content generation and display technologies, and are not well-understood, routine, and conventional elements.

39. For example, the claimed inventions of the ‘755 patent recite innovative, technical improvements that associate symbolic names with defined UI objects (e.g., UI objects for a widget) corresponding to web components of web services, and produce device-independent applications including those symbolic names, together with device-dependent players, to provide more uniform, data-efficient content display across different types of devices.

40. The technology claimed in the ‘755 patent does not preempt all ways for the computerized generation of code for a display of a device, nor any other well-known or prior art technology. For example, the specific, innovative technical improvements claimed in the ‘755 patent do not preempt well-known methods of generating code for a display of a device by programming in HTML or JavaScript code.

41. Each claim of the ‘755 patent thus recites a combination of elements sufficient to ensure that the claim amounts to significantly more than a patent on an ineligible concept.

42. On October 18, 2016, United States Patent No 9,471,287 entitled “Systems and Methods for Integrating Widgets on Mobile Devices,” was duly and legally issued to Steven H. Rempell, David Chrobak and Ken Brown after full and fair examination. Plaintiff is the lawful owner of all right, title, and interest in and to the ‘287 patent, including the right to recover for infringement thereof. A copy of the ‘287 patent is attached as Exhibit D.

43. The inventions of the ‘287 patent solve technical problems, such as those associated with methods and systems for displaying dynamic content on displays of devices by providing more efficient ways of generating code for more uniformly displaying dynamic content across different kinds of devices. For example, the inventions of the ‘287 patent allow a data-efficient and flexible association between a symbolic name and a UI object (e.g., a UI object for a widget) corresponding to a web component of a web service, that is defined for presentation on a display of a device. The defined UI object can be selected by a user of an authoring tool or automatically selected by a system based on a web component selected by the user. Further, the symbolic name

has a data format type corresponding to a subclass of UI objects that support the data format type of the symbolic name. A device-independent application including the symbolic name is then produced and provided to the device together with a device-platform-dependent player. Such operations provide a user-friendly platform allowing the UI object to be efficiently defined and more uniformly displayed across different kinds of devices (e.g., PC, mobile or tablet; or different browsers, operating systems, and applications, including also for example both native and browser-based applications). These features are implemented utilizing computer technology and solve technical problems in the prior art.

44. The claims of the ‘287 patent do not recite merely the performance of a known business practice on the Internet. Instead, the claims of the ‘287 patent recite inventive concepts grounded in the computerized, data-efficient definition and generation of content (e.g., a UI object for providing dynamic content) on displays for different types of devices, such as PC, tablet, or mobile devices, or different browsers and applications. Such features are specifically grounded in and overcome problems with data efficiency and flexibility specifically arising in, the realm of computerized content generation and display technologies, and are not well-understood, routine, and conventional elements.

45. For example, the claimed inventions of the ‘287 patent recite innovative, technical improvements that associate symbolic names with UI objects (e.g., UI objects for a widget) corresponding to web components of web services that are manually or automatically selected, and defined based on, for example, data format type, and produce device-independent applications including those symbolic names, together with device-dependent players, to provide more uniform, data-efficient content display across different types of devices.

46. The technology claimed in the ‘287 patent does not preempt all ways for the computerized generation of code for a display of a device nor any other well-known or prior art technology. For example, the specific, innovative technical improvements do not preempt well-known methods of generating code for a display of a device by programming in HTML or JavaScript code.

47. Each claim of the ‘287 patent thus recites a combination of elements sufficient to ensure that the claim amounts to significantly more than a patent on an ineligible concept.

48. On March 27, 2018, United States Patent No 9,928,044 entitled “Systems and Methods for Integrating Widgets on Mobile Devices,” was duly and legally issued to Steven H. Rempell, David Chrobak and Ken Brown after full and fair examination. Plaintiff is the lawful owner of all right, title, and interest in and to the ‘044 patent, including the right to recover for infringement thereof. A copy of the ‘044 patent is attached as Exhibit E.

49. The inventions of the ‘044 patent solve technical problems, such as those associated with methods and systems for displaying dynamic content on displays of devices by providing more efficient ways of generating, storing, and retrieving code for displaying dynamic content more uniformly across different kinds of devices. For example, the inventions of the ‘044 patent allow a data-efficient and flexible association between a symbolic name with a UI object (e.g., a UI object for a widget) corresponding to a web component of a web service, that is manually or automatically selected. The symbolic name has a data format type corresponding to a subclass of UI objects that support the data format type of the symbolic name, and is only available to UI objects that support the data format of the symbolic name. Information representative of the defined UI object can be stored in a database, and subsequently retrieved from the database to build an application consisting of at least a portion of the database using a player, which uses the information to generate one or more web pages for display across different kinds of devices (e.g., PC, mobile or tablet; or different browsers, operating systems, and applications, including also for example both native and browser-based applications). These features are implemented utilizing computer technology and solve technical problems in the prior art.

50. The claims of the ‘044 patent do not recite merely the performance of a known business practice on the Internet. Instead, the claims of the ‘044 patent recite inventive concepts grounded in the computerized, data-efficient definition, selection, storage and generation of content (e.g., a UI object providing dynamic content) on displays for different types of devices, such as PC, tablet, or mobile devices, or different browsers and applications. Such features are

specifically grounded in and overcome problems with data efficiency and flexibility specifically arising in, the realm of computerized content generation and display technologies, and are not well-understood, routine, and conventional elements.

51. For example, the claimed inventions of the ‘044 patent recite innovative, technical improvements that select and associate symbolic names with defined UI objects (e.g., UI objects for a widget) corresponding to web components of web services based on, for example, data format type, storing information representative of such settings in a database, and building applications, which together with players, generate more uniform, data-efficient content display across different types of devices.

52. The technology claimed in the ‘044 patent does not preempt all ways for the computerized generation of code for a display of a device nor any other well-known or prior art technology. For example, the specific, innovative technical improvements do not preempt well-known methods of generating code for a display of a device by programming in HTML or JavaScript code.

53. Each claim of the ‘044 patent thus recites a combination of elements sufficient to ensure that the claim amounts to significantly more than a patent on an ineligible concept.

BACKGROUND

54. Defendants are online travel and leisure companies providing customers with online tools and services to research and book accommodations, flights, rental cars, and restaurant reservations. At minimum, Defendants are each affiliated with each other in that they are all owned by wholly owned subsidiaries of the same parent holding company: Booking Holdings, Inc. Defendants provide services to merchants, end-users, and consumers through a series of websites and online tools and services:

55. Each of defendants Booking-BV, Priceline, Agoda, and OpenTable infringes the Patents-In-Suit by implementing Express Mobile’s patented technologies in a number of their products and services offered through Booking.com, Agoda.com, Priceline.com, and OpenTable.com, including, *inter alia*: (i) the WebDirect Platform; (ii) the Booking Platform for

and available through Booking.com, the https://join.booking.com/ website and related mobile application, and the https://partner.booking.com/en-us website and related mobile application; (iii) the YCS Platform available for and through Agoda.com and Priceline.com at the https://ycs.agoda.com/en-us/kipp/public/home website and related mobile application; and (iv) the OpenTable Platform available for and through OpenTable.com at https://restaurant.opentable.com/ website and related mobile application (collectively the "Accused Instrumentalities"). Defendants develop, market, sell, and distribute the Accused Instrumentalities, or products and services that use the Accused Instrumentalities, to consumers throughout the United States, including in this State and this Judicial District.

COUNT I – INFRINGEMENT OF U.S. PATENT NO. 6,546,397

56. Plaintiff incorporates by reference paragraphs 1 to 55 above as if fully set forth herein.

57. On information and belief, Defendants have infringed the '397 patent under 35 U.S.C. § 271, either literally and/or under the doctrine of equivalents, directly and/or indirectly.

58. On information and belief, Booking-BV has infringed the '397 patent by performing, without authority, one or more of the following acts during the term of the '397 patent: making, using, offering to sell, selling within, and importing into the United States products and services that practice the claimed inventions of the '397 patent, including but not limited to the WebDirect Platform and the Booking Platform.[1]

59. On information and belief, Priceline and Agoda (the "YCS infringers") have infringed the '397 patent by performing, without authority, one or more of the following acts during the term of the '397 patent: making, using, offering to sell, selling within, and importing

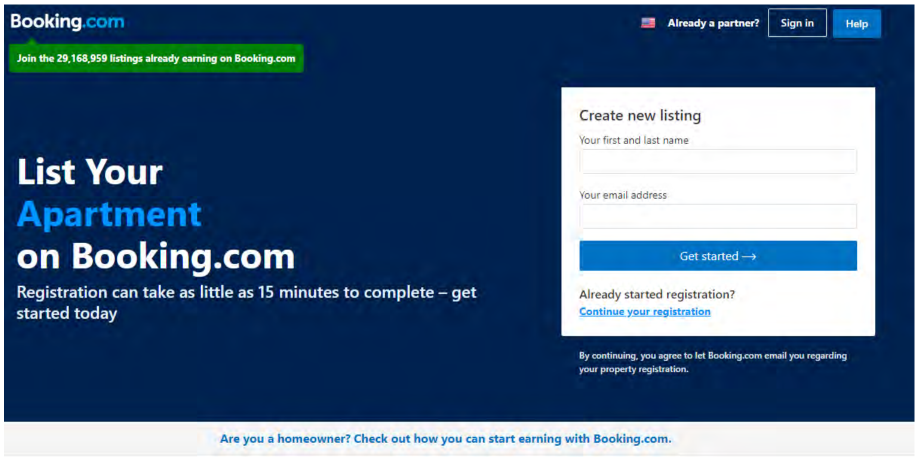
[1] This Count focuses its infringement allegations on the Booking Platform, YCS Platform, and the OpenTable Platform. Upon information and belief, the WebDirect Platform operates in a similar fashion as the Booking and YCS Platforms, and discovery of the WebDirect Platform, including confidential documents related thereto, will confirm these facts.

into the United States products and services that practice the claimed inventions of the ‘397 patent, including but not limited to the YCS Platform.

60. On information and belief, OpenTable has infringed the ‘397 patent by performing, without authority, one or more of the following acts during the term of the ‘397 patent: making, using, offering to sell, selling within, and importing into the United States products and services that practice the claimed inventions of the ‘397 patent, including but not limited to the OpenTable Platform.

The Booking Platform:

61. Booking-BV has infringed at least claim 1 of the ‘397 patent through a combination of features in the Booking Platform that collectively practiced each limitation of claim 1. By way of example, during the term of the ‘397 patent, Booking-BV provided the Booking Platform for creating property listing websites for hotels, homes, apartments, and other lodgings where travelers could view and book accommodations.


Booking.com

Already a partner? Sign in Help

Join the 29,168,959 listings already earning on Booking.com

List Your Apartment on Booking.com

Registration can take as little as 15 minutes to complete – get started today

Create new listing

Your first and last name

Your email address

Get started →

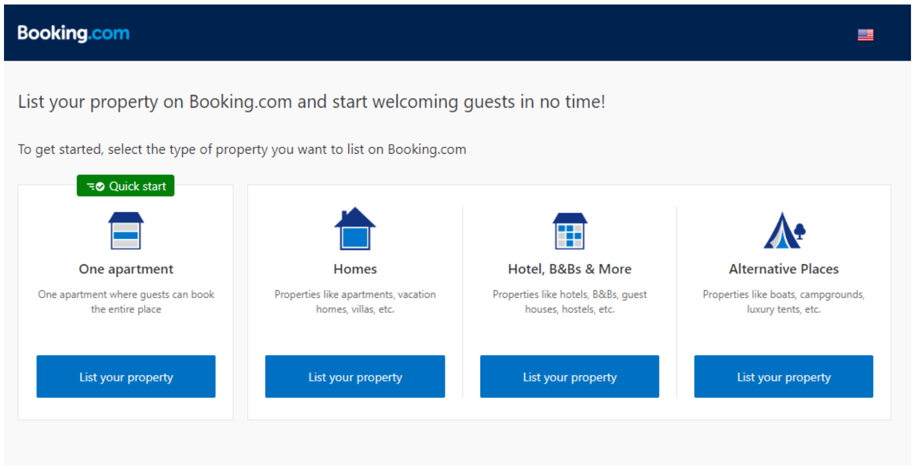
Already started registration?
Continue your registration

By continuing, you agree to let Booking.com email you regarding your property registration.

Are you a homeowner? Check out how you can start earning with Booking.com.

[2]

[2] Unless otherwise noted, the images presented in this Count were generated for investigative purposes by testing the Accused Instrumentalities on https://www.booking.com/, https://join.booking.com/, https://partner.booking.com/en-us, https://ycs.agoda.com/en-us/kipp/public/home, https://www.opentable.com/, https://support.opentable.com/s/?language=en_US, https://platform.opentable.com/documentation/, and/or other associated websites.



List your property on Booking.com and start welcoming guests in no time!

To get started, select the type of property you want to list on Booking.com

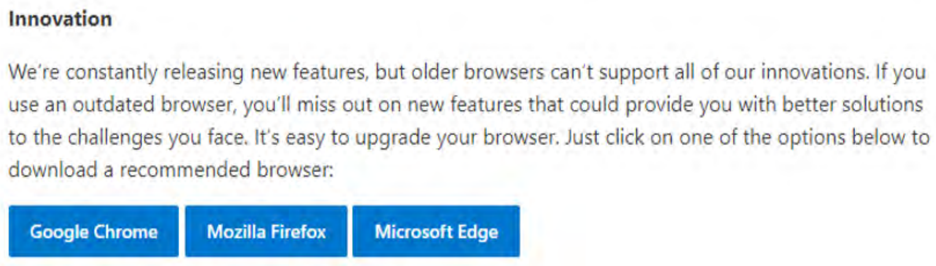
One apartment
One apartment where guests can book the entire place

Homes
Properties like apartments, vacation homes, villas, etc.

Hotel, B&Bs & More
Properties like hotels, B&Bs, guest houses, hostels, etc.

Alternative Places
Properties like boats, campgrounds, luxury tents, etc.

62. The property listings websites were, during the term of the '397 patent, created on and for computers having a browser and a virtual machine capable of generating displays. For example, the Booking Platform displayed content through modern browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge, that used browser engines (virtual machines) to render web pages on computers by interpreting and executing code such as JavaScript and HTML.



Innovation

We're constantly releasing new features, but older browsers can't support all of our innovations. If you use an outdated browser, you'll miss out on new features that could provide you with better solutions to the challenges you face. It's easy to upgrade your browser. Just click on one of the options below to download a recommended browser:

63. The Booking Platform allowed customization of the name, location, amenities, prices, descriptions, and images for a property listing website, through a menu with a panel of settings with which a user could select settings for elements on the website.

64. By way of example, a user could customize the images of the property to be displayed on the property listing's website through a panel that included options for uploading new images, dragging uploaded images to set the order in which the images were to be displayed, and tagging an image with a tag that would describe the image.

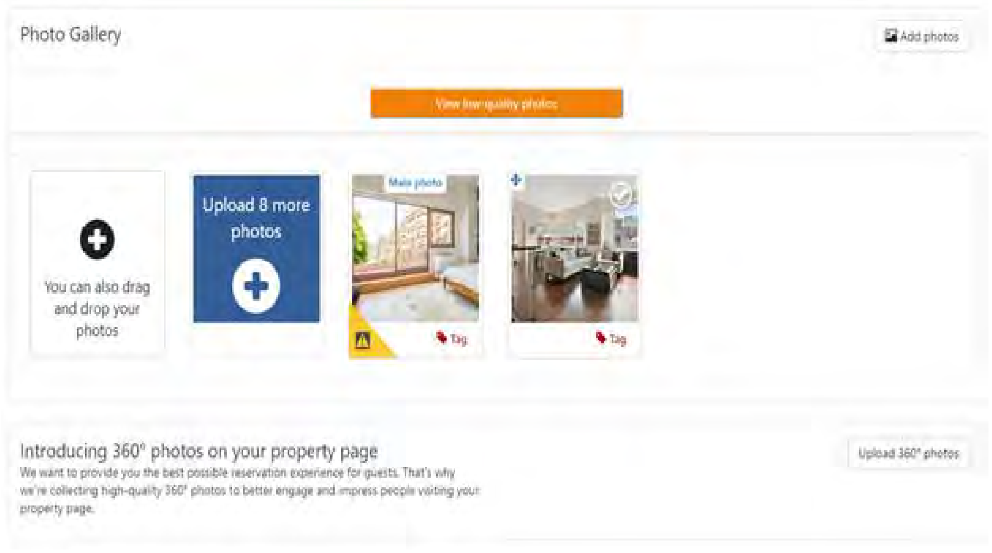
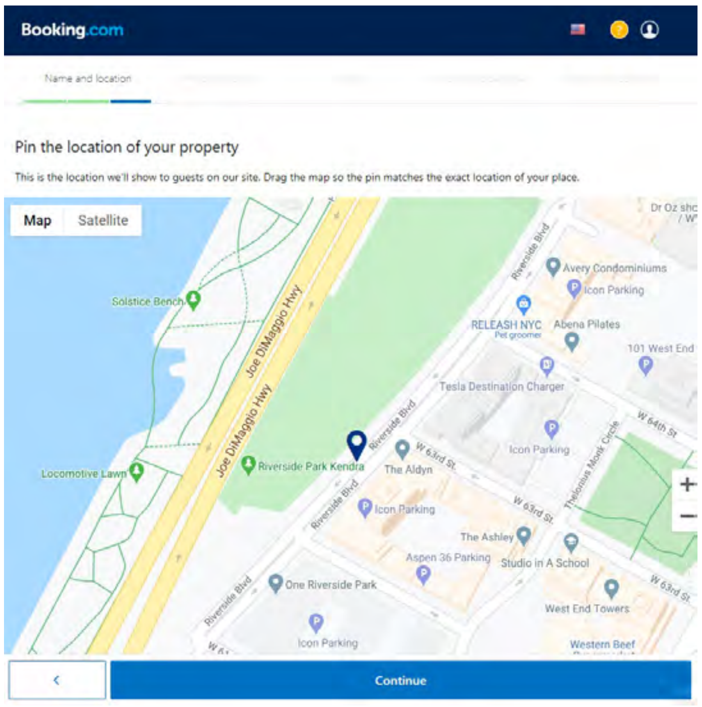


Photo Gallery
Add photos
View low quality photos
You can also drag and drop your photos
Upload 8 more photos
Main photo
Tag
Tag
Introducing 360° photos on your property page
We want to provide you the best possible reservation experience for guests. That's why we're collecting high-quality 360° photos to better engage and impress people visiting your property page.
Upload 360° photos

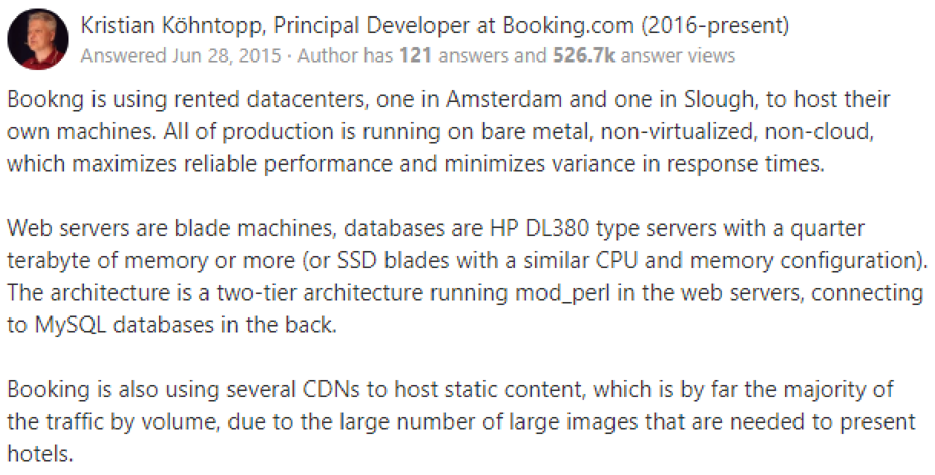
65. A user could also select settings to specify the property's location on a map by zooming in and out of or by dragging a pin on an interactive map.


Booking.com
Name and location
Pin the location of your property
This is the location we'll show to guests on our site. Drag the map so the pin matches the exact location of your place.
Map
Satellite
Joe DiMaggio Hwy
Riverside Park Kendra
Continue

66. As described above, the Booking Platform comprised a browser-based platform, and presented a user selectable panel of setting through a browser of a computer, which rendered websites using a browser engine (a virtual machine), based on commands to the browser engine, for display on the browser.

67. When a desired setting was selected, a display in accordance with that selection was generated substantially contemporaneously with the selection. In the example above, when a map location was selected on a map showing the property location, the setting was displayed substantially contemporaneously with the dragging and dropping of the pin. Similarly, when an image was selected and uploaded, the order of images was modified, or when an image was tagged, the settings were displayed substantially contemporaneously with the selections thereof.

68. Upon selection of the desired setting, information representative of the selected setting was stored in a database that supported the property listing website. In particular, websites for the Booking products were supported by a MySQL database, an open-source relational database management system, which served a backend database to retrieve and store data for Booking websites.


Kristian Köhntopp, Principal Developer at Booking.com (2016-present)
Answered Jun 28, 2015 · Author has **121** answers and **526.7k** answer views

Bookng is using rented datacenters, one in Amsterdam and one in Slough, to host their own machines. All of production is running on bare metal, non-virtualized, non-cloud, which maximizes reliable performance and minimizes variance in response times.

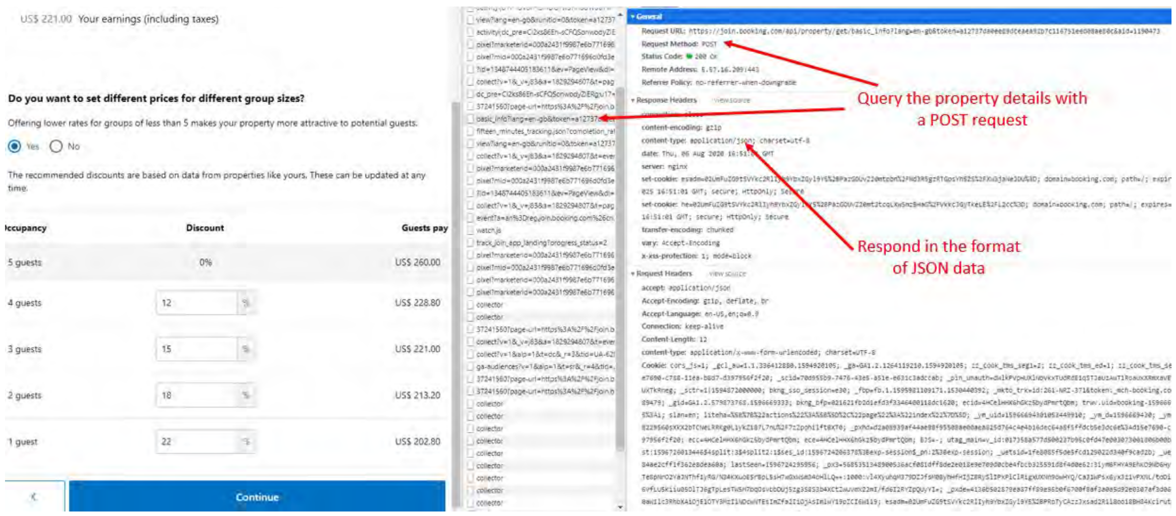
Web servers are blade machines, databases are HP DL380 type servers with a quarter terabyte of memory or more (or SSD blades with a similar CPU and memory configuration). The architecture is a two-tier architecture running mod_perl in the web servers, connecting to MySQL databases in the back.

Booking is also using several CDNs to host static content, which is by far the majority of the traffic by volume, due to the large number of large images that are needed to present hotels.

https://www.quora.com/Who-hosts-the-booking-com-website

69. A website for the property listing was generated in part by retrieving information and files for user selected settings stored in the MySQL database. As shown in the example below, if a user previously stored different price discounts (e.g., 22% for 1 guest, 18% for 2 guests, 15% for 3 guests, and 12% for 4 guests) for different groups of guests in the MySQL database in Booking's server, a browser could load the website by sending a HTTP POST request to query the property details and fetch the stored settings information (e.g., price discounts for 1 guest, 2 guests,

3 guests, and 4 guests) from the database in the Booking server "https://join.booking.com." In particular, the settings for the different price discounts (e.g., 22% for 1 guest, 18% for 2 guests, 15% for 3 guests, and 12% for 4 guests), were stored in the format of JSON data along with other property information, and communicated from Booking's server as a response to the HTTP POST request.


US$ 221.00 Your earnings (including taxes)

Do you want to set different prices for different group sizes?

Offering lower rates for groups of less than 5 makes your property more attractive to potential guests.

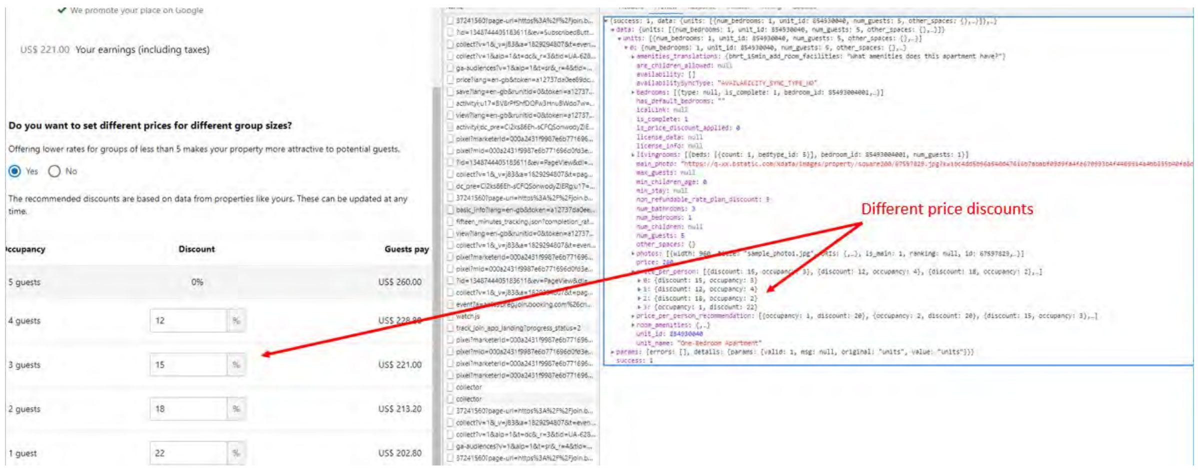
[✓ Yes] No

The recommended discounts are based on data from properties like yours. These can be updated at any time.

Occupancy	Discount	Guests pay
5 guests	0%	US$ 260.00
4 guests	12 %	US$ 228.80
3 guests	15 %	US$ 221.00
2 guests	18 %	US$ 213.20
1 guest	22 %	US$ 202.80

```
▾ General
    Request URL: https://join.booking.com/api/property/get/basic_info?lang=en-gb&token=a12737da0ee89dcea4a92b7c116751e6088ae80c&aid=1190473
    Request Method: POST
    Status Code: 200 OK
    Remote Address: 5.57.16.209:443
    Referrer Policy: no-referrer-when-downgrade
▾ Response Headers     view source
    content-encoding: gzip
    content-type: application/json; charset=utf-8
    date: Thu, 06 Aug 2020 16:51:01 GMT
    server: nginx
    set-cookie: esadm=02UmFuZG9tSVYkc2R1IyhRYbxZGy19YS%2BPazGOUv22emtzbn%2FNd3R5gzRTGosYh82S%2FXwGjaNe3OU%3D; domain=booking.com; path=/; expir
    025 16:51:01 GMT; secure; HttpOnly; Secure
    set-cookie: he=02UmFuZG9tSVYkc2R1IyhRYbxZGy19YS%2BPazGOUv22emt2tcqLXwSmzBHm%2FVkkc3GyTkeLE%2FL2cc%3D; domain=booking.com; path=/; expires=
    16:51:01 GMT; secure; HttpOnly; Secure
    transfer-encoding: chunked
    vary: Accept-Encoding
    x-xss-protection: 1; mode=block
▾ Request Headers     view source
    accept: application/json
    Accept-Encoding: gzip, deflate, br
    Accept-Language: en-US,en;q=0.9
    Connection: keep-alive
    Content-Length: 12
    content-type: application/x-www-form-urlencoded; charset=UTF-8
    Cookie: cors_js=1; _gcl_au=1.1.336412880.1594920105; _ga=GA1.2.1264119210.1594920105; ...
```

Query the property details with a POST request
Respond in the format of JSON data


✓ We promote your place on Google

US$ 221.00 Your earnings (including taxes)

Do you want to set different prices for different group sizes?

Offering lower rates for groups of less than 5 makes your property more attractive to potential guests.

[✓ Yes] No

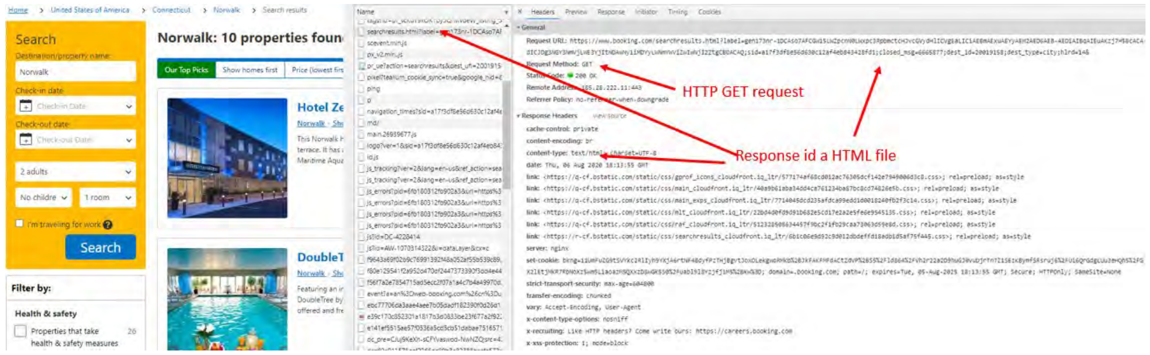
The recommended discounts are based on data from properties like yours. These can be updated at any time.

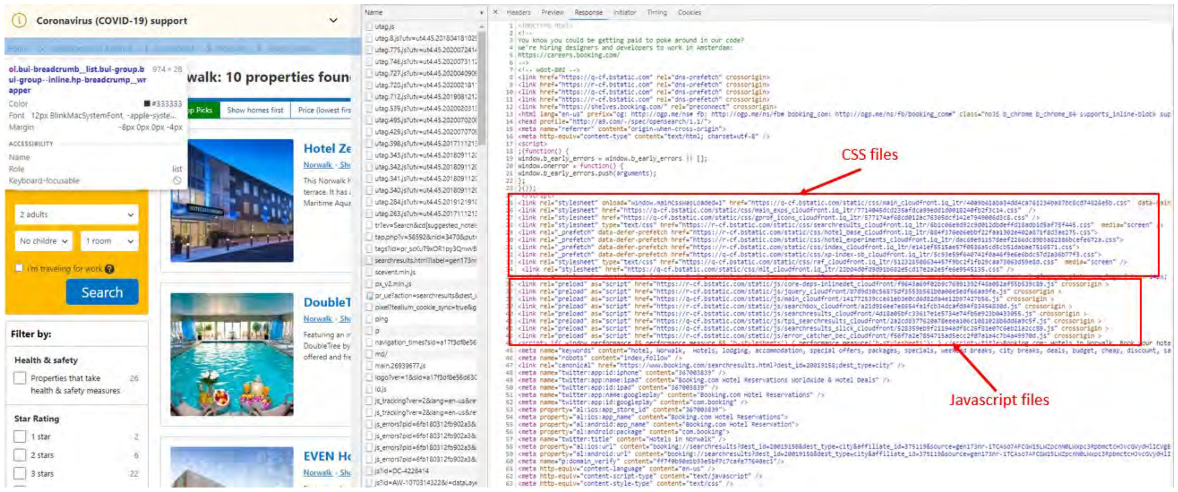
Occupancy	Discount	Guests pay
5 guests	0%	US$ 260.00
4 guests	12 %	US$ 228.80
3 guests	15 %	US$ 221.00
2 guests	18 %	US$ 213.20
1 guest	22 %	US$ 202.80

```
▾ {success: 1, data: {units: [{num_bedrooms: 1, unit_id: 854930040, num_guests: 5, other_spaces: {},…}]},…}
  ▾ data: {units: [{num_bedrooms: 1, unit_id: 854930040, num_guests: 5, other_spaces: {},…}]}
    ▾ units: [{num_bedrooms: 1, unit_id: 854930040, num_guests: 5, other_spaces: {},…}]
      ▾ 0: {num_bedrooms: 1, unit_id: 854930040, num_guests: 5, other_spaces: {},…}
        ▸ amenities_translations: {bhrt_15min_add_room_facilities: "What amenities does this apartment have?"}
          are_children_allowed: null
          availability: []
          availabilitySyncType: "AVAILABILITY_SYNC_TYPE_NO"
        ▸ bedrooms: [{type: null, is_complete: 1, bedroom_id: 85493004001,…}]
          has_default_bedrooms: ""
          icalLink: null
          is_complete: 1
          is_price_discount_applied: 0
          license_data: null
          license_info: null
        ▸ livingrooms: [{beds: [{count: 1, bedtype_id: 5}], bedroom_id: 85493004001, num_guests: 1}]
          main_photo: "https://q-xx.bstatic.com/xdata/images/property/square200/67597829.jpg?k=..."
          max_guests: null
          min_children_age: 0
          min_stay: null
          non_refundable_rate_plan_discount: 3
          num_bathrooms: 3
          num_bedrooms: 1
          num_children: null
          num_guests: 5
          other_spaces: {}
        ▸ photos: [{width: 960, title: "sample_photo1.jpg", URIS: {,…}, is_main: 1, ranking: null, id: 67597829,…}]
          price: 260
        ▾ price_per_person: [{discount: 15, occupancy: 3}, {discount: 12, occupancy: 4}, {discount: 18, occupancy: 2},…]
          ▸ 0: {discount: 15, occupancy: 3}
          ▸ 1: {discount: 12, occupancy: 4}
          ▸ 2: {discount: 18, occupancy: 2}
          ▸ 3: {occupancy: 1, discount: 22}
        ▸ price_per_person_recommendation: [{occupancy: 1, discount: 20}, {occupancy: 2, discount: 20}, {discount: 15, occupancy: 3},…]
        ▸ room_amenities: {,…}
          unit_id: 854930040
          unit_name: "One-Bedroom Apartment"
  ▸ params: {errors: [], details: {params: {valid: 1, msg: null, original: "units", value: "units"}}}
    success: 1
```

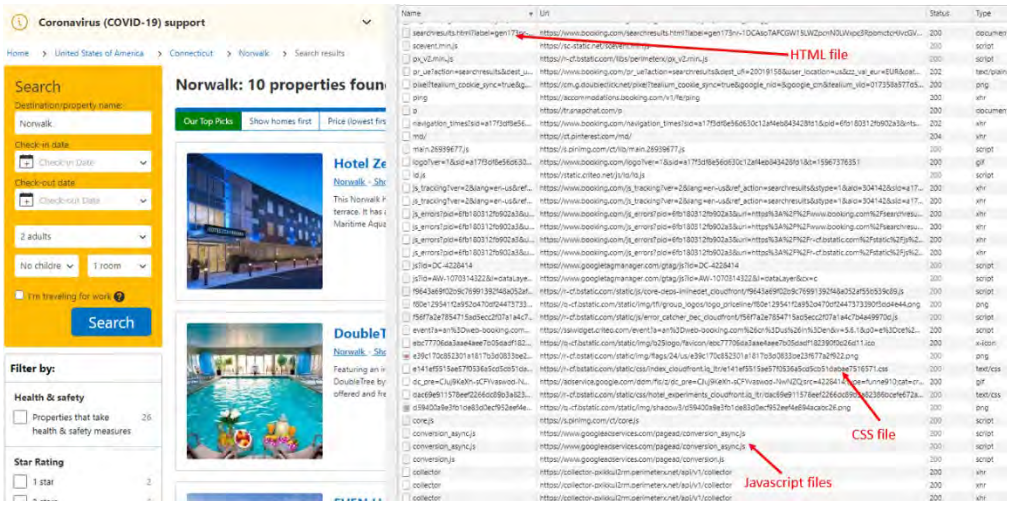
70. The Booking Platform built the property listing website comprising one or more web pages from the data for the selected settings and files stored in the database. Run time files (including CSS files and Javascript files) used the stored data and files to generate commands for the browser engine to display the one or more web pages. In particular, the Booking Platform relied on a browser engine to generate a website comprising one or more web pages based on settings

data extracted from at least a portion of the Booking Platform's database and at least one run time file. As shown in the example below, the Booking Platform downloaded an HTML file from the Booking Platform's servers with an HTTP GET request. When the browser build engine parsed the HTML file, the web browser made a request (e.g., GET method) to fetch the embedded CSS and Javascript run time files.





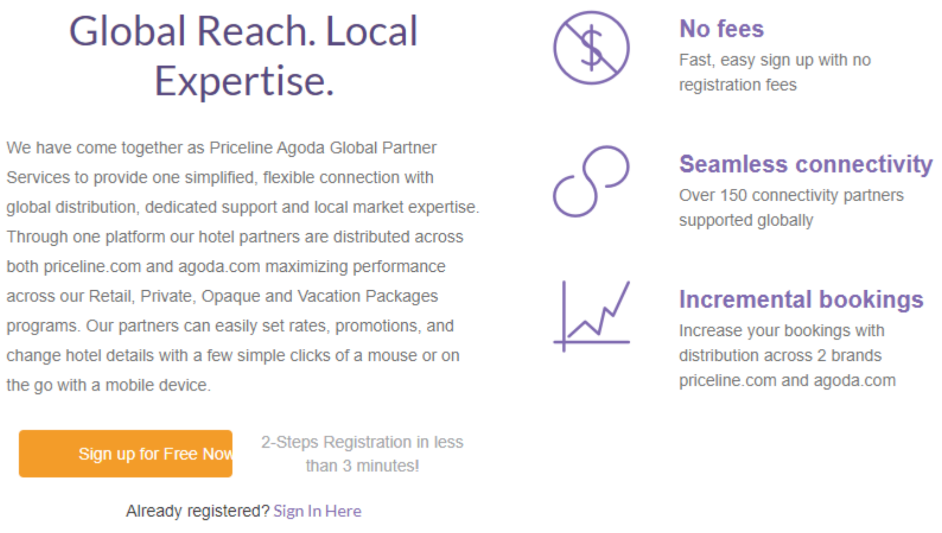
71. On information and belief, the Booking Platform fetched HTML (*.html) files, CSS (*.css) files, and Javascript (*.js) files from the Booking Platform's servers and converted them with additional contents (e.g., JSON files and image files) into a working website. In the example below, the Booking Platform's source code, such as HTML files and run time files (including CSS files, and Javascript files) were fetched from the Booking Platform's servers to build a "Search results" web page for display.



Name	Url	Status	Type
searchresults.html?label=gen173nr-...	https://www.booking.com/searchresults.html?label=gen173nr-1DCAsoTAFCGW15LWZpcnN0LWxpc3RpbmctcHVcGV...	200	documen
scevent.min.js	https://sc-static.net/scevent.min.js	200	script
px_v2.min.js	https://r-cf.bstatic.com/libs/perimeterx/px_v2.min.js	200	script
pr_ue?action=searchresults&dest_u...	https://www.booking.com/pr_ue?action=searchresults&dest_ufi=20019158&user_location=us&zz_val_eur=EUR&dat...	202	text/plain
pixel?tealium_cookie_sync=true&g...	https://cm.g.doubleclick.net/pixel?tealium_cookie_sync=true&google_nid=&google_cm&tealium_vid=017358a577d5...	200	png
ping	https://accommodations.booking.com/v1/fe/ping	200	xhr
p	https://tr.snapchat.com/p	200	documen
navigation_times?sid=a17f3df8e56...	https://www.booking.com/navigation_times?sid=a17f3df8e56d630c12af4eb843428fd1&pid=6fb180312fb902a3&ts...	202	xhr
md/	https://ct.pinterest.com/md/	204	xhr
main.26939677.js	https://s.pinimg.com/ct/lib/main.26939677.js	200	script
logo?ver=1&sid=a17f3df8e56d630...	https://www.booking.com/logo?ver=1&sid=a17f3df8e56d630c12af4eb843428fd1&t=15967376351	200	gif
ld.js	https://static.criteo.net/js/ld/ld.js	200	script
js_tracking?ver=2&lang=en-us&ref...	https://www.booking.com/js_tracking?ver=2&lang=en-us&ref_action=searchresults&stype=1&aid=304142&sid=a17...	200	xhr
js_tracking?ver=2&lang=en-us&ref...	https://www.booking.com/js_tracking?ver=2&lang=en-us&ref_action=searchresults&stype=1&aid=304142&sid=a17...	200	xhr
js_errors?pid=6fb180312fb902a3&u...	https://www.booking.com/js_errors?pid=6fb180312fb902a3&url=https%3A%2F%2Fwww.booking.com%2Fsearchresu...	200	xhr
js_errors?pid=6fb180312fb902a3&u...	https://www.booking.com/js_errors?pid=6fb180312fb902a3&url=https%3A%2F%2Fwww.booking.com%2Fsearchresu...	200	xhr
js_errors?pid=6fb180312fb902a3&u...	https://www.booking.com/js_errors?pid=6fb180312fb902a3&url=https%3A%2F%2Fr-cf.bstatic.com%2Fstatic%2Fjs%2...	200	xhr
js_errors?pid=6fb180312fb902a3&u...	https://www.booking.com/js_errors?pid=6fb180312fb902a3&url=https%3A%2F%2Fr-cf.bstatic.com%2Fstatic%2Fjs%2...	200	xhr
js?id=DC-4228414	https://www.googletagmanager.com/gtag/js?id=DC-4228414	200	script
js?id=AW-1070314322&l=dataLaye...	https://www.googletagmanager.com/gtag/js?id=AW-1070314322&l=dataLayer&cx=c	200	script
f9643a69f02b9c76991392f48a052af...	https://r-cf.bstatic.com/static/js/core-deps-inlinedet_cloudfront/f9643a69f02b9c76991392f48a052af55b539c89.js	200	script
f80e129541f2a952d470df24473733...	https://q-cf.bstatic.com/static/img/tfl/group_logos/logo_priceline/f80e129541f2a952d470df2447373390f3dd4e44.png	200	png
f56f7a2e7854715ad5ecc2f07a1a4c7...	https://r-cf.bstatic.com/static/js/error_catcher_bec_cloudfront/f56f7a2e7854715ad5ecc2f07a1a4c7b4a49970d.js	200	script
event?a=an%3Dweb-booking.com...	https://sslwidget.criteo.com/event?a=an%3Dweb-booking.com%26cn%3Dus%26ln%3Den&v=5.6.1&p0=e%3Dce%2...	200	script
ebc77706da3aae4aee7b05dadf182...	https://q-cf.bstatic.com/static/img/b25logo/favicon/ebc77706da3aae4aee7b05dadf182390f0c26d11.ico	200	x-icon
e39c170c852301a1817b3d0833be2...	https://r-cf.bstatic.com/static/img/flags/24/us/e39c170c852301a1817b3d0833be23f677a2f922.png	200	png
e141ef5515ae57f0536a5cd5cb51da...	https://r-cf.bstatic.com/static/css/index_cloudfront.iq_ltr/e141ef5515ae57f0536a5cd5cb51dabae7516571.css	200	text/css
dc_pre=CJuj9KeXn-sCFYvaswod-N...	https://adservice.google.com/ddm/fls/z/dc_pre=CJuj9KeXn-sCFYvaswod-NwNZQ;src=4228414;type=funne910;cat=cr...	200	gif
dac69e911578eef2266dc89b3a823...	https://r-cf.bstatic.com/static/css/hotel_experiments_cloudfront.iq_ltr/dac69e911578eef2266dc89b3a82386bcefe672a...	200	text/css
d59400a9e3fb1de83d0ecf952eef4e...	https://q-cf.bstatic.com/static/img/shadow3/d59400a9e3fb1de83d0ecf952eef4e694acabc26.png	200	png
core.js	https://s.pinimg.com/ct/core.js	200	script
conversion_async.js	https://www.googleadservices.com/pagead/conversion_async.js	200	script
conversion_async.js	https://www.googleadservices.com/pagead/conversion_async.js	200	script
conversion.js	https://www.googleadservices.com/pagead/conversion.js	200	script
collector	https://collector-pxlkkul2rm.perimeterx.net/api/v1/collector	200	xhr
collector	https://collector-pxlkkul2rm.perimeterx.net/api/v1/collector	200	xhr
collector	https://collector-pxlkkul2rm.perimeterx.net/api/v1/collector	200	xhr

<u>The YCS Platform:</u>

72. The YCS infringers infringed at least claim 1 of the '397 patent through a combination of features in the YCS Platform that collectively practiced each limitation of claim 1. By way of example, during the term of the '397 patent, the YCS infringers provided the YCS Platform to produce property listing websites for the Priceline Agoda products.


Global Reach. Local Expertise.

We have come together as Priceline Agoda Global Partner Services to provide one simplified, flexible connection with global distribution, dedicated support and local market expertise. Through one platform our hotel partners are distributed across both priceline.com and agoda.com maximizing performance across our Retail, Private, Opaque and Vacation Packages programs. Our partners can easily set rates, promotions, and change hotel details with a few simple clicks of a mouse or on the go with a mobile device.

Sign up for Free Now

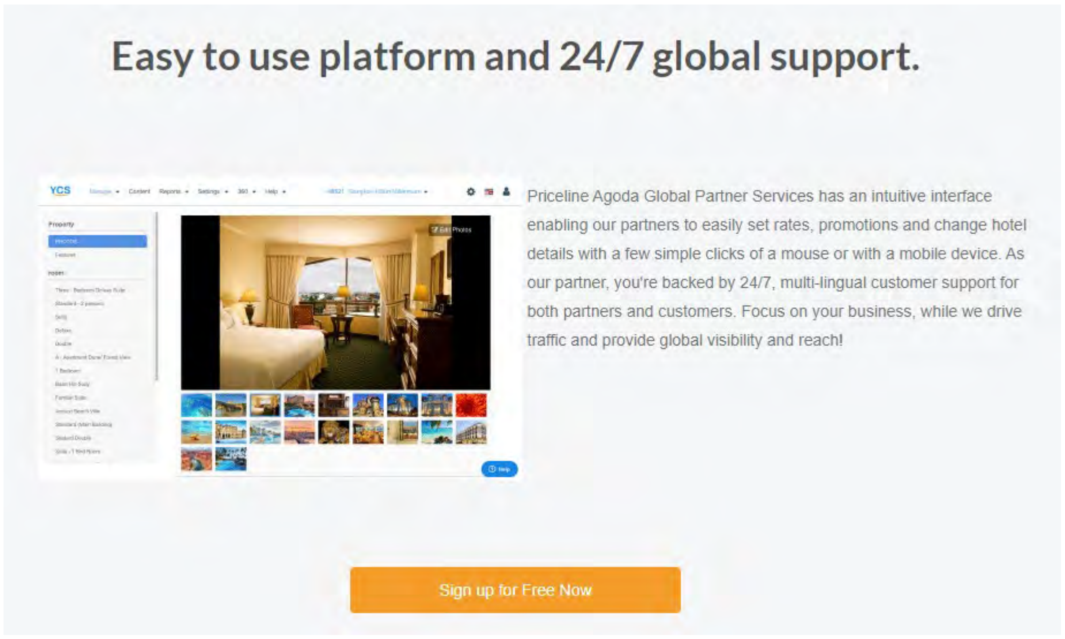
2-Steps Registration in less than 3 minutes!

Already registered? Sign In Here

No fees
Fast, easy sign up with no registration fees

Seamless connectivity
Over 150 connectivity partners supported globally

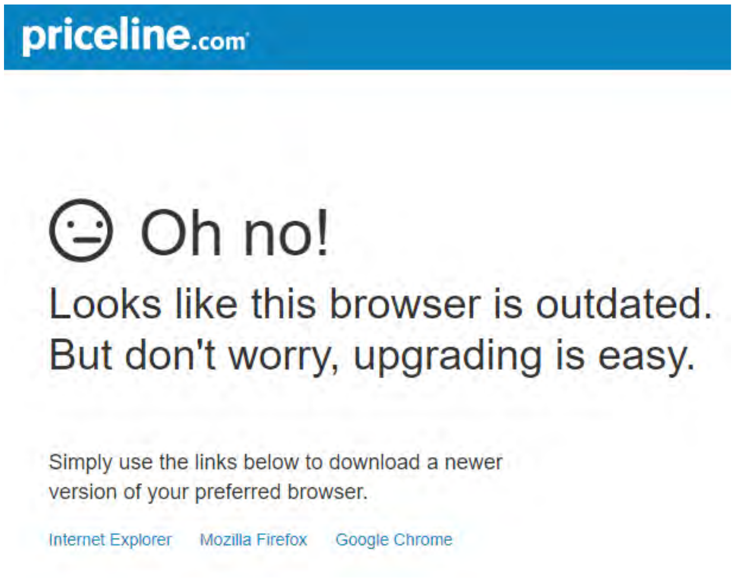
Incremental bookings
Increase your bookings with distribution across 2 brands priceline.com and agoda.com


Easy to use platform and 24/7 global support.

Priceline Agoda Global Partner Services has an intuitive interface enabling our partners to easily set rates, promotions and change hotel details with a few simple clicks of a mouse or with a mobile device. As our partner, you're backed by 24/7, multi-lingual customer support for both partners and customers. Focus on your business, while we drive traffic and provide global visibility and reach!

Sign up for Free Now

73. The property listings websites were, during the term of the '397 patent, created on and for computers having a browser and a virtual machine capable of generating displays. For example, Priceline.com websites displayed content through modern browsers such as Google Chrome, Mozilla Firefox, and Internet Explorer, which used browser engines (virtual machines) capable of generating a display by interpreting and executing code such as JavaScript and HTML to render web pages on a computer.


priceline.com

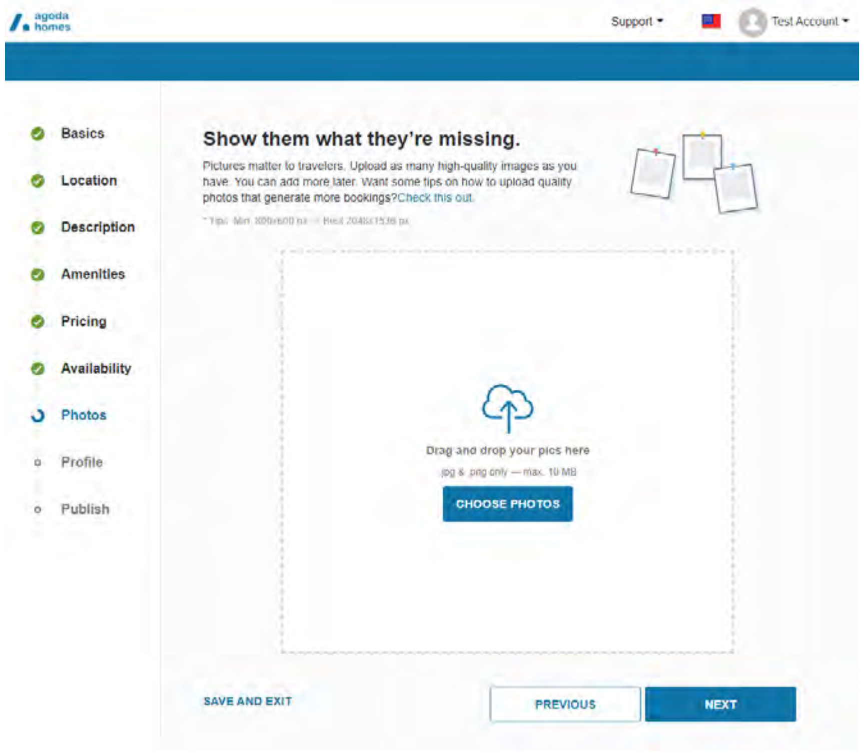
Oh no!

Looks like this browser is outdated. But don't worry, upgrading is easy.

Simply use the links below to download a newer version of your preferred browser.

Internet Explorer Mozilla Firefox Google Chrome https://www.priceline.com/static-pages/browser-upgrade.html

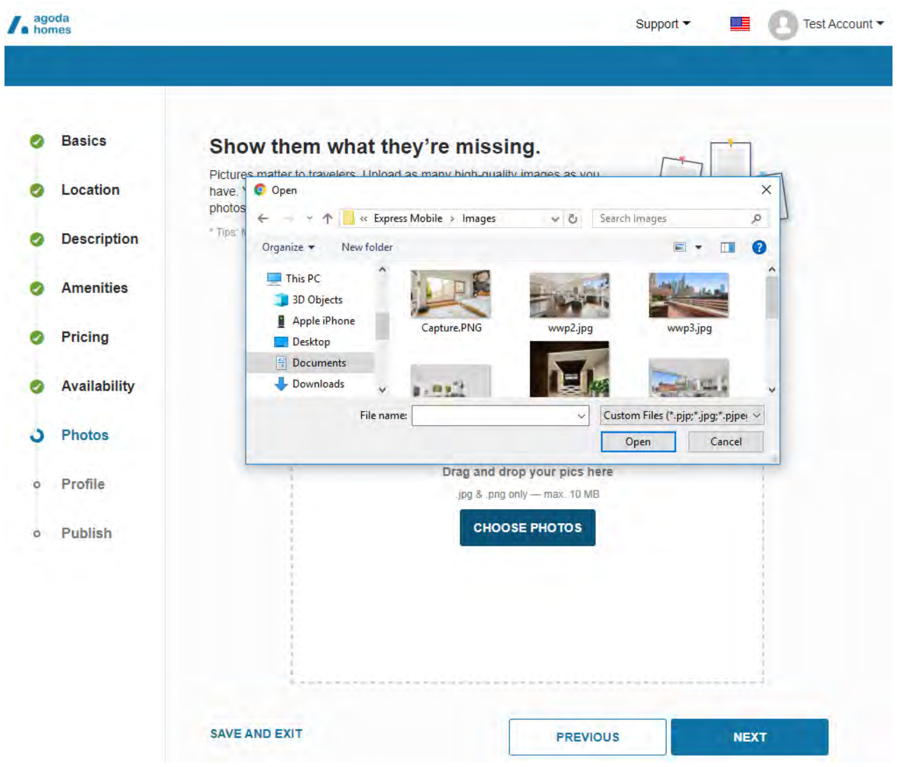
74. The YCS Platform allowed customization of the location, description, amenities, pricing, availability, photos, and profile for a Priceline.com or Agoda.com property listing website, through a menu having a panel of settings in which a user could select settings describing elements for the website. For example, the YCS Platform was configured to accept user settings for photos of the property, such as selection of photos to be displayed, selection of "main photo," and selection of a caption.

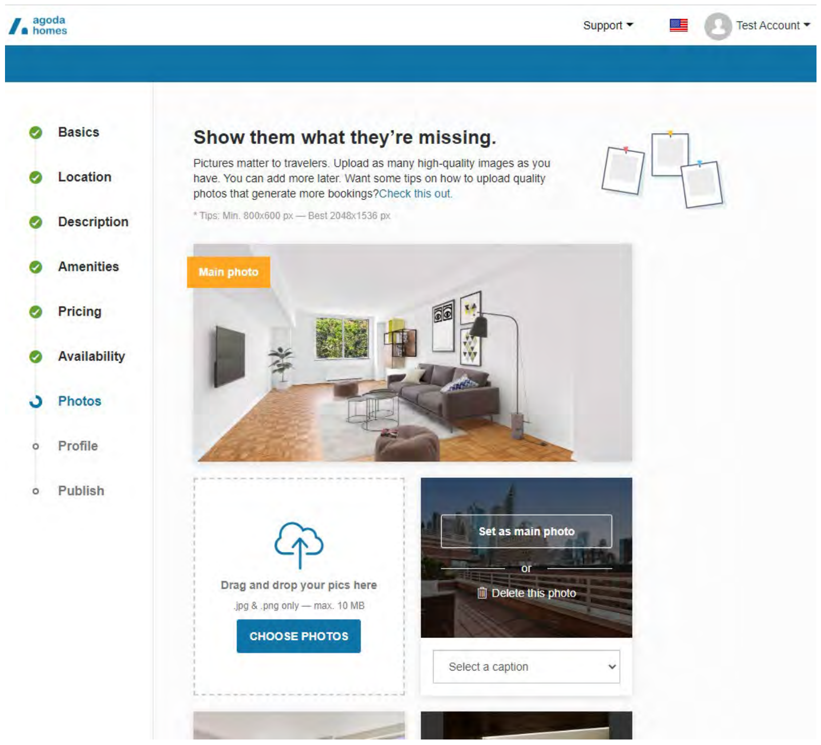


Show them what they're missing.

Pictures matter to travelers. Upload as many high-quality images as you have. You can add more later. Want some tips on how to upload quality photos that generate more bookings?Check this out.

* Tips: Min. 800x600 px — Best 2048x1536 px



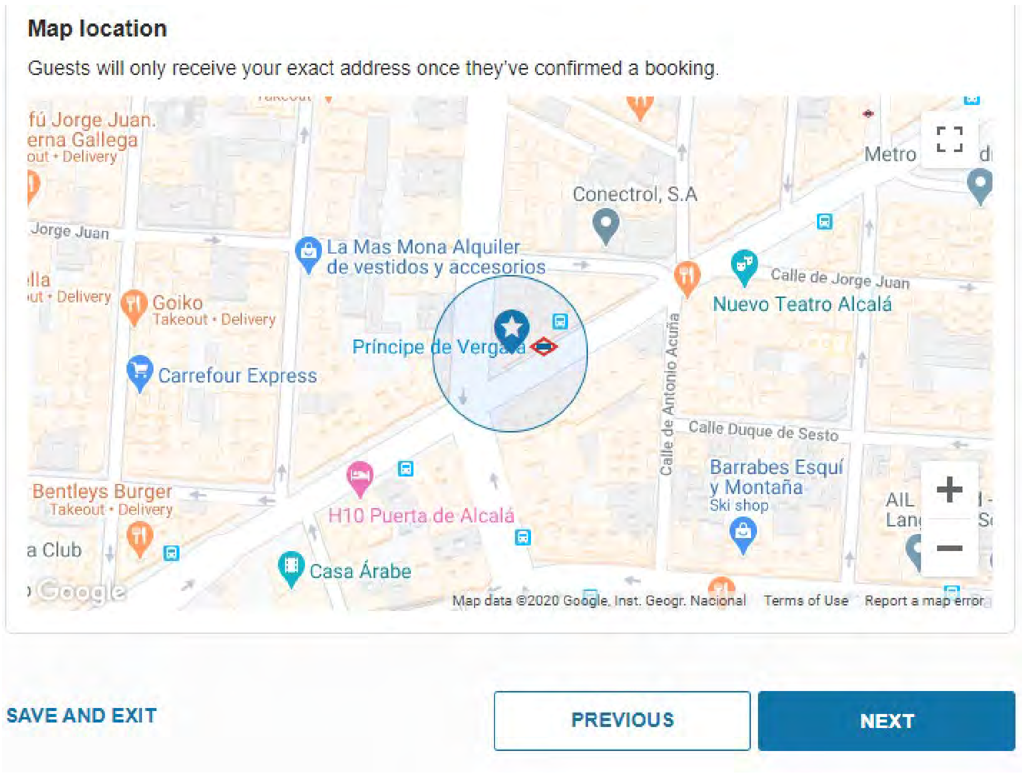


Pictures matter to travelers. Upload as many high-quality images as you have. You can add more later. Want some tips on how to upload quality photos that generate more bookings?Check this out.

* Tips: Min. 800x600 px — Best 2048x1536 px

https://www.priceline.com/static-pages/browser-upgrade.html

75. By way of another example, the YCS Platform was configured to accept user settings for the location of the property on a map by zooming in and out, or dragging and dropping a pin, on an interactive map.


Map location

Guests will only receive your exact address once they've confirmed a booking.

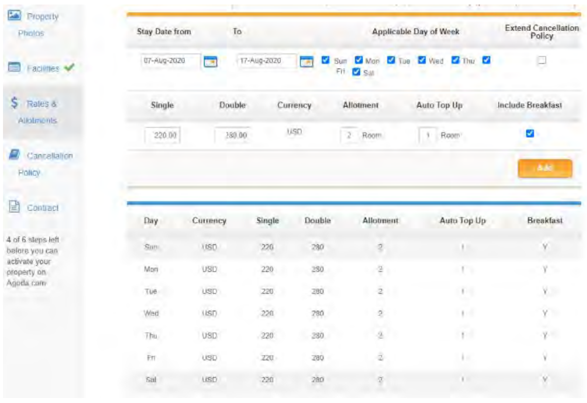
76. As described above, the YCS Platform comprised a browser-based platform, and presented a user selectable panel of settings through a browser of a computer, which rendered the property listing website using a browser engine (a virtual machine), based on commands to the browser engine, for display on the browser.

77. When a desired setting was selected, a display in accordance with that selection was generated substantially contemporaneously with the selection. In the examples above, when a map location was selected on the map showing the property location, the selected location was displayed substantially contemporaneously with the dragging and dropping of the pin. Similarly, when a particular image was selected and uploaded or a caption for an image was selected, the

selections were displayed substantially contemporaneously.

78. Upon selection of the desired setting, information representative of the selected setting was stored in a database that supported the property listing website. On information and belief, websites for the Priceline Agoda products were supported on database systems where settings for the Priceline or Agoda websites were stored.

79. A website for the property listing was generated in part by retrieving the information and files from a database. As shown in the example below, a user could set "single rate" to "220.00," "double rate" to "280.00," "Allotment" to "2," "Auto Top Up" to "1," and check the "Include Breakfast" option in the property's "Rates&Allotments" web page. These user input settings were sent to the YCS Platform's server and stored in its database via an HTTP POST request "16529523" in the format of JSON data. A browser could then load the website by retrieving the stored setting information from a response JSON data. In the example shown below, the settings "220.00" for "Single," "280.00" for "Double," "2" for "Allotment," "1" for "Auto Top Up," and "Y" for "Breakfast" were fetched from the YCS Platform's database and displayed in the table listed for each applicable day of week.

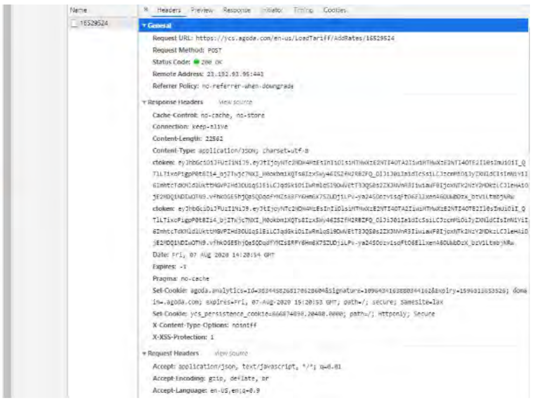


4 of 6 steps left before you can activate your property on Agoda.com

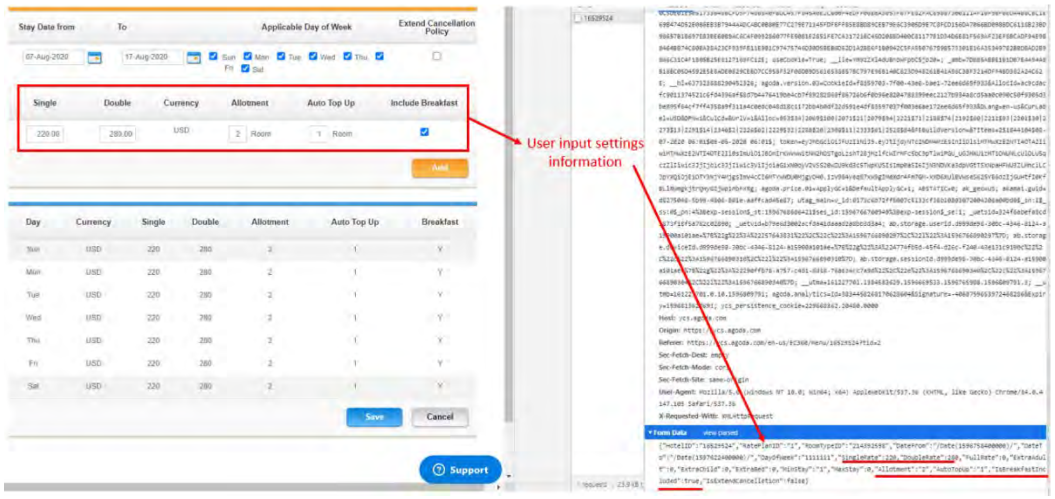
Stay Date from	To	Applicable Day of Week	Extend Cancellation Policy
07-Aug-2020	17-Aug-2020	✓ Sun ✓ Mon ✓ Tue ✓ Wed ✓ Thu ✓ Fri ✓ Sat	☐

Single	Double	Currency	Allotment	Auto Top Up	Include Breakfast
220.00	280.00	USD	2 Room	1 Room	✓

Day	Currency	Single	Double	Allotment	Auto Top Up	Breakfast
Sun	USD	220	280	2	1	Y
Mon	USD	220	280	2	1	Y
Tue	USD	220	280	2	1	Y
Wed	USD	220	280	2	1	Y
Thu	USD	220	280	2	1	Y
Fri	USD	220	280	2	1	Y
Sat	USD	220	280	2	1	Y



```
▾ General
  Request URL: https://ycs.agoda.com/en-us/LoadTariff/AddRates/16529524
  Request Method: POST
  Status Code: ● 200 OK
  Remote Address: 23.192.93.95:443
  Referrer Policy: no-referrer-when-downgrade
▾ Response Headers    view source
  Cache-Control: no-cache, no-store
  Connection: keep-alive
  Content-Length: 22562
  Content-Type: application/JSON; charset=utf-8
  ctoken: eyJhbGciOiJFUzI1NiJ9.eyJtIjoyNTc2NDM4MzEsInIiOlsiMTMwXzE2NTI4OTA2IiwiMTMwXzE2NTI4OTE2Il0sImJiOiI_Q
  TlLTixoPigpP0t8Zi4_bjZTWjc7NXI_M0okbm1XQTs8IzxSWy46ISZfM2RBZFQ_OlJiJ01IeldIcSsiLCJzcmMiOiJyZXNldCIsImNiYiI
  6ImhtcTdKMldlUkttMGVPZHd3OUlqSlEiLCJqdGkiOiIwRmlqSl9DWVEtT3JQS0s2ZXJNVnR3IiwiaWF0IjoxNTk2NzY2MDkzLCJleHAiO
  jE2MDQ1NDIwOTN9.vfhkOGE5hjQaSQDqdfYNZs8RFY6HmEX7SZUDjiLPv-ya24SOozvisqFtO6EllxenA6OUWbDzX_bzV1LtmbjNRw
  ctoken: eyJhbGciOiJFUzI1NiJ9.eyJtIjoyNTc2NDM4MzEsInIiOlsiMTMwXzE2NTI4OTA2IiwiMTMwXzE2NTI4OTE2Il0sImJiOiI_Q
  TlLTixoPigpP0t8Zi4_bjZTWjc7NXI_M0okbm1XQTs8IzxSWy46ISZfM2RBZFQ_OlJiJ01IeldIcSsiLCJzcmMiOiJyZXNldCIsImNiYiI
  6ImhtcTdKMldlUkttMGVPZHd3OUlqSlEiLCJqdGkiOiIwRmlqSl9DWVEtT3JQS0s2ZXJNVnR3IiwiaWF0IjoxNTk2NzY2MDkzLCJleHAiO
  jE2MDQ1NDIwOTN9.vfhkOGE5hjQaSQDqdfYNZs8RFY6HmEX7SZUDjiLPv-ya24SOozvisqFtO6EllxenA6OUWbDzX_bzV1LtmbjNRw
  Date: Fri, 07 Aug 2020 14:20:54 GMT
  Expires: -1
  Pragma: no-cache
  Set-Cookie: agoda.analytics=Id=3834458268178628604&Signature=1096434163880344162&Expiry=1596813653526; doma
  in=.agoda.com; expires=Fri, 07-Aug-2020 15:20:53 GMT; path=/; secure; samesite=lax
  Set-Cookie: ycs_persistence_cookie=666874030.20480.0000; path=/; Httponly; Secure
  X-Content-Type-Options: nosniff
  X-XSS-Protection: 1
▾ Request Headers    view source
  Accept: application/json, text/javascript, */*; q=0.01
  Accept-Encoding: gzip, deflate, br
  Accept-Language: en-US,en;q=0.9
```

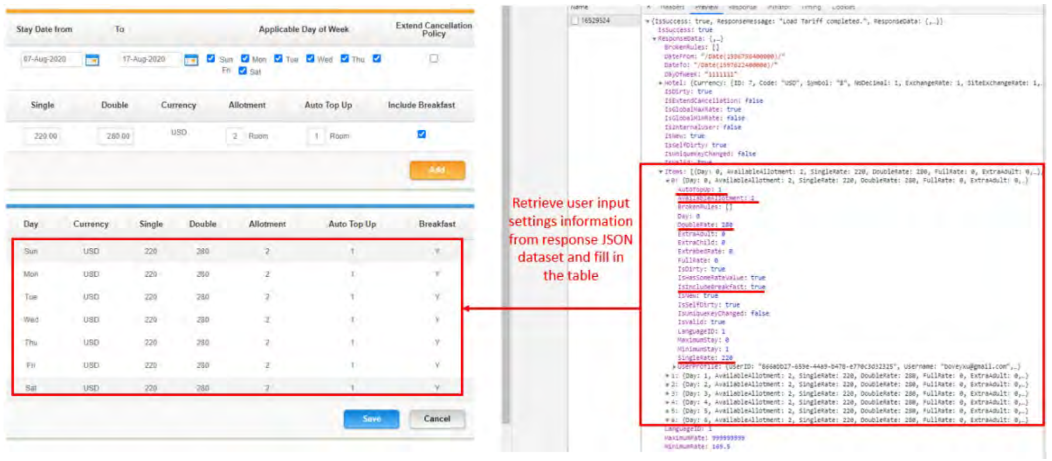


Stay Date from	To	Applicable Day of Week	Extend Cancellation Policy
07-Aug-2020	17-Aug-2020	✓ Sun ✓ Mon ✓ Tue ✓ Wed ✓ Thu ✓ Fri ✓ Sat	☐

Single	Double	Currency	Allotment	Auto Top Up	Include Breakfast
220.00	280.00	USD	2 Room	1 Room	✓

Day	Currency	Single	Double	Allotment	Auto Top Up	Breakfast
Sun	USD	220	280	2	1	Y
Mon	USD	220	280	2	1	Y
Tue	USD	220	280	2	1	Y
Wed	USD	220	280	2	1	Y
Thu	USD	220	280	2	1	Y
Fri	USD	220	280	2	1	Y
Sat	USD	220	280	2	1	Y

```
16529524
[illegible]
Host: ycs.agoda.com
Origin: https://ycs.agoda.com
Referer: https://ycs.agoda.com/en-us/EC368/Menu/16529524?tid=2
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36
X-Requested-With: XMLHttpRequest
▼ Form Data    view parsed
{"HotelID":"16529524","RatePlanID":"1","RoomTypeID":"214392598","DateFrom":"/Date(1596758400000)/","DateTo":"/Date(1597622400000)/","DayOfWeek":"1111111","SingleRate":220,"DoubleRate":280,"FullRate":0,"ExtraAdult":0,"ExtraChild":0,"ExtraBed":0,"MinStay":"1","MaxStay":0,"Allotment":"2","AutoTopUp":"1","IsBreakfastIncluded":true,"IsExtendCancellation":false}
```


Retrieve user input settings information from response JSON dataset and fill in the table
Headers
Preview
Response
Initiator
Timing
Cookies
Add
Save
Cancel

Stay Date from	To	Applicable Day of Week	Extend Cancellation Policy
07-Aug-2020	17-Aug-2020	✓ Sun ✓ Mon ✓ Tue ✓ Wed ✓ Thu ✓ Fri ✓ Sat	☐

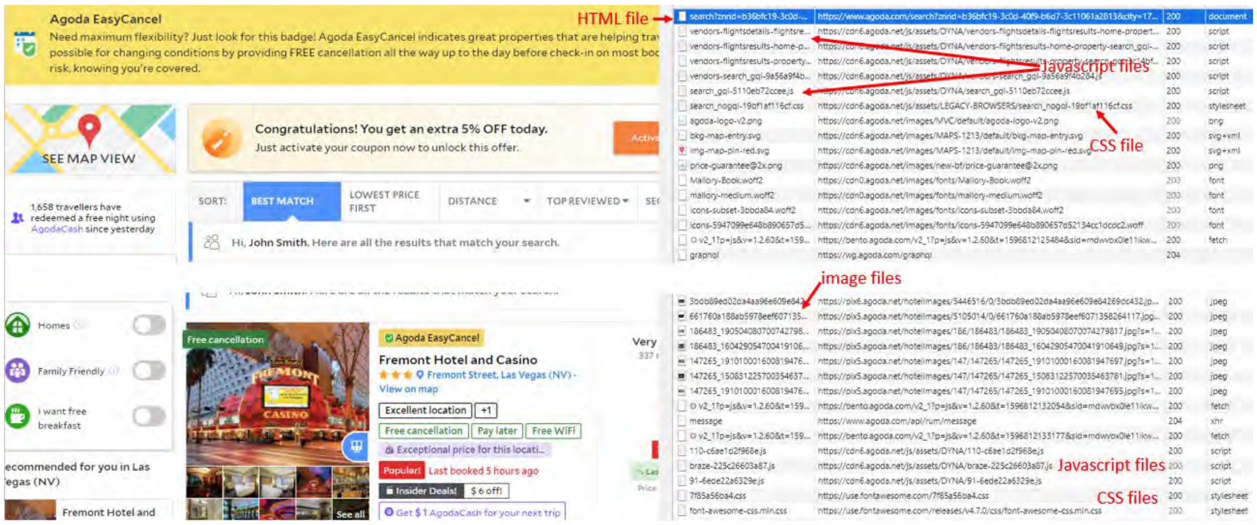
Single	Double	Currency	Allotment	Auto Top Up	Include Breakfast
220.00	280.00	USD	2 Room	1 Room	✓

Day	Currency	Single	Double	Allotment	Auto Top Up	Breakfast
Sun	USD	220	280	2	1	Y
Mon	USD	220	280	2	1	Y
Tue	USD	220	280	2	1	Y
Wed	USD	220	280	2	1	Y
Thu	USD	220	280	2	1	Y
Fri	USD	220	280	2	1	Y
Sat	USD	220	280	2	1	Y

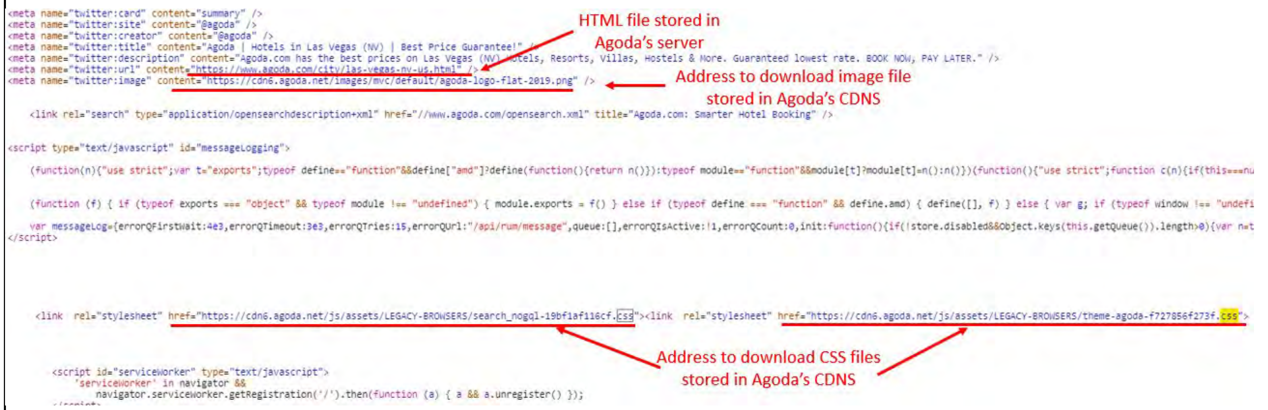
```
16529524
▼{IsSuccess: true, ResponseMessage: "Load Tariff completed.", ResponseData: {,…}}
  IsSuccess: true
 ▼ResponseData: {,…}
   BrokenRules: []
   DateFrom: "/Date(1596758400000)/"
   DateTo: "/Date(1597622400000)/"
   DayOfWeek: "1111111"
  ▸Hotel: {Currency: {ID: 7, Code: "USD", Symbol: "$", NoDecimal: 1, ExchangeRate: 1, SiteExchangeRate: 1,…
   IsDirty: true
   IsExtendCancellation: false
   IsGlobalMaxRate: true
   IsGlobalMinRate: false
   IsInternalUser: false
   IsNew: true
   IsSelfDirty: true
   IsUniqueKeyChanged: false
   IsValid: true
  ▼Items: [{Day: 0, AvailableAllotment: 2, SingleRate: 220, DoubleRate: 280, FullRate: 0, ExtraAdult: 0,…},
   ▼0: {Day: 0, AvailableAllotment: 2, SingleRate: 220, DoubleRate: 280, FullRate: 0, ExtraAdult: 0,…}
      AutoTopUp: 1
      AvailableAllotment: 2
      BrokenRules: []
      Day: 0
      DoubleRate: 280
      ExtraAdult: 0
      ExtraChild: 0
      ExtrabedRate: 0
      FullRate: 0
      IsDirty: true
      IsHasSomeRateValue: true
      IsIncludeBreakfast: true
      IsNew: true
      IsSelfDirty: true
      IsUniqueKeyChanged: false
      IsValid: true
      LanguageID: 1
      MaximumStay: 0
      MinimumStay: 1
      SingleRate: 220
     ▸UserProfile: {UserID: "866a0b27-659e-44a9-b478-e770c3d32325", Username: "boveyxu@gmail.com",…}
   ▸1: {Day: 1, AvailableAllotment: 2, SingleRate: 220, DoubleRate: 280, FullRate: 0, ExtraAdult: 0,…}
   ▸2: {Day: 2, AvailableAllotment: 2, SingleRate: 220, DoubleRate: 280, FullRate: 0, ExtraAdult: 0,…}
   ▸3: {Day: 3, AvailableAllotment: 2, SingleRate: 220, DoubleRate: 280, FullRate: 0, ExtraAdult: 0,…}
   ▸4: {Day: 4, AvailableAllotment: 2, SingleRate: 220, DoubleRate: 280, FullRate: 0, ExtraAdult: 0,…}
   ▸5: {Day: 5, AvailableAllotment: 2, SingleRate: 220, DoubleRate: 280, FullRate: 0, ExtraAdult: 0,…}
   ▸6: {Day: 6, AvailableAllotment: 2, SingleRate: 220, DoubleRate: 280, FullRate: 0, ExtraAdult: 0,…}
   LanguageID: 1
   MaximumRate: 999999999
   MinimumRate: 169.5
```

80. The YCS Platform built the property listing website comprising one or more web pages from the data for the selected settings and files stored in the database. Run time files (including CSS files and Javascript files) used the stored data and files to generate commands for the browser engine to display the one or more web pages. The YCS Platform relied on a browser engine to generate a website comprising one or more web pages based on settings data extracted from at least a portion of the YCS Platform's CDN database, and at least one run time file. As shown in the example below, the YCS Platform downloaded an HTML file from an Agoda server. When the browser build engine parsed the HTML file, the web browser could make HTTP GET requests to fetch the embedded HTML file from Agoda's server, and embedded run time files (e.g., CSS and Javascript files), along with additional content files (e.g., image files), from Agoda's

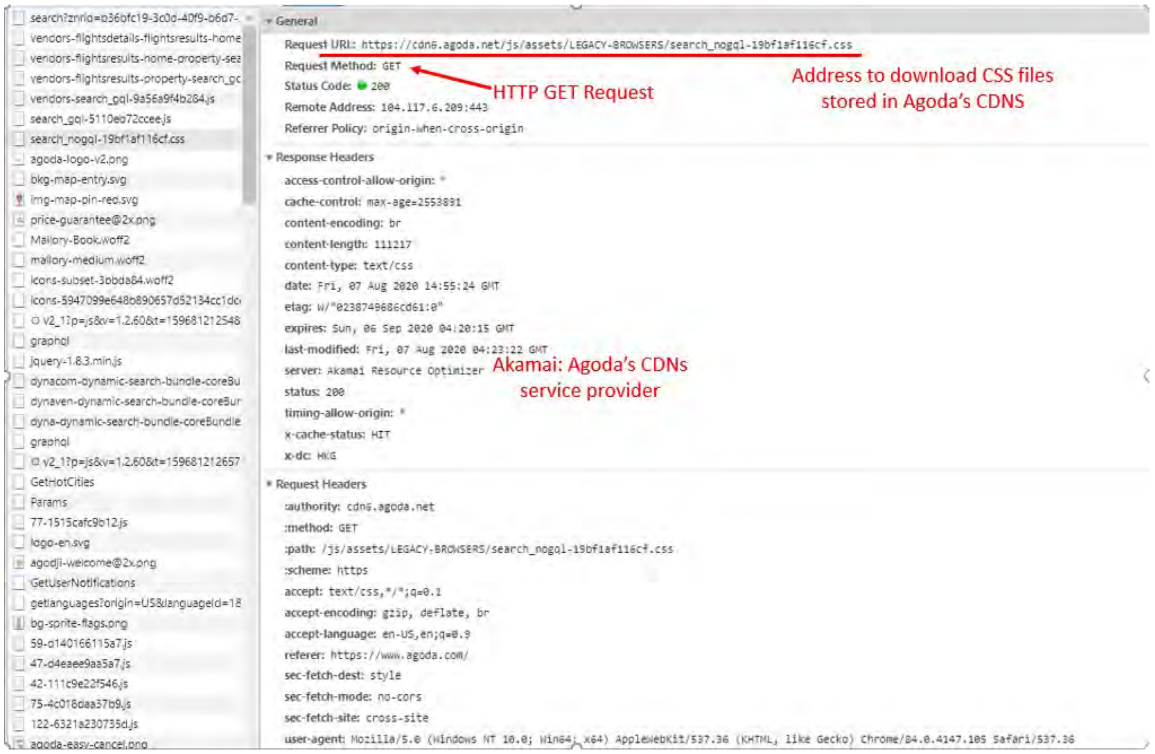
CDNs. The image files displayed in the website were downloaded from the Akamai server, the CDN service provider for Agoda.


Agoda EasyCancel
Need maximum flexibility? Just look for this badge! Agoda EasyCancel indicates great properties that are helping tra
possible for changing conditions by providing FREE cancellation all the way up to the day before check-in on most boo
risk, knowing you're covered.
SEE MAP VIEW
Congratulations! You get an extra 5% OFF today.
Just activate your coupon now to unlock this offer.
Activ
SORT:
BEST MATCH
LOWEST PRICE FIRST
DISTANCE
TOP REVIEWED
1,658 travellers have redeemed a free night using AgodaCash since yesterday
Hi, John Smith. Here are all the results that match your search.
Homes
Family Friendly
I want free breakfast
ecommended for you in Las egas (NV)
Fremont Hotel and
Free cancellation
Agoda EasyCancel
Fremont Hotel and Casino
Fremont Street, Las Vegas (NV) -
View on map
Excellent location
+1
Free cancellation
Pay later
Free WiFi
Exceptional price for this locati...
Popular!
Last booked 5 hours ago
Insider Deals!
$ 6 off!
Get $ 1 AgodaCash for your next trip
See all
Very
HTML file
Javascript files
CSS file
image files
Javascript files
CSS files

```
search?znrid=b36bfc19-3c0d-...    https://www.agoda.com/search?znrid=b36bfc19-3c0d-40f9-b6d7-3c11061a2813&city=17...    200  document
vendors-flightsdetails-flightsre...  https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsdetails-flightsresults-home-propert...  200  script
vendors-flightsresults-home-p...   https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsresults-home-property-search_gql-...  200  script
vendors-flightsresults-property... https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsresults-property-search_gql-3c14bf...  200  script
vendors-search_gql-9a56a9f4b...    https://cdn6.agoda.net/js/assets/DYNA/vendors-search_gql-9a56a9f4b284.js  200  script
search_gql-5110eb72ccee.js         https://cdn6.agoda.net/js/assets/DYNA/search_gql-5110eb72ccee.js  200  script
search_nogql-19bf1af116cf.css      https://cdn6.agoda.net/js/assets/LEGACY-BROWSERS/search_nogql-19bf1af116cf.css  200  stylesheet
agoda-logo-v2.png                  https://cdn6.agoda.net/images/MVC/default/agoda-logo-v2.png  200  png
bkg-map-entry.svg                  https://cdn6.agoda.net/images/MAPS-1213/default/bkg-map-entry.svg  200  svg+xml
img-map-pin-red.svg                https://cdn6.agoda.net/images/MAPS-1213/default/img-map-pin-red.svg  200  svg+xml
price-guarantee@2x.png             https://cdn6.agoda.net/images/new-bf/price-guarantee@2x.png  200  png
Mallory-Book.woff2                 https://cdn0.agoda.net/images/fonts/Mallory-Book.woff2  200  font
mallory-medium.woff2               https://cdn0.agoda.net/images/fonts/mallory-medium.woff2  200  font
icons-subset-3bbda84.woff2         https://cdn6.agoda.net/images/fonts/icons-subset-3bbda84.woff2  200  font
icons-5947099e648b890657d5...      https://cdn6.agoda.net/images/fonts/icons-5947099e648b890657d52134cc1dcdc2.woff  200  font
v2_1?p=js&v=1.2.60&t=159...        https://bento.agoda.com/v2_1?p=js&v=1.2.60&t=1596812125484&sid=mdwvbx0le11ikw...  200  fetch
graphql                            https://wg.agoda.com/graphql  204

3bdb89ed02da4aa96e609e842...       https://pix6.agoda.net/hotelImages/5446516/0/3bdb89ed02da4aa96e609e84269dc432.jp...  200  jpeg
661760a188ab5978eef607135...       https://pix5.agoda.net/hotelImages/5105014/0/661760a188ab5978eef6071358264117.jpg...  200  jpeg
186483_190504080700742798...       https://pix5.agoda.net/hotelImages/186/186483/186483_19050408070074279817.jpg?s=1...  200  jpeg
186483_160429054700419106...       https://pix5.agoda.net/hotelImages/186/186483/186483_16042905470041910649.jpg?s=1...  200  jpeg
147265_191010001600819476...       https://pix5.agoda.net/hotelImages/147/147265/147265_19101000160081947697.jpg?s=1...  200  jpeg
147265_150831225700354637...       https://pix5.agoda.net/hotelImages/147/147265/147265_15083122570035463781.jpg?s=1...  200  jpeg
147265_191010001600819476...       https://pix5.agoda.net/hotelImages/147/147265/147265_19101000160081947695.jpg?s=1...  200  jpeg
v2_1?p=js&v=1.2.60&t=159...        https://bento.agoda.com/v2_1?p=js&v=1.2.60&t=1596812132054&sid=mdwvbx0le11ikw...  200  fetch
message                            https://www.agoda.com/api/rum/message  204  xhr
v2_1?p=js&v=1.2.60&t=159...        https://bento.agoda.com/v2_1?p=js&v=1.2.60&t=1596812133177&sid=mdwvbx0le11ikw...  200  fetch
110-c6ae1d2f968e.js                https://cdn6.agoda.net/js/assets/DYNA/110-c6ae1d2f968e.js  200  script
braze-225c26603a87.js              https://cdn6.agoda.net/js/assets/DYNA/braze-225c26603a87.js  200  script
91-6ede22a6329e.js                 https://cdn6.agoda.net/js/assets/DYNA/91-6ede22a6329e.js  200  script
7f85a56ba4.css                     https://use.fontawesome.com/7f85a56ba4.css  200  stylesheet
font-awesome-css.min.css           https://use.fontawesome.com/releases/v4.7.0/css/font-awesome-css.min.css  200  stylesheet
```


HTML file stored in Agoda's server
Address to download image file stored in Agoda's CDNS
Address to download CSS files stored in Agoda's CDNS

```
<meta name="twitter:card" content="summary" />
<meta name="twitter:site" content="@agoda" />
<meta name="twitter:creator" content="@agoda" />
<meta name="twitter:title" content="Agoda | Hotels in Las Vegas (NV) | Best Price Guarantee!" />
<meta name="twitter:description" content="Agoda.com has the best prices on Las Vegas (NV) Hotels, Resorts, Villas, Hostels & More. Guaranteed lowest rate. BOOK NOW, PAY LATER." />
<meta name="twitter:url" content="https://www.agoda.com/city/las-vegas-nv-us.html" />
<meta name="twitter:image" content="https://cdn6.agoda.net/images/mvc/default/agoda-logo-flat-2019.png" />

    <link rel="search" type="application/opensearchdescription+xml" href="//www.agoda.com/opensearch.xml" title="Agoda.com: Smarter Hotel Booking" />

<script type="text/javascript" id="messageLogging">

    (function(n){"use strict";var t="exports";typeof define=="function"&&define["amd"]?define(function(){return n()}):typeof module=="function"&&module[t]?module[t]=n():n()})(function(){"use strict";function c(n){if(this===nu

    (function (f) { if (typeof exports === "object" && typeof module !== "undefined") { module.exports = f() } else if (typeof define === "function" && define.amd) { define([], f) } else { var g; if (typeof window !== "undefi

    var messageLog={errorQFirstWait:4e3,errorQTimeout:3e3,errorQTries:15,errorQUrl:"/api/rum/message",queue:[],errorQIsActive:!1,errorQCount:0,init:function(){if(!store.disabled&&Object.keys(this.getQueue()).length>0){var n=t
</script>


     <link  rel="stylesheet" href="https://cdn6.agoda.net/js/assets/LEGACY-BROWSERS/search_nogql-19bf1af116cf.css"><link  rel="stylesheet" href="https://cdn6.agoda.net/js/assets/LEGACY-BROWSERS/theme-agoda-f727856f273f.css">


       <script id="serviceWorker" type="text/javascript">
            'serviceWorker' in navigator &&
                navigator.serviceWorker.getRegistration('/').then(function (a) { a && a.unregister() });
```

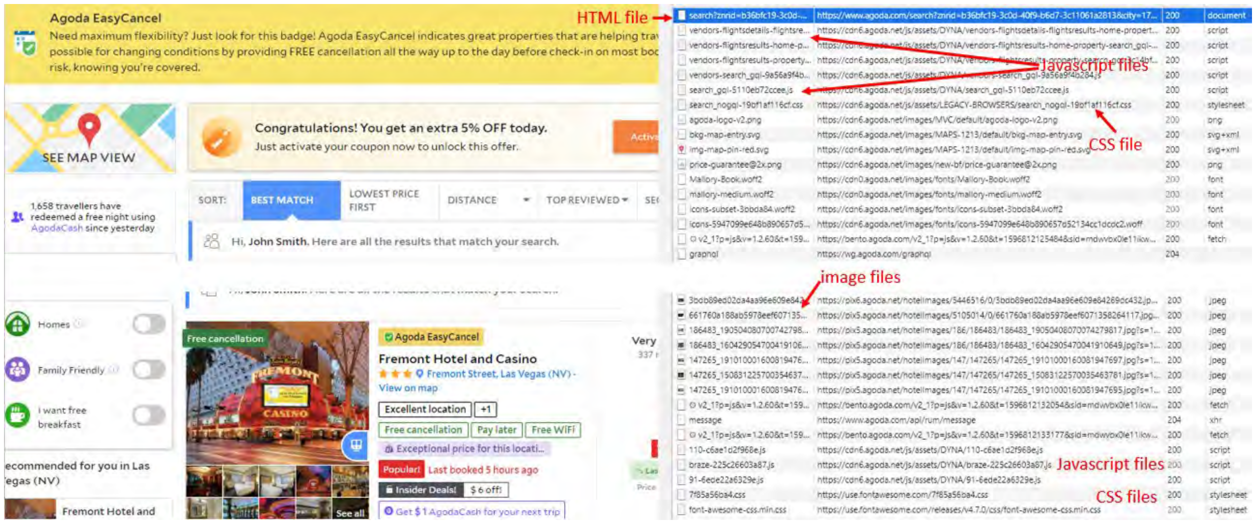


```
search?znrio=b36bfc19-3c0d-40f9-b6d7-
vendors-flightsdetails-flightsresults-home
vendors-flightsresults-home-property-sea
vendors-flightsresults-property-search_gc
vendors-search_gql-9a56a9f4b284.js
search_gql-5110eb72ccee.js
search_nogql-19bf1af116cf.css
agoda-logo-v2.png
bkg-map-entry.svg
img-map-pin-red.svg
price-guarantee@2x.png
Mallory-Book.woff2
mallory-medium.woff2
icons-subset-3bbda84.woff2
icons-5947099e648b890657d52134cc1dc
v2_1?p=js&v=1.2.60&t=159681212548
graphql
jquery-1.8.3.min.js
dynacom-dynamic-search-bundle-coreBu
dynaven-dynamic-search-bundle-coreBur
dyna-dynamic-search-bundle-coreBundle
graphql
v2_1?p=js&v=1.2.60&t=159681212657
GetHotCities
Params
77-1515cafc9b12.js
logo-en.svg
agodji-welcome@2x.png
GetUserNotifications
getlanguages?origin=US&languageId=18
bg-sprite-flags.png
59-d140166115a7.js
47-d4eaee9aa5a7.js
42-111c9e22f546.js
75-4c018daa37b9.js
122-6321a230735d.js
agoda-easy-cancel.png
```

```
▾ General
  Request URL: https://cdn6.agoda.net/js/assets/LEGACY-BROWSERS/search_nogql-19bf1af116cf.css
  Request Method: GET
  Status Code: ● 200
  Remote Address: 104.117.6.209:443
  Referrer Policy: origin-when-cross-origin
▾ Response Headers
  access-control-allow-origin: *
  cache-control: max-age=2553831
  content-encoding: br
  content-length: 111217
  content-type: text/css
  date: Fri, 07 Aug 2020 14:55:24 GMT
  etag: W/"0238749686cd61:0"
  expires: Sun, 06 Sep 2020 04:20:15 GMT
  last-modified: Fri, 07 Aug 2020 04:23:22 GMT
  server: Akamai Resource Optimizer
  status: 200
  timing-allow-origin: *
  x-cache-status: HIT
  x-dc: HKG
▾ Request Headers
  :authority: cdn6.agoda.net
  :method: GET
  :path: /js/assets/LEGACY-BROWSERS/search_nogql-19bf1af116cf.css
  :scheme: https
  accept: text/css,*/*;q=0.1
  accept-encoding: gzip, deflate, br
  accept-language: en-US,en;q=0.9
  referer: https://www.agoda.com/
  sec-fetch-dest: style
  sec-fetch-mode: no-cors
  sec-fetch-site: cross-site
  user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36
```

HTTP GET Request
Address to download CSS files stored in Agoda's CDNS
Akamai: Agoda's CDNs service provider

81. On information and belief, the YCS Platform fetched HTML (*.html) files, CSS (*.css) files, and Javascript (*.js) files from the YCS Platform's servers and converted them with additional contents (e.g., JSON files and image files) into a working website. As shown in the example below, the YCS Platform's source code, such as HTML files and run time files (including CSS files, and Javascript files), and image files, were fetched from the Agoda's servers to build a web page for display.


Agoda EasyCancel
Need maximum flexibility? Just look for this badge! Agoda EasyCancel indicates great properties that are helping trav
possible for changing conditions by providing FREE cancellation all the way up to the day before check-in on most boo
risk, knowing you're covered.

Congratulations! You get an extra 5% OFF today.
Just activate your coupon now to unlock this offer.

1,658 travellers have redeemed a free night using AgodaCash since yesterday

Hi, John Smith. Here are all the results that match your search.

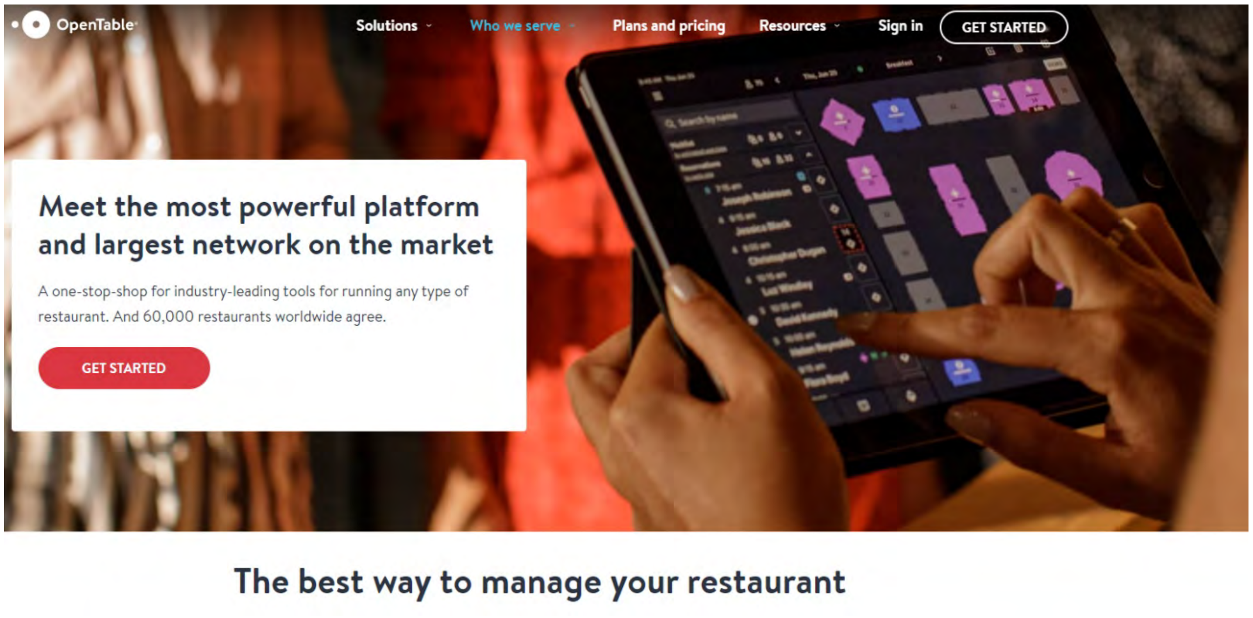
Fremont Hotel and Casino
Fremont Street, Las Vegas (NV) - View on map
Excellent location | +1
Free cancellation | Pay later | Free WiFi
Exceptional price for this locati...
Popular! Last booked 5 hours ago
Insider Deals! $ 6 off!
Get $ 1 AgodaCash for your next trip

ecommended for you in Las
egas (NV)

```
search?znrid=b36bfc19-3c0d-...   https://www.agoda.com/search?znrid=b36bfc19-3c0d-40f9-b6d7-3c11061a2813&city=17...   200   document
vendors-flightsdetails-flightsre...   https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsdetails-flightsresults-home-propert...   200   script
vendors-flightsresults-home-p...   https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsresults-home-property-search_gql-...   200   script
vendors-flightsresults-property...   https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsresults-property-search_gql-3c14bf...   200   script
vendors-search_gql-9a56a9f4b...   https://cdn6.agoda.net/js/assets/DYNA/vendors-search_gql-9a56a9f4b284.js   200   script
search_gql-5110eb72ccee.js   https://cdn6.agoda.net/js/assets/DYNA/search_gql-5110eb72ccee.js   200   script
search_nogql-19bf1af116cf.css   https://cdn6.agoda.net/js/assets/LEGACY-BROWSERS/search_nogql-19bf1af116cf.css   200   stylesheet
agoda-logo-v2.png   https://cdn6.agoda.net/images/MVC/default/agoda-logo-v2.png   200   png
bkg-map-entry.svg   https://cdn6.agoda.net/images/MAPS-1213/default/bkg-map-entry.svg   200   svg+xml
img-map-pin-red.svg   https://cdn6.agoda.net/images/MAPS-1213/default/img-map-pin-red.svg   200   svg+xml
price-guarantee@2x.png   https://cdn6.agoda.net/images/new-bf/price-guarantee@2x.png   200   png
Mallory-Book.woff2   https://cdn0.agoda.net/images/fonts/Mallory-Book.woff2   200   font
mallory-medium.woff2   https://cdn0.agoda.net/images/fonts/mallory-medium.woff2   200   font
icons-subset-3bbda84.woff2   https://cdn6.agoda.net/images/fonts/icons-subset-3bbda84.woff2   200   font
icons-5947099e648b890657d5...   https://cdn6.agoda.net/images/fonts/icons-5947099e648b890657d52134cc1dcdc2.woff   200   font
v2_1?p=js&v=1.2.60&t=159...   https://bento.agoda.com/v2_1?p=js&v=1.2.60&t=1596812125484&sid=mdwvbx0le11ikw...   200   fetch
graphql   https://wg.agoda.com/graphql   204

3bdb89ed02da4aa96e609e842...   https://pix6.agoda.net/hotelImages/5446516/0/3bdb89ed02da4aa96e609e84269dc432.jp...   200   jpeg
661760a188ab5978eef607135...   https://pix5.agoda.net/hotelImages/5105014/0/661760a188ab5978eef6071358264117.jpg...   200   jpeg
186483_190504080700742798...   https://pix5.agoda.net/hotelImages/186/186483/186483_19050408070074279817.jpg?s=1...   200   jpeg
186483_160429054700419106...   https://pix5.agoda.net/hotelImages/186/186483/186483_16042905470041910649.jpg?s=1...   200   jpeg
147265_191010001600819476...   https://pix5.agoda.net/hotelImages/147/147265/147265_19101000160081947697.jpg?s=1...   200   jpeg
147265_150831225700354637...   https://pix5.agoda.net/hotelImages/147/147265/147265_15083122570035463781.jpg?s=1...   200   jpeg
147265_191010001600819476...   https://pix5.agoda.net/hotelImages/147/147265/147265_19101000160081947695.jpg?s=1...   200   jpeg
v2_1?p=js&v=1.2.60&t=159...   https://bento.agoda.com/v2_1?p=js&v=1.2.60&t=1596812132054&sid=mdwvbx0le11ikw...   200   fetch
message   https://www.agoda.com/api/rum/message   204   xhr
v2_1?p=js&v=1.2.60&t=159...   https://bento.agoda.com/v2_1?p=js&v=1.2.60&t=1596812133177&sid=mdwvbx0le11ikw...   200   fetch
110-c6ae1d2f968e.js   https://cdn6.agoda.net/js/assets/DYNA/110-c6ae1d2f968e.js   200   script
braze-225c26603a87.js   https://cdn6.agoda.net/js/assets/DYNA/braze-225c26603a87.js   200   script
91-6ede22a6329e.js   https://cdn6.agoda.net/js/assets/DYNA/91-6ede22a6329e.js   200   script
7f85a56ba4.css   https://use.fontawesome.com/7f85a56ba4.css   200   stylesheet
font-awesome-css.min.css   https://use.fontawesome.com/releases/v4.7.0/css/font-awesome-css.min.css   200   stylesheet
```

<u>The OpenTable Platform:</u>

82. OpenTable has infringed at least claim 1 of the '397 patent through a combination of features in the OpenTable Platform that collectively practiced each limitation of claim 1. By way of example, during the term of the '397 patent, OpenTable provided the OpenTable Platform, a browser-based platform for, *inter alia*, creating restaurant profile websites.

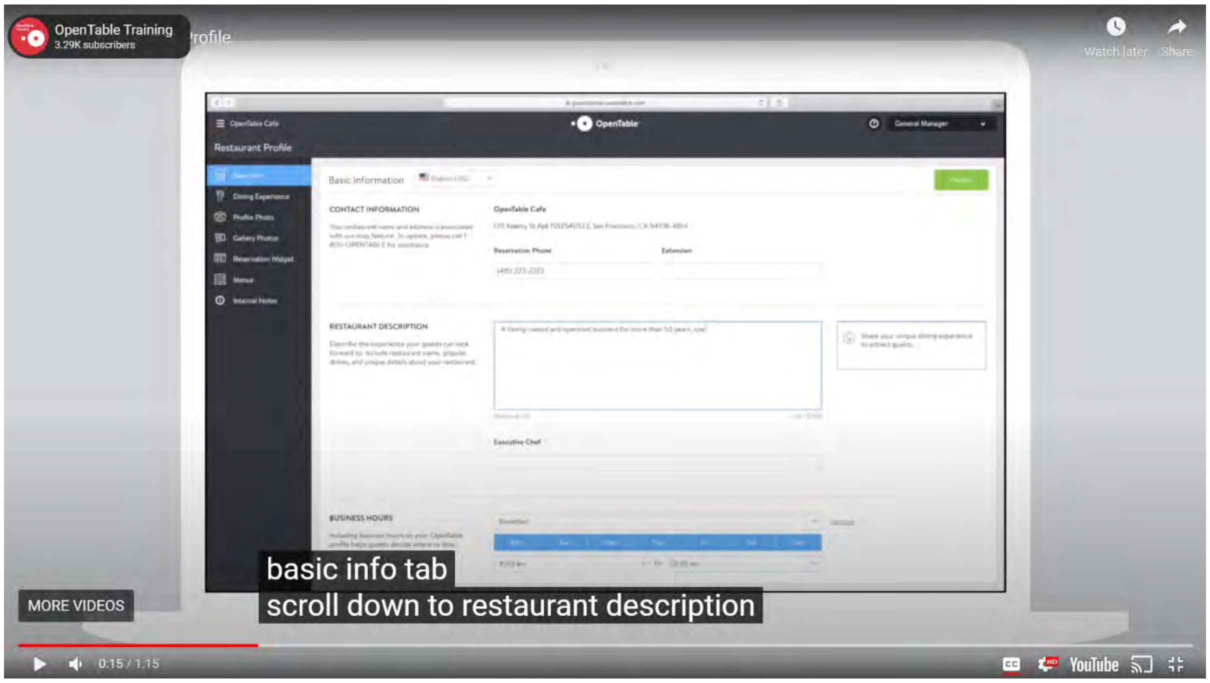

Meet the most powerful platform and largest network on the market

A one-stop-shop for industry-leading tools for running any type of restaurant. And 60,000 restaurants worldwide agree.

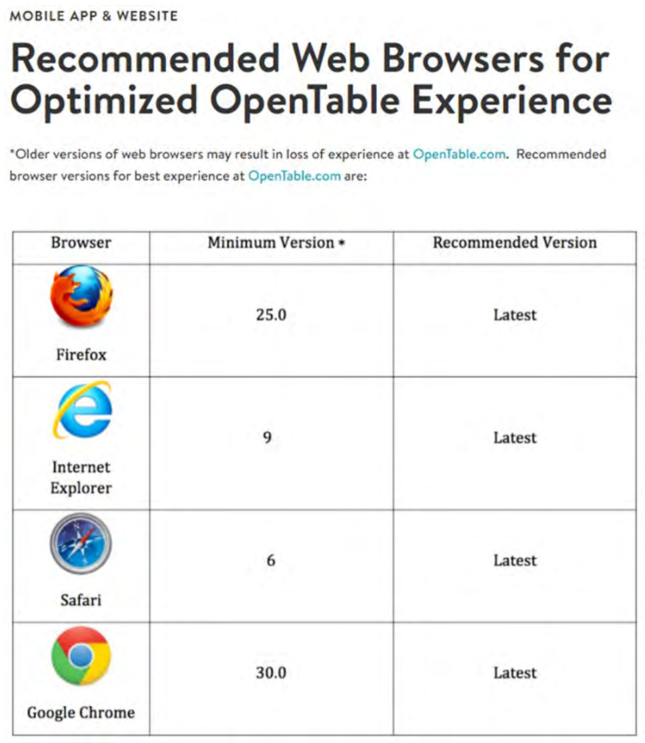
The best way to manage your restaurant

https://restaurant.opentable.com/why-opentable/

83. The OpenTable Platform created restaurant profile websites on and for computers having a browser and a virtual machine capable of generating displays. Upon information and belief, the restaurant profile websites were created and displayed on browsers, such as Google Chrome, Mozilla Firefox, and Microsoft Edge, which used browser engines (virtual machines), to render web pages on computers by interpreting and executing code such as JavaScript and HTML.



https://support.opentable.com/s/article/Updating-your-restaurant-profile-in-GuestCenter?language=en_US (video: https://youtu.be/eQVviz7sYnQ)



MOBILE APP & WEBSITE

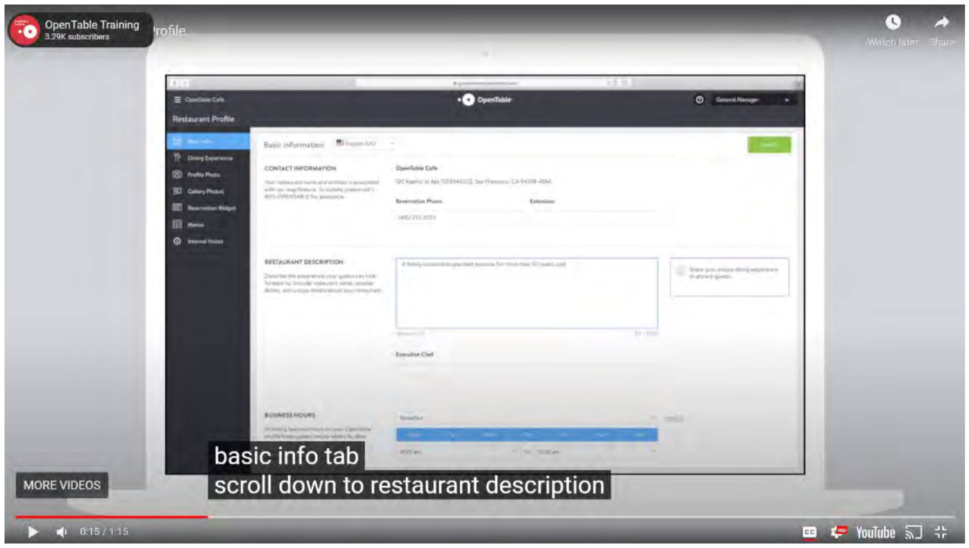
Recommended Web Browsers for Optimized OpenTable Experience

*Older versions of web browsers may result in loss of experience at OpenTable.com. Recommended browser versions for best experience at OpenTable.com are:

Browser	Minimum Version *	Recommended Version
Firefox	25.0	Latest
Internet Explorer	9	Latest
Safari	6	Latest
Google Chrome	30.0	Latest

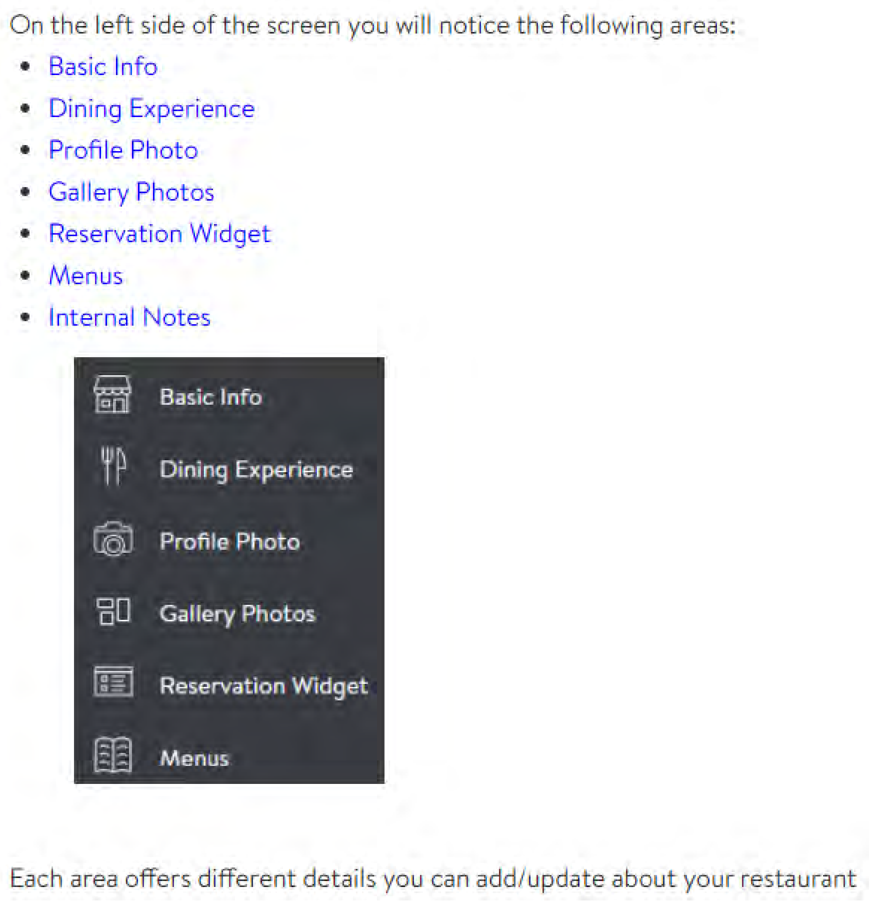
https://help.opentable.com/s/article/Recommended-Web-Browsers-for-Optimized-OpenTable-Experience-1505260708133?language=en_US

84. The OpenTable Platform presented a menu with a user selectable panel of settings that permitted the selection of settings describing elements for a restaurant profile. This permitted the customization of the basic information of the restaurant, dining experience, profile photo of the restaurant, reservation widget and menus.



https://support.opentable.com/s/article/Updating-your-restaurant-profile-in-GuestCenter?language=en_US (video: https://youtu.be/eQVviz7sYnQ)

85. By way of example, a user could customize the images of the restaurant to be displayed on the restaurant profile website through a panel that included options for uploading new images, dragging uploaded images to set the order in which the images were to be displayed, and tagging an image with a tag that would describe the image.


On the left side of the screen you will notice the following areas:

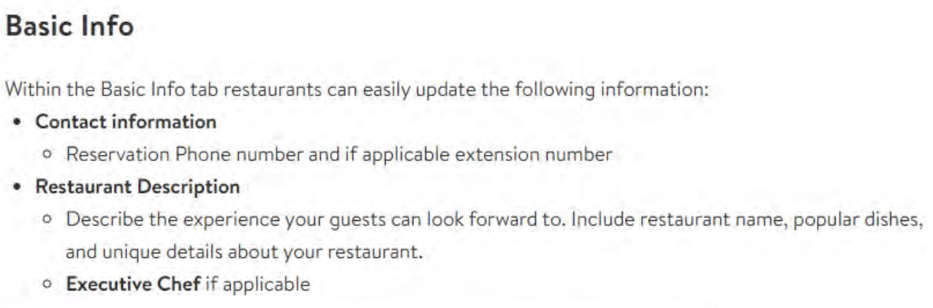
- Basic Info
- Dining Experience
- Profile Photo
- Gallery Photos
- Reservation Widget
- Menus
- Internal Notes

Each area offers different details you can add/update about your restaurant

https://support.opentable.com/s/article/Updating-your-restaurant-profile-in-GuestCenter?language=en_US

86. As described above, the OpenTable Platform comprised a browser-based platform and presented a user selectable panel of settings through a browser of a computer. The computer and browser accepted user inputs for settings such as the basic information of the restaurant, dining experience, profile photo of the restaurant, reservation widget and menus for the restaurant profile.

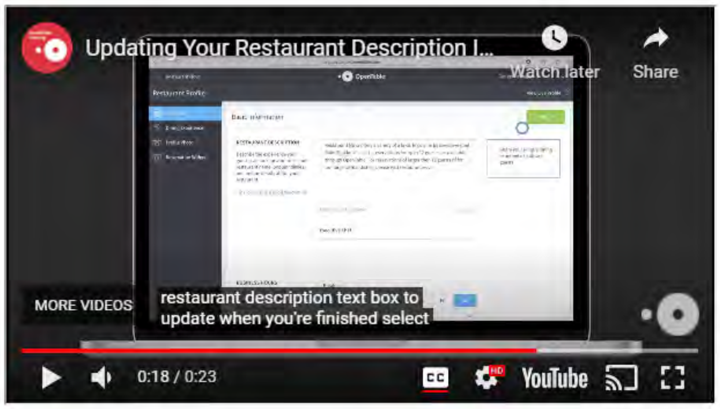
For example, the panel included options for uploading contact information and restaurant description, such as the reservation phone number, restaurant name, popular dishes and unique details.



Basic Info

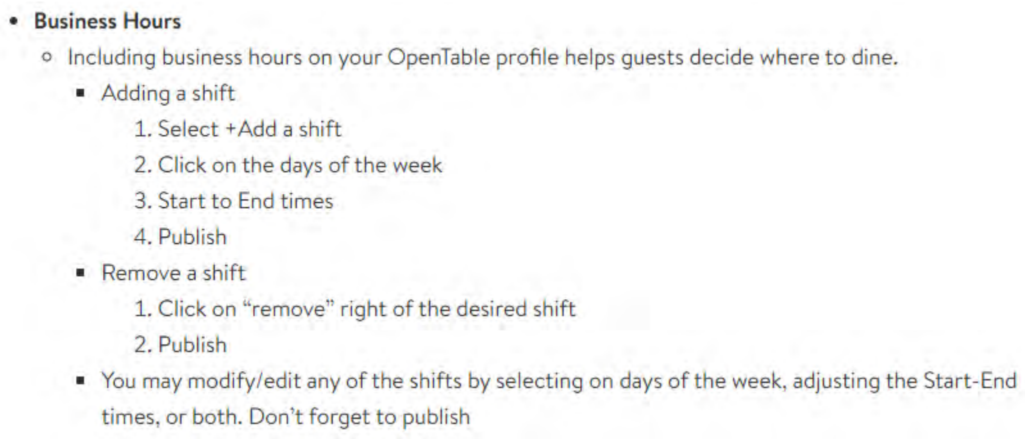
Within the Basic Info tab restaurants can easily update the following information:

- **Contact information**
  - Reservation Phone number and if applicable extension number
- **Restaurant Description**
  - Describe the experience your guests can look forward to. Include restaurant name, popular dishes, and unique details about your restaurant.
  - **Executive Chef** if applicable

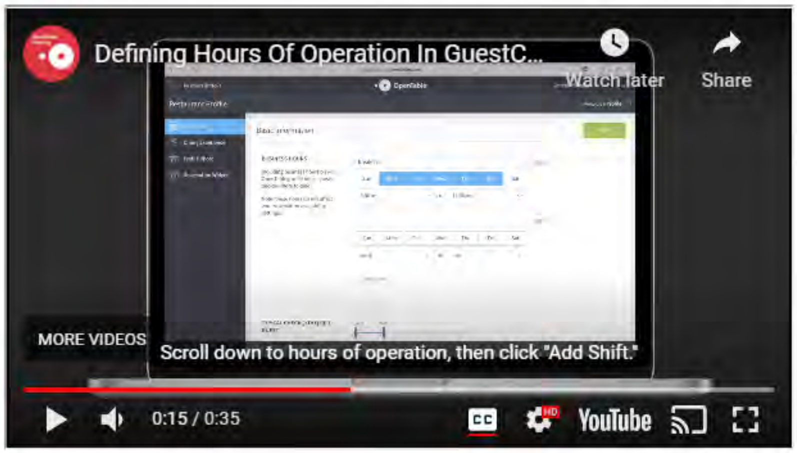

Updating Your Restaurant Description I...
Watch later
Share
MORE VIDEOS
restaurant description text box to
update when you're finished select
0:18 / 0:23
YouTube

https://support.opentable.com/s/article/Updating-your-restaurant-profile-in-GuestCenter?language=en_US (video: https://youtu.be/ta64-nFW8rA)

87. By way of another example, the panel could accept settings to specify a restaurant's hours of operation.



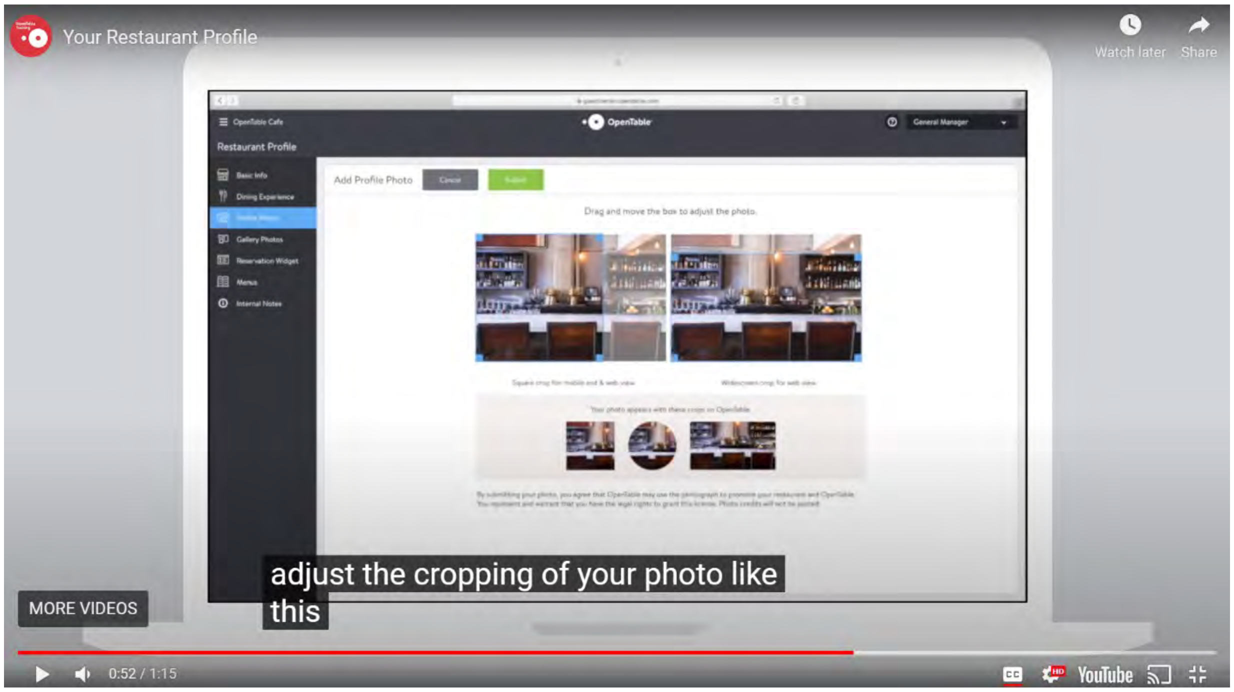
- **Business Hours**
  - Including business hours on your OpenTable profile helps guests decide where to dine.
    - Adding a shift
      1. Select +Add a shift
      2. Click on the days of the week
      3. Start to End times
      4. Publish
    - Remove a shift
      1. Click on "remove" right of the desired shift
      2. Publish
    - You may modify/edit any of the shifts by selecting on days of the week, adjusting the Start-End times, or both. Don't forget to publish


Defining Hours Of Operation In GuestC...
Watch later
Share
MORE VIDEOS
Scroll down to hours of operation, then click "Add Shift."
0:15 / 0:35

Note: these hours do not affect your reservation availability settings. Business hours let diners know when your doors unlocked and locked.

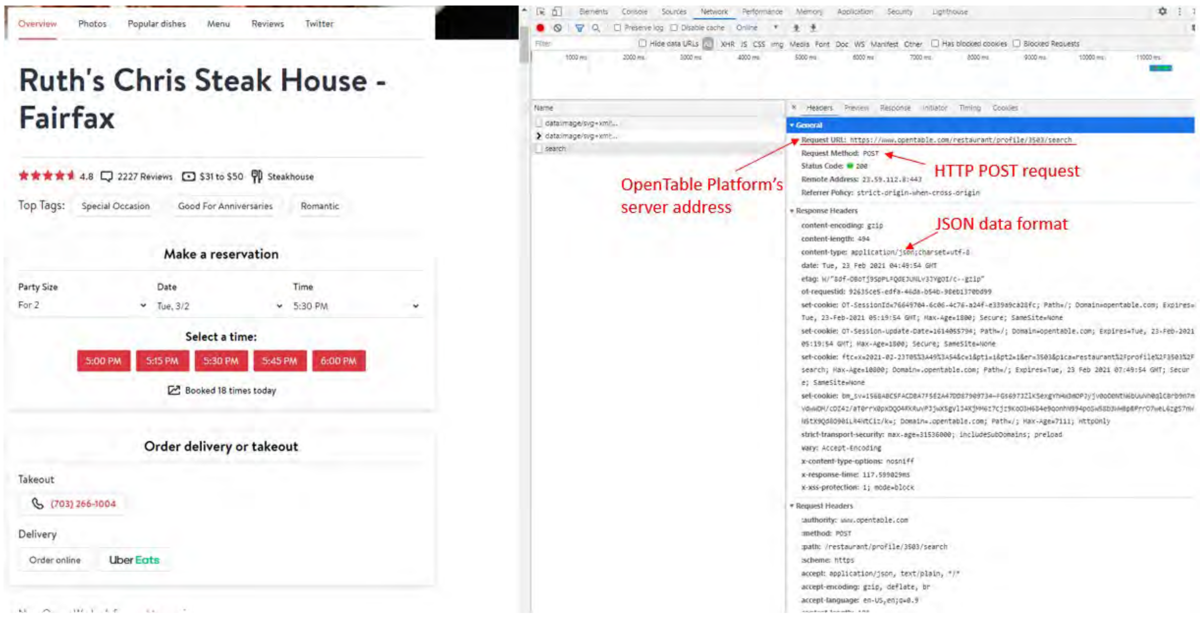
https://support.opentable.com/s/article/Updating-your-restaurant-profile-in-GuestCenter?language=en_US (video: https://youtu.be/VbCAysF9UvI)

88. When a desired setting was selected, the OpenTable Platform generated a display in accordance with that selection substantially contemporaneously with the selection. As shown in the example above, when an image was selected and uploaded, and a user selected to drag and move the box of the image to crop the displayed image, the OpenTable platform displayed these settings substantially contemporaneously with the selections thereof.

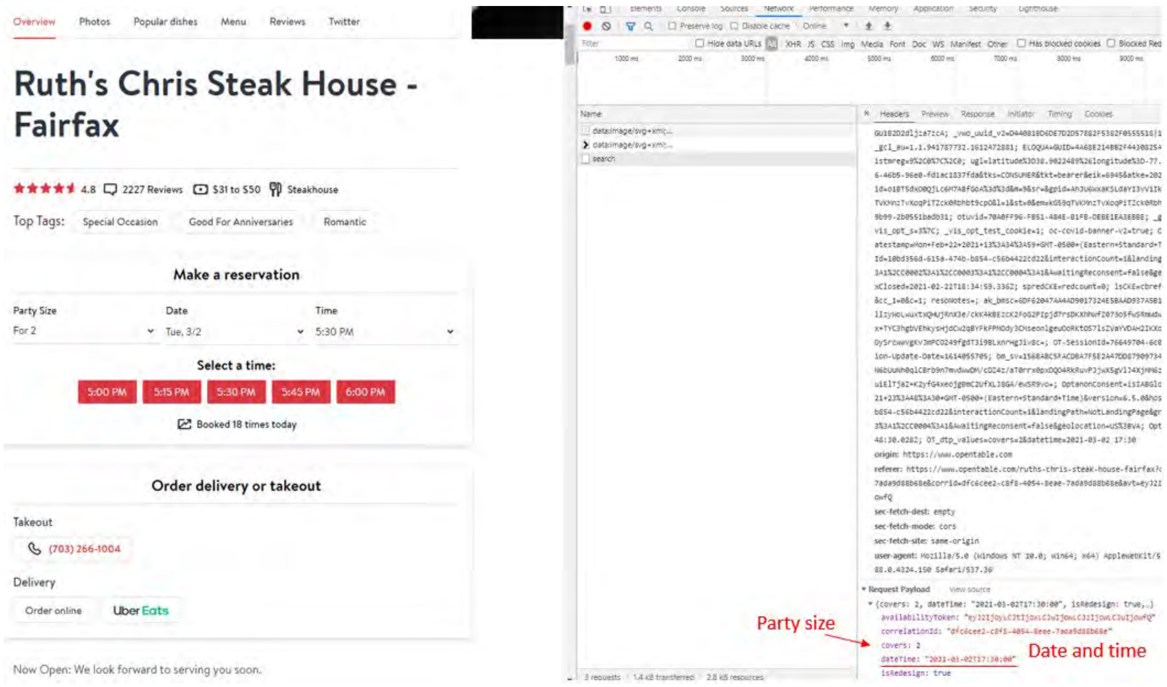


https://support.opentable.com/s/article/Updating-your-restaurant-profile-in-GuestCenter?language=en_US (video: https://youtu.be/eQVviz7sYnQ)

89. Upon information and belief, the OpenTable Platform stored information representative of the selected setting in a database that supporting the restaurant profile. In the example below, the OpenTable Platform sends the user selected settings for a reservation, such as party size, reservation date and time through a "search" HTTP POST request to the OpenTable Platform's server and stores the information representative of the said user selected settings in an OpenTable Platform's database. In particular, the user selected settings are transferred and stored in JSON data format.



```
General
Request URL: https://www.opentable.com/restaurant/profile/3503/search
Request Method: POST
Status Code: 200
Remote Address: 23.59.112.8:443
Referrer Policy: strict-origin-when-cross-origin
Response Headers
content-encoding: gzip
content-length: 494
content-type: application/json;charset=utf-8
date: Tue, 23 Feb 2021 04:49:54 GMT
etag: W/"8df-OBoTj9SpPLFQdEJUNLV3JYgOI/c--gzip"
ot-requestid: 92635ce5-edfa-46da-b54b-98eb1370bd99
set-cookie: OT-SessionId=76649704-6c06-4c76-a24f-e339a9ca28fc; Path=/; Domain=opentable.com; Expires=Tue, 23-Feb-2021 05:19:54 GMT; Max-Age=1800; Secure; SameSite=None
set-cookie: OT-Session-Update-Date=1614055794; Path=/; Domain=opentable.com; Expires=Tue, 23-Feb-2021 05:19:54 GMT; Max-Age=1800; Secure; SameSite=None
set-cookie: ftc=x=2021-02-23T05%3A49%3A54&c=1&pt1=1&pt2=1&er=3503&p1ca=restaurant%2Fprofile%2F3503%2Fsearch; Max-Age=10800; Domain=.opentable.com; Path=/; Expires=Tue, 23 Feb 2021 07:49:54 GMT; Secure; SameSite=None
set-cookie: bm_sv=156BABC5FACDBA7F5E2A47DD87909734~FGs697JZlKSexgYhHW3mDPJyjv0oD0NtN6bUuVh0qlCBrb9n7mVdwWDM/cDZ4z/aT0rrX0pXDQO4RKAuvPJjwX5gVlJ4XjMM6z7cjz9KoO3H6B4e9QonhN994poSw58bJW4Bp8FrrO7weL6zgS7mWN5tX9Qd8O90iLR4HtCiz/k=; Domain=.opentable.com; Path=/; Max-Age=7111; HttpOnly
strict-transport-security: max-age=31536000; includeSubDomains; preload
vary: Accept-Encoding
x-content-type-options: nosniff
x-response-time: 117.599029ms
x-xss-protection: 1; mode=block
Request Headers
:authority: www.opentable.com
:method: POST
:path: /restaurant/profile/3503/search
:scheme: https
accept: application/json, text/plain, */*
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
```



Ruth's Chris Steak House - Fairfax

4.8 2227 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Romantic

Make a reservation

Party Size: For 2
Date: Tue, 3/2
Time: 5:30 PM

Select a time:

5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM

Booked 18 times today

Order delivery or takeout

Takeout

(703) 266-1004

Delivery

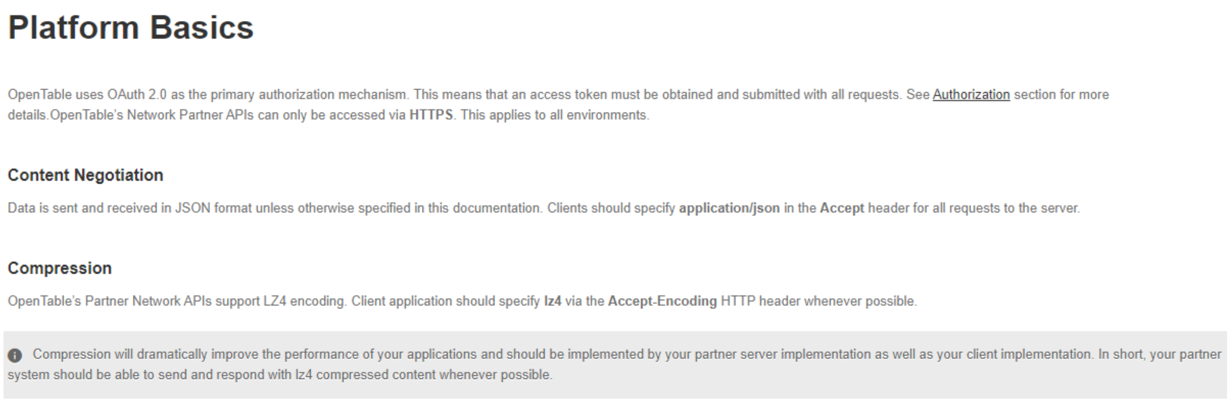
Order online Uber Eats

Now Open: We look forward to serving you soon.

```
data:image/svg+xml;...
data:image/svg+xml;...
search
```

```
GU1B2D2dljz87zcA; _vwo_uuid_v2=D440818D6DE7D2D57882F5382F0555518|1
_gcl_au=1.1.941787732.1612472881; ELOQUA=GUID=4A68E214BB2F44308254
istmreg=9%2C0%7C%2C0; ugl=latitude%3D38.9022489%26longitude%3D-77.
6-46b5-96e0-fd1ac1837fda&tks=CONSUMER&tkt=bearer&eik=6945&atke=202
id=01BT5dXO0QjLC6M7A8fGOA%3d%3d&m=9&sr=&gpid=AhJU6WXaKSLd8YI3VV1Ik
TVKHnzTvKoqPiTZck0Rbhbt9cpO&l=1&st=0&em=kG59qTVKHnzTvKoqPiTZck0Rbh
9b99-2b0551badb31; otuvid=70A0FF96-FB51-484E-B1FB-DEBE1EA3EBBE; _g
vis_opt_s=3%7C; _vis_opt_test_cookie=1; oc-covid-banner-v2=true; C
atestamp=Mon+Feb+22+2021+13%3A34%3A59+GMT-0500+(Eastern+Standard+T
Id=10bd356d-6158-474b-b854-c56b4422cd22&interactionCount=1&landing
3A1%2CC0002%3A1%2CC0003%3A1%2CC0004%3A1&AwaitingReconsent=false&ge
xClosed=2021-02-22T18:34:59.336Z; spredCKE=redcount=0; lsCKE=cbref
&cc_1=0&c=1; resoNotes=; ak_bmsc=6DF62047AA4AD9017324E5BAAD937A5B1
lIzyHoLwuKtXQHUjRnX3e/cKK4kBEzCK2FoG2PIpjd7rsDKXhhwfZ073o5fwSRmadw
x+TYC3hgbVEhkysHjdCw2qBYFkFPNOdy3CHseonlgeuDoRKtO57lsZVaYVDAH2IKXc
OySrcwwVgKvJmPCO249fgdT3i9BLxnrHgJivBc=; OT-SessionId=76649704-6c8
ion-Update-Date=1614055705; bm_sv=1568ABC5FACD8A7F5E2A47DD87909734
N6bUuNh0qlCBrb9n7mvdwwDM/cDZ4z/aT0rrx0pxDQO4RkRuvPJjwX5gVlJ4XjMM6z
uiElTjaZ+K2yfG4xeojgBmC2UfXLJ8GA/ewSR9vo=; OptanonConsent=isIABGlo
21+23%3A48%3A30+GMT-0500+(Eastern+Standard+Time)&version=6.5.0&hos
b854-c56b4422cd22&interactionCount=1&landingPath=NotLandingPage&gr
3%3A1%2CC0004%3A1&AwaitingReconsent=false&geolocation=US%3BVA; Opt
48:30.028Z; OT_dtp_values=covers=2&datetime=2021-03-02 17:30
origin: https://www.opentable.com
referer: https://www.opentable.com/ruths-chris-steak-house-fairfax?
7ada9d88b68e&corrid=dfc6cee2-c8f8-4054-8eae-7ada9d88b68e&avt=eyJ2I
owfQ
sec-fetch-dest: empty
sec-fetch-mode: cors
sec-fetch-site: same-origin
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/5
88.0.4324.150 Safari/537.36
▼ Request Payload    view source
 ▼ {covers: 2, dateTime: "2021-03-02T17:30:00", isRedesign: true,…}
    availabilityToken: "eyJ2IjoyLCJtIjoxLCJwIjowLCJzIjowLCJuIjowfQ"
    correlationId: "dfc6cee2-c8f8-4054-8eee-7ada9d88b68e"
    covers: 2
    dateTime: "2021-03-02T17:30:00"
    isRedesign: true
```

90. The OpenTable Platform generated a restaurant profile website at least in part by retrieving information and files for user selected settings stored in the database. As shown in the example below, the OpenTable Platform could load the restaurant profile on a browser by sending an HTTP GET request to fetch the stored settings information (e.g., overview, photos, popular dishes, reservation, menu, and reviews of the restaurant) from the database in the OpenTable Platform's server. In particular, the OpenTable Platform stored the images of the restaurant (e.g., restaurant profile image) along with other user selected settings, which it in turn communicated from the server as a response to an HTTP GET request. On information and belief, these functionalities were present during the term of the '397 patent.



Platform Basics

OpenTable uses OAuth 2.0 as the primary authorization mechanism. This means that an access token must be obtained and submitted with all requests. See Authorization section for more details.OpenTable's Network Partner APIs can only be accessed via **HTTPS**. This applies to all environments.

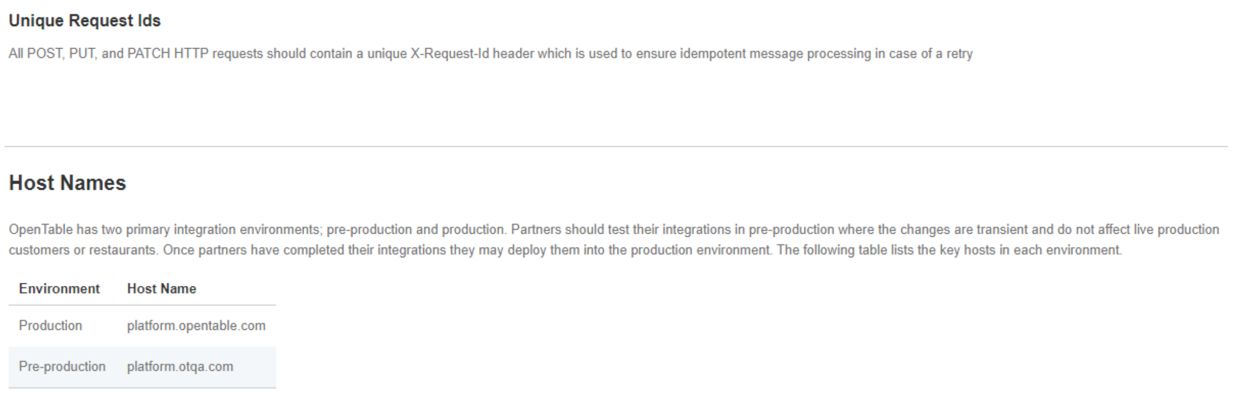
Content Negotiation

Data is sent and received in JSON format unless otherwise specified in this documentation. Clients should specify **application/json** in the **Accept** header for all requests to the server.

Compression

OpenTable's Partner Network APIs support LZ4 encoding. Client application should specify **lz4** via the **Accept-Encoding** HTTP header whenever possible.

Compression will dramatically improve the performance of your applications and should be implemented by your partner server implementation as well as your client implementation. In short, your partner system should be able to send and respond with lz4 compressed content whenever possible.



Unique Request Ids

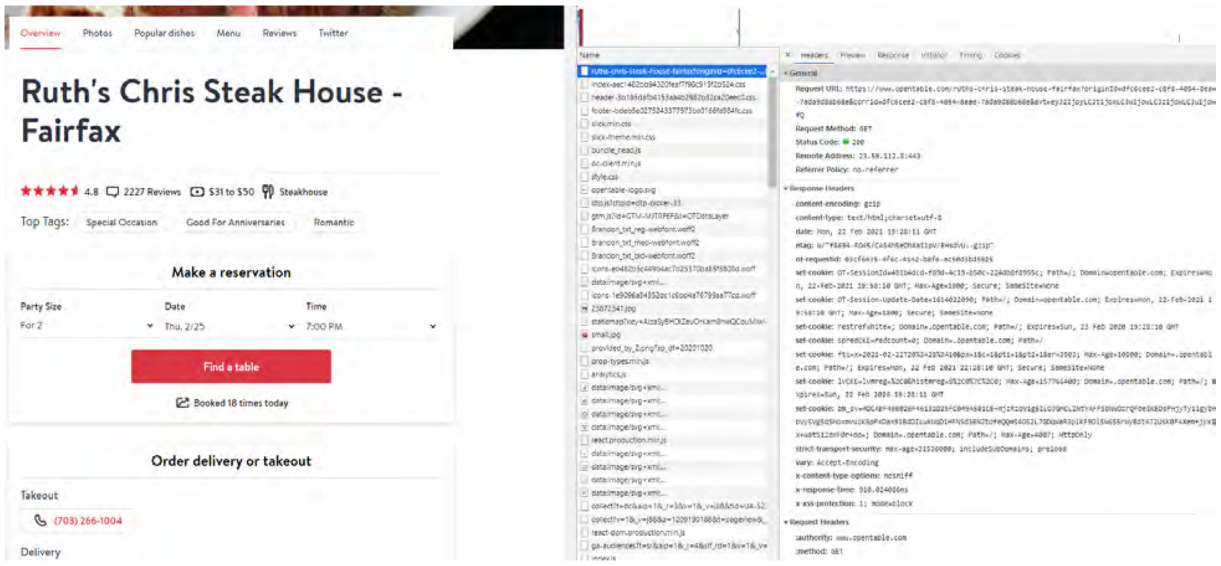
All POST, PUT, and PATCH HTTP requests should contain a unique X-Request-Id header which is used to ensure idempotent message processing in case of a retry

Host Names

OpenTable has two primary integration environments; pre-production and production. Partners should test their integrations in pre-production where the changes are transient and do not affect live production customers or restaurants. Once partners have completed their integrations they may deploy them into the production environment. The following table lists the key hosts in each environment.

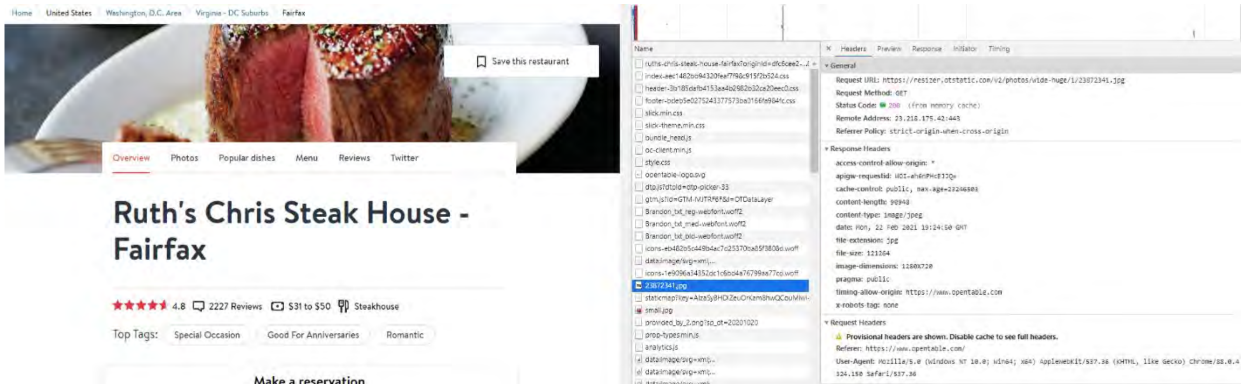
Environment	Host Name
Production	platform.opentable.com
Pre-production	platform.otqa.com

https://platform.opentable.com/documentation/?shell#platform-basics



```
ruths-chris-steak-house-fairfax?originId=dfc6cee2-...
index-aec1462bb94320feaf7f90c915f2b524.css
header-3b185dafb4153ab4b2982b32ca20eec0.css
footer-bdeb5e3275243377573ba0166fa984fc.css
slick.min.css
slick-theme.min.css
bundle_head.js
oc-client.min.js
style.css
opentable-logo.svg
dtp.js?dtpid=dtp-picker-33
gtm.js?id=GTM-MJTRF6F&l=OTDataLayer
Brandon_txt_reg-webfont.woff2
Brandon_txt_med-webfont.woff2
Brandon_txt_bld-webfont.woff2
icons-eb482b5c449b4ac7c25370ba85f3808d.woff
data:image/svg+xml...
icons-1e9096a34352ac1c6bd4a76799aa77cd.woff
23872341.jpg
staticmap?key=AIzaSyBHCXZeuOrKam8nwQCouMiwi-
small.jpg
provided_by_2.png?so_dt=20201020
prop-types.min.js
analytics.js
data:image/svg+xml...
data:image/svg+xml...
data:image/svg+xml...
data:image/svg+xml...
react.production.min.js
data:image/svg+xml...
data:image/svg+xml...
data:image/svg+xml...
data:image/svg+xml...
collect?t=dc&aip=1&_r=3&v=1&_v=j88&tid=UA-52
collect?v=1&_v=j88&a=1209190188&t=pageview&_
react-dom.production.min.js
ga-audiences?t=sr&aip=1&_r=4&slf_rd=1&v=1&_v=
index.js
```

```
▾ General
  Request URL: https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8ea
  -7ada9d88b68e&corrid=dfc6cee2-c8f8-4054-8e8e-7ada9d88b68e&avt=eyJ2IjoyLCJtIjoxLCJwIjowLCJzIjowLCJuIjow
  fQ
  Request Method: GET
  Status Code: 200
  Remote Address: 23.59.112.8:443
  Referrer Policy: no-referrer
▾ Response Headers
  content-encoding: gzip
  content-type: text/html;charset=utf-8
  date: Mon, 22 Feb 2021 19:28:11 GMT
  etag: W/"f8694-RO45/CAS4h6eOhKatIpV/EHsdVU.-gzip"
  ot-requestid: e3cf6a35-4f6c-4142-b8f6-ac50d3bd5925
  set-cookie: OT-SessionId=401b4dcd-f09d-4c19-b50c-224d00f0955c; Path=/; Domain=opentable.com; Expires=Mo
  n, 22-Feb-2021 19:58:10 GMT; Max-Age=1800; Secure; SameSite=None
  set-cookie: OT-Session-Update-Date=1614022090; Path=/; Domain=.opentable.com; Expires=Mon, 22-Feb-2021 1
  9:58:10 GMT; Max-Age=1800; Secure; SameSite=None
  set-cookie: restrefWhite=; Domain=.opentable.com; Path=/; Expires=Sun, 23 Feb 2020 19:28:10 GMT
  set-cookie: spredCKI=redcount=0; Domain=.opentable.com; Path=/
  set-cookie: ftc=x=2021-02-22T20%3A28%3A10&px=1&c=1&pt1=1&pt2=1&er=2502; Max-Age=10800; Domain=.opentabl
  e.com; Path=/; Expires=Mon, 22 Feb 2021 22:28:10 GMT; Secure; SameSite=None
  set-cookie: lvCKI=lvmreg=%2C0&histmreg=0%2C0%7C%2C0; Max-Age=157766400; Domain=.opentable.com; Path=/; E
  xpires=Sun, 22 Feb 2026 19:28:11 GMT
  set-cookie: bm_sv=[illegible]
  [illegible]
  X+u8tS12dnF0r+dd=; Domain=.opentable.com; Path=/; Max-Age=4007; HttpOnly
  strict-transport-security: max-age=31536000; includeSubDomains; preload
  vary: Accept-Encoding
  x-content-type-options: nosniff
  x-response-time: 910.024000ms
  x-xss-protection: 1; mode=block
▾ Request Headers
  :authority: www.opentable.com
  :method: GET
```

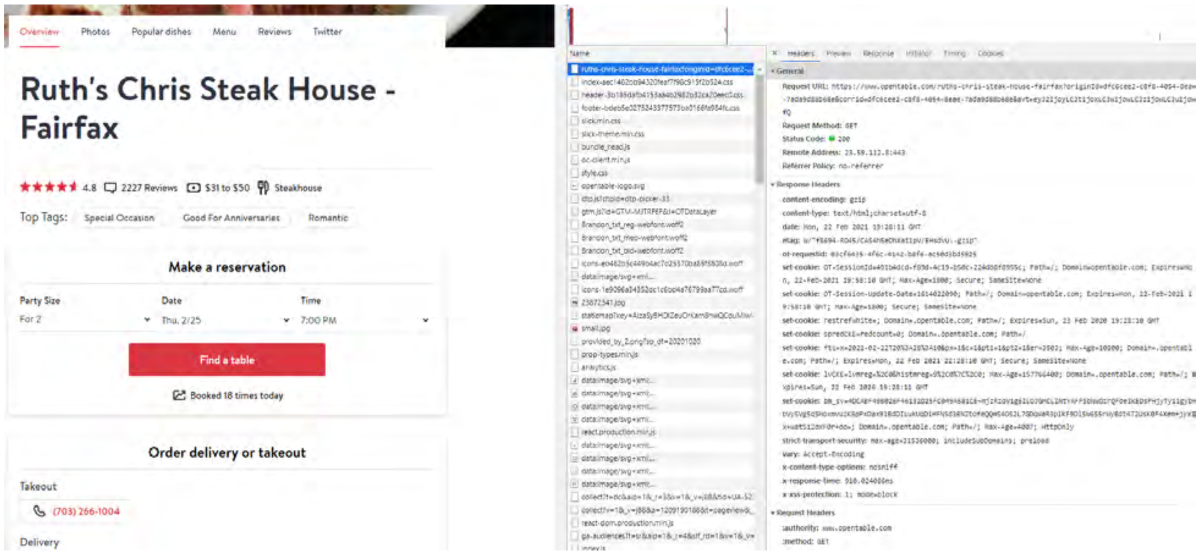


```
ruths-chris-steak-house-fairfax?originId=dfc6cee2-...
index-aec1462bb94320feaf7f90c915f2b524.css
header-3b185dafb4153ab4b2982b32ca20eec0.css
footer-bdeb5e3275243377573ba0166fa984fc.css
slick.min.css
slick-theme.min.css
bundle_head.js
oc-client.min.js
style.css
opentable-logo.svg
dtp.js?dtpid=dtp-picker-33
gtm.js?id=GTM-MJTRF6F&l=OTDataLayer
Brandon_txt_reg-webfont.woff2
Brandon_txt_med-webfont.woff2
Brandon_txt_bld-webfont.woff2
icons-eb482b5c449b4ac7c25370ba85f3808d.woff
data:image/svg+xml...
icons-1e9096a34352ac1c6bd4a76799aa77cd.woff
23872341.jpg
staticmap?key=AIzaSyBHCXZeuOrKam8nwQCouMiwi-
small.jpg
provided_by_2.png?so_dt=20201020
prop-types.min.js
analytics.js
data:image/svg+xml...
data:image/svg+xml...
```

```
▾ General
  Request URL: https://resizer.otstatic.com/v2/photos/wide-huge/1/23872341.jpg
  Request Method: GET
  Status Code: 200  (from memory cache)
  Remote Address: 23.218.175.42:443
  Referrer Policy: strict-origin-when-cross-origin
▾ Response Headers
  access-control-allow-origin: *
  apigw-requestid: WOI-ah6nPHcEJJQ=
  cache-control: public, max-age=23246503
  content-length: 90948
  content-type: image/jpeg
  date: Mon, 22 Feb 2021 19:24:50 GMT
  file-extension: jpg
  file-size: 121264
  image-dimensions: 1280X720
  pragma: public
  timing-allow-origin: https://www.opentable.com
  x-robots-tag: none
▾ Request Headers
  ⚠ Provisional headers are shown. Disable cache to see full headers.
  Referer: https://www.opentable.com/
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4
  324.150 Safari/537.36
```

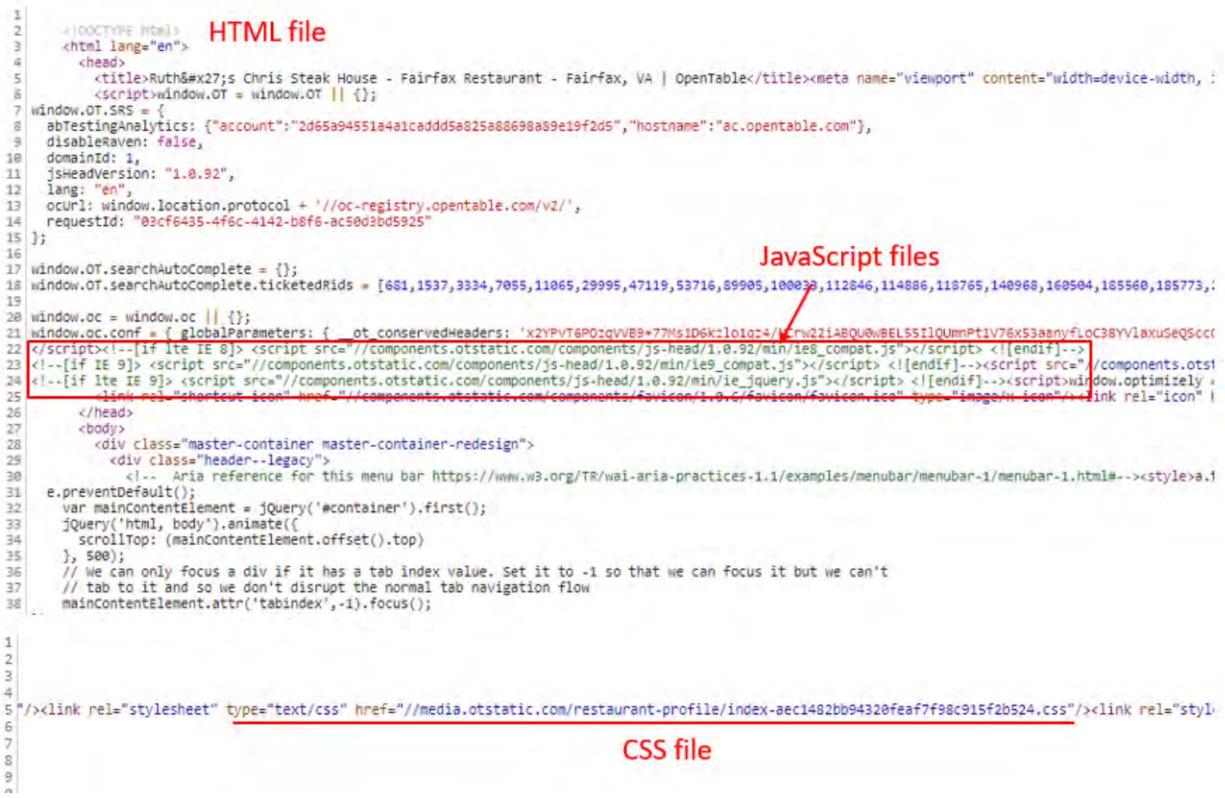
91. The OpenTable Platform built the property listing website comprising one or more web pages from the data for the selected settings and files stored in the database, and one or more run time files. The run time files (including, e.g., CSS files and Javascript files) used the stored data and files to generate commands for the browser engine to display the one or more web pages. In particular, the OpenTable Platform relied on a browser engine to generate a website comprising one or more web pages based on settings data extracted from at least a portion of the OpenTable Platform's database and the run time files.

92. In the example below, the OpenTable Platform downloaded an HTML file from the OpenTable Platform's servers with an HTTP GET request. When the OpenTable Platform parsed the HTML file through a browser build engine, the OpenTable Platform would make a request (e.g., through the GET method) to fetch the embedded CSS and Javascript run time files through the web browser. On information and belief, these functionalities were present during the term of the '397 patent.



```
ruths-chris-steak-house-fairfax?originId=dfc6cee2-...
index-aec1462bb94320feaf7f90c915f2b524.css
header-3b185d9fb4153ab4b2982b32ca20eec2.css
footer-bdeb5e027524337757 3ba0166fa984fc.css
slick.min.css
slick-theme.min.css
bundle_head.js
oc-client.min.js
style.css
opentable-logo.svg
dtp.js?chpid=dtp-picker-33
gtm.js?id=GTM-MJTRPEF&l=OTDataLayer
Brandon_txt_reg-webfont.woff2
Brandon_txt_med-webfont.woff2
Brandon_txt_bld-webfont.woff2
icons-e0482b5c449b4ac7c25370ba89f5820d.woff
data:image/svg+xml...
icons-1e9096a34352cc1c6bd4a76799aa77cd.woff
23872341.jpg
staticmap?key=AIzaSyBHOXZeuOrKam8nwQCouMiw-
small.jpg
provided_by_2.png?so_dt=20201020
prop-types.min.js
analytics.js
data:image/svg+xml...
data:image/svg+xml...
data:image/svg+xml...
data:image/svg+xml...
react.production.min.js
data:image/svg+xml...
data:image/svg+xml...
data:image/svg+xml...
data:image/svg+xml...
collect?t=dc&aip=1&_r=3&v=1&_v=j88&tid=UA-52
collect?v=1&_v=j88&a=1209190188&t=pageview&_
react-dom.production.min.js
ga-audiences?t=sr&aip=1&_r=4&slf_rd=1&v=1&_v=
index.js
```

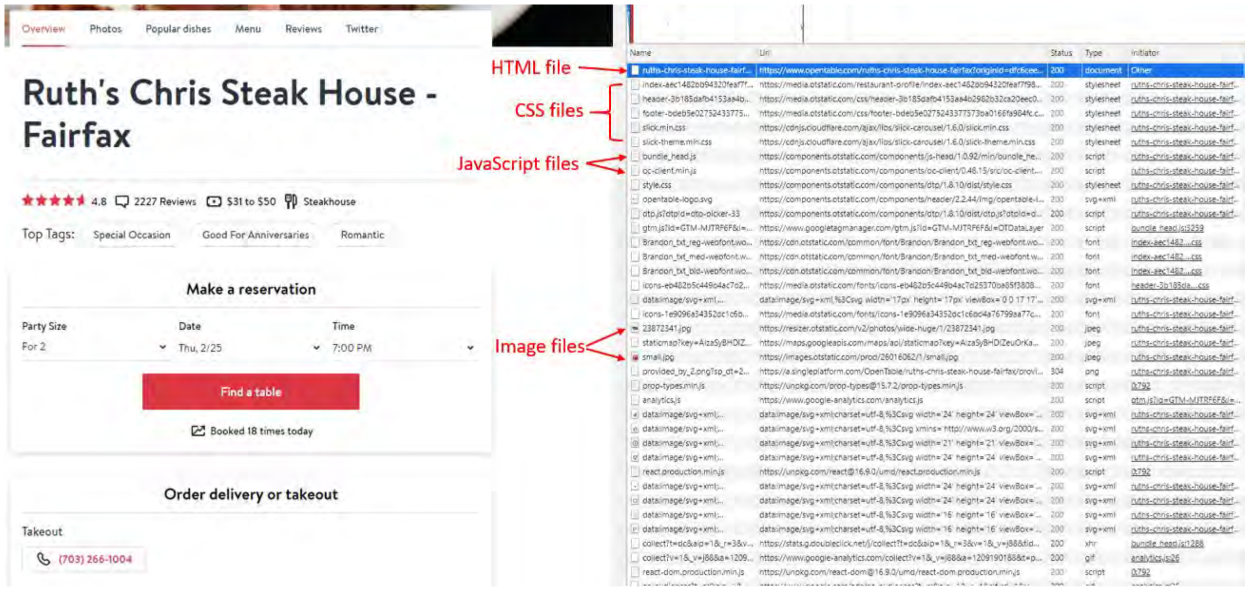
```
▾ General
  Request URL: https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8ee
  -7ada9d88b68e&corrid=dfc6cee2-c8f8-4054-8ee-7ada9d88b68e&avt=eyJ2IjoyLC3tIjoxLCJwIjowLCJzIjowLCJuIjow
  fQ
  Request Method: GET
  Status Code: ● 200
  Remote Address: 23.59.112.8:443
  Referrer Policy: no-referrer
▾ Response Headers
  content-encoding: gzip
  content-type: text/html;charset=utf-8
  date: Mon, 22 Feb 2021 19:28:11 GMT
  etag: W/"f5694-RO45/CA54h5eOhXatIpV/EHsdVU.-gzip"
  ot-requestid: 03cf6635-4f6c-4142-b8f6-ac5e03bd5825
  set-cookie: OT-SessionId=401b4dcd-f09d-4c19-b50c-224d90f0955c; Path=/; Domain=opentable.com; Expires=Mo
  n, 22-Feb-2021 19:58:10 GMT; Max-Age=1800; Secure; SameSite=None
  set-cookie: OT-Session-Update-Date=1614022090; Path=/; Domain=opentable.com; Expires=Mon, 22-Feb-2021 1
  9:58:10 GMT; Max-Age=1800; Secure; SameSite=None
  set-cookie: restrefwhite=; Domain=.opentable.com; Path=/; Expires=Sun, 23 Feb 2020 19:28:10 GMT
  set-cookie: spredCKI=redcount=0; Domain=.opentable.com; Path=/
  set-cookie: ftc=x=2021-02-22T20%3A28%3A10&px=1&c=1&pt1=1&pt2=1&er=2502; Max-Age=10800; Domain=.opentabl
  e.com; Path=/; Expires=Mon, 22 Feb 2021 22:28:10 GMT; Secure; SameSite=None
  set-cookie: lvCKI=lvmreg=%2C0&histmreg=0%2C0%7C%2C0; Max-Age=157766400; Domain=.opentable.com; Path=/; E
  xpires=Sun, 22 Feb 2026 19:28:11 GMT
  set-cookie: bm_sv=4DCAEF498026F46132D25FC0494681C6~mjZ4ZOV1g8ILOJ9MCLINTYAFFSbNwDIrQFOeIXEDsFHjyTy1igyb=
  bVySVg5qSHoxmvU2K8oPxDax91RdDIuuKUdDi#FNS61ENZto#eQQm64O52L7GDqwaR3plkF9OlSW6SrWy8OT472U6K0F4Xmm+jyx
  X+uatS12mF0r+do=; Domain=.opentable.com; Path=/; Max-Age=4007; HttpOnly
  strict-transport-security: max-age=31536000; includeSubDomains; preload
  vary: Accept-Encoding
  x-content-type-options: nosniff
  x-response-time: 918.024086ms
  x-xss-protection: 1; mode=block
▾ Request Headers
  :authority: www.opentable.com
  :method: GET
```



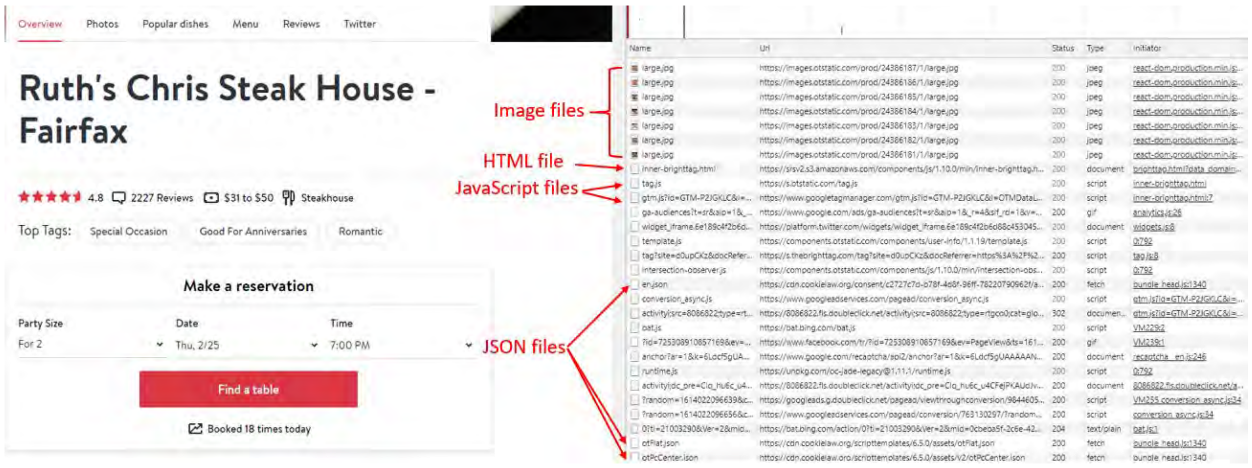
```
    <!DOCTYPE html>
    <html lang="en">
      <head>
        <title>Ruth&#x27;s Chris Steak House - Fairfax Restaurant - Fairfax, VA | OpenTable</title><meta name="viewport" content="width=device-width, :
        <script>window.OT = window.OT || {};
window.OT.SRS = {
  abTestingAnalytics: {"account":"2d65a94551a4a1caddd5a825a88698a89e19f2d5","hostname":"ac.opentable.com"},
  disableRaven: false,
  domainId: 1,
  jsHeadVersion: "1.0.92",
  lang: "en",
  ocUrl: window.location.protocol + '//oc-registry.opentable.com/v2/',
  requestId: "03cf6435-4f6c-4142-b8f6-ac50d3bd5925"
};

window.OT.searchAutoComplete = {};
window.OT.searchAutoComplete.ticketedRids = [681,1537,3334,7055,11065,29995,47119,53716,89905,100038,112846,114886,118765,140968,160504,185560,185773,:

window.oc = window.oc || {};
window.oc.conf = { globalParameters: { __ot_conservedHeaders: 'x2YPVT6POzqVVB9+77Ms1D6kzlo1qz4/Wcrw22iABQU0wBEL55I1QUmnPt1V76x53aanyfLoC38YVlaxuSeQScc(
</script><!--[if lte IE 8]> <script src="//components.otstatic.com/components/js-head/1.0.92/min/ie8_compat.js"></script> <![endif]-->
<!--[if IE 9]> <script src="//components.otstatic.com/components/js-head/1.0.92/min/ie9_compat.js"></script> <![endif]--><script src="//components.otst
<!--[if lte IE 9]> <script src="//components.otstatic.com/components/js-head/1.0.92/min/ie_jquery.js"></script> <![endif]--><script>window.optimizely :
        <link rel="shortcut icon" href="//components.otstatic.com/components/favicon/1.0.6/favicon/favicon.ico" type="image/x-icon"/><link rel="icon" |
      </head>
      <body>
        <div class="master-container master-container-redesign">
          <div class="header--legacy">
            <!--  Aria reference for this menu bar https://www.w3.org/TR/wai-aria-practices-1.1/examples/menubar/menubar-1/menubar-1.html#--><style>a.f
  e.preventDefault();
    var mainContentElement = jQuery('#container').first();
    jQuery('html, body').animate({
      scrollTop: (mainContentElement.offset().top)
    }, 500);
    // We can only focus a div if it has a tab index value. Set it to -1 so that we can focus it but we can't
    // tab to it and so we don't disrupt the normal tab navigation flow
    mainContentElement.attr('tabindex',-1).focus();
```

```




"/><link rel="stylesheet" type="text/css" href="//media.otstatic.com/restaurant-profile/index-aec1482bb94320feaf7f98c915f2b524.css"/><link rel="styl
```

93. On information and belief, the OpenTable Platform fetched HTML (*.html) files, CSS (*.css) files, and Javascript (*.js) files from the OpenTable Platform's servers and converted them with additional contents (e.g., JSON files and image files) into a working website. In the example below, the OpenTable Platform fetches its source code, such as HTML files and run time files (including CSS files, and Javascript files) from the OpenTable Platform's servers to build a restaurant profile for display. On information and belief, this functionality was present during the term of the '397 patent.



Name	Url	Status	Type	Initiator
ruths-chris-steak-house-fairf...	https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee...	200	document	Other
index-aec1482bb94320feaf7f...	https://media.otstatic.com/restaurant-profile/index-aec1482bb94320feaf7f98...	200	stylesheet	ruths-chris-steak-house-fairf...
header-3b185dafb4153aa4b...	https://media.otstatic.com/css/header-3b185dafb4153aa4b2982b32ca20eec0...	200	stylesheet	ruths-chris-steak-house-fairf...
footer-bdeb5e02752433775...	https://media.otstatic.com/css/footer-bdeb5e0275243377573ba0166fa984fc.c...	200	stylesheet	ruths-chris-steak-house-fairf...
slick.min.css	https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick.min.css	200	stylesheet	ruths-chris-steak-house-fairf...
slick-theme.min.css	https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick-theme.min.css	200	stylesheet	ruths-chris-steak-house-fairf...
bundle_head.js	https://components.otstatic.com/components/js-head/1.0.92/min/bundle_he...	200	script	ruths-chris-steak-house-fairf...
oc-client.min.js	https://components.otstatic.com/components/oc-client/0.48.15/src/oc-client...	200	script	ruths-chris-steak-house-fairf...
style.css	https://components.otstatic.com/components/dtp/1.8.10/dist/style.css	200	stylesheet	ruths-chris-steak-house-fairf...
opentable-logo.svg	https://components.otstatic.com/components/header/2.2.44/img/opentable-l...	200	svg+xml	ruths-chris-steak-house-fairf...
dtp.js?dtpId=dtp-picker-33	https://components.otstatic.com/components/dtp/1.8.10/dist/dtp.js?dtpId=d...	200	script	ruths-chris-steak-house-fairf...
gtm.js?id=GTM-MJTRF6F&l=...	https://www.googletagmanager.com/gtm.js?id=GTM-MJTRF6F&l=OTDataLayer	200	script	bundle_head.js:3259
Brandon_txt_reg-webfont.wo...	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_reg-webfont.wo...	200	font	index-aec1482....css
Brandon_txt_med-webfont.w...	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_med-webfont.w...	200	font	index-aec1482....css
Brandon_txt_bld-webfont.wo...	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_bld-webfont.wo...	200	font	index-aec1482....css
icons-eb482b5c449b4ac7d2...	https://media.otstatic.com/fonts/icons-eb482b5c449b4ac7d25370ba85f3808...	200	font	header-3b185da....css
data:image/svg+xml,...	data:image/svg+xml,%3Csvg width='17px' height='17px' viewBox='0 0 17 17'...	200	svg+xml	ruths-chris-steak-house-fairf...
icons-1e9096a34352dc1c6b...	https://media.otstatic.com/fonts/icons-1e9096a34352dc1c6bd4a76799aa77c...	200	font	ruths-chris-steak-house-fairf...
23872341.jpg	https://resizer.otstatic.com/v2/photos/wide-huge/1/23872341.jpg	200	jpeg	ruths-chris-steak-house-fairf...
staticmap?key=AIzaSyBHDIZ...	https://maps.googleapis.com/maps/api/staticmap?key=AIzaSyBHDIZeuOrKa...	200	jpeg	ruths-chris-steak-house-fairf...
small.jpg	https://images.otstatic.com/prod/26016062/1/small.jpg	200	jpeg	ruths-chris-steak-house-fairf...
provided_by_2.png?sp_dt=2...	https://a.singleplatform.com/OpenTable/ruths-chris-steak-house-fairfax/provi...	304	png	ruths-chris-steak-house-fairf...
prop-types.min.js	https://unpkg.com/prop-types@15.7.2/prop-types.min.js	200	script	0:792
analytics.js	https://www.google-analytics.com/analytics.js	200	script	gtm.js?id=GTM-MJTRF6F&l=...
data:image/svg+xml;...	data:image/svg+xml;charset=utf-8,%3Csvg width='24' height='24' viewBox='...	200	svg+xml	ruths-chris-steak-house-fairf...
data:image/svg+xml;...	data:image/svg+xml;charset=utf-8,%3Csvg xmlns='http://www.w3.org/2000/s...	200	svg+xml	ruths-chris-steak-house-fairf...
data:image/svg+xml;...	data:image/svg+xml;charset=utf-8,%3Csvg width='21' height='21' viewBox='...	200	svg+xml	ruths-chris-steak-house-fairf...
data:image/svg+xml;...	data:image/svg+xml;charset=utf-8,%3Csvg width='24' height='24' viewBox='...	200	svg+xml	ruths-chris-steak-house-fairf...
react.production.min.js	https://unpkg.com/react@16.9.0/umd/react.production.min.js	200	script	0:792
data:image/svg+xml;...	data:image/svg+xml;charset=utf-8,%3Csvg width='24' height='24' viewBox='...	200	svg+xml	ruths-chris-steak-house-fairf...
data:image/svg+xml;...	data:image/svg+xml;charset=utf-8,%3Csvg width='24' height='24' viewBox='...	200	svg+xml	ruths-chris-steak-house-fairf...
data:image/svg+xml;...	data:image/svg+xml;charset=utf-8,%3Csvg width='16' height='16' viewBox='...	200	svg+xml	ruths-chris-steak-house-fairf...
data:image/svg+xml;...	data:image/svg+xml;charset=utf-8,%3Csvg width='16' height='16' viewBox='...	200	svg+xml	ruths-chris-steak-house-fairf...
collect?t=dc&aip=1&_r=3&v...	https://stats.g.doubleclick.net/j/collect?t=dc&aip=1&_r=3&v=1&_v=j88&tid...	200	xhr	bundle_head.js:1288
collect?v=1&_v=j88&a=1209...	https://www.google-analytics.com/collect?v=1&_v=j88&a=120919018&t=p...	200	gif	analytics.js:26
react-dom.production.min.js	https://unpkg.com/react-dom@16.9.0/umd/react-dom.production.min.js	200	script	0:792



Name	Url	Status	Type	Initiator
large.jpg	https://images.otstatic.com/prod/24386187/1/large.jpg	200	jpeg	react-dom.production.min.js:...
large.jpg	https://images.otstatic.com/prod/24386186/1/large.jpg	200	jpeg	react-dom.production.min.js:...
large.jpg	https://images.otstatic.com/prod/24386185/1/large.jpg	200	jpeg	react-dom.production.min.js:...
large.jpg	https://images.otstatic.com/prod/24386184/1/large.jpg	200	jpeg	react-dom.production.min.js:...
large.jpg	https://images.otstatic.com/prod/24386183/1/large.jpg	200	jpeg	react-dom.production.min.js:...
large.jpg	https://images.otstatic.com/prod/24386182/1/large.jpg	200	jpeg	react-dom.production.min.js:...
large.jpg	https://images.otstatic.com/prod/24386181/1/large.jpg	200	jpeg	react-dom.production.min.js:...
inner-brighttag.html	https://srsv2.s3.amazonaws.com/components/js/1.10.0/min/inner-brighttag.h...	200	document	brighttag.html?data_domain...
tag.js	https://s.otstatic.com/tag.js	200	script	inner-brighttag.html
gtm.js?id=GTM-P2JGKLC&l=...	https://www.googletagmanager.com/gtm.js?id=GTM-P2JGKLC&l=OTMDataL...	200	script	inner-brighttag.html:7
ga-audiences?t=sr&aip=1&_...	https://www.google.com/ads/ga-audiences?t=sr&aip=1&_r=4&slf_rd=1&v=...	200	gif	analytics.js:26
widget_iframe.6e189c4f2b6d...	https://platform.twitter.com/widgets/widget_iframe.6e189c4f2b6d88c453045...	200	document	widgets.js:8
template.js	https://components.otstatic.com/components/user-info/1.1.19/template.js	200	script	0:792
tag?site=d0upCKz&docRefer...	https://s.thebrighttag.com/tag?site=d0upCKz&docReferrer=https%3A%2F%2...	200	script	tag.js:8
intersection-observer.js	https://components.otstatic.com/components/js/1.10.0/min/intersection-obs...	200	script	0:792
en.json	https://cdn.cookielaw.org/consent/c2727c7d-b78f-4d8f-96ff-78220790962f/a...	200	fetch	bundle_head.js:1340
conversion_async.js	https://www.googleadservices.com/pagead/conversion_async.js	200	script	gtm.js?id=GTM-P2JGKLC&l=...
activity;src=8086822;type=rt...	https://8086822.fls.doubleclick.net/activity;src=8086822;type=rtgco0;cat=glo...	302	documen...	gtm.js?id=GTM-P2JGKLC&l=...
bat.js	https://bat.bing.com/bat.js	200	script	VM229:2
?id=725308910857169&ev=...	https://www.facebook.com/tr/?id=725308910857169&ev=PageView&ts=161...	200	gif	VM239:1
anchor?ar=1&k=6Ldcf5gUA...	https://www.google.com/recaptcha/api2/anchor?ar=1&k=6Ldcf5gUAAAAAN...	200	document	recaptcha__en.js:246
runtime.js	https://unpkg.com/oc-jade-legacy@1.11.1/runtime.js	200	script	0:792
activity;dc_pre=Clq_hu6c_u4...	https://8086822.fls.doubleclick.net/activity;dc_pre=Clq_hu6c_u4CFejPKAUdJv...	200	document	8086822.fls.doubleclick.net/a...
?random=1614022096639&c...	https://googleads.g.doubleclick.net/pagead/viewthroughconversion/9844605...	200	script	VM255 conversion_async.js:34
?random=1614022096656&c...	https://www.googleadservices.com/pagead/conversion/763130297/?random...	200	script	conversion_async.js:34
0?ti=21003290&Ver=2&mid...	https://bat.bing.com/action/0?ti=21003290&Ver=2&mid=0cbeba5f-2c6e-42...	204	text/plain	bat.js:1
otFlat.json	https://cdn.cookielaw.org/scripttemplates/6.5.0/assets/otFlat.json	200	fetch	bundle_head.js:1340
otPcCenter.json	https://cdn.cookielaw.org/scripttemplates/6.5.0/assets/v2/otPcCenter.json	200	fetch	bundle_head.js:1340

94. The presence of the above referenced features, which on information and belief were present during the term of the ‘397 patent, is demonstrated, by way of example, by testing the Accused Instrumentalities for investigative purposes on https://www.booking.com/, https://join.booking.com/, https://partner.booking.com/en-us, and/or https://ycs.agoda.com/en-us/kipp/public/home, and by reference to publicly available information, including the following:

- https://join.booking.com/,
- https://partner.booking.com/en-us,

- https://partner.booking.com/en-us/help/working-booking/how-do-i-join-bookingcom,
- https://partner.booking.com/en-us/help/working-booking/how-can-i-set-my-property-easily,
- https://ycs.agoda.com/en-us/kipp/public/home,
- https://www.agoda.com/info/ycs-online-registration.html?cid=1844104,
- https://www.agoda.com/info/privacy.html?cid=1844104,
- https://partners.agoda.com/en-us/faq.html,
- https://agodapropertyhelp.zendesk.com/hc/en-us,
- https://partners.agoda.com/,
- https://agodapropertyhelp.zendesk.com/hc/en-us/articles/115009545508-How-do-I-list-my-property-on-Agoda-com-,
- https://www.opentable.com/,
- https://support.opentable.com/s/?language=en_US,
- https://platform.opentable.com/documentation/, and
- https://restaurant.opentable.com/.

95. On information and belief, Defendants have had knowledge of the ‘397 patent and their infringement thereof at least as early as October 3, 2019, and no later than November 30, 2020, when Plaintiff provided notice of the ‘397 patent and Defendants’ infringement of the ‘397 patent. Furthermore, Defendants have been aware of the ‘397 patent and their infringement thereof since at least the filing of the original Complaint, D.I. 1, on December 1, 2020.

96. On information and belief, Defendants have contributed to the infringement of the ‘397 patent because Defendants knew that the infringing aspects of their infringing products and services, including but not limited to the Accused Instrumentalities, were made for use in an infringement, and were not staple articles of commerce suitable for substantial non-infringing uses.

97. On information and belief, Defendants have induced the infringement of the ‘397 patent, with knowledge of the ‘397 patent and that their acts, including without limitation using,

offering to sell, selling within, and importing into the United States, the Accused Instrumentalities, would aid and abet and induce infringement by customers, clients, partners, developers, and end users of the foregoing.

98. In particular, Defendants' actions that aided and abetted others such as customers, clients, partners, developers, and end users to infringe included advertising and distributing the Accused Instrumentalities, providing instructional materials, training, and other services regarding the Accused Instrumentalities, and providing free listings for the Accused Instrumentalities. Defendants actively encouraged the adoption of the Accused Instrumentalities and provided support sites for the vast network of developers working with the Accused Instrumentalities, emphasizing the simple and user-friendly nature of the Accused Instrumentalities, for example, explaining that "Registration can take as little as 15 minutes to complete—get started today" and that Defendants provides "24/7 support by phone or email" (*see, e.g.*, https://join.booking.com/), that "Through one platform our hotel partners are distributed across both priceline.com and agoda.com maximizing performance across our Retail, Private, Opaque and Vacation Packages programs. Our partners can easily set rates, promotions, and change hotel details with a few simple clicks of a mouse or on the go with a mobile device" (*see, e.g.*, https://ycs.agoda.com/en-us/kipp/public/home), and that "From online ordering and takeout to powerful marketing and experiences, make more money when you access our network of millions" and that restaurants can "[g]et discovered and capture the business of the millions of people, around the world and in your neighborhood, searching on OpenTable" (*see, e.g.*, https://restaurant.opentable.com/?utm_source=dinersite&utm_medium=referral&utm_campaign=topnav&Lead.LeadSource=DinerSite&Lead.Marketing_ID__c=topnav). On information and belief, Defendants engaged in such actions with specific intent to cause infringement or with willful blindness to the resulting infringement because Defendants had actual knowledge of the '397 patent and knowledge that their acts were inducing infringement of the '397 patent since at least the date Defendants received notice that their activities infringed the '397 patent.

99. Defendants' acts of infringement have caused damage to Plaintiff, and Plaintiff is

entitled to recover damages from Defendants in an amount subject to proof at trial.

100. On information and belief, Defendants have acted with disregard of Plaintiff's patent rights, without any reasonable basis for doing so, and have willfully infringed the '397 patent.

101. The foregoing is illustrative of Defendants' infringement of the '397 patent. Plaintiff reserves the right to identify additional claims and Accused Instrumentalities in accordance with the Court's local rules and applicable scheduling orders.

COUNT II – INFRINGEMENT OF U.S. PATENT NO. 7,594,168

102. Plaintiff incorporates by reference paragraphs 1 to 55 above as if fully set forth herein.

103. On information and belief, Defendants have infringed and continue to infringe the '168 patent under 35 U.S.C. § 271, either literally and/or under the doctrine of equivalents, directly and/or indirectly.

104. On information and belief, Booking-BV has infringed and continues to infringe the '168 patent by performing, without authority, one or more of the following acts during relevant time periods: making, using, offering to sell, selling within, and importing into, the United States products and services that practice the claimed inventions of the '168 patent, including but not limited to the WebDirect Platform and the Booking Platform.[3]

105. On information and belief, Priceline and Agoda (the "YCS infringers") have infringed and continue to infringe the '168 patent by performing, without authority, one or more of the following acts during relevant time periods: making, using, offering to sell, selling within, and importing into, the United States products and services that practice the claimed inventions of

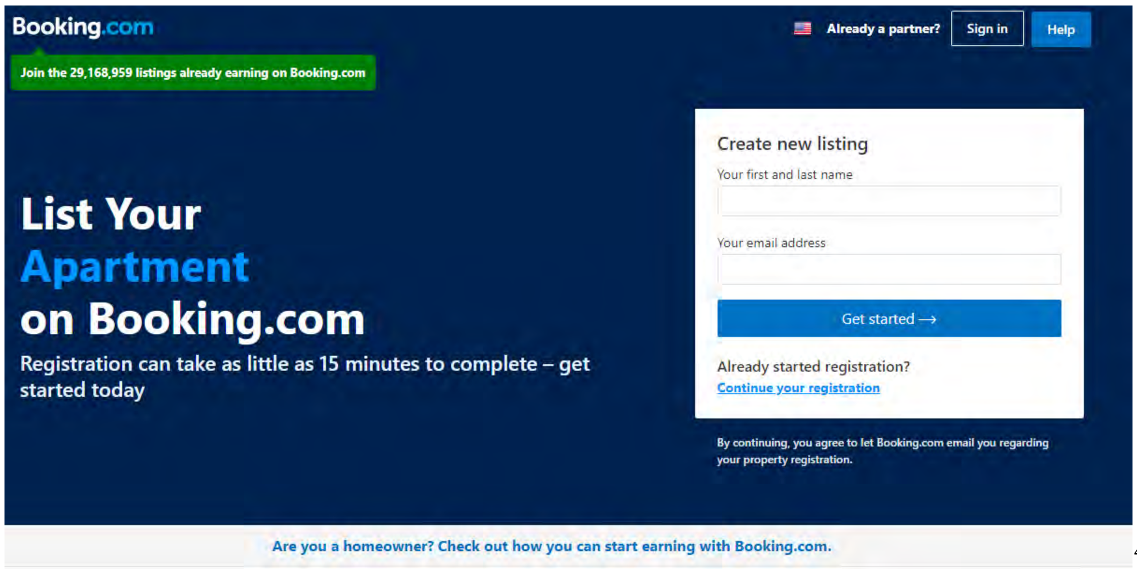
[3] This Count focuses its infringement allegations on the Booking Platform, YCS Platform, and the OpenTable Platform. Upon information and belief, the WebDirect Platform operates in a similar fashion as the Booking and YCS Platforms, and discovery of the WebDirect Platform, including confidential documents related thereto, will confirm these facts.

the ‘168 patent, including but not limited to the YCS Platform.

106. On information and belief, OpenTable has infringed and continues to infringe the ‘168 patent by performing, without authority, one or more of the following acts during relevant time periods: making, using, offering to sell, selling within, and importing into the United States products and services that practice the claimed inventions of the ‘168 patent, including but not limited to the OpenTable Platform.

<u>The Booking Platform:</u>

107. Booking-BV infringes at least claim 1 of the ‘168 patent through a combination of features in the Booking Platform that collectively practice each limitation of claim 1. By way of example, Booking-BV provides the Booking Platform for creating property listing websites for hotels, homes, apartments, and other lodgings where travelers can view and book accommodations.


Booking.com

Join the 29,168,959 listings already earning on Booking.com

Already a partner? Sign in Help

List Your Apartment on Booking.com

Registration can take as little as 15 minutes to complete – get started today

Create new listing

Your first and last name

Your email address

Get started ⟶

Already started registration?

Continue your registration

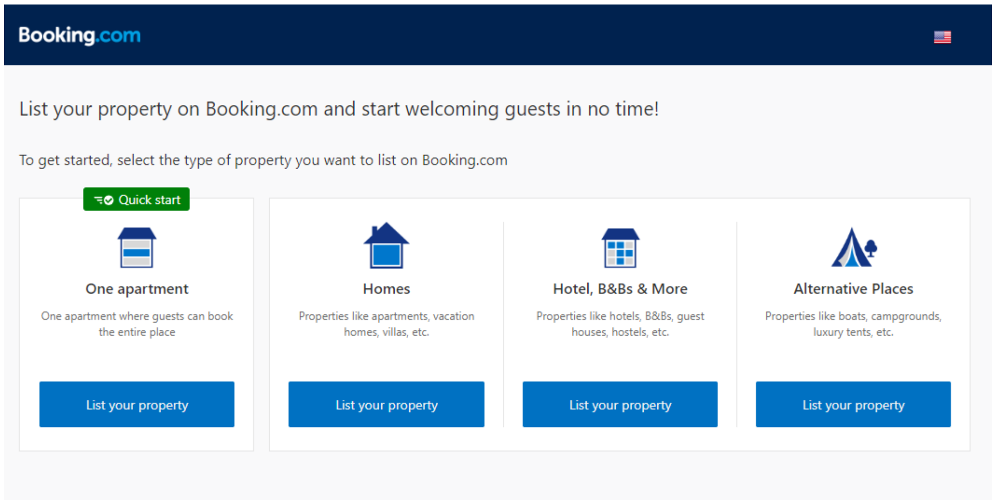
By continuing, you agree to let Booking.com email you regarding your property registration.

Are you a homeowner? Check out how you can start earning with Booking.com.

[4]

---

4 Unless otherwise noted, the images presented in this Count were generated for investigative purposes by testing the Accused Instrumentalities on https://www.booking.com/, https://join.booking.com/, https://partner.booking.com/en-us, https://ycs.agoda.com/en-us/kipp/public/home, https://www.opentable.com/, https://support.opentable.com/s/?language

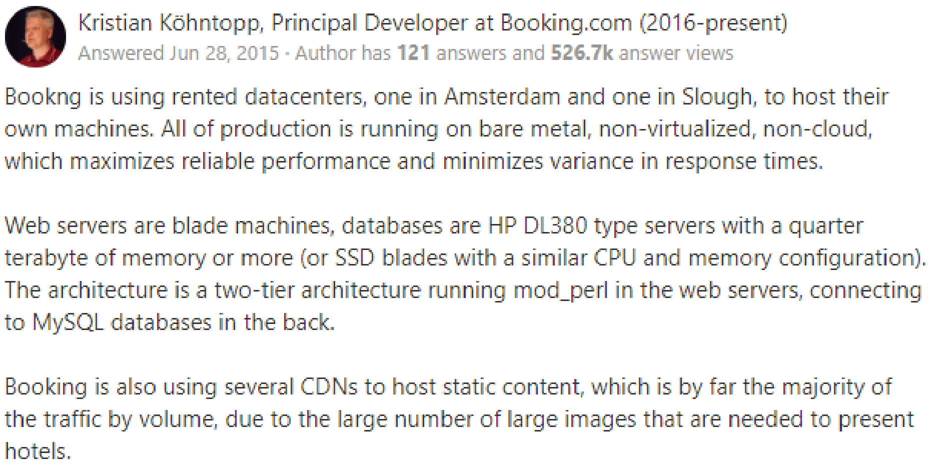

Booking.com

List your property on Booking.com and start welcoming guests in no time!

To get started, select the type of property you want to list on Booking.com

Quick start
One apartment
One apartment where guests can book the entire place
List your property
Homes
Properties like apartments, vacation homes, villas, etc.
List your property
Hotel, B&Bs & More
Properties like hotels, B&Bs, guest houses, hostels, etc.
List your property
Alternative Places
Properties like boats, campgrounds, luxury tents, etc.
List your property

108. On information and belief, property listing websites created on the Booking Platform are supported by MySQL, a relational database management system (RDBMS) that serves a backend server to retrieve and store data for the websites, including data relating to inputs from users.


Kristian Köhntopp, Principal Developer at Booking.com (2016-present)
Answered Jun 28, 2015 · Author has **121** answers and **526.7k** answer views

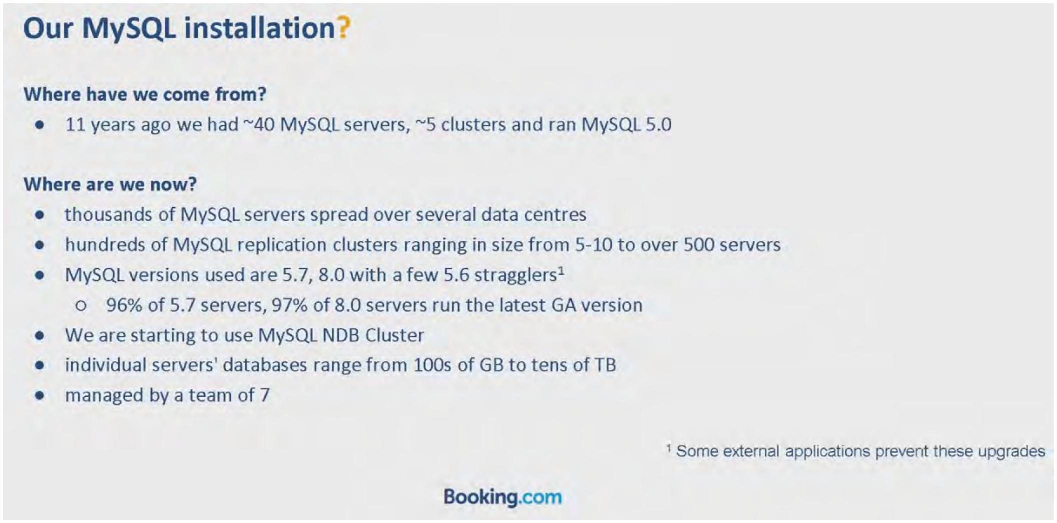
Bookng is using rented datacenters, one in Amsterdam and one in Slough, to host their own machines. All of production is running on bare metal, non-virtualized, non-cloud, which maximizes reliable performance and minimizes variance in response times.

Web servers are blade machines, databases are HP DL380 type servers with a quarter terabyte of memory or more (or SSD blades with a similar CPU and memory configuration). The architecture is a two-tier architecture running mod_perl in the web servers, connecting to MySQL databases in the back.

Booking is also using several CDNs to host static content, which is by far the majority of the traffic by volume, due to the large number of large images that are needed to present hotels.

https://www.quora.com/Who-hosts-the-booking-com-website

=en_US, https://platform.opentable.com/documentation/, and/or other associated websites.



Our MySQL installation?

Where have we come from?

- 11 years ago we had ~40 MySQL servers, ~5 clusters and ran MySQL 5.0

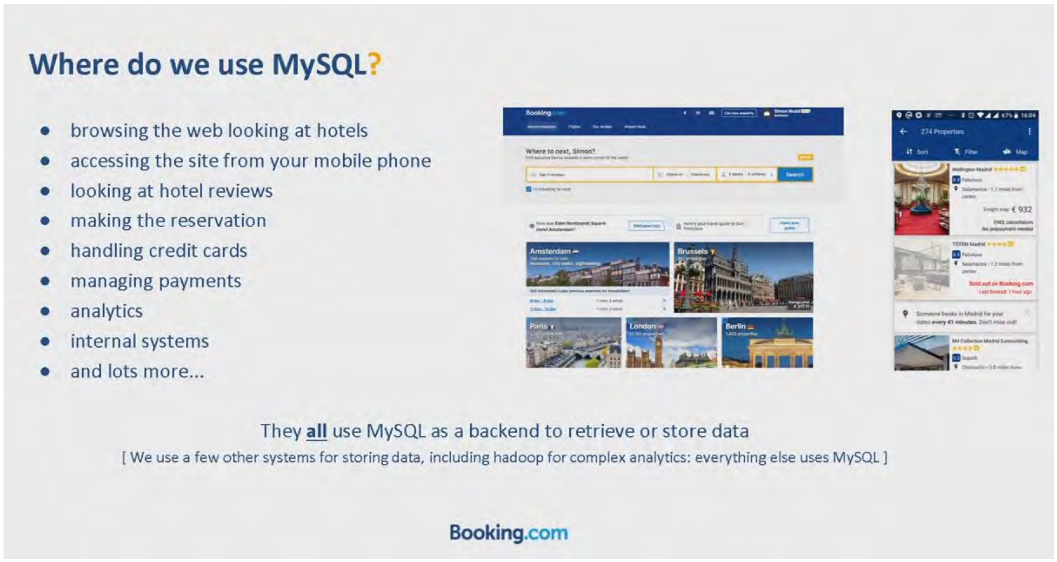
Where are we now?

- thousands of MySQL servers spread over several data centres
- hundreds of MySQL replication clusters ranging in size from 5-10 to over 500 servers
- MySQL versions used are 5.7, 8.0 with a few 5.6 stragglers[1]
  - 96% of 5.7 servers, 97% of 8.0 servers run the latest GA version
- We are starting to use MySQL NDB Cluster
- individual servers' databases range from 100s of GB to tens of TB
- managed by a team of 7

[1] Some external applications prevent these upgrades

Booking.com

https://www.youtube.com/watch?v=iNxqZSbaHYQ&feature=youtu.be



Where do we use MySQL?

- browsing the web looking at hotels
- accessing the site from your mobile phone
- looking at hotel reviews
- making the reservation
- handling credit cards
- managing payments
- analytics
- internal systems
- and lots more...

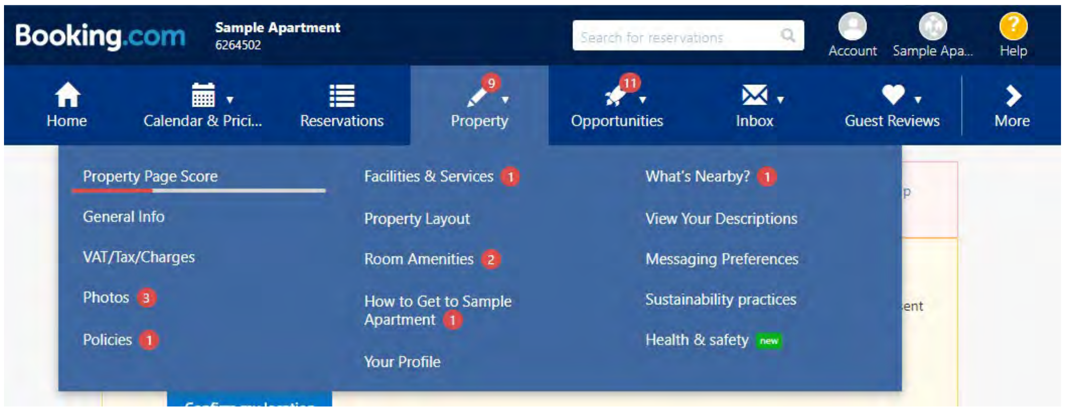
They **all** use MySQL as a backend to retrieve or store data

[We use a few other systems for storing data, including hadoop for complex analytics: everything else uses MySQL]

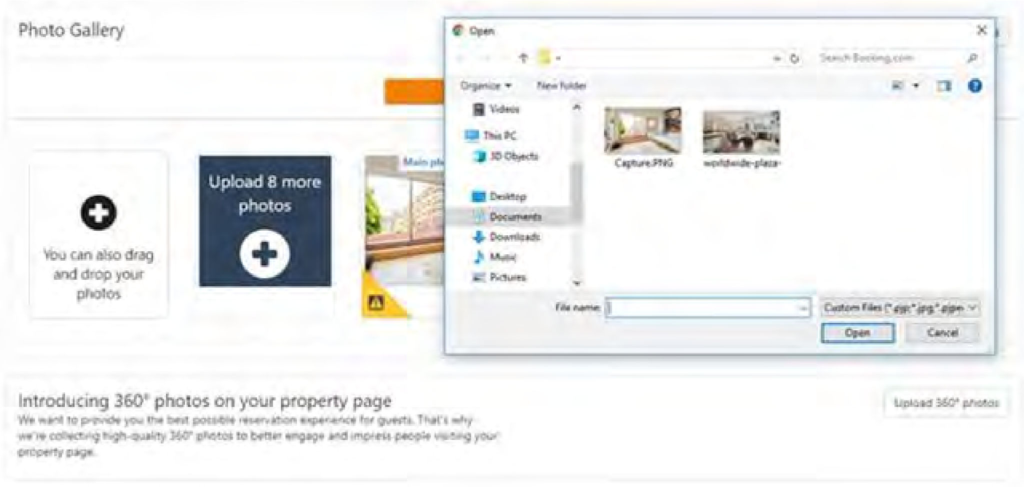
Booking.com

https://www.youtube.com/watch?v=iNxqZSbaHYQ&feature=youtu.be

109. The Booking Platform's MySQL server thus operates as a build engine that can accept user input to create a website comprising a plurality of web pages that each include objects. User input for a property listing website may include customization of general information such as name and location of the property, facilities and services, amenities, descriptions, and images.



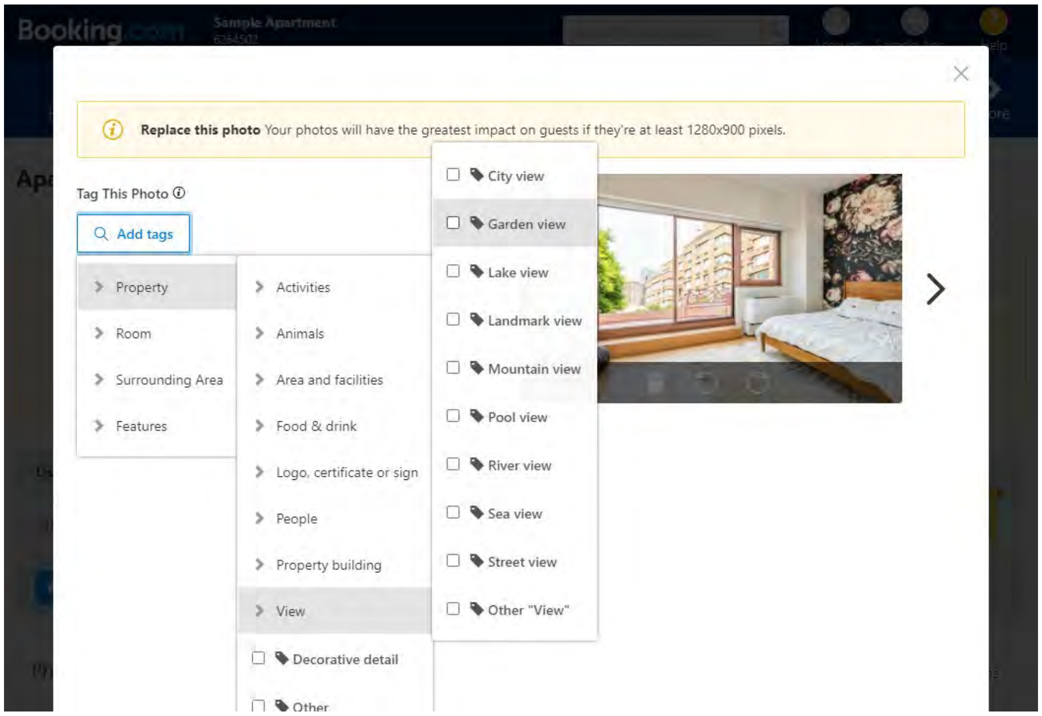
110. For example, a user may select images of the property to be displayed on the property website and may select the order in which the images are to be displayed by dragging the images with a mouse.



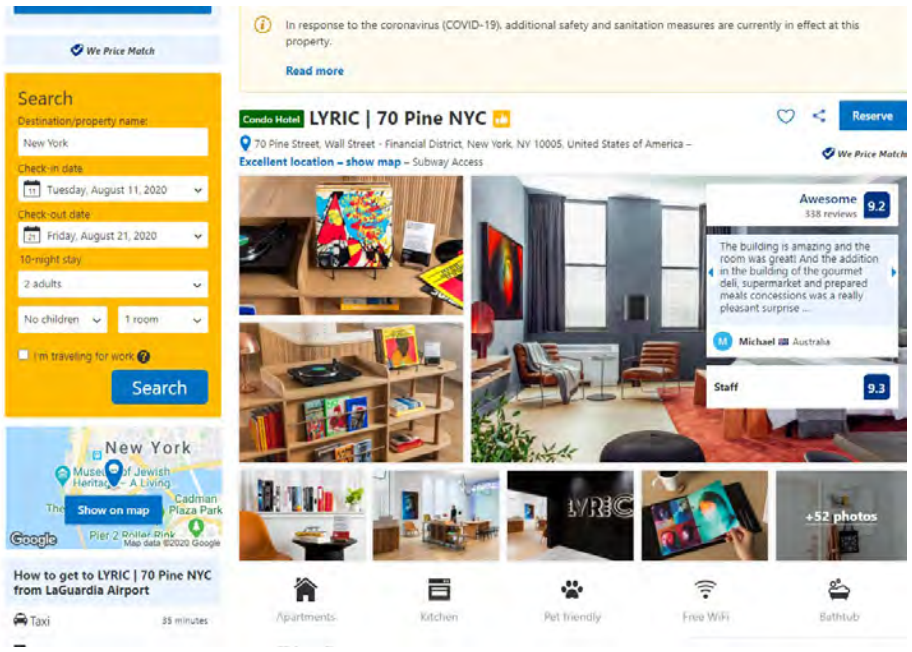
Introducing 360° photos on your property page

We want to provide you the best possible reservation experience for guests. That's why we're collecting high-quality 360° photos to better engage and impress people visiting your property page.

111. Upon input of an image, a user may associate a particular style to be associated with the image object. For example, a user may associate tags with an image. In the example below, the tag “Garden view” is associated with an image.


Booking.com
Sample Apartment
Replace this photo Your photos will have the greatest impact on guests if they're at least 1280x900 pixels.
Tag This Photo
Add tags
Property
Room
Surrounding Area
Features
Activities
Animals
Area and facilities
Food & drink
Logo, certificate or sign
People
Property building
View
Decorative detail
Other
City view
Garden view
Lake view
Landmark view
Mountain view
Pool view
River view
Sea view
Street view
Other "View"

112. A style for an image on the website may define transformations and time lines for the image. For example, when a thumbnail image in the photo gallery for a property on a Booking website is clicked on, the thumbnail image transforms into an enlarged image including a black background and an image caption at the bottom of screen. Further, the forward arrow on an enlarged image can be clicked on to transform the current enlarged image into the next enlarged image of the photo gallery, providing a photo gallery slideshow.


In response to the coronavirus (COVID-19), additional safety and sanitation measures are currently in effect at this property.

Read more

Condo Hotel LYRIC | 70 Pine NYC

70 Pine Street, Wall Street - Financial District, New York, NY 10005, United States of America – Excellent location – show map – Subway Access

Search

Destination/property name: New York

Check-in date: Tuesday, August 11, 2020

Check-out date: Friday, August 21, 2020

10-night stay

2 adults

No children

1 room

I'm traveling for work

Awesome 9.2 338 reviews

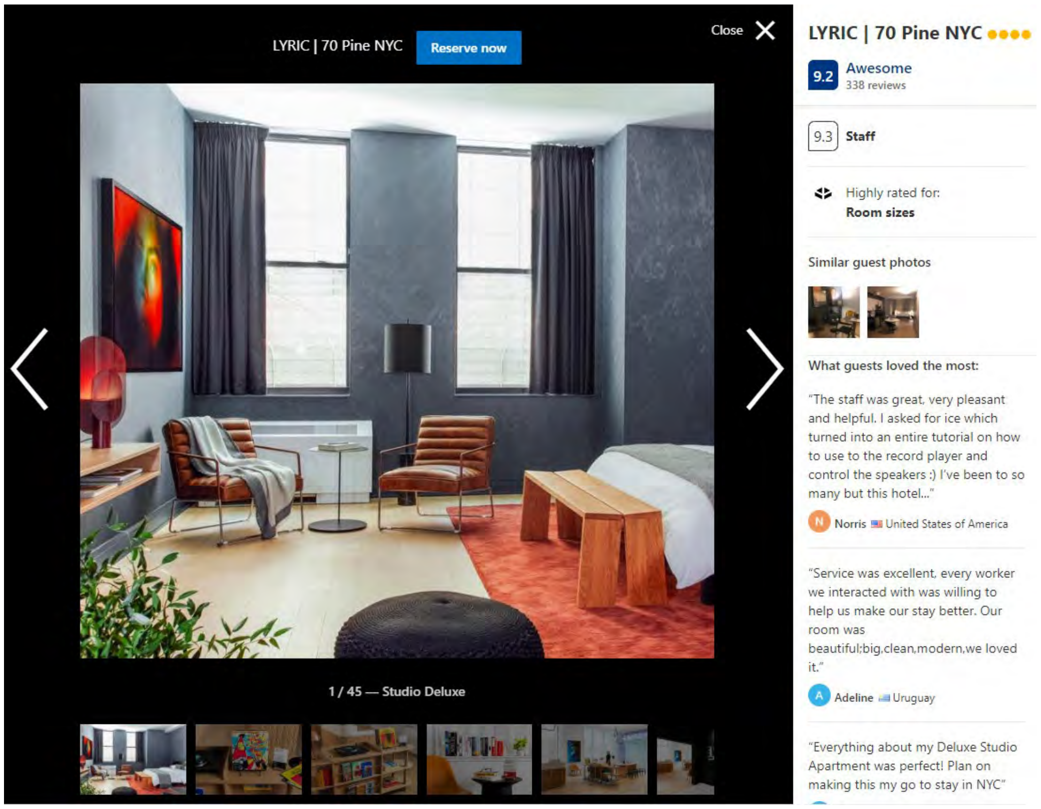
The building is amazing and the room was great! And the addition in the building of the gourmet deli, supermarket and prepared meals concessions was a really pleasant surprise ...

Michael Australia

Staff 9.3

How to get to LYRIC | 70 Pine NYC from LaGuardia Airport

Taxi 35 minutes



"The staff was great, very pleasant and helpful. I asked for ice which turned into an entire tutorial on how to use to the record player and control the speakers :) I've been to so many but this hotel..."

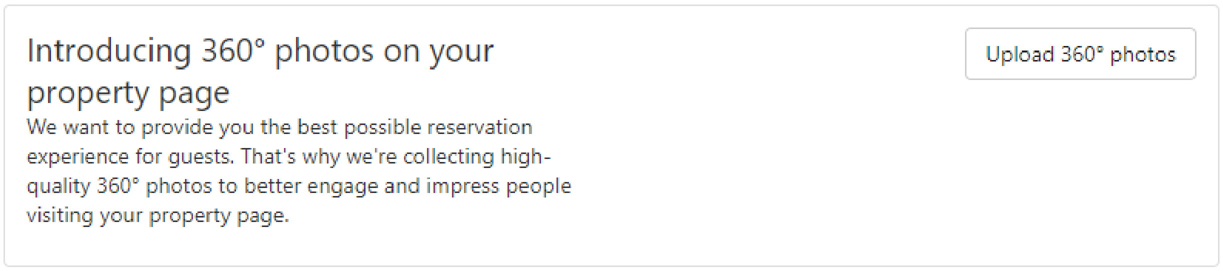
Norris United States of America

"Service was excellent, every worker we interacted with was willing to help us make our stay better. Our room was beautiful;big,clean,modern,we loved it."

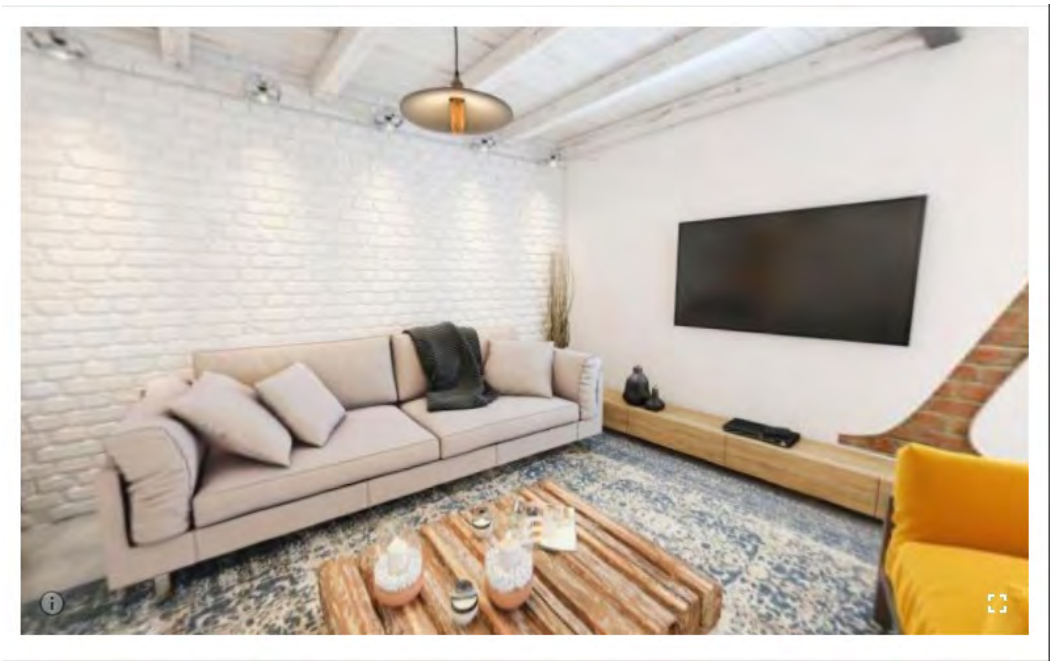
Adeline Uruguay

"Everything about my Deluxe Studio Apartment was perfect! Plan on making this my go to stay in NYC"

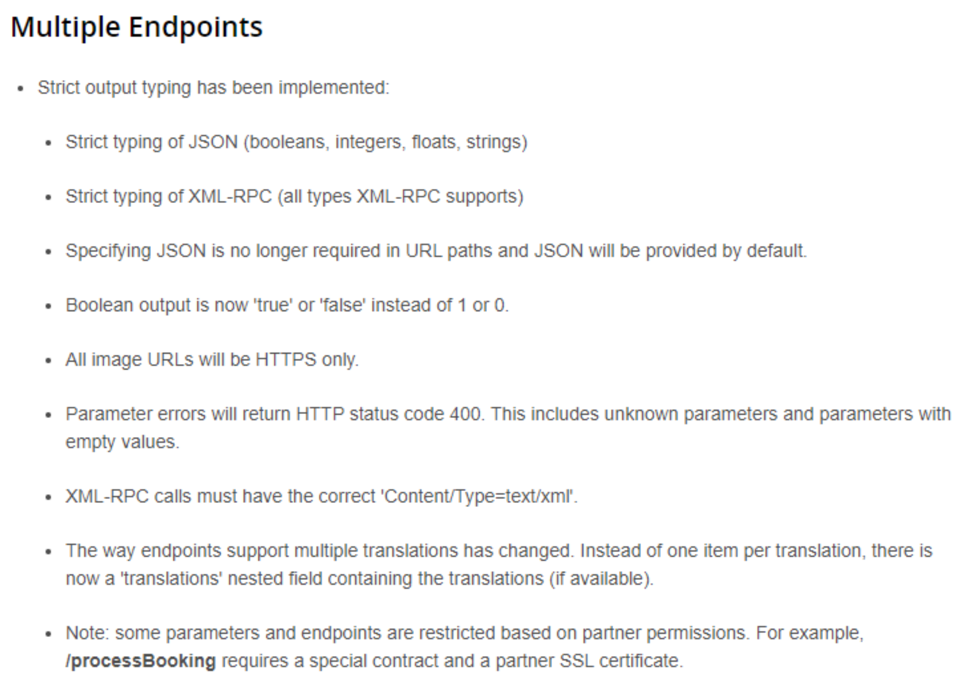
113. A 360 panoramic view style is another example of a style defining transformations and time lines that can be associated with an image. In the example below, a 360 view style for an image allows website visitors to click and drag on the image to get a 360 degree view of a particular room or area of the property.


Introducing 360° photos on your property page

We want to provide you the best possible reservation experience for guests. That's why we're collecting high-quality 360° photos to better engage and impress people visiting your property page.



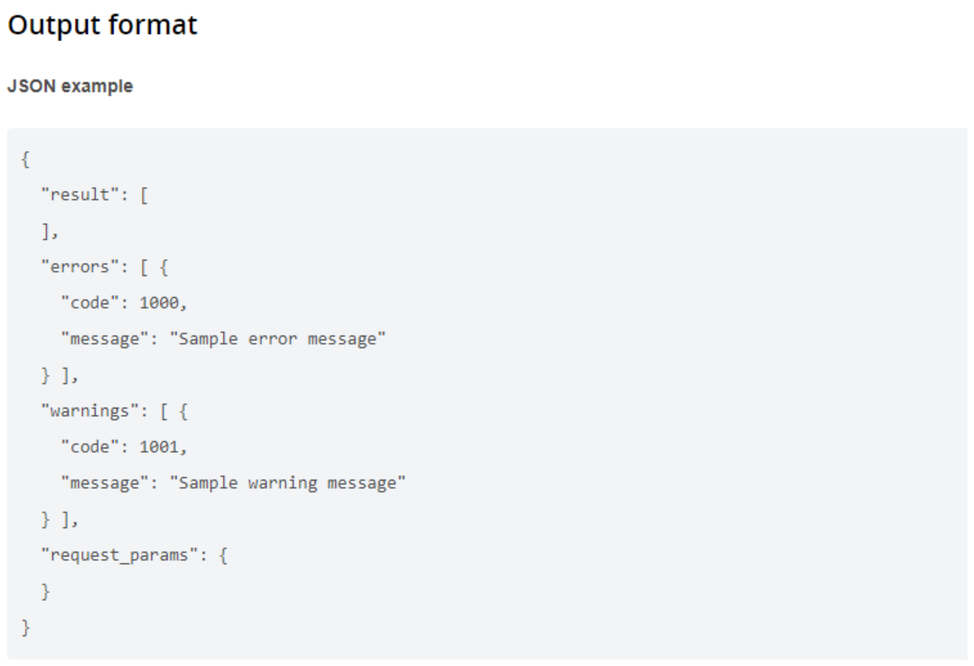
114. The Booking Platform is configured to produce a database with multidimensional array including the objects that comprise the website. For example, as described above, user inputs for Booking websites are stored in MySQL databases that are multidimensional array databases. Further, Booking websites are rendered using JSON dataset, which are multidimensional in nature, comprising key/value pairs and an ordered list of values.



Multiple Endpoints

- Strict output typing has been implemented:
  - Strict typing of JSON (booleans, integers, floats, strings)
  - Strict typing of XML-RPC (all types XML-RPC supports)
  - Specifying JSON is no longer required in URL paths and JSON will be provided by default.
  - Boolean output is now 'true' or 'false' instead of 1 or 0.
  - All image URLs will be HTTPS only.
  - Parameter errors will return HTTP status code 400. This includes unknown parameters and parameters with empty values.
  - XML-RPC calls must have the correct 'Content/Type=text/xml'.
  - The way endpoints support multiple translations has changed. Instead of one item per translation, there is now a 'translations' nested field containing the translations (if available).
  - Note: some parameters and endpoints are restricted based on partner permissions. For example, **/processBooking** requires a special contract and a partner SSL certificate.

https://developers.booking.com/api/commercial/index.html?version=2.5&page_url=migration-guide

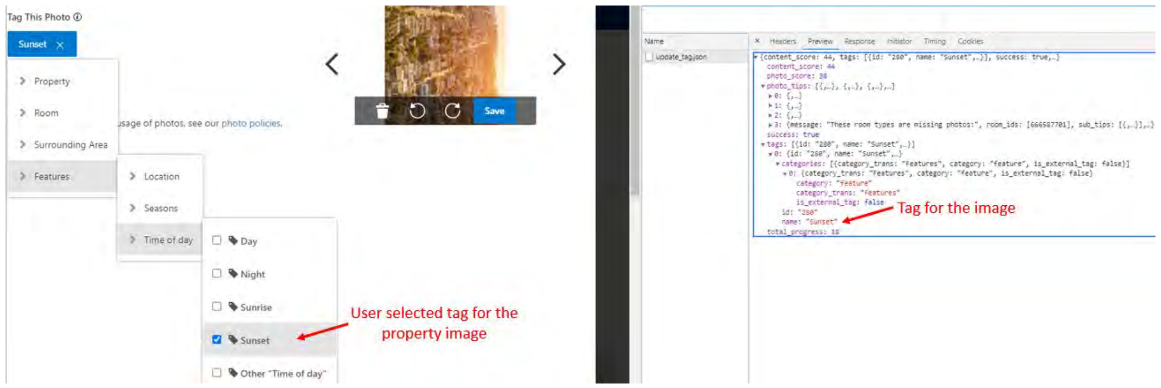

Output format

JSON example

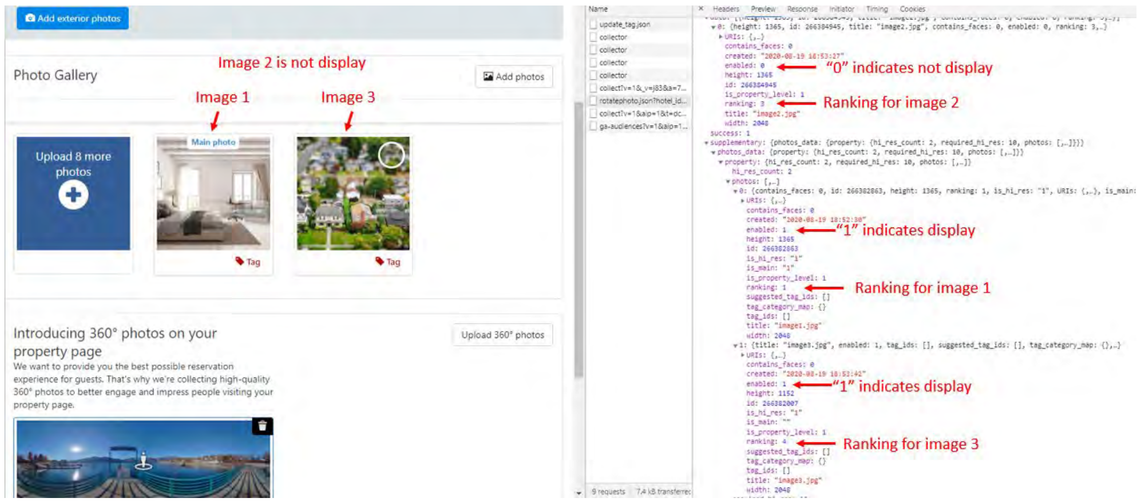
```
{
  "result": [
  ],
  "errors": [ {
    "code": 1000,
    "message": "Sample error message"
  } ],
  "warnings": [ {
    "code": 1001,
    "message": "Sample warning message"
  } ],
  "request_params": {
  }
}
```

115. For each website object, JSON dataset includes object style, object identifiers, and an indication of the web page that the object is part of. This data is provided to the Booking's server, accessible to a web browser to generate a website based on user inputs. In the example below, a user selects a tag "Sunset" for an image object "image2.jpg" and rotates the image by 90 degrees clockwise. In the response JSON dataset, the symbolic name "tags" is associated with the user selected tag name "Sunset" for the image object. The symbolic name "is_external_tag," which is associated with value "false," indicates the style of the tab object for the image object. In addition, the symbolic name "title" is the object identifier for each image object. The symbolic name "ranking" is the object style that indicates the display order of each image object. Since the symbolic name "ranking" is set to "1," the image object "image1.jpg" is presented first in order in the photo gallery for the property. The symbolic name "enabled" is another object style that indicates whether an image object is displayed or not on the Booking flatform. Since the symbolic

name "enabled" is set to "0," the image object "image2.jpg" is not immediately displayed on the website.



```
▾{content_score: 44, tags: [{id: "280", name: "Sunset",…}], success: true,…}
  content_score: 44
  photo_score: 38
 ▾photo_tips: [{,…}, {,…}, {,…},…]
  ▸0: {,…}
  ▸1: {,…}
  ▸2: {,…}
  ▸3: {message: "These room types are missing photos:", room_ids: [666587701], sub_tips: [{,…}],…}
  success: true
 ▾tags: [{id: "280", name: "Sunset",…}]
  ▾0: {id: "280", name: "Sunset",…}
   ▾categories: [{category_trans: "Features", category: "feature", is_external_tag: false}]
    ▾0: {category_trans: "Features", category: "feature", is_external_tag: false}
       category: "feature"
       category_trans: "Features"
       is_external_tag: false
     id: "280"
     name: "Sunset"
  total_progress: 38
```



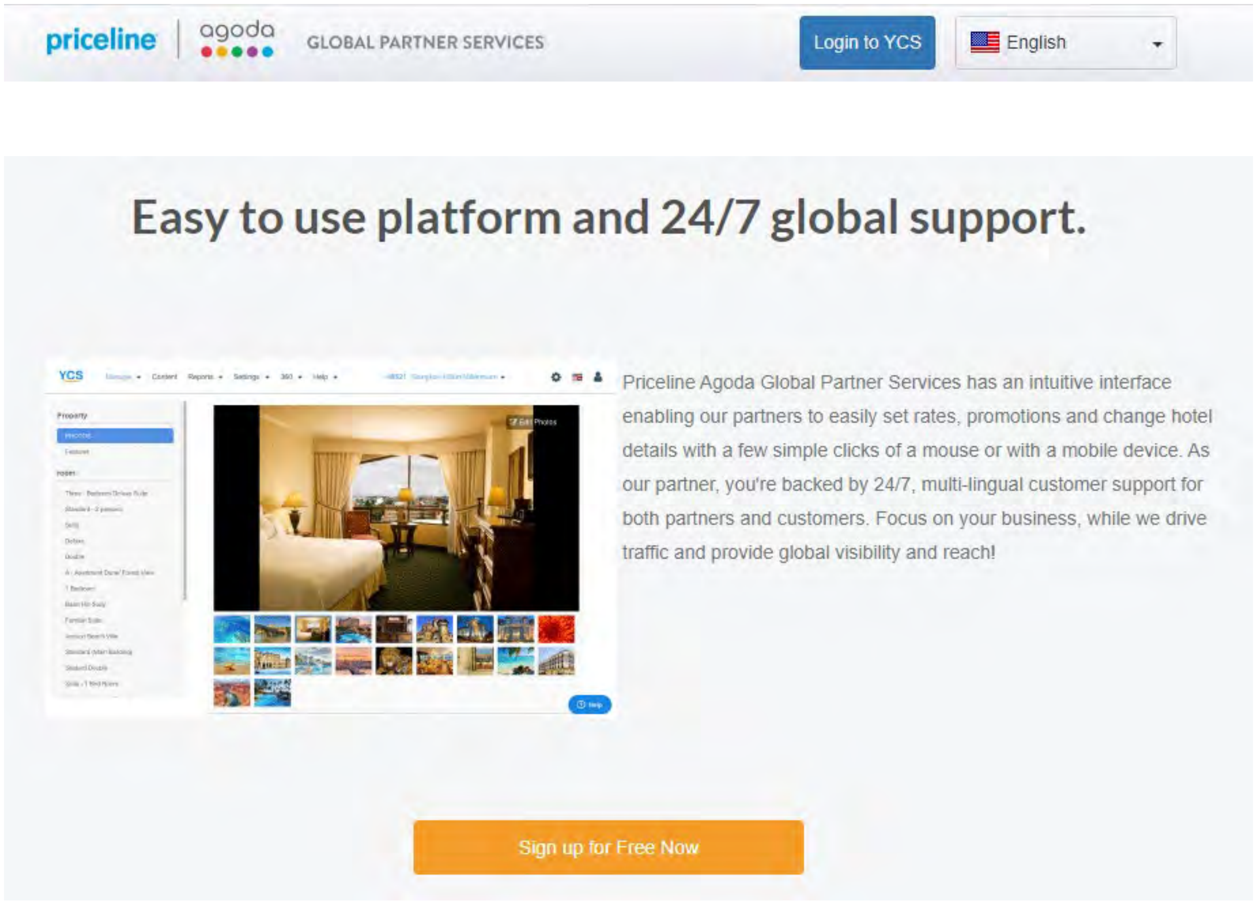
```
 ▾0: {height: 1365, id: 266384945, title: "image2.jpg", contains_faces: 0, enabled: 0, ranking: 3,…}
   ▸URIs: {,…}
    contains_faces: 0
    created: "2020-08-19 18:53:27"
    enabled: 0
    height: 1365
    id: 266384945
    is_property_level: 1
    ranking: 3
    title: "image2.jpg"
    width: 2048
 success: 1
▾supplementary: {photos_data: {property: {hi_res_count: 2, required_hi_res: 10, photos: [,…]}}}
 ▾photos_data: {property: {hi_res_count: 2, required_hi_res: 10, photos: [,…]}}
  ▾property: {hi_res_count: 2, required_hi_res: 10, photos: [,…]}
     hi_res_count: 2
    ▾photos: [,…]
     ▾0: {contains_faces: 0, id: 266382863, height: 1365, ranking: 1, is_hi_res: "1", URIs: {,…}, is_main:
       ▸URIs: {,…}
        contains_faces: 0
        created: "2020-08-19 18:52:30"
        enabled: 1
        height: 1365
        id: 266382863
        is_hi_res: "1"
        is_main: "1"
        is_property_level: 1
        ranking: 1
        suggested_tag_ids: []
        tag_category_map: {}
        tag_ids: []
        title: "image1.jpg"
        width: 2048
     ▾1: {title: "image3.jpg", enabled: 1, tag_ids: [], suggested_tag_ids: [], tag_category_map: {},…}
       ▸URIs: {,…}
        contains_faces: 0
        created: "2020-08-19 18:53:42"
        enabled: 1
        height: 1152
        id: 266382007
        is_hi_res: "1"
        is_main: ""
        is_property_level: 1
        ranking: 4
        suggested_tag_ids: []
        tag_category_map: {}
        tag_ids: []
        title: "image3.jpg"
        width: 2048
```

116. A web browser with access to a runtime engine can then generate the user configured website including the user input objects and style data extracted from the database. For example, a web browser can generate a web page for "Photo Gallery." Whether the image objects are to be displayed on the "Photo Gallery" web page is determined based on the value of the "enabled" object style, and their display order is determined based on the value of the "ranking" object style. In other words, the "enabled" and "ranking" are symbolic names associated with user input style data and "tags" and "title" are symbolic names associated with user input tag objects

and image objects, respectively.

<u>The YCS Platform:</u>

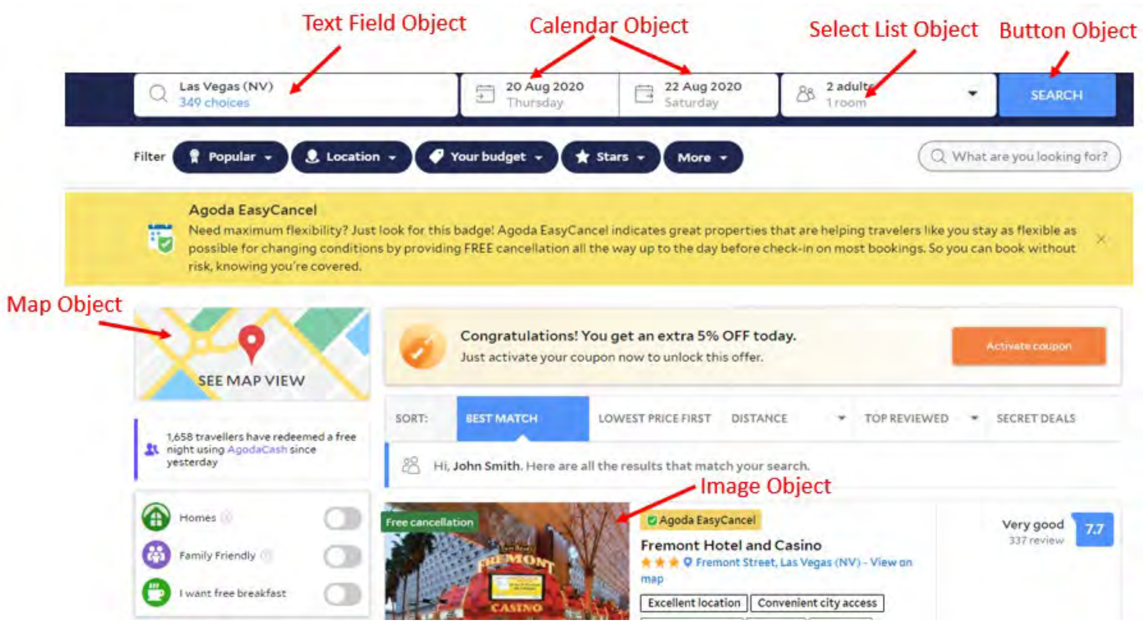
117. The YCS infringers infringe at least claim 1 of the '168 patent through a combination of features in the YCS Platform that collectively practice each limitation of claim 1. By way of example, the YCS infringers provide the YCS Platform for creating property listing websites for hotels and other lodgings where travelers can view and book accommodations.



Easy to use platform and 24/7 global support.

Priceline Agoda Global Partner Services has an intuitive interface enabling our partners to easily set rates, promotions and change hotel details with a few simple clicks of a mouse or with a mobile device. As our partner, you're backed by 24/7, multi-lingual customer support for both partners and customers. Focus on your business, while we drive traffic and provide global visibility and reach!

118. On information and belief, the YCS Platform is supported by a server comprising a build engine configured to accept user input to create a website comprising a plurality of web pages that each include objects displaying different elements of the website. The exemplary screenshot below shows a web page including various objects displaying different elements of a property listing website.



Agoda EasyCancel

Need maximum flexibility? Just look for this badge! Agoda EasyCancel indicates great properties that are helping travelers like you stay as flexible as possible for changing conditions by providing FREE cancellation all the way up to the day before check-in on most bookings. So you can book without risk, knowing you're covered.

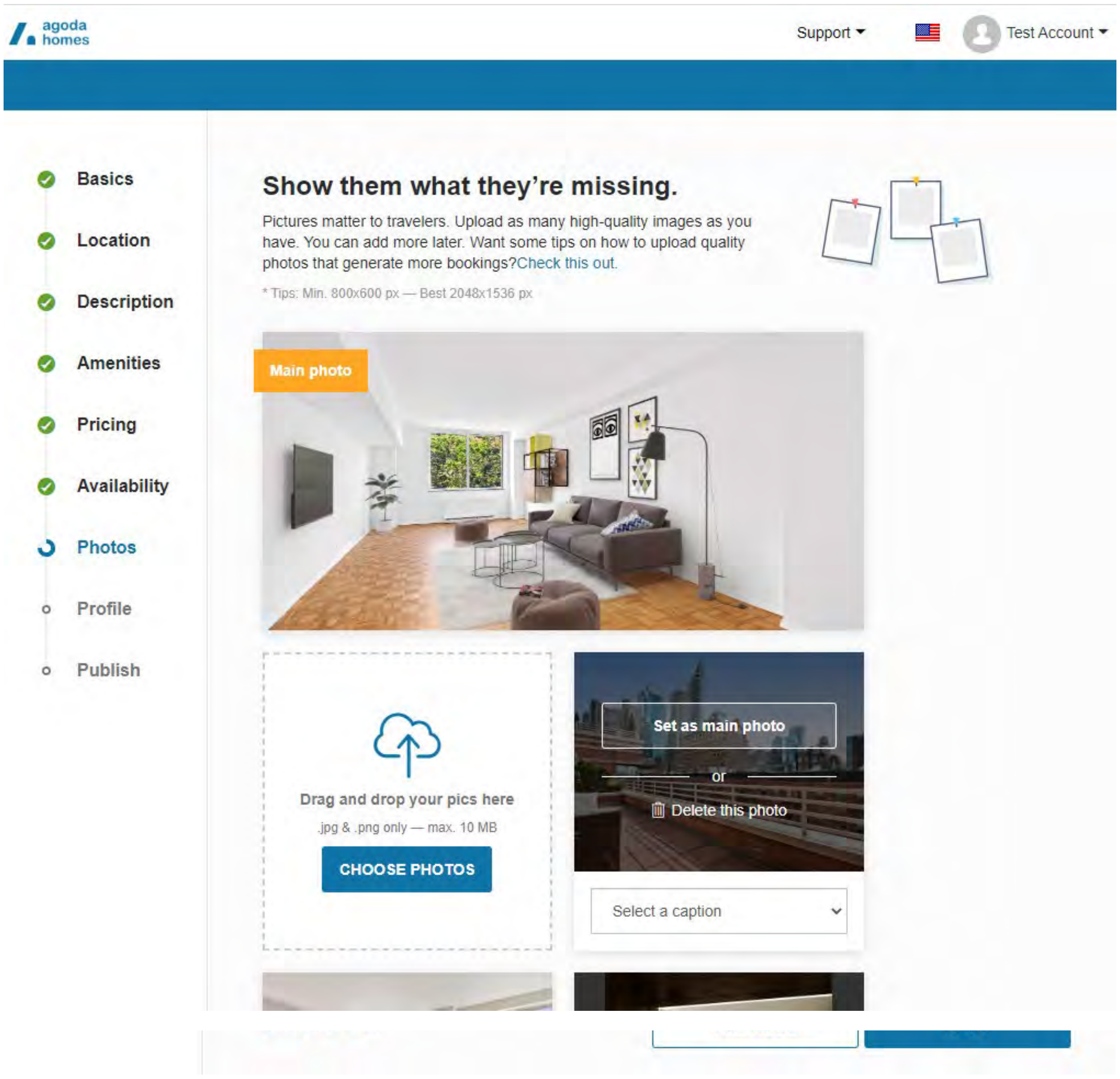
Congratulations! You get an extra 5% OFF today.
Just activate your coupon now to unlock this offer.

1,658 travellers have redeemed a free night using AgodaCash since yesterday

Hi, John Smith. Here are all the results that match your search.

Fremont Hotel and Casino
Fremont Street, Las Vegas (NV) - View on map

119. User input for a website may include customization of the location, description, amenities, pricing, availability, photos, and profile for a listed property. Further, a user may enter inputs to associate styles with image objects, including styles that define a transformation and time line. For example, a user may define a caption for an image that is displayed when the image is transformed, as shown below.

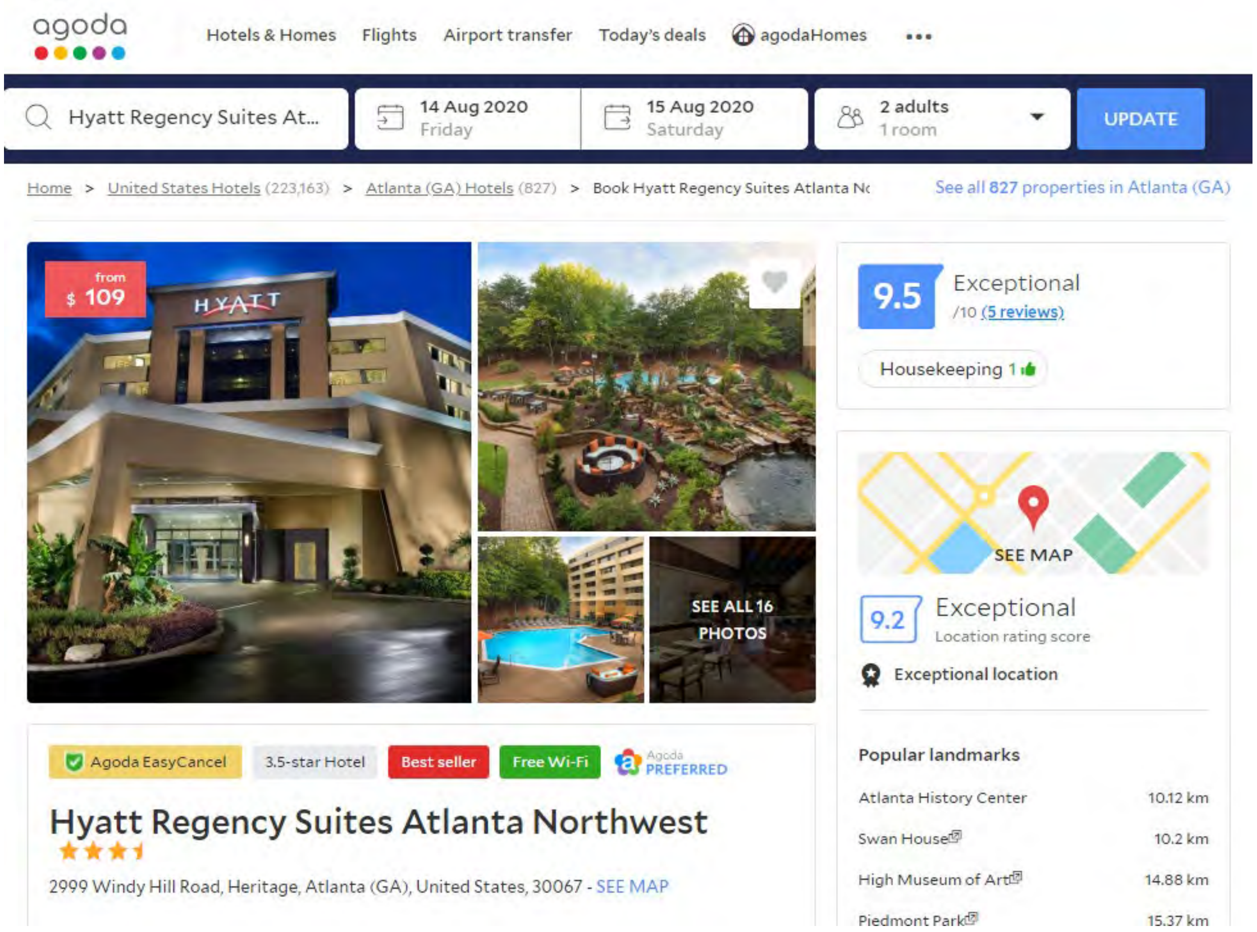


Show them what they're missing.

Pictures matter to travelers. Upload as many high-quality images as you have. You can add more later. Want some tips on how to upload quality photos that generate more bookings?Check this out.

* Tips: Min. 800x600 px — Best 2048x1536 px

120. When a thumbnail image of a property on the property listing website is clicked on, the thumbnail image is transformed into an enlarged image, and displayed together with a black background and the selected caption describing the image. Further, the forward arrow on an enlarged image can be clicked on to transform the current enlarged image into the next enlarged image of the photo gallery, providing a photo gallery slideshow.

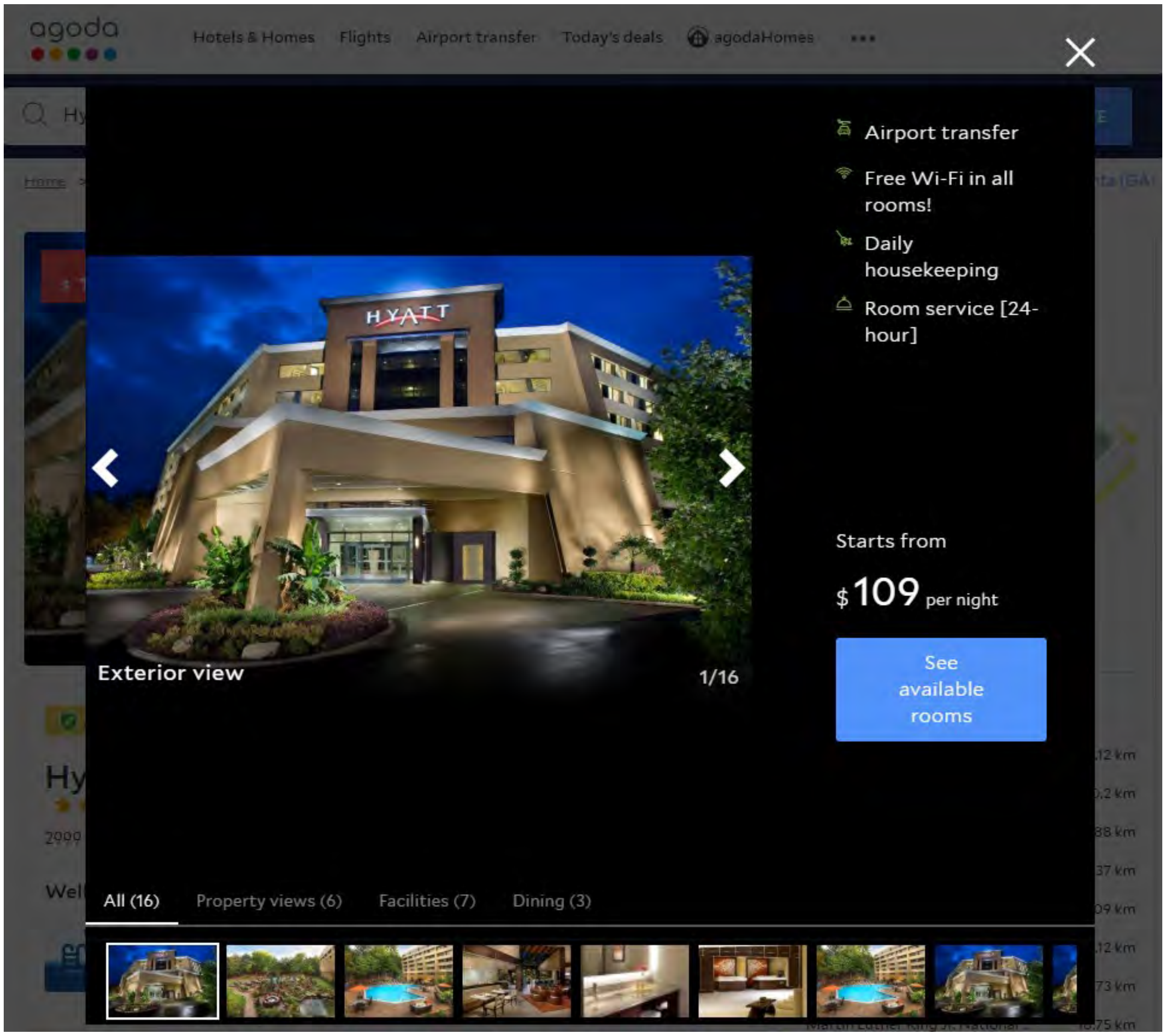


Hyatt Regency Suites Atlanta Northwest

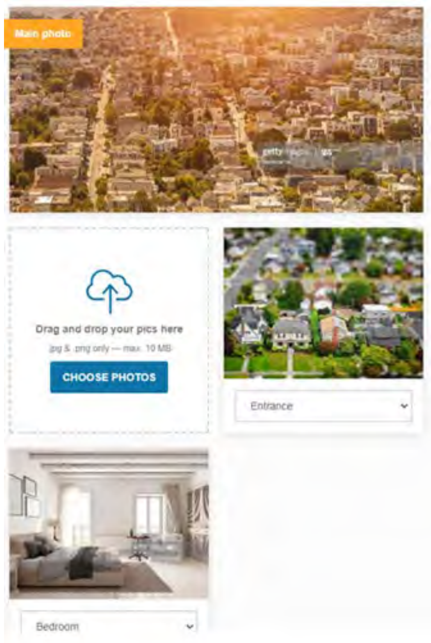
2999 Windy Hill Road, Heritage, Atlanta (GA), United States, 30067 - SEE MAP

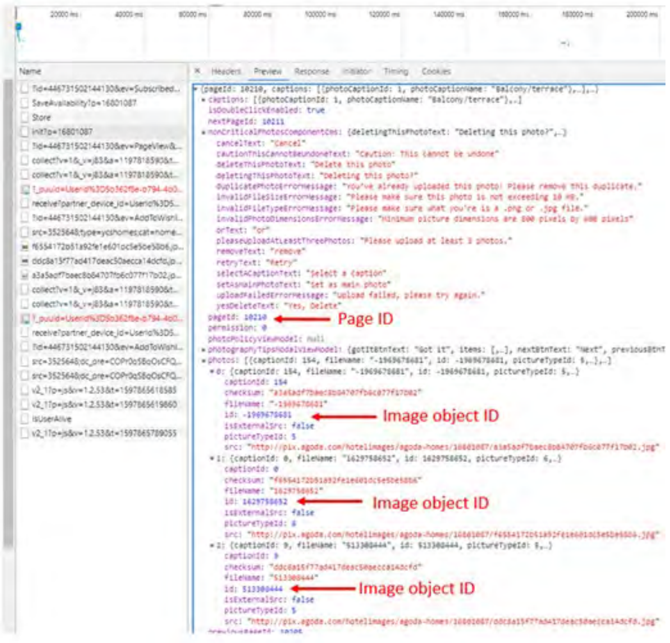
Popular landmarks

Atlanta History Center	10.12 km
Swan House	10.2 km
High Museum of Art	14.88 km
Piedmont Park	15.37 km


agoda
Hotels & Homes
Flights
Airport transfer
Today's deals
agodaHomes
Airport transfer
Free Wi-Fi in all rooms!
Daily housekeeping
Room service [24-hour]
HYATT
Exterior view
1/16
Starts from
$109 per night
See available rooms
All (16)
Property views (6)
Facilities (7)
Dining (3)

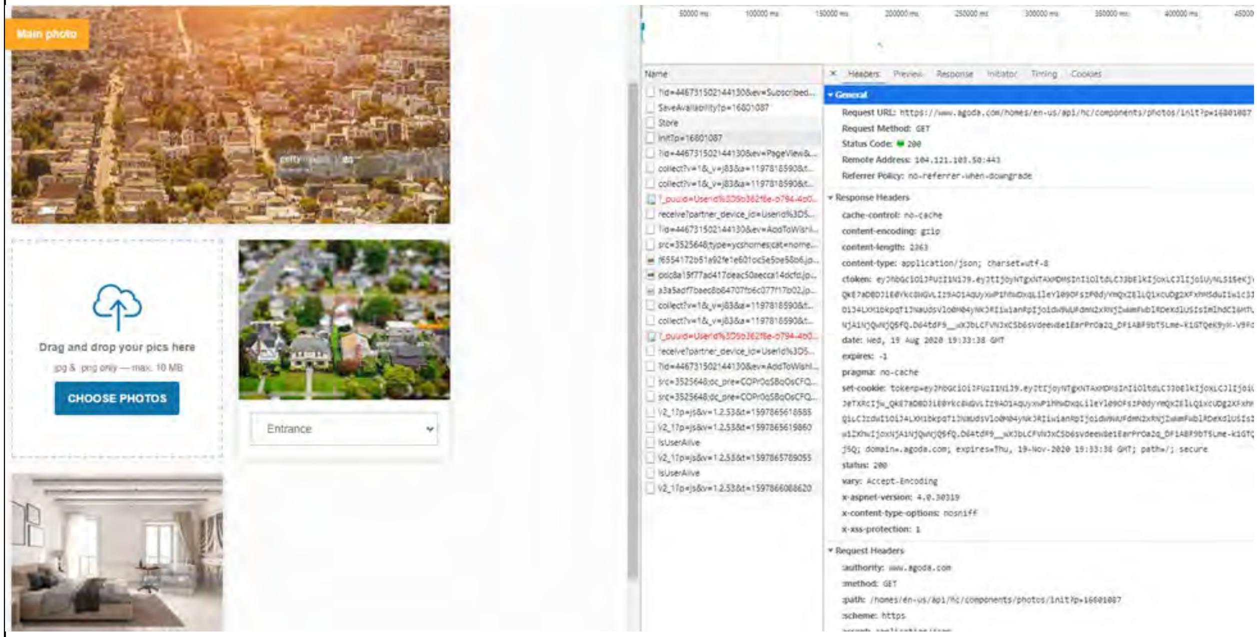
121. On information and belief, the YCS Platform is configured to produce a database with multidimensional array comprising the objects that comprise the website. Further, on information and belief, Priceline and Agoda websites are rendered using JSON dataset, which are multidimensional in nature. On information and belief, the JSON dataset for an object on a Priceline or Agoda website includes object style, object identifiers, and an indication of the web page that the object is part of. This data is provided to the YCS server, accessible to a web browser to generate a website based on user inputs. As shown in the example below, a user can upload photos for a listing property. The response is in the format of JSON dataset, in which the symbolic name "id" is the object identifier of each image object and symbolic name "pageId" is the indication of the web page that the object is part of. On information and belief, style settings are stored and transmitted in a similar manner.





```
▾ {pageId: 10210, captions: [{photoCaptionId: 1, photoCaptionName: "Balcony/terrace"},…],…}
  ▸ captions: [{photoCaptionId: 1, photoCaptionName: "Balcony/terrace"},…]
    isDoubleClickEnabled: true
    nextPageId: 10211
  ▾ nonCriticalPhotosComponentCms: {deletingThisPhotoText: "Deleting this photo?",…}
      cancelText: "Cancel"
      cautionThisCannotBeundoneText: "Caution: This cannot be undone"
      deleteThisPhotoText: "Delete this photo"
      deletingThisPhotoText: "Deleting this photo?"
      duplicatePhotoErrorMessage: "You've already uploaded this photo! Please remove this duplicate."
      invalidFileSizeErrorMessage: "Please make sure this photo is not exceeding 10 MB."
      invalidFileTypeErrorMessage: "Please make sure what you're is a .png or .jpg file."
      invalidPhotoDimensionsErrorMessage: "Minimum picture dimensions are 800 pixels by 600 pixels"
      orText: "or"
      pleaseUploadAtLeastThreePhotos: "Please upload at least 3 photos."
      removeText: "remove"
      retryText: "Retry"
      selectACaptionText: "Select a caption"
      setAsMainPhotoText: "Set as main photo"
      uploadFailedErrorMessage: "Upload failed, please try again."
      yesDeleteText: "Yes, Delete"
    pageId: 10210
    permission: 0
    photoPolicyViewModel: null
  ▸ photographyTipsModalViewModel: {gotItBtnText: "Got it", items: […], nextBtnText: "Next", previousBtnT
  ▾ photos: [{captionId: 154, fileName: "-1969678681", id: -1969678681, pictureTypeId: 5,…},…]
    ▾ 0: {captionId: 154, fileName: "-1969678681", id: -1969678681, pictureTypeId: 5,…}
        captionId: 154
        checksum: "a1a5adf7baec8b04707fb6c077f17082"
        fileName: "-1969678681"
        id: -1969678681
        isExternalSrc: false
        pictureTypeId: 5
        src: "http://pix.agoda.com/hotelimages/agoda-homes/16601087/a1a5adf7baec8b04707fb6c077f17082.jpg"
    ▾ 1: {captionId: 0, fileName: "1629758652", id: 1629758652, pictureTypeId: 6,…}
        captionId: 0
        checksum: "f65541720b1a92fe1e601dc5e5be5806"
        fileName: "1629758652"
        id: 1629758652
        isExternalSrc: false
        pictureTypeId: 8
        src: "http://pix.agoda.com/hotelimages/agoda-homes/16601087/f65541720b1a92fe1e601dc5e5be5806.jpg"
    ▾ 2: {captionId: 9, fileName: "513380444", id: 513380444, pictureTypeId: 5,…}
        captionId: 9
        checksum: "ddc8a15f77ad417deac50aecca14dcfd"
        fileName: "513380444"
        id: 513380444
        isExternalSrc: false
        pictureTypeId: 5
        src: "http://pix.agoda.com/hotelimages/agoda-homes/16601087/ddc8a15f77ad417deac50aecca14dcfd.jpg"
  previousPageId: 10208
```

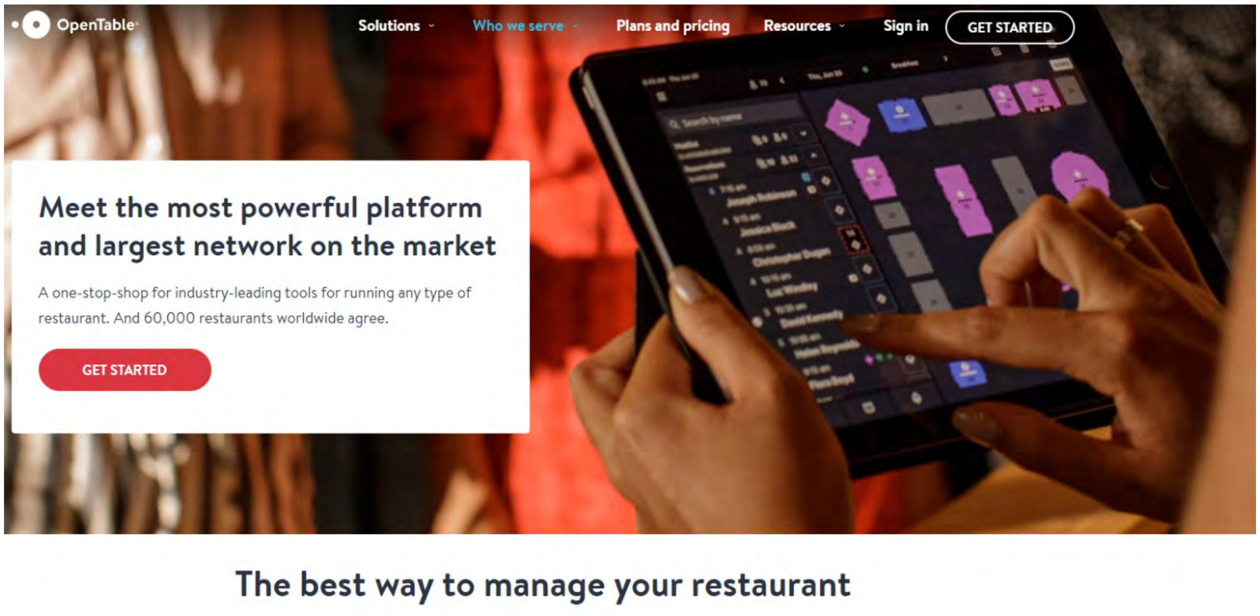
122. A web browser with access to a runtime engine can then generate the user configured website including the user input objects and style data extracted from the database. In the example below, image objects are fetched from the YCS Platform server and displayed on the YCS Platform with a simple HTTP GET request.



```
50000 ms   100000 ms   150000 ms   200000 ms   250000 ms   300000 ms   350000 ms   400000 ms   45000

Name
?id=446731502144130&ev=Subscribed...
SaveAvailability?p=16801087
Store
init?p=16801087
?id=446731502144130&ev=PageView&...
collect?v=1&_v=j83&a=1197818590&t...
collect?v=1&_v=j83&a=1197818590&t...
?_puuid=Userid%3D5b362f6e-b794-4b0...
receive?partner_device_id=Userid%3D5...
?id=446731502144130&ev=AddToWishl...
src=3525648;type=ycshomes;cat=home...
f6554172b51a92fe1e601dc5e5be58b6.jp...
bdc8a19f77ad417deac50aecca14dcfd.jp...
a3a5adf7baec8b84707fb6c077f17b02.jp...
collect?v=1&_v=j83&a=1197818590&t...
collect?v=1&_v=j83&a=1197818590&t...
?_puuid=Userid%3D5b362f6e-b794-4b0...
receive?partner_device_id=Userid%3D5...
?id=446731502144130&ev=AddToWishl...
src=3525648;dc_pre=COPr0oSBqOsCFQ...
src=3525648;dc_pre=COPr0oSBqOsCFQ...
v2_1?p=js&v=1.2.53&t=1597865618585
v2_1?p=js&v=1.2.53&t=1597865619860
isUserAlive
v2_1?p=js&v=1.2.53&t=1597865789055
isUserAlive
v2_1?p=js&v=1.2.53&t=1597866088620

×  Headers  Preview  Response  Initiator  Timing  Cookies
▾ General
    Request URL: https://www.agoda.com/homes/en-us/api/hc/components/photos/init?p=16801087
    Request Method: GET
    Status Code: ● 200
    Remote Address: 104.121.103.50:443
    Referrer Policy: no-referrer-when-downgrade
▾ Response Headers
    cache-control: no-cache
    content-encoding: gzip
    content-length: 2363
    content-type: application/json; charset=utf-8
    ctoken: [illegible]
    date: Wed, 19 Aug 2020 19:33:38 GMT
    expires: -1
    pragma: no-cache
    set-cookie: [illegible]
    j5Q; domain=.agoda.com; expires=Thu, 19-Nov-2020 19:33:38 GMT; path=/; secure
    status: 200
    vary: Accept-Encoding
    x-aspnet-version: 4.0.30319
    x-content-type-options: nosniff
    x-xss-protection: 1
▾ Request Headers
    :authority: www.agoda.com
    :method: GET
    :path: /homes/en-us/api/hc/components/photos/init?p=16801087
    :scheme: https
```

The OpenTable Platform:

123. OpenTable infringes at least claim 1 of the '168 patent through a combination of features in the OpenTable Platform that collectively practice each limitation of claim 1. By way of example, OpenTable provides the OpenTable Platform, a browser-based platform for, *inter alia*, creating restaurant profile websites.



Meet the most powerful platform and largest network on the market

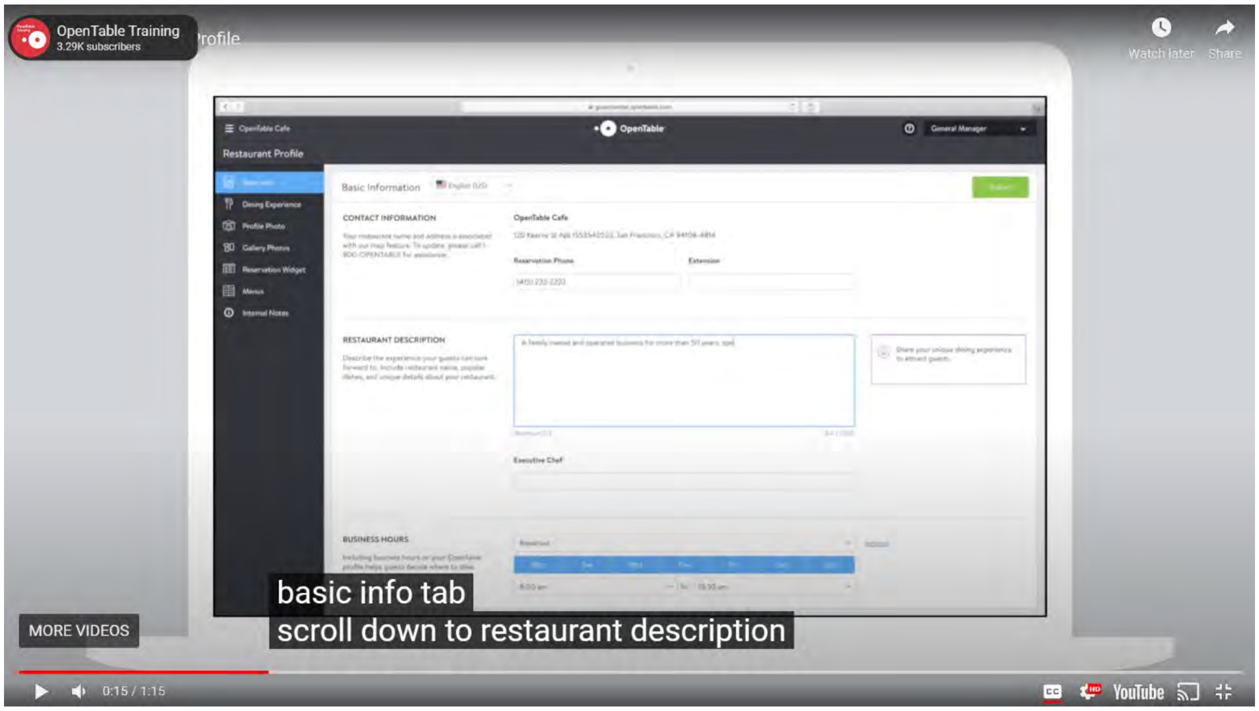
A one-stop-shop for industry-leading tools for running any type of restaurant. And 60,000 restaurants worldwide agree.

The best way to manage your restaurant

https://restaurant.opentable.com/why-opentable/

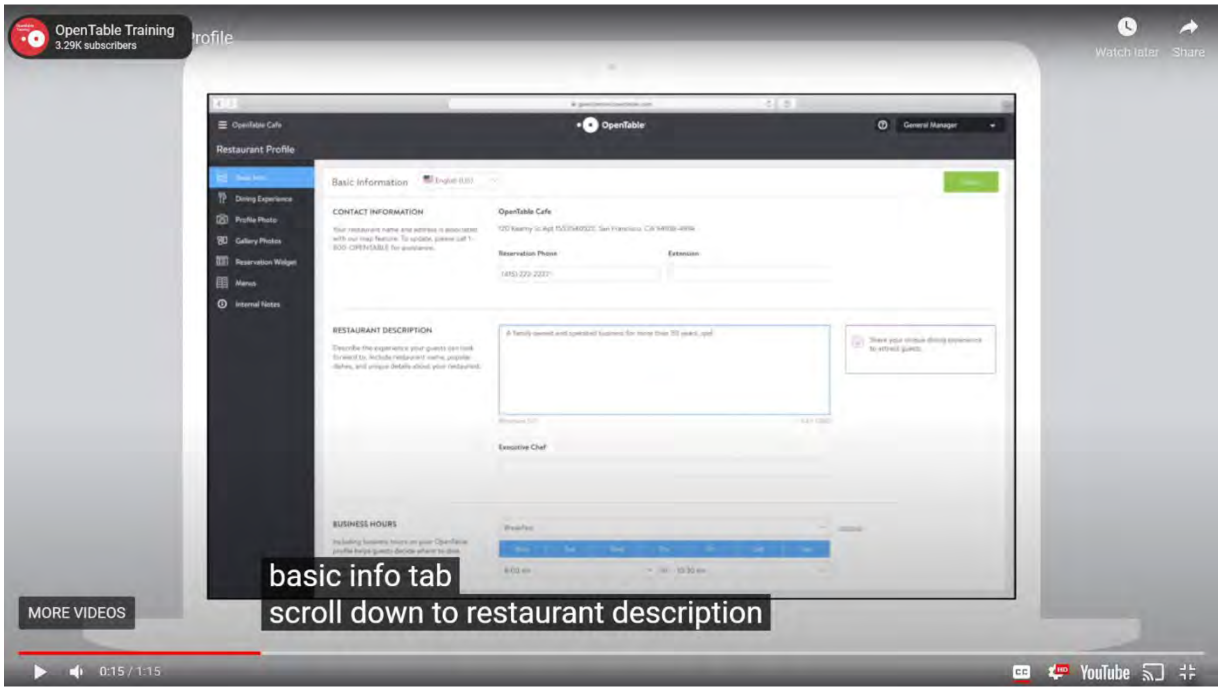
125. On information and belief, the OpenTable Platform comprises a server that serves a backend server to retrieve and store data, including data relating to user input data, for the OpenTable Platform and restaurant profile websites created thereon.

126. The OpenTable Platform's server comprises a build engine for creating and configuring restaurant profile websites.



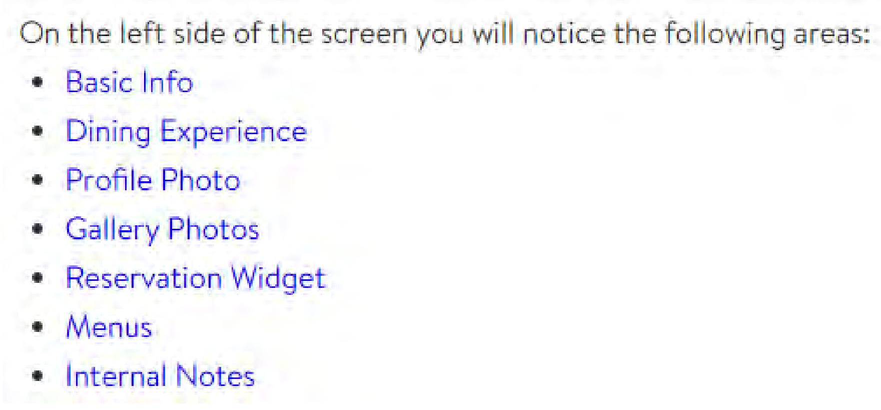
https://support.opentable.com/s/article/Updating-your-restaurant-profile-in-GuestCenter?language=en_US (video: https://youtu.be/eQVviz7sYnQ)

127. The OpenTable Platform's build engine is configured to accept user input to create a website comprising a plurality of web pages that each include objects. User input for a website may include basic information of the restaurant, dining experience, profile photo of the restaurant, reservation widget and menus for the restaurant profile websites.



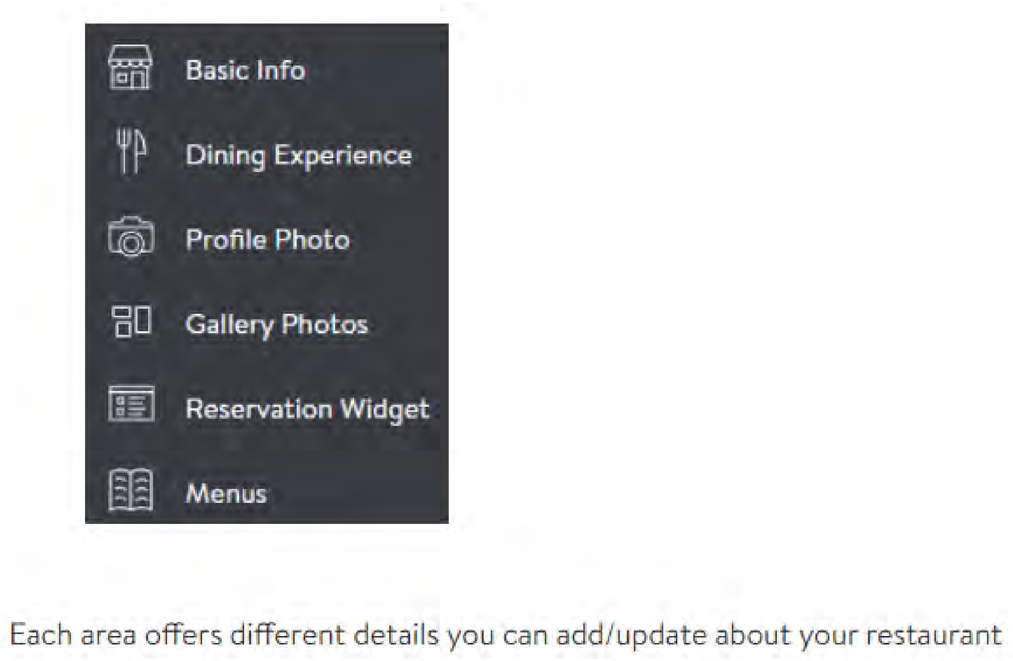
basic info tab
scroll down to restaurant description

https://support.opentable.com/s/article/Updating-your-restaurant-profile-in-GuestCenter?language=en_US (video: https://youtu.be/eQVviz7sYnQ)



On the left side of the screen you will notice the following areas:

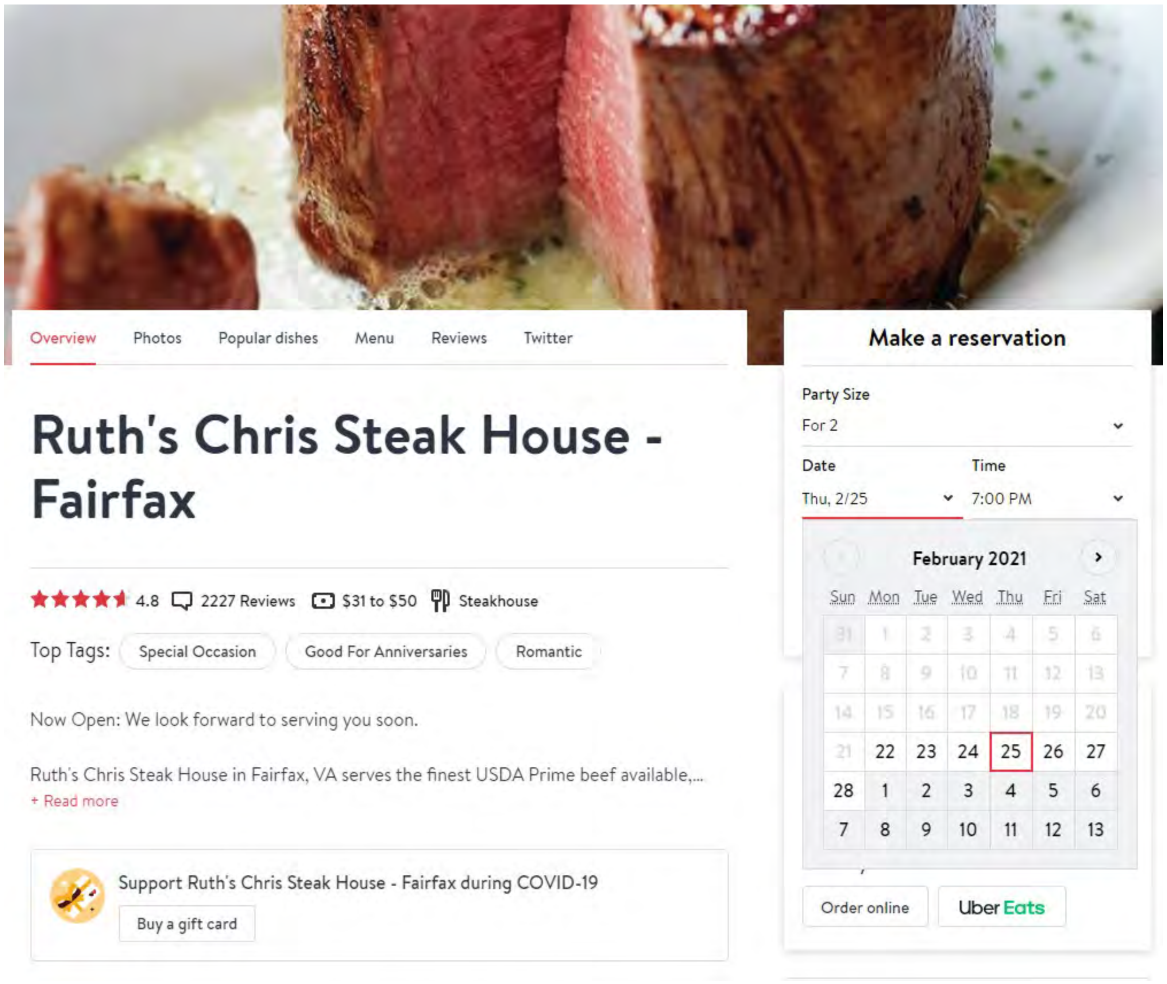
- Basic Info
- Dining Experience
- Profile Photo
- Gallery Photos
- Reservation Widget
- Menus
- Internal Notes



Each area offers different details you can add/update about your restaurant

https://support.opentable.com/s/article/Updating-your-restaurant-profile-in-GuestCenter?language=en_US

128. Restaurant profile websites created with the OpenTable Platform's build engine comprise objects, such as image, drop-down list, and date picker objects, as shown in the example below.



Ruth's Chris Steak House - Fairfax

4.8 2227 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Romantic

Now Open: We look forward to serving you soon.

Ruth's Chris Steak House in Fairfax, VA serves the finest USDA Prime beef available,...
+ Read more

Support Ruth's Chris Steak House - Fairfax during COVID-19
Buy a gift card

Make a reservation

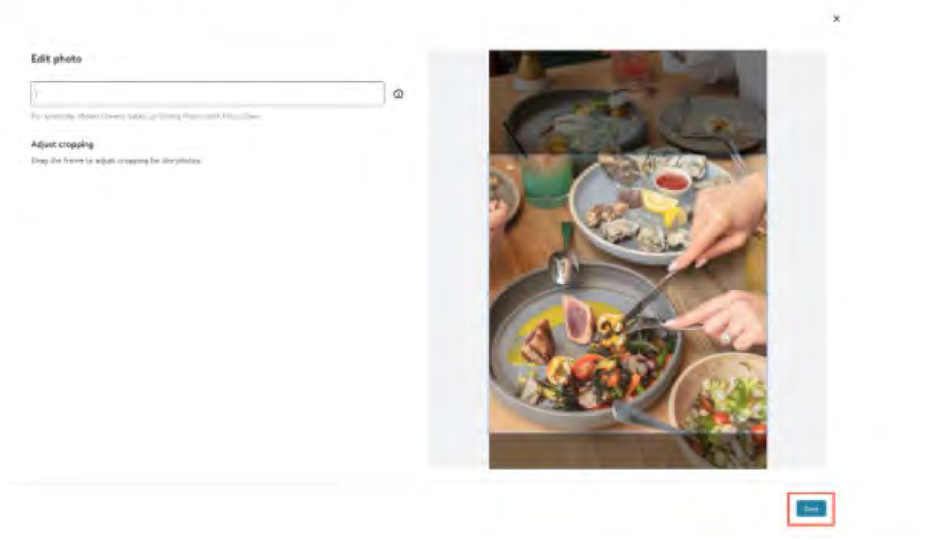
Party Size: For 2
Date: Thu, 2/25
Time: 7:00 PM

February 2021

Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	1	2	3	4	5	6
7	8	9	10	11	12	13

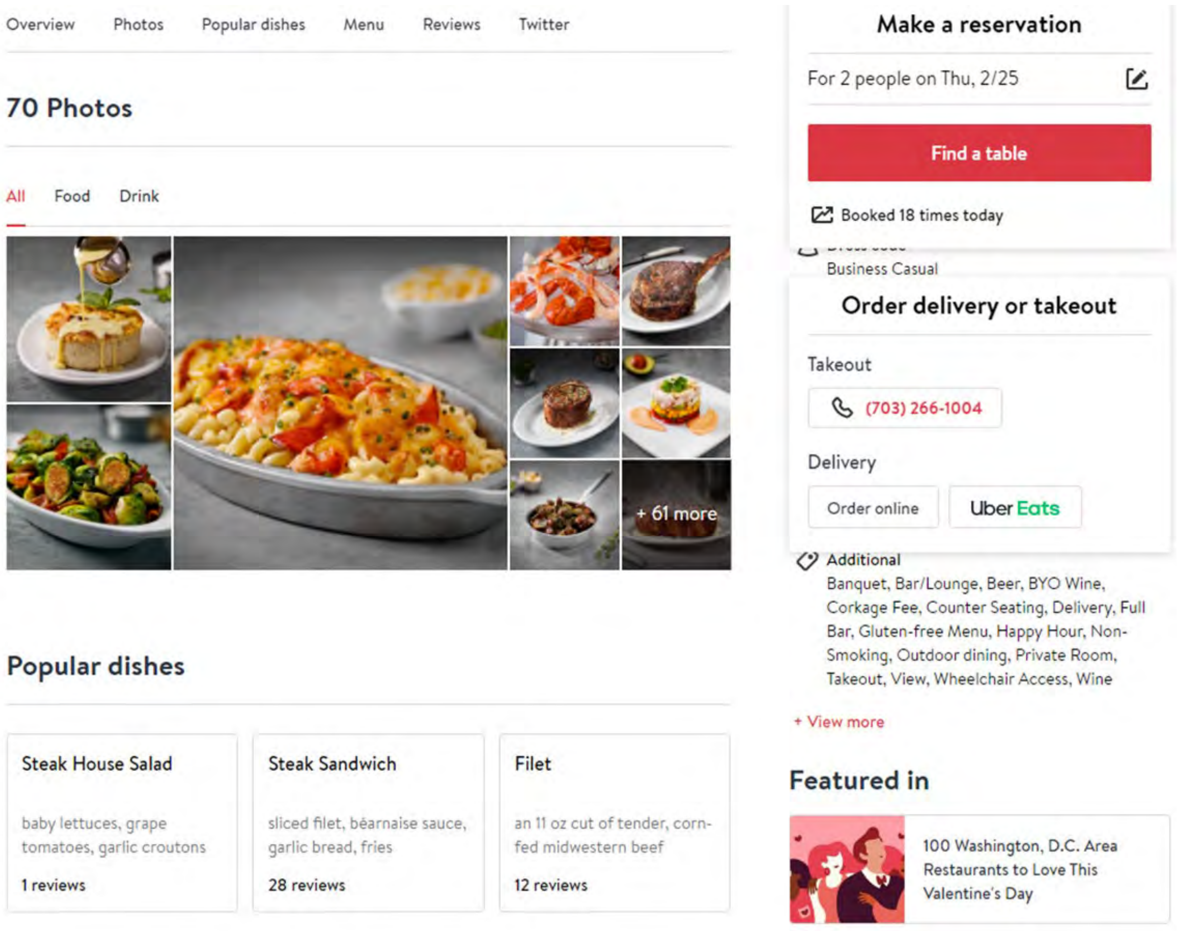
129. The OpenTable Platform's build engine is configured to accept user inputs to associate a style with image objects for a photo gallery of the restaurant profile. By way of example, images of the property can be selected to be displayed on the restaurant profile and the OpenTable Platform's build engine accepts user inputs for the titles of the images and adjusted cropping of the displayed images. Upon selection of an image and other inputs, the OpenTable Platform's build engine can associate a particular style with the image object.

- Here you can adjust the cropping of the photo and add a title to best describe the image. When you're finished, click **Save**.



https://support.opentable.com/s/article/How-to-update-photos-on-OpenTable-com-profile?language=en_US

130. Each web page includes at least one button object or at least one image object. For example, a restaurant profile web page can include image objects for a photo gallery of the restaurant and a "Find a table" button object permitting a visitor to make a reservation.


Overview Photos Popular dishes Menu Reviews Twitter

70 Photos

All Food Drink

+ 61 more

Popular dishes

Steak House Salad
baby lettuces, grape tomatoes, garlic croutons
1 reviews

Steak Sandwich
sliced filet, béarnaise sauce, garlic bread, fries
28 reviews

Filet
an 11 oz cut of tender, corn-fed midwestern beef
12 reviews

Make a reservation
For 2 people on Thu, 2/25
Find a table
Booked 18 times today

Business Casual

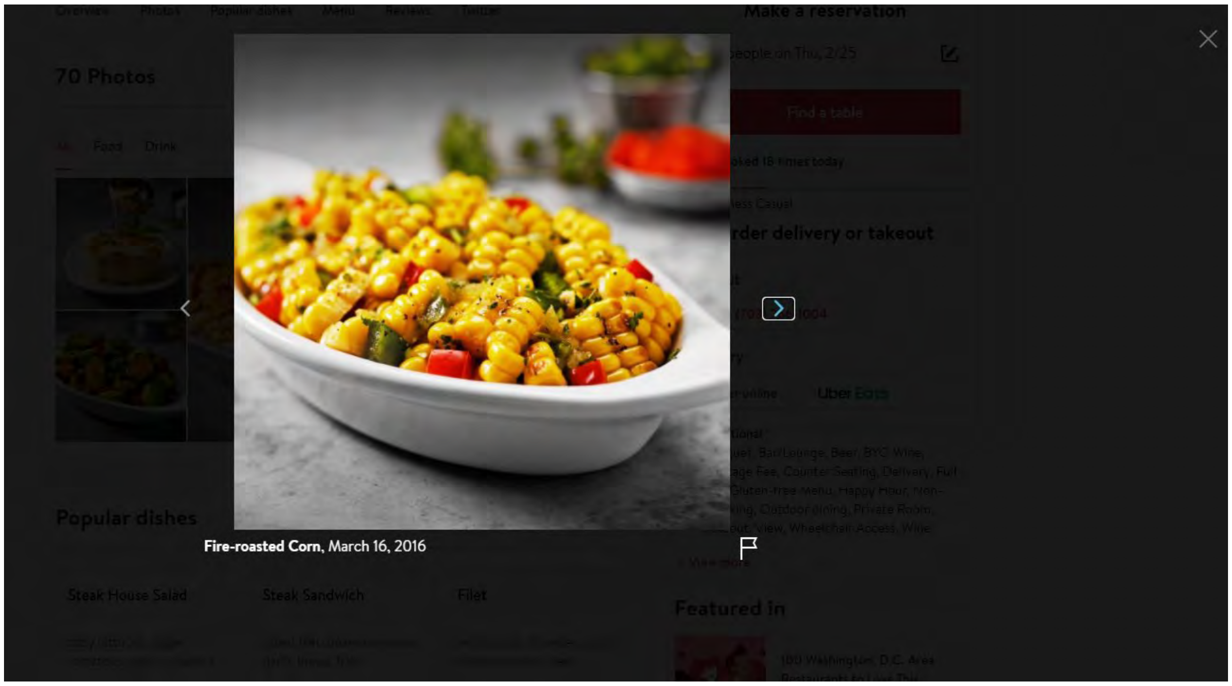
Order delivery or takeout
Takeout
(703) 266-1004
Delivery
Order online
Uber Eats

Additional
Banquet, Bar/Lounge, Beer, BYO Wine, Corkage Fee, Counter Seating, Delivery, Full Bar, Gluten-free Menu, Happy Hour, Non-Smoking, Outdoor dining, Private Room, Takeout, View, Wheelchair Access, Wine

+ View more

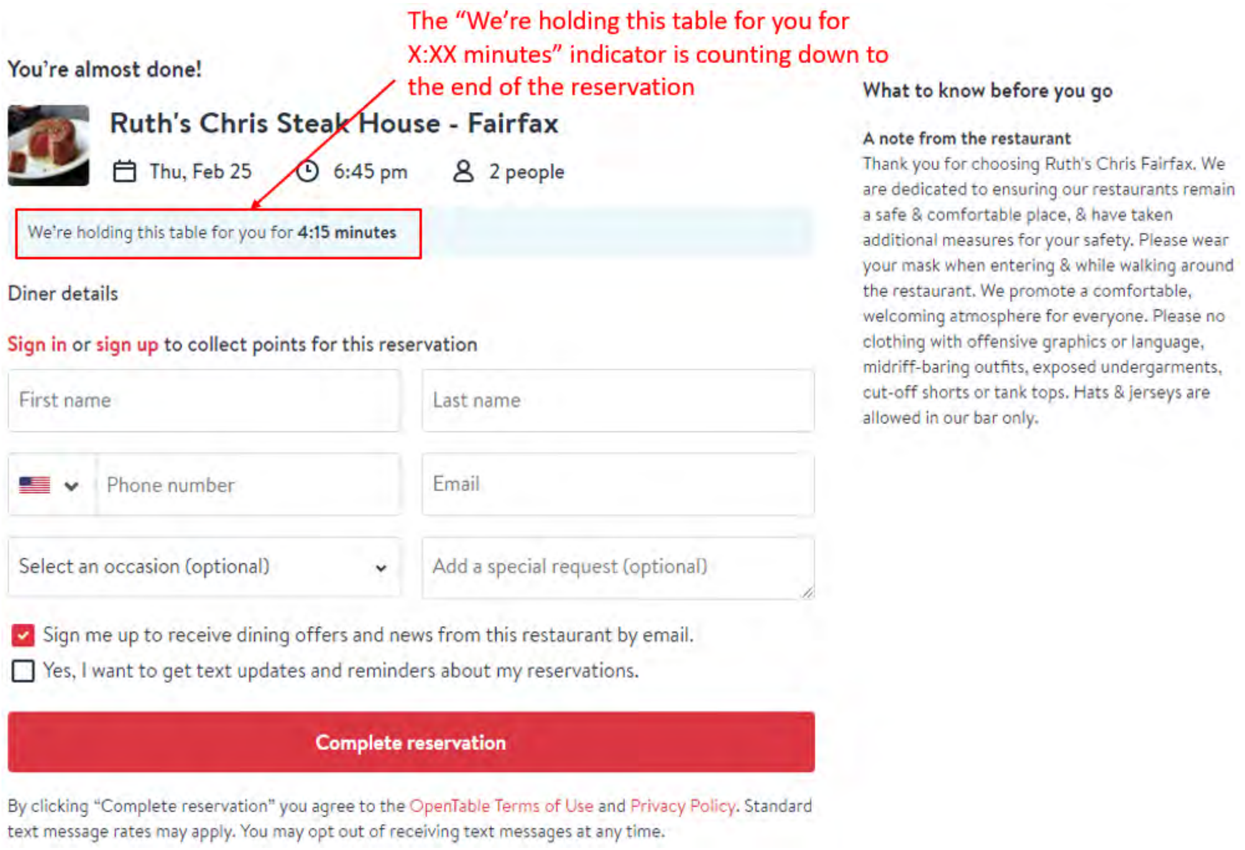
Featured in
100 Washington, D.C. Area Restaurants to Love This Valentine's Day

146. An image object is associated with styles defining transformations and timelines for the image. By way of another example, when the forward arrow on an enlarged image is clicked on, the current full-screen image is transformed into the next full-screen image of the photo gallery, providing a photo slideshow with the selected image.



https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8eae-7ada9d88b68e&corrid=dfc6cee2-c8f8-4054-8eae-7ada9d88b68e&avt=eyJ2IjoyLCJtIjoxLCJwIjowLCJzIjowLCJuIjowfQ

147. By way of another example, an indicator counts down to the end of the reservation (here, showing 4:15 minutes left), where the counting down value defines the timelines for the "Complete reservation" button object.



The "We're holding this table for you for X:XX minutes" indicator is counting down to the end of the reservation

You're almost done!

Ruth's Chris Steak House - Fairfax

Thu, Feb 25 6:45 pm 2 people

We're holding this table for you for **4:15 minutes**

Diner details

Sign in or sign up to collect points for this reservation

First name | Last name

Phone number | Email

Select an occasion (optional) | Add a special request (optional)

[✓] Sign me up to receive dining offers and news from this restaurant by email.

[ ] Yes, I want to get text updates and reminders about my reservations.

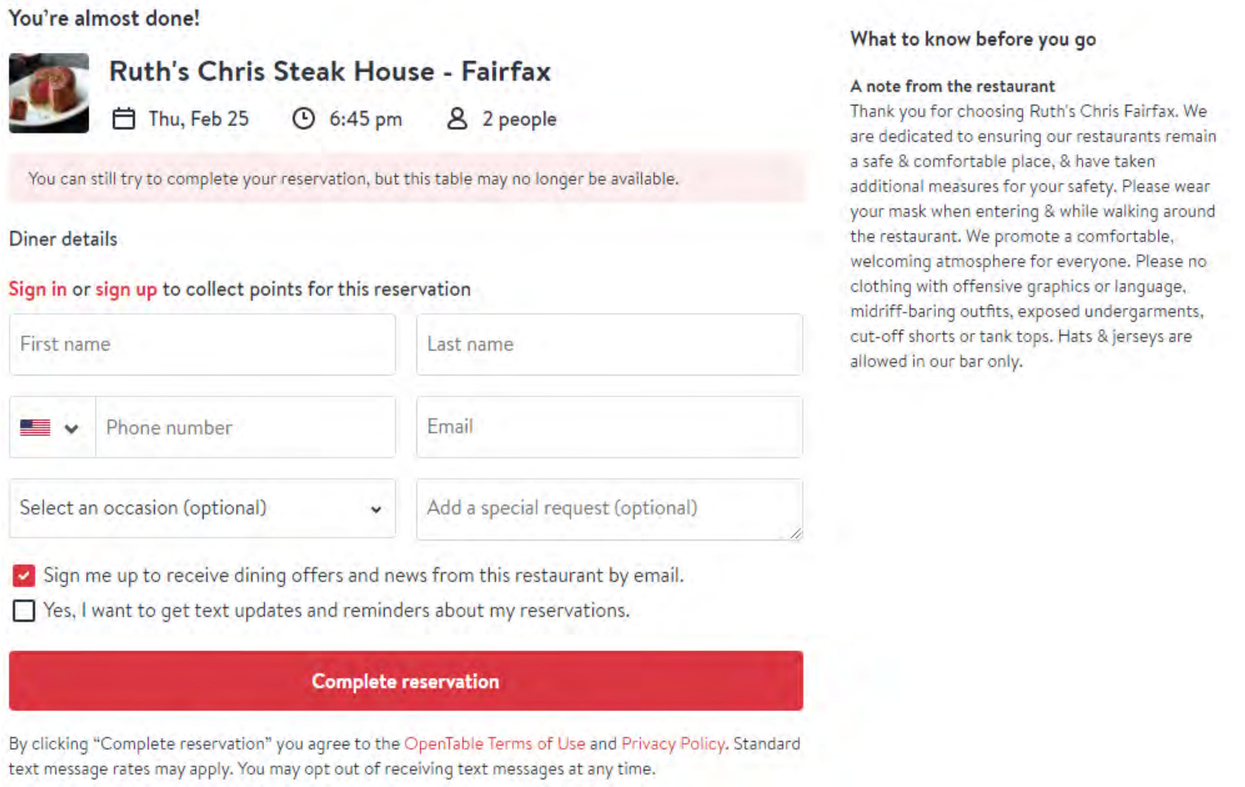
Complete reservation

By clicking "Complete reservation" you agree to the OpenTable Terms of Use and Privacy Policy. Standard text message rates may apply. You may opt out of receiving text messages at any time.

What to know before you go

A note from the restaurant

Thank you for choosing Ruth's Chris Fairfax. We are dedicated to ensuring our restaurants remain a safe & comfortable place, & have taken additional measures for your safety. Please wear your mask when entering & while walking around the restaurant. We promote a comfortable, welcoming atmosphere for everyone. Please no clothing with offensive graphics or language, midriff-baring outfits, exposed undergarments, cut-off shorts or tank tops. Hats & jerseys are allowed in our bar only.



You're almost done!

Ruth's Chris Steak House - Fairfax

Thu, Feb 25 6:45 pm 2 people

You can still try to complete your reservation, but this table may no longer be available.

Diner details

Sign in or sign up to collect points for this reservation

First name | Last name

Phone number | Email

Select an occasion (optional) | Add a special request (optional)

[✓] Sign me up to receive dining offers and news from this restaurant by email.

[ ] Yes, I want to get text updates and reminders about my reservations.

Complete reservation

By clicking "Complete reservation" you agree to the OpenTable Terms of Use and Privacy Policy. Standard text message rates may apply. You may opt out of receiving text messages at any time.

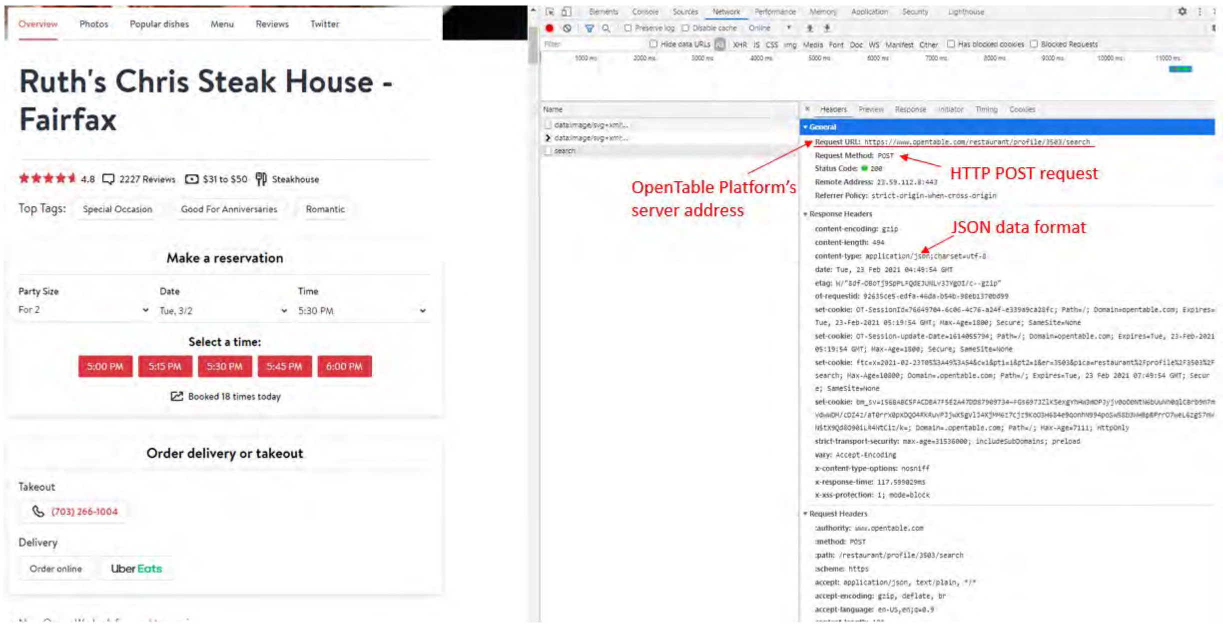
What to know before you go

A note from the restaurant

Thank you for choosing Ruth's Chris Fairfax. We are dedicated to ensuring our restaurants remain a safe & comfortable place, & have taken additional measures for your safety. Please wear your mask when entering & while walking around the restaurant. We promote a comfortable, welcoming atmosphere for everyone. Please no clothing with offensive graphics or language, midriff-baring outfits, exposed undergarments, cut-off shorts or tank tops. Hats & jerseys are allowed in our bar only.

160. Each web page for a restaurant profile created through the OpenTable Platform is defined by the objects and styles comprising that web page, such as the image objects and styles described above.

161. On information and belief, the OpenTable Platform's build engine is configured to produce a database with a multidimensional array comprising the objects that comprise the website. As described above, inputs for the restaurant profile websites are stored in the OpenTable Platform's database, which comprises multidimensional arrays. In the example below, the OpenTable Platform sends the user selected settings for a reservation, such as party size, reservation date and time through a "search" HTTP POST request to the OpenTable Platform's server and stores the information representative of the said user selected settings in an OpenTable Platform's database. In particular, the user selected settings are transferred and stored in JSON data format.

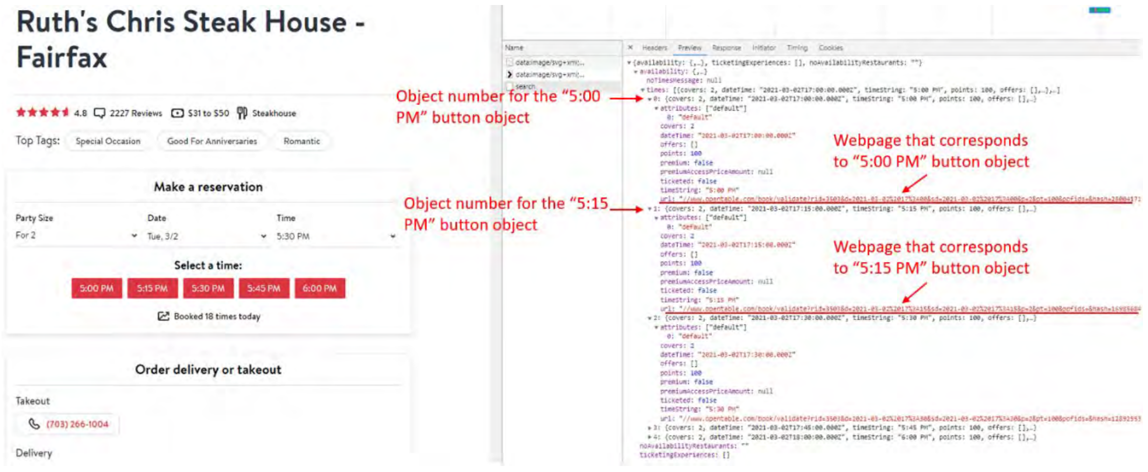


[redacted]

```
▾ General
Request URL: https://www.opentable.com/restaurant/profile/3503/search
Request Method: POST
Status Code: ● 200
Remote Address: 23.59.112.8:443
Referrer Policy: strict-origin-when-cross-origin
▾ Response Headers
content-encoding: gzip
content-length: 494
content-type: application/json;charset=utf-8
date: Tue, 23 Feb 2021 04:49:54 GMT
etag: W/"8df-OBoTj9SpPL+QdEJUNLv3JYgOI/c--gzip"
ot-requestid: 92635ce5-edfa-46da-b54b-98eb1370bd99
set-cookie: OT-SessionId=76649704-6c06-4c76-a24f-e339a9ca28fc; Path=/; Domain=opentable.com; Expires=
Tue, 23-Feb-2021 05:19:54 GMT; Max-Age=1800; Secure; SameSite=None
set-cookie: OT-Session-Update-Date=1614055794; Path=/; Domain=opentable.com; Expires=Tue, 23-Feb-2021
05:19:54 GMT; Max-Age=1800; Secure; SameSite=None
set-cookie: ftc=x=2021-02-23T05%3A49%3A54&c=1&pt1=1&pt2=1&er=3503&p1ca=restaurant%2Fprofile%2F3503%2F
search; Max-Age=10800; Domain=.opentable.com; Path=/; Expires=Tue, 23 Feb 2021 07:49:54 GMT; Secur
e; SameSite=None
set-cookie: bm_sv=156BABC5FACD8A7F5E2A47DD87909734~FGs697JZlKSexgYhHW3mDPJyjvOoO0NtN6bUuVh0qlCBrb9n7m
VdwWDH/cDZ4z/aT0rrX0pXDQO4RX4uVPJjwX5gVlJ4XjMM6z7CjZ9KoO3H6B4e9QonhN994poSW58bJWM8p8Frr07weL6zgS7nW
N5tX9Qd8O901LR4NtCiz/k=; Domain=.opentable.com; Path=/; Max-Age=7111; HttpOnly
strict-transport-security: max-age=31536000; includeSubDomains; preload
vary: Accept-Encoding
x-content-type-options: nosniff
x-response-time: 117.599829ms
x-xss-protection: 1; mode=block
▾ Request Headers
:authority: www.opentable.com
:method: POST
:path: /restaurant/profile/3503/search
:scheme: https
accept: application/json, text/plain, */*
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
```

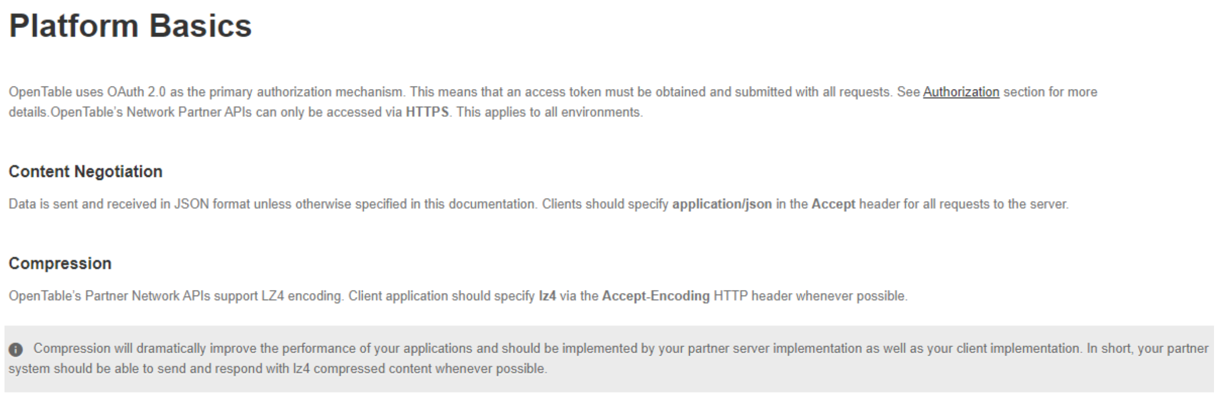
162. Further, the restaurant profile websites are rendered with data in JSON format, which is multidimensional in nature. In particular, each website object rendered with JSON dataset includes object styles, object identifiers, and an indication of the web page that the object is part of. For example, five button UI objects can be used to select a reservation time, and user selection of one of the five button UI objects is stored in a multidimensional array, e.g., JSON dataset,

including data defining, for each button object, the object style, an object number, and an indication of the web page that each object is part of. For example, "0" is the object number for the "5:00 PM" button object and symbolic name "url" associates with an indication (i.e., URL address) of the web page that each button object is part of. In addition, the JSON dataset also includes data defining, for each button object, the object style, such as the "attributes" is set to "default".


Ruth's Chris Steak House - Fairfax
4.8 2227 Reviews $31 to $50 Steakhouse
Top Tags: Special Occasion Good For Anniversaries Romantic
Make a reservation
Party Size For 2
Date Tue, 3/2
Time 5:30 PM
Select a time:
5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM
Booked 18 times today
Order delivery or takeout
Takeout
(703) 266-1004
Delivery
Object number for the "5:00 PM" button object
Object number for the "5:15 PM" button object
Webpage that corresponds to "5:00 PM" button object
Webpage that corresponds to "5:15 PM" button object
Headers Preview Response Initiator Timing Cookies

```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
    noTimesMessage: null
   ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
    ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:00:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:00 PM"
      url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A00&sd=2021-03-02%2017%3A00&p=2&pt=100&pofids=&hash=2800417
    ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:15:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:15 PM"
      url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A15&sd=2021-03-02%2017%3A15&p=2&pt=100&pofids=&hash=1698568
    ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:30:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:30 PM"
      url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A30&sd=2021-03-02%2017%3A30&p=2&pt=100&pofids=&hash=1189255
    ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
    ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```

163. On information and belief, the OpenTable Platform's build engine stores the data relating to the inputs and settings for the restaurant profile websites in the database and provides the date to the OpenTable Platform's server which is accessible to a web browser to generate a restaurant profile website including the inputs and settings. In particular, data is sent and received in JSON format between the OpenTable Platform's server and the web browser of a client device.



Platform Basics

OpenTable uses OAuth 2.0 as the primary authorization mechanism. This means that an access token must be obtained and submitted with all requests. See Authorization section for more details. OpenTable's Network Partner APIs can only be accessed via **HTTPS**. This applies to all environments.

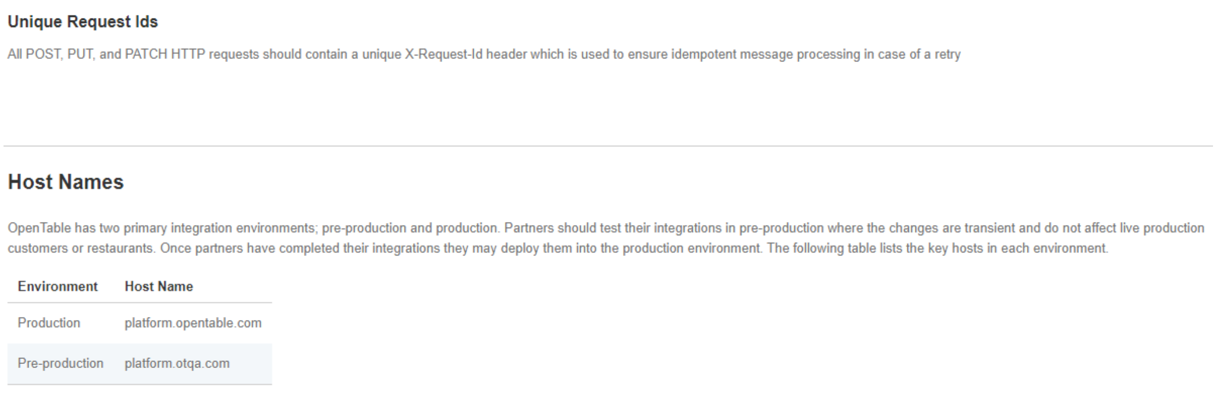
Content Negotiation

Data is sent and received in JSON format unless otherwise specified in this documentation. Clients should specify **application/json** in the **Accept** header for all requests to the server.

Compression

OpenTable's Partner Network APIs support LZ4 encoding. Client application should specify **lz4** via the **Accept-Encoding** HTTP header whenever possible.

Compression will dramatically improve the performance of your applications and should be implemented by your partner server implementation as well as your client implementation. In short, your partner system should be able to send and respond with lz4 compressed content whenever possible.



Unique Request Ids

All POST, PUT, and PATCH HTTP requests should contain a unique X-Request-Id header which is used to ensure idempotent message processing in case of a retry

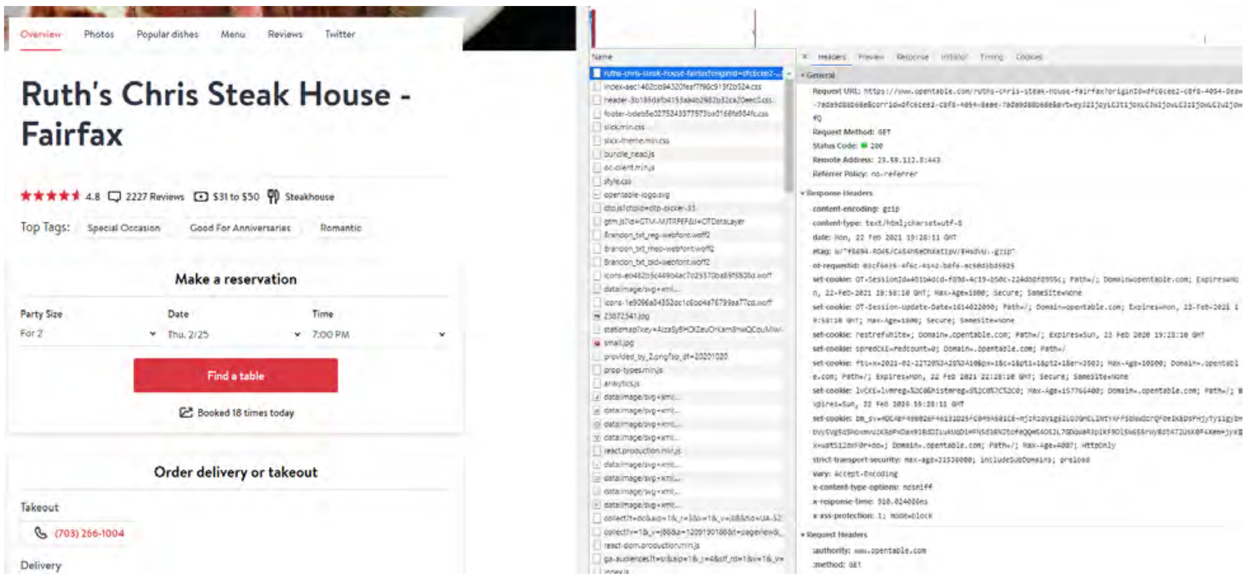
Host Names

OpenTable has two primary integration environments; pre-production and production. Partners should test their integrations in pre-production where the changes are transient and do not affect live production customers or restaurants. Once partners have completed their integrations they may deploy them into the production environment. The following table lists the key hosts in each environment.

Environment	Host Name
Production	platform.opentable.com
Pre-production	platform.otqa.com

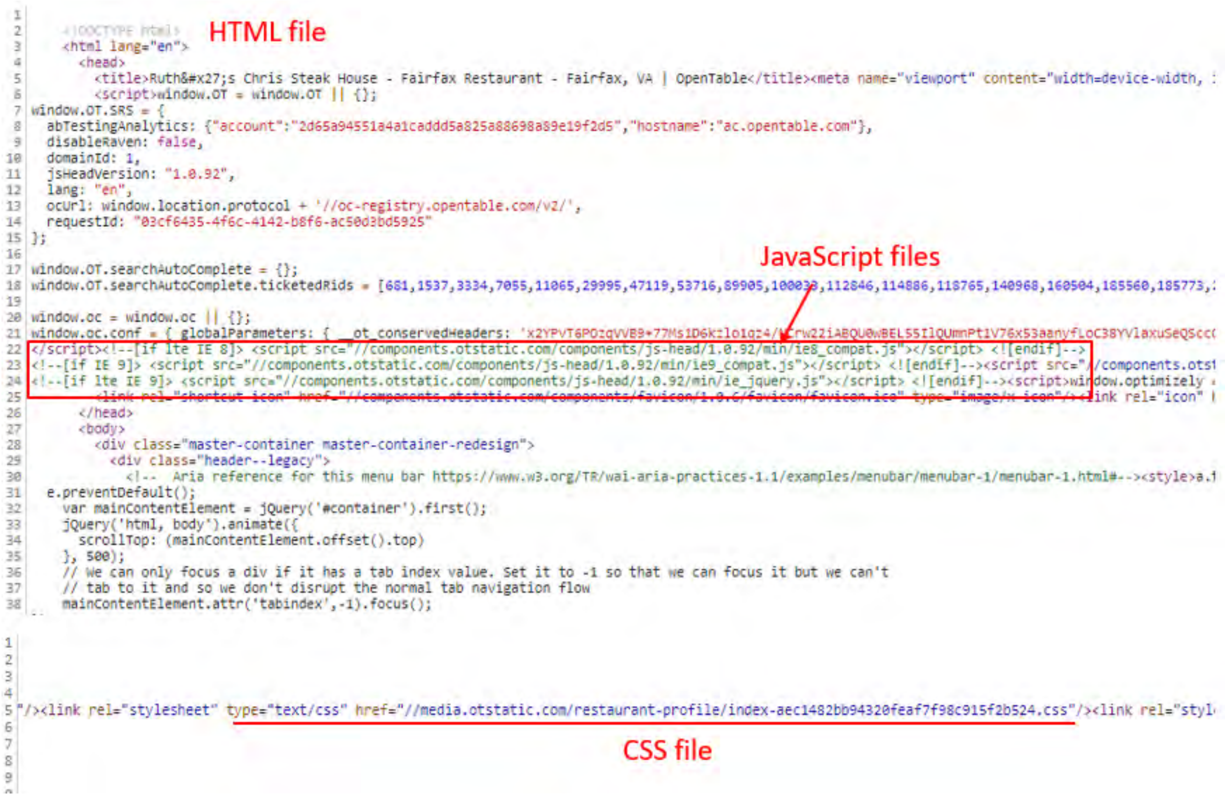
https://platform.opentable.com/documentation/?shell#platform-basics

164. The OpenTable Platform's database is produced such that a web browser with access to a runtime engine can generate the restaurant profile websites including the input objects and style data extracted from the database. In the example below, the OpenTable Platform downloads an HTML file of a restaurant profile website from the OpenTable Platform's server with an HTTP GET request. When the OpenTable Platform parses the HTML file through a browser build engine, the OpenTable Platform makes a request (e.g., GET method) through a web browser to fetch the embedded files such as CSS and Javascript run time files, which are processed with a runtime engine.



```
ruths-chris-steak-house-fairfax?originId=dfc6cee2-...
index-aec1462bb94320feaf7f90c915f2b524.css
header-3b185daf54153ab4b2982b32ca20eec0.css
footer-bdeb5e0275243377573ba0166fa984fc.css
slick.min.css
slick-theme.min.css
bundle_head.js
oc-client.min.js
style.css
opentable-logo.svg
dtp.js?dtpid=dtp-picker-33
gtm.js?id=GTM-MJTRPEF&l=OTDataLayer
Brandon_txt_reg-webfont.woff2
Brandon_txt_med-webfont.woff2
Brandon_txt_bld-webfont.woff2
icons-e0482b5c449b4ac7c25370ba89f5808d.woff
data:image/svg+xml...
icons-1e9096a34352dc1c6bd4a76799aa77cd.woff
23872341.jpg
staticmap?key=AIzaSyBHOXZeuOrKam8rwQCouMiw-
small.jpg
provided_by_2.png?sp_dt=20201020
prop-types.min.js
analytics.js
data:image/svg+xml...
data:image/svg+xml...
data:image/svg+xml...
data:image/svg+xml...
react.production.min.js
data:image/svg+xml...
data:image/svg+xml...
data:image/svg+xml...
data:image/svg+xml...
collect?t=dc&aip=1&_r=3&v=1&_v=j88&tid=UA-52
collect?v=1&_v=j88&a=1209190188&t=pageview&_
react-dom.production.min.js
ga-audiences?t=sr&aip=1&_r=4&slf_rd=1&v=1&_v=
index.js
```

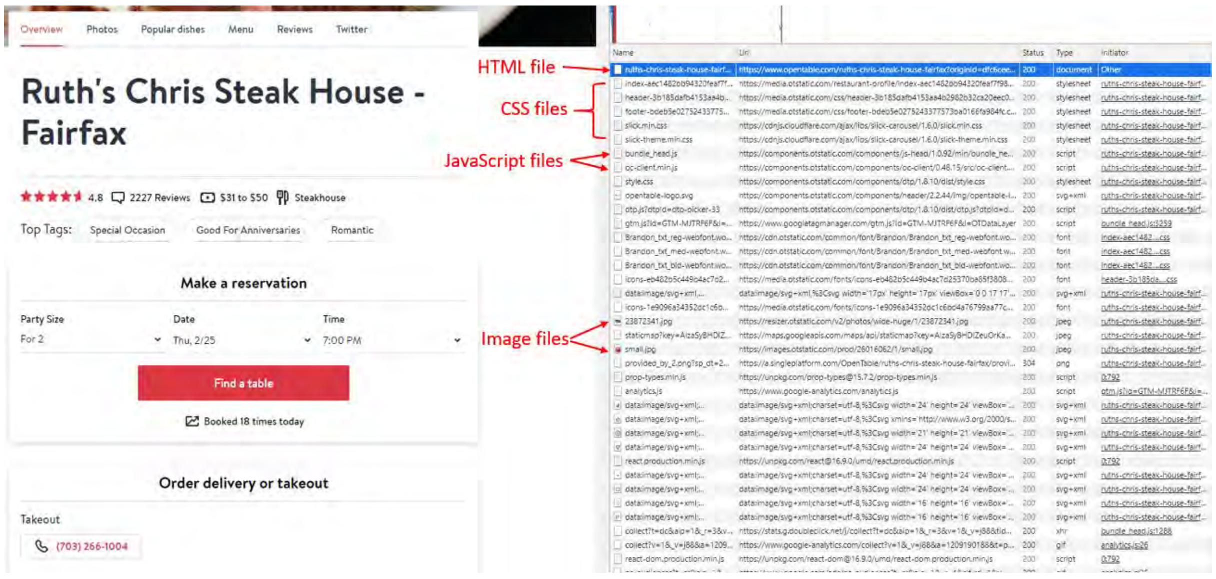
```
▾ General
  Request URL: https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8ea
  -7ada9d88b68e&corrid=dfc6cee2-c8f8-4054-8ee-7ada9d88b68e&avt=eyJ2IjoyLCJtIjoxLCJwIjowLCJzIjowLCJuIjow
  fQ
  Request Method: GET
  Status Code: ● 200
  Remote Address: 23.59.112.8:443
  Referrer Policy: no-referrer
▾ Response Headers
  content-encoding: gzip
  content-type: text/html;charset=utf-8
  date: Mon, 22 Feb 2021 19:28:11 GMT
  etag: W/"f8694-RO45/CAS4n6eOhkatIpV/EHsdVU.-gzip"
  ot-requestid: e3cf6435-4f6c-4142-b8f6-ac58d3bd5925
  set-cookie: OT-SessionId=401b4dcd-f89d-4c19-b50c-224d80f0955c; Path=/; Domain=opentable.com; Expires=Mo
  n, 22-Feb-2021 19:58:10 GMT; Max-Age=1800; Secure; SameSite=None
  set-cookie: OT-Session-Update-Date=1614022090; Path=/; Domain=opentable.com; Expires=Mon, 22-Feb-2021 1
  9:58:10 GMT; Max-Age=1800; Secure; SameSite=None
  set-cookie: restrefwhite=; Domain=.opentable.com; Path=/; Expires=Sun, 23 Feb 2020 19:28:10 GMT
  set-cookie: spredCKI=redcount=0; Domain=.opentable.com; Path=/
  set-cookie: ftc=x=2021-02-22T20%3A28%3A10&px=1&c=1&pt1=1&pt2=1&er=3503; Max-Age=10800; Domain=.opentabl
  e.com; Path=/; Expires=Mon, 22 Feb 2021 22:28:10 GMT; Secure; SameSite=None
  set-cookie: lvCKI=lvmreg=%2C0&histmreg=0%2C0%7C%2C0; Max-Age=157766400; Domain=.opentable.com; Path=/; E
  xpires=Sun, 22 Feb 2026 19:28:11 GMT
  set-cookie: bm_sv=4DEABF488026F46131D25FC0494581CE~mjZRzoV1g8IiO70MCLINTYAFF5DbwDzrQFOeIkEDsPHjyTy1igb=
  bVy5Vg5q5HoxmvUzK8dPxDax91BdDIuAUdDi#FN5d38N2to#eQQm54O5JL7GDqWaR3p1kF9Ol5W6S6rWyBdT472UsK0F4Xem+jyX
  X+u8tS12Bxf0r+0o=; Domain=.opentable.com; Path=/; Max-Age=4007; HttpOnly
  strict-transport-security: max-age=31536000; includeSubDomains; preload
  vary: Accept-Encoding
  x-content-type-options: nosniff
  x-response-time: 910.024000ms
  x-xss-protection: 1; mode=block
▾ Request Headers
  :authority: www.opentable.com
  :method: GET
```



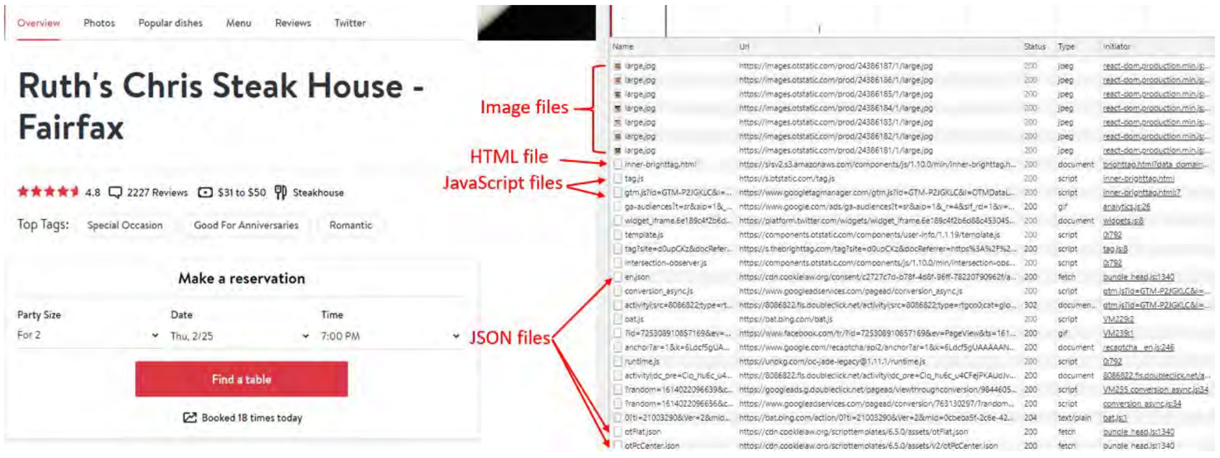
```
    <!DOCTYPE html>
    <html lang="en">
      <head>
        <title>Ruth&#x27;s Chris Steak House - Fairfax Restaurant - Fairfax, VA | OpenTable</title><meta name="viewport" content="width=device-width, :
        <script>window.OT = window.OT || {};
window.OT.SRS = {
  abTestingAnalytics: {"account":"2d65a94551a4a1caddd5a825a88698a89e19f2d5","hostname":"ac.opentable.com"},
  disableRaven: false,
  domainId: 1,
  jsHeadVersion: "1.0.92",
  lang: "en",
  ocUrl: window.location.protocol + '//oc-registry.opentable.com/v2/',
  requestId: "03cf6435-4f6c-4142-b8f6-ac50d3bd5925"
};

window.OT.searchAutoComplete = {};
window.OT.searchAutoComplete.ticketedRids = [681,1537,3334,7055,11065,29995,47119,53716,89905,100038,112846,114886,118765,140968,160504,185560,185773,:

window.oc = window.oc || {};
window.oc.conf = { globalParameters: { __ot_conservedHeaders: 'x2YPVT6POzqVVB9+77Ms1D6kzlo1qz4/Mcrw22iABQU0wBEL55I1QUmnPt1V76x53aanyfLoC38YVlaxuSeQScc(
</script><!--[if lte IE 8]> <script src="//components.otstatic.com/components/js-head/1.0.92/min/ie8_compat.js"></script> <![endif]-->
<!--[if IE 9]> <script src="//components.otstatic.com/components/js-head/1.0.92/min/ie9_compat.js"></script> <![endif]--><script src="//components.otst
<!--[if lte IE 9]> <script src="//components.otstatic.com/components/js-head/1.0.92/min/ie_jquery.js"></script> <![endif]--><script>window.optimizely :
        <link rel="shortcut icon" href="//components.otstatic.com/components/favicon/1.0.6/favicon/favicon.ico" type="image/x-icon"/><link rel="icon" |
      </head>
      <body>
        <div class="master-container master-container-redesign">
          <div class="header--legacy">
            <!--  Aria reference for this menu bar https://www.w3.org/TR/wai-aria-practices-1.1/examples/menubar/menubar-1/menubar-1.html#--><style>a.1
  e.preventDefault();
    var mainContentElement = jQuery('#container').first();
    jQuery('html, body').animate({
      scrollTop: (mainContentElement.offset().top)
    }, 500);
    // We can only focus a div if it has a tab index value. Set it to -1 so that we can focus it but we can't
    // tab to it and so we don't disrupt the normal tab navigation flow
    mainContentElement.attr('tabindex',-1).focus();
```

```
"/><link rel="stylesheet" type="text/css" href="//media.otstatic.com/restaurant-profile/index-aec1482bb94320feaf7f98c915f2b524.css"/><link rel="styl
```

165. The OpenTable Platform fetches HTML (*.html) files, CSS (*.css) files, and Javascript (*.js) files from the OpenTable Platform's servers and converts them with additional contents (e.g., JSON files and image files) into a working website. In the example below, the OpenTable Platform fetches its source code, such as HTML files and run time files (including CSS files, and Javascript files) from the OpenTable Platform's servers to build a restaurant profile for display. On information and belief, this functionality was present during relevant time periods of infringement.



Name	Url	Status	Type	Initiator
ruths-chris-steak-house-fairf...	https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee...	200	document	Other
index-aec1482bb94320feaf7f...	https://media.otstatic.com/restaurant-profile/index-aec1482bb94320feaf7f98...	200	stylesheet	ruths-chris-steak-house-fairf...
header-3b185dafb4153aa4b...	https://media.otstatic.com/css/header-3b185dafb4153aa4b2982b32ca20eec0...	200	stylesheet	ruths-chris-steak-house-fairf...
footer-bdeb5e02752433775...	https://media.otstatic.com/css/footer-bdeb5e0275243377573ba0166fa984fc.c...	200	stylesheet	ruths-chris-steak-house-fairf...
slick.min.css	https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick.min.css	200	stylesheet	ruths-chris-steak-house-fairf...
slick-theme.min.css	https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick-theme.min.css	200	stylesheet	ruths-chris-steak-house-fairf...
bundle_head.js	https://components.otstatic.com/components/js-head/1.0.92/min/bundle_he...	200	script	ruths-chris-steak-house-fairf...
oc-client.min.js	https://components.otstatic.com/components/oc-client/0.48.15/src/oc-client...	200	script	ruths-chris-steak-house-fairf...
style.css	https://components.otstatic.com/components/dtp/1.8.10/dist/style.css	200	stylesheet	ruths-chris-steak-house-fairf...
opentable-logo.svg	https://components.otstatic.com/components/header/2.2.44/img/opentable-l...	200	svg+xml	ruths-chris-steak-house-fairf...
dtp.js?dtpId=dtp-picker-33	https://components.otstatic.com/components/dtp/1.8.10/dist/dtp.js?dtpId=d...	200	script	ruths-chris-steak-house-fairf...
gtm.js?id=GTM-MJTRF6F&l=...	https://www.googletagmanager.com/gtm.js?id=GTM-MJTRF6F&l=OTDataLayer	200	script	bundle_head.js:3259
Brandon_txt_reg-webfont.wo...	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_reg-webfont.wo...	200	font	index-aec1482....css
Brandon_txt_med-webfont.w...	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_med-webfont.w...	200	font	index-aec1482....css
Brandon_txt_bld-webfont.wo...	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_bld-webfont.wo...	200	font	index-aec1482....css
icons-eb482b5c449b4ac7d2...	https://media.otstatic.com/fonts/icons-eb482b5c449b4ac7d25370ba85f3808...	200	font	header-3b185da....css
data:image/svg+xml,...	data:image/svg+xml,%3Csvg width='17px' height='17px' viewBox='0 0 17 17'...	200	svg+xml	ruths-chris-steak-house-fairf...
icons-1e9096a34352dc1c6b...	https://media.otstatic.com/fonts/icons-1e9096a34352dc1c6bd4a76799aa77c...	200	font	ruths-chris-steak-house-fairf...
23872341.jpg	https://resizer.otstatic.com/v2/photos/wide-huge/1/23872341.jpg	200	jpeg	ruths-chris-steak-house-fairf...
staticmap?key=AIzaSyBHDIZ...	https://maps.googleapis.com/maps/api/staticmap?key=AIzaSyBHDIZeuOrKa...	200	jpeg	ruths-chris-steak-house-fairf...
small.jpg	https://images.otstatic.com/prod/26016062/1/small.jpg	200	jpeg	ruths-chris-steak-house-fairf...
provided_by_2.png?sp_dt=2...	https://a.singleplatform.com/OpenTable/ruths-chris-steak-house-fairfax/provi...	304	png	ruths-chris-steak-house-fairf...
prop-types.min.js	https://unpkg.com/prop-types@15.7.2/prop-types.min.js	200	script	0:792
analytics.js	https://www.google-analytics.com/analytics.js	200	script	gtm.js?id=GTM-MJTRF6F&l=...
data:image/svg+xml;...	data:image/svg+xml;charset=utf-8,%3Csvg width='24' height='24' viewBox='...	200	svg+xml	ruths-chris-steak-house-fairf...
data:image/svg+xml;...	data:image/svg+xml;charset=utf-8,%3Csvg xmlns='http://www.w3.org/2000/s...	200	svg+xml	ruths-chris-steak-house-fairf...
data:image/svg+xml;...	data:image/svg+xml;charset=utf-8,%3Csvg width='21' height='21' viewBox='...	200	svg+xml	ruths-chris-steak-house-fairf...
data:image/svg+xml;...	data:image/svg+xml;charset=utf-8,%3Csvg width='24' height='24' viewBox='...	200	svg+xml	ruths-chris-steak-house-fairf...
react.production.min.js	https://unpkg.com/react@16.9.0/umd/react.production.min.js	200	script	0:792
data:image/svg+xml;...	data:image/svg+xml;charset=utf-8,%3Csvg width='24' height='24' viewBox='...	200	svg+xml	ruths-chris-steak-house-fairf...
data:image/svg+xml;...	data:image/svg+xml;charset=utf-8,%3Csvg width='24' height='24' viewBox='...	200	svg+xml	ruths-chris-steak-house-fairf...
data:image/svg+xml;...	data:image/svg+xml;charset=utf-8,%3Csvg width='16' height='16' viewBox='...	200	svg+xml	ruths-chris-steak-house-fairf...
data:image/svg+xml;...	data:image/svg+xml;charset=utf-8,%3Csvg width='16' height='16' viewBox='...	200	svg+xml	ruths-chris-steak-house-fairf...
collect?t=dc&aip=1&_r=3&v...	https://stats.g.doubleclick.net/j/collect?t=dc&aip=1&_r=3&v=1&_v=j88&tid...	200	xhr	bundle_head.js:1288
collect?v=1&_v=j88&a=1209...	https://www.google-analytics.com/collect?v=1&_v=j88&a=1209190188&t=p...	200	gif	analytics.js:26
react-dom.production.min.js	https://unpkg.com/react-dom@16.9.0/umd/react-dom.production.min.js	200	script	0:792



Name	Url	Status	Type	Initiator
large.jpg	https://images.otstatic.com/prod/24386187/1/large.jpg	200	jpeg	react-dom.production.min.js:...
large.jpg	https://images.otstatic.com/prod/24386186/1/large.jpg	200	jpeg	react-dom.production.min.js:...
large.jpg	https://images.otstatic.com/prod/24386185/1/large.jpg	200	jpeg	react-dom.production.min.js:...
large.jpg	https://images.otstatic.com/prod/24386184/1/large.jpg	200	jpeg	react-dom.production.min.js:...
large.jpg	https://images.otstatic.com/prod/24386183/1/large.jpg	200	jpeg	react-dom.production.min.js:...
large.jpg	https://images.otstatic.com/prod/24386182/1/large.jpg	200	jpeg	react-dom.production.min.js:...
large.jpg	https://images.otstatic.com/prod/24386181/1/large.jpg	200	jpeg	react-dom.production.min.js:...
inner-brighttag.html	https://srsv2.s3.amazonaws.com/components/js/1.10.0/min/inner-brighttag.h...	200	document	brighttag.html?data_domain...
tag.js	https://s.otstatic.com/tag.js	200	script	inner-brighttag.html
gtm.js?id=GTM-P2JGKLC&l=...	https://www.googletagmanager.com/gtm.js?id=GTM-P2JGKLC&l=OTMDataL...	200	script	inner-brighttag.html:7
ga-audiences?t=sr&aip=1&_...	https://www.google.com/ads/ga-audiences?t=sr&aip=1&_r=4&slf_rd=1&v=...	200	gif	analytics.js:26
widget_iframe.6e189c4f2b6d...	https://platform.twitter.com/widgets/widget_iframe.6e189c4f2b6d88c453045...	200	document	widgets.js:8
template.js	https://components.otstatic.com/components/user-info/1.1.19/template.js	200	script	0:792
tag?site=d0upCKz&docRefer...	https://s.thebrighttag.com/tag?site=d0upCKz&docReferrer=https%3A%2F%2...	200	script	tag.js:8
intersection-observer.js	https://components.otstatic.com/components/js/1.10.0/min/intersection-obs...	200	script	0:792
en.json	https://cdn.cookielaw.org/consent/c2727c7d-b78f-4d8f-96ff-78220790962f/a...	200	fetch	bundle_head.js:1340
conversion_async.js	https://www.googleadservices.com/pagead/conversion_async.js	200	script	gtm.js?id=GTM-P2JGKLC&l=...
activity;src=8086822;type=rt...	https://8086822.fls.doubleclick.net/activity;src=8086822;type=rtgco0;cat=glo...	302	documen...	gtm.js?id=GTM-P2JGKLC&l=...
bat.js	https://bat.bing.com/bat.js	200	script	VM229:2
?id=725308910857169&ev=...	https://www.facebook.com/tr/?id=725308910857169&ev=PageView&ts=161...	200	gif	VM239:1
anchor?ar=1&k=6Ldcf5gUA...	https://www.google.com/recaptcha/api2/anchor?ar=1&k=6Ldcf5gUAAAAAN...	200	document	recaptcha__en.js:246
runtime.js	https://unpkg.com/oc-jade-legacy@1.11.1/runtime.js	200	script	0:792
activity;dc_pre=Clq_hu6c_u4...	https://8086822.fls.doubleclick.net/activity;dc_pre=Clq_hu6c_u4CFeJPKAUdJv...	200	document	8086822.fls.doubleclick.net/a...
?random=1614022096639&c...	https://googleads.g.doubleclick.net/pagead/viewthroughconversion/9844605...	200	script	VM255 conversion_async.js:34
?random=1614022096656&c...	https://www.googleadservices.com/pagead/conversion/763130297/?random...	200	script	conversion_async.js:34
0?ti=21003290&Ver=2&mid...	https://bat.bing.com/action/0?ti=21003290&Ver=2&mid=0cbeba5f-2c6e-42...	204	text/plain	bat.js:1
otFlat.json	https://cdn.cookielaw.org/scripttemplates/6.5.0/assets/otFlat.json	200	fetch	bundle_head.js:1340
otPcCenter.json	https://cdn.cookielaw.org/scripttemplates/6.5.0/assets/v2/otPcCenter.json	200	fetch	bundle_head.js:1340

166. The presence of the above referenced features is demonstrated, by way of example, by testing the Accused Instrumentalities for investigative purposes on https://www.booking.com/, https://join.booking.com/, https://partner.booking.com/en-us, and/or https://ycs.agoda.com/en-us/kipp/public/home, and by reference to publicly available information, including the following:

- https://www.booking.com/,
- https://join.booking.com/,
- https://partner.booking.com/en-us,
- https://partner.booking.com/en-us/help/working-booking/how-do-i-join-

bookingcom,

- https://partner.booking.com/en-us/help/working-booking/how-can-i-set-my-property-easily,
- https://ycs.agoda.com/en-us/kipp/public/home,
- https://www.agoda.com/info/ycs-online-registration.html?cid=1844104,
- https://www.agoda.com/info/privacy.html?cid=1844104,
- https://partners.agoda.com/en-us/faq.html,
- https://agodapropertyhelp.zendesk.com/hc/en-us,
- https://partners.agoda.com/,
- https://agodapropertyhelp.zendesk.com/hc/en-us/articles/115009545508-How-do-I-list-my-property-on-Agoda-com-,
- https://www.opentable.com/,
- https://support.opentable.com/s/?language=en_US,
- https://platform.opentable.com/documentation/, and
- https://restaurant.opentable.com/.

167. On information and belief, Defendants have had knowledge of the ‘168 patent and their infringement thereof at least as early as October 3, 2019, and no later than November 30, 2020, when Plaintiff provided notice of the ‘168 patent and Defendants’ infringement of the ‘168 patent. Furthermore, Defendants have been aware of the ‘168 patent and their infringement thereof since at least the filing of the original Complaint, D.I. 1, on December 1, 2020.

168. On information and belief, Defendants have contributed and are contributing to the infringement of the ‘168 patent because Defendants know that the infringing aspects of their infringing products and services, including but not limited to the Accused Instrumentalities, are made for use in an infringement, and are not staple articles of commerce suitable for substantial non-infringing uses.

169. On information and belief, Defendants have induced and are inducing the infringement of the ‘168 patent, with knowledge of the ‘168 patent and that their acts, including

without limitation using, offering to sell, selling within, and importing into the United States, the Accused Instrumentalities, would aid and abet and induce infringement by customers, clients, partners, developers, and end users of the foregoing.

170. In particular, Defendants' actions that aid and abet others such as customers, clients, partners, developers, and end users to infringe include advertising and distributing the Accused Instrumentalities, providing instructional materials, training, and other services regarding the Accused Instrumentalities, and providing free listings for the Accused Instrumentalities. Defendants actively encouraged the adoption of the Accused Instrumentalities and provided support sites for the vast network of developers working with the Accused Instrumentalities, emphasizing the simple and user-friendly nature of the Accused Instrumentalities, for example, explaining that "Registration can take as little as 15 minutes to complete—get started today" and that Defendants provides "24/7 support by phone or email" (*see, e.g.*, https://join.booking.com/), that "Through one platform our hotel partners are distributed across both priceline.com and agoda.com maximizing performance across our Retail, Private, Opaque and Vacation Packages programs. Our partners can easily set rates, promotions, and change hotel details with a few simple clicks of a mouse or on the go with a mobile device" (*see, e.g.*, https://ycs.agoda.com/en-us/kipp/public/home), and that "From online ordering and takeout to powerful marketing and experiences, make more money when you access our network of millions" and that restaurants can "[g]et discovered and capture the business of the millions of people, around the world and in your neighborhood, searching on OpenTable" (*see, e.g.*, https://restaurant.opentable.com/?utm_source=dinersite&utm_medium=referral&utm_campaign=topnav&Lead.LeadSource=DinerSite&Lead.Marketing_ID__c=topnav). On information and belief, Defendants have engaged in such actions with specific intent to cause infringement or with willful blindness to the resulting infringement because Defendants have had actual knowledge of the '168 patent and knowledge that their acts were inducing infringement of the '168 patent since at least the date Defendants received notice that their activities infringed the '168 patent.

171. Defendants' acts of infringement have caused damage to Plaintiff, and Plaintiff

is entitled to recover damages from Defendants in an amount subject to proof at trial.

172. Defendants' infringement of Plaintiff's rights under the '168 patent will continue to damage Plaintiff's business, causing irreparable harm, for which there is no adequate remedy at law, unless enjoined by this Court.

173. On information and belief, Defendants have acted with disregard of Plaintiff's patent rights, without any reasonable basis for doing so, and have willfully infringed and do willfully infringe the '168 patent.

174. The foregoing is illustrative of Defendants' infringement of the '168 patent. Plaintiff reserves the right to identify additional claims and Accused Instrumentalities in accordance with the Court's local rules and applicable scheduling orders.

COUNT III – INFRINGEMENT OF U.S. PATENT NO. 9,063,755

175. Plaintiff incorporates by reference paragraphs 1 to 55 above as if fully set forth herein.

176. On information and belief, Defendants have infringed and are infringing the '755 patent under 35 U.S.C. § 271, either literally and/or under the doctrine of equivalents, directly and/or indirectly.

177. On information and belief, Booking-BV has infringed and continues to infringe the '755 patent by performing, without authority, one or more of the following acts: making, using, offering to sell, selling within, and importing into, the United States products and services that practice the claimed inventions of the '755 patent, including but not limited to the WebDirect Platform and the Booking Platform.[5]

178. On information and belief, Priceline and Agoda (the "YCS infringers") have

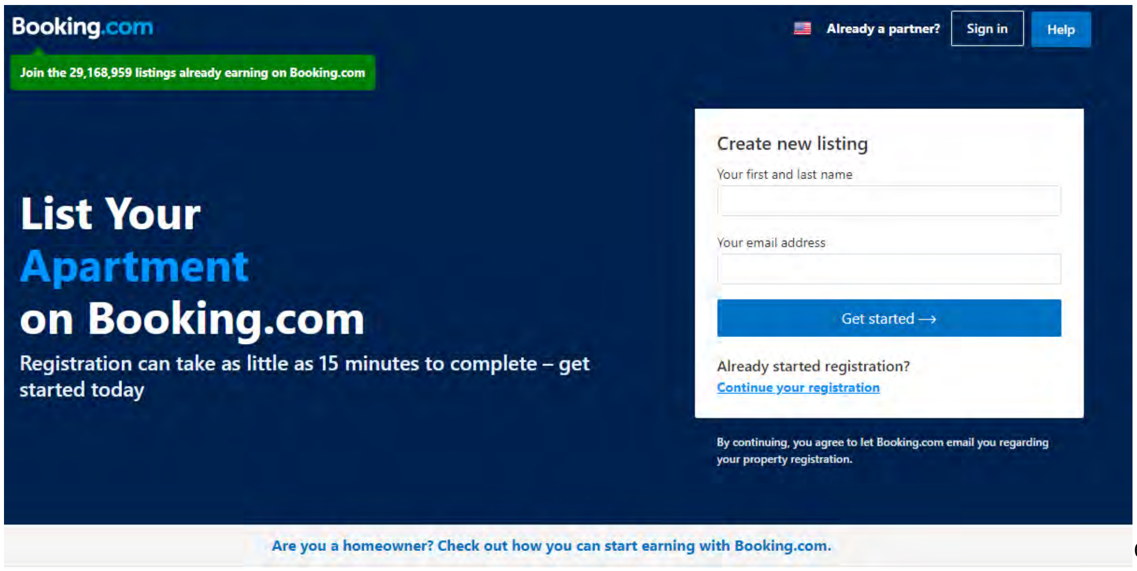
[5] This Count focuses its infringement allegations on the Booking Platform, YCS Platform, and the OpenTable Platform. Upon information and belief, the WebDirect Platform operates in a similar fashion as the Booking and YCS Platforms, and discovery of the WebDirect Platform, including confidential documents related thereto, will confirm these facts.

infringed and continue to infringe the '755 patent by performing, without authority, one or more of the following acts: making, using, offering to sell, selling within, and importing into, the United States products and services that practice the claimed inventions of the '755 patent, including but not limited to the WebDirect Platform and the Booking Platform.

179. On information and belief, OpenTable has infringed and continues to infringe the '755 patent by performing, without authority, one or more of the following acts during relevant time periods: making, using, offering to sell, selling within, and importing into the United States products and services that practice the claimed inventions of the '755 patent, including but not limited to the OpenTable Platform.

The Booking Platform:

180. Booking-BV infringes at least claim 12 of the '755 patent through a combination of features in the Booking Platform that collectively practice each claimed limitation of claim 12. By way of example, Booking-BV provides the Booking Platform for creating property listing websites for hotels, homes, apartments, and other lodgings where travelers can view and book accommodations.


Booking.com

Already a partner? Sign in Help

Join the 29,168,959 listings already earning on Booking.com

List Your Apartment on Booking.com

Registration can take as little as 15 minutes to complete – get started today

Create new listing

Your first and last name

Your email address

Get started →

Already started registration?

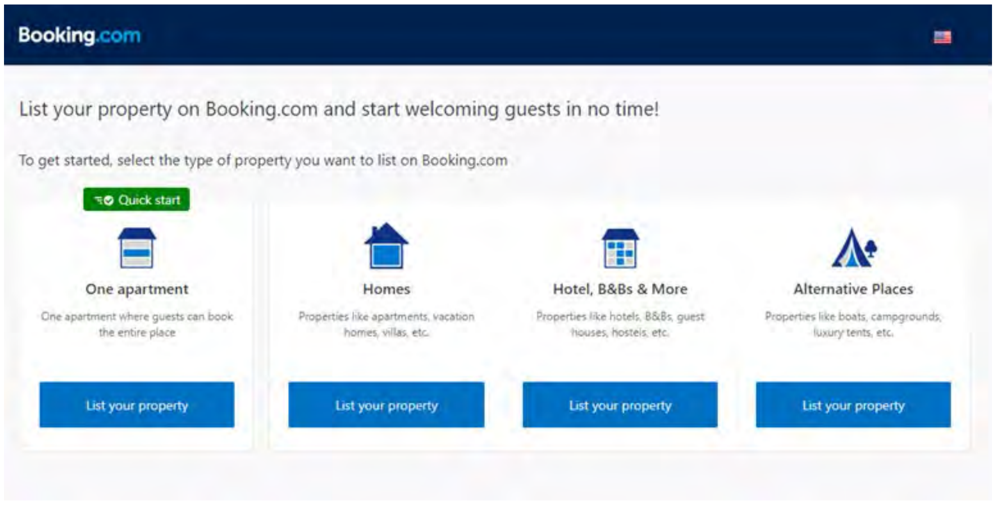
Continue your registration

By continuing, you agree to let Booking.com email you regarding your property registration.

Are you a homeowner? Check out how you can start earning with Booking.com.

[6]

[6] Unless otherwise noted, the images presented in this Count were generated for investigative


Booking.com

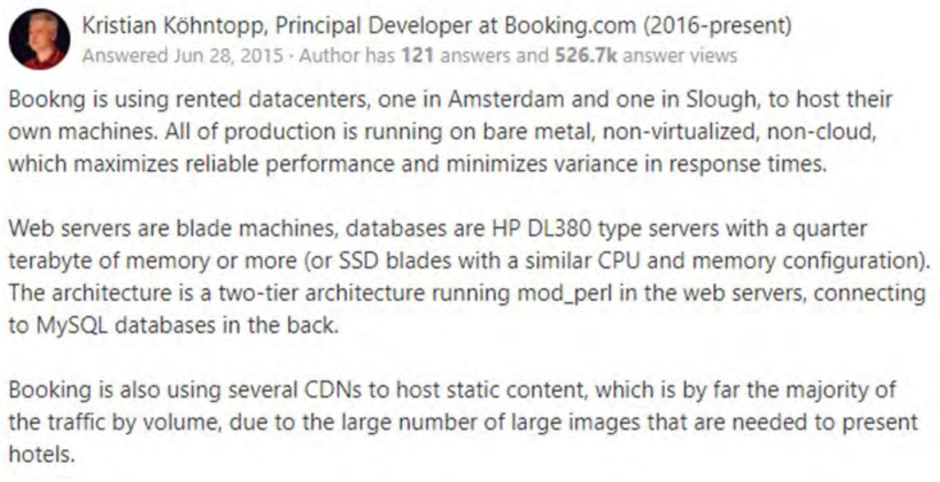
List your property on Booking.com and start welcoming guests in no time!

To get started, select the type of property you want to list on Booking.com

Quick start
One apartment
One apartment where guests can book the entire place
List your property
Homes
Properties like apartments, vacation homes, villas, etc.
List your property
Hotel, B&Bs & More
Properties like hotels, B&Bs, guest houses, hostels, etc.
List your property
Alternative Places
Properties like boats, campgrounds, luxury tents, etc.
List your property

181. The Booking Platform utilizes a registry of web components related to inputs and outputs of web services, with each web component including a plurality of corresponding symbolic names for inputs and outputs. On information and belief, the Booking Platform is supported by MySQL, a relational database management system that serves as a back-end server supporting Booking websites. This includes support for website visitors creating, reading, updating, and deleting (CRUD) datasets within a Booking property listing website, providing a registry of web components for inputs and outputs of web services.

purposes by testing the Accused Instrumentalities on https://www.booking.com/, https://join.booking.com/, https://partner.booking.com/en-us, https://ycs.agoda.com/en-us/kipp/public/home, https://www.opentable.com/, https://support.opentable.com/s/?language=en_US, https://platform.opentable.com/documentation/, and/or other associated websites.

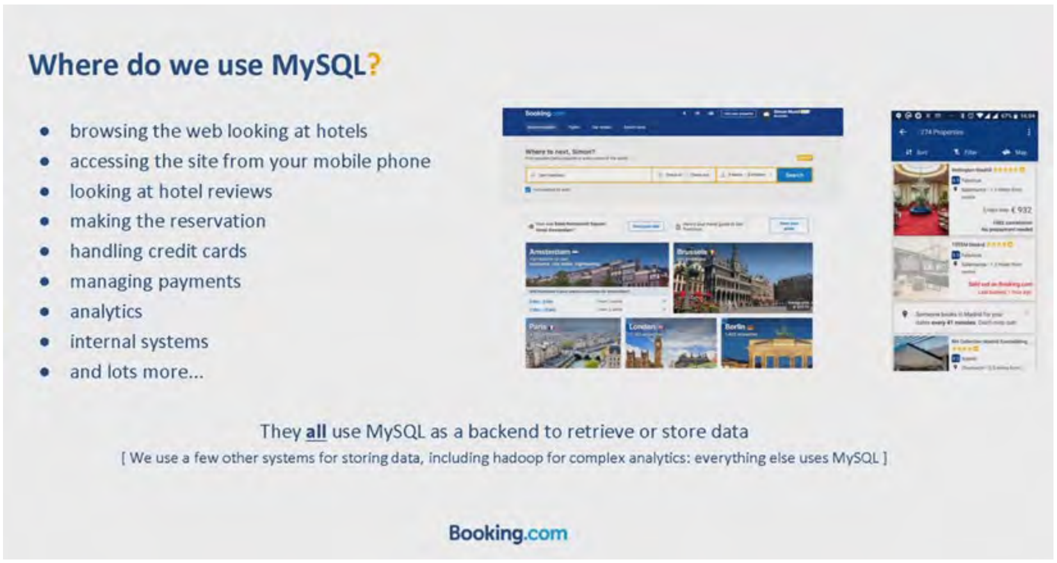

Kristian Köhntopp, Principal Developer at Booking.com (2016-present)
Answered Jun 28, 2015 · Author has **121** answers and **526.7k** answer views

Bookng is using rented datacenters, one in Amsterdam and one in Slough, to host their own machines. All of production is running on bare metal, non-virtualized, non-cloud, which maximizes reliable performance and minimizes variance in response times.

Web servers are blade machines, databases are HP DL380 type servers with a quarter terabyte of memory or more (or SSD blades with a similar CPU and memory configuration). The architecture is a two-tier architecture running mod_perl in the web servers, connecting to MySQL databases in the back.

Booking is also using several CDNs to host static content, which is by far the majority of the traffic by volume, due to the large number of large images that are needed to present hotels.

https://www.quora.com/Who-hosts-the-booking-com-website


Where do we use MySQL?

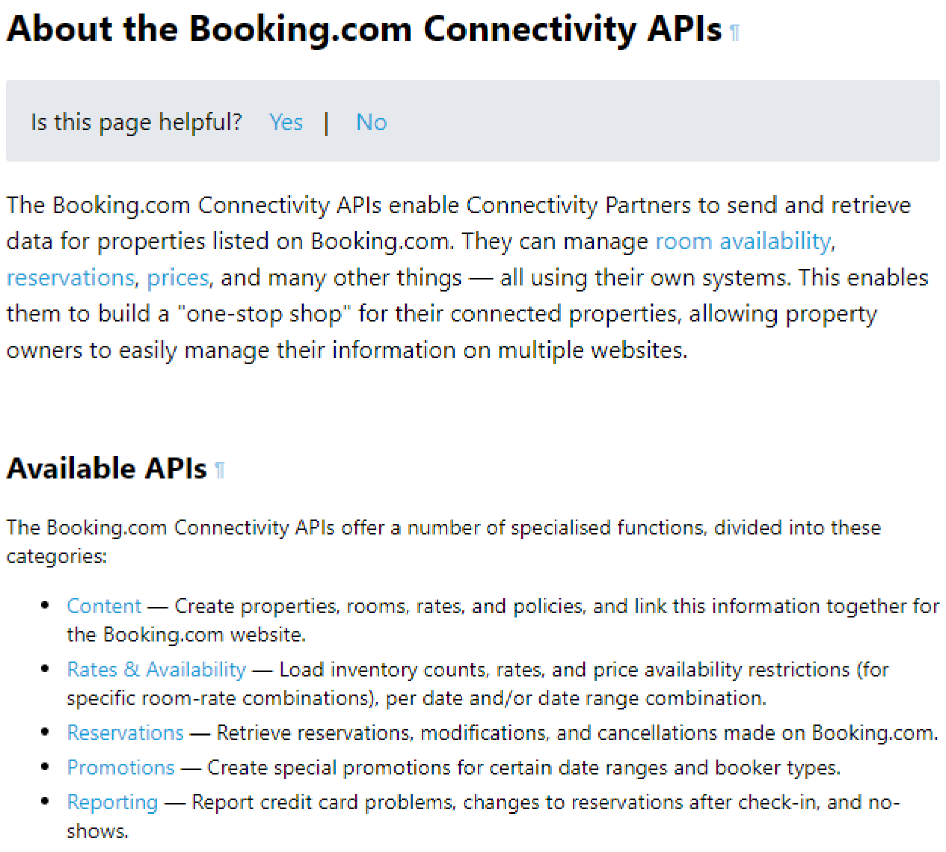
- browsing the web looking at hotels
- accessing the site from your mobile phone
- looking at hotel reviews
- making the reservation
- handling credit cards
- managing payments
- analytics
- internal systems
- and lots more...

They **all** use MySQL as a backend to retrieve or store data
[We use a few other systems for storing data, including hadoop for complex analytics: everything else uses MySQL]

Booking.com

https://www.youtube.com/watch?v=iNxqZSbaHYQ&feature=youtu.be

182. Web services on the Booking Platform include various Application Programming Interfaces (APIs) that may be integrated on Booking websites.



About the Booking.com Connectivity APIs ¶

Is this page helpful? Yes | No

The Booking.com Connectivity APIs enable Connectivity Partners to send and retrieve data for properties listed on Booking.com. They can manage room availability, reservations, prices, and many other things — all using their own systems. This enables them to build a "one-stop shop" for their connected properties, allowing property owners to easily manage their information on multiple websites.

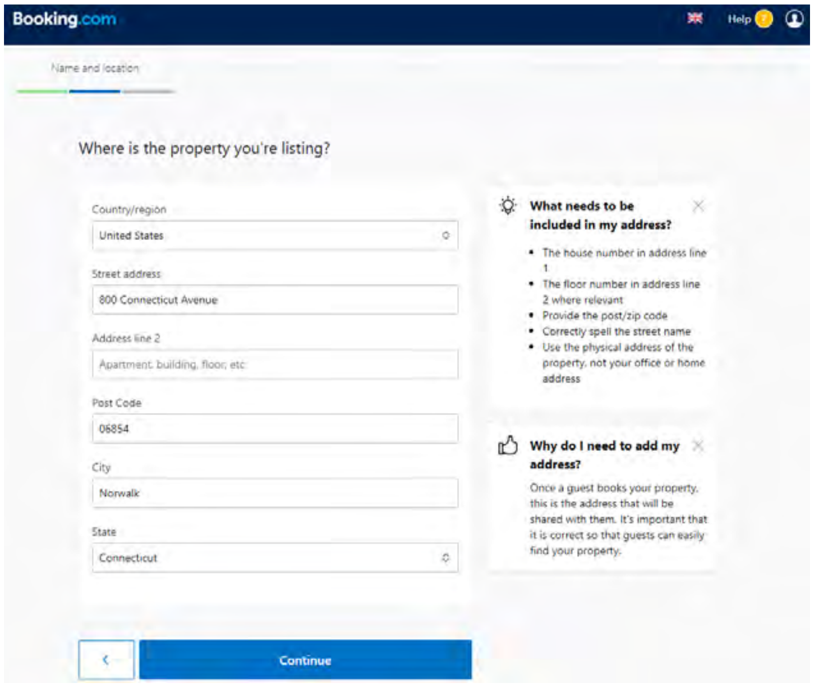
Available APIs ¶

The Booking.com Connectivity APIs offer a number of specialised functions, divided into these categories:

- Content — Create properties, rooms, rates, and policies, and link this information together for the Booking.com website.
- Rates & Availability — Load inventory counts, rates, and price availability restrictions (for specific room-rate combinations), per date and/or date range combination.
- Reservations — Retrieve reservations, modifications, and cancellations made on Booking.com.
- Promotions — Create special promotions for certain date ranges and booker types.
- Reporting — Report credit card problems, changes to reservations after check-in, and no-shows.

https://connect.booking.com/user_guide/site/en-US/

183. By way of example, the Google Maps API be embedded on a Booking website to display a property's location and other nearby attractions and points of interest. As shown below, a user can input a property address, which is embedded in a request URL and sent to the Google Maps API via an HTTP GET request for Geocoding service, which in turn converts the input address into geographic coordinates and places a marker on the map indicating the property's location.



Booking.com

Name and location

Where is the property you're listing?

Country/region: United States

Street address: 800 Connecticut Avenue

Address line 2: Apartment, building, floor, etc

Post Code: 06854

City: Norwalk

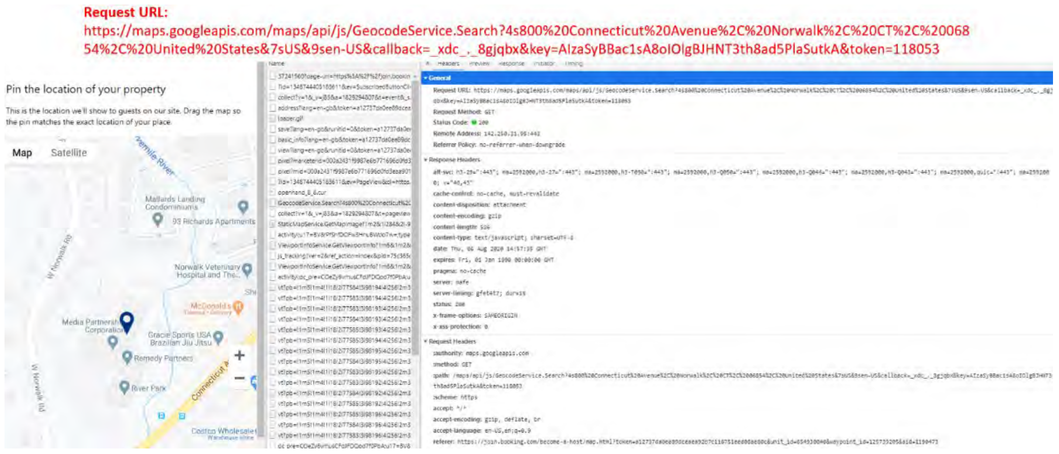
State: Connecticut

What needs to be included in my address?

- The house number in address line 1
- The floor number in address line 2 where relevant
- Provide the post/zip code
- Correctly spell the street name
- Use the physical address of the property, not your office or home address

Why do I need to add my address?

Once a guest books your property, this is the address that will be shared with them. It's important that it is correct so that guests can easily find your property.



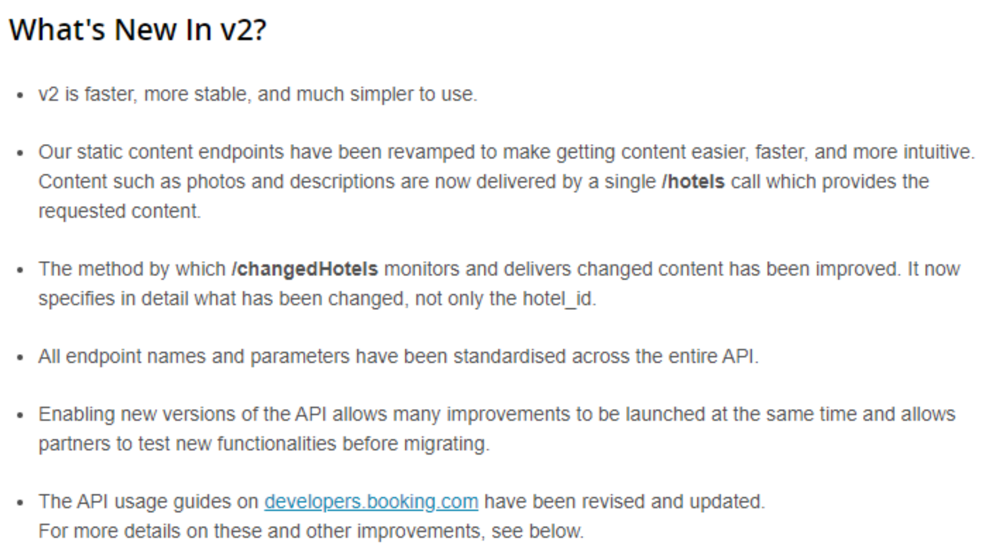
Request URL:
https://maps.googleapis.com/maps/api/js/GeocodeService.Search?4s800%20Connecticut%20Avenue%2C%20Norwalk%2C%20CT%2C%2006854%2C%20United%20States&7sUS&9sen-US&callback=_xdc_._8gjqbx&key=AIzaSyBBac1sA8oIOlgBJHNT3th8ad5PlaSutkA&token=118053

Pin the location of your property

This is the location we'll show to guests on our site. Drag the map so the pin matches the exact location of your place.

```
General
Request URL: https://maps.googleapis.com/maps/api/js/GeocodeService.Search?4s800%20Connecticut%20Avenue%2C%20Norwalk%2C%20CT%2C%2006854%2C%20United%20States&7sUS&9sen-US&callback=_xdc_._8gjqbx&key=AIzaSyBBac1sA8oIOlgBJHNT3th8ad5PlaSutkA&token=118053
Request Method: GET
Status Code: 200
Remote Address: 142.250.31.95:443
Referrer Policy: no-referrer-when-downgrade
Response Headers
alt-svc: [illegible]
cache-control: no-cache, must-revalidate
content-disposition: attachment
content-encoding: gzip
content-length: 536
content-type: text/javascript; charset=UTF-8
date: Thu, 06 Aug 2020 14:57:39 GMT
expires: Fri, 01 Jan 1990 00:00:00 GMT
pragma: no-cache
server: mafe
server-timing: gfet4t7; dur=39
status: 200
x-frame-options: SAMEORIGIN
x-xss-protection: 0
Request Headers
:authority: maps.googleapis.com
:method: GET
:path: /maps/api/js/GeocodeService.Search?4s800%20Connecticut%20Avenue%2C%20Norwalk%2C%20CT%2C%2006854%2C%20United%20States&7sUS&9sen-US&callback=_xdc_._8gjqbx&key=AIzaSyBBac1sA8oIOlgBJHNT3th8ad5PlaSutkA&token=118053
:scheme: https
accept: */*
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
referer: [illegible]
```

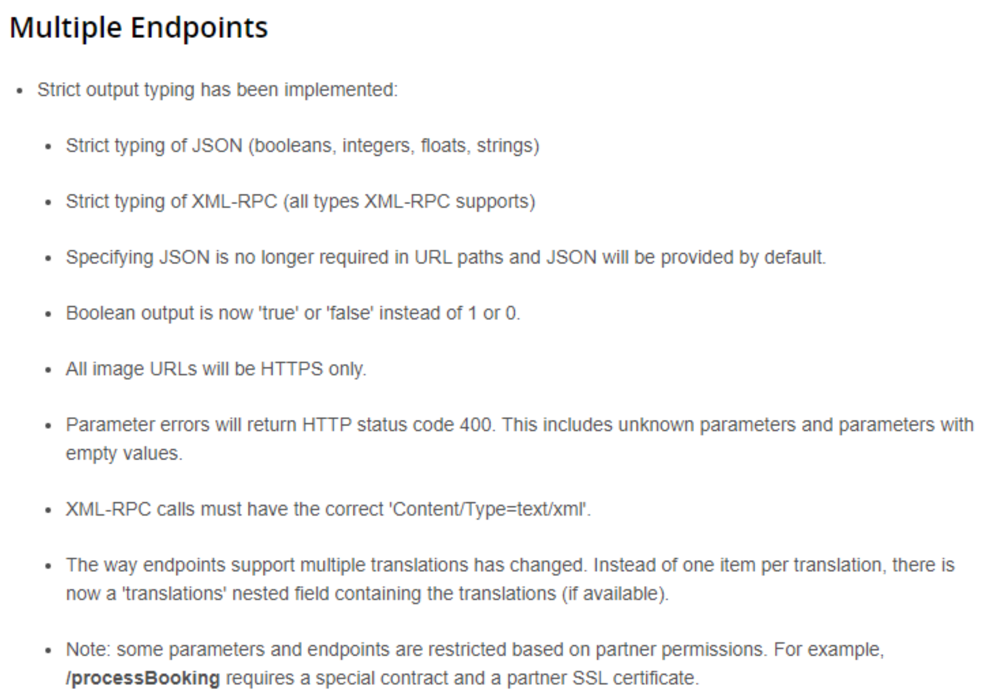
184. On information and belief, the Booking Platform utilizes JSON data to evoke web components and communicate inputs and outputs for the APIs. Each web component includes a plurality of corresponding symbolic names for inputs and outputs for a web service. These symbolic names are required for evoking the web components and constitute character strings that do not contain a persistent address or pointer. In particular, JSON datasets used by the Booking Platform comprise name/value pairs that are essentially character strings (i.e., symbolic names) with no persistent address or pointers.



What's New In v2?

- v2 is faster, more stable, and much simpler to use.
- Our static content endpoints have been revamped to make getting content easier, faster, and more intuitive. Content such as photos and descriptions are now delivered by a single **/hotels** call which provides the requested content.
- The method by which **/changedHotels** monitors and delivers changed content has been improved. It now specifies in detail what has been changed, not only the hotel_id.
- All endpoint names and parameters have been standardised across the entire API.
- Enabling new versions of the API allows many improvements to be launched at the same time and allows partners to test new functionalities before migrating.
- The API usage guides on developers.booking.com have been revised and updated. For more details on these and other improvements, see below.

https://developers.booking.com/api/commercial/index.html?version=2.5&page_url=migration-guide



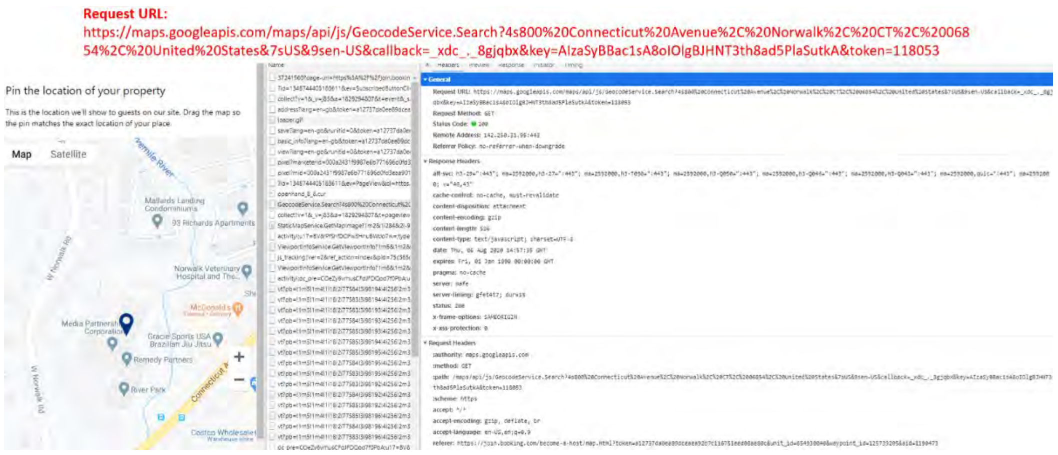
Multiple Endpoints

- Strict output typing has been implemented:
  - Strict typing of JSON (booleans, integers, floats, strings)
  - Strict typing of XML-RPC (all types XML-RPC supports)
  - Specifying JSON is no longer required in URL paths and JSON will be provided by default.
  - Boolean output is now 'true' or 'false' instead of 1 or 0.
  - All image URLs will be HTTPS only.
  - Parameter errors will return HTTP status code 400. This includes unknown parameters and parameters with empty values.
  - XML-RPC calls must have the correct 'Content/Type=text/xml'.
  - The way endpoints support multiple translations has changed. Instead of one item per translation, there is now a 'translations' nested field containing the translations (if available).
  - Note: some parameters and endpoints are restricted based on partner permissions. For example, **/processBooking** requires a special contract and a partner SSL certificate.

https://developers.booking.com/api/commercial/index.html?version=2.5&page_url=migration-guide

185. The registry for the Booking Platform also includes addresses for the web

services where the input symbolic names and output symbolic names can be sent to and received from. For example, the screenshot below shows a request URL including an address of the Google Maps web service when a user inputs an address for Geocoding service in the Google Maps API. In particular, the user-input property address "800 Connecticut Avenue. Norwalk, CT 06854" is embedded in the request URL and sent to the Google Maps API via an HTTP GET request. The Geocoding service in the Google Maps API then responds with geographic coordinates for the user-input address in the format of JSON data and places a marker on the map indicating the property's location. The geographic coordinates and the user input address are stored in the Booking's database for further use.



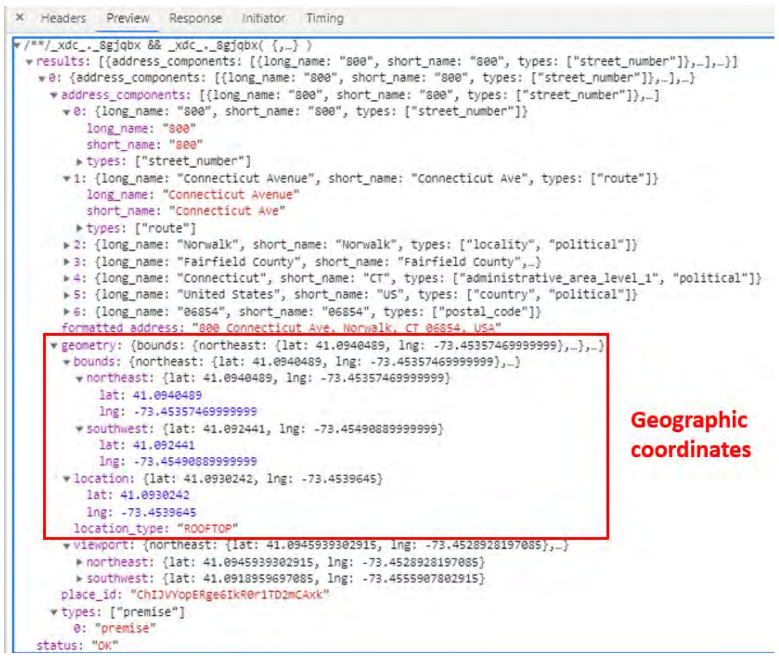
Request URL:
https://maps.googleapis.com/maps/api/js/GeocodeService.Search?4s800%20Connecticut%20Avenue%2C%20Norwalk%2C%20CT%2C%2006854%2C%20United%20States&7sUS&9sen-US&callback=_xdc_._8gjqbx&key=AIzaSyBBac1sA8oIOlgBJHNT3th8ad5PlaSutkA&token=118053

Pin the location of your property

This is the location we'll show to guests on our site. Drag the map so the pin matches the exact location of your place.

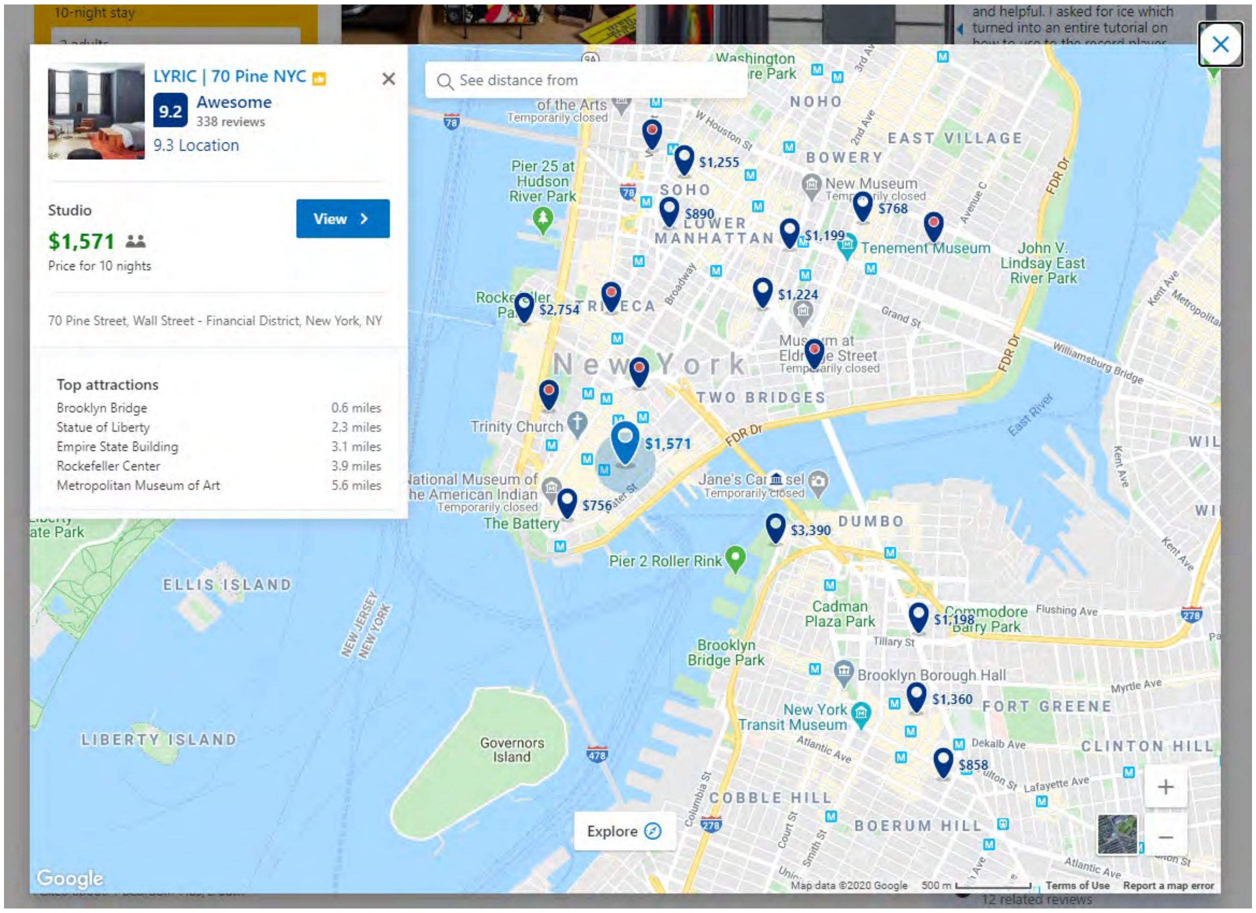
Map
Satellite
Mallards Landing Condominiums
93 Richards Apartments
Norwalk Veterinary Hospital and The...
McDonald's
Media Partners Corporation
Gracie Sports USA Brazilian Jiu Jitsu
Remedy Partners
River Park
Costco Wholesale
Headers
Preview
Response
Initiator
Timing

```
▾ General
Request URL: https://maps.googleapis.com/maps/api/js/GeocodeService.Search?4s800%20Connecticut%20Avenue%2C%20Norwalk%2C%20CT%2C%2006854%2C%20United%20States&7sUS&9sen-US&callback=_xdc_._8gjqbx&key=AIzaSyBBac1sA8oIOlgBJHNT3th8ad5PlaSutkA&token=118053
Request Method: GET
Status Code: ● 200
Remote Address: 142.250.31.95:443
Referrer Policy: no-referrer-when-downgrade
▾ Response Headers
alt-svc: h3-29=":443"; ma=2592000,h3-27=":443"; ma=2592000,h3-T050=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"
cache-control: no-cache, must-revalidate
content-disposition: attachment
content-encoding: gzip
content-length: 526
content-type: text/javascript; charset=UTF-8
date: Thu, 06 Aug 2020 14:57:35 GMT
expires: Fri, 01 Jan 1990 00:00:00 GMT
pragma: no-cache
server: mafe
server-timing: gfet4t7; dur=35
status: 200
x-frame-options: SAMEORIGIN
x-xss-protection: 0
▾ Request Headers
:authority: maps.googleapis.com
:method: GET
:path: /maps/api/js/GeocodeService.Search?4s800%20Connecticut%20Avenue%2C%20Norwalk%2C%20CT%2C%2006854%2C%20United%20States&7sUS&9sen-US&callback=_xdc_._8gjqbx&key=AIzaSyBBac1sA8oIOlgBJHNT3th8ad5PlaSutkA&token=118053
:scheme: https
accept: */*
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
referer: https://join.booking.com/become-a-host/map.html?token=a12737da0e89dcee89207c116751eed00a680c&unit_id=6549300400&waypoint_id=1257332054&sid=1190473
```



```
▾ /**/_xdc_._8gjqbx && _xdc_._8gjqbx( {,…} )
  ▾ results: [{address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…],…}]
    ▾ 0: {address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…],…}
      ▾ address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…]
        ▾ 0: {long_name: "800", short_name: "800", types: ["street_number"]}
            long_name: "800"
            short_name: "800"
          ▸ types: ["street_number"]
        ▾ 1: {long_name: "Connecticut Avenue", short_name: "Connecticut Ave", types: ["route"]}
            long_name: "Connecticut Avenue"
            short_name: "Connecticut Ave"
          ▸ types: ["route"]
        ▸ 2: {long_name: "Norwalk", short_name: "Norwalk", types: ["locality", "political"]}
        ▸ 3: {long_name: "Fairfield County", short_name: "Fairfield County",…}
        ▸ 4: {long_name: "Connecticut", short_name: "CT", types: ["administrative_area_level_1", "political"]}
        ▸ 5: {long_name: "United States", short_name: "US", types: ["country", "political"]}
        ▸ 6: {long_name: "06854", short_name: "06854", types: ["postal_code"]}
        formatted address: "800 Connecticut Ave, Norwalk, CT 06854, USA"
      ▾ geometry: {bounds: {northeast: {lat: 41.0940489, lng: -73.45357469999999},…},…}
        ▾ bounds: {northeast: {lat: 41.0940489, lng: -73.45357469999999},…}
          ▾ northeast: {lat: 41.0940489, lng: -73.45357469999999}
              lat: 41.0940489
              lng: -73.45357469999999
          ▾ southwest: {lat: 41.092441, lng: -73.45490889999999}
              lat: 41.092441
              lng: -73.45490889999999
        ▾ location: {lat: 41.0930242, lng: -73.4539645}
            lat: 41.0930242
            lng: -73.4539645
          location_type: "ROOFTOP"
        ▾ viewport: {northeast: {lat: 41.0945939302915, lng: -73.4528928197085},…}
          ▸ northeast: {lat: 41.0945939302915, lng: -73.4528928197085}
          ▸ southwest: {lat: 41.0918959697085, lng: -73.4555907802915}
        place_id: "ChIJVYopERge6IkR0r1TD2mCAxk"
      ▾ types: ["premise"]
          0: "premise"
    status: "OK"
```

186. The Booking Platform defines a UI object corresponding to a web component for an input or output of a web service, for presentation on the display. For example, when the Google Maps API is embedded onto a Booking website, UI objects for web components such as a map image, a pin indicating the property's location, zoom-in and zoom-out buttons, and points showing nearby attractions and points of interest are defined for the Google Map web service.


LYRIC | 70 Pine NYC

9.2 Awesome
338 reviews
9.3 Location

Studio
$1,571
Price for 10 nights

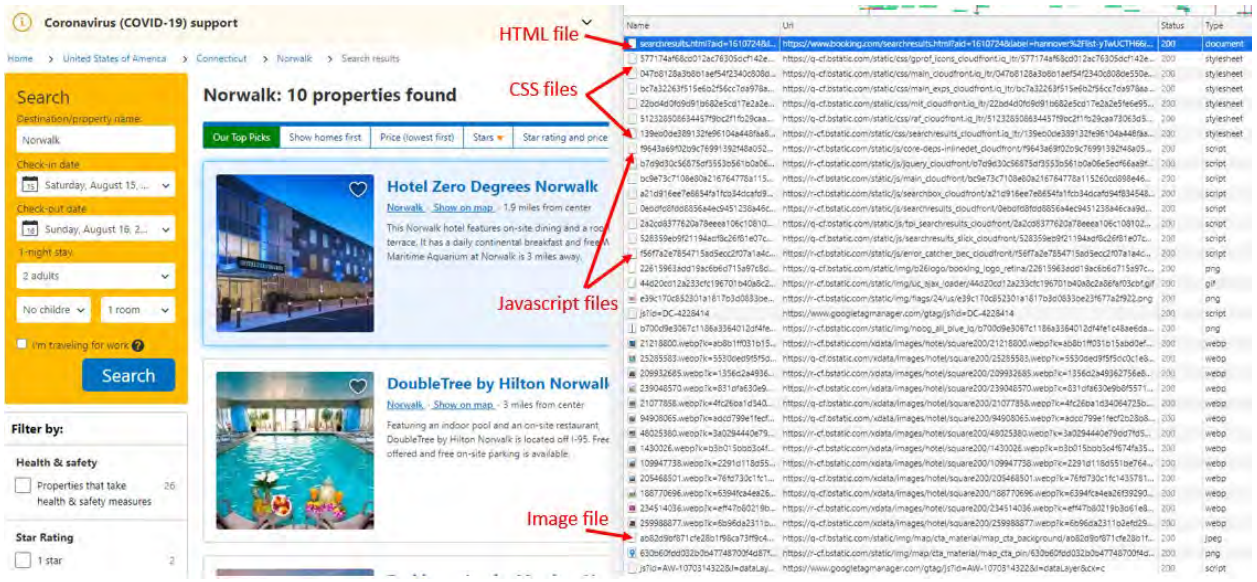
View >

70 Pine Street, Wall Street - Financial District, New York, NY

Top attractions

Brooklyn Bridge	0.6 miles
Statue of Liberty	2.3 miles
Empire State Building	3.1 miles
Rockefeller Center	3.9 miles
Metropolitan Museum of Art	5.6 miles

187. When a UI object is defined, a symbolic name (e.g., JSON data) from the web component corresponding to the defined UI object is selected and associated with the defined UI object. The Booking Platform then produces a device-independent application that includes the JSON data, as well as standard HTML, CSS and Javascript code. The application is executed on a device together with a device-dependent player, such as a device-dependent code for a browser engine for a specific browser, or an application or operating system for a particular kind of device. *See, e.g.*, *Shopify Inc. v. Express Mobile, Inc.*, Case No. 1:19-cv-00439-RGA, D. Del., D.I. 137. In the example below, the Booking Platform converts HTML, CSS, Javascript, image, and other files into an active property website.

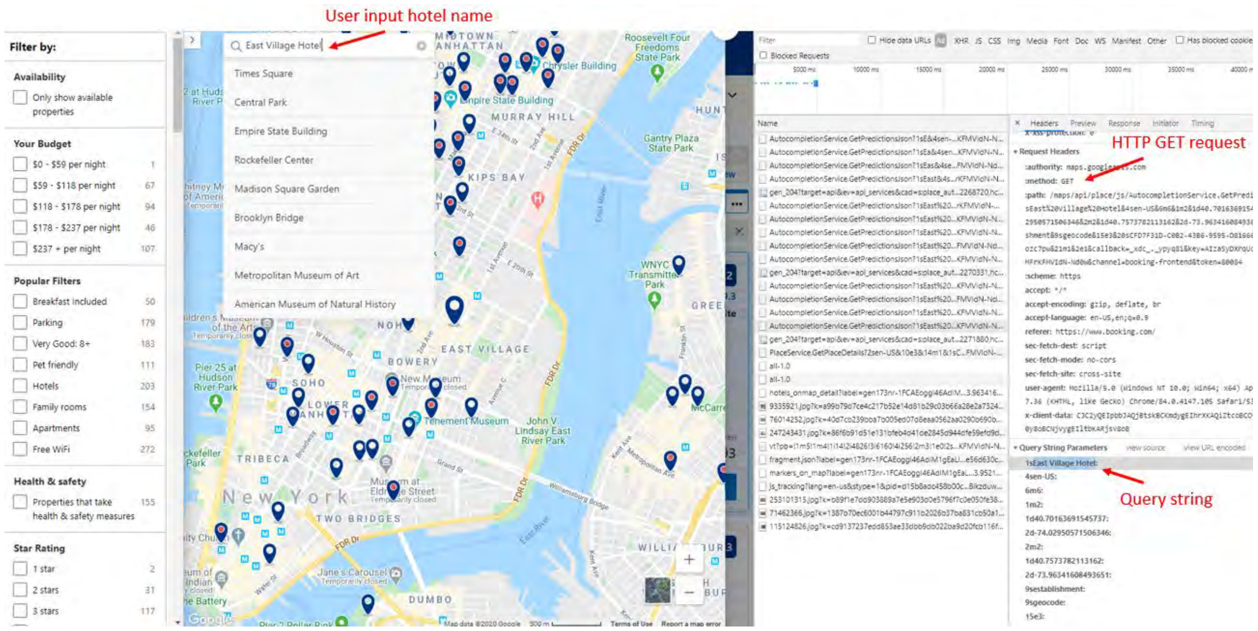

Coronavirus (COVID-19) support
Home > United States of America > Connecticut > Norwalk > Search results
Search
Destination/property name:
Norwalk
Check-in date
Saturday, August 15, ...
Check-out date
Sunday, August 16, 2...
1-night stay
2 adults
No childre
1 room
I'm traveling for work
Search
Filter by:
Health & safety
Properties that take health & safety measures 26
Star Rating
1 star 2
Norwalk: 10 properties found
Our Top Picks
Show homes first
Price (lowest first)
Stars
Star rating and price
Hotel Zero Degrees Norwalk
Norwalk Show on map 1.9 miles from center
DoubleTree by Hilton Norwalk
Norwalk Show on map 3 miles from center
HTML file
CSS files
Javascript files
Image file

This Norwalk hotel features on-site dining and a roof terrace. It has a daily continental breakfast and free ... Maritime Aquarium at Norwalk is 3 miles away.

Featuring an indoor pool and an on-site restaurant, DoubleTree by Hilton Norwalk is located off I-95. Free ... offered and free on-site parking is available.

Name	Url	Status	Type
searchresults.html?aid=1610724&l...	https://www.booking.com/searchresults.html?aid=1610724&label=hannover%2Flist-yTwUCTH66i...	200	document
577174af68cd012ac76305dcf142e...	https://q-cf.bstatic.com/static/css/gprof_icons_cloudfront.iq_ltr/577174af68cd012ac76305dcf142e...	200	stylesheet
047b8128a3b8b1aef54f2340c808d...	https://q-cf.bstatic.com/static/css/main_cloudfront.iq_ltr/047b8128a3b8b1aef54f2340c808de550e...	200	stylesheet
bc7a32263f515e6b2f56cc7da978a...	https://q-cf.bstatic.com/static/css/main_exps_cloudfront.iq_ltr/bc7a32263f515e6b2f56cc7da978aa...	200	stylesheet
22bd4d0fd9d91b682e5cd17e2a2e...	https://q-cf.bstatic.com/static/css/mit_cloudfront.iq_ltr/22bd4d0fd9d91b682e5cd17e2a2e5fe6e95...	200	stylesheet
512328508634457f9bc2f1fb29caa...	https://q-cf.bstatic.com/static/css/raf_cloudfront.iq_ltr/512328508634457f9bc2f1fb29caa73063d5...	200	stylesheet
139eb0de389132fe96104a448faa8...	https://r-cf.bstatic.com/static/css/searchresults_cloudfront.iq_ltr/139eb0de389132fe96104a448faa...	200	stylesheet
f9643a69f02b9c76991392f48a052...	https://r-cf.bstatic.com/static/js/core-deps-inlinedet_cloudfront/f9643a69f02b9c76991392f48a05...	200	script
b7d9d30c56875df3553b561b0a06...	https://r-cf.bstatic.com/static/js/jquery_cloudfront/b7d9d30c56875df3553b561b0a06e5edf66aa9f...	200	script
bc9e73c7108e80a216764778a115...	https://r-cf.bstatic.com/static/js/main_cloudfront/bc9e73c7108e80a216764778a115260cd898e46...	200	script
a21d916ee7e8654fa1fcb34dcafd9...	https://r-cf.bstatic.com/static/js/searchbox_cloudfront/a21d916ee7e8654fa1fcb34dcafd94f834548...	200	script
0ebdfc8fdd8856a4ec9451238a46c...	https://r-cf.bstatic.com/static/js/searchresults_cloudfront/0ebdfc8fdd8856a4ec9451238a46caa9d...	200	script
2a2cd8377620a78eeea106c10810...	https://q-cf.bstatic.com/static/js/tpi_searchresults_cloudfront/2a2cd8377620a78eeea106c108102...	200	script
528359eb9f21194adf8c26f81e07c...	https://q-cf.bstatic.com/static/js/searchresults_slick_cloudfront/528359eb9f21194adf8c26f81e07c...	200	script
f56f7a2e7854715ad5ecc2f07a1a4c...	https://r-cf.bstatic.com/static/js/error_catcher_bec_cloudfront/f56f7a2e7854715ad5ecc2f07a1a4c...	200	script
22615963add19ac6b6d715a97c8d...	https://q-cf.bstatic.com/static/img/b26logo/booking_logo_retina/22615963add19ac6b6d715a97c...	200	png
44d20cd12a233cfc196701b40a8c2...	https://r-cf.bstatic.com/static/img/uc_ajax_loader/44d20cd12a233cfc196701b40a8c2a86faf03cbf.gif	200	gif
e39c170c852301a1817b3d0833be...	https://r-cf.bstatic.com/static/img/flags/24/us/e39c170c852301a1817b3d0833be23f677a2f922.png	200	png
js?id=DC-4228414	https://www.googletagmanager.com/gtag/js?id=DC-4228414	200	script
b700d9e3067c1186a3364012df4fe...	https://r-cf.bstatic.com/static/img/nobg_all_blue_iq/b700d9e3067c1186a3364012df4fe1c48ae6da...	200	png
21218800.webp?k=ab8b1ff031b15...	https://r-cf.bstatic.com/xdata/images/hotel/square200/21218800.webp?k=ab8b1ff031b15abd0ef...	200	webp
25285583.webp?k=5530ded9f5f5d...	https://q-cf.bstatic.com/xdata/images/hotel/square200/25285583.webp?k=5530ded9f5f5dc0c1e8...	200	webp
209932685.webp?k=1356d2a4936...	https://r-cf.bstatic.com/xdata/images/hotel/square200/209932685.webp?k=1356d2a49362756e8...	200	webp
239048570.webp?k=831dfa630e9...	https://r-cf.bstatic.com/xdata/images/hotel/square200/239048570.webp?k=831dfa630e9b8f5571...	200	webp
21077858.webp?k=4fc26ba1d340...	https://q-cf.bstatic.com/xdata/images/hotel/square200/21077858.webp?k=4fc26ba1d34064725b...	200	webp
94908065.webp?k=adcd799e1fecf...	https://q-cf.bstatic.com/xdata/images/hotel/square200/94908065.webp?k=adcd799e1fecf2b28b8...	200	webp
48025380.webp?k=3a0294440e79...	https://r-cf.bstatic.com/xdata/images/hotel/square200/48025380.webp?k=3a0294440e79dd7fd5...	200	webp
1430026.webp?k=b3b015bbb3c4f...	https://r-cf.bstatic.com/xdata/images/hotel/square200/1430026.webp?k=b3b015bbb3c4f674fa35...	200	webp
109947738.webp?k=2291d118d55...	https://r-cf.bstatic.com/xdata/images/hotel/square200/109947738.webp?k=2291d118d551be764...	200	webp
205468501.webp?k=76fd730c1fc1...	https://q-cf.bstatic.com/xdata/images/hotel/square200/205468501.webp?k=76fd730c1fc1435781...	200	webp
188770696.webp?k=6394fca4ea26...	https://q-cf.bstatic.com/xdata/images/hotel/square200/188770696.webp?k=6394fca4ea26f39290...	200	webp
234514036.webp?k=eff47b80219b...	https://r-cf.bstatic.com/xdata/images/hotel/square200/234514036.webp?k=eff47b80219b3d61e8...	200	webp
259988877.webp?k=6b96da2311b...	https://q-cf.bstatic.com/xdata/images/hotel/square200/259988877.webp?k=6b96da2311b2efd29...	200	webp
ab82d9bf871cfe28b1f98ca73ff9c4...	https://q-cf.bstatic.com/static/img/map/cta_material/map_cta_background/ab82d9bf871cfe28b1f...	200	jpeg
630b60fdd032b0b47748700f4d87f...	https://r-cf.bstatic.com/static/img/map/cta_material/map_cta_pin/630b60fdd032b0b47748700f4d...	200	png
js?id=AW-1070314322&l=dataLay...	https://www.googletagmanager.com/gtag/js?id=AW-1070314322&l=dataLayer&cx=c	200	script

188. The Booking Platform can accept an input, and transmit the input with a corresponding input symbolic name to a web service. By way of example, when the Google Maps API accepts an input value (e.g., a hotel name, a zoom-in click, or a click on a nearby hotel) from a user into the API, a corresponding input symbolic name (e.g., a query string or JSON data for the input) is transmitted to the Google Maps web service through an HTTP request protocol, such as a GET method call. The example below demonstrates how a user can search for "East Village Hotel" in the Google Maps API on a property listing. When the user types "East Village Hotel," a query string with parameters including the symbolic name "1s" and associated value "East Village Hotel" is sent to the Google Maps web service to initiate an autocompletion service with an HTTP GET request.



Filter by:

Availability
- Only show available properties

Your Budget

$0 - $59 per night	1
$59 - $118 per night	67
$118 - $178 per night	94
$178 - $237 per night	46
$237 + per night	107

Popular Filters

Breakfast Included	50
Parking	179
Very Good: 8+	183
Pet friendly	111
Hotels	203
Family rooms	154
Apartments	95
Free WiFi	272

Health & safety

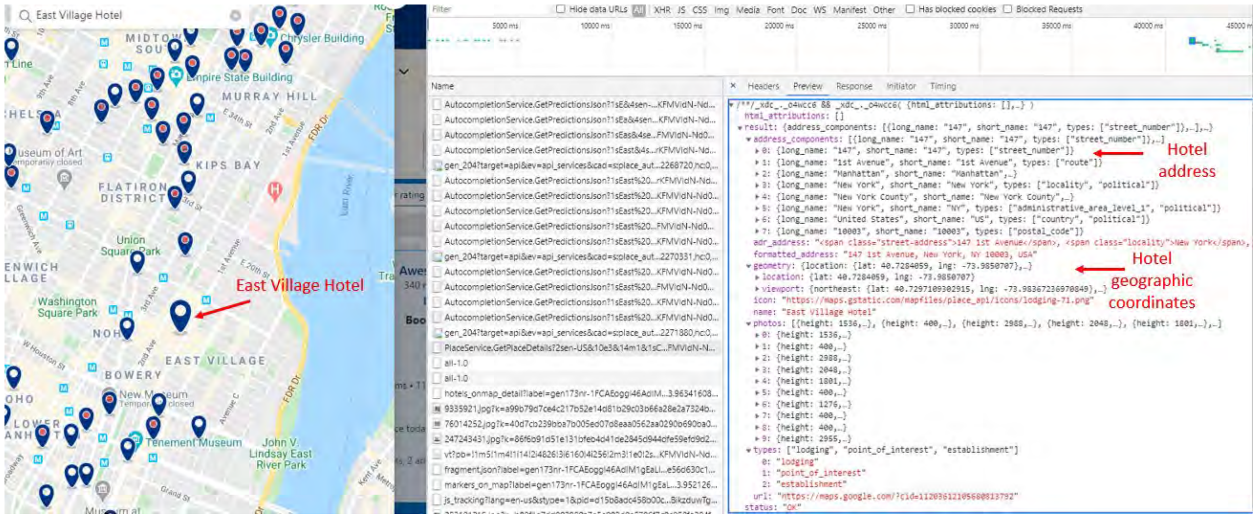
Properties that take health & safety measures	155

Star Rating

1 star	2
2 stars	31
3 stars	117

```
Request Headers
:authority: maps.googleapis.com
:method: GET
:path: /maps/api/place/js/AutocompletionService.GetPredi
sEast%20Village%20Hotel&4sen-US&6m6&1m2&1d40.70163691545
29505715063468&2m2&1d40.7573782113162&2d-73.963416084936
shment&9sgeocode&15e3&20sCFD7F31D-C0B2-43B6-9595-D81666
ozc7pu&21m1&2e1&callback=_xdc_._ypyq8i&key=AIzaSyDXrqUc
HFrKFHVIdN-Nd0w&channel=booking-frontend&token=80084
:scheme: https
accept: */*
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
referer: https://www.booking.com/
sec-fetch-dest: script
sec-fetch-mode: no-cors
sec-fetch-site: cross-site
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) Ap
7.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/53
x-client-data: CJC2yQEIpbbJAQjBtskBCKmdygEIhrXKAQiZtcoBCO
0yBoBCNjvygEIltbKARjsvBoB
Query String Parameters
1sEast Village Hotel:
4sen-US:
6m6:
1m2:
1d40.70163691545737:
2d-74.02950571506346:
2m2:
1d40.7573782113162:
2d-73.96341608493651:
9sestablishment:
9sgeocode:
15e3:
```

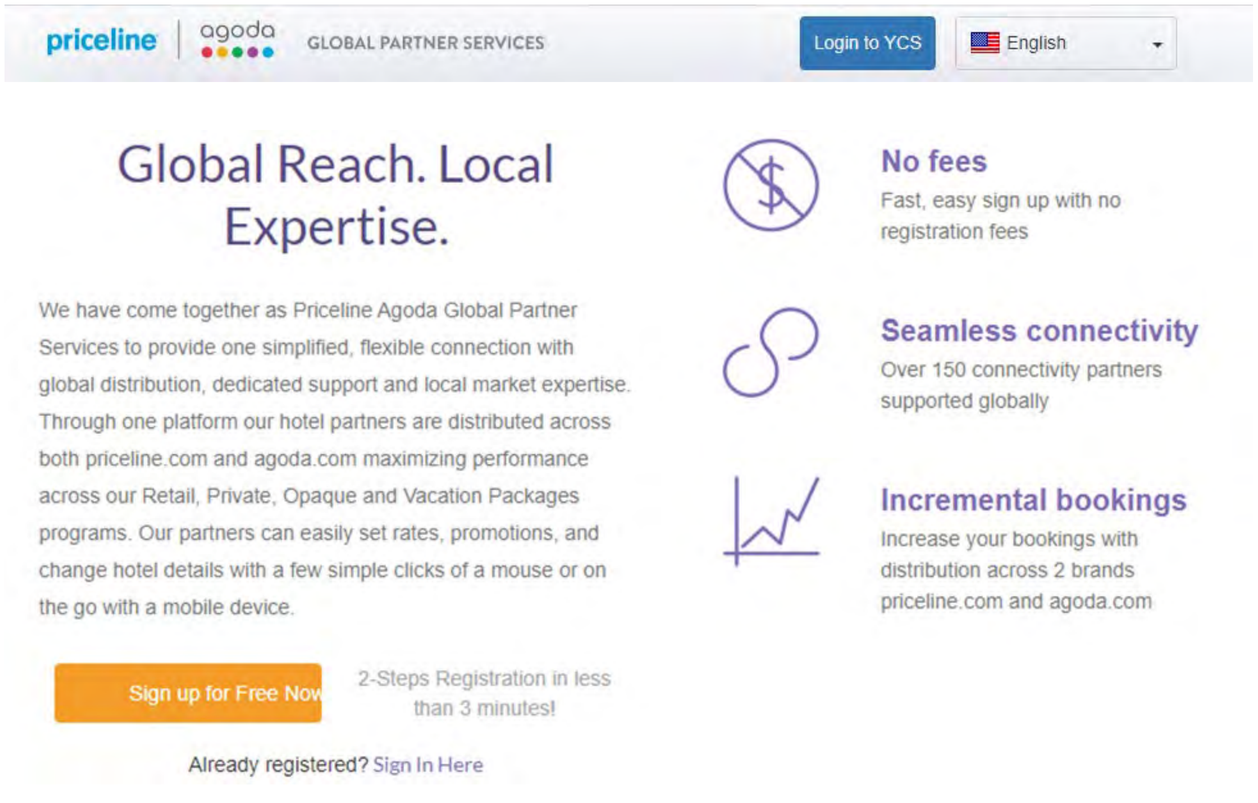
189. The input symbolic name is utilized by the Google Maps web service to generate an output value with an associated output symbolic name (e.g., JSON data for the output). The player then provides instructions to the display of the device to present and display the output in the Google Maps API. For example, when a user searches "East Village Hotel" and selects an autocomplete suggestion from the Google Maps API, the Google Maps web service generates and transmits a corresponding hotel address and geographic coordinates in the format of JSON data to the web browser and displays a pin indicating the location of the selected hotel on the Google Maps API.


East Village Hotel
Hotel address
Hotel geographic coordinates
Headers Preview Response Initiator Timing

```
/**/_xdc_._o4wcc6 && _xdc_._o4wcc6( {html_attributions: [],…} )
  html_attributions: []
 result: {address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…],…}
  address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…]
   0: {long_name: "147", short_name: "147", types: ["street_number"]}
   1: {long_name: "1st Avenue", short_name: "1st Avenue", types: ["route"]}
   2: {long_name: "Manhattan", short_name: "Manhattan",…}
   3: {long_name: "New York", short_name: "New York", types: ["locality", "political"]}
   4: {long_name: "New York County", short_name: "New York County",…}
   5: {long_name: "New York", short_name: "NY", types: ["administrative_area_level_1", "political"]}
   6: {long_name: "United States", short_name: "US", types: ["country", "political"]}
   7: {long_name: "10003", short_name: "10003", types: ["postal_code"]}
  adr_address: "<span class="street-address">147 1st Avenue</span>, <span class="locality">New York</span>,
  formatted_address: "147 1st Avenue, New York, NY 10003, USA"
  geometry: {location: {lat: 40.7284059, lng: -73.9850707},…}
   location: {lat: 40.7284059, lng: -73.9850707}
   viewport: {northeast: {lat: 40.7297109302915, lng: -73.98367236970849},…}
  icon: "https://maps.gstatic.com/mapfiles/place_api/icons/lodging-71.png"
  name: "East Village Hotel"
  photos: [{height: 1536,…}, {height: 400,…}, {height: 2988,…}, {height: 2048,…}, {height: 1801,…},…]
   0: {height: 1536,…}
   1: {height: 400,…}
   2: {height: 2988,…}
   3: {height: 2048,…}
   4: {height: 1801,…}
   5: {height: 400,…}
   6: {height: 1276,…}
   7: {height: 400,…}
   8: {height: 400,…}
   9: {height: 2955,…}
  types: ["lodging", "point_of_interest", "establishment"]
   0: "lodging"
   1: "point_of_interest"
   2: "establishment"
  url: "https://maps.google.com/?cid=11203612105680813792"
 status: "OK"
```

<u>The YCS Platform:</u>

190. The YCS infringers infringe at least claim 12 of the '755 patent through a combination of features in the YCS Platform that collectively practice each claimed limitation of claim 12. By way of example, the YCS infringers provide the YCS Platform for creating property listing websites for hotels and other lodgings where travelers can view and book accommodations.



Global Reach. Local Expertise.

We have come together as Priceline Agoda Global Partner Services to provide one simplified, flexible connection with global distribution, dedicated support and local market expertise. Through one platform our hotel partners are distributed across both priceline.com and agoda.com maximizing performance across our Retail, Private, Opaque and Vacation Packages programs. Our partners can easily set rates, promotions, and change hotel details with a few simple clicks of a mouse or on the go with a mobile device.

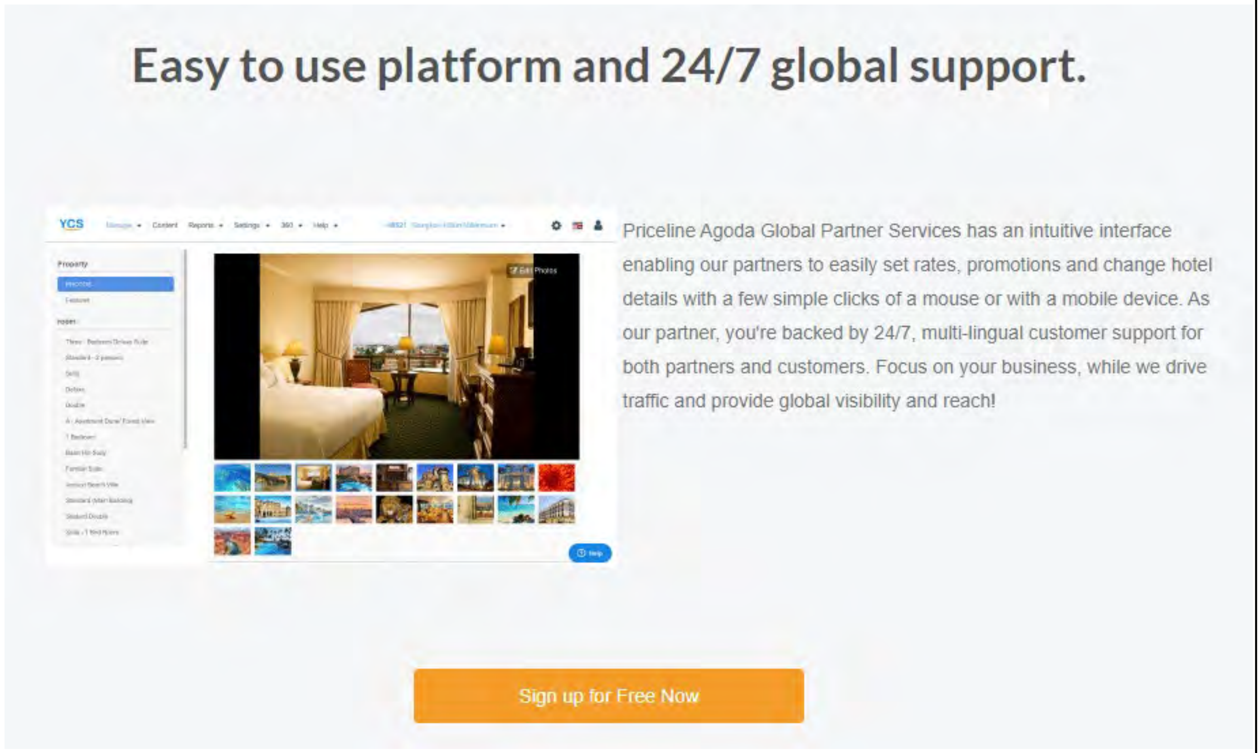
2-Steps Registration in less than 3 minutes!

Already registered? Sign In Here

No fees
Fast, easy sign up with no registration fees

Seamless connectivity
Over 150 connectivity partners supported globally

Incremental bookings
Increase your bookings with distribution across 2 brands priceline.com and agoda.com

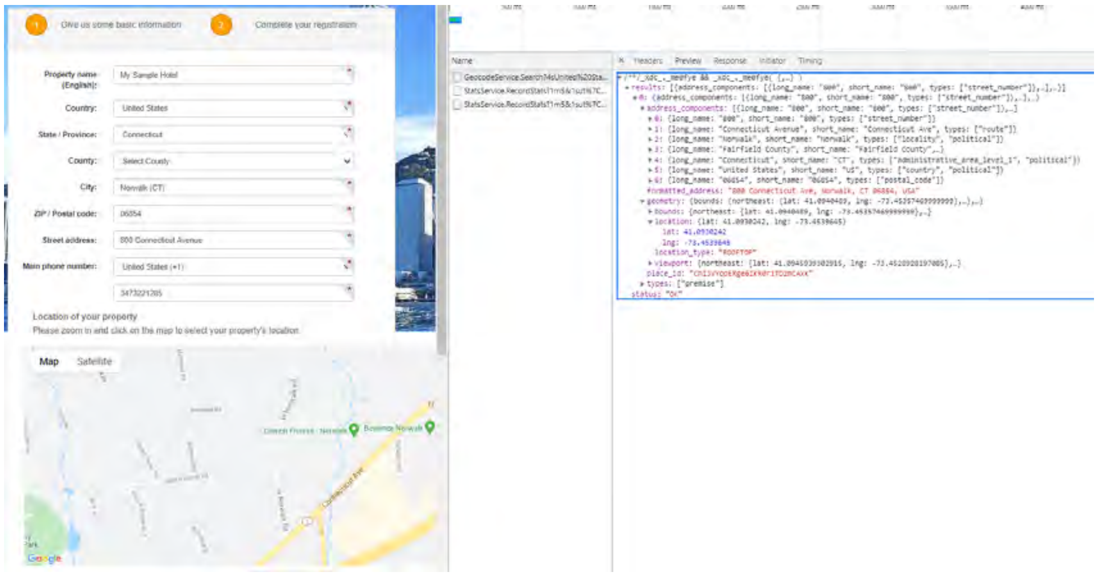


Easy to use platform and 24/7 global support.

Priceline Agoda Global Partner Services has an intuitive interface enabling our partners to easily set rates, promotions and change hotel details with a few simple clicks of a mouse or with a mobile device. As our partner, you're backed by 24/7, multi-lingual customer support for both partners and customers. Focus on your business, while we drive traffic and provide global visibility and reach!

191. On information and belief, the YCS Platform utilizes a registry, located in a server database and/or a CDN, of web components related to inputs and outputs of web services, with each web component including a plurality of corresponding symbolic names for inputs and outputs. Web services that may be integrated on Priceline and Agoda websites include, for example, the Google Maps API.

192. On information and belief, Priceline and Agoda websites utilize JSON data to evoke web components and communicate inputs and outputs for the APIs. JSON data comprise key/value pairs that are essentially character strings (i.e., symbolic names) with no persistent address or pointer. For example, an Agoda property listing website can include a Google Maps API integrated on the website to display the location of the property listing. The Google Maps API utilizes symbolic names (e.g., JSON data) to transfer the property address information and geographic coordinates to the Agoda website. In the example below, "lat" and "lng" are symbolic names associated with the latitude and longitude coordinates of the property address "800 Connecticut Avenue, Norwalk, CT 06854, USA."



1 Give us some basic information
2 Complete your registration
Map
Satellite
Google
Headers
Preview
Response
Initiator
Timing
Name

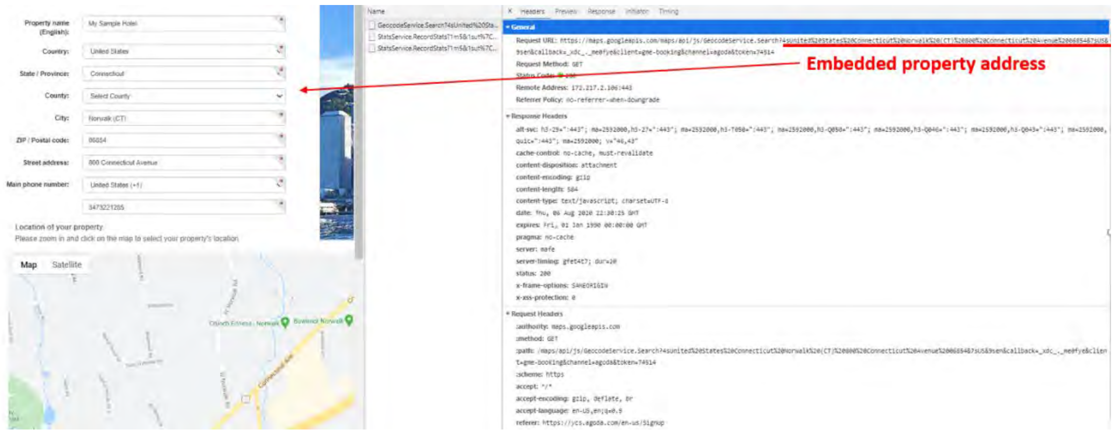
Property name (English):	My Sample Hotel
Country:	United States
State / Province:	Connecticut
County:	Select County
City:	Norwalk (CT)
ZIP / Postal code:	06854
Street address:	800 Connecticut Avenue
Main phone number:	United States (+1)
	3473221205

Location of your property
Please zoom in and click on the map to select your property's location

```
GeocodeService.Search?4sUnited%20Sta...
StatsService.RecordStats?1m5&1sut%7C...
StatsService.RecordStats?1m5&1sut%7C...
```

```
▾/**/_xdc_._me0fye && _xdc_._me0fye( {,…} )
  ▾results: [{address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…],…}]
    ▾0: {address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…],…}
      ▾address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…]
        ▸0: {long_name: "800", short_name: "800", types: ["street_number"]}
        ▸1: {long_name: "Connecticut Avenue", short_name: "Connecticut Ave", types: ["route"]}
        ▸2: {long_name: "Norwalk", short_name: "Norwalk", types: ["locality", "political"]}
        ▸3: {long_name: "Fairfield County", short_name: "Fairfield County",…}
        ▸4: {long_name: "Connecticut", short_name: "CT", types: ["administrative_area_level_1", "political"]}
        ▸5: {long_name: "United States", short_name: "US", types: ["country", "political"]}
        ▸6: {long_name: "06854", short_name: "06854", types: ["postal_code"]}
       formatted_address: "800 Connecticut Ave, Norwalk, CT 06854, USA"
      ▾geometry: {bounds: {northeast: {lat: 41.0940489, lng: -73.45357469999999},…},…}
        ▸bounds: {northeast: {lat: 41.0940489, lng: -73.45357469999999},…}
        ▾location: {lat: 41.0930242, lng: -73.4539645}
            lat: 41.0930242
            lng: -73.4539645
          location_type: "ROOFTOP"
        ▸viewport: {northeast: {lat: 41.0945039302915, lng: -73.4520928197085},…}
       place_id: "ChIJVYQOERge6IkRWr1TD2MCAXk"
      ▸types: ["premise"]
   status: "OK"
```

193. The registry for the YCS Platform also includes addresses for the web services where the input symbolic names and output symbolic names can be sent to and received from. For example, the screenshot below shows a request URL including an address of the Google Maps web service. In particular, the user-input property address "800 Connecticut Avenue. Norwalk, CT 06854" is embedded in the request URL and sent to the Google Maps API via an HTTP GET request for Geocoding service. The Geocoding service in the Google Maps API then converts the property address to geographic coordinates and sends the geographic coordinates with other property information in the format of JSON data to the Agoda website. The geographic coordinates and the user input address are stored in Booking's database for further use.

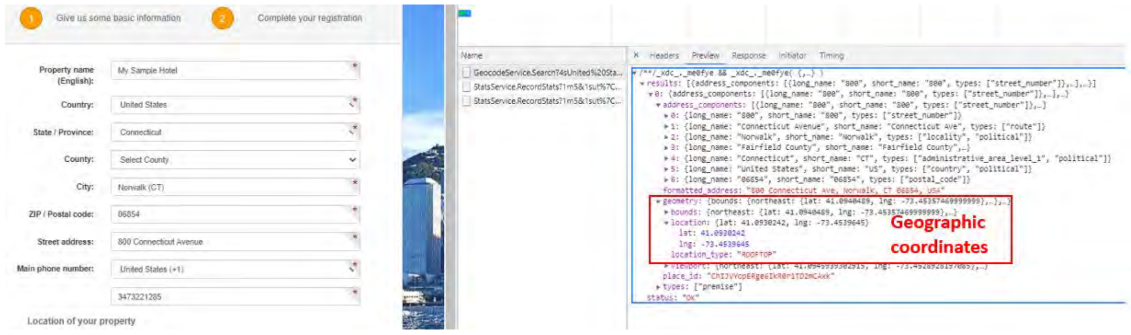


Property name (English):	My Sample Hotel
Country:	United States
State / Province:	Connecticut
County:	Select County
City:	Norwalk (CT)
ZIP / Postal code:	06854
Street address:	800 Connecticut Avenue
Main phone number:	United States (+1)
	3473221205

Location of your property
Please zoom in and click on the map to select your property's location

```
GeocodeService.Search?4sUnited%20Sta...
StatsService.RecordStats?1m5&1sut%7C...
StatsService.RecordStats?1m5&1sut%7C...
```

```
▾General
  Request URL: https://maps.googleapis.com/maps/api/js/GeocodeService.Search?4sUnited%20States%20Connecticut%20Norwalk%20(CT)%20800%20Connecticut%20Avenue%2006854&7sUS&
  9sen&callback=_xdc_._me0fye&client=gme-booking&channel=agoda&token=74514
  Request Method: GET
  Status Code: 200
  Remote Address: 172.217.2.106:443
  Referrer Policy: no-referrer-when-downgrade
▾Response Headers
  alt-svc: h3-29=":443"; ma=2592000,h3-27=":443"; ma=2592000,h3-T050=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,
  quic=":443"; ma=2592000; v="46,43"
  cache-control: no-cache, must-revalidate
  content-disposition: attachment
  content-encoding: gzip
  content-length: 584
  content-type: text/javascript; charset=UTF-8
  date: Thu, 06 Aug 2020 22:30:25 GMT
  expires: Fri, 01 Jan 1990 00:00:00 GMT
  pragma: no-cache
  server: mafe
  server-timing: gfet4t7; dur=10
  status: 200
  x-frame-options: SAMEORIGIN
  x-xss-protection: 0
▾Request Headers
  :authority: maps.googleapis.com
  :method: GET
  :path: /maps/api/js/GeocodeService.Search?4sUnited%20States%20Connecticut%20Norwalk%20(CT)%20800%20Connecticut%20Avenue%2006854&7sUS&9sen&callback=_xdc_._me0fye&clien
  t=gme-booking&channel=agoda&token=74514
  :scheme: https
  accept: */*
  accept-encoding: gzip, deflate, br
  accept-language: en-US,en;q=0.9
  referer: https://ycs.agoda.com/en-us/Signup
```



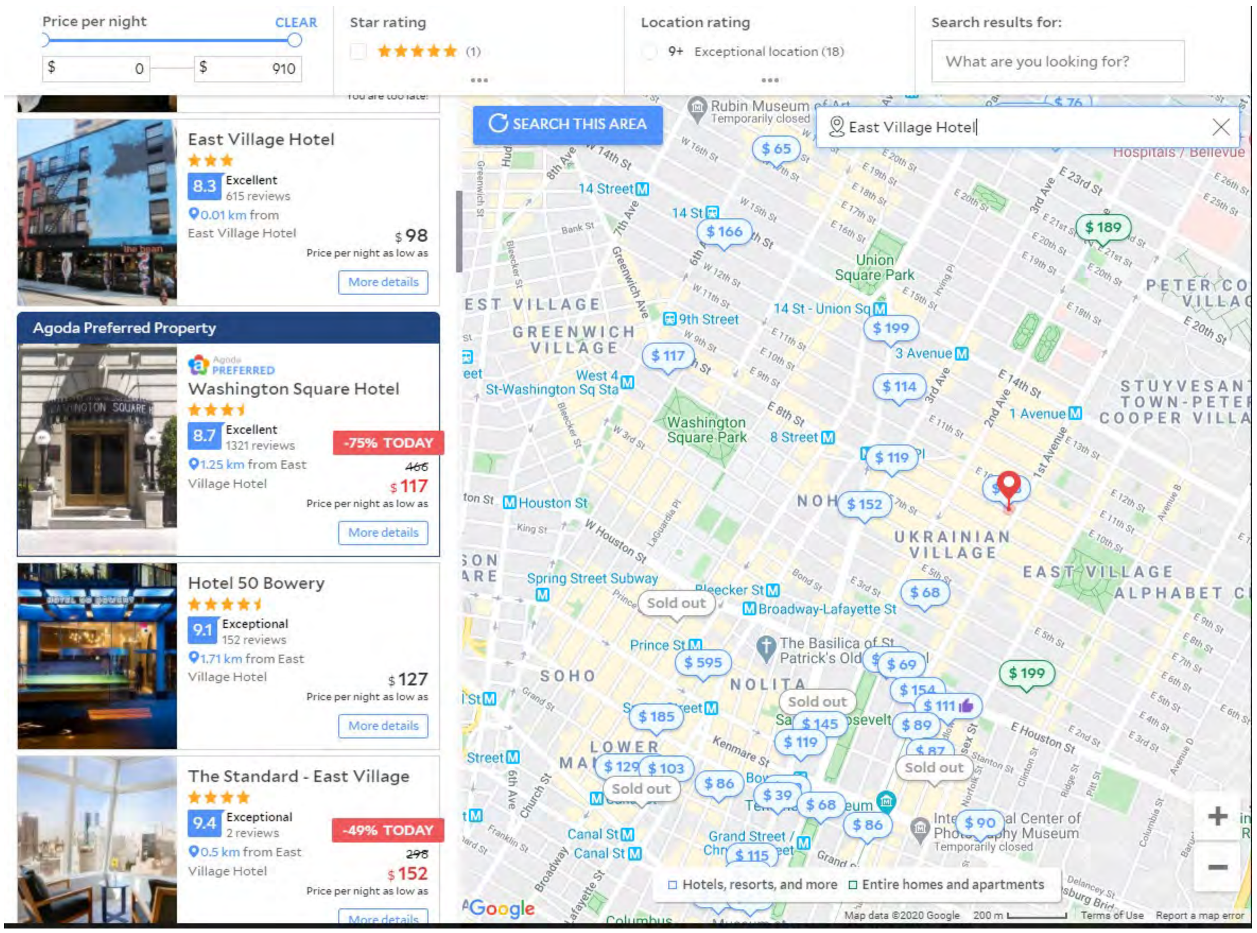
Property name (English):	My Sample Hotel
Country:	United States
State / Province:	Connecticut
County:	Select County
City:	Norwalk (CT)
ZIP / Postal code:	06854
Street address:	800 Connecticut Avenue
Main phone number:	United States (+1)
	3473221285

Location of your property

```
GeocodeService.Search?4sUnited%20Sta...
StatsService.RecordStats?1m5&1sut%7C...
StatsService.RecordStats?1m5&1sut%7C...
```

```
▾ /**/_xdc_._me0fye && _xdc_._me0fye( {,…} )
  ▾ results: [{address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…],…}]
    ▾ 0: {address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…],…}
      ▾ address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…]
        ▸ 0: {long_name: "800", short_name: "800", types: ["street_number"]}
        ▸ 1: {long_name: "Connecticut Avenue", short_name: "Connecticut Ave", types: ["route"]}
        ▸ 2: {long_name: "Norwalk", short_name: "Norwalk", types: ["locality", "political"]}
        ▸ 3: {long_name: "Fairfield County", short_name: "Fairfield County",…}
        ▸ 4: {long_name: "Connecticut", short_name: "CT", types: ["administrative_area_level_1", "political"]}
        ▸ 5: {long_name: "United States", short_name: "US", types: ["country", "political"]}
        ▸ 6: {long_name: "06854", short_name: "06854", types: ["postal_code"]}
        formatted_address: "800 Connecticut Ave, Norwalk, CT 06854, USA"
      ▾ geometry: {bounds: {northeast: {lat: 41.0940489, lng: -73.45357469999999},…},…}
        ▸ bounds: {northeast: {lat: 41.0940489, lng: -73.45357469999999},…}
        ▾ location: {lat: 41.0930242, lng: -73.4539645}
            lat: 41.0930242
            lng: -73.4539645
          location_type: "ROOFTOP"
        ▸ viewport: {northeast: {lat: 41.0945939302915, lng: -73.4526928197085},…}
        place_id: "ChIJVVopERge6IkR0r1TD2mCAxk"
      ▸ types: ["premise"]
    status: "OK"
```

194. The YCS Platform defines a UI object corresponding to a web component for an input or output of a web service, for presentation on the display. For example, when the Google Maps API is embedded onto a Priceline or Agoda property listing website, UI objects for web components such as the zoom-in and zoom-out buttons, a search field, a search button, a slide bar for "Price per night," a check box for "Star rating," and a radio button for "Location rating" are automatically selected by the YCS Platform as preferred UI objects.


Price per night CLEAR
$ 0 $ 910

Star rating
★★★★★ (1)

Location rating
9+ Exceptional location (18)

Search results for:
What are you looking for?

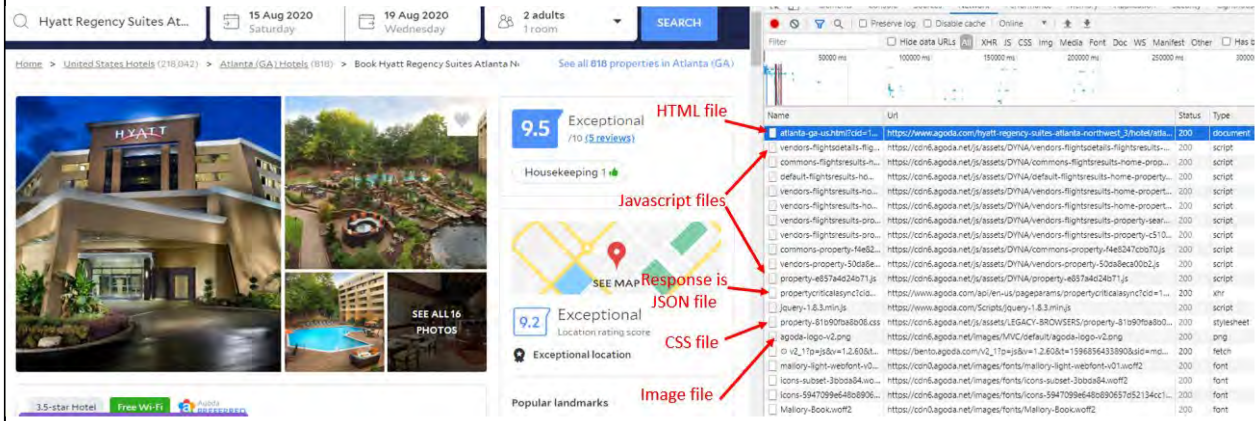
East Village Hotel
★★★
8.3 Excellent
615 reviews
0.01 km from East Village Hotel
$98
Price per night as low as
More details

Agoda Preferred Property
Agoda PREFERRED
Washington Square Hotel
★★★½
8.7 Excellent
1321 reviews
-75% TODAY
1.25 km from East Village Hotel
~~466~~
$117
Price per night as low as
More details

Hotel 50 Bowery
★★★★½
9.1 Exceptional
152 reviews
1.71 km from East Village Hotel
$127
Price per night as low as
More details

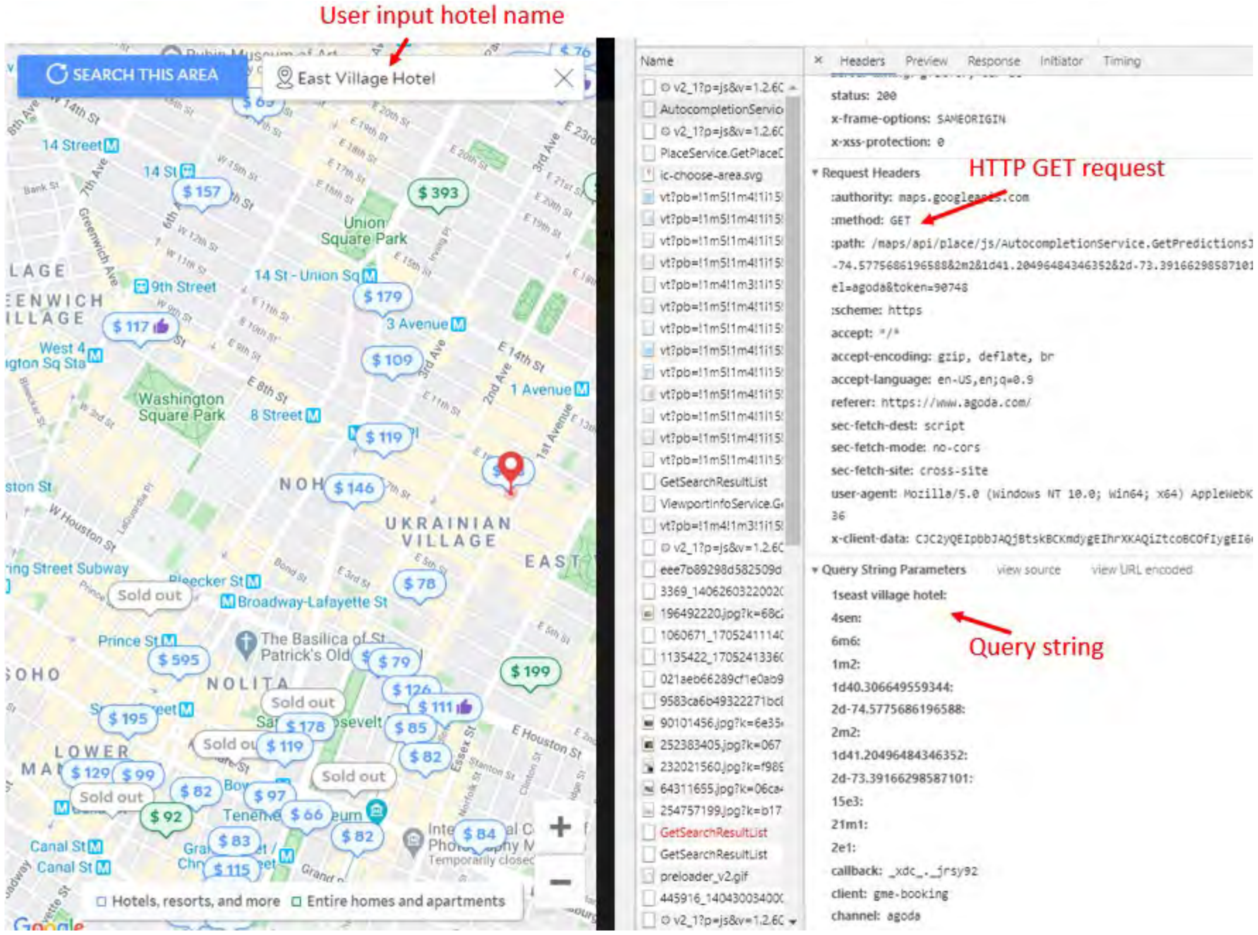
The Standard - East Village
★★★★
9.4 Exceptional
2 reviews
-49% TODAY
0.5 km from East Village Hotel
~~298~~
$152
Price per night as low as
More details

195. When a UI object is defined, a symbolic name (e.g., JSON data) from the web component corresponding to the defined UI object is selected and associated with the defined UI object. The YCS Platform then produces a device-independent application that includes the JSON data, as well as standard HTML, CSS and Javascript code. The application is executed on a device together with a device-dependent player, such as a device-dependent code for a browser engine for a specific browser, or an application or operating system for a particular kind of device. *See, e.g.*, *Shopify Inc. v. Express Mobile, Inc.*, Case No. 1:19-cv-00439-RGA, D. Del., D.I. 137. In the example below, the YCS Platform converts HTML, CSS, Javascript, JSON, and other files into an active property listing website for “Hyatt Regency Suites Atlanta.”


Hyatt Regency Suites At...
15 Aug 2020 Saturday
19 Aug 2020 Wednesday
2 adults 1 room
SEARCH
Home > United States Hotels (218,042) > Atlanta (GA) Hotels (818) > Book Hyatt Regency Suites Atlanta N
See all 818 properties in Atlanta (GA)
9.5 Exceptional /10 (5 reviews)
Housekeeping 1
SEE MAP
9.2 Exceptional Location rating score
Exceptional location
Popular landmarks
SEE ALL 16 PHOTOS
3.5-star Hotel
Free Wi-Fi
HTML file
Javascript files
Response is JSON file
CSS file
Image file
Preserve log
Disable cache
Online
Filter
Hide data URLs
All XHR JS CSS Img Media Font Doc WS Manifest Other
50000 ms 100000 ms 150000 ms 200000 ms 250000 ms

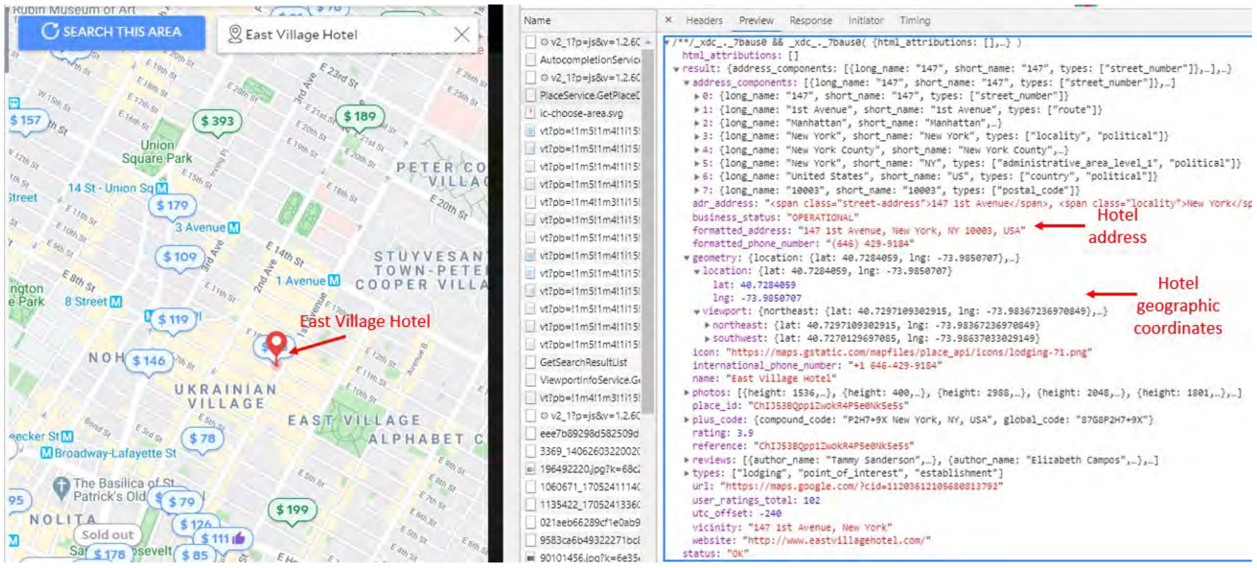
Name	Url	Status	Type
atlanta-ga-us.html?cid=1...	https://www.agoda.com/hyatt-regency-suites-atlanta-northwest_3/hotel/atla...	200	document
vendors-flightsdetails-flig...	https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsdetails-flightsresults-...	200	script
commons-flightsresults-h...	https://cdn6.agoda.net/js/assets/DYNA/commons-flightsresults-home-prop...	200	script
default-flightsresults-ho...	https://cdn6.agoda.net/js/assets/DYNA/default-flightsresults-home-property...	200	script
vendors-flightsresults-ho...	https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsresults-home-propert...	200	script
vendors-flightsresults-ho...	https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsresults-home-propert...	200	script
vendors-flightsresults-pro...	https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsresults-property-sear...	200	script
vendors-flightsresults-pro...	https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsresults-property-c510...	200	script
commons-property-f4e82...	https://cdn6.agoda.net/js/assets/DYNA/commons-property-f4e8247cbb70.js	200	script
vendors-property-50da8e...	https://cdn6.agoda.net/js/assets/DYNA/vendors-property-50da8eca00b2.js	200	script
property-e857a4d24b71.js	https://cdn6.agoda.net/js/assets/DYNA/property-e857a4d24b71.js	200	script
propertycriticalasync?cid...	https://www.agoda.com/api/en-us/pageparams/propertycriticalasync?cid=1...	200	xhr
jquery-1.8.3.min.js	https://www.agoda.com/Scripts/jquery-1.8.3.min.js	200	script
property-81b90fba8b08.css	https://cdn6.agoda.net/js/assets/LEGACY-BROWSERS/property-81b90fba8b0...	200	stylesheet
agoda-logo-v2.png	https://cdn6.agoda.net/images/MVC/default/agoda-logo-v2.png	200	png
v2_1?p=js&v=1.2.60&t...	https://bento.agoda.com/v2_1?p=js&v=1.2.60&t=1596856433890&sid=md...	200	fetch
mallory-light-webfont-v0...	https://cdn0.agoda.net/images/fonts/mallory-light-webfont-v01.woff2	200	font
icons-subset-3bbda84.wo...	https://cdn6.agoda.net/images/fonts/icons-subset-3bbda84.woff2	200	font
icons-5947099e648b8906...	https://cdn6.agoda.net/images/fonts/icons-5947099e648b890657d52134cc1...	200	font
Mallory-Book.woff2	https://cdn0.agoda.net/images/fonts/Mallory-Book.woff2	200	font

196. The YCS Platform is configured to accept an input, and transmit the input with a corresponding input symbolic name to a web service. By way of example, when the Google Maps API accepts an input value (e.g., an input hotel name, a zoom-in click, or a click on the "Explore" tab) from a user, a corresponding input symbolic name (e.g., a query string or JSON data for the input) is transmitted to the Google Maps web service through an HTTP request protocol, such as a GET method call. The example below demonstrates a user searching for "East Village Hotel" on the Google Maps API on an Agoda website. When the user types "East Village Hotel," a query string with parameters including the symbolic name "1s" and the associated value "East Village Hotel" is sent to the Google Maps web service to initiate an autocompletion service with an HTTP GET request.



```
status: 200
x-frame-options: SAMEORIGIN
x-xss-protection: 0
▾ Request Headers
:authority: maps.googleapis.com
:method: GET
:path: /maps/api/place/js/AutocompletionService.GetPredictionsJ
-74.5775686196588&2m2&1d41.20496484346352&2d-73.39166298587101
el=agoda&token=90748
:scheme: https
accept: */*
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
referer: https://www.agoda.com/
sec-fetch-dest: script
sec-fetch-mode: no-cors
sec-fetch-site: cross-site
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebK
36
x-client-data: CJC2yQEIpbbJAQjBtskBCKmdygEIhrXKAQiZtcoBCOfIygEI6
▾ Query String Parameters    view source    view URL encoded
1seast village hotel:
4sen:
6m6:
1m2:
1d40.306649559344:
2d-74.5775686196588:
2m2:
1d41.20496484346352:
2d-73.39166298587101:
15e3:
21m1:
2e1:
callback: _xdc_._jrsy92
client: gme-booking
channel: agoda
```

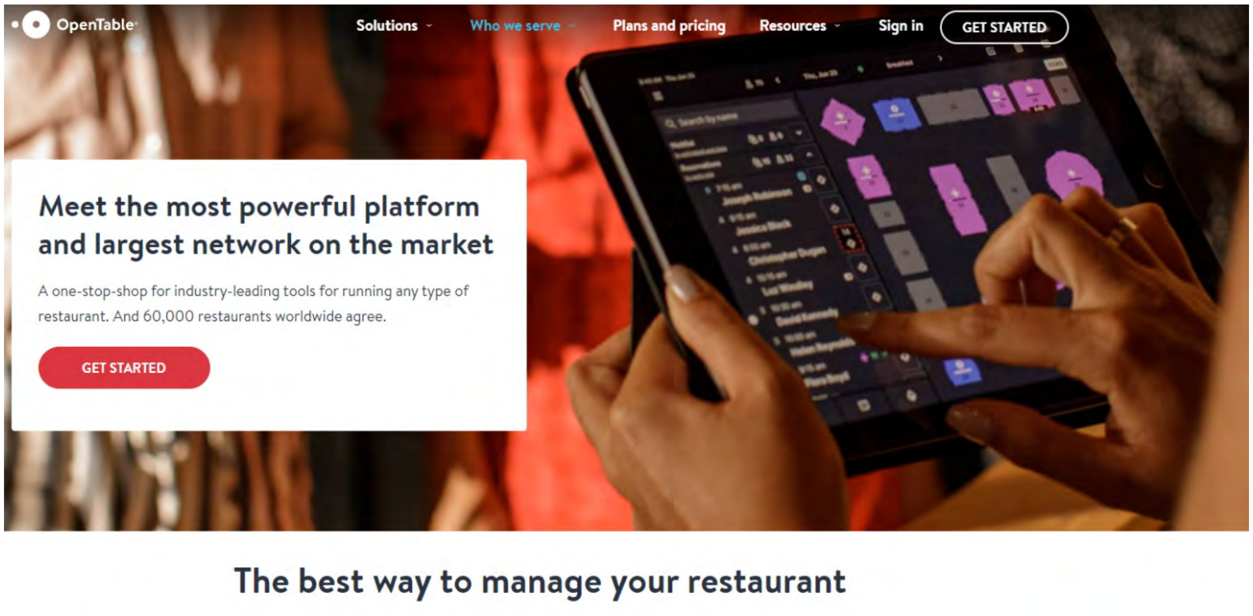
197. The input symbolic name is utilized by the Google Map web service to generate an output value with an associated output symbolic name (e.g., JSON data for the output). The player then provides instructions to the display of the device to present the output value in the API. For example, when a user searches "East Village Hotel" and selects an autocomplete suggestion from the Google Maps API, the Google Maps web service generates and transmits a corresponding hotel address and geographic coordinates in the format of JSON data to the web browser and displays a pin indicating the location of the hotel on the Google Maps API.



```
/**/_xdc_._7baus0 && _xdc_._7baus0( {html_attributions: [],…} )
  html_attributions: []
 result: {address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…],…}
  address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…]
   0: {long_name: "147", short_name: "147", types: ["street_number"]}
   1: {long_name: "1st Avenue", short_name: "1st Avenue", types: ["route"]}
   2: {long_name: "Manhattan", short_name: "Manhattan",…}
   3: {long_name: "New York", short_name: "New York", types: ["locality", "political"]}
   4: {long_name: "New York County", short_name: "New York County",…}
   5: {long_name: "New York", short_name: "NY", types: ["administrative_area_level_1", "political"]}
   6: {long_name: "United States", short_name: "US", types: ["country", "political"]}
   7: {long_name: "10003", short_name: "10003", types: ["postal_code"]}
  adr_address: "<span class="street-address">147 1st Avenue</span>, <span class="locality">New York</sp
  business_status: "OPERATIONAL"
  formatted_address: "147 1st Avenue, New York, NY 10003, USA"
  formatted_phone_number: "(646) 429-9184"
  geometry: {location: {lat: 40.7284059, lng: -73.9850707},…}
   location: {lat: 40.7284059, lng: -73.9850707}
     lat: 40.7284059
     lng: -73.9850707
   viewport: {northeast: {lat: 40.7297109302915, lng: -73.98367236970849},…}
    northeast: {lat: 40.7297109302915, lng: -73.98367236970849}
    southwest: {lat: 40.7270129697085, lng: -73.98637033029149}
  icon: "https://maps.gstatic.com/mapfiles/place_api/icons/lodging-71.png"
  international_phone_number: "+1 646-429-9184"
  name: "East Village Hotel"
  photos: [{height: 1536,…}, {height: 400,…}, {height: 2988,…}, {height: 2048,…}, {height: 1801,…},…]
  place_id: "ChIJ53BQpp1ZwokR4P5e0Nk5e5s"
  plus_code: {compound_code: "P2H7+9X New York, NY, USA", global_code: "87G8P2H7+9X"}
  rating: 3.9
  reference: "ChIJ53BQpp1ZwokR4P5e0Nk5e5s"
  reviews: [{author_name: "Tammy Sanderson",…}, {author_name: "Elizabeth Campos",…},…]
  types: ["lodging", "point_of_interest", "establishment"]
  url: "https://maps.google.com/?cid=11203612105680813792"
  user_ratings_total: 102
  utc_offset: -240
  vicinity: "147 1st Avenue, New York"
  website: "http://www.eastvillagehotel.com/"
 status: "OK"
```

The OpenTable Platform:

198. OpenTable infringes at least claim 12 of the '755 patent through a combination of features in the OpenTable Platform that collectively practice each limitation of claim 12. By way of example, OpenTable provides the OpenTable Platform, a browser-based platform for, *inter alia*, creating restaurant profile websites that can be displayed on a display of a device.

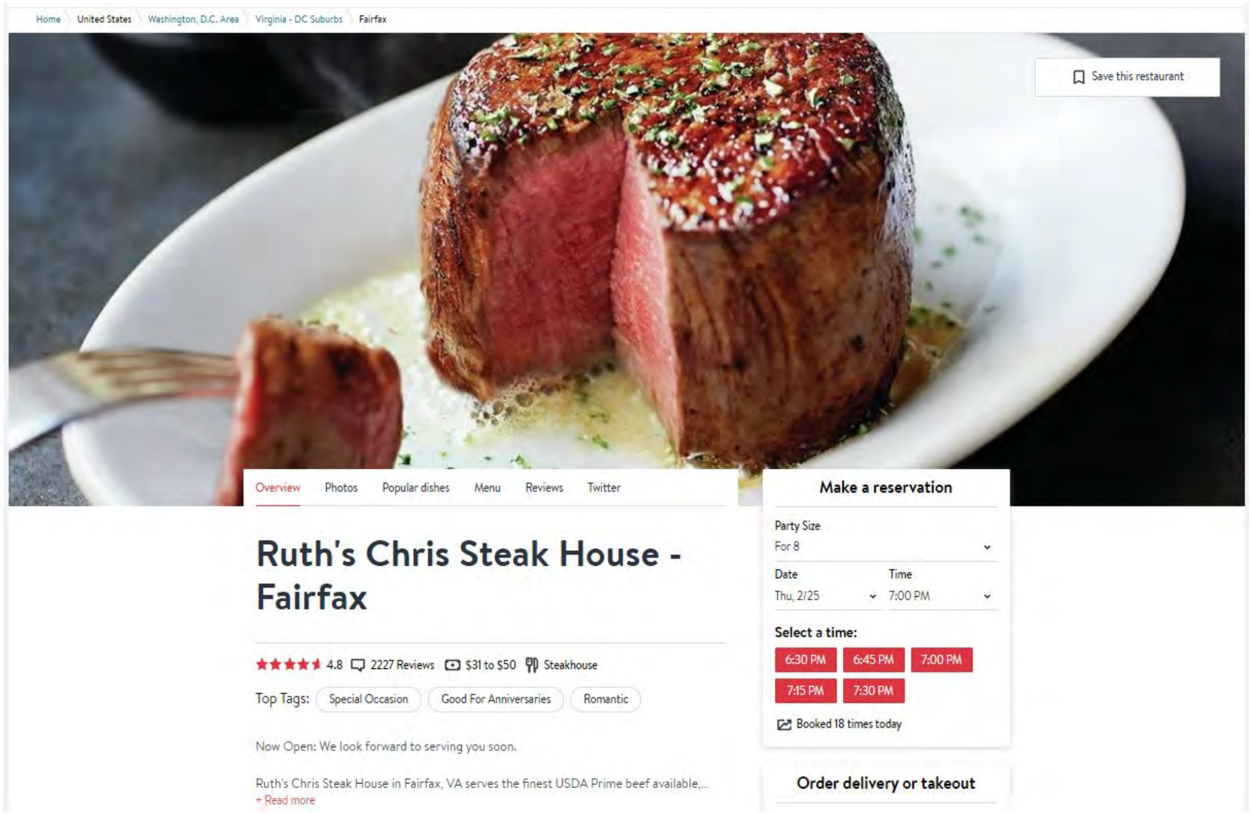


Meet the most powerful platform and largest network on the market

A one-stop-shop for industry-leading tools for running any type of restaurant. And 60,000 restaurants worldwide agree.

The best way to manage your restaurant

https://restaurant.opentable.com/why-opentable/



Ruth's Chris Steak House - Fairfax

4.8 2227 Reviews $31 to $50 Steakhouse

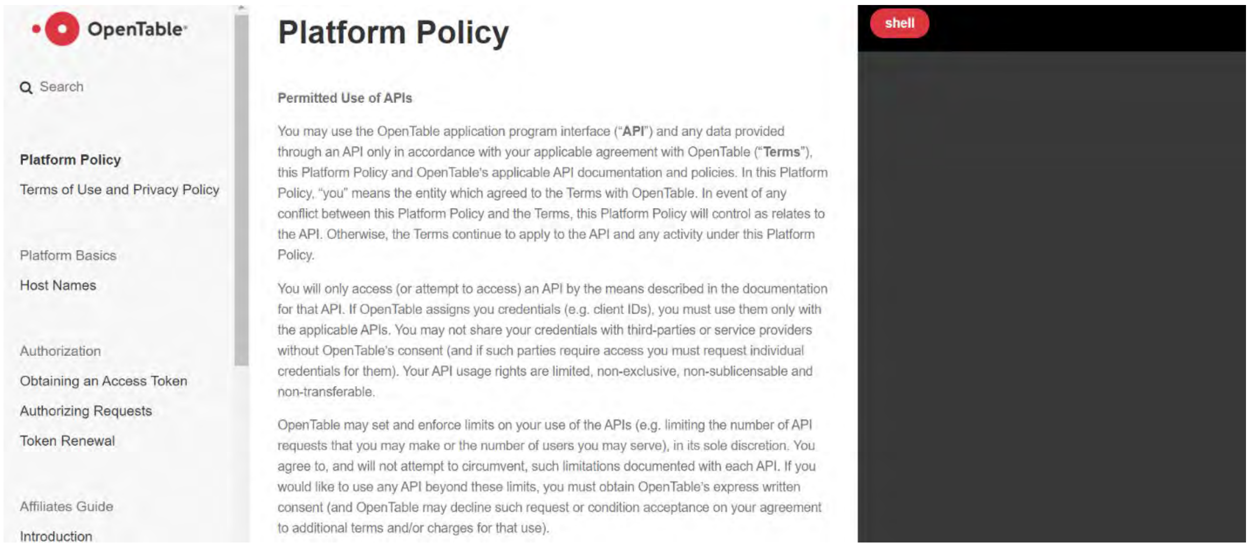
Top Tags: Special Occasion Good For Anniversaries Romantic

Now Open: We look forward to serving you soon.

Ruth's Chris Steak House in Fairfax, VA serves the finest USDA Prime beef available,...
+ Read more

https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8eae-7ada9d88b68e&corrid=dfc6cee2-c8f8-4054-8eae-7ada9d88b68e&avt=eyJ2IjoyLCJtIjoxLCJwIjowLCJzIjowLCJuIjowfQ

199. On information and belief, the OpenTable Platform uses a registry of symbolic names for evoking web components for inputs and outputs of web services on the property listing websites. The OpenTable Platform provides various web services through the OpenTable application programming interface (API). A client can use the OpenTable API to access the OpenTable Platform for data and services, such as make a reservation. In particular, the data is sent and received in JSON format.



Platform Policy

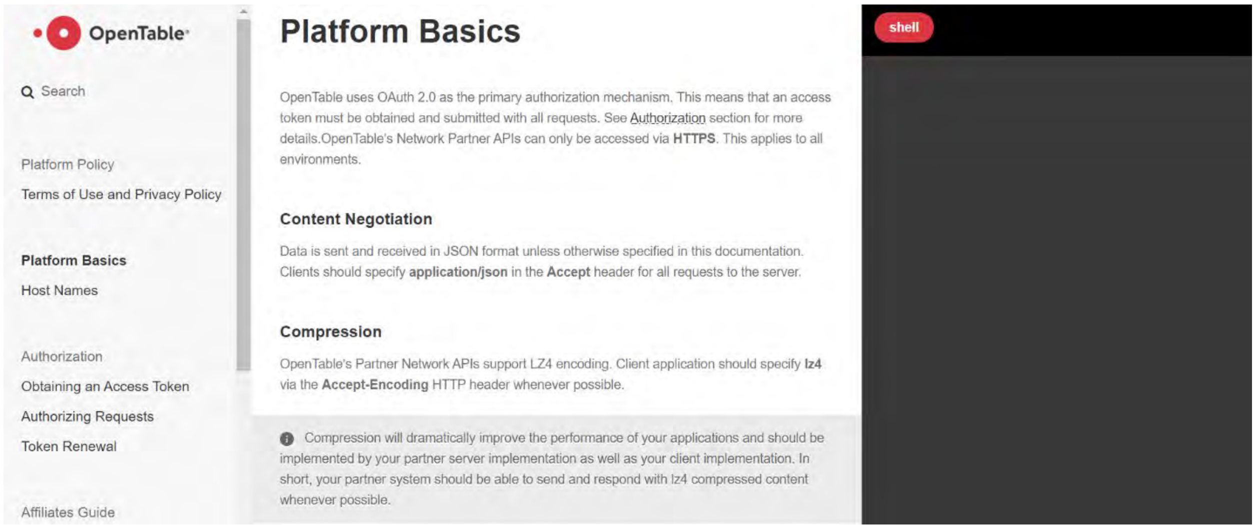
Permitted Use of APIs

You may use the OpenTable application program interface ("**API**") and any data provided through an API only in accordance with your applicable agreement with OpenTable ("**Terms**"), this Platform Policy and OpenTable's applicable API documentation and policies. In this Platform Policy, "you" means the entity which agreed to the Terms with OpenTable. In event of any conflict between this Platform Policy and the Terms, this Platform Policy will control as relates to the API. Otherwise, the Terms continue to apply to the API and any activity under this Platform Policy.

You will only access (or attempt to access) an API by the means described in the documentation for that API. If OpenTable assigns you credentials (e.g. client IDs), you must use them only with the applicable APIs. You may not share your credentials with third-parties or service providers without OpenTable's consent (and if such parties require access you must request individual credentials for them). Your API usage rights are limited, non-exclusive, non-sublicensable and non-transferable.

OpenTable may set and enforce limits on your use of the APIs (e.g. limiting the number of API requests that you may make or the number of users you may serve), in its sole discretion. You agree to, and will not attempt to circumvent, such limitations documented with each API. If you would like to use any API beyond these limits, you must obtain OpenTable's express written consent (and OpenTable may decline such request or condition acceptance on your agreement to additional terms and/or charges for that use).

https://platform.opentable.com/documentation/#platform-policy



Platform Basics

OpenTable uses OAuth 2.0 as the primary authorization mechanism. This means that an access token must be obtained and submitted with all requests. See Authorization section for more details.OpenTable's Network Partner APIs can only be accessed via **HTTPS**. This applies to all environments.

Content Negotiation

Data is sent and received in JSON format unless otherwise specified in this documentation. Clients should specify **application/json** in the **Accept** header for all requests to the server.

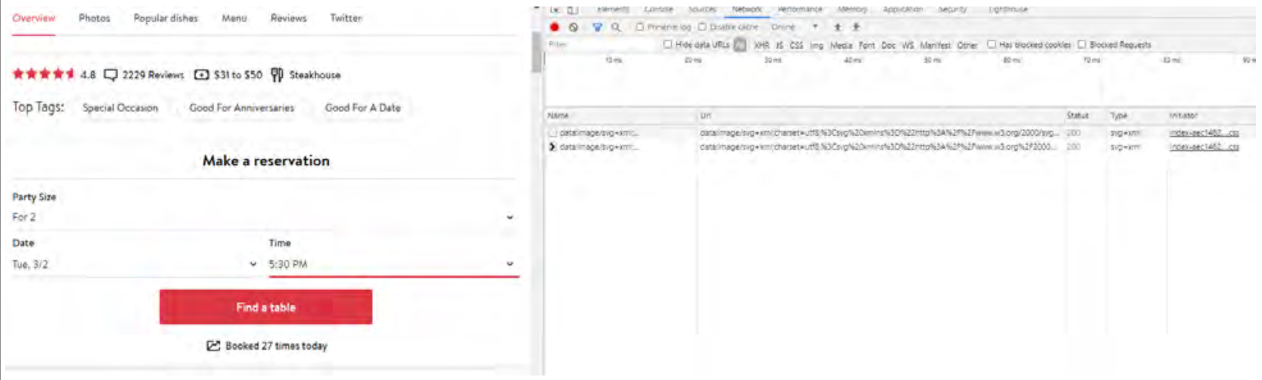
Compression

OpenTable's Partner Network APIs support LZ4 encoding. Client application should specify **lz4** via the **Accept-Encoding** HTTP header whenever possible.

Compression will dramatically improve the performance of your applications and should be implemented by your partner server implementation as well as your client implementation. In short, your partner system should be able to send and respond with lz4 compressed content whenever possible.

https://platform.opentable.com/documentation/#platform-basics

200. By way of example, when a client makes a reservation on a restaurant profile, the OpenTable Platform can display the available time slots as outputs in response to the client's input. In the example below, when "Find a table" button is pressed, user input party size, date and time are sent to the OpenTable API via an HTTP POST request for the OpenTable's reservation web service, which in turn creates five available time slots for the restaurant reservation.


Overview Photos Popular dishes Menu Reviews Twitter

★★★★★ 4.8 2229 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Good For A Date

Make a reservation

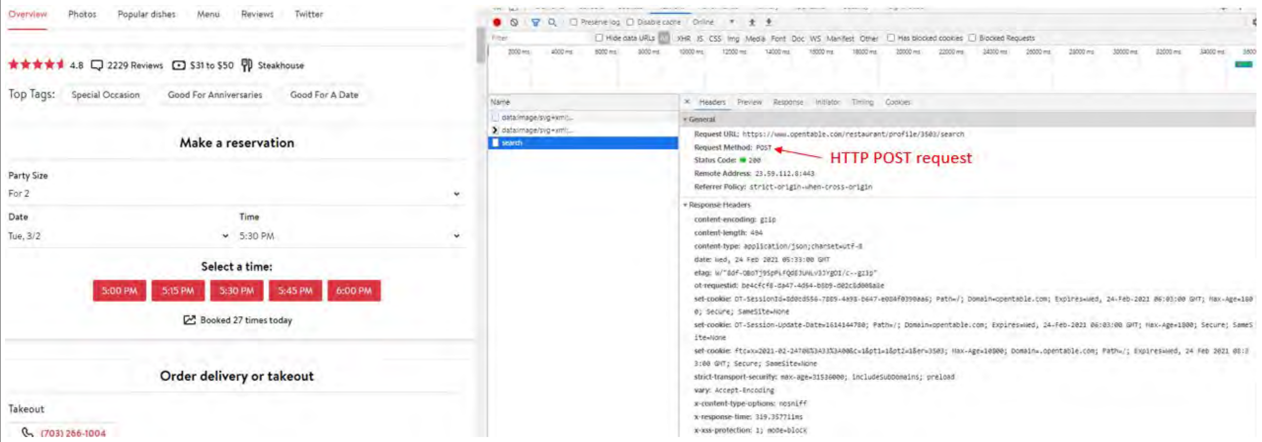
Party Size
For 2

Date Tue, 3/2
Time 5:30 PM

Find a table

Booked 27 times today

```
Name                     Url                                                                                  Status  Type     Initiator
data:image/svg+xml;...   data:image/svg+xml;charset=utf8,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org/2000/svg...  200  svg+xml  index-aec1482...css
data:image/svg+xml;...   data:image/svg+xml;charset=utf8,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000...    200  svg+xml  index-aec1482...css
```


Overview Photos Popular dishes Menu Reviews Twitter

★★★★★ 4.8 2229 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Good For A Date

Make a reservation

Party Size
For 2

Date Tue, 3/2
Time 5:30 PM

Select a time:
5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM

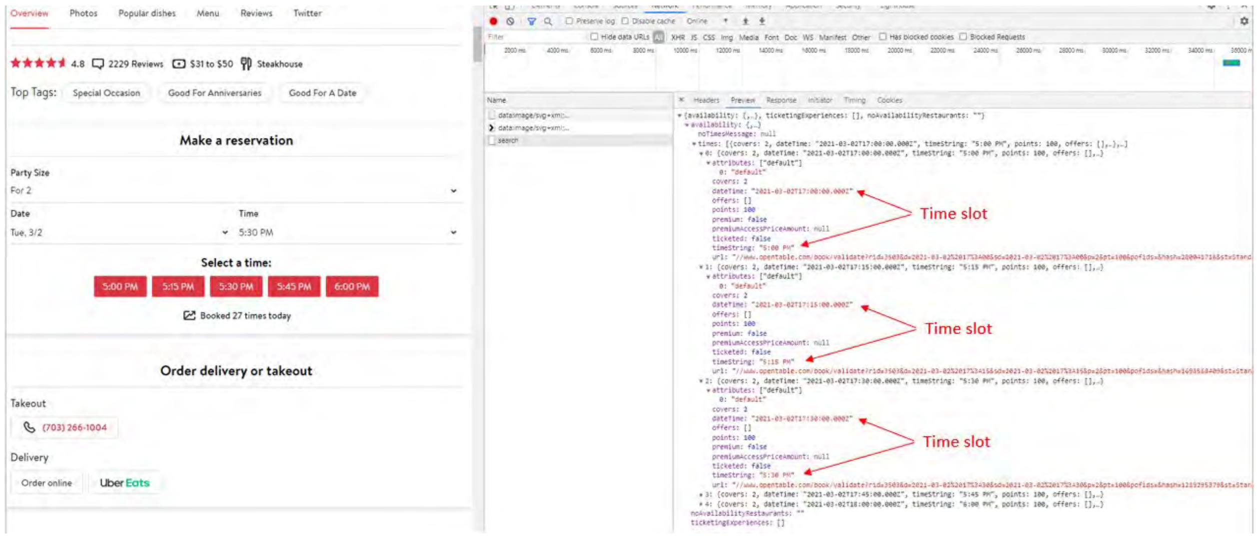
Booked 27 times today

Order delivery or takeout

Takeout
(703) 266-1004

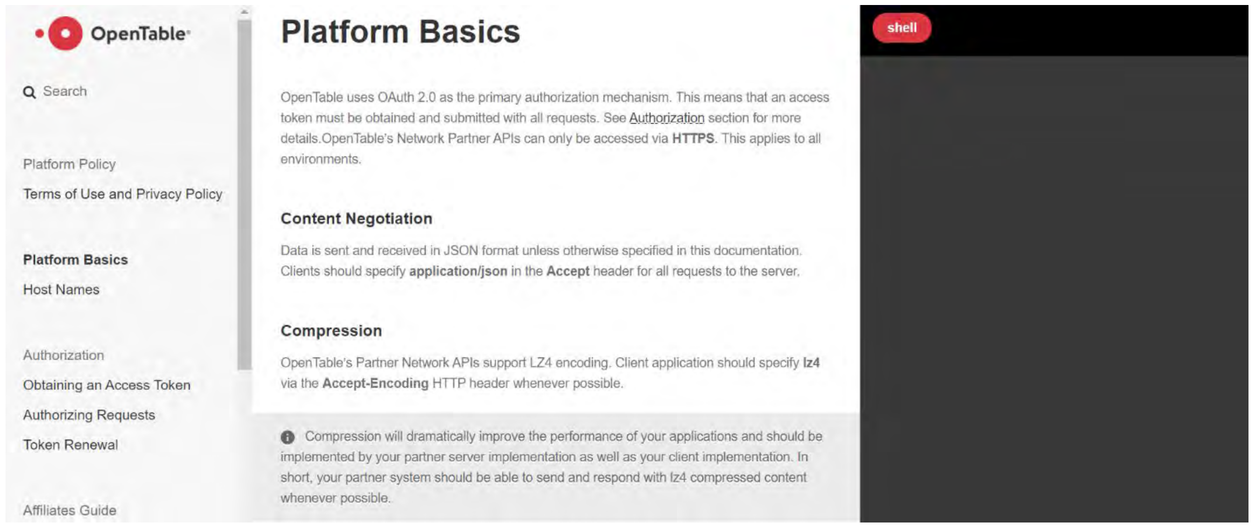
```
Name
data:image/svg+xml;...
data:image/svg+xml;...
search

Headers  Preview  Response  Initiator  Timing  Cookies
▾ General
  Request URL: https://www.opentable.com/restaurant/profile/3503/search
  Request Method: POST
  Status Code: ● 200
  Remote Address: 23.59.112.8:443
  Referrer Policy: strict-origin-when-cross-origin
▾ Response Headers
  content-encoding: gzip
  content-length: 494
  content-type: application/json;charset=utf-8
  date: Wed, 24 Feb 2021 05:33:00 GMT
  etag: W/"8df-OBoTj9SpPLfQdEJUNLv3JYgDI/c--gzip"
  ot-requestid: be4cfcf8-da47-4d54-b8b9-d02c8d008a3e
  set-cookie: OT-SessionId=8d0cd556-7889-4a98-b647-e084f0390aa6; Path=/; Domain=opentable.com; Expires=Wed, 24-Feb-2021 06:03:00 GMT; Max-Age=180
  0; Secure; SameSite=None
  set-cookie: OT-Session-Update-Date=1614144780; Path=/; Domain=opentable.com; Expires=Wed, 24-Feb-2021 06:03:00 GMT; Max-Age=1800; Secure; SameS
  ite=None
  set-cookie: ftc=x=2021-02-24T08%3A33%3A00&c=1&pt1=1&pt2=1&er=3503; Max-Age=10800; Domain=.opentable.com; Path=/; Expires=Wed, 24 Feb 2021 08:3
  3:00 GMT; Secure; SameSite=None
  strict-transport-security: max-age=31536000; includeSubDomains; preload
  vary: Accept-Encoding
  x-content-type-options: nosniff
  x-response-time: 319.357711ms
  x-xss-protection: 1; mode=block
```

201. On information and belief, the OpenTable Platform utilizes JSON data, which comprise symbolic names, to evoke web components to communicate inputs and outputs for the APIs and other web services. In the example below, output available time slots for the OpenTable's reservation web service are transmitted in JSON data.



```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
    noTimesMessage: null
   ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
    ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:00:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:00 PM"
      url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A00&sd=2021-03-02%2017%3A00&p=2&pt=100&pofids=&hash=2000417168st=Stand
    ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:15:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:15 PM"
      url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A15&sd=2021-03-02%2017%3A15&p=2&pt=100&pofids=&hash=1693584098st=Stan
    ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:30:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:30 PM"
      url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A30&sd=2021-03-02%2017%3A30&p=2&pt=100&pofids=&hash=1289295379&st=Stan
    ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
    ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```

202. On information and belief, the registry includes symbolic names in the form of JSON data, to evoke web components for the applications and to communicate inputs and outputs for the web services over the Internet. JSON data comprise key/value pairs that are essentially character strings (i.e., symbolic names) with no persistent address or pointer.



Platform Basics

OpenTable uses OAuth 2.0 as the primary authorization mechanism. This means that an access token must be obtained and submitted with all requests. See Authorization section for more details.OpenTable's Network Partner APIs can only be accessed via **HTTPS**. This applies to all environments.

Content Negotiation

Data is sent and received in JSON format unless otherwise specified in this documentation. Clients should specify **application/json** in the **Accept** header for all requests to the server.

Compression

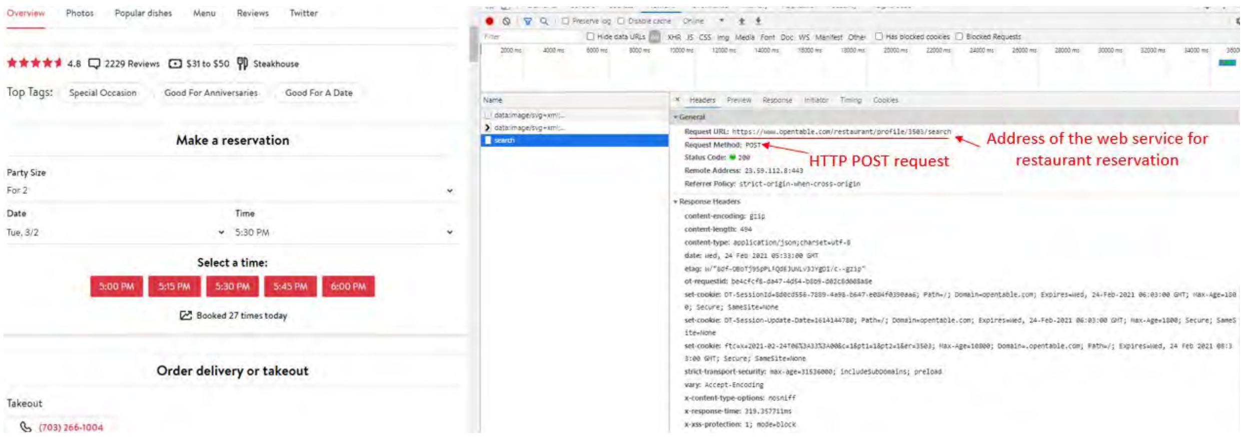
OpenTable's Partner Network APIs support LZ4 encoding. Client application should specify **lz4** via the **Accept-Encoding** HTTP header whenever possible.

Compression will dramatically improve the performance of your applications and should be implemented by your partner server implementation as well as your client implementation. In short, your partner system should be able to send and respond with lz4 compressed content whenever possible.

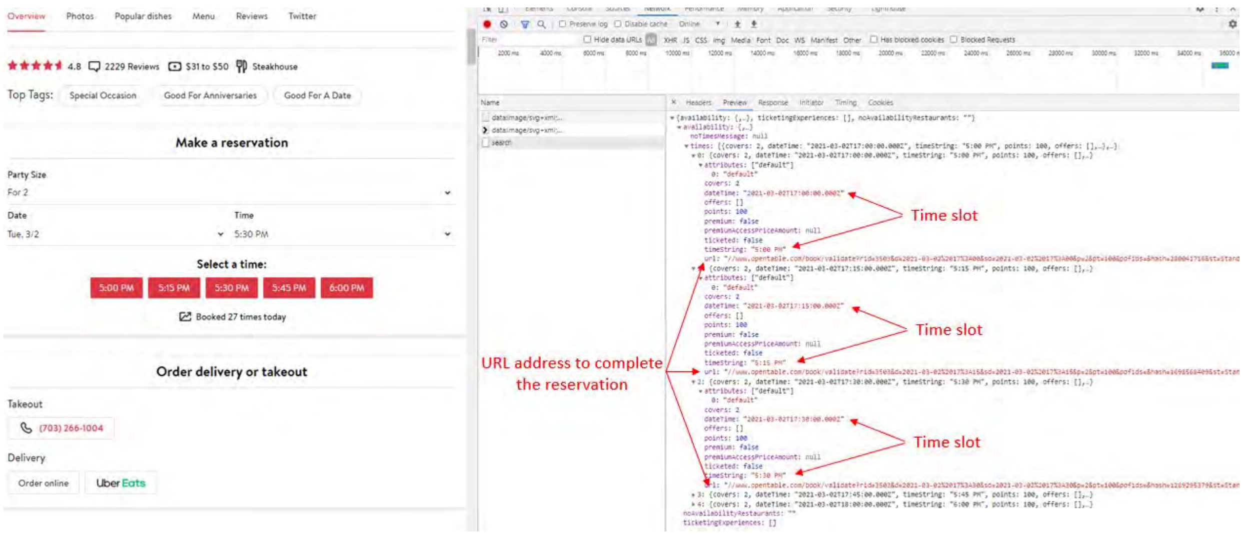
https://platform.opentable.com/documentation/#platform-basics

203. The registry also includes addresses for the web services where the input symbolic names and output symbolic names can be sent to and received from. The exemplary screenshot below shows a request URL including an address of the OpenTable's reservation web service. The OpenTable Platform transfers user input settings, e.g., party size, date and time, in JSON format

and sends them to the OpenTable's reservation web service via an HTTP POST request. The OpenTable's reservation web service responds with five available time slots for the input date and time in JSON format and the OpenTable Platform creates five web components (e.g., button UI objects) to display the five available time slots. Each of the five web components further includes a URL address to complete the reservation in the specified time slot.

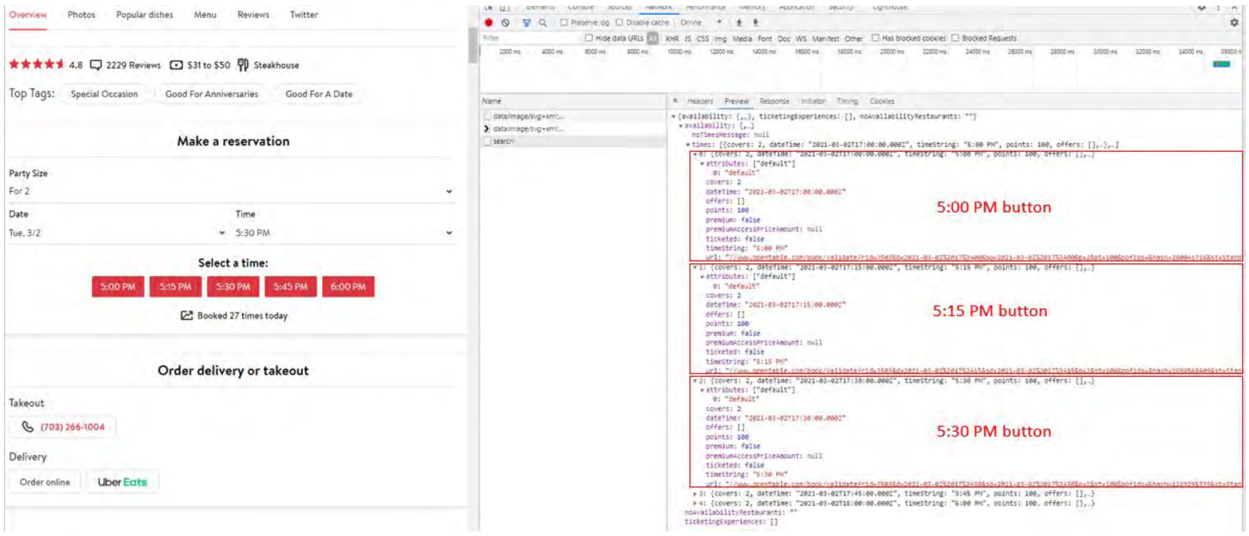


```
▾ General
   Request URL: https://www.opentable.com/restaurant/profile/3503/search
   Request Method: POST
   Status Code: 200
   Remote Address: 23.59.112.8:443
   Referrer Policy: strict-origin-when-cross-origin
▾ Response Headers
   content-encoding: gzip
   content-length: 494
   content-type: application/json;charset=utf-8
   date: Wed, 24 Feb 2021 05:33:00 GMT
   etag: W/"8df-OBoTj9SpPLFQdEJUNLV3JYgD1/c--gzip"
   ot-requestid: be4cfcf8-da47-4d54-b8b9-d02c8d008a8e
   set-cookie: OT-SessionId=8d0cd556-7889-4e98-b647-e084f0390ea6; Path=/; Domain=opentable.com; Expires=Wed, 24-Feb-2021 06:03:00 GMT; Max-Age=180
   0; Secure; SameSite=None
   set-cookie: OT-Session-Update-Date=1614144780; Path=/; Domain=opentable.com; Expires=Wed, 24-Feb-2021 06:03:00 GMT; Max-Age=1800; Secure; SameS
   ite=None
   set-cookie: ftc=x=2021-02-24T06%3A33%3A00&c=1&pt1=1&pt2=1&er=3503; Max-Age=10800; Domain=.opentable.com; Path=/; Expires=Wed, 24 Feb 2021 08:3
   3:00 GMT; Secure; SameSite=None
   strict-transport-security: max-age=31536000; includeSubDomains; preload
   vary: Accept-Encoding
   x-content-type-options: nosniff
   x-response-time: 319.357711ms
   x-xss-protection: 1; mode=block
```



```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
    noTimesMessage: null
   ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
     ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
       ▾attributes: ["default"]
          0: "default"
        covers: 2
        dateTime: "2021-03-02T17:00:00.000Z"
        offers: []
        points: 100
        premium: false
        premiumAccessPriceAmount: null
        ticketed: false
        timeString: "5:00 PM"
        url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A00&sd=2021-03-02%2017%3A00&p=2&pt=100&pofIds=&hash=2800417168st=Stand
     ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
       ▾attributes: ["default"]
          0: "default"
        covers: 2
        dateTime: "2021-03-02T17:15:00.000Z"
        offers: []
        points: 100
        premium: false
        premiumAccessPriceAmount: null
        ticketed: false
        timeString: "5:15 PM"
        url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A15&sd=2021-03-02%2017%3A15&p=2&pt=100&pofIds=&hash=1698568409&st=Stan
     ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
       ▾attributes: ["default"]
          0: "default"
        covers: 2
        dateTime: "2021-03-02T17:30:00.000Z"
        offers: []
        points: 100
        premium: false
        premiumAccessPriceAmount: null
        ticketed: false
        timeString: "5:30 PM"
        url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A30&sd=2021-03-02%2017%3A30&p=2&pt=100&pofIds=&hash=1269295379&st=Stan
     ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
     ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
    noAvailabilityRestaurants: ""
    ticketingExperiences: []
```

204. The authoring tool is configured to, among other things, define a UI object corresponding to a web component in the registry for an input or output of a web service, for presentation on the display. By way of example, the authoring tool defines UI objects for web components such as a button to represent the available time slot.



4.8 2229 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Good For A Date

Party Size: For 2

Date: Tue, 3/2 Time: 5:30 PM

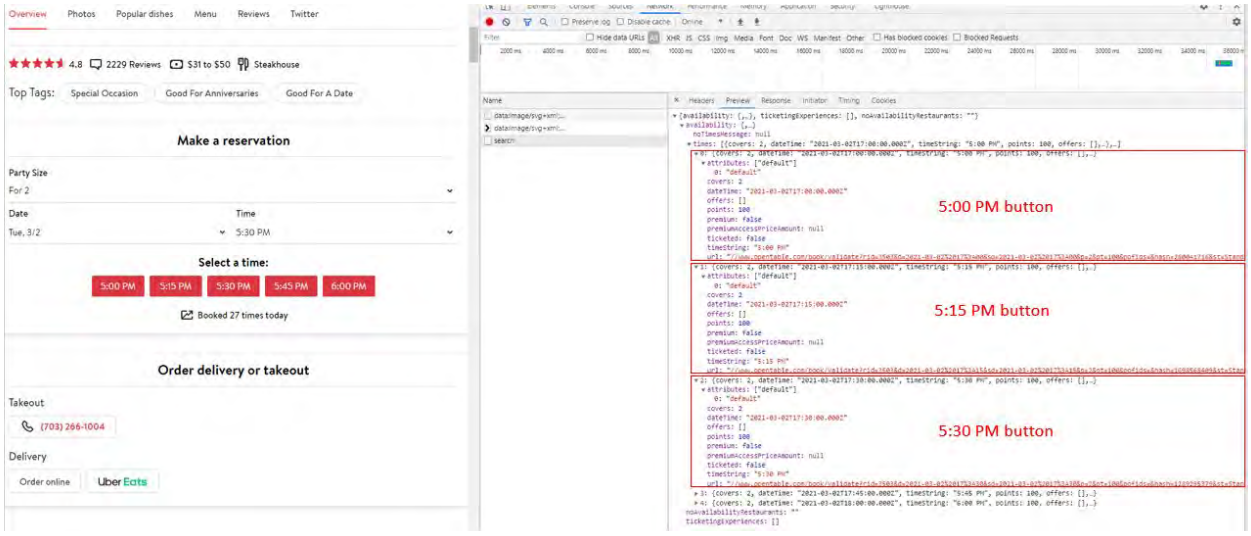
Select a time: 5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM

Booked 27 times today

Takeout: (703) 266-1004

```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
    noTimesMessage: null
   ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
    ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
      ▾attributes: ["default"]
         0: "default"
       covers: 2
       dateTime: "2021-03-02T17:00:00.000Z"
       offers: []
       points: 100
       premium: false
       premiumAccessPriceAmount: null
       ticketed: false
       timeString: "5:00 PM"
       url: "//www.opentable.com/book/validate?rid=35678&d=2021-03-02%2017%3A00&sd=2021-03-02%2017%3A00&p=2&pt=100&pofids=&hash=2800417168&st=Stand
    ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
      ▾attributes: ["default"]
         0: "default"
       covers: 2
       dateTime: "2021-03-02T17:15:00.000Z"
       offers: []
       points: 100
       premium: false
       premiumAccessPriceAmount: null
       ticketed: false
       timeString: "5:15 PM"
       url: "//www.opentable.com/book/validate?rid=35678&d=2021-03-02%2017%3A15&sd=2021-03-02%2017%3A15&p=2&pt=100&pofids=&hash=1088568409&st=Stan
    ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
      ▾attributes: ["default"]
         0: "default"
       covers: 2
       dateTime: "2021-03-02T17:30:00.000Z"
       offers: []
       points: 100
       premium: false
       premiumAccessPriceAmount: null
       ticketed: false
       timeString: "5:30 PM"
       url: "//www.opentable.com/book/validate?rid=35678&d=2021-03-02%2017%3A30&sd=2021-03-02%2017%3A30&p=2&pt=100&pofids=&hash=1289295379&st=Stan
    ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
    ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```

205. On information and belief, the authoring tool selects a symbolic name (e.g., JSON data) from the web component corresponding to the defined UI object. In the example below, symbolic name "0" corresponds to the "5:00 PM" button UI object; symbolic name "1" corresponds to the "5:15 PM" button UI object; symbolic name "2" corresponds to the "5:30 PM" button UI object.



4.8 2229 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Good For A Date

Party Size: For 2

Date: Tue, 3/2 Time: 5:30 PM

Select a time: 5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM

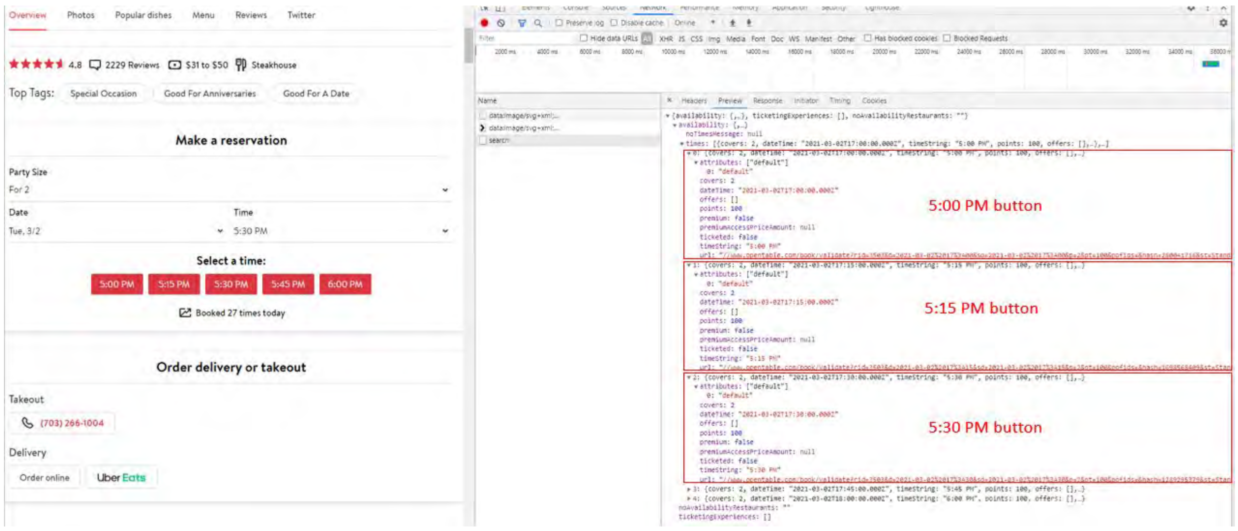
Booked 27 times today

Takeout: (703) 266-1004

```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
    noTimesMessage: null
   ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
    ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
      ▾attributes: ["default"]
         0: "default"
       covers: 2
       dateTime: "2021-03-02T17:00:00.000Z"
       offers: []
       points: 100
       premium: false
       premiumAccessPriceAmount: null
       ticketed: false
       timeString: "5:00 PM"
       url: "//www.opentable.com/book/validate?rid=35678&d=2021-03-02%2017%3A00&sd=2021-03-02%2017%3A00&p=2&pt=100&pofids=&hash=2800417168&st=Stand
    ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
      ▾attributes: ["default"]
         0: "default"
       covers: 2
       dateTime: "2021-03-02T17:15:00.000Z"
       offers: []
       points: 100
       premium: false
       premiumAccessPriceAmount: null
       ticketed: false
       timeString: "5:15 PM"
       url: "//www.opentable.com/book/validate?rid=35678&d=2021-03-02%2017%3A15&sd=2021-03-02%2017%3A15&p=2&pt=100&pofids=&hash=1088568409&st=Stan
    ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
      ▾attributes: ["default"]
         0: "default"
       covers: 2
       dateTime: "2021-03-02T17:30:00.000Z"
       offers: []
       points: 100
       premium: false
       premiumAccessPriceAmount: null
       ticketed: false
       timeString: "5:30 PM"
       url: "//www.opentable.com/book/validate?rid=35678&d=2021-03-02%2017%3A30&sd=2021-03-02%2017%3A30&p=2&pt=100&pofids=&hash=1289295379&st=Stan
    ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
    ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```

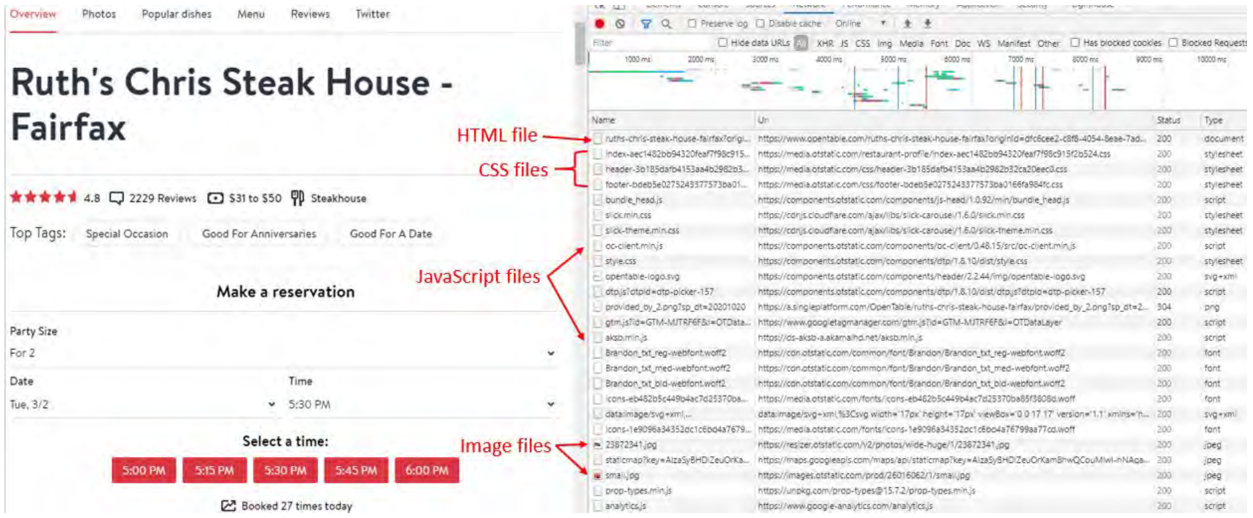
206. The authoring tool associates the selected symbolic name from the web component corresponding to the defined UI object with the defined UI object. In the example below, the authoring tool associates "5:00 PM" button UI object with the symbolic name "0". In particular,

the symbolic name "timeString" associates with the value "5:00 PM" that is displayed on the button UI object.



```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
    noTimesMessage: null
   ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
    ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:00:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:00 PM"
      url: "//www.opentable.com/book/validate?rid=[illegible]
    ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:15:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:15 PM"
      url: "//www.opentable.com/book/validate?rid=[illegible]
    ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:30:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:30 PM"
      url: "//www.opentable.com/book/validate?rid=[illegible]
    ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
    ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```

207. The authoring tool produces a device-independent code (i.e., an application) that includes the JSON data, as well as standard HTML, CSS and Javascript code. In the example below, the authoring tool generates a device-independent application for a restaurant profile, in which five button UI objects are used to represent five available time slots for restaurant reservation. The symbolic names "0" is associated with the "5:00 PM" button UI object; the symbolic name "1" is associated with the "5:15 PM" button UI object; the symbolic name "2" is associated with the "5:30 PM" button UI object; the symbolic name "3" is associated with the "5:45 PM" button UI object; and the symbolic name "4" is associated with the "6:00 PM" button UI object.



Overview Photos Popular dishes Menu Reviews Twitter

Ruth's Chris Steak House - Fairfax

4.8 2229 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Good For A Date

Make a reservation

Party Size For 2

Date Tue, 3/2 Time 5:30 PM

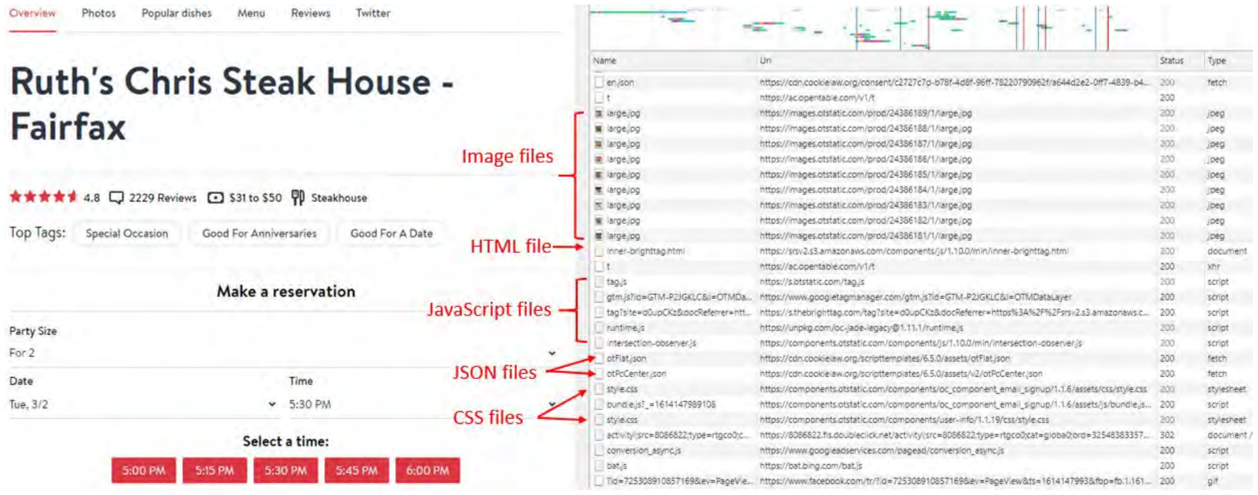
Select a time: 5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM

Booked 27 times today

Preserve log Disable cache Online

Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other Has blocked cookies Blocked Requests

Name	Url	Status	Type
ruths-chris-steak-house-fairfax?origi...	https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8eae-7ad...	200	document
index-aec1482bb94320feaf7f98c915...	https://media.otstatic.com/restaurant-profile/index-aec1482bb94320feaf7f98c915f2b524.css	200	stylesheet
header-3b185dafb4153aa4b2982b3...	https://media.otstatic.com/css/header-3b185dafb4153aa4b2982b32ca20eec0.css	200	stylesheet
footer-bdeb5e027524337757 3ba01...	https://media.otstatic.com/css/footer-bdeb5e0275243377573ba0166fa984fc.css	200	stylesheet
bundle_head.js	https://components.otstatic.com/components/js-head/1.0.92/min/bundle_head.js	200	script
slick.min.css	https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick.min.css	200	stylesheet
slick-theme.min.css	https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick-theme.min.css	200	stylesheet
oc-client.min.js	https://components.otstatic.com/components/oc-client/0.48.15/src/oc-client.min.js	200	script
style.css	https://components.otstatic.com/components/dtp/1.8.10/dist/style.css	200	stylesheet
opentable-logo.svg	https://components.otstatic.com/components/header/2.2.44/img/opentable-logo.svg	200	svg+xml
dtp.js?dtpId=dtp-picker-157	https://components.otstatic.com/components/dtp/1.8.10/dist/dtp.js?dtpId=dtp-picker-157	200	script
provided_by_2.png?sp_dt=20201020	https://a.singleplatform.com/OpenTable/ruths-chris-steak-house-fairfax/provided_by_2.png?sp_dt=2...	304	png
gtm.js?id=GTM-MJTRF6F&l=OTData...	https://www.googletagmanager.com/gtm.js?id=GTM-MJTRF6F&l=OTDataLayer	200	script
aksb.min.js	https://ds-aksb-a.akamaihd.net/aksb.min.js	200	script
Brandon_txt_reg-webfont.woff2	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_reg-webfont.woff2	200	font
Brandon_txt_med-webfont.woff2	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_med-webfont.woff2	200	font
Brandon_txt_bld-webfont.woff2	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_bld-webfont.woff2	200	font
icons-eb482b5c449b4ac7d25370ba...	https://media.otstatic.com/fonts/icons-eb482b5c449b4ac7d25370ba85f3808d.woff	200	font
data:image/svg+xml,...	data:image/svg+xml,%3Csvg width='17px' height='17px' viewBox='0 0 17 17' version='1.1' xmlns='h...	200	svg+xml
icons-1e9096a34352dc1c6bd4a7679...	https://media.otstatic.com/fonts/icons-1e9096a34352dc1c6bd4a76799aa77cd.woff	200	font
23872341.jpg	https://resizer.otstatic.com/v2/photos/wide-huge/1/23872341.jpg	200	jpeg
staticmap?key=AIzaSyBHDiZeuOrKa...	https://maps.googleapis.com/maps/api/staticmap?key=AIzaSyBHDiZeuOrKam8hwQCouMlwI-nNAga...	200	jpeg
small.jpg	https://images.otstatic.com/prod/26016062/1/small.jpg	200	jpeg
prop-types.min.js	https://unpkg.com/prop-types@15.7.2/prop-types.min.js	200	script
analytics.js	https://www.google-analytics.com/analytics.js	200	script



Overview Photos Popular dishes Menu Reviews Twitter

Ruth's Chris Steak House - Fairfax

4.8 2229 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Good For A Date

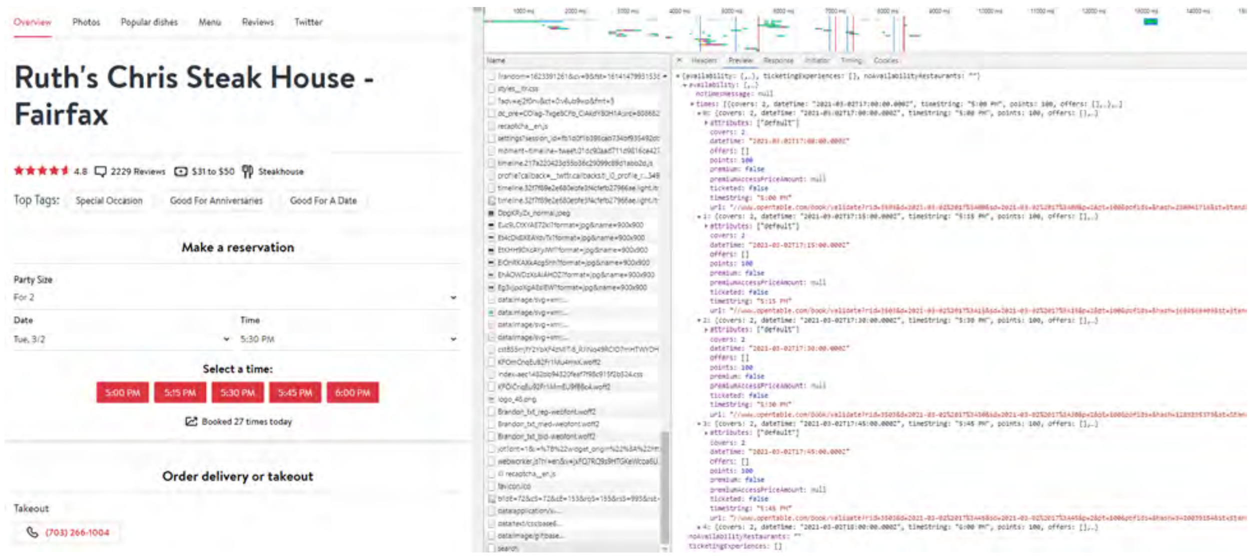
Make a reservation

Party Size For 2

Date Tue, 3/2 Time 5:30 PM

Select a time: 5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM

Name	Url	Status	Type
en.json	https://cdn.cookielaw.org/consent/c2727c7d-b78f-4d8f-96ff-78220790962f/a644d2e2-0ff7-4839-b4...	200	fetch
t	https://ac.opentable.com/v1/t	200	
large.jpg	https://images.otstatic.com/prod/24386189/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386188/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386187/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386186/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386185/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386184/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386183/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386182/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386181/1/large.jpg	200	jpeg
inner-brighttag.html	https://srsv2.s3.amazonaws.com/components/js/1.10.0/min/inner-brighttag.html	200	document
t	https://ac.opentable.com/v1/t	200	xhr
tag.js	https://s.otstatic.com/tag.js	200	script
gtm.js?id=GTM-P2JGKLC&l=OTMDa...	https://www.googletagmanager.com/gtm.js?id=GTM-P2JGKLC&l=OTMDataLayer	200	script
tag?site=d0upCKz&docReferrer=htt...	https://s.thebrighttag.com/tag?site=d0upCKz&docReferrer=https%3A%2F%2Fsrsv2.s3.amazonaws.c...	200	script
runtime.js	https://unpkg.com/oc-jade-legacy@1.11.1/runtime.js	200	script
intersection-observer.js	https://components.otstatic.com/components/js/1.10.0/min/intersection-observer.js	200	script
otFlat.json	https://cdn.cookielaw.org/scripttemplates/6.5.0/assets/otFlat.json	200	fetch
otPcCenter.json	https://cdn.cookielaw.org/scripttemplates/6.5.0/assets/v2/otPcCenter.json	200	fetch
style.css	https://components.otstatic.com/components/oc_component_email_signup/1.1.6/assets/css/style.css	200	stylesheet
bundle.js?_=1614147989106	https://components.otstatic.com/components/oc_component_email_signup/1.1.6/assets/js/bundle.js...	200	script
style.css	https://components.otstatic.com/components/user-info/1.1.19/css/style.css	200	stylesheet
activity;src=8086822;type=rtgco0;c...	https://8086822.fls.doubleclick.net/activity;src=8086822;type=rtgco0;cat=globa0;ord=32548383357...	302	document /
conversion_async.js	https://www.googleadservices.com/pagead/conversion_async.js	200	script
bat.js	https://bat.bing.com/bat.js	200	script
?id=725308910857169&ev=PageVie...	https://www.facebook.com/tr/?id=725308910857169&ev=PageView&ts=1614147993&fbp=fb.1.161...	200	gif



Overview Photos Popular dishes Menu Reviews Twitter

Ruth's Chris Steak House - Fairfax

4.8 2229 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Good For A Date

Make a reservation

Party Size For 2

Date Tue, 3/2 Time 5:30 PM

Select a time: 5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM

Booked 27 times today

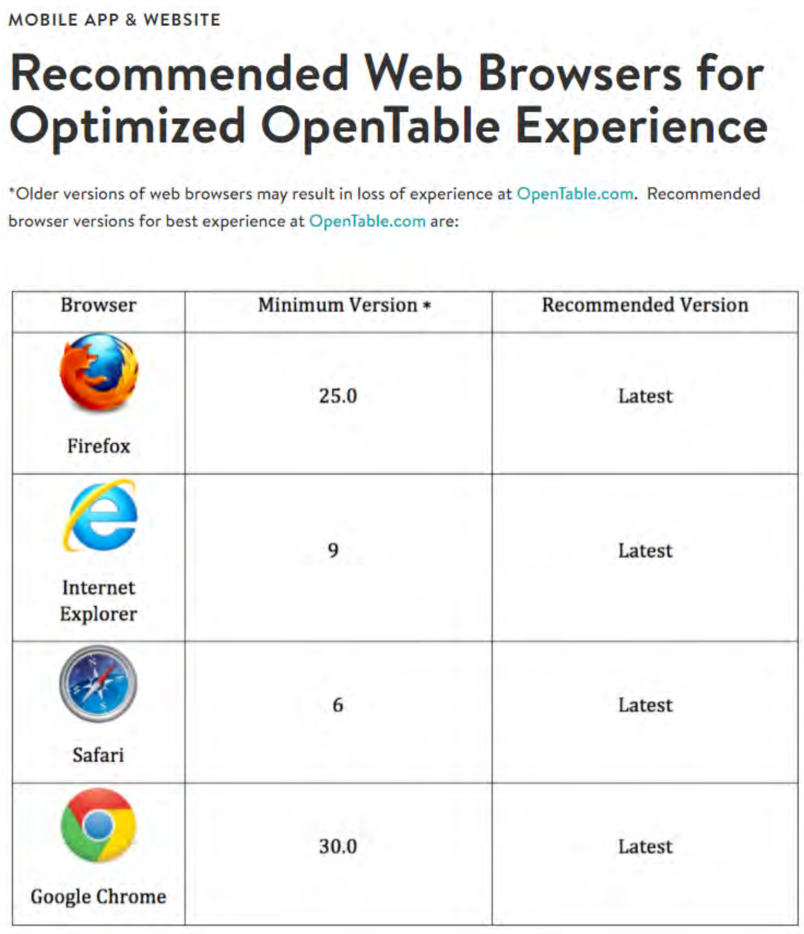
Order delivery or takeout

Takeout (703) 266-1004

Headers Preview Response Initiator Timing Cookies

```
▾ {availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾ availability: {,…}
    notimesmessage: null
    ▾ times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
      ▾ 0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
        ▸ attributes: ["default"]
          covers: 2
          dateTime: "2021-03-02T17:00:00.000Z"
          offers: []
          points: 100
          premium: false
          premiumAccessPriceAmount: null
          ticketed: false
          timeString: "5:00 PM"
          url: "//www.opentable.com/book/validate?rid=[illegible]"
      ▾ 1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
        ▸ attributes: ["default"]
          covers: 2
          dateTime: "2021-03-02T17:15:00.000Z"
          offers: []
          points: 100
          premium: false
          premiumAccessPriceAmount: null
          ticketed: false
          timeString: "5:15 PM"
          url: "//www.opentable.com/book/validate?rid=[illegible]"
      ▾ 2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
        ▸ attributes: ["default"]
          covers: 2
          dateTime: "2021-03-02T17:30:00.000Z"
          offers: []
          points: 100
          premium: false
          premiumAccessPriceAmount: null
          ticketed: false
          timeString: "5:30 PM"
          url: "//www.opentable.com/book/validate?rid=[illegible]"
      ▾ 3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
        ▸ attributes: ["default"]
          covers: 2
          dateTime: "2021-03-02T17:45:00.000Z"
          offers: []
          points: 100
          premium: false
          premiumAccessPriceAmount: null
          ticketed: false
          timeString: "5:45 PM"
          url: "//www.opentable.com/book/validate?rid=[illegible]"
      ▸ 4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
    noAvailabilityRestaurants: ""
    ticketingExperiences: []
```

208. The authoring tool produces a player, such as a device-dependent code for a browser engine for a browser, or for an operating system or application of a device. See, e.g., Shopify Inc. v. Express Mobile, Inc., Case No. 1:19-cv-00439-RGA, D. Del., D.I. 137. For example, the OpenTable Platform can display a restaurant profile on a Google Chrome, Mozilla Firefox, Apple Safari, or Microsoft Internet Explorer using the player.



MOBILE APP & WEBSITE

Recommended Web Browsers for Optimized OpenTable Experience

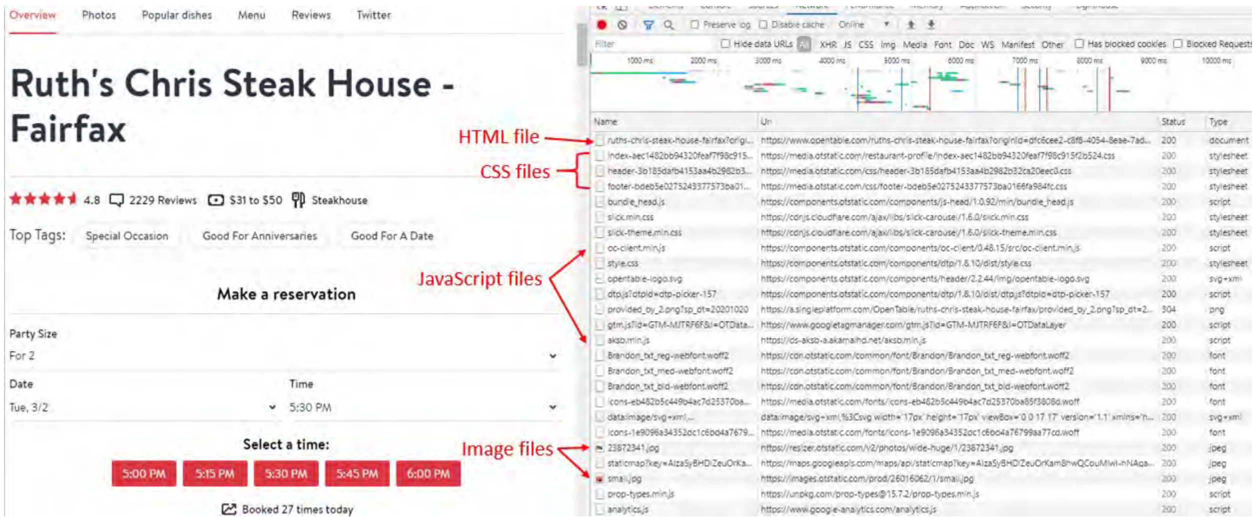
*Older versions of web browsers may result in loss of experience at OpenTable.com. Recommended browser versions for best experience at OpenTable.com are:

Browser	Minimum Version *	Recommended Version
Firefox	25.0	Latest
Internet Explorer	9	Latest
Safari	6	Latest
Google Chrome	30.0	Latest

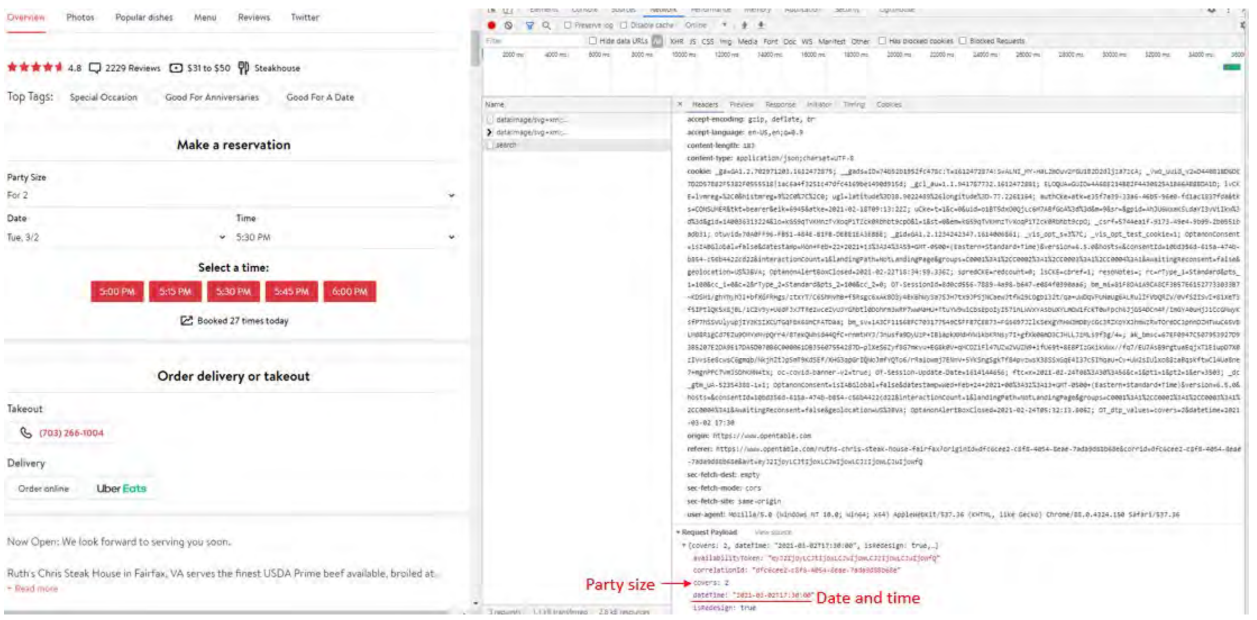
https://help.opentable.com/s/article/Recommended-Web-Browsers-for-Optimized-OpenTable-Experience-1505260708133?language=en_US

209. The OpenTable Platform can execute the application and the player on a device to produce the contents for the restaurant profile, where a user can input a value associated with an input symbolic name to an input of a defined UI object. In the example below, the OpenTable Platform converts HTML, CSS, Javascript, image, and other files from an application into an active property listing using a player. The active restaurant profile accepts a search input value

(e.g., party size, date, and time). For example, a client can make a reservation for party size of "for 2", date of "Tue, 3/2", and time of "5:30 PM". An input symbolic name "covers" is associated with party size and another input symbolic name "dateTime" is associated with the date and time. These symbolic names along with the input values are transmitted to the OpenTable API through an HTTP request protocol such as a POST method call.



Name	Url	Status	Type
ruths-chris-steak-house-fairfax?origi...	https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8eae-7ad...	200	document
index-aec1482bb94320feaf7f98c915...	https://media.otstatic.com/restaurant-profile/index-aec1482bb94320feaf7f98c915f2b524.css	200	stylesheet
header-3b185dafb4153aa4b2982b3...	https://media.otstatic.com/css/header-3b185dafb4153aa4b2982b32ca20eec0.css	200	stylesheet
footer-bdeb5e0275243377573ba01...	https://media.otstatic.com/css/footer-bdeb5e0275243377573ba0166fa984fc.css	200	stylesheet
bundle_head.js	https://components.otstatic.com/components/js-head/1.0.92/min/bundle_head.js	200	script
slick.min.css	https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick.min.css	200	stylesheet
slick-theme.min.css	https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick-theme.min.css	200	stylesheet
oc-client.min.js	https://components.otstatic.com/components/oc-client/0.48.15/src/oc-client.min.js	200	script
style.css	https://components.otstatic.com/components/dtp/1.8.10/dist/style.css	200	stylesheet
opentable-logo.svg	https://components.otstatic.com/components/header/2.2.44/img/opentable-logo.svg	200	svg+xml
dtp.js?dtpId=dtp-picker-157	https://components.otstatic.com/components/dtp/1.8.10/dist/dtp.js?dtpId=dtp-picker-157	200	script
provided_by_2.png?sp_dt=20201020	https://a.singleplatform.com/OpenTable/ruths-chris-steak-house-fairfax/provided_by_2.png?sp_dt=2...	304	png
gtm.js?id=GTM-MJTRF6F&l=OTData...	https://www.googletagmanager.com/gtm.js?id=GTM-MJTRF6F&l=OTDataLayer	200	script
aksb.min.js	https://ds-aksb-a.akamaihd.net/aksb.min.js	200	script
Brandon_txt_reg-webfont.woff2	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_reg-webfont.woff2	200	font
Brandon_txt_med-webfont.woff2	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_med-webfont.woff2	200	font
Brandon_txt_bld-webfont.woff2	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_bld-webfont.woff2	200	font
icons-eb482b5c449b4ac7d25370ba...	https://media.otstatic.com/fonts/icons-eb482b5c449b4ac7d25370ba85f3808d.woff	200	font
data:image/svg+xml,...	data:image/svg+xml,%3Csvg width='17px' height='17px' viewBox='0 0 17 17' version='1.1' xmlns='h...	200	svg+xml
icons-1e9096a34352dc1c6bd4a7679...	https://media.otstatic.com/fonts/icons-1e9096a34352dc1c6bd4a76799aa77cd.woff	200	font
23872341.jpg	https://resizer.otstatic.com/v2/photos/wide-huge/1/23872341.jpg	200	jpeg
staticmap?key=AIzaSyBHDIZeuOrKa...	https://maps.googleapis.com/maps/api/staticmap?key=AIzaSyBHDIZeuOrKamBhwQCouMiwI-nNAqa...	200	jpeg
small.jpg	https://images.otstatic.com/prod/26016062/1/small.jpg	200	jpeg
prop-types.min.js	https://unpkg.com/prop-types@15.7.2/prop-types.min.js	200	script
analytics.js	https://www.google-analytics.com/analytics.js	200	script



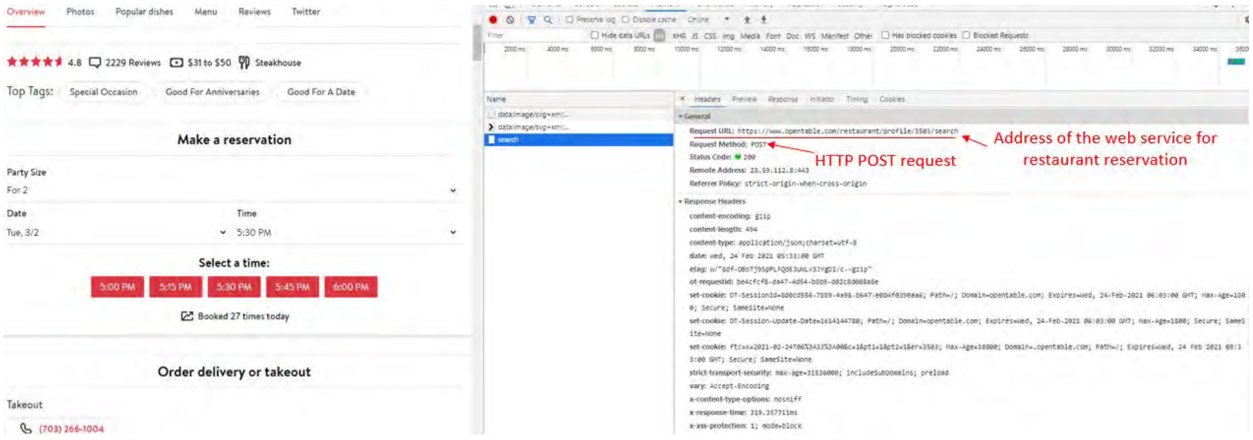
Now Open: We look forward to serving you soon.

Ruth's Chris Steak House in Fairfax, VA serves the finest USDA Prime beef available, broiled at...
+ Read more

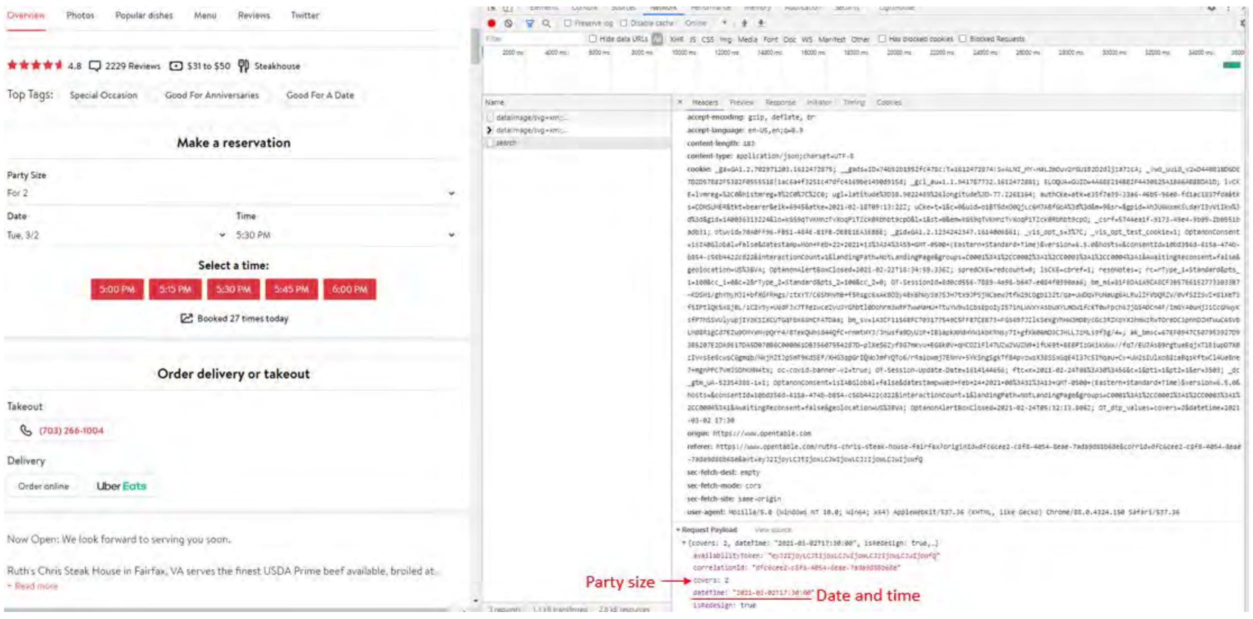
```
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
content-length: 183
content-type: application/json;charset=UTF-8
cookie: _ga=GA1.2.702971203.1612472875; __gads=ID=74052b1952fc478c:T=1612472874:S=ALNI_MY-H8L2mOuv2rGU1B2D2dlj1872cA; [illegible] OT_dtp_values=covers=2&datetime=2021-03-02 17:30
origin: https://www.opentable.com
referer: https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8eae-7ada9d8b68e&corrid=dfc6cee2-c8f8-4054-8eae-7ada9d8b68e&avt=eyJ2IjoyLCJtIjoxLCJwIjowLCJzIjowLCJuIjowfQ
sec-fetch-dest: empty
sec-fetch-mode: cors
sec-fetch-site: same-origin
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36
▼ Request Payload    View source
▼ {covers: 2, dateTime: "2021-03-02T17:30:00", isRedesign: true,…}
    availabilityToken: "eyJ2IjoyLCJtIjoxLCJwIjowLCJzIjowLCJuIjowfQ"
    correlationId: "dfc6cee2-c8f8-4054-8eae-7ada9d8b68e"
    covers: 2
    dateTime: "2021-03-02T17:30:00"
    isRedesign: true
```

210. When the OpenTable Platform executes the application and player on the device, the OpenTable Platform can cause the device to provide an input value and corresponding input

symbolic name to a web service. By way of example, the OpenTable Platform causes the device to transmit the input value and a corresponding input symbolic name (e.g., JSON data for the input) to the OpenTable's reservation web service through an HTTP request protocol, such as a POST method call. In the example below, when a client wants to make a reservation for party size of " for 2", date of "Tue, 3/2", and time of "5:30 PM", the OpenTable Platform causes the device to send a JSON data including the symbolic name "covers" associating with value "2" and the symbolic name "covers" associating with the value "2021-03-02T17:30:00" to the OpenTable's reservation web service with an HTTP POST request.



```
▾ General
   Request URL: https://www.opentable.com/restaurant/profile/3503/search
   Request Method: POST
   Status Code: ● 200
   Remote Address: 23.59.112.8:443
   Referrer Policy: strict-origin-when-cross-origin
▾ Response Headers
   content-encoding: gzip
   content-length: 494
   content-type: application/json;charset=utf-8
   date: Wed, 24 Feb 2021 05:33:00 GMT
   etag: W/"1df-OBoTj95pPLfQdEJUNLv3JYgDI/c--gzip"
   ot-requestid: be4cfcf8-da47-4d54-b8b9-d02c8d008a6e
   set-cookie: OT-SessionId=8d0cd556-7889-4a98-b647-e084f0390ea6; Path=/; Domain=opentable.com; Expires=Wed, 24-Feb-2021 06:03:00 GMT; Max-Age=180
   0; Secure; SameSite=None
   set-cookie: OT-Session-Update-Date=1614144780; Path=/; Domain=opentable.com; Expires=Wed, 24-Feb-2021 06:03:00 GMT; Max-Age=1800; Secure; SameS
   ite=None
   set-cookie: ftc=x=2021-02-24T06%3A33%3A00&c=1&pt1=1&pt2=1&er=3503; Max-Age=10800; Domain=.opentable.com; Path=/; Expires=Wed, 24 Feb 2021 08:3
   3:00 GMT; Secure; SameSite=None
   strict-transport-security: max-age=31536000; includeSubDomains; preload
   vary: Accept-Encoding
   x-content-type-options: nosniff
   x-response-time: 319.357711ms
   x-xss-protection: 1; mode=block
```

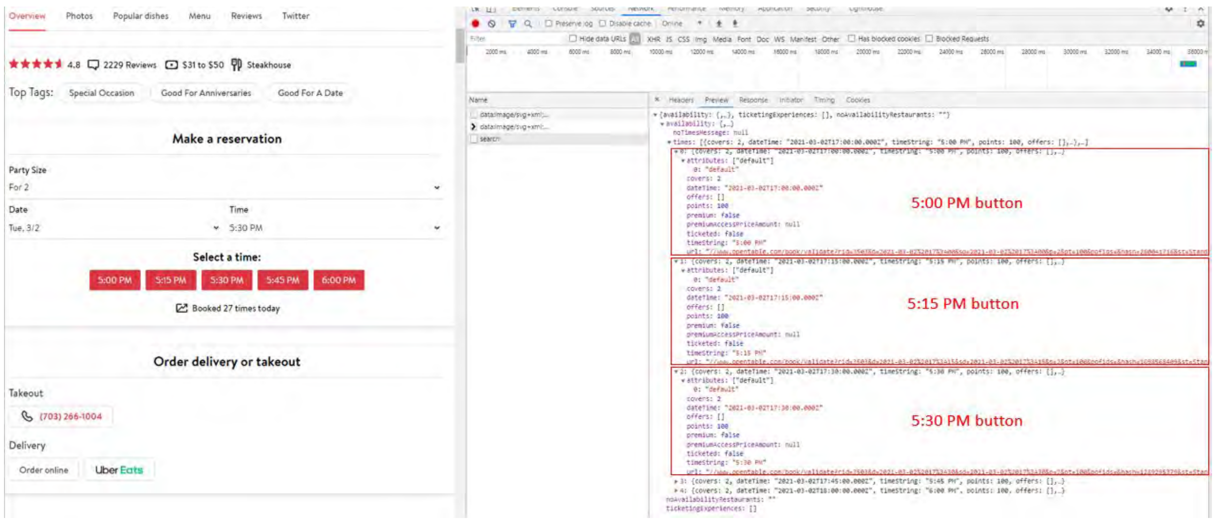


Now Open: We look forward to serving you soon.

Ruth's Chris Steak House in Fairfax, VA serves the finest USDA Prime beef available, broiled at...
+ Read more

```
   accept-encoding: gzip, deflate, br
   accept-language: en-US,en;q=0.9
   content-length: 183
   content-type: application/json;charset=UTF-8
   cookie: [illegible]
   -03-02 17:30
   origin: https://www.opentable.com
   referer: https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8eae-7ada9d8b6de&corrId=dfc6cee2-c8f8-4054-8eae
   -7ada9d8b6e&avt=eyJ2IjoyLCJtIjoxLCJwIjowLCJjIjowLCJuIjowfQ
   sec-fetch-dest: empty
   sec-fetch-mode: cors
   sec-fetch-site: same-origin
   user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36
▾ Request Payload    View source
   ▾ {covers: 2, dateTime: "2021-03-02T17:30:00", isRedesign: true,…}
      availabilityToken: "eyJ2IjoyLCJtIjoxLCJwIjowLCJjIjowLCJuIjowfQ"
      correlationId: "dfc6cee2-c8f8-4054-8eae-7ada9d8b6e"
      covers: 2
      dateTime: "2021-03-02T17:30:00"
      isRedesign: true
```

211. The OpenTable Platform provides the input value and a corresponding input symbolic name to the web service, and in turn causes the web service to generate an output value and a corresponding symbolic name for the output. In the example described below, when a search for "2021-03-02T17:30:00" is input, the OpenTable Platform causes the OpenTable's reservation web service to generate five corresponding output available time slots around the date and time in the form of JSON data.


Overview Photos Popular dishes Menu Reviews Twitter

4.8 2229 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Good For A Date

Make a reservation

Party Size: For 2

Date: Tue, 3/2 Time: 5:30 PM

Select a time: 5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM

Booked 27 times today

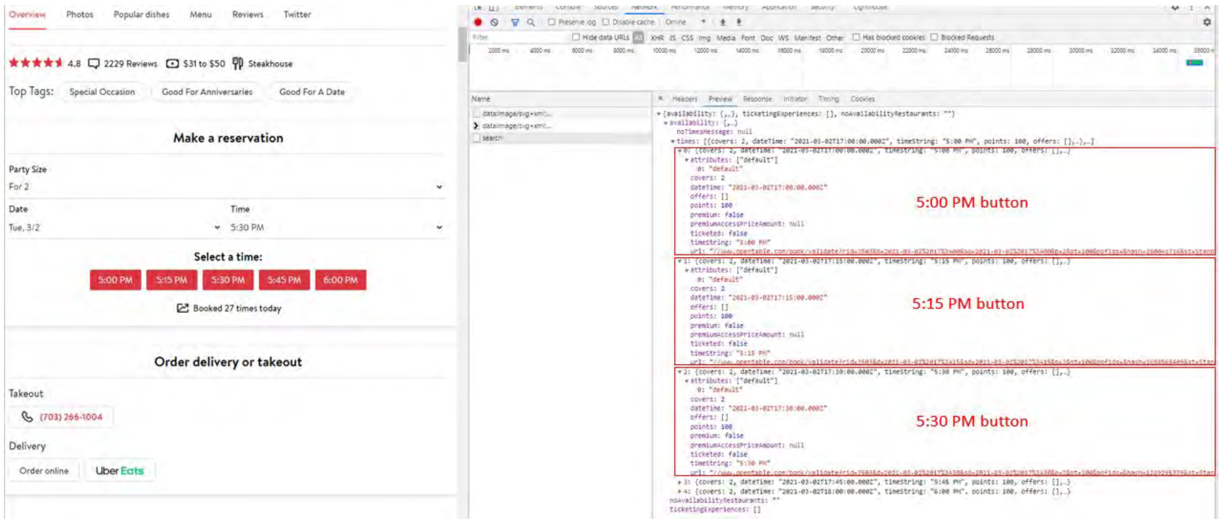
Order delivery or takeout

Takeout: (703) 266-1004

Delivery: Order online Uber Eats

```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
    noTimesMessage: null
   ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
    ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
      ▾attributes: ["default"]
         0: "default"
       covers: 2
       dateTime: "2021-03-02T17:00:00.000Z"
       offers: []
       points: 100
       premium: false
       premiumAccessPriceAmount: null
       ticketed: false
       timeString: "5:00 PM"
       url: "//www.opentable.com/book/validate?rid=35038&d=2021-03-02%2017%3A00&sd=2021-03-02%2017%3A00&p=2&pt=100&pofids=&hash=2800417168&st=Stand
    ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
      ▾attributes: ["default"]
         0: "default"
       covers: 2
       dateTime: "2021-03-02T17:15:00.000Z"
       offers: []
       points: 100
       premium: false
       premiumAccessPriceAmount: null
       ticketed: false
       timeString: "5:15 PM"
       url: "//www.opentable.com/book/validate?rid=35038&d=2021-03-02%2017%3A15&sd=2021-03-02%2017%3A15&p=2&pt=100&pofids=&hash=1098568409&st=Stan
    ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
      ▾attributes: ["default"]
         0: "default"
       covers: 2
       dateTime: "2021-03-02T17:30:00.000Z"
       offers: []
       points: 100
       premium: false
       premiumAccessPriceAmount: null
       ticketed: false
       timeString: "5:30 PM"
       url: "//www.opentable.com/book/validate?rid=35038&d=2021-03-02%2017%3A30&sd=2021-03-02%2017%3A30&p=2&pt=100&pofids=&hash=1289295379&st=Stan
    ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
    ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```

212. The player receives the output symbolic name and output value and provides instructions to the display of the device to present the output value in the API. In the example below, the player provides instructions to the browser of the device to display five output button UI objects.


Overview Photos Popular dishes Menu Reviews Twitter

4.8 2229 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Good For A Date

Make a reservation

Party Size: For 2

Date: Tue, 3/2 Time: 5:30 PM

Select a time: 5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM

Booked 27 times today

Order delivery or takeout

Takeout: (703) 266-1004

Delivery: Order online Uber Eats

213. The presence of the above referenced features is demonstrated, by way of example, by testing the Accused Instrumentalities for investigative purposes on https://www.booking.com/, https://join.booking.com/, https://partner.booking.com/en-us, and/or https://ycs.agoda.com/en-us/kipp/public/home, and by reference to publicly available information, including the following:

- https://www.booking.com/,
- https://join.booking.com/,
- https://partner.booking.com/en-us,
- https://partner.booking.com/en-us/help/working-booking/how-do-i-join-bookingcom,
- https://partner.booking.com/en-us/help/working-booking/how-can-i-set-my-property-easily,
- https://ycs.agoda.com/en-us/kipp/public/home,
- https://www.agoda.com/info/ycs-online-registration.html?cid=1844104,
- https://www.agoda.com/info/privacy.html?cid=1844104,
- https://partners.agoda.com/en-us/faq.html,
- https://agodapropertyhelp.zendesk.com/hc/en-us,
- https://partners.agoda.com/,
- https://agodapropertyhelp.zendesk.com/hc/en-us/articles/115009545508-How-do-I-list-my-property-on-Agoda-com-,
- https://www.opentable.com/,
- https://support.opentable.com/s/?language=en_US,
- https://platform.opentable.com/documentation/, and
- https://restaurant.opentable.com/.

214. On information and belief, Defendants have had knowledge of the ‘755 patent and their infringement thereof at least as early as October 3, 2019, when plaintiff provided notice of the ‘287 and ‘044 patents and Defendants’ infringement thereof, both of which identify on their face the ‘755 patent, and its issuance from a parent patent application of the ‘287 and ‘044 patents,

and no later than November 30, 2020, when Plaintiff provided notice of the ‘755 patent and Defendants’ infringement of the ‘755 patent. Furthermore, Defendants have been aware of the ‘755 patent and their infringement thereof since at least the filing of the original Complaint, D.I. 1, on December 1, 2020.

215. On information and belief, Defendants have contributed and are contributing to the infringement of the ‘755 patent because Defendants know that the infringing aspects of their infringing products and services, including but not limited to the Accused Instrumentalities, are made for use in an infringement, and are not staple articles of commerce suitable for substantial non-infringing uses.

216. On information and belief, Defendants have induced and are inducing the infringement of the ‘755 patent, with knowledge of the ‘755 patent and that their acts, including without limitation using, offering to sell, selling within, and importing into the United States, the Accused Instrumentalities, would aid and abet and induce infringement by customers, clients, partners, developers, and end users of the foregoing.

217. In particular, Defendants’ actions that aid and abet others such as customers, clients, partners, developers, and end users to infringe include advertising and distributing the Accused Instrumentalities, providing instructional materials, training, and other services regarding the Accused Instrumentalities, and providing free listings for the Accused Instrumentalities. Defendants actively encouraged the adoption of the Accused Instrumentalities and provided support sites for the vast network of developers working with the Accused Instrumentalities, emphasizing the simple and user-friendly nature of the Accused Instrumentalities, for example, explaining that “Registration can take as little as 15 minutes to complete—get started today” and that Defendants provides “24/7 support by phone or email” (*see, e.g.*, https://join.booking.com/), that “Through one platform our hotel partners are distributed across both priceline.com and agoda.com maximizing performance across our Retail, Private, Opaque and Vacation Packages programs. Our partners can easily set rates, promotions, and change hotel details with a few simple clicks of a mouse or on the go with a mobile device” (*see, e.g.*, https://ycs.agoda.com/en-

us/kipp/public/home), and that "From online ordering and takeout to powerful marketing and experiences, make more money when you access our network of millions" and that restaurants can "[g]et discovered and capture the business of the millions of people, around the world and in your neighborhood, searching on OpenTable" (*see, e.g.*, https://restaurant.opentable.com/?utm_source=dinersite&utm_medium=referral&utm_campaign=topnav&Lead.LeadSource=DinerSite&Lead.Marketing_ID__c=topnav). On information and belief, Defendants have engaged in such actions with specific intent to cause infringement or with willful blindness to the resulting infringement because Defendants have had actual knowledge of the '397 patent and knowledge that their acts were inducing infringement of the '397 patent since at least the date Defendants received notice that their activities infringed the '397 patent.

218. Defendants' acts of infringement have caused damage to Plaintiff, and Plaintiff is entitled to recover damages from Defendants in an amount subject to proof at trial.

219. Defendants' infringement of Plaintiff's rights under the '755 patent will continue to damage Plaintiff's business, causing irreparable harm, for which there is no adequate remedy at law, unless enjoined by this Court.

220. On information and belief, Defendants have acted with disregard of Plaintiff's patent rights, without any reasonable basis for doing so, and have willfully infringed and do willfully infringe the '755 patent.

221. The foregoing is illustrative of Defendants' infringement of the '755 patent. Plaintiff reserves the right to identify additional claims and Accused Instrumentalities in accordance with the Court's local rules and applicable scheduling orders.

COUNT IV – INFRINGEMENT OF U.S. PATENT NO. 9,471,287

222. Plaintiff incorporates by reference paragraphs 1 to 55 above as if fully set forth herein.

223. On information and belief, Defendants have infringed and are infringing the '287 patent under 35 U.S.C. § 271, either literally and/or under the doctrine of equivalents, directly and/or indirectly.

224. On information and belief, Booking-BV has infringed and continues to infringe the ‘287 patent by performing, without authority, one or more of the following acts: making, using, offering to sell, selling within, and importing into, the United States products and services that practice the claimed inventions of the ‘287 patent, including but not limited to the WebDirect Platform and the Booking Platform.[7]

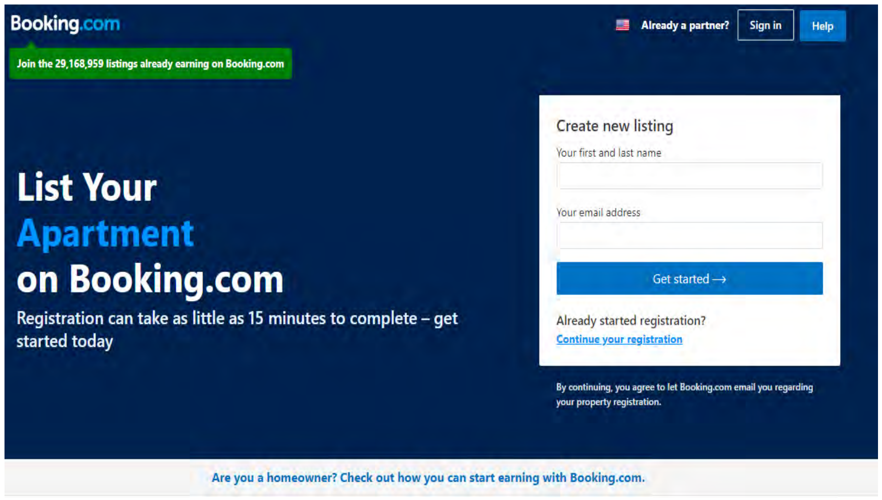
225. On information and belief, Priceline and Agoda (the “YCS infringers”) have infringed and continue to infringe the ‘287 patent by performing, without authority, one or more of the following acts: making, using, offering to sell, selling within, and importing into, the United States products and services that practice the claimed inventions of the ‘287 patent, including but not limited to the YCS platform.

226. On information and belief, OpenTable has infringed and continues to infringe the ‘287 patent by performing, without authority, one or more of the following acts during relevant time periods: making, using, offering to sell, selling within, and importing into the United States products and services that practice the claimed inventions of the ‘287 patent, including but not limited to the OpenTable Platform.

<u>The Booking Platform:</u>

227. Booking-BV infringes at least claim 15 of the ‘287 patent through a combination of features in the Booking Platform that collectively practice each claimed limitation of claim 15. By way of example, Booking-BV provides the Booking Platform for creating property listing websites for hotels, homes, apartments, and other lodgings where travelers can view and book accommodations.

[7] This Count focuses its infringement allegations on the Booking Platform, YCS Platform, and the OpenTable Platform. Upon information and belief, the WebDirect Platform operates in a similar fashion as the Booking and YCS Platforms, and discovery of the WebDirect Platform, including confidential documents related thereto, will confirm these facts.


Booking.com

Join the 29,168,959 listings already earning on Booking.com

Already a partner? Sign in Help

List Your Apartment on Booking.com

Registration can take as little as 15 minutes to complete – get started today

Create new listing

Your first and last name

Your email address

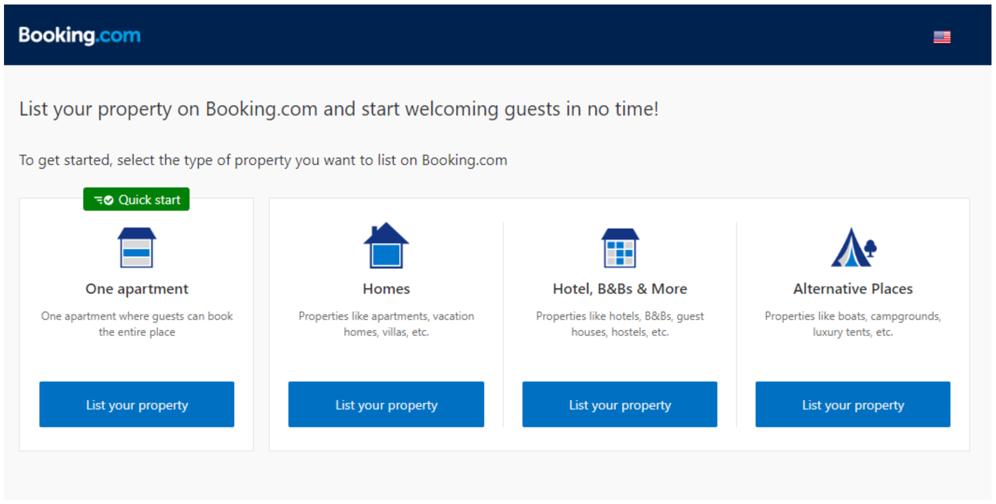
Get started →

Already started registration?

Continue your registration

By continuing, you agree to let Booking.com email you regarding your property registration.

Are you a homeowner? Check out how you can start earning with Booking.com.


Booking.com

List your property on Booking.com and start welcoming guests in no time!

To get started, select the type of property you want to list on Booking.com

Quick start

One apartment — One apartment where guests can book the entire place — List your property

Homes — Properties like apartments, vacation homes, villas, etc. — List your property

Hotel, B&Bs & More — Properties like hotels, B&Bs, guest houses, hostels, etc. — List your property

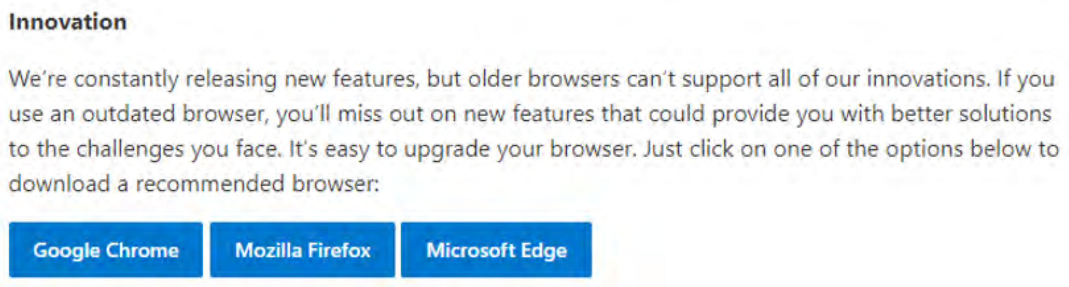
Alternative Places — Properties like boats, campgrounds, luxury tents, etc. — List your property

228. The Booking Platform displays content through a device that has a device-dependent player, such as a device-dependent code for a browser engine for a specific browser, or an operating system or application for a particular kind of device. *See, e.g.*, *Shopify Inc. v. Express Mobile, Inc.*, Case No. 1:19-cv-00439-RGA, D. Del., D.I. 137. For example, the Booking Platform

---

[8] Unless otherwise noted, the images presented in this Count were generated for investigative purposes by testing the Accused Instrumentalities on https://www.booking.com/, https://join.booking.com/, https://partner.booking.com/en-us, https://ycs.agoda.com/en-us/kipp/public/home, https://www.opentable.com/, https://support.opentable.com/s/?language=en_US, https://platform.opentable.com/documentation/, and/or other associated websites.

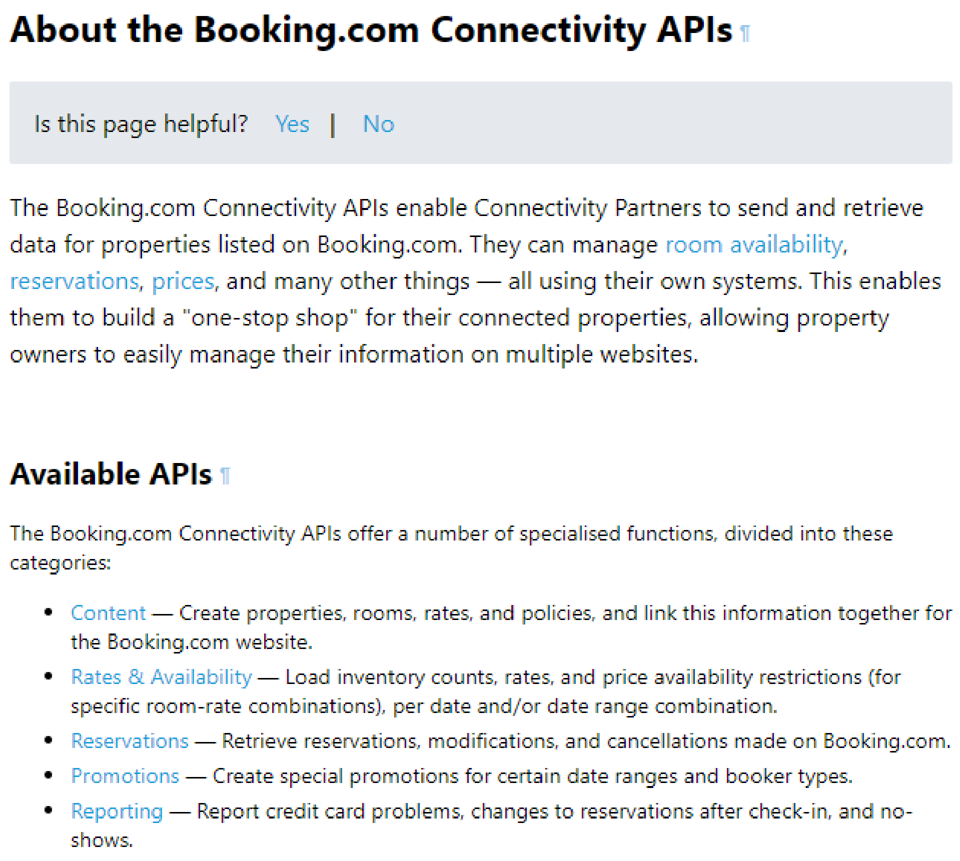
displays content through modern browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge.



Innovation

We're constantly releasing new features, but older browsers can't support all of our innovations. If you use an outdated browser, you'll miss out on new features that could provide you with better solutions to the challenges you face. It's easy to upgrade your browser. Just click on one of the options below to download a recommended browser:

229. The Booking Platform defines a UI object for presentation on the display, the UI object corresponding to a web component included in a registry of one or more web components selected from an input of a web service and an output of a web service. Web services that can be integrated on a property listing include Application Programming Interfaces (APIs) providing various kinds of services.



About the Booking.com Connectivity APIs ¶

The Booking.com Connectivity APIs enable Connectivity Partners to send and retrieve data for properties listed on Booking.com. They can manage room availability, reservations, prices, and many other things — all using their own systems. This enables them to build a "one-stop shop" for their connected properties, allowing property owners to easily manage their information on multiple websites.

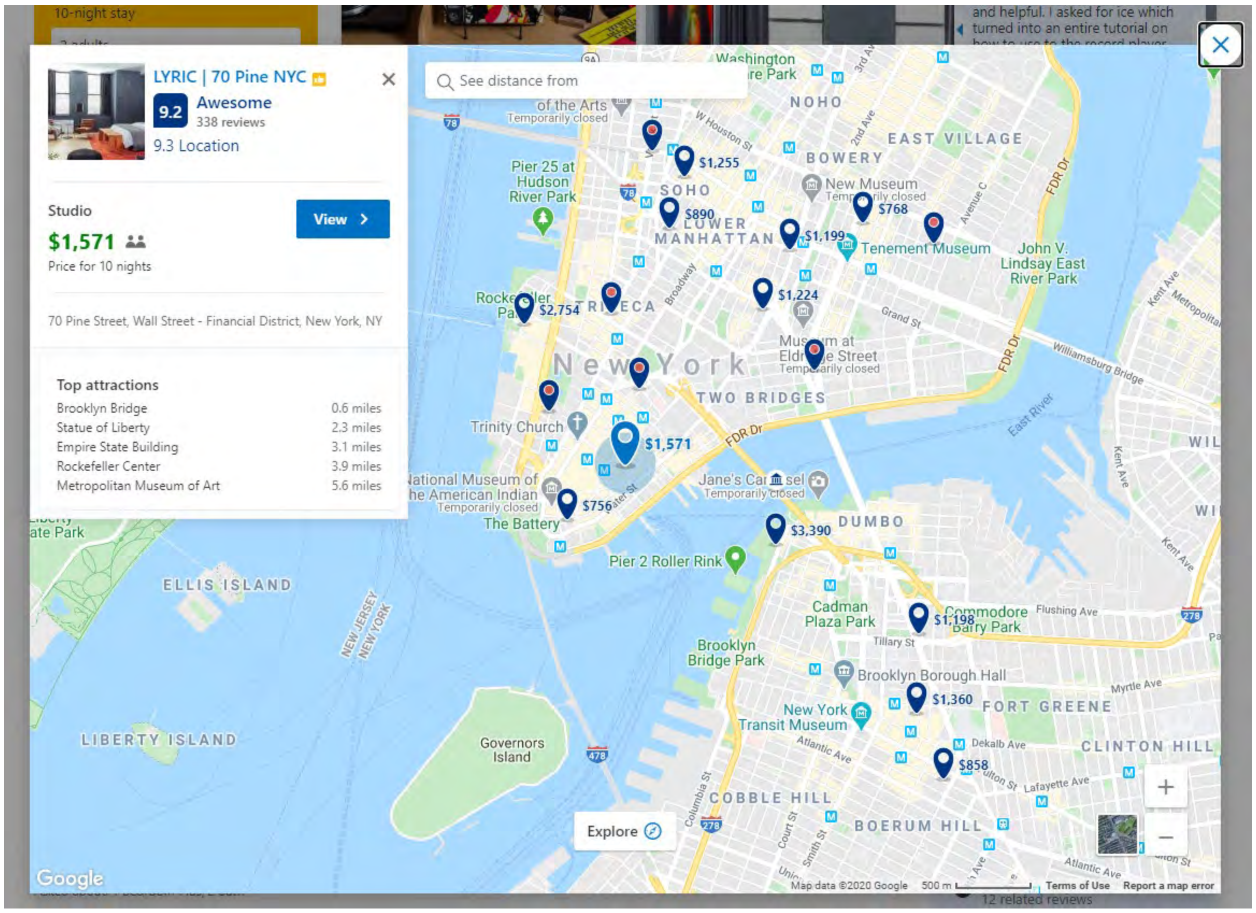
Available APIs ¶

The Booking.com Connectivity APIs offer a number of specialised functions, divided into these categories:

- Content — Create properties, rooms, rates, and policies, and link this information together for the Booking.com website.
- Rates & Availability — Load inventory counts, rates, and price availability restrictions (for specific room-rate combinations), per date and/or date range combination.
- Reservations — Retrieve reservations, modifications, and cancellations made on Booking.com.
- Promotions — Create special promotions for certain date ranges and booker types.
- Reporting — Report credit card problems, changes to reservations after check-in, and no-shows.

https://connect.booking.com/user_guide/site/en-US/

230. By way of example, when the Google Maps API is embedded onto a Booking website, UI objects for web components such as a map image, a pin indicating the property's location, zoom-in and zoom-out buttons, and pins indicating nearby attractions and points of interest are defined for the Google Map web service.


LYRIC | 70 Pine NYC

9.2 Awesome
338 reviews
9.3 Location

Studio
$1,571
Price for 10 nights

70 Pine Street, Wall Street - Financial District, New York, NY

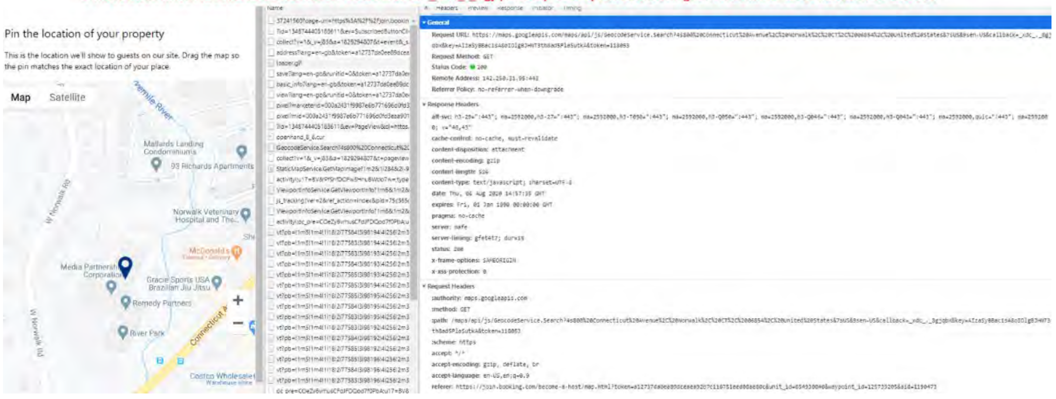
Top attractions

Brooklyn Bridge	0.6 miles
Statue of Liberty	2.3 miles
Empire State Building	3.1 miles
Rockefeller Center	3.9 miles
Metropolitan Museum of Art	5.6 miles

231. In order to store and transmit data for APIs (web services) such as the Google Maps API, the Booking Platform employs JSON data to evoke web components and communicate inputs and outputs for the APIs. JSON data comprise key/value pairs that are essentially character strings (i.e., symbolic names) with no persistent address or pointer. In the example below, when the Google Maps API is embedded on the property listing, the Booking Platform sets the location of the property "800 Connecticut Avenue. Norwalk, CT 06854" with a simple HTTP GET request.

In particular, the property address is sent to the Google Maps API's Geocoding service, which converts the property address to geographic coordinates that are sent to the Booking Platform in JSON dataset.

Request URL:
https://maps.googleapis.com/maps/api/js/GeocodeService.Search?4s800%20Connecticut%20Avenue%2C%20Norwalk%2C%20CT%2C%2006854%2C%20United%20States&7sUS&9sen-US&callback=_xdc_._8gjqbx&key=AIzaSyBBac1sA8oIOlgBJHNT3th8ad5PlaSutkA&token=118053

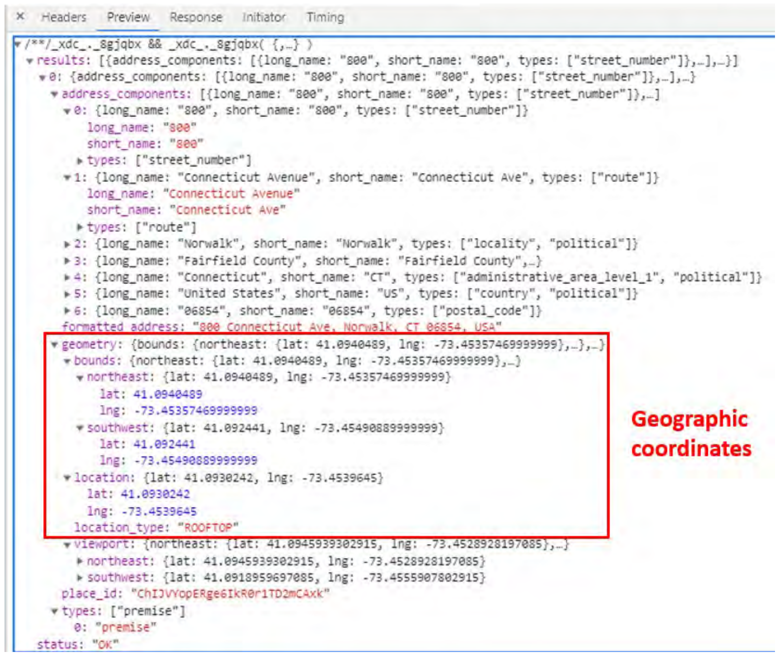

Pin the location of your property
This is the location we'll show to guests on our site. Drag the map so the pin matches the exact location of your place.
Map
Satellite
Five Mile River
Mallards Landing Condominiums
93 Richards Apartments
Norwalk Veterinary Hospital and The...
McDonald's
Media Partners Corporation
Gracie Sports USA Brazilian Jiu Jitsu
Remedy Partners
River Park
Connecticut Ave
Costco Wholesale
W Norwalk Rd

Headers Preview Response Initiator Timing

▾ General
Request URL: https://maps.googleapis.com/maps/api/js/GeocodeService.Search?4s800%20Connecticut%20Avenue%2C%20Norwalk%2C%20CT%2C%2006854%2C%20United%20States&7sUS&9sen-US&callback=_xdc_._8gjqbx&key=AIzaSyBBac1sA8oIOlgBJHNT3th8ad5PlaSutkA&token=118053
Request Method: GET
Status Code: 200
Remote Address: 142.250.31.95:443
Referrer Policy: no-referrer-when-downgrade

▾ Response Headers
alt-svc: [illegible]
cache-control: no-cache, must-revalidate
content-disposition: attachment
content-encoding: gzip
content-length: 536
content-type: text/javascript; charset=UTF-8
date: Thu, 06 Aug 2020 14:57:39 GMT
expires: Fri, 01 Jan 1990 00:00:00 GMT
pragma: no-cache
server: mafe
server-timing: gfet4t7; dur=15
status: 200
x-frame-options: SAMEORIGIN
x-xss-protection: 0

▾ Request Headers
:authority: maps.googleapis.com
:method: GET
:path: /maps/api/js/GeocodeService.Search?4s800%20Connecticut%20Avenue%2C%20Norwalk%2C%20CT%2C%2006854%2C%20United%20States&7sUS&9sen-US&callback=_xdc_._8gjqbx&key=AIzaSyBBac1sA8oIOlgBJHNT3th8ad5PlaSutkA&token=118053
:scheme: https
accept: */*
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
referer: https://join.booking.com/become-a-host/map.html?token=[illegible]&unit_id=[illegible]&waypoint_id=[illegible]&aid=[illegible]

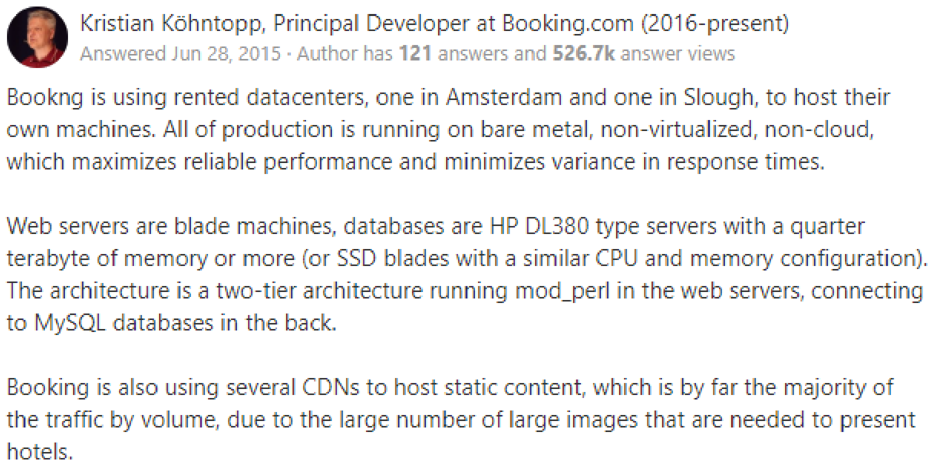


× Headers Preview Response Initiator Timing

```
▾ /**/_xdc_._8gjqbx && _xdc_._8gjqbx( {,…} )
  ▾ results: [{address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…],…}]
    ▾ 0: {address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…],…}
      ▾ address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…]
        ▾ 0: {long_name: "800", short_name: "800", types: ["street_number"]}
            long_name: "800"
            short_name: "800"
          ▸ types: ["street_number"]
        ▾ 1: {long_name: "Connecticut Avenue", short_name: "Connecticut Ave", types: ["route"]}
            long_name: "Connecticut Avenue"
            short_name: "Connecticut Ave"
          ▸ types: ["route"]
        ▸ 2: {long_name: "Norwalk", short_name: "Norwalk", types: ["locality", "political"]}
        ▸ 3: {long_name: "Fairfield County", short_name: "Fairfield County",…}
        ▸ 4: {long_name: "Connecticut", short_name: "CT", types: ["administrative_area_level_1", "political"]}
        ▸ 5: {long_name: "United States", short_name: "US", types: ["country", "political"]}
        ▸ 6: {long_name: "06854", short_name: "06854", types: ["postal_code"]}
        formatted_address: "800 Connecticut Ave, Norwalk, CT 06854, USA"
      ▾ geometry: {bounds: {northeast: {lat: 41.0940489, lng: -73.45357469999999},…},…}
        ▾ bounds: {northeast: {lat: 41.0940489, lng: -73.45357469999999},…}
          ▾ northeast: {lat: 41.0940489, lng: -73.45357469999999}
              lat: 41.0940489
              lng: -73.45357469999999
          ▾ southwest: {lat: 41.092441, lng: -73.45490889999999}
              lat: 41.092441
              lng: -73.45490889999999
        ▾ location: {lat: 41.0930242, lng: -73.4539645}
            lat: 41.0930242
            lng: -73.4539645
          location_type: "ROOFTOP"
        ▾ viewport: {northeast: {lat: 41.0945939302915, lng: -73.4528928197085},…}
          ▸ northeast: {lat: 41.0945939302915, lng: -73.4528928197085}
          ▸ southwest: {lat: 41.0918959697085, lng: -73.4555907802915}
        place_id: "ChIJVVopERge6IkR0r1TD2mCAxk"
      ▾ types: ["premise"]
          0: "premise"
  status: "OK"
```

Geographic coordinates

232. The Booking Platform's web components and symbolic names (e.g., JSON data) described above, are located in a MySQL system. MySQL is a relational database management system that operates as a back-end server supporting Booking websites.

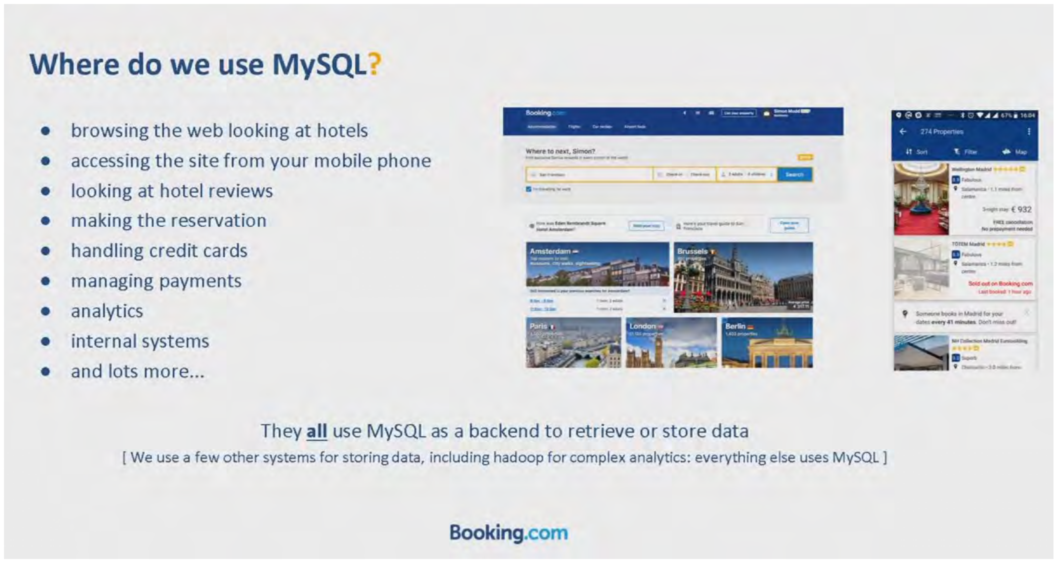

Kristian Köhntopp, Principal Developer at Booking.com (2016-present)
Answered Jun 28, 2015 · Author has **121** answers and **526.7k** answer views

Bookng is using rented datacenters, one in Amsterdam and one in Slough, to host their own machines. All of production is running on bare metal, non-virtualized, non-cloud, which maximizes reliable performance and minimizes variance in response times.

Web servers are blade machines, databases are HP DL380 type servers with a quarter terabyte of memory or more (or SSD blades with a similar CPU and memory configuration). The architecture is a two-tier architecture running mod_perl in the web servers, connecting to MySQL databases in the back.

Booking is also using several CDNs to host static content, which is by far the majority of the traffic by volume, due to the large number of large images that are needed to present hotels.

https://www.quora.com/Who-hosts-the-booking-com-website


Where do we use MySQL?

- browsing the web looking at hotels
- accessing the site from your mobile phone
- looking at hotel reviews
- making the reservation
- handling credit cards
- managing payments
- analytics
- internal systems
- and lots more...

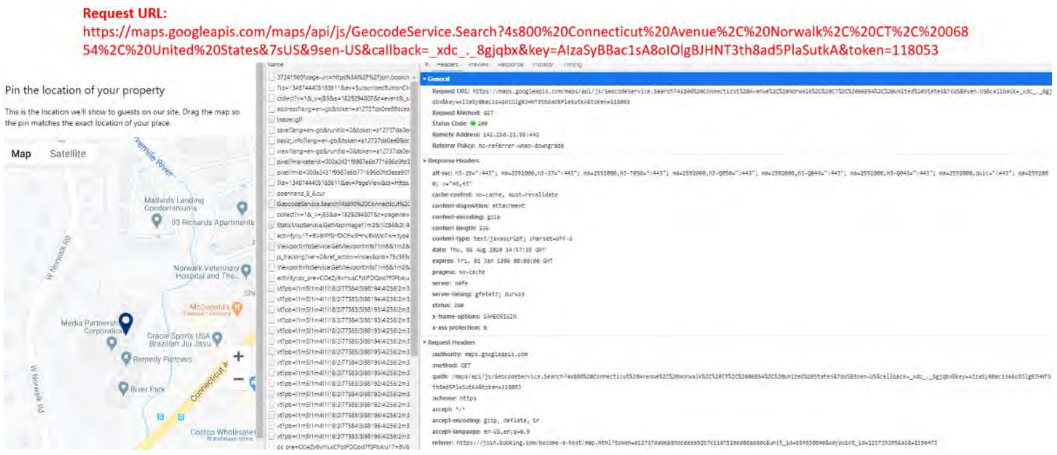
They **all** use MySQL as a backend to retrieve or store data
[We use a few other systems for storing data, including hadoop for complex analytics: everything else uses MySQL]

Booking.com

https://www.youtube.com/watch?v=iNxqZSbaHYQ&feature=youtu.be

233. The registry for the Booking Platform also includes addresses for the web services where the input symbolic names and output symbolic names can be sent to and received from. For example, the screenshot below shows a request URL including an address of the Google Maps

web service when a user inputs an address for Geocoding service in the Google Maps API. In particular, the user-input property address "800 Connecticut Avenue. Norwalk, CT 06854" is embedded in the request URL and sent to the Google Maps API via an HTTP GET request. The Geocoding service in the Google Maps API then responds with geographic coordinates for the user-input address in the format of JSON data and places a marker on the map indicating the property's location. The geographic coordinates and the user input address are stored in the Booking's database for further use.



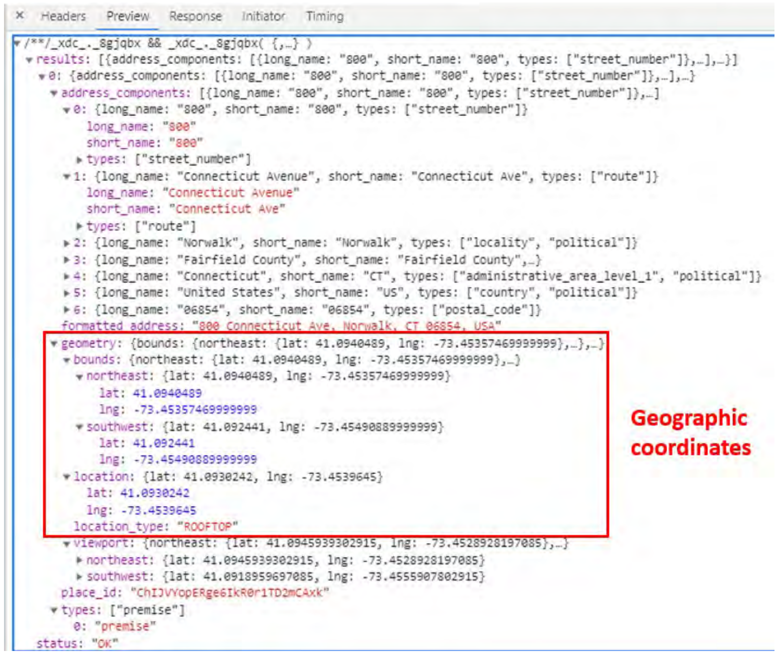
Request URL:
https://maps.googleapis.com/maps/api/js/GeocodeService.Search?4s800%20Connecticut%20Avenue%2C%20Norwalk%2C%20CT%2C%2006854%2C%20United%20States&7sUS&9sen-US&callback=_xdc_._8gjqbx&key=AIzaSyBBac1sA8oIOlgBJHNT3th8ad5PlaSutkA&token=118053

Pin the location of your property

This is the location we'll show to guests on our site. Drag the map so the pin matches the exact location of your place.

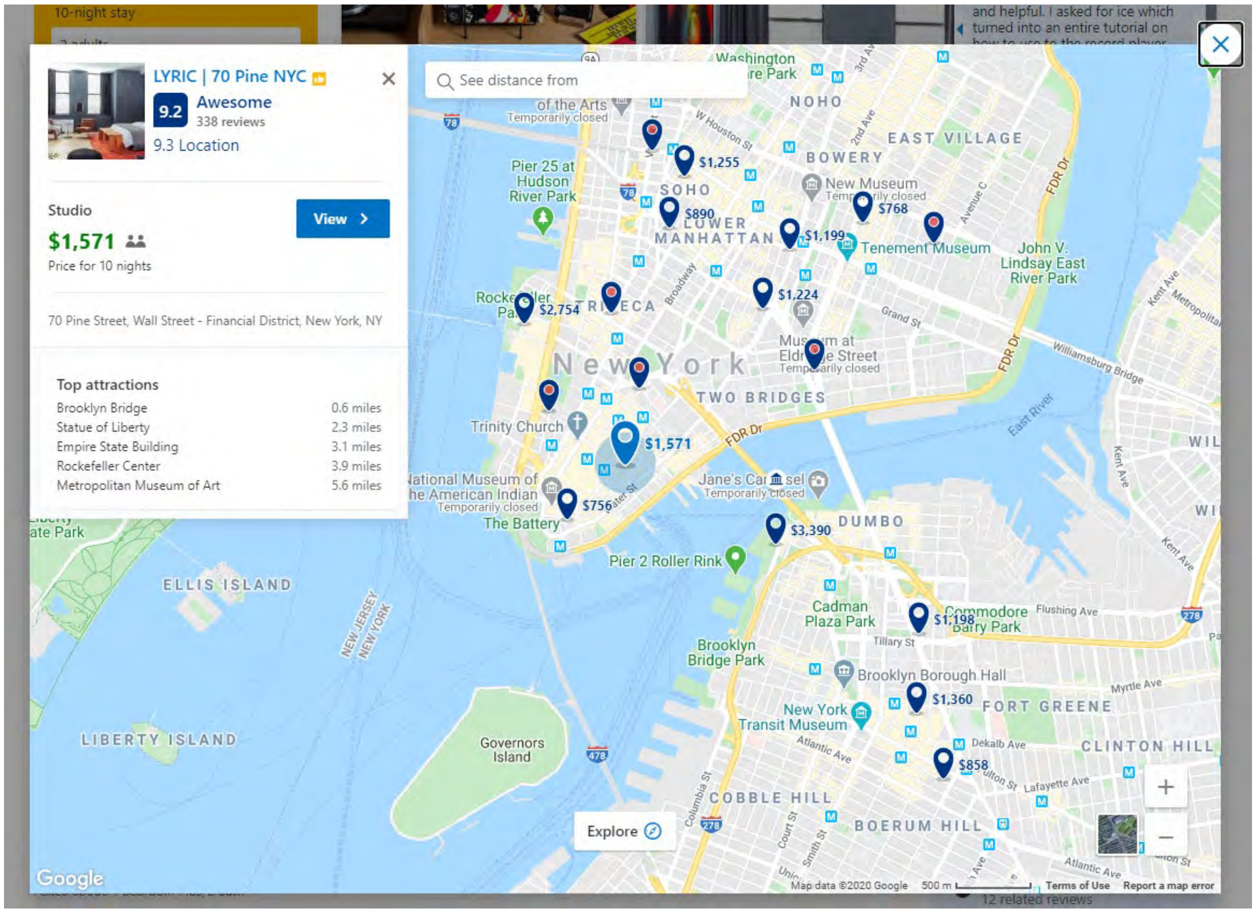
Map
Satellite
Mallards Landing Condominiums
93 Richards Apartments
Norwalk Veterinary Hospital and The...
McDonald's
Media Partners Corporation
Gracie Sports USA Brazilian Jiu Jitsu
Remedy Partners
River Park
Costco Wholesale
Connecticut Ave
W Norwalk Rd

```
▾ General
  Request URL: https://maps.googleapis.com/maps/api/js/GeocodeService.Search?4s800%20Connecticut%20Avenue%2C%20Norwalk%2C%20CT%2C%2006854%2C%20United%20States&7sUS&9sen-US&callback=_xdc_._8gjqbx&key=AIzaSyBBac1sA8oIOlgBJHNT3th8ad5PlaSutkA&token=118053
  Request Method: GET
  Status Code: ● 200
  Remote Address: 142.250.21.95:443
  Referrer Policy: no-referrer-when-downgrade
▾ Response Headers
  alt-svc: h3-29=":443"; ma=2592000,h3-27=":443"; ma=2592000,h3-T050=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"
  cache-control: no-cache, must-revalidate
  content-disposition: attachment
  content-encoding: gzip
  content-length: 516
  content-type: text/javascript; charset=UTF-8
  date: Thu, 06 Aug 2020 14:57:35 GMT
  expires: Fri, 01 Jan 1990 00:00:00 GMT
  pragma: no-cache
  server: mafe
  server-timing: gfet4t7; dur=35
  status: 200
  x-frame-options: SAMEORIGIN
  x-xss-protection: 0
▾ Request Headers
  :authority: maps.googleapis.com
  :method: GET
  :path: /maps/api/js/GeocodeService.Search?4s800%20Connecticut%20Avenue%2C%20Norwalk%2C%20CT%2C%2006854%2C%20United%20States&7sUS&9sen-US&callback=_xdc_._8gjqbx&key=AIzaSyBBac1sA8oIOlgBJHNT3th8ad5PlaSutkA&token=118053
  :scheme: https
  accept: */*
  accept-encoding: gzip, deflate, br
  accept-language: en-US,en;q=0.9
  referer: https://join.booking.com/become-a-host/map.html?token=[illegible]
```



```
▾ /**/_xdc_._8gjqbx && _xdc_._8gjqbx( {,…} )
  ▾ results: [{address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…],…}]
    ▾ 0: {address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…],…}
      ▾ address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…]
        ▾ 0: {long_name: "800", short_name: "800", types: ["street_number"]}
            long_name: "800"
            short_name: "800"
          ▸ types: ["street_number"]
        ▾ 1: {long_name: "Connecticut Avenue", short_name: "Connecticut Ave", types: ["route"]}
            long_name: "Connecticut Avenue"
            short_name: "Connecticut Ave"
          ▸ types: ["route"]
        ▸ 2: {long_name: "Norwalk", short_name: "Norwalk", types: ["locality", "political"]}
        ▸ 3: {long_name: "Fairfield County", short_name: "Fairfield County",…}
        ▸ 4: {long_name: "Connecticut", short_name: "CT", types: ["administrative_area_level_1", "political"]}
        ▸ 5: {long_name: "United States", short_name: "US", types: ["country", "political"]}
        ▸ 6: {long_name: "06854", short_name: "06854", types: ["postal_code"]}
      formatted_address: "800 Connecticut Ave, Norwalk, CT 06854, USA"
    ▾ geometry: {bounds: {northeast: {lat: 41.0940489, lng: -73.45357469999999},…},…}
      ▾ bounds: {northeast: {lat: 41.0940489, lng: -73.45357469999999},…}
        ▾ northeast: {lat: 41.0940489, lng: -73.45357469999999}
            lat: 41.0940489
            lng: -73.45357469999999
        ▾ southwest: {lat: 41.092441, lng: -73.45490889999999}
            lat: 41.092441
            lng: -73.45490889999999
      ▾ location: {lat: 41.0930242, lng: -73.4539645}
          lat: 41.0930242
          lng: -73.4539645
        location_type: "ROOFTOP"
      ▾ viewport: {northeast: {lat: 41.0945939302915, lng: -73.4528928197085},…}
        ▸ northeast: {lat: 41.0945939302915, lng: -73.4528928197085}
        ▸ southwest: {lat: 41.0918959697085, lng: -73.4555907802915}
      place_id: "ChIJVYopERge6IkR0r1TD2mCAxk"
    ▾ types: ["premise"]
        0: "premise"
  status: "OK"
```

234. The defined UI objects described above are either selected by a user of an authoring tool or automatically selected by the Booking Platform as preferred UI objects. For example, when the Google Maps API is integrated onto a property listing as described above, UI objects for web components such as the zoom-in and zoom-out buttons, and pins indicating nearby hotels and points of interest are automatically selected by the Booking Platform as preferred UI objects.



LYRIC | 70 Pine NYC

9.2 Awesome

338 reviews

9.3 Location

Studio

$1,571

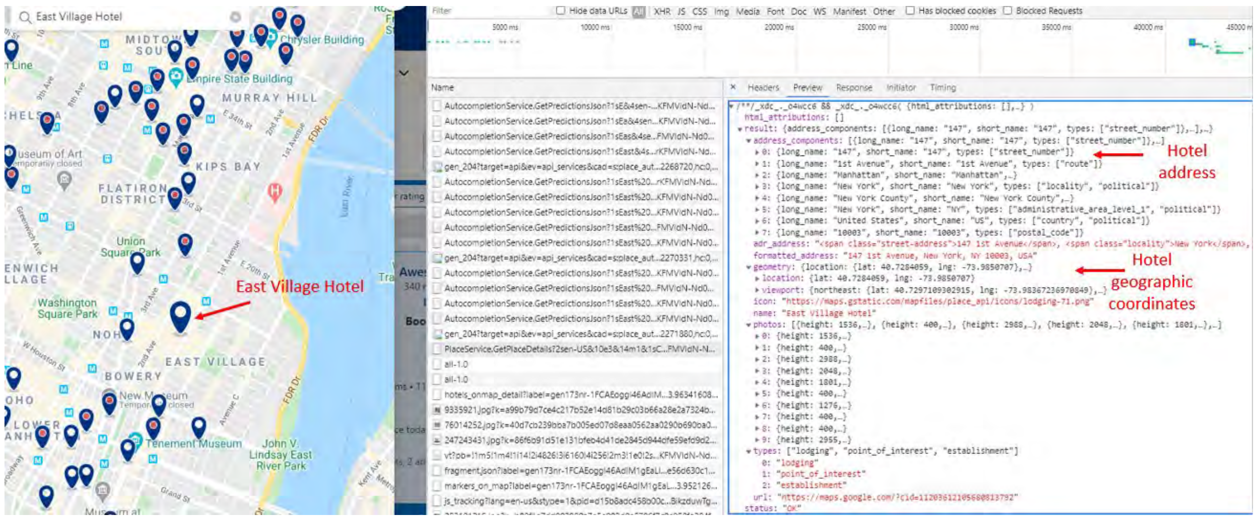
Price for 10 nights

70 Pine Street, Wall Street - Financial District, New York, NY

Top attractions	
Brooklyn Bridge	0.6 miles
Statue of Liberty	2.3 miles
Empire State Building	3.1 miles
Rockefeller Center	3.9 miles
Metropolitan Museum of Art	5.6 miles

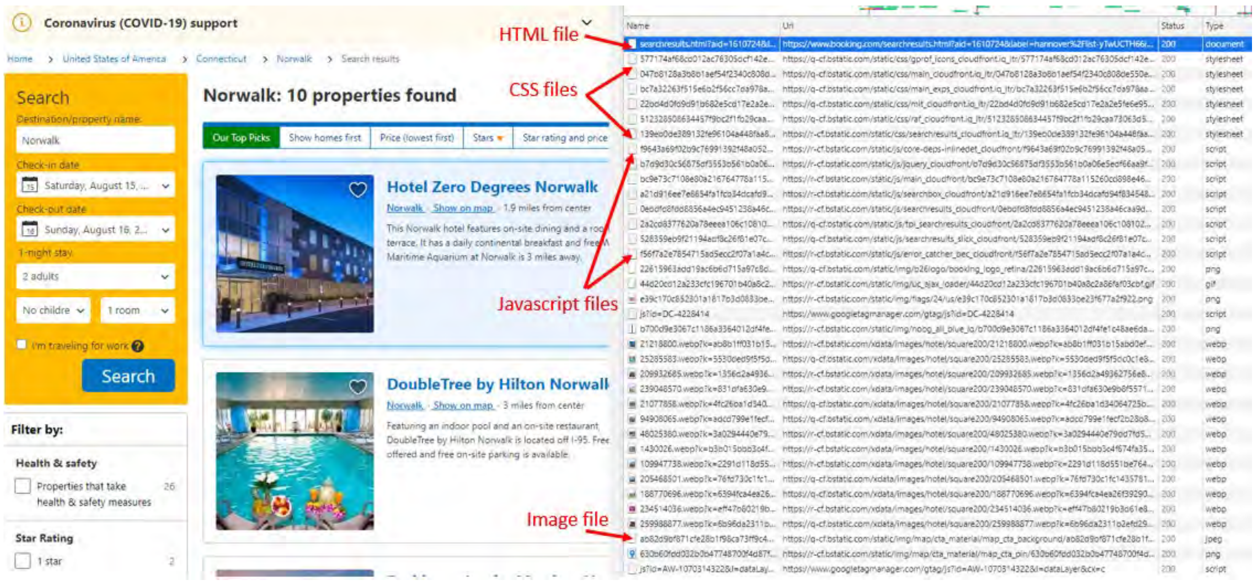
235. When a UI object is so defined, a symbolic name (e.g., JSON data) from the web component corresponding to the defined UI object is selected and associated with the defined UI object. The symbolic name has an associated data format class type corresponding to a subclass of UI objects that support the data format type of the symbolic name and has the preferred UI object. The example below demonstrates how a user can search for "East Village Hotel" in the Google Maps API on a property listing. When the hotel name "East Village Hotel" in entered into the search box of the Google Maps API, the Google Maps web service responds by sending a

JSON dataset to the Booking Platform. In particular, the symbolic names "address_components" and "formatted_address" associated with the hotel address, and the symbolic name "geometry" associated with the geographic coordinates of the hotel are used to display the location of the property on the Google Maps API. In the example below, a large pin marker object associated with the address and geographic coordinates of the hotel is dropped on the map to indicate the location of the "East Village Hotel." The large pin marker object is a preferred UI object to represent the target hotel, which is distinguishable from other pins indicating nearby hotels and points of interest.



```
/**/_xdc_._o4wcc6 && _xdc_._o4wcc6( {html_attributions: [],…} )
  html_attributions: []
  result: {address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…],…}
    address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…]
      0: {long_name: "147", short_name: "147", types: ["street_number"]}
      1: {long_name: "1st Avenue", short_name: "1st Avenue", types: ["route"]}
      2: {long_name: "Manhattan", short_name: "Manhattan",…}
      3: {long_name: "New York", short_name: "New York", types: ["locality", "political"]}
      4: {long_name: "New York County", short_name: "New York County",…}
      5: {long_name: "New York", short_name: "NY", types: ["administrative_area_level_1", "political"]}
      6: {long_name: "United States", short_name: "US", types: ["country", "political"]}
      7: {long_name: "10003", short_name: "10003", types: ["postal_code"]}
    adr_address: "<span class="street-address">147 1st Avenue</span>, <span class="locality">New York</span>,
    formatted_address: "147 1st Avenue, New York, NY 10003, USA"
    geometry: {location: {lat: 40.7284059, lng: -73.9850707},…}
      location: {lat: 40.7284059, lng: -73.9850707}
      viewport: {northeast: {lat: 40.7297109302915, lng: -73.98367236970849},…}
    icon: "https://maps.gstatic.com/mapfiles/place_api/icons/lodging-71.png"
    name: "East Village Hotel"
    photos: [{height: 1536,…}, {height: 400,…}, {height: 2988,…}, {height: 2048,…}, {height: 1801,…},…]
      0: {height: 1536,…}
      1: {height: 400,…}
      2: {height: 2988,…}
      3: {height: 2048,…}
      4: {height: 1801,…}
      5: {height: 400,…}
      6: {height: 1276,…}
      7: {height: 400,…}
      8: {height: 400,…}
      9: {height: 2955,…}
    types: ["lodging", "point_of_interest", "establishment"]
      0: "lodging"
      1: "point_of_interest"
      2: "establishment"
    url: "https://maps.google.com/?cid=11203612105680813792"
  status: "OK"
```

236. The Booking Platform then produces a device-independent application that includes the JSON data, as well as standard HTML, CSS and Javascript code. The application is executed on a device together with a device-dependent player, such as a device-dependent code for a browser engine for a specific browser, or an operating system or application for a particular kind of device. *See, e.g.*, *Shopify Inc. v. Express Mobile, Inc.*, Case No. 1:19-cv-00439-RGA, D. Del., D.I. 137. In the example below, the Booking Platform converts HTML, CSS, Javascript, image, and other files into an active property website.



Coronavirus (COVID-19) support

Home > United States of America > Connecticut > Norwalk > Search results

Search
Destination/property name:
Norwalk
Check-in date
Saturday, August 15, ...
Check-out date
Sunday, August 16, 2...
1-night stay
2 adults
No children
1 room
I'm traveling for work
Search

Filter by:
Health & safety
Properties that take health & safety measures 26
Star Rating
1 star 2

Norwalk: 10 properties found

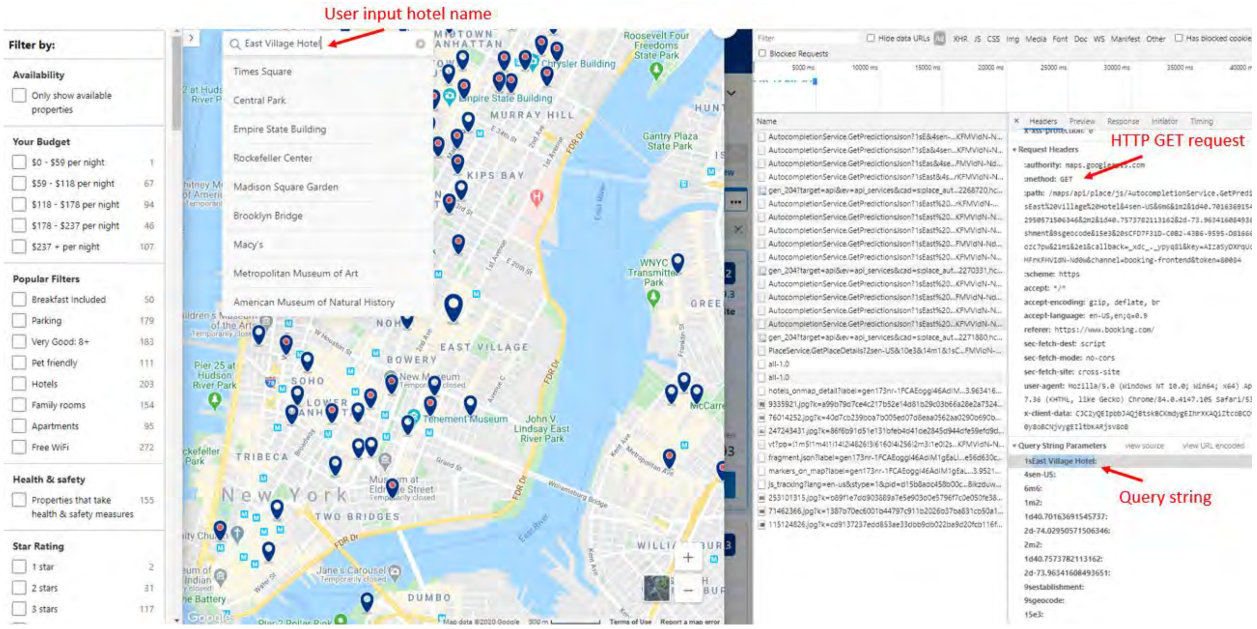
Our Top Picks | Show homes first | Price (lowest first) | Stars | Star rating and price

Hotel Zero Degrees Norwalk
Norwalk · Show on map · 1.9 miles from center
This Norwalk hotel features on-site dining and a roo terrace. It has a daily continental breakfast and free Maritime Aquarium at Norwalk is 3 miles away.

DoubleTree by Hilton Norwalk
Norwalk · Show on map · 3 miles from center
Featuring an indoor pool and an on-site restaurant, DoubleTree by Hilton Norwalk is located off I-95. Free offered and free on-site parking is available.

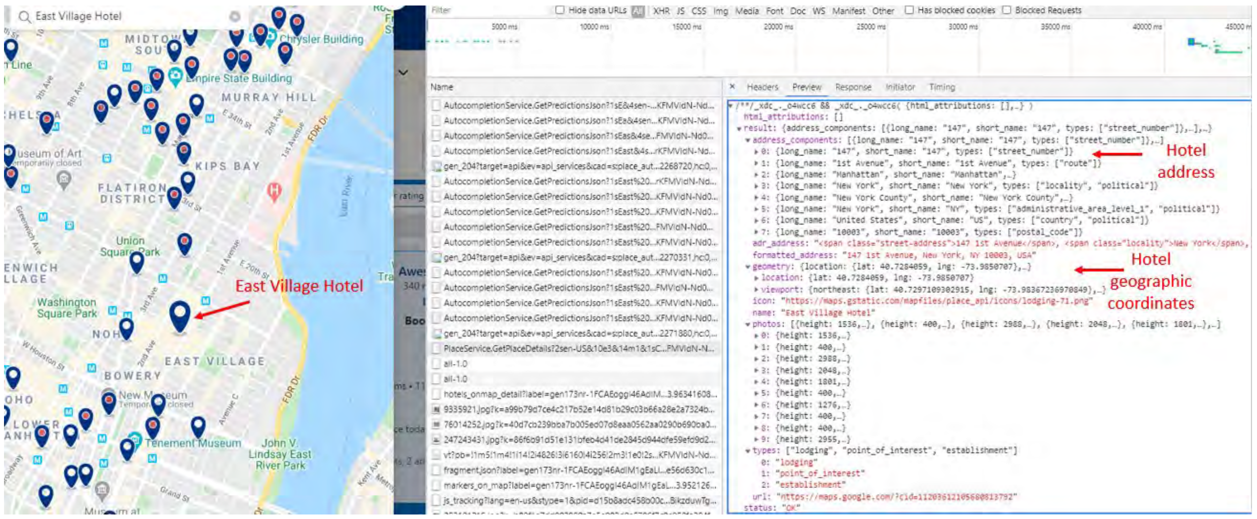
Name	Url	Status	Type
searchresults.html?aid=1610724&l...	https://www.booking.com/searchresults.html?aid=1610724&label=hannover%2Flist-yTwUCTH66i...	200	document
577174af68cd012ac76305dcf142e...	https://q-cf.bstatic.com/static/css/gprof_icons_cloudfront.iq_ltr/577174af68cd012ac76305dcf142e...	200	stylesheet
047b8128a3b8b1aef54f2340c808d...	https://q-cf.bstatic.com/static/css/main_cloudfront.iq_ltr/047b8128a3b8b1aef54f2340c808de550e...	200	stylesheet
bc7a32263f515e6b2f56cc7da978a...	https://q-cf.bstatic.com/static/css/main_exps_cloudfront.iq_ltr/bc7a32263f515e6b2f56cc7da978aa...	200	stylesheet
22bd4d0fd9d91b682e5cd17e2a2e...	https://q-cf.bstatic.com/static/css/mlt_cloudfront.iq_ltr/22bd4d0fd9d91b682e5cd17e2a2e5fe6e95...	200	stylesheet
5123285086344579bc2f1fb29caa...	https://q-cf.bstatic.com/static/css/raf_cloudfront.iq_ltr/5123285086344579bc2f1fb29caa73063d5...	200	stylesheet
139eb0de389132fe96104a448faa8...	https://r-cf.bstatic.com/static/css/searchresults_cloudfront.iq_ltr/139eb0de389132fe96104a448faa...	200	stylesheet
f9643a69f02b9c76991392f48a052...	https://r-cf.bstatic.com/static/js/core-deps-inlinedet_cloudfront/f9643a69f02b9c76991392f48a05...	200	script
b7d9d30c56875df3553b561b0a06...	https://r-cf.bstatic.com/static/js/jquery_cloudfront/b7d9d30c56875df3553b561b0a06e5edf66aa9f...	200	script
bc9e73c7108e80a216764778a115...	https://r-cf.bstatic.com/static/js/main_cloudfront/bc9e73c7108e80a216764778a115260cd898e46...	200	script
a21d916ee7e8654fa1fcb34dcafd9...	https://r-cf.bstatic.com/static/js/searchbox_cloudfront/a21d916ee7e8654fa1fcb34dcafd94f834548...	200	script
0ebdfd8fdd8856a4ec9451238a46c...	https://r-cf.bstatic.com/static/js/searchresults_cloudfront/0ebdfd8fdd8856a4ec9451238a46caa9d...	200	script
2a2cd8377620a78eeea106c10810...	https://q-cf.bstatic.com/static/js/tpi_searchresults_cloudfront/2a2cd8377620a78eeea106c108102...	200	script
528359eb9f21194adf8c26f81e07c...	https://q-cf.bstatic.com/static/js/searchresults_slick_cloudfront/528359eb9f21194adf8c26f81e07c...	200	script
f56f7a2e7854715ad5ecc2f07a1a4c...	https://r-cf.bstatic.com/static/js/error_catcher_bec_cloudfront/f56f7a2e7854715ad5ecc2f07a1a4c...	200	script
22615963add19ac6b6d715a97c8d...	https://q-cf.bstatic.com/static/img/b26logo/booking_logo_retina/22615963add19ac6b6d715a97c...	200	png
44d20cd12a233cfc196701b40a8c2...	https://r-cf.bstatic.com/static/img/uc_ajax_loader/44d20cd12a233cfc196701b40a8c2a86faf03cbf.gif	200	gif
e39c170c852301a1817b3d0833be...	https://r-cf.bstatic.com/static/img/flags/24/us/e39c170c852301a1817b3d0833be23f677a2f922.png	200	png
js?id=DC-4228414	https://www.googletagmanager.com/gtag/js?id=DC-4228414	200	script
b700d9e3067c1186a3364012df4fe...	https://r-cf.bstatic.com/static/img/nobg_all_blue_iq/b700d9e3067c1186a3364012df4fe1c48ae6da...	200	png
21218800.webp?k=ab8b1ff031b15...	https://r-cf.bstatic.com/xdata/images/hotel/square200/21218800.webp?k=ab8b1ff031b15abd0ef...	200	webp
25285583.webp?k=5530ded9f5f5d...	https://q-cf.bstatic.com/xdata/images/hotel/square200/25285583.webp?k=5530ded9f5f5dc0c1e8...	200	webp
209932685.webp?k=1356d2a4936...	https://r-cf.bstatic.com/xdata/images/hotel/square200/209932685.webp?k=1356d2a49362756e8...	200	webp
239048570.webp?k=831dfa630e9...	https://r-cf.bstatic.com/xdata/images/hotel/square200/239048570.webp?k=831dfa630e9b8f5571...	200	webp
21077858.webp?k=4fc26ba1d340...	https://q-cf.bstatic.com/xdata/images/hotel/square200/21077858.webp?k=4fc26ba1d34064725b...	200	webp
94908065.webp?k=adcd799e1fecf...	https://q-cf.bstatic.com/xdata/images/hotel/square200/94908065.webp?k=adcd799e1fecf2b2808...	200	webp
48025380.webp?k=3a0294440e79...	https://r-cf.bstatic.com/xdata/images/hotel/square200/48025380.webp?k=3a0294440e79dd7fd5...	200	webp
1430026.webp?k=b3b015bbb3d4f...	https://r-cf.bstatic.com/xdata/images/hotel/square200/1430026.webp?k=b3b015bbb3d4f674fa35...	200	webp
109947738.webp?k=2291d118d55...	https://r-cf.bstatic.com/xdata/images/hotel/square200/109947738.webp?k=2291d118d551be764...	200	webp
205468501.webp?k=76fd730c1fc1...	https://q-cf.bstatic.com/xdata/images/hotel/square200/205468501.webp?k=76fd730c1fc1435781...	200	webp
188770696.webp?k=6394fca4ea26...	https://q-cf.bstatic.com/xdata/images/hotel/square200/188770696.webp?k=6394fca4ea26f39290...	200	webp
234514036.webp?k=eff47b80219b...	https://r-cf.bstatic.com/xdata/images/hotel/square200/234514036.webp?k=eff47b80219b3d61e8...	200	webp
259988877.webp?k=6b96da2311b...	https://q-cf.bstatic.com/xdata/images/hotel/square200/259988877.webp?k=6b96da2311b2efd29...	200	webp
ab82d9bf871cfe28b1f98ca73ff9c4...	https://q-cf.bstatic.com/static/img/map/cta_material/map_cta_background/ab82d9bf871cfe28b1f...	200	jpeg
630b60fdd032b0b47748700f4d87f...	https://r-cf.bstatic.com/static/img/map/cta_material/map_cta_pin/630b60fdd032b0b47748700f4d...	200	png
js?id=AW-1070314322&l=dataLay...	https://www.googletagmanager.com/gtag/js?id=AW-1070314322&l=dataLayer&cx=c	200	script

237. When the application and player are executed on the device, an input value can be provided by a user. The device then provides the input value and a corresponding input symbolic name to the web service, which in turn generates an output value and a corresponding symbolic name for the output. By way of example, when the Google Maps API accepts an input value (e.g., a hotel name, a zoom-in click, or a click on a nearby hotel) from a user into the API, a corresponding input symbolic name (e.g., a query string or JSON data for the input) is transmitted to the Google Maps web service through an HTTP request protocol, such as a GET method call. The example below demonstrates how a user can search for "East Village Hotel" in the Google Maps API on a property listing. When the user types "East Village Hotel," a query string with parameters including the symbolic name "1s" and associated value "East Village Hotel" is sent to the Google Maps web service to initiate an autocompletion service with an HTTP GET request.



```
Request Headers
:authority: maps.googleapis.com
:method: GET
:path: /maps/api/place/js/AutocompletionService.GetPredi
sEast%20Village%20Hotel&4sen-US&6m6&1m2&1d40.7016369154
29505715063468&2m2&1d40.7573782113162&2d-73.963416084936
shment&9sgeocode&15e3&20sCFD7F31D-C0B2-43B6-9595-D8166
ozc7pw&21m1&2e1&callback=_xdc_._ypyq8i&key=AIzaSyDXroU
HFrKFHVIdN-Nd0w&channel=booking-frontend&token=80084
:scheme: https
accept: */*
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
referer: https://www.booking.com/
sec-fetch-dest: script
sec-fetch-mode: no-cors
sec-fetch-site: cross-site
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) Ap
7.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/5
x-client-data: CJC2yQEIpbbJAQjBtskBCKmdygEIhrXKAQiZtcoBCO
0y8oBCNjVygEIltbKARjsv8oB
Query String Parameters
1sEast Village Hotel:
4sen-US:
6m6:
1m2:
1d40.70163691545737:
2d-74.02950571506346:
2m2:
1d40.7573782113162:
2d-73.96341608493651:
9sestablishment:
9sgeocode:
15e3:
```

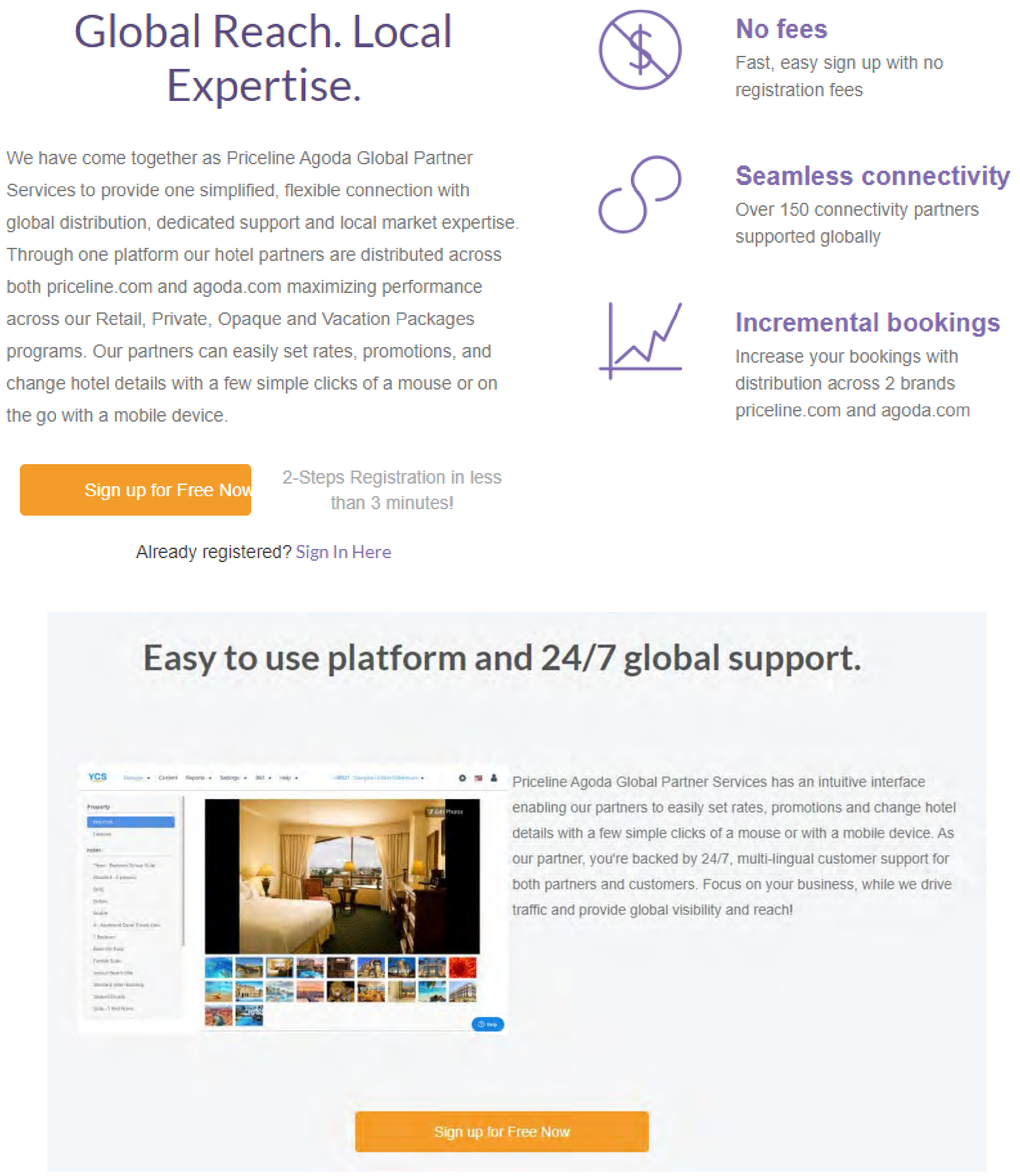
238. The input symbolic name is utilized by the Booking server to generate an output value with an associated output symbolic name (e.g., JSON data for the output). The player then provides instructions to the display of the device to present and display the output in the Google Maps API. For example, when a user searches "East Village Hotel" and selects an autocomplete suggestion from the Google Maps API, the Google Maps web service generates and transmits a corresponding hotel address and geographic coordinates in the format of JSON data to the web browser and displays a pin indicating the location of the selected hotel on the Google Maps API.



```
/**/_xdc_._o4wcc6 && _xdc_._o4wcc6( {html_attributions: [],…} )
  html_attributions: []
  result: {address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…],…}
    address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…]
      0: {long_name: "147", short_name: "147", types: ["street_number"]}
      1: {long_name: "1st Avenue", short_name: "1st Avenue", types: ["route"]}
      2: {long_name: "Manhattan", short_name: "Manhattan",…}
      3: {long_name: "New York", short_name: "New York", types: ["locality", "political"]}
      4: {long_name: "New York County", short_name: "New York County",…}
      5: {long_name: "New York", short_name: "NY", types: ["administrative_area_level_1", "political"]}
      6: {long_name: "United States", short_name: "US", types: ["country", "political"]}
      7: {long_name: "10003", short_name: "10003", types: ["postal_code"]}
    adr_address: "<span class="street-address">147 1st Avenue</span>, <span class="locality">New York</span>,
    formatted_address: "147 1st Avenue, New York, NY 10003, USA"
    geometry: {location: {lat: 40.7284059, lng: -73.9850707},…}
      location: {lat: 40.7284059, lng: -73.9850707}
      viewport: {northeast: {lat: 40.7297109302915, lng: -73.98367236970849},…}
    icon: "https://maps.gstatic.com/mapfiles/place_api/icons/lodging-71.png"
    name: "East Village Hotel"
    photos: [{height: 1536,…}, {height: 400,…}, {height: 2988,…}, {height: 2048,…}, {height: 1801,…},…]
      0: {height: 1536,…}
      1: {height: 400,…}
      2: {height: 2988,…}
      3: {height: 2048,…}
      4: {height: 1801,…}
      5: {height: 400,…}
      6: {height: 1276,…}
      7: {height: 400,…}
      8: {height: 400,…}
      9: {height: 2955,…}
    types: ["lodging", "point_of_interest", "establishment"]
      0: "lodging"
      1: "point_of_interest"
      2: "establishment"
    url: "https://maps.google.com/?cid=11203612105680813792"
  status: "OK"
```

<u>The YCS Platform:</u>

239. The YCS infringers infringe at least claim 15 of the ‘287 patent through a combination of features in the YCS Platform that collectively practice each claimed limitation of claim 15. By way of example, the YCS infringers provide the YCS Platform for creating property listing websites for hotels and other lodgings where travelers can view and book accommodations.



Global Reach. Local Expertise.

We have come together as Priceline Agoda Global Partner Services to provide one simplified, flexible connection with global distribution, dedicated support and local market expertise. Through one platform our hotel partners are distributed across both priceline.com and agoda.com maximizing performance across our Retail, Private, Opaque and Vacation Packages programs. Our partners can easily set rates, promotions, and change hotel details with a few simple clicks of a mouse or on the go with a mobile device.

Sign up for Free Now

2-Steps Registration in less than 3 minutes!

Already registered? Sign In Here

No fees

Fast, easy sign up with no registration fees

Seamless connectivity

Over 150 connectivity partners supported globally

Incremental bookings

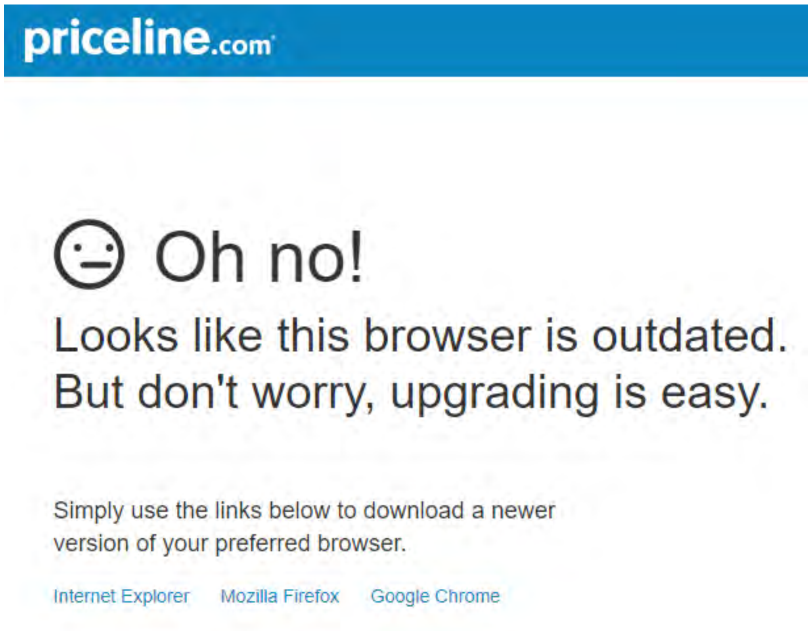
Increase your bookings with distribution across 2 brands priceline.com and agoda.com

Easy to use platform and 24/7 global support.

Priceline Agoda Global Partner Services has an intuitive interface enabling our partners to easily set rates, promotions and change hotel details with a few simple clicks of a mouse or with a mobile device. As our partner, you're backed by 24/7, multi-lingual customer support for both partners and customers. Focus on your business, while we drive traffic and provide global visibility and reach!

Sign up for Free Now

240. The YCS Platform displays content through a device that has a device-dependent player, such as a device-dependent code for a browser engine for a specific browser, or an operating system or application for a particular kind of device. *See, e.g., Shopify Inc. v. Express Mobile, Inc.*, Case No. 1:19-cv-00439-RGA, D. Del., D.I. 137. For example, the YCS Platform displays content through modern browsers such as Google Chrome, Mozilla Firefox, and Internet Explorer.


priceline.com

Oh no!

Looks like this browser is outdated.
But don't worry, upgrading is easy.

Simply use the links below to download a newer version of your preferred browser.

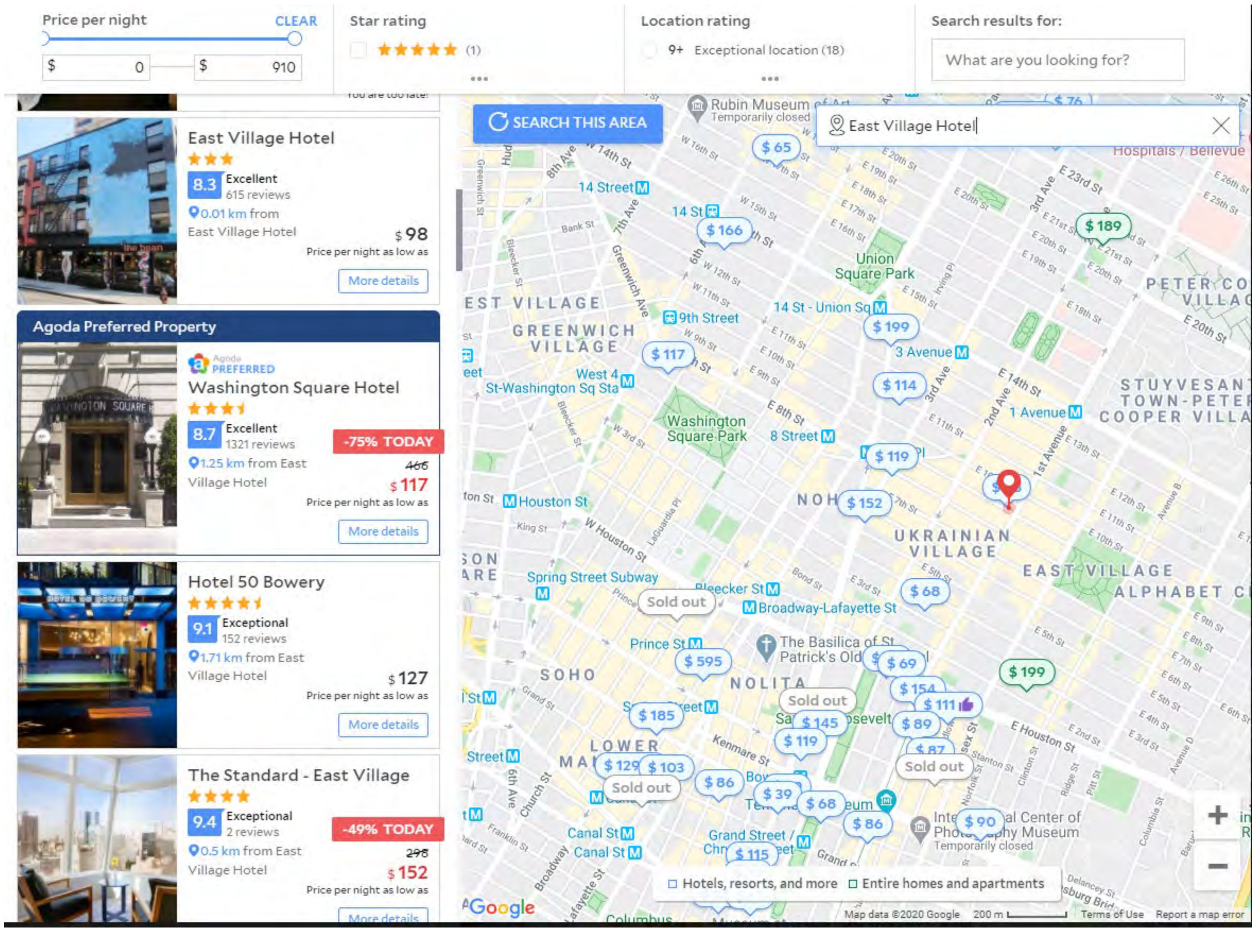
Internet Explorer Mozilla Firefox Google Chrome

https://www.priceline.com/static-pages/browser-upgrade.html

241. The YCS Platform defines a UI object for presentation on the display, where the UI object corresponds to a web component included in a registry of one or more web components selected from an input of a web service and an output of a web service. Web services that can be integrated on the property listings include Application Programming Interfaces (APIs) such as the Google Maps API.

242. By way of example, when a Google Maps API is integrated on a property listing to embed an interactive map of the property, UI objects for web components such as zoom-in and zoom-out buttons, a search field, a search button, a slide bar for "Price per night," a check box for

"Star rating," and a radio button for "Location rating" are automatically selected by the YCS Platform as preferred UI objects.



Price per night CLEAR
$ 0 $ 910

Star rating
★★★★★ (1)

Location rating
9+ Exceptional location (18)

Search results for:
What are you looking for?

East Village Hotel
★★★
8.3 Excellent
615 reviews
0.01 km from East Village Hotel
$ 98
Price per night as low as
More details

Agoda Preferred Property
Agoda PREFERRED
Washington Square Hotel
★★★½
8.7 Excellent
1321 reviews
1.25 km from East Village Hotel
-75% TODAY
~~466~~
$ 117
Price per night as low as
More details

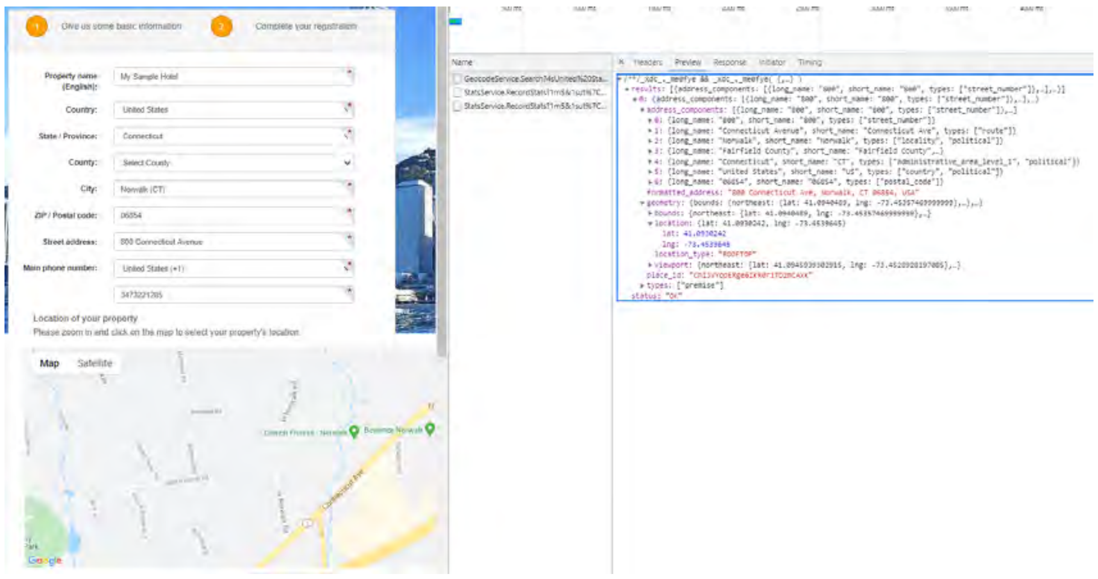
Hotel 50 Bowery
★★★★½
9.1 Exceptional
152 reviews
1.71 km from East Village Hotel
$ 127
Price per night as low as
More details

The Standard - East Village
★★★★
9.4 Exceptional
2 reviews
0.5 km from East Village Hotel
-49% TODAY
~~298~~
$ 152
Price per night as low as
More details

SEARCH THIS AREA
East Village Hotel

Hotels, resorts, and more
Entire homes and apartments
Google
Map data ©2020 Google 200 m
Terms of Use Report a map error

243. On information and belief, Priceline and Agoda websites utilize JSON data to evoke web components for the APIs and to communicate inputs and outputs for the APIs. JSON data comprise key/value pairs that are essentially character strings (i.e., symbolic names) with no persistent address or pointer. For example, an Agoda property listing website can include a Google Maps API integrated on the website to display the location of the property listing. The Google Maps API utilizes symbolic names (e.g., JSON data) to transfer the property address information and geographic coordinates to the Agoda website. In the example below, "lat" and "lng" are symbolic names associated with the latitude and longitude coordinates of the property address "800 Connecticut Avenue, Norwalk, CT 06854, USA."


1 Give us some basic information
2 Complete your registration
Map
Satellite
Google

Property name (English): My Sample Hotel
Country: United States
State / Province: Connecticut
County: Select County
City: Norwalk (CT)
ZIP / Postal code: 06854
Street address: 800 Connecticut Avenue
Main phone number: United States (+1)
3473221265

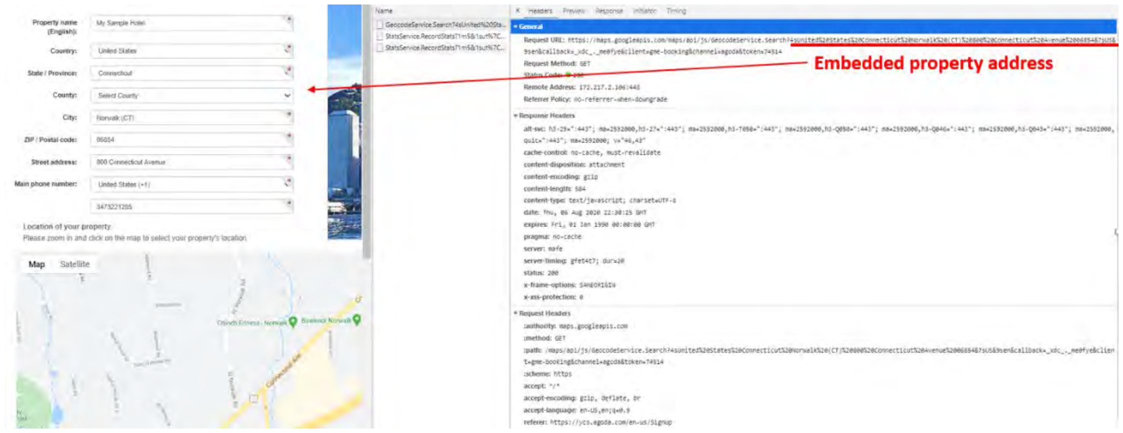
Location of your property
Please zoom in and click on the map to select your property's location

Name
GeocodeService.Search?4sUnited%20Sta...
StatsService.RecordStats?1m5&1sut%7C...
StatsService.RecordStats?1m5&1sut%7C...

Headers Preview Response Initiator Timing

```
▾/**/_xdc_._meofye && _xdc_._meofye( {,…} )
 ▾results: [{address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…],…}]
  ▾0: {address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…],…}
   ▾address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…]
    ▸0: {long_name: "800", short_name: "800", types: ["street_number"]}
    ▸1: {long_name: "Connecticut Avenue", short_name: "Connecticut Ave", types: ["route"]}
    ▸2: {long_name: "Norwalk", short_name: "Norwalk", types: ["locality", "political"]}
    ▸3: {long_name: "Fairfield County", short_name: "Fairfield County",…}
    ▸4: {long_name: "Connecticut", short_name: "CT", types: ["administrative_area_level_1", "political"]}
    ▸5: {long_name: "United States", short_name: "US", types: ["country", "political"]}
    ▸6: {long_name: "06854", short_name: "06854", types: ["postal_code"]}
    formatted_address: "800 Connecticut Ave, Norwalk, CT 06854, USA"
   ▾geometry: {bounds: {northeast: {lat: 41.0940489, lng: -73.45357469999999},…},…}
    ▸bounds: {northeast: {lat: 41.0940489, lng: -73.45357469999999},…}
    ▾location: {lat: 41.0930242, lng: -73.4539645}
      lat: 41.0930242
      lng: -73.4539645
     location_type: "ROOFTOP"
    ▸viewport: {northeast: {lat: 41.0945039302915, lng: -73.4520920197085},…}
    place_id: "ChIJVYODERge6IkR0r1TO2mCAXk"
   ▸types: ["premise"]
  status: "OK"
```

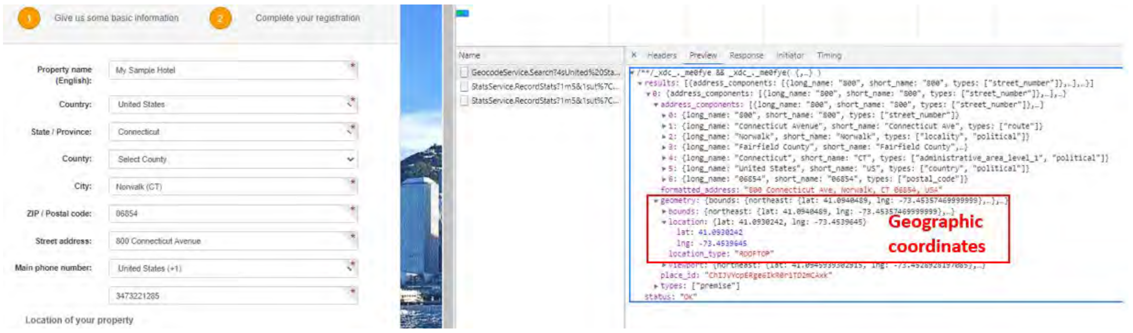
244. On information and belief, the YCS Platform's web components and symbolic names (e.g., JSON data) described above are located in a registry such as a database and/or a CDN. The registry for the YCS Platform also includes addresses for the web services where the input symbolic names and output symbolic names can be sent to and received from. For example, the screenshot below shows a request URL including an address of the Google Maps web service. In particular, the user-input property address "800 Connecticut Avenue. Norwalk, CT 06854" is embedded in the request URL and sent to the Google Maps API via an HTTP GET request for Geocoding service. The Geocoding service in the Google Maps API then converts the property address to geographic coordinates and sends the geographic coordinates with other property information in the format of JSON data to the Agoda website. The geographic coordinates and the user input address are stored in Booking's database for further use.



Property name (English):	My Sample Hotel
Country:	United States
State / Province:	Connecticut
County:	Select County
City:	Norwalk (CT)
ZIP / Postal code:	06854
Street address:	800 Connecticut Avenue
Main phone number:	United States (+1)
	3473221285

Location of your property
Please zoom in and click on the map to select your property's location

```
Name
GeocodeService.Search?4sUnited%20Sta...
StatsService.RecordStats?1m5&1sut%7C...
StatsService.RecordStats?1m5&1sut%7C...

× Headers Preview Response Initiator Timing
▾ General
  Request URL: https://maps.googleapis.com/maps/api/js/GeocodeService.Search?4sUnited%20States%20Connecticut%20Norwalk%20(CT)%20800%20Connecticut%20Avenue%2006854&7sUS&
  9sen&callback=_xdc_._me0fye&client=gme-booking&channel=agoda&token=74514
  Request Method: GET
  Status Code: 200
  Remote Address: 172.217.2.106:443
  Referrer Policy: no-referrer-when-downgrade
▾ Response Headers
  alt-svc: h3-29=":443"; ma=2592000,h3-27=":443"; ma=2592000,h3-T050=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,
  quic=":443"; ma=2592000; v="46,43"
  cache-control: no-cache, must-revalidate
  content-disposition: attachment
  content-encoding: gzip
  content-length: 584
  content-type: text/javascript; charset=UTF-8
  date: Thu, 06 Aug 2020 22:30:25 GMT
  expires: Fri, 01 Jan 1990 00:00:00 GMT
  pragma: no-cache
  server: mafe
  server-timing: gfet4t7; dur=20
  status: 200
  x-frame-options: SAMEORIGIN
  x-xss-protection: 0
▾ Request Headers
  :authority: maps.googleapis.com
  :method: GET
  :path: /maps/api/js/GeocodeService.Search?4sUnited%20States%20Connecticut%20Norwalk%20(CT)%20800%20Connecticut%20Avenue%2006854&7sUS&9sen&callback=_xdc_._me0fye&clien
  t=gme-booking&channel=agoda&token=74514
  :scheme: https
  accept: */*
  accept-encoding: gzip, deflate, br
  accept-language: en-US,en;q=0.9
  referer: https://ycs.agoda.com/en-us/Signup
```



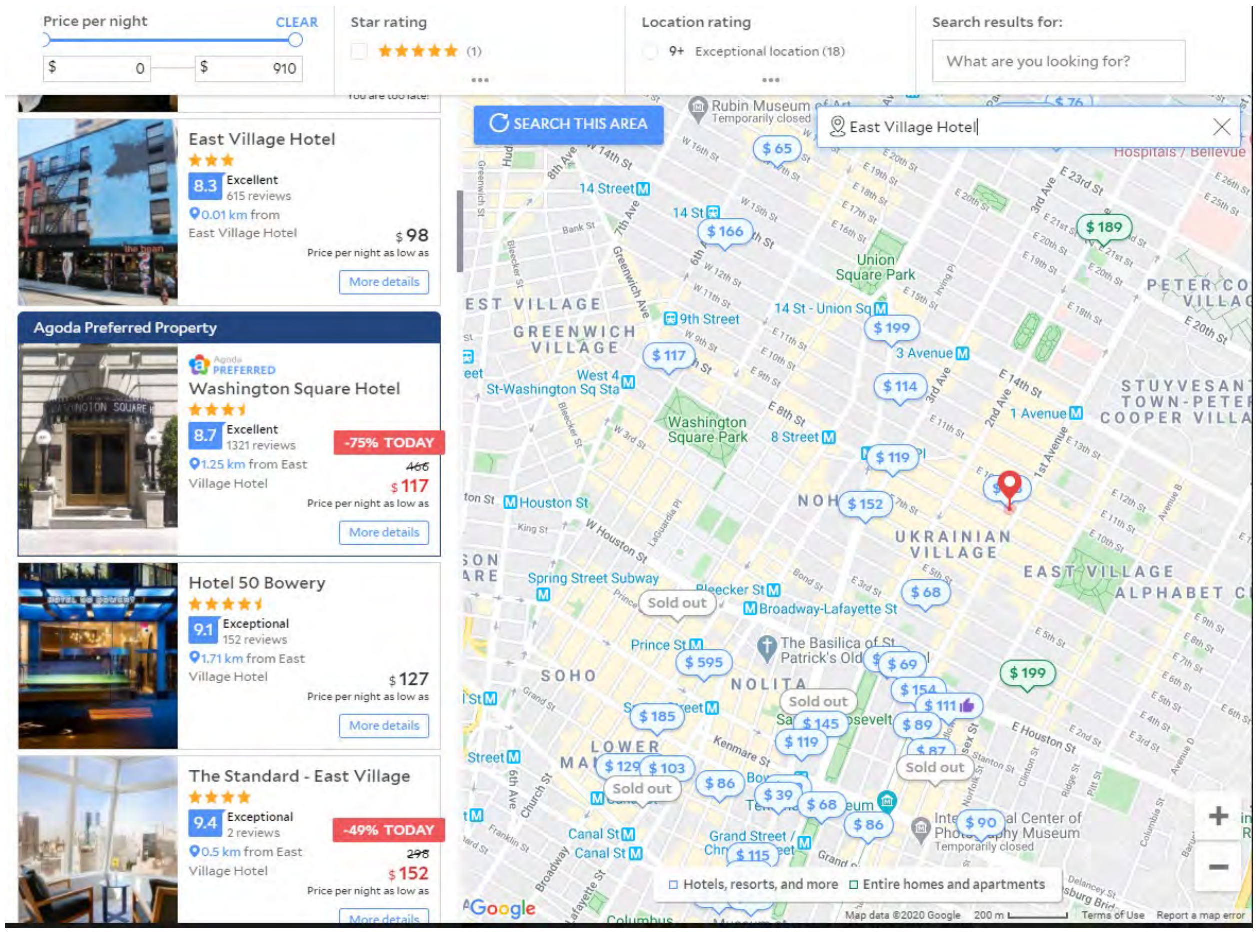
1 Give us some basic information
2 Complete your registration

Property name (English):	My Sample Hotel
Country:	United States
State / Province:	Connecticut
County:	Select County
City:	Norwalk (CT)
ZIP / Postal code:	06854
Street address:	800 Connecticut Avenue
Main phone number:	United States (+1)
	3473221285

Location of your property

```
Name
GeocodeService.Search?4sUnited%20Sta...
StatsService.RecordStats?1m5&1sut%7C...
StatsService.RecordStats?1m5&1sut%7C...

× Headers Preview Response Initiator Timing
▾ /**/_xdc_._me0fye && _xdc_._me0fye( {,…} )
  ▾ results: [{address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…],…}]
    ▾ 0: {address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…],…}
      ▾ address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…]
        ▸ 0: {long_name: "800", short_name: "800", types: ["street_number"]}
        ▸ 1: {long_name: "Connecticut Avenue", short_name: "Connecticut Ave", types: ["route"]}
        ▸ 2: {long_name: "Norwalk", short_name: "Norwalk", types: ["locality", "political"]}
        ▸ 3: {long_name: "Fairfield County", short_name: "Fairfield County",…}
        ▸ 4: {long_name: "Connecticut", short_name: "CT", types: ["administrative_area_level_1", "political"]}
        ▸ 5: {long_name: "United States", short_name: "US", types: ["country", "political"]}
        ▸ 6: {long_name: "06854", short_name: "06854", types: ["postal_code"]}
        formatted_address: "800 Connecticut Ave, Norwalk, CT 06854, USA"
      ▾ geometry: {bounds: {northeast: {lat: 41.0940489, lng: -73.45357469999999},…},…}
        ▸ bounds: {northeast: {lat: 41.0940489, lng: -73.45357469999999},…}
        ▾ location: {lat: 41.0930242, lng: -73.4539645}
            lat: 41.0930242
            lng: -73.4539645
          location_type: "ROOFTOP"
        ▸ viewport: {northeast: {lat: 41.0945939302915, lng: -73.4526928197085},…}
        place_id: "ChIJVVcpERge6IkR0r1TD2mCAxk"
      ▸ types: ["premise"]
  status: "OK"
```

245. The defined UI objects described above are either selected by a user of an authoring tool or automatically selected by the YCS Platform as preferred UI objects. For example, when the Google Maps API is embedded on a Priceline or Agoda property listing, UI objects for web components such as zoom-in and zoom-out buttons, a search field, a search button, a slide bar for "Price per night," a check box for "Star rating," and a radio button for "Location rating" are automatically selected by the YCS Platform as preferred UI objects.



Price per night CLEAR
$ 0 — $ 910

Star rating
★★★★★ (1)

Location rating
9+ Exceptional location (18)

Search results for:
What are you looking for?

East Village Hotel
★★★
8.3 Excellent
615 reviews
0.01 km from East Village Hotel
$ 98
Price per night as low as
More details

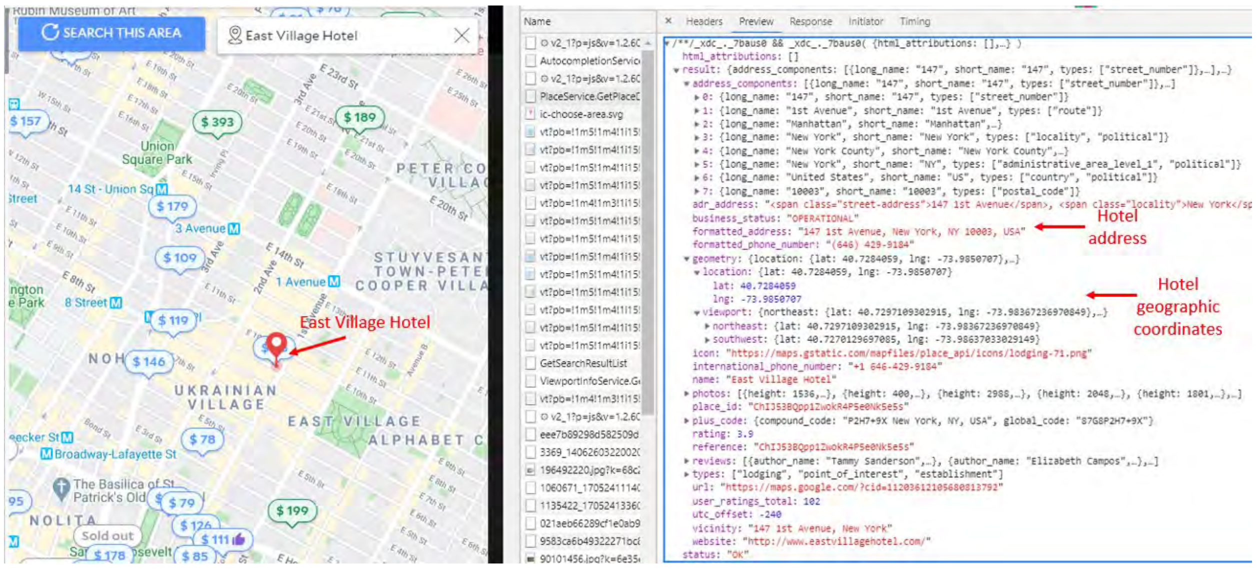
Agoda Preferred Property
Agoda PREFERRED
Washington Square Hotel
★★★½
8.7 Excellent
1321 reviews
-75% TODAY
1.25 km from East Village Hotel
~~466~~
$ 117
Price per night as low as
More details

Hotel 50 Bowery
★★★★½
9.1 Exceptional
152 reviews
1.71 km from East Village Hotel
$ 127
Price per night as low as
More details

The Standard - East Village
★★★★
9.4 Exceptional
2 reviews
-49% TODAY
0.5 km from East Village Hotel
~~298~~
$ 152
Price per night as low as
More details

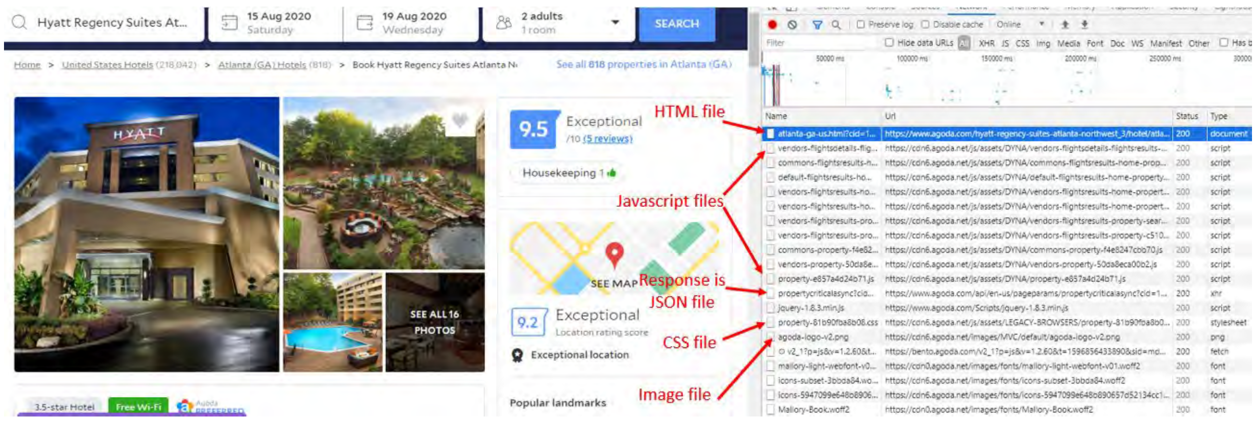
246. When a UI object is so defined, a symbolic name (e.g., JSON data) from the web component corresponding to the defined UI object is selected and associated with the defined UI object. The symbolic name has an associated data format class type corresponding to a subclass of UI objects that support the data format type of the symbolic name, and has the preferred UI object. The example below demonstrates a user searching for "East Village Hotel" on the Google Maps API. When the user types "East Village Hotel" into the search field of the Google Maps API, the Google Maps web service responds by sending a JSON dataset to the YCS Platform. In particular, the symbolic names "address_components" and "formatted_address" are associated with the hotel address, and the symbolic name "geometry" is associated with the geographic coordinates of the hotel. This information is further used to display the property's location on the Google Maps API. In the example below, a pin object associated with the address and geographic coordinates of the property is dropped on the map to indicate the location of the "East Village

Hotel." The pin object is a preferred UI object indicating the target hotel, distinguishable from bubbles indicating locations and prices for other nearby hotels.



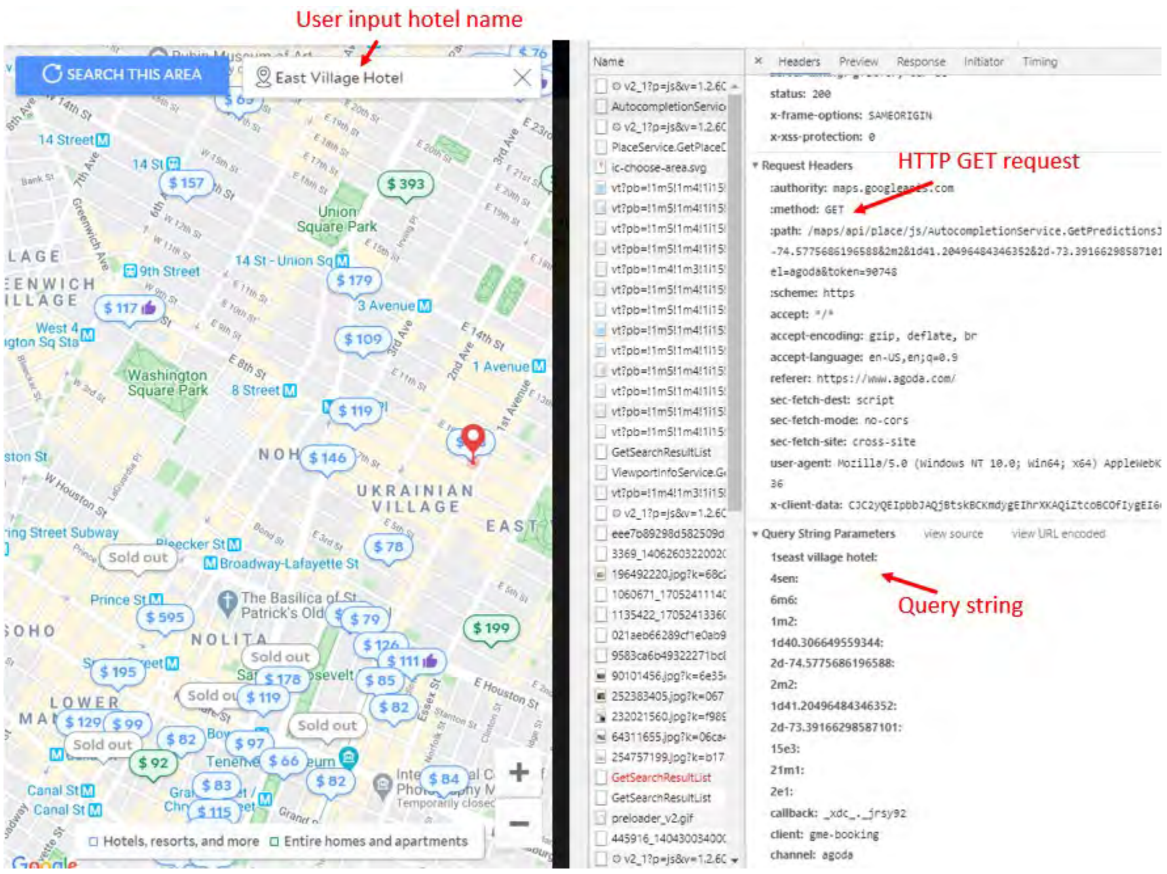
```
▾/**/_xdc_._7baus0 && _xdc_._7baus0( {html_attributions: [],…} )
   html_attributions: []
 ▾result: {address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…],…}
  ▾address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…]
   ▸0: {long_name: "147", short_name: "147", types: ["street_number"]}
   ▸1: {long_name: "1st Avenue", short_name: "1st Avenue", types: ["route"]}
   ▸2: {long_name: "Manhattan", short_name: "Manhattan",…}
   ▸3: {long_name: "New York", short_name: "New York", types: ["locality", "political"]}
   ▸4: {long_name: "New York County", short_name: "New York County",…}
   ▸5: {long_name: "New York", short_name: "NY", types: ["administrative_area_level_1", "political"]}
   ▸6: {long_name: "United States", short_name: "US", types: ["country", "political"]}
   ▸7: {long_name: "10003", short_name: "10003", types: ["postal_code"]}
   adr_address: "<span class="street-address">147 1st Avenue</span>, <span class="locality">New York</sp
   business_status: "OPERATIONAL"
   formatted_address: "147 1st Avenue, New York, NY 10003, USA"
   formatted_phone_number: "(646) 429-9184"
  ▾geometry: {location: {lat: 40.7284059, lng: -73.9850707},…}
   ▾location: {lat: 40.7284059, lng: -73.9850707}
      lat: 40.7284059
      lng: -73.9850707
   ▾viewport: {northeast: {lat: 40.7297109302915, lng: -73.98367236970849},…}
    ▸northeast: {lat: 40.7297109302915, lng: -73.98367236970849}
    ▸southwest: {lat: 40.7270129697085, lng: -73.98637033029149}
   icon: "https://maps.gstatic.com/mapfiles/place_api/icons/lodging-71.png"
   international_phone_number: "+1 646-429-9184"
   name: "East Village Hotel"
  ▸photos: [{height: 1536,…}, {height: 400,…}, {height: 2988,…}, {height: 2048,…}, {height: 1801,…},…]
   place_id: "ChIJ53BQpp1ZwokR4P5e0Nk5e5s"
  ▸plus_code: {compound_code: "P2H7+9X New York, NY, USA", global_code: "87G8P2H7+9X"}
   rating: 3.9
   reference: "ChIJ53BQpp1ZwokR4P5e0Nk5e5s"
  ▸reviews: [{author_name: "Tammy Sanderson",…}, {author_name: "Elizabeth Campos",…},…]
  ▸types: ["lodging", "point_of_interest", "establishment"]
   url: "https://maps.google.com/?cid=11203612105680813792"
   user_ratings_total: 102
   utc_offset: -240
   vicinity: "147 1st Avenue, New York"
   website: "http://www.eastvillagehotel.com/"
 status: "OK"
```

247. The YCS Platform then produces a device-independent application that includes the JSON data, as well as standard HTML, CSS and Javascript code. The application is executed on a device together with a device-dependent player, such as a device-dependent code for a browser engine for a specific browser, or an operating system or application for a particular kind of device. *See, e.g., Shopify Inc. v. Express Mobile, Inc.*, Case No. 1:19-cv-00439-RGA, D. Del., D.I. 137. In the example below, the YCS Platform converts HTML, CSS, Javascript, JSON, and other files into an active property listing website for "Hyatt Regency Suites Atlanta."



Name	Url	Status	Type
atlanta-ga-us.html?cid=1...	https://www.agoda.com/hyatt-regency-suites-atlanta-northwest_3/hotel/atla...	200	document
vendors-flightsdetails-flig...	https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsdetails-flightsresults-...	200	script
commons-flightsresults-h...	https://cdn6.agoda.net/js/assets/DYNA/commons-flightsresults-home-prop...	200	script
default-flightsresults-ho...	https://cdn6.agoda.net/js/assets/DYNA/default-flightsresults-home-property...	200	script
vendors-flightsresults-ho...	https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsresults-home-propert...	200	script
vendors-flightsresults-ho...	https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsresults-home-propert...	200	script
vendors-flightsresults-pro...	https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsresults-property-sear...	200	script
vendors-flightsresults-pro...	https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsresults-property-c510...	200	script
commons-property-f4e82...	https://cdn6.agoda.net/js/assets/DYNA/commons-property-f4e8247cbb70.js	200	script
vendors-property-50da8e...	https://cdn6.agoda.net/js/assets/DYNA/vendors-property-50da8eca00b2.js	200	script
property-e857a4d24b71.js	https://cdn6.agoda.net/js/assets/DYNA/property-e857a4d24b71.js	200	script
propertycriticalasync?cid...	https://www.agoda.com/api/en-us/pageparams/propertycriticalasync?cid=1...	200	xhr
jquery-1.8.3.min.js	https://www.agoda.com/Scripts/jquery-1.8.3.min.js	200	script
property-81b90fba8b08.css	https://cdn6.agoda.net/js/assets/LEGACY-BROWSERS/property-81b90fba8b0...	200	stylesheet
agoda-logo-v2.png	https://cdn6.agoda.net/images/MVC/default/agoda-logo-v2.png	200	png
v2_1?p=js&v=1.2.60&t...	https://bento.agoda.com/v2_1?p=js&v=1.2.60&t=1596856433890&sid=md...	200	fetch
mallory-light-webfont-v0...	https://cdn0.agoda.net/images/fonts/mallory-light-webfont-v01.woff2	200	font
icons-subset-3bbda84.wo...	https://cdn6.agoda.net/images/fonts/icons-subset-3bbda84.woff2	200	font
icons-5947099e648b8906...	https://cdn6.agoda.net/images/fonts/icons-5947099e648b890657d52134cc1...	200	font
Mallory-Book.woff2	https://cdn0.agoda.net/images/fonts/Mallory-Book.woff2	200	font

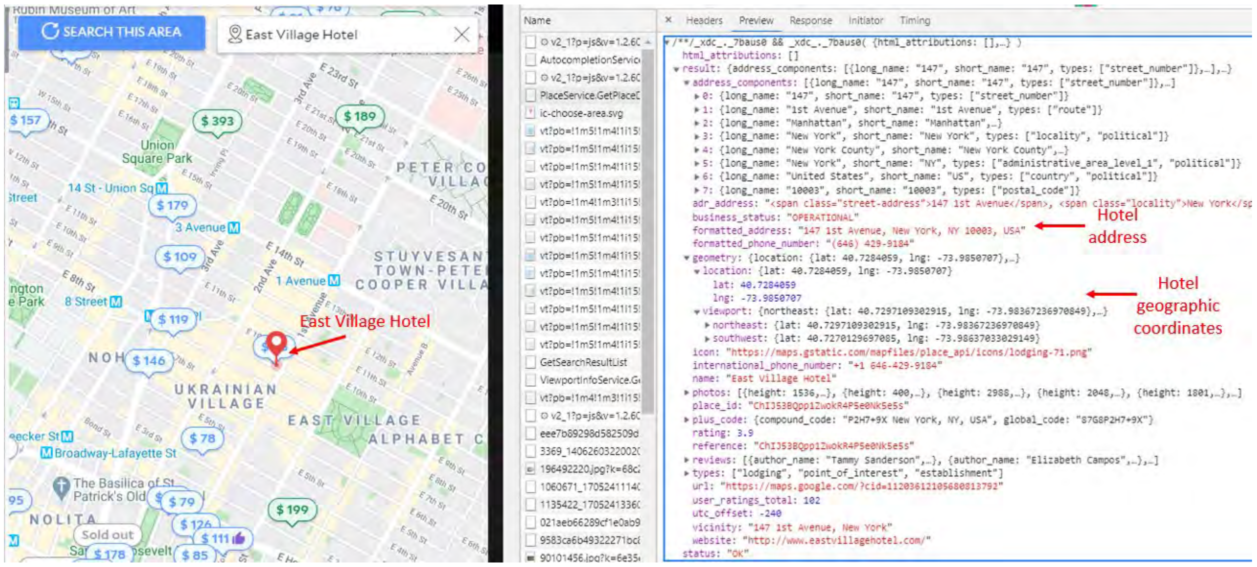
248. When the application and player are executed on the device, an input value can be provided by a user. The device then provides the input value and a corresponding input symbolic name to the web service, which in turn generates an output value and a corresponding symbolic name for the output. By way of example, when the Google Maps API accepts an input value (e.g., an input hotel name, a zoom-in click, or a click on the "Explore" tab) from a user, a corresponding input symbolic name (e.g., a query string or JSON data for the input) is transmitted to the Google Maps web service through an HTTP request protocol, such as a GET method call. The example below demonstrates a user searching for "East Village Hotel" on the Google Maps API on an Agoda website. When the user types "East Village Hotel," a query string with parameters including the symbolic name "1s" and the associated value "East Village Hotel" is sent to the Google Maps web service to initiate an autocompletion service with an HTTP GET request.



```
Name
v2_1?p=js&v=1.2.6
AutocompletionServic
v2_1?p=js&v=1.2.6
PlaceService.GetPlace
ic-choose-area.svg
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m4!1m3!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
GetSearchResultList
ViewportInfoService.G
vt?pb=!1m4!1m3!1i15
v2_1?p=js&v=1.2.6
eee7b89298d582509d
3369_1406260322002
196492220.jpg?k=68c
1060671_1705241114
1135422_1705241336
021aeb66289cf1e0ab9
9583ca6b49322271bc
90101456.jpg?k=6e35
252383405.jpg?k=067
232021560.jpg?k=f98
64311655.jpg?k=06ca
254757199.jpg?k=b17
GetSearchResultList
GetSearchResultList
preloader_v2.gif
445916_14043003400
v2_1?p=js&v=1.2.6
```

```
status: 200
x-frame-options: SAMEORIGIN
x-xss-protection: 0
▾ Request Headers
:authority: maps.googleapis.com
:method: GET
:path: /maps/api/place/js/AutocompletionService.GetPredictionsJ
-74.5775686196588&2m2&1d41.20496484346352&2d-73.3916629858710
el=agoda&token=90748
:scheme: https
accept: */*
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
referer: https://www.agoda.com/
sec-fetch-dest: script
sec-fetch-mode: no-cors
sec-fetch-site: cross-site
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebK
36
x-client-data: CJC2yQEIpbbJAQjBtskBCKmdygEIhrXKAQiZtcoBCOfIygEI6
▾ Query String Parameters    view source    view URL encoded
1seast village hotel:
4sen:
6m6:
1m2:
1d40.306649559344:
2d-74.5775686196588:
2m2:
1d41.20496484346352:
2d-73.39166298587101:
15e3:
21m1:
2e1:
callback: _xdc_._jrsy92
client: gme-booking
channel: agoda
```

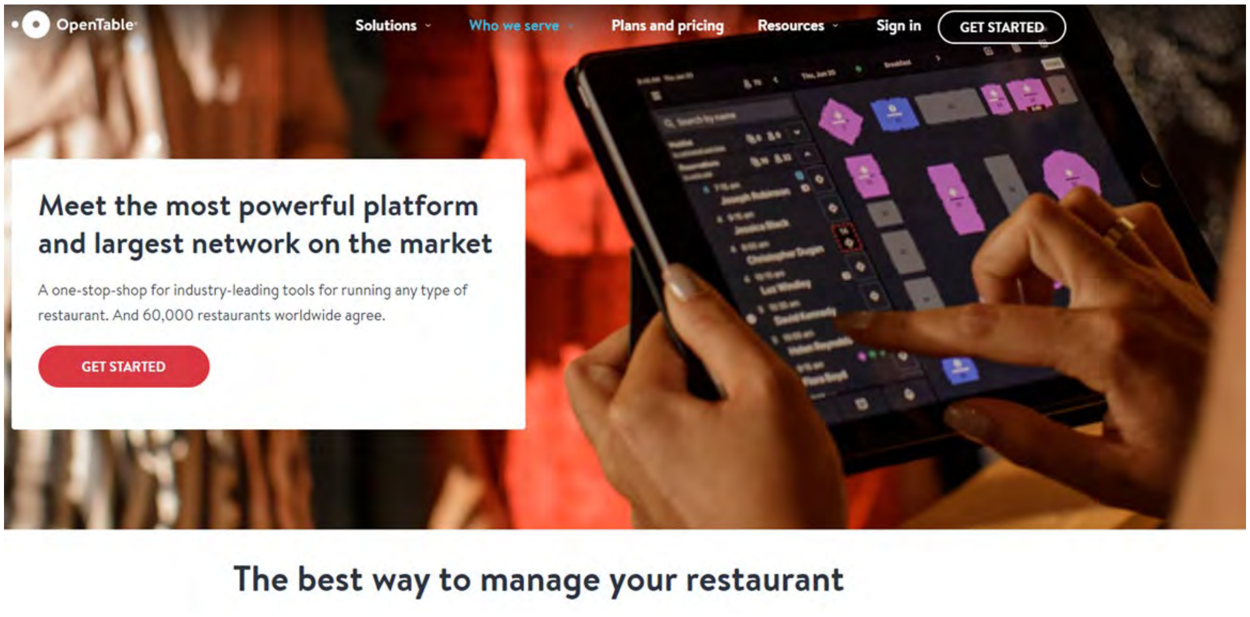
249. The input symbolic name is utilized by the Google Map web service to generate an output value with an associated output symbolic name (e.g., JSON data for the output). The player then provides instructions to the display of the device to present the output value in the API. For example, when a user searches "East Village Hotel" and selects an autocomplete suggestion from the Google Maps API, the Google Maps web service generates and transmits a corresponding hotel address and geographic coordinates in the format of JSON data to the web browser and displays a pin indicating the location of the hotel on the Google Maps API.



```
/**/_xdc_._7baus0 && _xdc_._7baus0( {html_attributions: [],…} )
  html_attributions: []
  result: {address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…],…}
    address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…]
      0: {long_name: "147", short_name: "147", types: ["street_number"]}
      1: {long_name: "1st Avenue", short_name: "1st Avenue", types: ["route"]}
      2: {long_name: "Manhattan", short_name: "Manhattan",…}
      3: {long_name: "New York", short_name: "New York", types: ["locality", "political"]}
      4: {long_name: "New York County", short_name: "New York County",…}
      5: {long_name: "New York", short_name: "NY", types: ["administrative_area_level_1", "political"]}
      6: {long_name: "United States", short_name: "US", types: ["country", "political"]}
      7: {long_name: "10003", short_name: "10003", types: ["postal_code"]}
    adr_address: "<span class="street-address">147 1st Avenue</span>, <span class="locality">New York</sp
    business_status: "OPERATIONAL"
    formatted_address: "147 1st Avenue, New York, NY 10003, USA"
    formatted_phone_number: "(646) 429-9184"
    geometry: {location: {lat: 40.7284059, lng: -73.9850707},…}
      location: {lat: 40.7284059, lng: -73.9850707}
        lat: 40.7284059
        lng: -73.9850707
      viewport: {northeast: {lat: 40.7297109302915, lng: -73.98367236970849},…}
        northeast: {lat: 40.7297109302915, lng: -73.98367236970849}
        southwest: {lat: 40.7270129697085, lng: -73.98637033029149}
    icon: "https://maps.gstatic.com/mapfiles/place_api/icons/lodging-71.png"
    international_phone_number: "+1 646-429-9184"
    name: "East Village Hotel"
    photos: [{height: 1536,…}, {height: 400,…}, {height: 2988,…}, {height: 2048,…}, {height: 1801,…},…]
    place_id: "ChIJ53BQpp1ZwokR4P5e0Nk5e5s"
    plus_code: {compound_code: "P2H7+9X New York, NY, USA", global_code: "87G8P2H7+9X"}
    rating: 3.9
    reference: "ChIJ53BQpp1ZwokR4P5e0Nk5e5s"
    reviews: [{author_name: "Tammy Sanderson",…}, {author_name: "Elizabeth Campos",…},…]
    types: ["lodging", "point_of_interest", "establishment"]
    url: "https://maps.google.com/?cid=11203612105680813792"
    user_ratings_total: 102
    utc_offset: -240
    vicinity: "147 1st Avenue, New York"
    website: "http://www.eastvillagehotel.com/"
  status: "OK"
```

The OpenTable Platform:

250. OpenTable infringes at least claim 15 of the '287 patent through a combination of features in the OpenTable Platform that collectively practice each limitation of claim 15. By way of example, OpenTable provides the OpenTable Platform, a browser-based platform for, *inter alia*, creating restaurant profile websites that can be displayed on a display of a device.

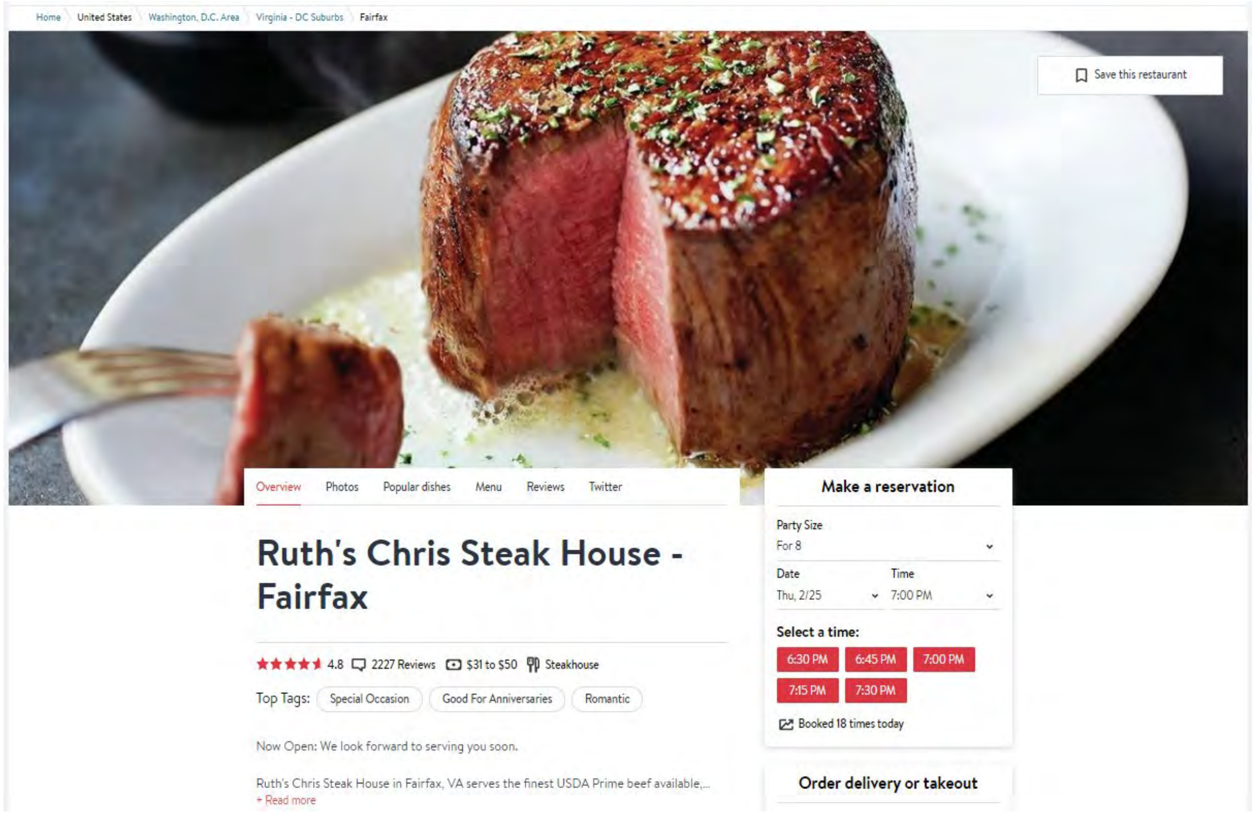


Meet the most powerful platform and largest network on the market

A one-stop-shop for industry-leading tools for running any type of restaurant. And 60,000 restaurants worldwide agree.

The best way to manage your restaurant

https://restaurant.opentable.com/why-opentable/


Home > United States > Washington, D.C. Area > Virginia - DC Suburbs > Fairfax
Save this restaurant
Overview
Photos
Popular dishes
Menu
Reviews
Twitter

Ruth's Chris Steak House - Fairfax

★★★★★ 4.8 2227 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Romantic

Now Open: We look forward to serving you soon.

Ruth's Chris Steak House in Fairfax, VA serves the finest USDA Prime beef available,...
+ Read more

Make a reservation

Party Size: For 8

Date: Thu, 2/25

Time: 7:00 PM

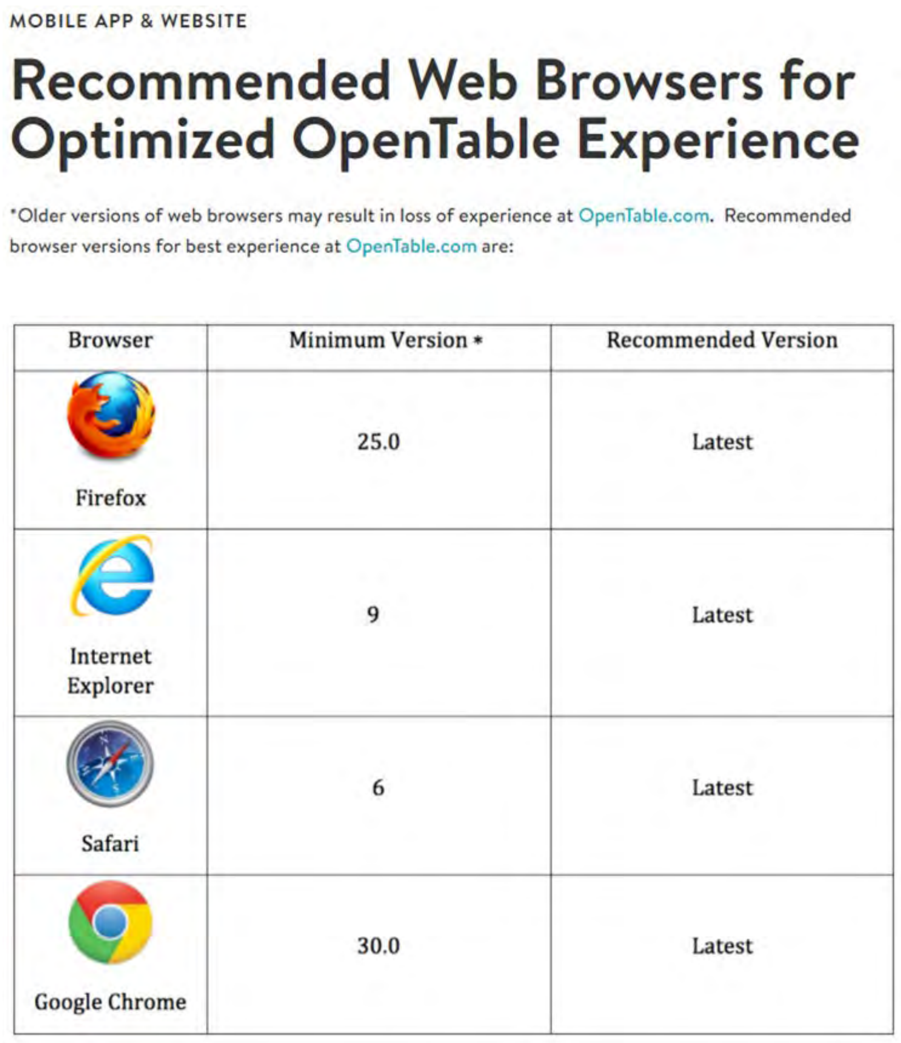
Select a time:

Booked 18 times today

Order delivery or takeout

https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8eae-7ada9d88b68e&corrid=dfc6cee2-c8f8-4054-8eae-7ada9d88b68e&avt=eyJ2IjoyLCJtIjoxLCJwIjowLCJzIjowLCJuIjowfQ

251. The OpenTable platform displays restaurant profile on a device having a player, such as a device-dependent code for a browser engine for a browser, or for an operating system or application of a device. See, e.g., Shopify Inc. v. Express Mobile, Inc., Case No. 1:19-cv-00439-RGA, D. Del., D.I. 137. For example, the OpenTable platform can display a restaurant profile on a Google Chrome, Mozilla Firefox, Apple Safari, or Microsoft Internet Explorer using the player.



MOBILE APP & WEBSITE

Recommended Web Browsers for Optimized OpenTable Experience

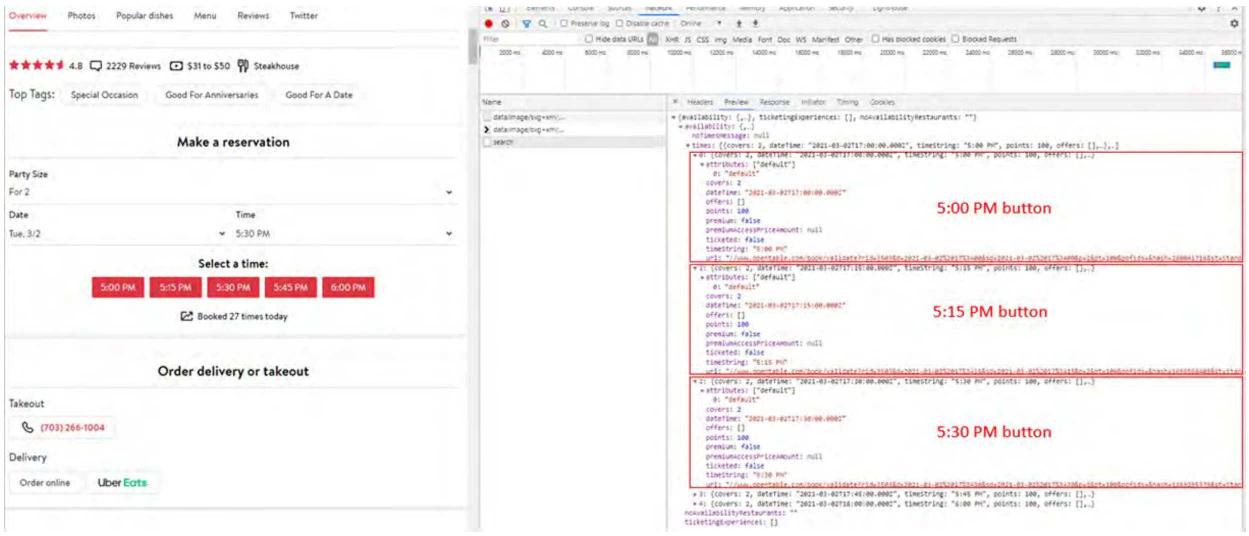
*Older versions of web browsers may result in loss of experience at OpenTable.com. Recommended browser versions for best experience at OpenTable.com are:

Browser	Minimum Version *	Recommended Version
Firefox	25.0	Latest
Internet Explorer	9	Latest
Safari	6	Latest
Google Chrome	30.0	Latest

https://help.opentable.com/s/article/Recommended-Web-Browsers-for-Optimized-OpenTable-Experience-1505260708133?language=en_US

252. The OpenTable platform defines a UI object corresponding to a web component in the registry for an input or output of a web service, for presentation on the display. By way of

example, the authoring tool defines UI objects for web components such as a button to represent the available time slot.


Overview Photos Popular dishes Menu Reviews Twitter

4.8 2229 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Good For A Date

Make a reservation

Party Size
For 2

Date Tue, 3/2 Time 5:30 PM

Select a time:

5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM

Booked 27 times today

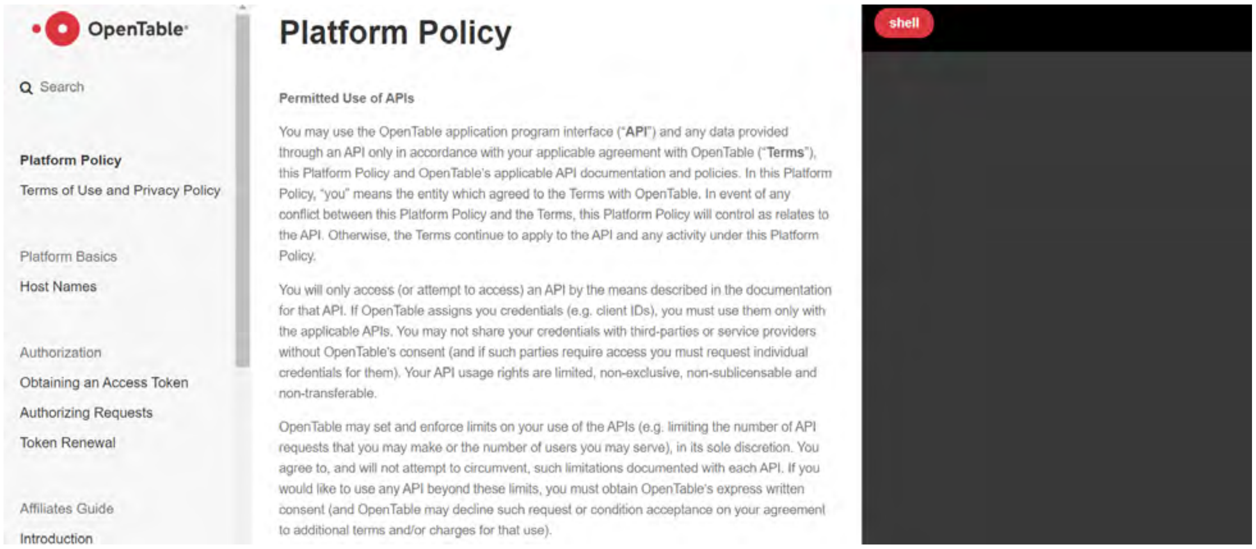
Order delivery or takeout

Takeout
(703) 266-1004

Delivery
Order online Uber Eats

```
▾ {availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾ availability: {,…}
      noTimesMessage: null
    ▾ times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
      ▾ 0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
        ▾ attributes: ["default"]
            0: "default"
          covers: 2
          dateTime: "2021-03-02T17:00:00.000Z"
          offers: []
          points: 100
          premium: false
          premiumAccessPriceAmount: null
          ticketed: false
          timeString: "5:00 PM"
          url: "//www.opentable.com/book/validate?[illegible]
      ▾ 1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
        ▾ attributes: ["default"]
            0: "default"
          covers: 2
          dateTime: "2021-03-02T17:15:00.000Z"
          offers: []
          points: 100
          premium: false
          premiumAccessPriceAmount: null
          ticketed: false
          timeString: "5:15 PM"
          url: "//www.opentable.com/book/validate?[illegible]
      ▾ 2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
        ▾ attributes: ["default"]
            0: "default"
          covers: 2
          dateTime: "2021-03-02T17:30:00.000Z"
          offers: []
          points: 100
          premium: false
          premiumAccessPriceAmount: null
          ticketed: false
          timeString: "5:30 PM"
          url: "//www.opentable.com/book/validate?[illegible]
      ▸ 3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
      ▸ 4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
    noAvailabilityRestaurants: ""
    ticketingExperiences: []
```

253. The OpenTable platform provides various web services through the OpenTable application programming interface (API). A client can use the OpenTable API to access the OpenTable platform for data and services, such as make a reservation. In particular, the data is sent and received in JSON format.


OpenTable

Search

Platform Policy
Terms of Use and Privacy Policy

Platform Basics
Host Names

Authorization
Obtaining an Access Token
Authorizing Requests
Token Renewal

Affiliates Guide
Introduction

Platform Policy

Permitted Use of APIs

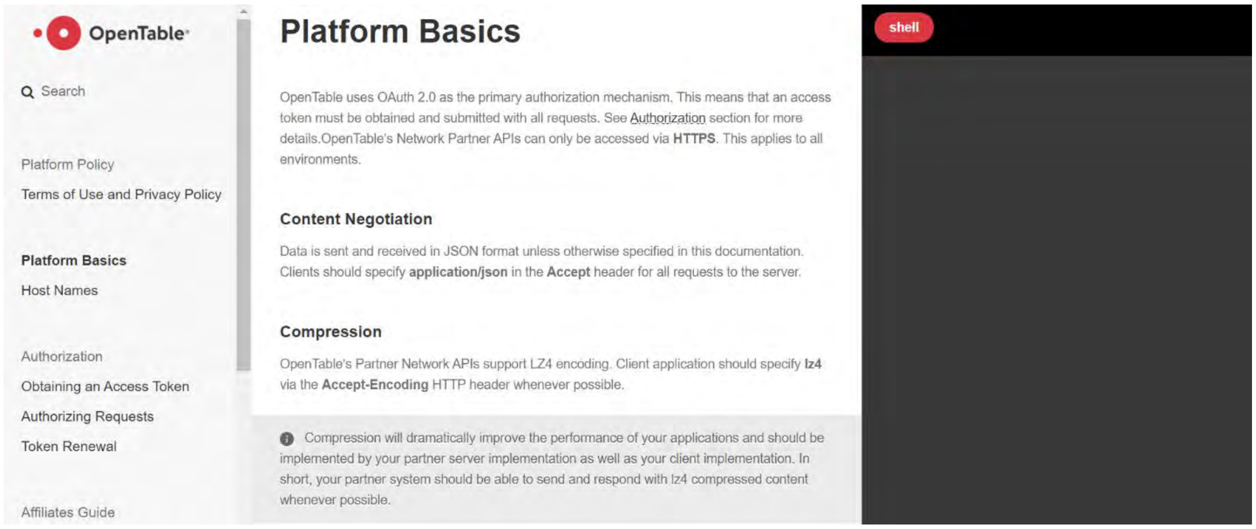
You may use the OpenTable application program interface ("API") and any data provided through an API only in accordance with your applicable agreement with OpenTable ("Terms"), this Platform Policy and OpenTable's applicable API documentation and policies. In this Platform Policy, "you" means the entity which agreed to the Terms with OpenTable. In event of any conflict between this Platform Policy and the Terms, this Platform Policy will control as relates to the API. Otherwise, the Terms continue to apply to the API and any activity under this Platform Policy.

You will only access (or attempt to access) an API by the means described in the documentation for that API. If OpenTable assigns you credentials (e.g. client IDs), you must use them only with the applicable APIs. You may not share your credentials with third-parties or service providers without OpenTable's consent (and if such parties require access you must request individual credentials for them). Your API usage rights are limited, non-exclusive, non-sublicensable and non-transferable.

OpenTable may set and enforce limits on your use of the APIs (e.g. limiting the number of API requests that you may make or the number of users you may serve), in its sole discretion. You agree to, and will not attempt to circumvent, such limitations documented with each API. If you would like to use any API beyond these limits, you must obtain OpenTable's express written consent (and OpenTable may decline such request or condition acceptance on your agreement to additional terms and/or charges for that use).

shell

https://platform.opentable.com/documentation/#platform-policy



Platform Basics

OpenTable uses OAuth 2.0 as the primary authorization mechanism. This means that an access token must be obtained and submitted with all requests. See Authorization section for more details.OpenTable's Network Partner APIs can only be accessed via **HTTPS**. This applies to all environments.

Content Negotiation

Data is sent and received in JSON format unless otherwise specified in this documentation. Clients should specify **application/json** in the **Accept** header for all requests to the server.

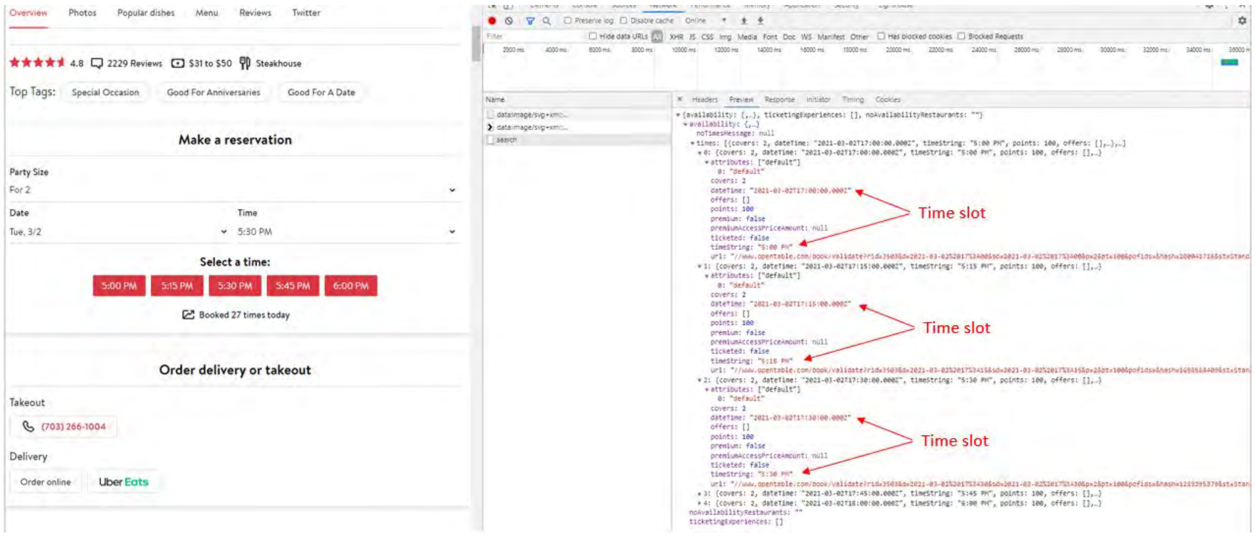
Compression

OpenTable's Partner Network APIs support LZ4 encoding. Client application should specify **lz4** via the **Accept-Encoding** HTTP header whenever possible.

Compression will dramatically improve the performance of your applications and should be implemented by your partner server implementation as well as your client implementation. In short, your partner system should be able to send and respond with lz4 compressed content whenever possible.

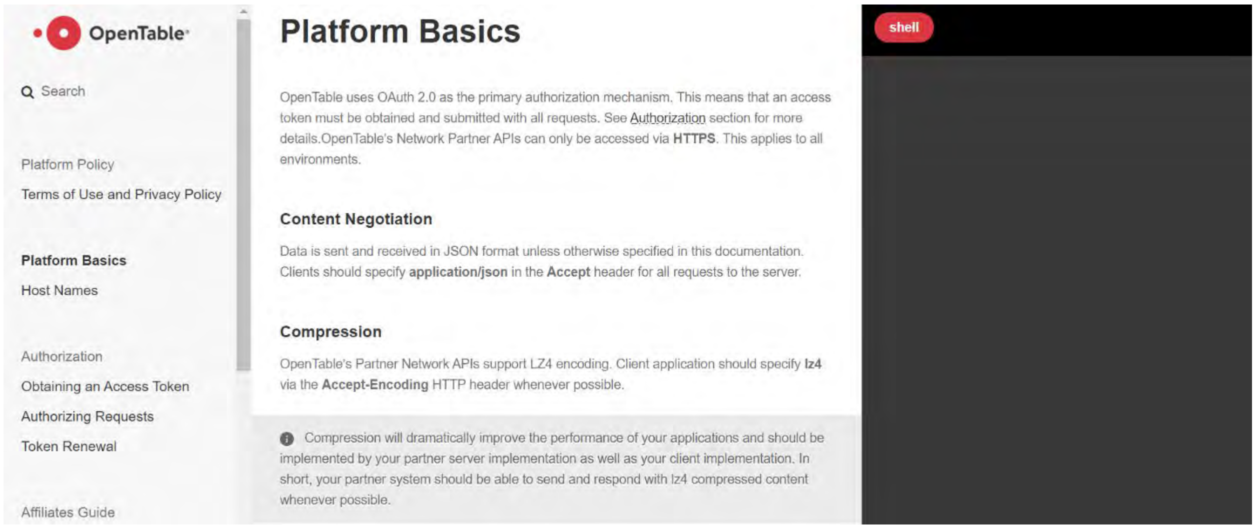
https://platform.opentable.com/documentation/#platform-basics

254. On information and belief, the OpenTable platform utilizes JSON data, which comprise symbolic names, to evoke web components to communicate inputs and outputs for the APIs and other web services. In the example below, output available time slots for the OpenTable's reservation web service are transmitted in JSON data.



```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
    noTimesMessage: null
   ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
    ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
       covers: 2
       dateTime: "2021-03-02T17:00:00.000Z"
       offers: []
       points: 100
       premium: false
       premiumAccessPriceAmount: null
       ticketed: false
       timeString: "5:00 PM"
       url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A00&sd=2021-03-02%2017%3A00&p=2&pt=100&pofids=&hash=2000417168st=Stand
    ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
       covers: 2
       dateTime: "2021-03-02T17:15:00.000Z"
       offers: []
       points: 100
       premium: false
       premiumAccessPriceAmount: null
       ticketed: false
       timeString: "5:15 PM"
       url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A15&sd=2021-03-02%2017%3A15&p=2&pt=100&pofids=&hash=1693584096st=Stan
    ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
       covers: 2
       dateTime: "2021-03-02T17:30:00.000Z"
       offers: []
       points: 100
       premium: false
       premiumAccessPriceAmount: null
       ticketed: false
       timeString: "5:30 PM"
       url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A30&sd=2021-03-02%2017%3A30&p=2&pt=100&pofids=&hash=1232953798st=Stan
    ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
    ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```

255. On information and belief, the registry includes symbolic names in the form of JSON data, to evoke web components for the applications and to communicate inputs and outputs for the web services over the Internet. JSON data comprise key/value pairs that are essentially character strings (i.e., symbolic names) with no persistent address or pointer.



Platform Basics

OpenTable uses OAuth 2.0 as the primary authorization mechanism. This means that an access token must be obtained and submitted with all requests. See Authorization section for more details.OpenTable's Network Partner APIs can only be accessed via **HTTPS**. This applies to all environments.

Content Negotiation

Data is sent and received in JSON format unless otherwise specified in this documentation. Clients should specify **application/json** in the **Accept** header for all requests to the server.

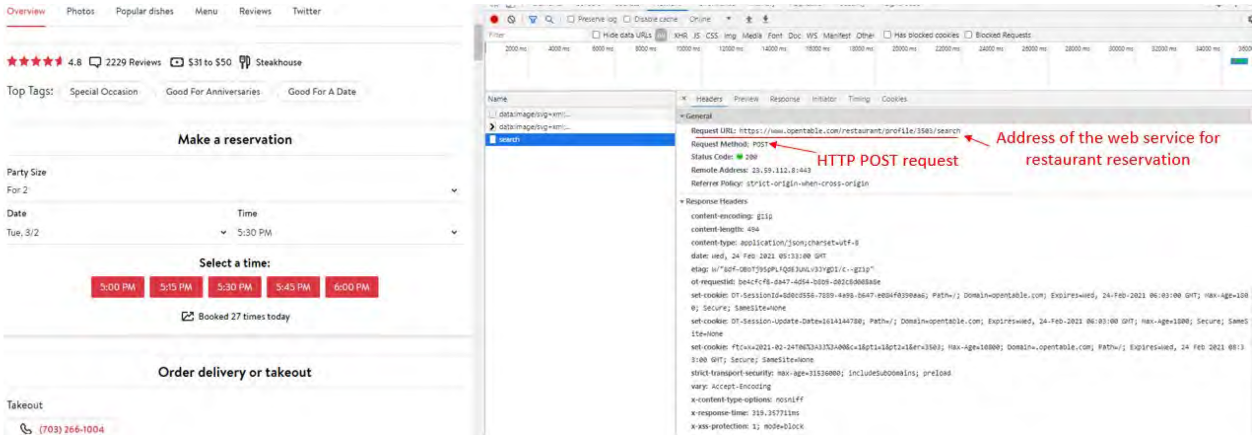
Compression

OpenTable's Partner Network APIs support LZ4 encoding. Client application should specify **lz4** via the **Accept-Encoding** HTTP header whenever possible.

Compression will dramatically improve the performance of your applications and should be implemented by your partner server implementation as well as your client implementation. In short, your partner system should be able to send and respond with lz4 compressed content whenever possible.

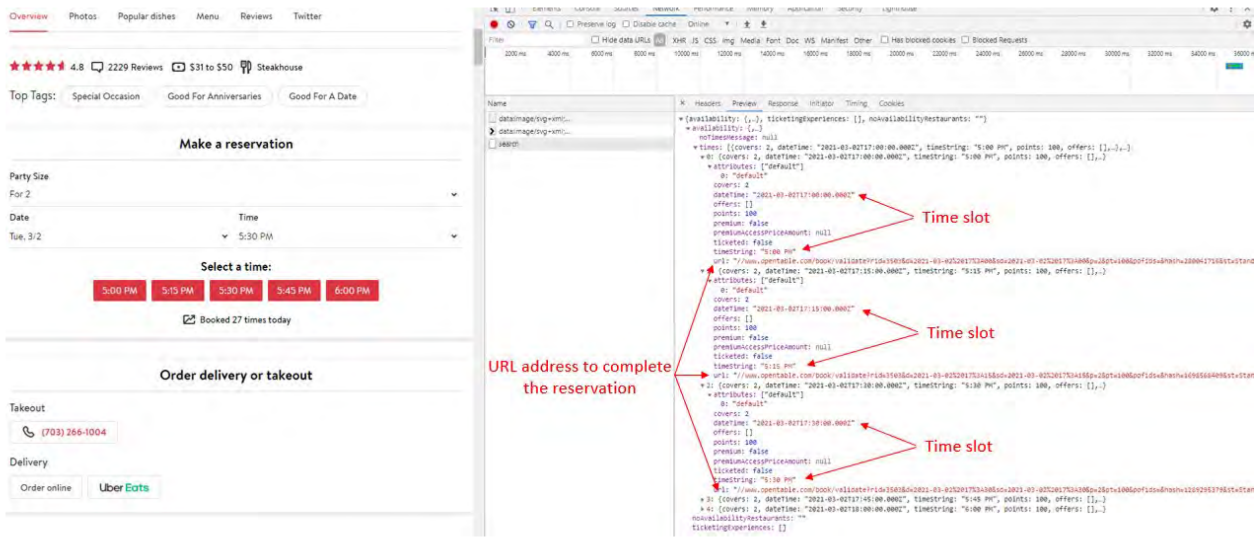
https://platform.opentable.com/documentation/#platform-basics

256. The registry also includes addresses for the web services where the input symbolic names and output symbolic names can be sent to and received from. The exemplary screenshot below shows a request URL including an address of the OpenTable's reservation web service. The OpenTable platform transfers user input settings, e.g., party size, date and time, in JSON format and sends them to the OpenTable's reservation web service via an HTTP POST request. The OpenTable's reservation web service responds with five available time slots for the input date and time in JSON format and the OpenTable platform creates five web components (e.g., button UI objects) to display the five available time slots. Each of the five web components further includes a URL address to complete the reservation in the specified time slot.



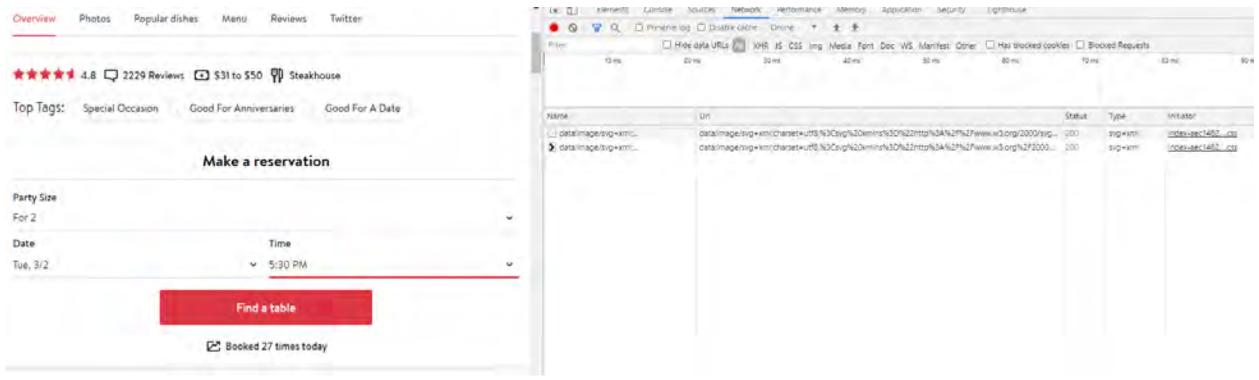
Party Size: For 2
Date: Tue, 3/2
Time: 5:30 PM

```
▾ General
    Request URL: https://www.opentable.com/restaurant/profile/3503/search
    Request Method: POST
    Status Code: 200
    Remote Address: 23.59.112.8:443
    Referrer Policy: strict-origin-when-cross-origin
▾ Response Headers
    content-encoding: gzip
    content-length: 494
    content-type: application/json;charset=utf-8
    date: Wed, 24 Feb 2021 05:33:00 GMT
    etag: W/"8df-OBoTj9SpPLfQdEJUNLv3JYgDI/c--gzip"
    ot-requestid: be4cfcf8-da47-4d54-b8b9-d02c8d008a6e
    set-cookie: OT-SessionId=8d0cd556-7889-4a98-b647-e084f0390aa6; Path=/; Domain=opentable.com; Expires=Wed, 24-Feb-2021 06:03:00 GMT; Max-Age=180
    0; Secure; SameSite=None
    set-cookie: OT-Session-Update-Date=1614144780; Path=/; Domain=opentable.com; Expires=Wed, 24-Feb-2021 06:03:00 GMT; Max-Age=1800; Secure; SameS
    ite=None
    set-cookie: ftc=x=2021-02-24T06%3A33%3A00&c=1&pt1=1&pt2=1&er=3503; Max-Age=10800; Domain=.opentable.com; Path=/; Expires=Wed, 24 Feb 2021 08:3
    3:00 GMT; Secure; SameSite=None
    strict-transport-security: max-age=31536000; includeSubDomains; preload
    vary: Accept-Encoding
    x-content-type-options: nosniff
    x-response-time: 319.357711ms
    x-xss-protection: 1; mode=block
```

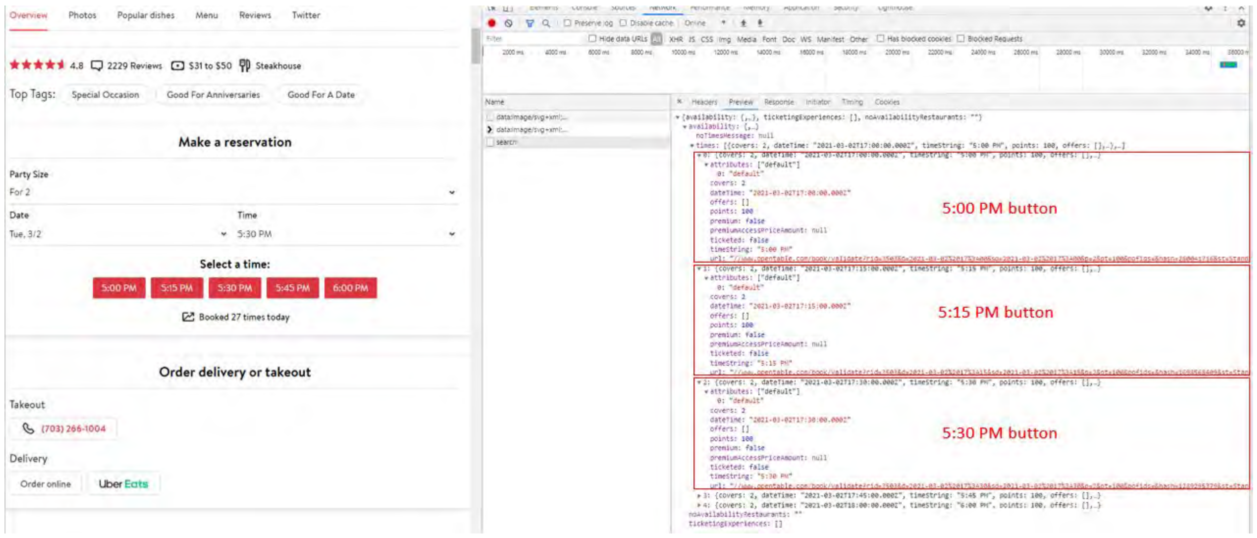


```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
    noTimesMessage: null
   ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
    ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:00:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:00 PM"
      url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A00&sd=2021-03-02%2017%3A00&p=2&pt=100&pofids=&hash=2800417168&st=Stand
    ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:15:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:15 PM"
      url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A15&sd=2021-03-02%2017%3A15&p=2&pt=100&pofids=&hash=1698568409&st=Stan
    ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:30:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:30 PM"
      url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A30&sd=2021-03-02%2017%3A30&p=2&pt=100&pofids=&hash=1289295379&st=Stan
    ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
    ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```

257. Each defined UI object is either selected by a user of an authoring tool or automatically selected by the OpenTable platform as the preferred UI object corresponding to the symbolic name of the web component selected by the user of the authoring tool. For example, the OpenTable platform automatically selects the button as preferred UI objects.



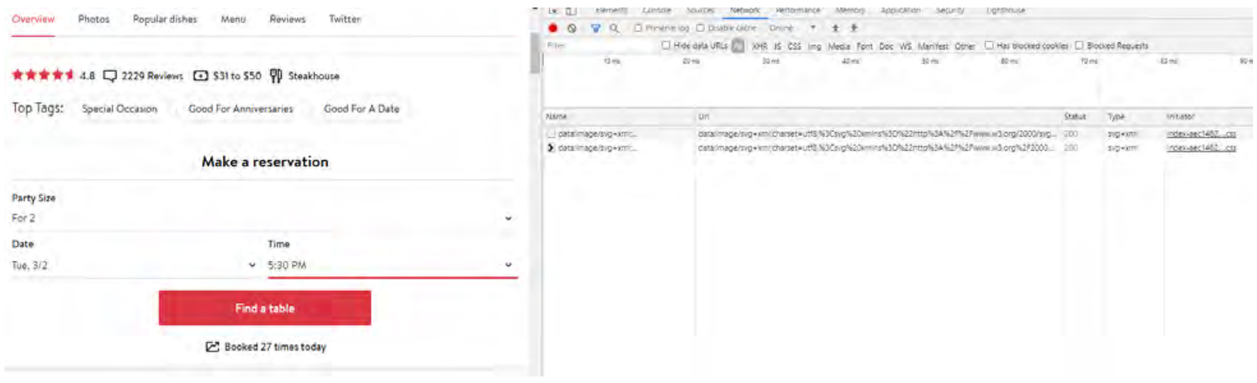
Name	Url	Status	Type	Initiator
data:image/svg+xml;...	data:image/svg+xml;charset=utf8,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org/2000/svg...	200	svg+xml	index-aec1482...css
data:image/svg+xml;...	data:image/svg+xml;charset=utf8,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000...	200	svg+xml	index-aec1482...css

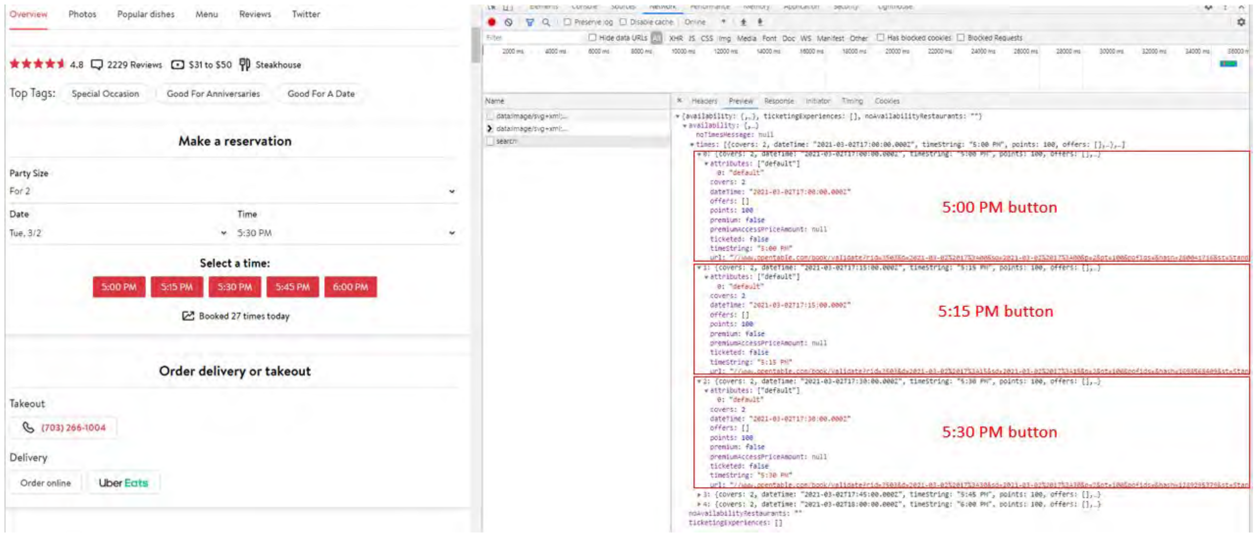


```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
    noTimesMessage: null
   ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
     ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
       ▾attributes: ["default"]
          0: "default"
        covers: 2
        dateTime: "2021-03-02T17:00:00.000Z"
        offers: []
        points: 100
        premium: false
        premiumAccessPriceAmount: null
        ticketed: false
        timeString: "5:00 PM"
        url: "//www.opentable.com/book/validate?rid=[illegible]
     ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
       ▾attributes: ["default"]
          0: "default"
        covers: 2
        dateTime: "2021-03-02T17:15:00.000Z"
        offers: []
        points: 100
        premium: false
        premiumAccessPriceAmount: null
        ticketed: false
        timeString: "5:15 PM"
        url: "//www.opentable.com/book/validate?rid=[illegible]
     ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
       ▾attributes: ["default"]
          0: "default"
        covers: 2
        dateTime: "2021-03-02T17:30:00.000Z"
        offers: []
        points: 100
        premium: false
        premiumAccessPriceAmount: null
        ticketed: false
        timeString: "5:30 PM"
        url: "//www.opentable.com/book/validate?rid=[illegible]
     ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
     ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```

258. On information and belief, the authoring tool selects a symbolic name (e.g., JSON data) from the web component corresponding to the defined UI object. In the example below, symbolic name "0" corresponds to the "5:00 PM" button UI object; symbolic name "1" corresponds to the "5:15 PM" button UI object; symbolic name "2" corresponds to the "5:30 PM" button UI object.

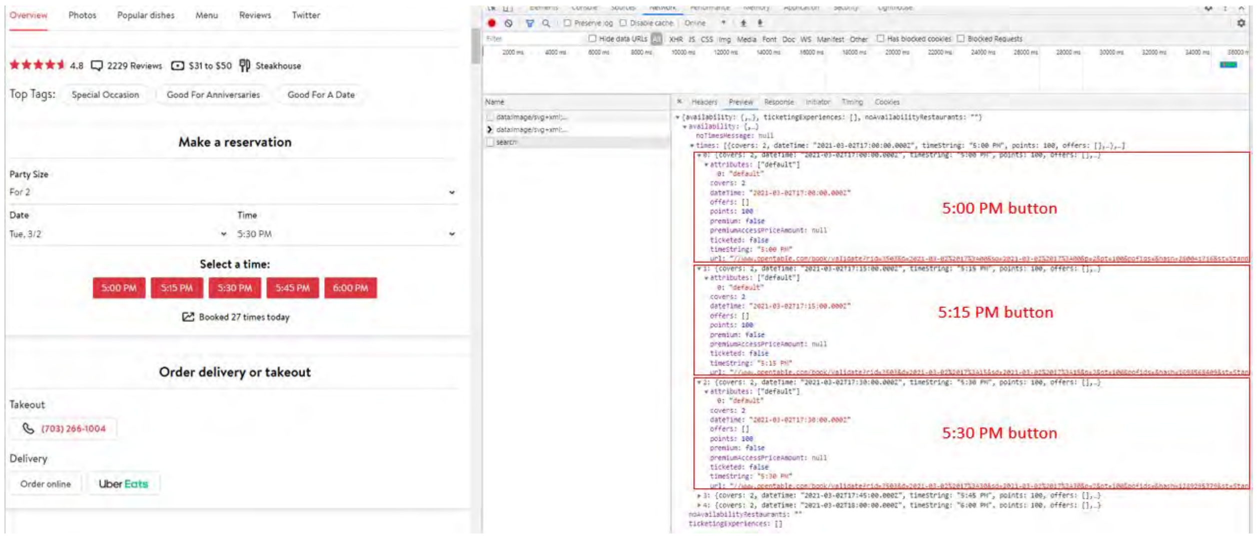
259. Each symbolic name has an associated data format class type corresponding to a subclass of User Interface (UI) objects that support the data format type of the symbolic name, and has a preferred UI object. By way of example, when "Find a table" button is pressed, the OpenTable platform creates and displays a subclass of UI objects, e.g., five button objects, that represent five available time slots. In particular, symbolic name "0" associates with the "5:00 PM" button and the button object is the preferred UI object.





```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
    noTimesMessage: null
   ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
    ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:00:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:00 PM"
      url: "//www.opentable.com/book/validate?rid=[illegible]
    ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:15:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:15 PM"
      url: "//www.opentable.com/book/validate?rid=[illegible]
    ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:30:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:30 PM"
      url: "//www.opentable.com/book/validate?rid=[illegible]
    ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
    ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```

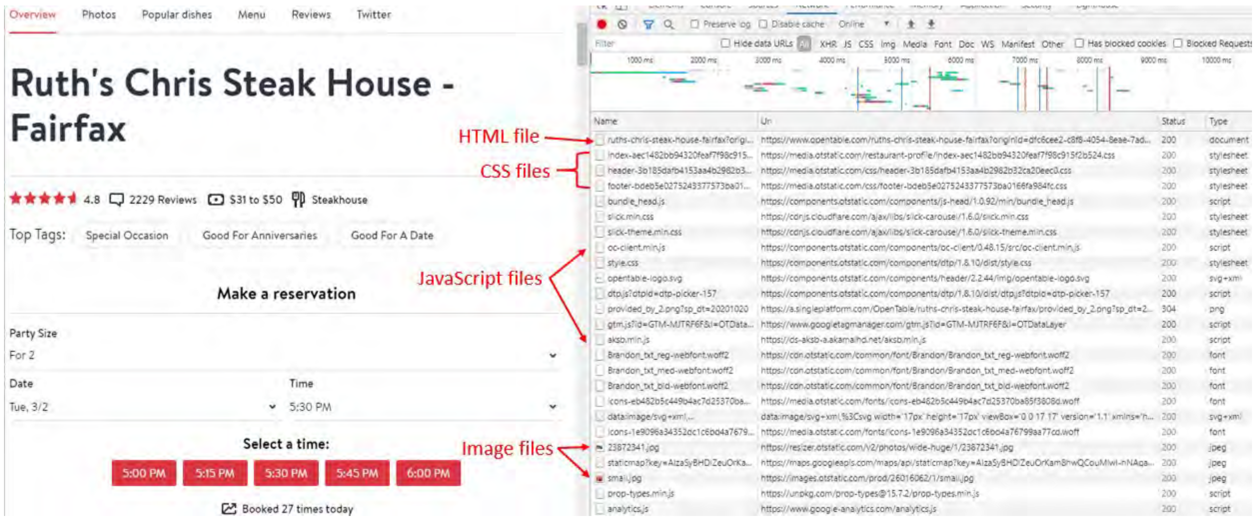
260. The authoring tool associates the selected symbolic name from the web component corresponding to the defined UI object with the defined UI object. In the example below, the authoring tool associates "5:00 PM" button UI object with the symbolic name "0". In particular, the symbolic name "timeString" associates with the value "5:00 PM" that is displayed on the button UI object.



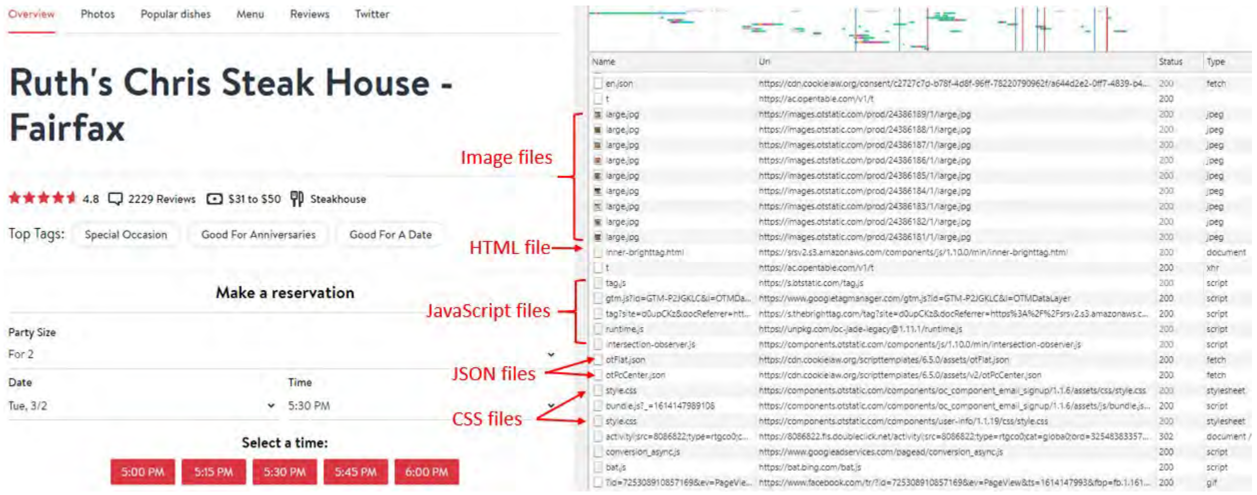
```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
    noTimesMessage: null
   ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
    ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:00:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:00 PM"
      url: "//www.opentable.com/book/validate?rid=[illegible]
    ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:15:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:15 PM"
      url: "//www.opentable.com/book/validate?rid=[illegible]
    ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:30:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:30 PM"
      url: "//www.opentable.com/book/validate?rid=[illegible]
    ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
    ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```

261. The authoring tool produces a device-independent code (i.e., an application) that includes the JSON data, as well as standard HTML, CSS and Javascript code. In the example below, the authoring tool generates a device-independent application for a restaurant profile, in which five button UI objects are used to represent five available time slots for restaurant

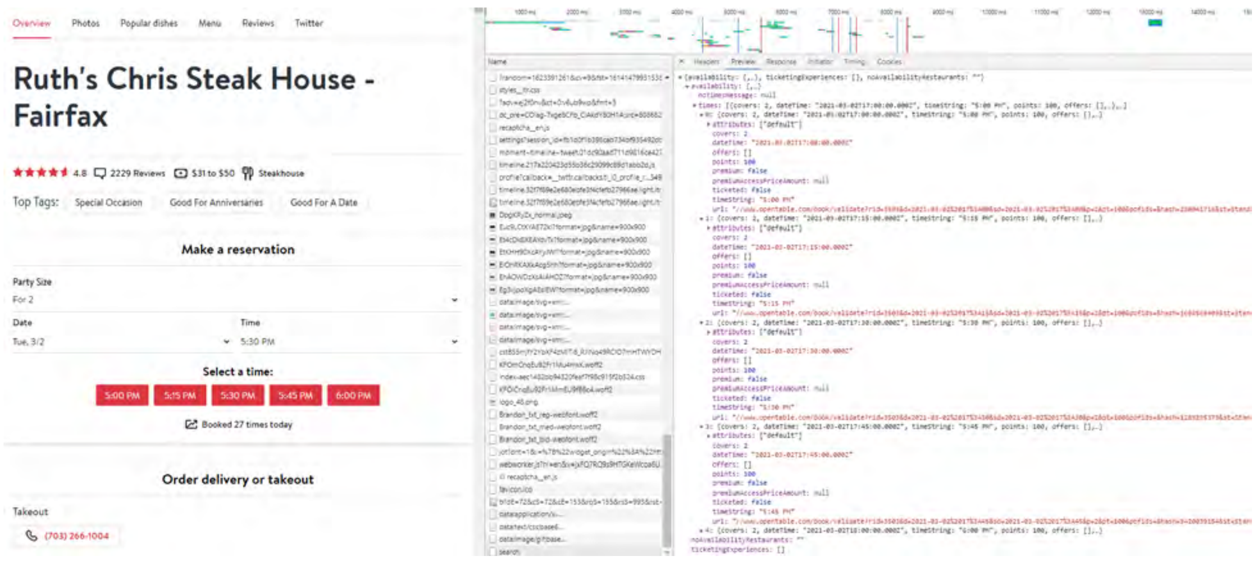
reservation. The symbolic names "0" is associated with the "5:00 PM" button UI object; the symbolic name "1" is associated with the "5:15 PM" button UI object; the symbolic name "2" is associated with the "5:30 PM" button UI object; the symbolic name "3" is associated with the "5:45 PM" button UI object; and the symbolic name "4" is associated with the "6:00 PM" button UI object.



Name	Url	Status	Type
ruths-chris-steak-house-fairfax?origi...	https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8eae-7ad...	200	document
index-aec1482bb94320feaf7f98c915...	https://media.otstatic.com/restaurant-profile/index-aec1482bb94320feaf7f98c915f2b524.css	200	stylesheet
header-3b185dafb4153aa4b2982b3...	https://media.otstatic.com/css/header-3b185dafb4153aa4b2982b32ca20eec0.css	200	stylesheet
footer-bdeb5e027524337757 3ba01...	https://media.otstatic.com/css/footer-bdeb5e0275243377573ba0166fa984fc.css	200	stylesheet
bundle_head.js	https://components.otstatic.com/components/js-head/1.0.92/min/bundle_head.js	200	script
slick.min.css	https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick.min.css	200	stylesheet
slick-theme.min.css	https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick-theme.min.css	200	stylesheet
oc-client.min.js	https://components.otstatic.com/components/oc-client/0.48.15/src/oc-client.min.js	200	script
style.css	https://components.otstatic.com/components/dtp/1.8.10/dist/style.css	200	stylesheet
opentable-logo.svg	https://components.otstatic.com/components/header/2.2.44/img/opentable-logo.svg	200	svg+xml
dtp.js?dtpId=dtp-picker-157	https://components.otstatic.com/components/dtp/1.8.10/dist/dtp.js?dtpId=dtp-picker-157	200	script
provided_by_2.png?sp_dt=20201020	https://a.singleplatform.com/OpenTable/ruths-chris-steak-house-fairfax/provided_by_2.png?sp_dt=2...	304	png
gtm.js?id=GTM-MJTRF6F&l=OTData...	https://www.googletagmanager.com/gtm.js?id=GTM-MJTRF6F&l=OTDataLayer	200	script
aksb.min.js	https://ds-aksb-a.akamaihd.net/aksb.min.js	200	script
Brandon_txt_reg-webfont.woff2	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_reg-webfont.woff2	200	font
Brandon_txt_med-webfont.woff2	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_med-webfont.woff2	200	font
Brandon_txt_bld-webfont.woff2	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_bld-webfont.woff2	200	font
icons-eb482b5c449b4ac7d25370ba...	https://media.otstatic.com/fonts/icons-eb482b5c449b4ac7d25370ba85f3808d.woff	200	font
data:image/svg+xml,...	data:image/svg+xml,%3Csvg width='17px' height='17px' viewBox='0 0 17 17' version='1.1' xmlns='h...	200	svg+xml
icons-1e9096a34352dc1c6bd4a7679...	https://media.otstatic.com/fonts/icons-1e9096a34352dc1c6bd4a76799aa77cd.woff	200	font
23872341.jpg	https://resizer.otstatic.com/v2/photos/wide-huge/1/23872341.jpg	200	jpeg
staticmap?key=AIzaSyBHDlZeuOrKa...	https://maps.googleapis.com/maps/api/staticmap?key=AIzaSyBHDlZeuOrKam8hwQCouMlwI-nNAqa...	200	jpeg
small.jpg	https://images.otstatic.com/prod/26016062/1/small.jpg	200	jpeg
prop-types.min.js	https://unpkg.com/prop-types@15.7.2/prop-types.min.js	200	script
analytics.js	https://www.google-analytics.com/analytics.js	200	script



Name	Url	Status	Type
en.json	https://cdn.cookielaw.org/consent/c2727c7d-b78f-4d8f-96ff-78220790962f/a644d2e2-0ff7-4839-b4...	200	fetch
t	https://ac.opentable.com/v1/t	200	
large.jpg	https://images.otstatic.com/prod/24386189/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386188/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386187/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386186/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386185/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386184/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386183/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386182/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386181/1/large.jpg	200	jpeg
inner-brighttag.html	https://srsv2.s3.amazonaws.com/components/js/1.10.0/min/inner-brighttag.html	200	document
t	https://ac.opentable.com/v1/t	200	xhr
tag.js	https://s.otstatic.com/tag.js	200	script
gtm.js?id=GTM-P2JGKLC&l=OTMDa...	https://www.googletagmanager.com/gtm.js?id=GTM-P2JGKLC&l=OTMDataLayer	200	script
tag?site=d0upCKz&docReferrer=htt...	https://s.thebrighttag.com/tag?site=d0upCKz&docReferrer=https%3A%2F%2Fsrsv2.s3.amazonaws.c...	200	script
runtime.js	https://unpkg.com/oc-jade-legacy@1.11.1/runtime.js	200	script
intersection-observer.js	https://components.otstatic.com/components/js/1.10.0/min/intersection-observer.js	200	script
otFlat.json	https://cdn.cookielaw.org/scripttemplates/6.5.0/assets/otFlat.json	200	fetch
otPcCenter.json	https://cdn.cookielaw.org/scripttemplates/6.5.0/assets/v2/otPcCenter.json	200	fetch
style.css	https://components.otstatic.com/components/oc_component_email_signup/1.1.6/assets/css/style.css	200	stylesheet
bundle.js?_=1614147989106	https://components.otstatic.com/components/oc_component_email_signup/1.1.6/assets/js/bundle.js...	200	script
style.css	https://components.otstatic.com/components/user-info/1.1.19/css/style.css	200	stylesheet
activity;src=8086822;type=rtgco0;c...	https://8086822.fls.doubleclick.net/activity;src=8086822;type=rtgco0;cat=globa0;ord=32548383357...	302	document /
conversion_async.js	https://www.googleadservices.com/pagead/conversion_async.js	200	script
bat.js	https://bat.bing.com/bat.js	200	script
?id=725308910857169&ev=PageVie...	https://www.facebook.com/tr/?id=725308910857169&ev=PageView&ts=1614147993&fbp=fb.1.161...	200	gif



Ruth's Chris Steak House - Fairfax

4.8 2229 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Good For A Date

Make a reservation

Party Size: For 2

Date: Tue, 3/2

Time: 5:30 PM

Select a time:

5:00 PM | 5:15 PM | 5:30 PM | 5:45 PM | 6:00 PM

Booked 27 times today

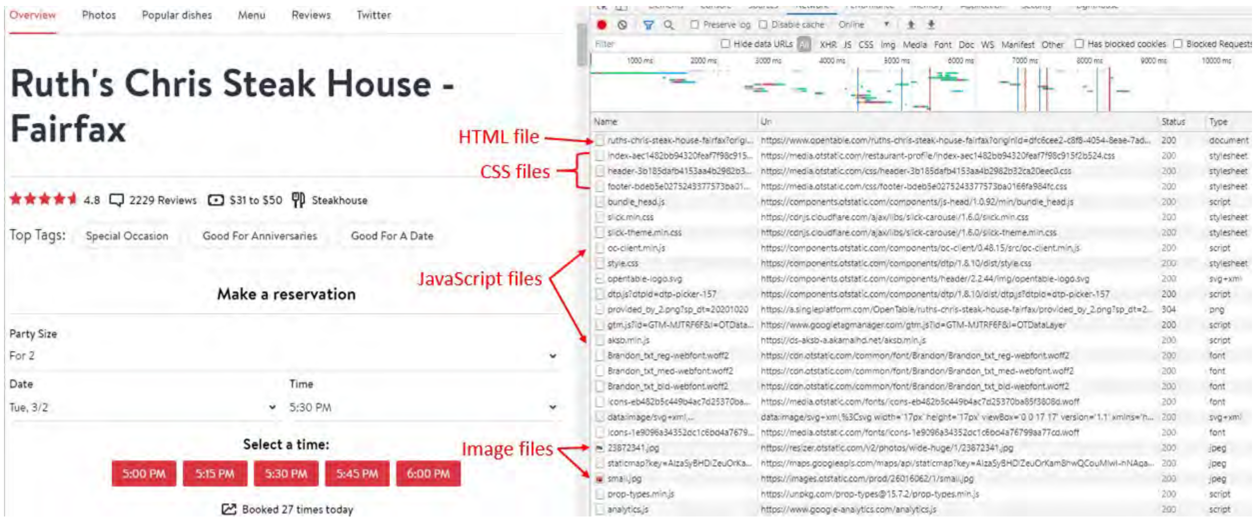
Order delivery or takeout

Takeout

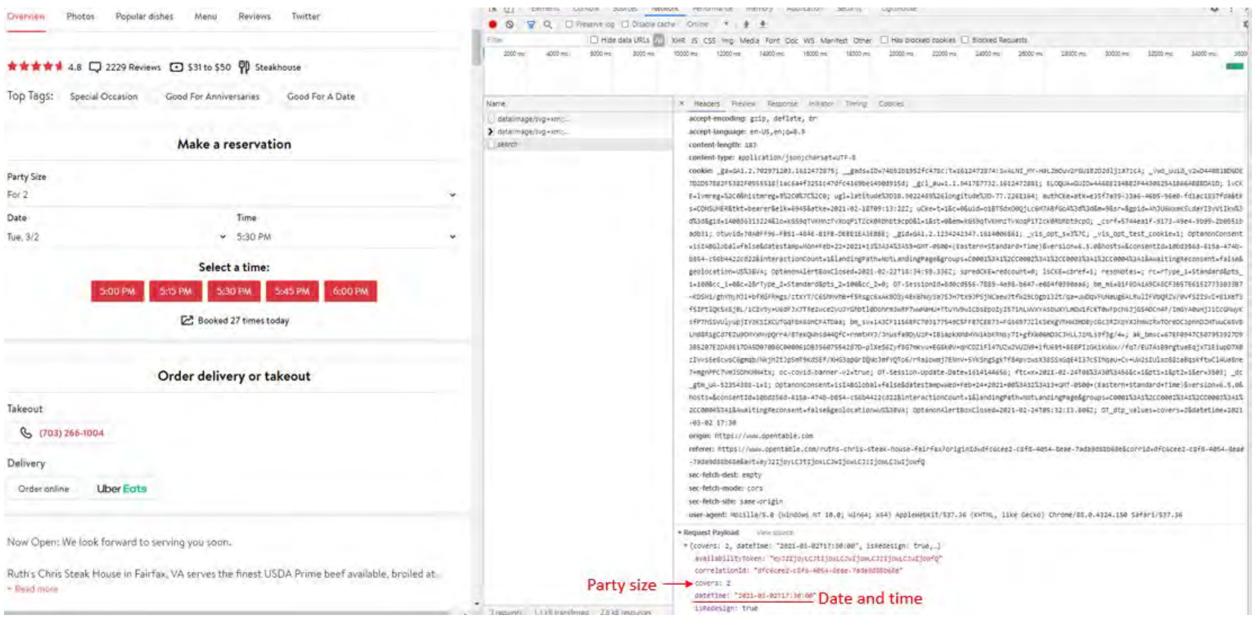
(703) 266-1004

```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
    noTimesMessage: null
   ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
    ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
      ▸attributes: ["default"]
       covers: 2
       dateTime: "2021-03-02T17:00:00.000Z"
       offers: []
       points: 100
       premium: false
       premiumAccessPriceAmount: null
       ticketed: false
       timeString: "5:00 PM"
       url: "//www.opentable.com/book/validate?rid=[illegible]
    ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
      ▸attributes: ["default"]
       covers: 2
       dateTime: "2021-03-02T17:15:00.000Z"
       offers: []
       points: 100
       premium: false
       premiumAccessPriceAmount: null
       ticketed: false
       timeString: "5:15 PM"
       url: "//www.opentable.com/book/validate?rid=[illegible]
    ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
      ▸attributes: ["default"]
       covers: 2
       dateTime: "2021-03-02T17:30:00.000Z"
       offers: []
       points: 100
       premium: false
       premiumAccessPriceAmount: null
       ticketed: false
       timeString: "5:30 PM"
       url: "//www.opentable.com/book/validate?rid=[illegible]
    ▾3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
      ▸attributes: ["default"]
       covers: 2
       dateTime: "2021-03-02T17:45:00.000Z"
       offers: []
       points: 100
       premium: false
       premiumAccessPriceAmount: null
       ticketed: false
       timeString: "5:45 PM"
       url: "//www.opentable.com/book/validate?rid=[illegible]
    ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```

262. The OpenTable platform can execute the application and the player on a device to produce the contents for the restaurant profile, where a user can input a value associated with an input symbolic name to an input of a defined UI object. In the example below, the OpenTable platform converts HTML, CSS, Javascript, image, and other files from an application into an active property listing using a player. The active restaurant profile accepts a search input value (e.g., party size, date, and time). For example, a client can make a reservation for party size of " for 2", date of "Tue, 3/2", and time of "5:30 PM". An input symbolic name "covers" is associated with party size and another input symbolic name "dateTime" is associated with the date and time. These symbolic names along with the input values are transmitted to the OpenTable API through an HTTP request protocol such as a POST method call.



Name	Url	Status	Type
ruths-chris-steak-house-fairfax?origi...	https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8eae-7ad...	200	document
index-aec1482bb94320feaf7f98c915...	https://media.otstatic.com/restaurant-profile/index-aec1482bb94320feaf7f98c915f2b524.css	200	stylesheet
header-3b185dafb4153aa4b2982b3...	https://media.otstatic.com/css/header-3b185dafb4153aa4b2982b32ca20eec0.css	200	stylesheet
footer-bdeb5e0275243377573ba01...	https://media.otstatic.com/css/footer-bdeb5e0275243377573ba0166fa984fc.css	200	stylesheet
bundle_head.js	https://components.otstatic.com/components/js-head/1.0.92/min/bundle_head.js	200	script
slick.min.css	https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick.min.css	200	stylesheet
slick-theme.min.css	https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick-theme.min.css	200	stylesheet
oc-client.min.js	https://components.otstatic.com/components/oc-client/0.48.15/src/oc-client.min.js	200	script
style.css	https://components.otstatic.com/components/dtp/1.8.10/dist/style.css	200	stylesheet
opentable-logo.svg	https://components.otstatic.com/components/header/2.2.44/img/opentable-logo.svg	200	svg+xml
dtp.js?dtpId=dtp-picker-157	https://components.otstatic.com/components/dtp/1.8.10/dist/dtp.js?dtpId=dtp-picker-157	200	script
provided_by_2.png?sp_dt=20201020	https://a.singleplatform.com/OpenTable/ruths-chris-steak-house-fairfax/provided_by_2.png?sp_dt=2...	304	png
gtm.js?id=GTM-MJTRF6F&l=OTData...	https://www.googletagmanager.com/gtm.js?id=GTM-MJTRF6F&l=OTDataLayer	200	script
aksb.min.js	https://ds-aksb-a.akamaihd.net/aksb.min.js	200	script
Brandon_txt_reg-webfont.woff2	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_reg-webfont.woff2	200	font
Brandon_txt_med-webfont.woff2	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_med-webfont.woff2	200	font
Brandon_txt_bld-webfont.woff2	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_bld-webfont.woff2	200	font
icons-eb482b5c449b4ac7d25370ba...	https://media.otstatic.com/fonts/icons-eb482b5c449b4ac7d25370ba85f3808d.woff	200	font
data:image/svg+xml,...	data:image/svg+xml,%3Csvg width='17px' height='17px' viewBox='0 0 17 17' version='1.1' xmlns='h...	200	svg+xml
icons-1e9096a34352dc1c6bd4a7679...	https://media.otstatic.com/fonts/icons-1e9096a34352dc1c6bd4a76799aa77cd.woff	200	font
23872341.jpg	https://resizer.otstatic.com/v2/photos/wide-huge/1/23872341.jpg	200	jpeg
staticmap?key=AIzaSyBHDIZeuOrKa...	https://maps.googleapis.com/maps/api/staticmap?key=AIzaSyBHDIZeuOrKam8hwQCouMIwI-nNAqa...	200	jpeg
small.jpg	https://images.otstatic.com/prod/26016062/1/small.jpg	200	jpeg
prop-types.min.js	https://unpkg.com/prop-types@15.7.2/prop-types.min.js	200	script
analytics.js	https://www.google-analytics.com/analytics.js	200	script



Now Open: We look forward to serving you soon.

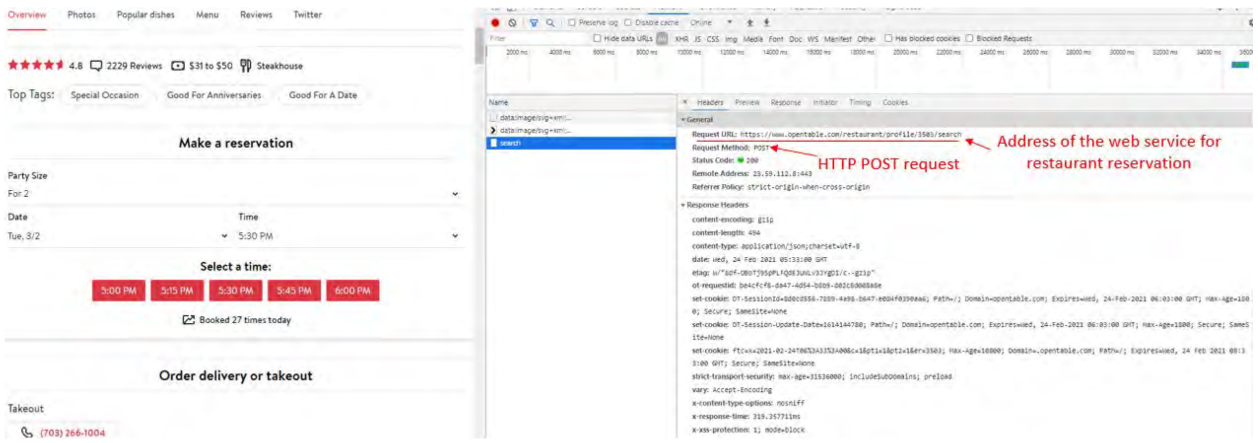
Ruth's Chris Steak House in Fairfax, VA serves the finest USDA Prime beef available, broiled at...

+ Read more

```
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
content-length: 183
content-type: application/json;charset=UTF-8
cookie: _ga=GA1.2.702971203.1612472875; __gads=ID=74b52b1952fc47bc:T=1612472874:S=ALNI_MY-H8L2mOuv2rGU182D2dljj871cA; [illegible]
... OT_dtp_values=covers=2&datetime=2021-03-02 17:30
origin: https://www.opentable.com
referer: https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8eae-7ada9d88b68e&corrid=dfc6cee2-c8f8-4054-8eae-7ada9d88b68e&avt=eyJ2IjoyLCJtIjoxLCJwIjowLCJiIjowLCJuIjowfQ
sec-fetch-dest: empty
sec-fetch-mode: cors
sec-fetch-site: same-origin
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36
▾ Request Payload    view source
▾ {covers: 2, dateTime: "2021-03-02T17:30:00", isRedesign: true,…}
    availabilityToken: "eyJ2IjoyLCJtIjoxLCJwIjowLCJiIjowLCJuIjowfQ"
    correlationId: "dfc6cee2-c8f8-4054-8eae-7ada9d88b68e"
    covers: 2
    dateTime: "2021-03-02T17:30:00"
    isRedesign: true
```

263. When the OpenTable platform executes the application and player on the device, the OpenTable platform can cause the device to provide an input value and corresponding input symbolic name to a web service. By way of example, the OpenTable platform causes the device to transmit the input value and a corresponding input symbolic name (e.g., JSON data for the input) to the OpenTable's reservation web service through an HTTP request protocol, such as a POST method call. In the example below, when a client wants to make a reservation for party size of " for 2", date of "Tue, 3/2", and time of "5:30 PM", the OpenTable platform causes the device to

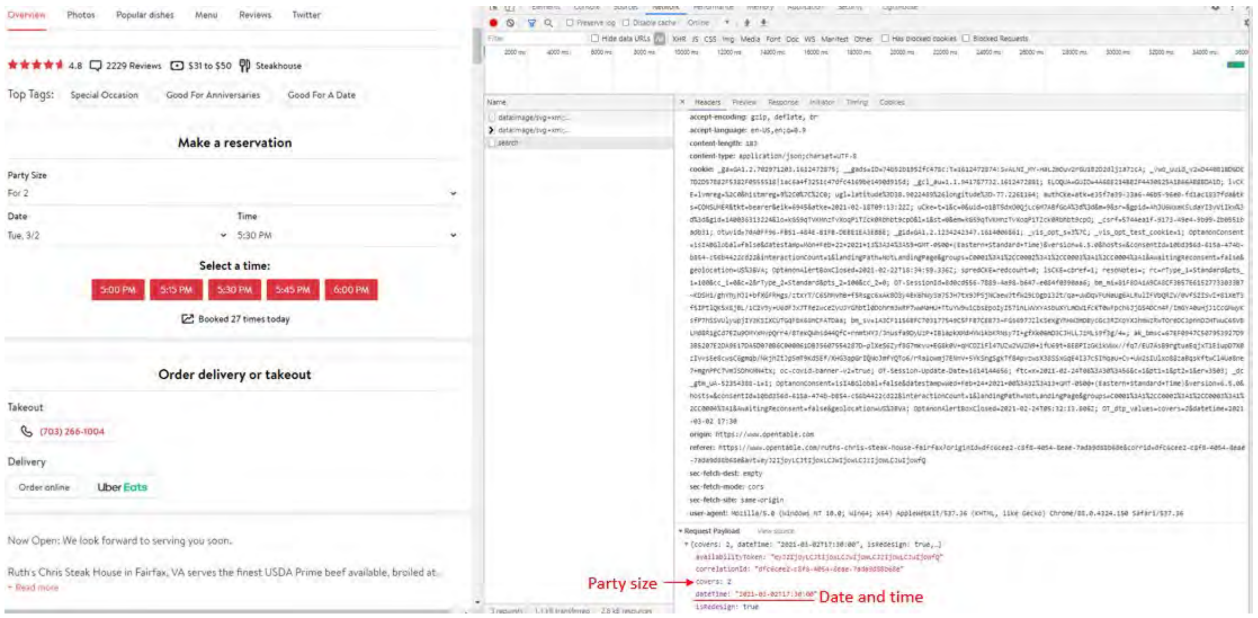
send a JSON data including the symbolic name "covers" associating with value "2" and the symbolic name "covers" associating with the value "2021-03-02T17:30:00" to the OpenTable's reservation web service with an HTTP POST request.



Overview Photos Popular dishes Menu Reviews Twitter
4.8 2229 Reviews $31 to $50 Steakhouse
Top Tags: Special Occasion Good For Anniversaries Good For A Date
Make a reservation
Party Size For 2
Date Tue, 3/2
Time 5:30 PM
Select a time:
5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM
Booked 27 times today
Order delivery or takeout
Takeout (703) 266-1004
Preserve log Disable cache Online
Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other Has blocked cookies Blocked Requests
Name
search
Headers Preview Response Initiator Timing Cookies
Address of the web service for restaurant reservation
HTTP POST request

▾ General
Request URL: https://www.opentable.com/restaurant/profile/3503/search
Request Method: POST
Status Code: 200
Remote Address: 23.59.112.8:443
Referrer Policy: strict-origin-when-cross-origin

▾ Response Headers
content-encoding: gzip
content-length: 494
content-type: application/json;charset=utf-8
date: Wed, 24 Feb 2021 05:33:00 GMT
etag: W/"1df-OBoTJ9SpPLFQdEJUNLV3JYgD1/c--gzip"
ot-requestid: be4cfcf8-da47-4d54-b8b9-d02c6d008a5e
set-cookie: OT-SessionId=8d0cd556-7889-4a98-b647-e084f0390aa6; Path=/; Domain=opentable.com; Expires=Wed, 24-Feb-2021 06:03:00 GMT; Max-Age=1800; Secure; SameSite=None
set-cookie: OT-Session-Update-Date=1614144780; Path=/; Domain=opentable.com; Expires=Wed, 24-Feb-2021 06:03:00 GMT; Max-Age=1800; Secure; SameSite=None
set-cookie: ftc=x=2021-02-24T06%3A33%3A00&c=1&pt1=1&pt2=1&er=3503; Max-Age=10800; Domain=.opentable.com; Path=/; Expires=Wed, 24 Feb 2021 08:33:00 GMT; Secure; SameSite=None
strict-transport-security: max-age=31536000; includeSubDomains; preload
vary: Accept-Encoding
x-content-type-options: nosniff
x-response-time: 319.357711ms
x-xss-protection: 1; mode=block



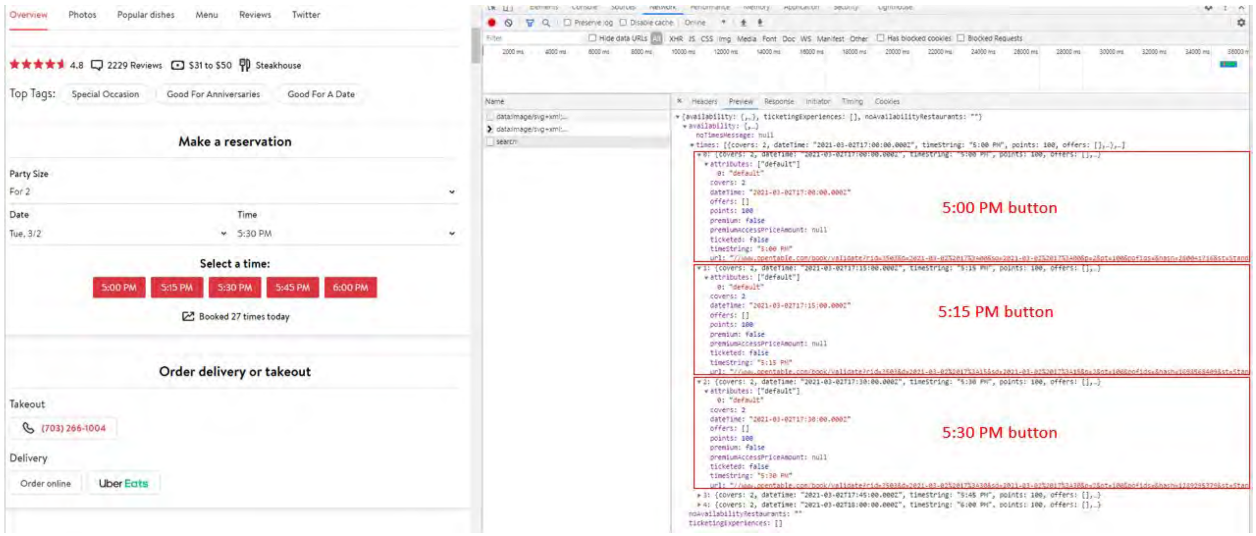
Overview Photos Popular dishes Menu Reviews Twitter
4.8 2229 Reviews $31 to $50 Steakhouse
Top Tags: Special Occasion Good For Anniversaries Good For A Date
Make a reservation
Party Size For 2
Date Tue, 3/2
Time 5:30 PM
Select a time:
5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM
Booked 27 times today
Order delivery or takeout
Takeout (703) 266-1004
Delivery Order online Uber Eats
Now Open: We look forward to serving you soon.
Ruth's Chris Steak House in Fairfax, VA serves the finest USDA Prime beef available, broiled at...
+ Read more
Headers Preview Response Initiator Timing Cookies
Party size
Date and time

accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
content-length: 183
content-type: application/json;charset=UTF-8
cookie: _ga=GA1.2.702971203.1612472875; [illegible] OT_dtp_values=covers=2&datetime=2021-03-02 17:30
origin: https://www.opentable.com
referer: https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8eae-7ada9d88b68e&corrid=dfc6cee2-c8f8-4054-8eae-7ada9d88b68e&avt=eyJ2IjoyLCJtIjoxLCJwIjowLCJjIjowLCJuIjowfQ
sec-fetch-dest: empty
sec-fetch-mode: cors
sec-fetch-site: same-origin
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36

▾ Request Payload view source
```
▾ {covers: 2, dateTime: "2021-03-02T17:30:00", isRedesign: true,…}
    availabilityToken: "eyJ2IjoyLCJtIjoxLCJwIjowLCJjIjowLCJuIjowfQ"
    correlationId: "dfc6cee2-c8f8-4054-8eae-7ada9d88b68e"
    covers: 2
    dateTime: "2021-03-02T17:30:00"
    isRedesign: true
```

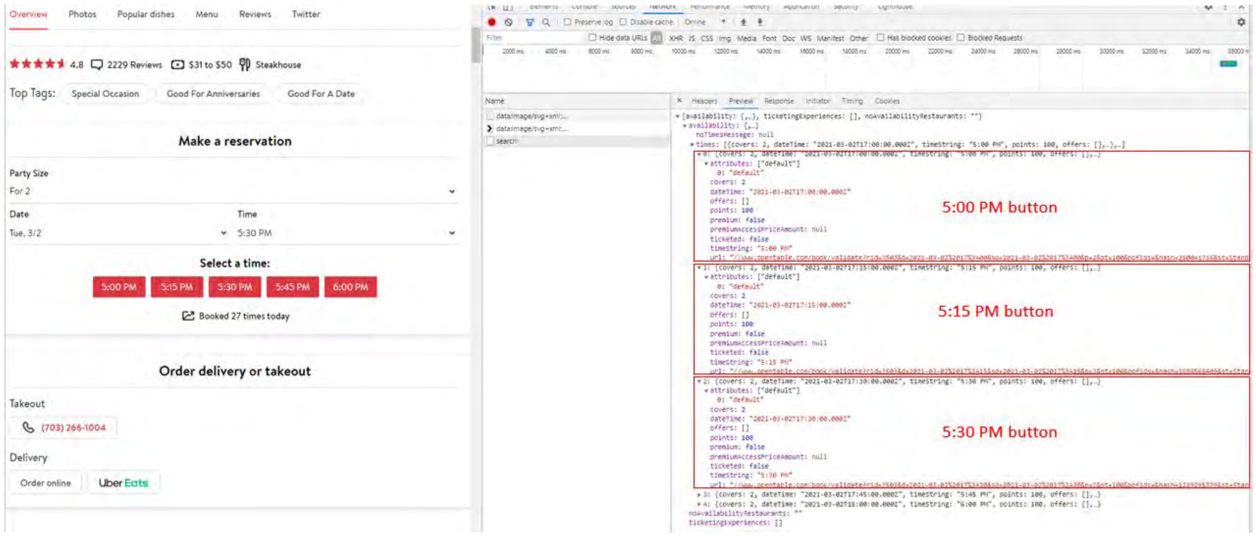
264. The OpenTable platform provides the input value and a corresponding input symbolic name to the web service, and in turn causes the web service to generate an output value and a corresponding symbolic name for the output. In the example described below, when a search for "2021-03-02T17:30:00" is input, the OpenTable platform causes the OpenTable's reservation

web service to generate five corresponding output available time slots around the date and time in the form of JSON data.



```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
 ▾availability: {,…}
   noTimesMessage: null
  ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
   ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
    ▾attributes: ["default"]
      0: "default"
     covers: 2
     dateTime: "2021-03-02T17:00:00.000Z"
     offers: []
     points: 100
     premium: false
     premiumAccessPriceAmount: null
     ticketed: false
     timeString: "5:00 PM"
     url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A00&sd=2021-03-02%2017%3A00&p=2&pt=100&pofids=&hash=2800417168&st=Stand
   ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
    ▾attributes: ["default"]
      0: "default"
     covers: 2
     dateTime: "2021-03-02T17:15:00.000Z"
     offers: []
     points: 100
     premium: false
     premiumAccessPriceAmount: null
     ticketed: false
     timeString: "5:15 PM"
     url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A15&sd=2021-03-02%2017%3A15&p=2&pt=100&pofids=&hash=1698568409&st=Stan
   ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
    ▾attributes: ["default"]
      0: "default"
     covers: 2
     dateTime: "2021-03-02T17:30:00.000Z"
     offers: []
     points: 100
     premium: false
     premiumAccessPriceAmount: null
     ticketed: false
     timeString: "5:30 PM"
     url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A30&sd=2021-03-02%2017%3A30&p=2&pt=100&pofids=&hash=1289295379&st=Stan
   ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
   ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```

265. The player receives the output symbolic name and output value and provides instructions to the display of the device to present the output value in the API. In the example below, the player provides instructions to the browser of the device to display five output button UI objects.



```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
 ▾availability: {,…}
   noTimesMessage: null
  ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
   ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
   ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
   ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
   ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
   ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```

266. The presence of the above referenced features is demonstrated, by way of example, by testing the Accused Instrumentalities for investigative purposes on https://www.booking.com/,

https://join.booking.com/, https://partner.booking.com/en-us, and/or https://ycs.agoda.com/en-us/kipp/public/home, and by reference to publicly available information, including the following:

- https://www.booking.com/,
- https://join.booking.com/,
- https://partner.booking.com/en-us,
- https://partner.booking.com/en-us/help/working-booking/how-do-i-join-bookingcom,
- https://partner.booking.com/en-us/help/working-booking/how-can-i-set-my-property-easily,
- https://ycs.agoda.com/en-us/kipp/public/home,
- https://www.agoda.com/info/ycs-online-registration.html?cid=1844104,
- https://www.agoda.com/info/privacy.html?cid=1844104,
- https://partners.agoda.com/en-us/faq.html,
- https://agodapropertyhelp.zendesk.com/hc/en-us,
- https://partners.agoda.com/,
- https://agodapropertyhelp.zendesk.com/hc/en-us/articles/115009545508-How-do-I-list-my-property-on-Agoda-com-,
- https://www.opentable.com/,
- https://support.opentable.com/s/?language=en_US,
- https://platform.opentable.com/documentation/, and
- https://restaurant.opentable.com/.

267. On information and belief, Defendants have had knowledge of the ‘287 patent and their infringement thereof at least as early as July 27, 2019, and no later than November 30, 2020, when Plaintiff provided notice of the ‘287 patent and Defendants’ infringement of the ‘287 patent. Furthermore, Defendants have been aware of the ‘287 patent and their infringement thereof since at least the filing of the original Complaint, D.I. 1, on December 1, 2020.

268. On information and belief, Defendants have contributed and are contributing to the infringement of the ‘287 patent because Defendants know that the infringing aspects of their infringing products and services, including but not limited to the Accused Instrumentalities, are made for use in an infringement, and are not staple articles of commerce suitable for substantial non-infringing uses.

269. On information and belief, Defendants have induced and are inducing the infringement of the ‘287 patent, with knowledge of the ‘287 patent and that their acts, including without limitation using, offering to sell, selling within, and importing into the United States, the Accused Instrumentalities, would aid and abet and induce infringement by end users of the foregoing.

270. In particular, Defendants’ actions that aid and abet others such as customers, clients, partners, developers, and end users to infringe include advertising and distributing the Accused Instrumentalities, providing instructional materials, training, and other services regarding the Accused Instrumentalities, and providing free listings for the Accused Instrumentalities. Defendants actively encouraged the adoption of the Accused Instrumentalities and provided support sites for the vast network of developers working with the Accused Instrumentalities, emphasizing the simple and user-friendly nature of the Accused Instrumentalities, for example, explaining that “Registration can take as little as 15 minutes to complete—get started today” and that Defendants provides “24/7 support by phone or email” (*see, e.g.*, https://join.booking.com/), that “Through one platform our hotel partners are distributed across both priceline.com and agoda.com maximizing performance across our Retail, Private, Opaque and Vacation Packages programs. Our partners can easily set rates, promotions, and change hotel details with a few simple clicks of a mouse or on the go with a mobile device” (*see, e.g.*, https://ycs.agoda.com/en-us/kipp/public/home), and that “From online ordering and takeout to powerful marketing and experiences, make more money when you access our network of millions” and that restaurants can “[g]et discovered and capture the business of the millions of people, around the world and in your neighborhood, searching on OpenTable” (*see, e.g.*,

https://restaurant.opentable.com/?utm_source=dinersite&utm_medium=referral&utm_campaign =topnav&Lead.LeadSource=DinerSite&Lead.Marketing_ID__c=topnav). On information and belief, Defendants have engaged in such actions with specific intent to cause infringement or with willful blindness to the resulting infringement because Defendants have had actual knowledge of the ‘287 patent and knowledge that their acts were inducing infringement of the ‘287 patent since at least the date Defendants received notice that their activities infringed the ‘287 patent.

271. Defendants' acts of infringement have caused damage to Plaintiff, and Plaintiff is entitled to recover damages from Defendants in an amount subject to proof at trial.

272. Defendants' infringement of Plaintiff’s rights under the ‘287 patent will continue to damage Plaintiff’s business, causing irreparable harm, for which there is no adequate remedy at law, unless enjoined by this Court.

273. On information and belief, Defendants haveacted with disregard of Plaintiff’s patent rights, without any reasonable basis for doing so, and have willfully infringed and do willfully infringe the ‘287 patent.

274. The foregoing is illustrative of Defendants' infringement of the ‘287 patent. Plaintiff reserves the right to identify additional claims and Accused Instrumentalities in accordance with the Court’s local rules and applicable scheduling orders.

COUNT V – INFRINGEMENT OF U.S. PATENT NO. 9,928,044

275. Plaintiff incorporates by reference paragraphs 1 to 55 above as if fully set forth herein.

276. On information and belief, Defendants have infringed and are infringing the ‘044 patent under 35 U.S.C. § 271, either literally and/or under the doctrine of equivalents, directly and/or indirectly.

277. On information and belief, Booking-BV has infringed and continues to infringe the ‘044 patent by performing, without authority, one or more of the following acts: making, using, offering to sell, selling within, and importing into, the United States products and services that practice the claimed inventions of the ‘044 patent, including but not limited to the WebDirect

Platform and the Booking Platform.[9]

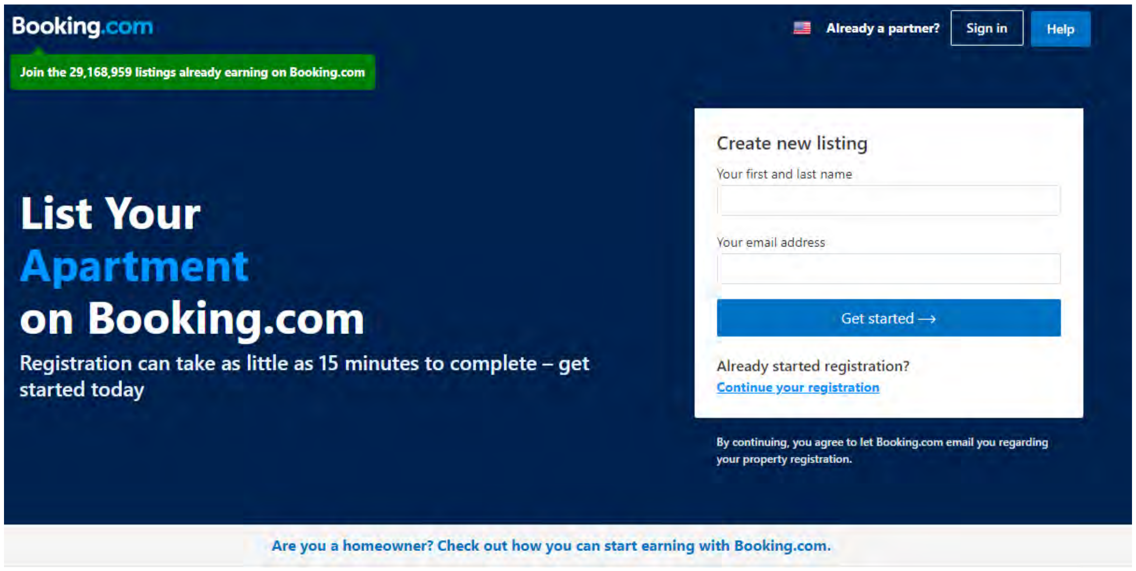
278. On information and belief, Priceline and Agoda (the "YCS infringers") have infringed and continue to infringe the '044 patent by performing, without authority, one or more of the following acts: making, using, offering to sell, selling within, and importing into, the United States products and services that practice the claimed inventions of the '044 patent, including but not limited to the YCS Platform.

279. On information and belief, OpenTable has infringed and continues to infringe the '044 patent by performing, without authority, one or more of the following acts during relevant time periods: making, using, offering to sell, selling within, and importing into the United States products and services that practice the claimed inventions of the '044 patent, including but not limited to the OpenTable Platform.

The Booking Platform:

280. Booking-BV infringes at least claim 15 of the '044 patent through a combination of features in the Booking Platform that collectively practice each claimed limitation of claim 15. By way of example, Booking-BV provides the Booking Platform for creating property listing websites for hotels, homes, apartments, and other lodgings where travelers can view and book accommodations.

[9] This Count focuses its infringement allegations on the Booking Platform, YCS Platform, and the OpenTable Platform. Upon information and belief, the WebDirect Platform operates in a similar fashion as the Booking and YCS Platforms, and discovery of the WebDirect Platform, including confidential documents related thereto, will confirm these facts.



List Your Apartment on Booking.com

Registration can take as little as 15 minutes to complete – get started today

Create new listing

Your first and last name

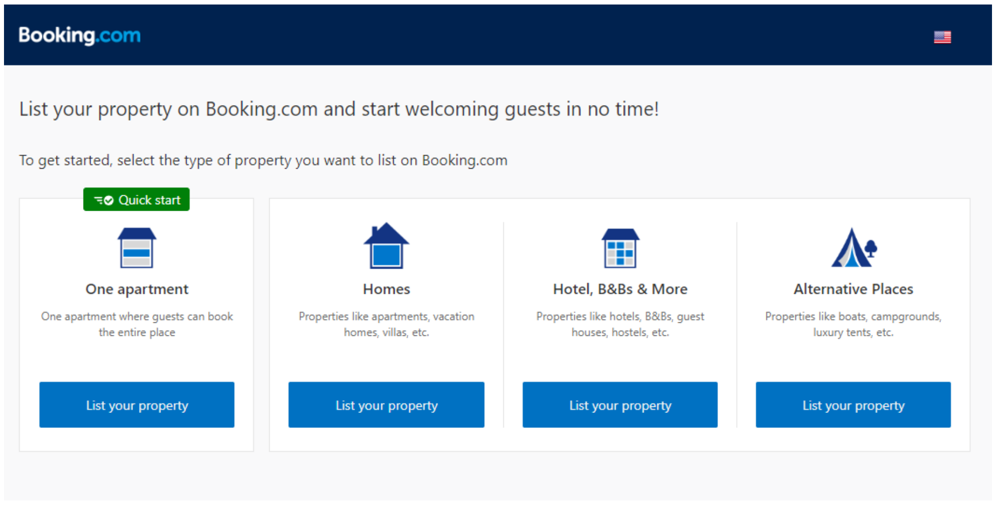
Your email address

Already started registration?
Continue your registration

By continuing, you agree to let Booking.com email you regarding your property registration.

Are you a homeowner? Check out how you can start earning with Booking.com.

[10]



List your property on Booking.com and start welcoming guests in no time!

To get started, select the type of property you want to list on Booking.com

One apartment — One apartment where guests can book the entire place

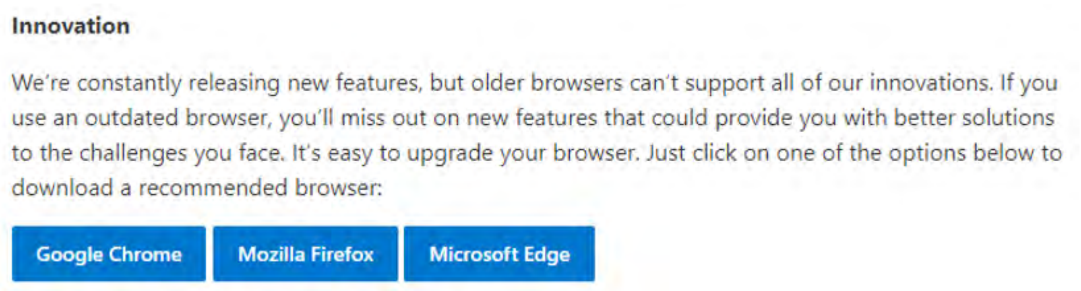
Homes — Properties like apartments, vacation homes, villas, etc.

Hotel, B&Bs & More — Properties like hotels, B&Bs, guest houses, hostels, etc.

Alternative Places — Properties like boats, campgrounds, luxury tents, etc.

[10] Unless otherwise noted, the images presented in this Count were generated for investigative purposes by testing the Accused Instrumentalities on https://www.booking.com/, https://join.booking.com/, https://partner.booking.com/en-us, https://ycs.agoda.com/en-us/kipp/public/home, https://www.opentable.com/, https://support.opentable.com/s/?language=en_US, https://platform.opentable.com/documentation/, and/or other associated websites.

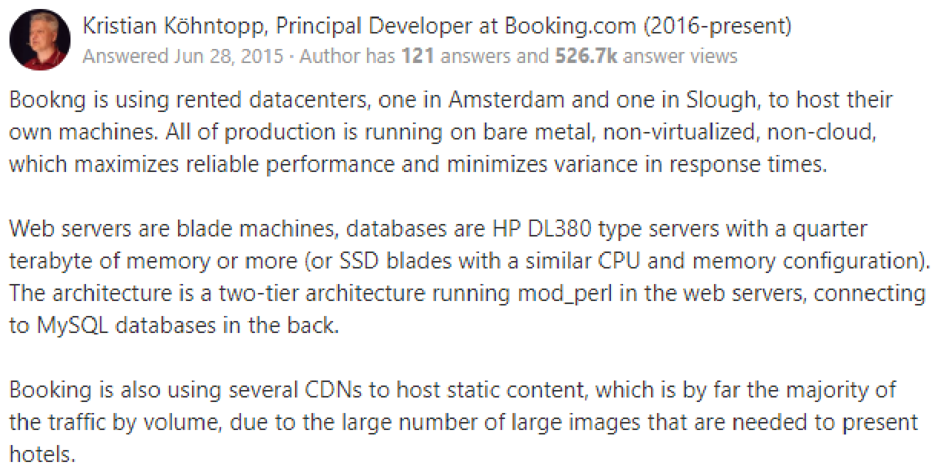
281. The Booking Platform displays content on a display of a device having a player, such as a device-dependent code for a browser engine for a specific browser, or an operating system or application for a particular kind of device. *See, e.g.*, *Shopify Inc. v. Express Mobile, Inc.*, Case No. 1:19-cv-00439-RGA, D. Del., D.I. 137. For example, the Booking Platform displays content through modern browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge.



Innovation

We're constantly releasing new features, but older browsers can't support all of our innovations. If you use an outdated browser, you'll miss out on new features that could provide you with better solutions to the challenges you face. It's easy to upgrade your browser. Just click on one of the options below to download a recommended browser:

282. The Booking Platform includes a non-volatile memory for storing symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network. In particular, the Booking Platform is supported by a MySQL database that serves as a back-end server to support Booking websites.



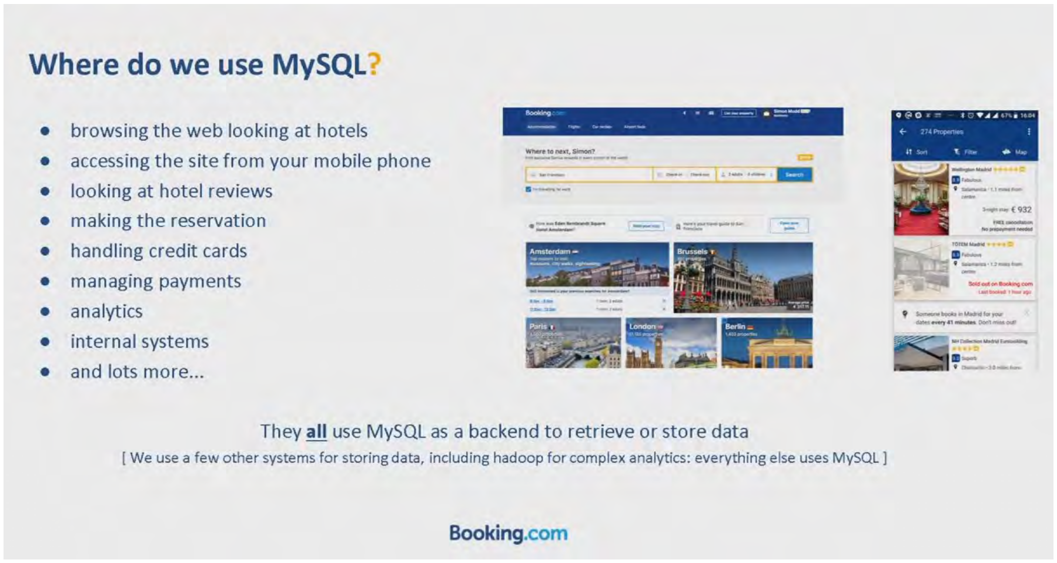
Kristian Köhntopp, Principal Developer at Booking.com (2016-present)
Answered Jun 28, 2015 · Author has **121** answers and **526.7k** answer views

Bookng is using rented datacenters, one in Amsterdam and one in Slough, to host their own machines. All of production is running on bare metal, non-virtualized, non-cloud, which maximizes reliable performance and minimizes variance in response times.

Web servers are blade machines, databases are HP DL380 type servers with a quarter terabyte of memory or more (or SSD blades with a similar CPU and memory configuration). The architecture is a two-tier architecture running mod_perl in the web servers, connecting to MySQL databases in the back.

Booking is also using several CDNs to host static content, which is by far the majority of the traffic by volume, due to the large number of large images that are needed to present hotels.

https://www.quora.com/Who-hosts-the-booking-com-website



Where do we use MySQL?

- browsing the web looking at hotels
- accessing the site from your mobile phone
- looking at hotel reviews
- making the reservation
- handling credit cards
- managing payments
- analytics
- internal systems
- and lots more...

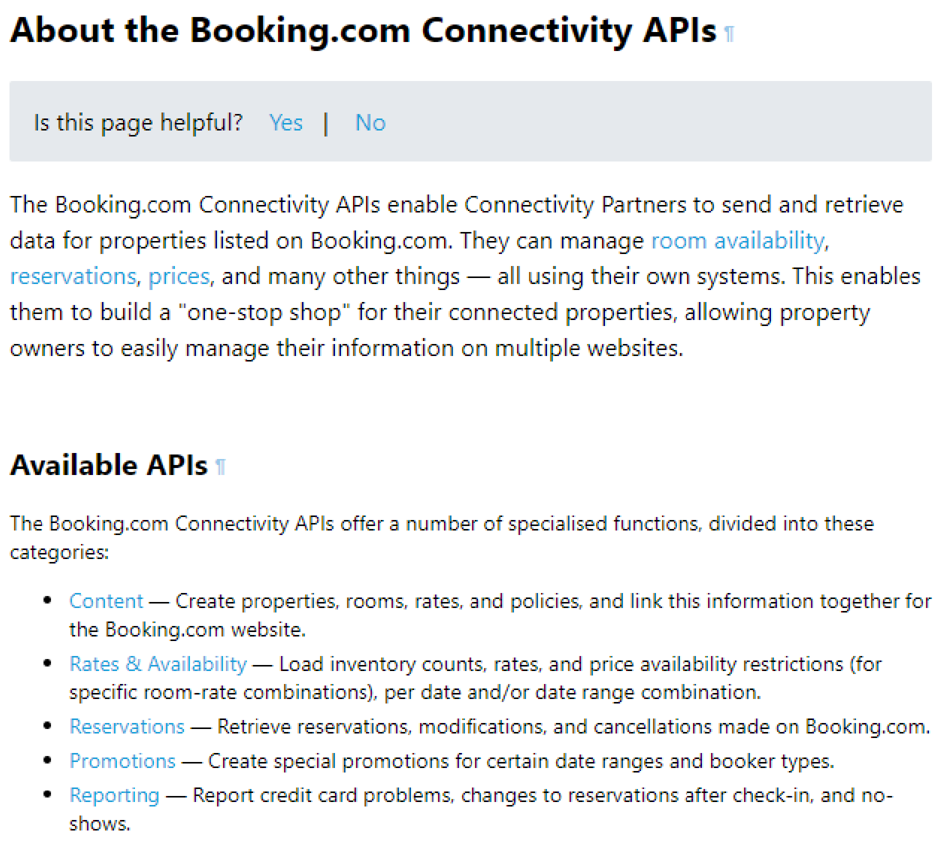
They **all** use MySQL as a backend to retrieve or store data

[We use a few other systems for storing data, including hadoop for complex analytics: everything else uses MySQL]

Booking.com

https://www.youtube.com/watch?v=iNxqZSbaHYQ&feature=youtu.be

283. Databases conventionally use non-volatile computer memories to store data. The computer memory in the Booking Platform thus stores data for the property listings, including symbolic names for evoking inputs and outputs of web services such as Application Programming Interfaces (APIs) that can be integrated on the property listings to provide various kinds of services.



About the Booking.com Connectivity APIs ¶

Is this page helpful? Yes | No

The Booking.com Connectivity APIs enable Connectivity Partners to send and retrieve data for properties listed on Booking.com. They can manage room availability, reservations, prices, and many other things — all using their own systems. This enables them to build a "one-stop shop" for their connected properties, allowing property owners to easily manage their information on multiple websites.

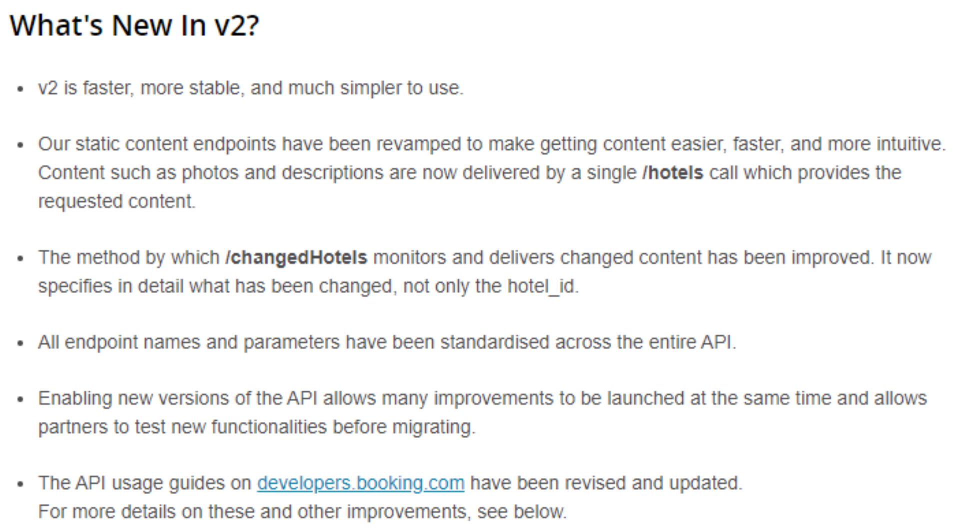
Available APIs ¶

The Booking.com Connectivity APIs offer a number of specialised functions, divided into these categories:

- Content — Create properties, rooms, rates, and policies, and link this information together for the Booking.com website.
- Rates & Availability — Load inventory counts, rates, and price availability restrictions (for specific room-rate combinations), per date and/or date range combination.
- Reservations — Retrieve reservations, modifications, and cancellations made on Booking.com.
- Promotions — Create special promotions for certain date ranges and booker types.
- Reporting — Report credit card problems, changes to reservations after check-in, and no-shows.

https://connect.booking.com/user_guide/site/en-US/

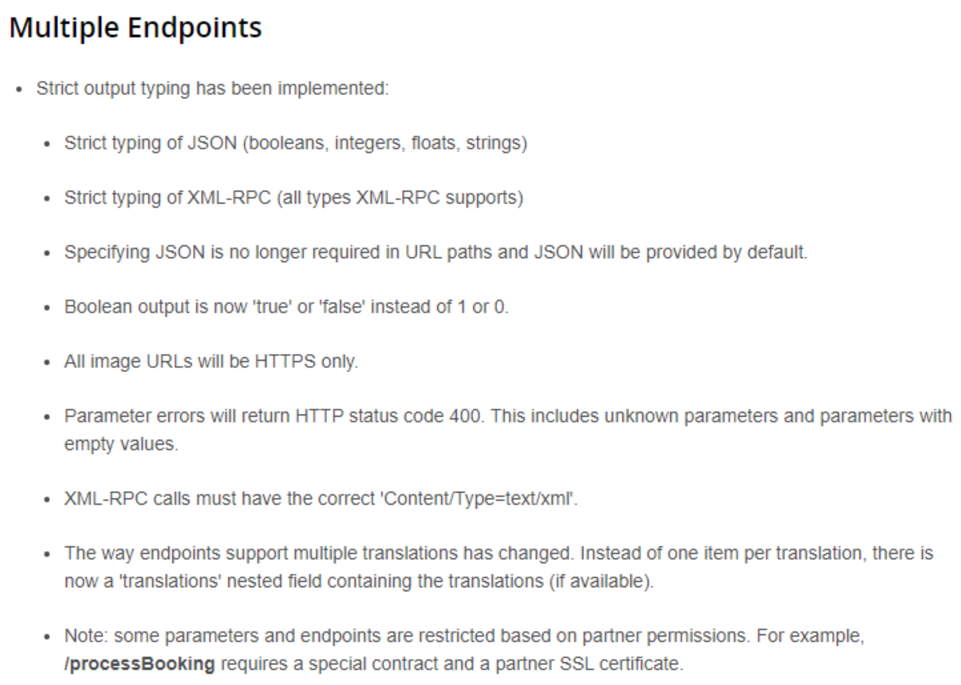
284. In order to store and transmit data for APIs (i.e., web services) such as the Google Maps API, the Booking Platform employs JSON data to evoke web components and communicate inputs and outputs for the APIs. JSON data comprise key/value pairs that are essentially character strings (i.e., symbolic names) with no persistent address or pointer.



What's New In v2?

- v2 is faster, more stable, and much simpler to use.
- Our static content endpoints have been revamped to make getting content easier, faster, and more intuitive. Content such as photos and descriptions are now delivered by a single **/hotels** call which provides the requested content.
- The method by which **/changedHotels** monitors and delivers changed content has been improved. It now specifies in detail what has been changed, not only the hotel_id.
- All endpoint names and parameters have been standardised across the entire API.
- Enabling new versions of the API allows many improvements to be launched at the same time and allows partners to test new functionalities before migrating.
- The API usage guides on developers.booking.com have been revised and updated. For more details on these and other improvements, see below.

https://developers.booking.com/api/commercial/index.html?version=2.5&page_url=migration-guide

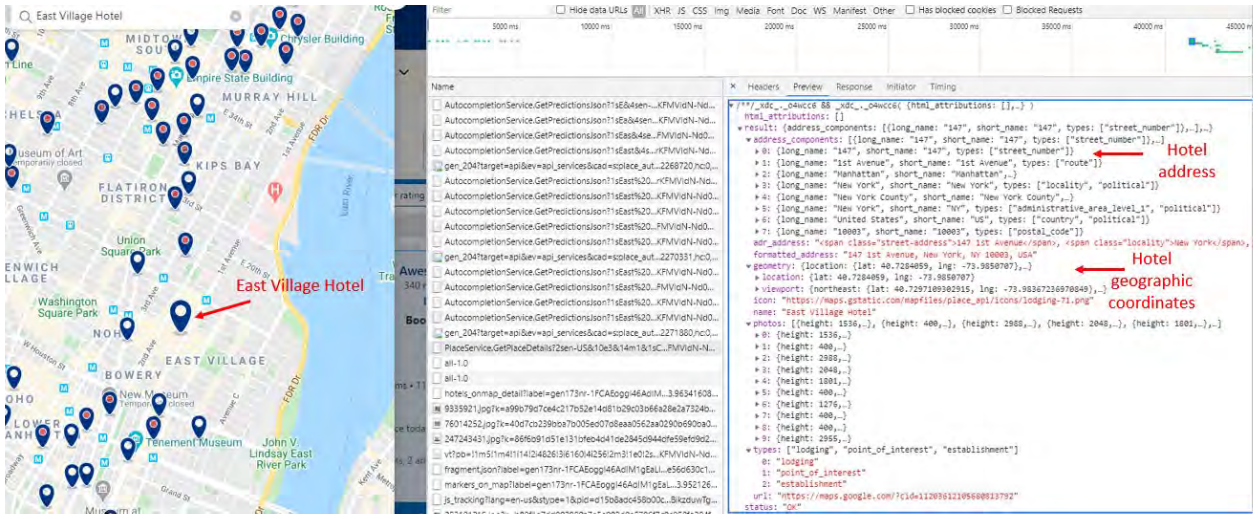


Multiple Endpoints

- Strict output typing has been implemented:
    - Strict typing of JSON (booleans, integers, floats, strings)
    - Strict typing of XML-RPC (all types XML-RPC supports)
    - Specifying JSON is no longer required in URL paths and JSON will be provided by default.
    - Boolean output is now 'true' or 'false' instead of 1 or 0.
    - All image URLs will be HTTPS only.
    - Parameter errors will return HTTP status code 400. This includes unknown parameters and parameters with empty values.
    - XML-RPC calls must have the correct 'Content/Type=text/xml'.
    - The way endpoints support multiple translations has changed. Instead of one item per translation, there is now a 'translations' nested field containing the translations (if available).
    - Note: some parameters and endpoints are restricted based on partner permissions. For example, **/processBooking** requires a special contract and a partner SSL certificate.

https://developers.booking.com/api/commercial/index.html?version=2.5&page_url=migration-guide

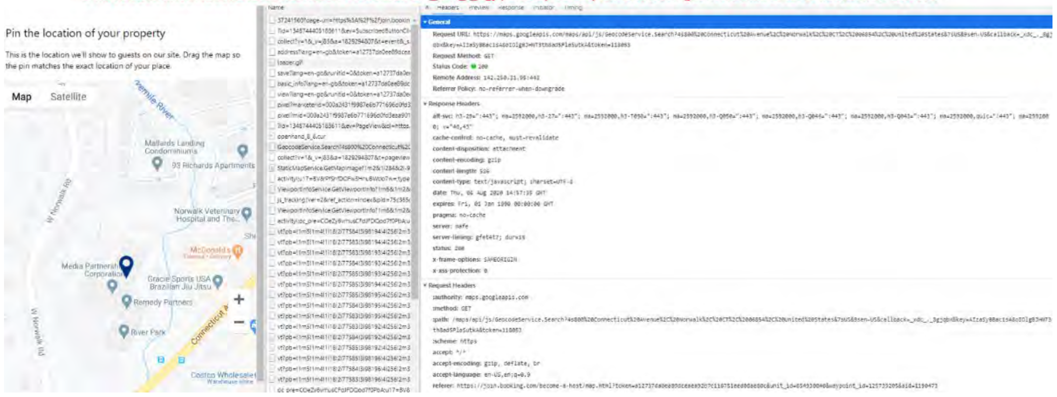
285. Each symbolic name has an associated data format class type corresponding to a subclass of UI objects that support the data format type of the symbolic name, and has the preferred UI object. The example below shows that the symbolic names “address_components” and “formatted_address” are associated with the address for “East Village Hotel,” and the symbolic name “geometry” is associated with the geographic coordinates of the “East Village Hotel.” A large pin marker object associated with the address and geographic coordinates of the hotel, is dropped on a Google Maps API map to indicate the location of the “East Village Hotel.” The large pin marker object is a preferred UI object indicating the target hotel, which is distinguishable from pins indicating other nearby hotels and points of interest.



```
/**/_xdc_._o4wcc6 && _xdc_._o4wcc6( {html_attributions: [],…} )
  html_attributions: []
  result: {address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…],…}
    address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…]
      0: {long_name: "147", short_name: "147", types: ["street_number"]}
      1: {long_name: "1st Avenue", short_name: "1st Avenue", types: ["route"]}
      2: {long_name: "Manhattan", short_name: "Manhattan",…}
      3: {long_name: "New York", short_name: "New York", types: ["locality", "political"]}
      4: {long_name: "New York County", short_name: "New York County",…}
      5: {long_name: "New York", short_name: "NY", types: ["administrative_area_level_1", "political"]}
      6: {long_name: "United States", short_name: "US", types: ["country", "political"]}
      7: {long_name: "10003", short_name: "10003", types: ["postal_code"]}
    adr_address: "<span class="street-address">147 1st Avenue</span>, <span class="locality">New York</span>,
    formatted_address: "147 1st Avenue, New York, NY 10003, USA"
    geometry: {location: {lat: 40.7284059, lng: -73.9850707},…}
      location: {lat: 40.7284059, lng: -73.9850707}
      viewport: {northeast: {lat: 40.7297109302915, lng: -73.98367236970849},…}
    icon: "https://maps.gstatic.com/mapfiles/place_api/icons/lodging-71.png"
    name: "East Village Hotel"
    photos: [{height: 1536,…}, {height: 400,…}, {height: 2988,…}, {height: 2048,…}, {height: 1801,…},…]
      0: {height: 1536,…}
      1: {height: 400,…}
      2: {height: 2988,…}
      3: {height: 2048,…}
      4: {height: 1801,…}
      5: {height: 400,…}
      6: {height: 1276,…}
      7: {height: 400,…}
      8: {height: 400,…}
      9: {height: 2955,…}
    types: ["lodging", "point_of_interest", "establishment"]
      0: "lodging"
      1: "point_of_interest"
      2: "establishment"
    url: "https://maps.google.com/?cid=11203612105680813792"
  status: "OK"
```

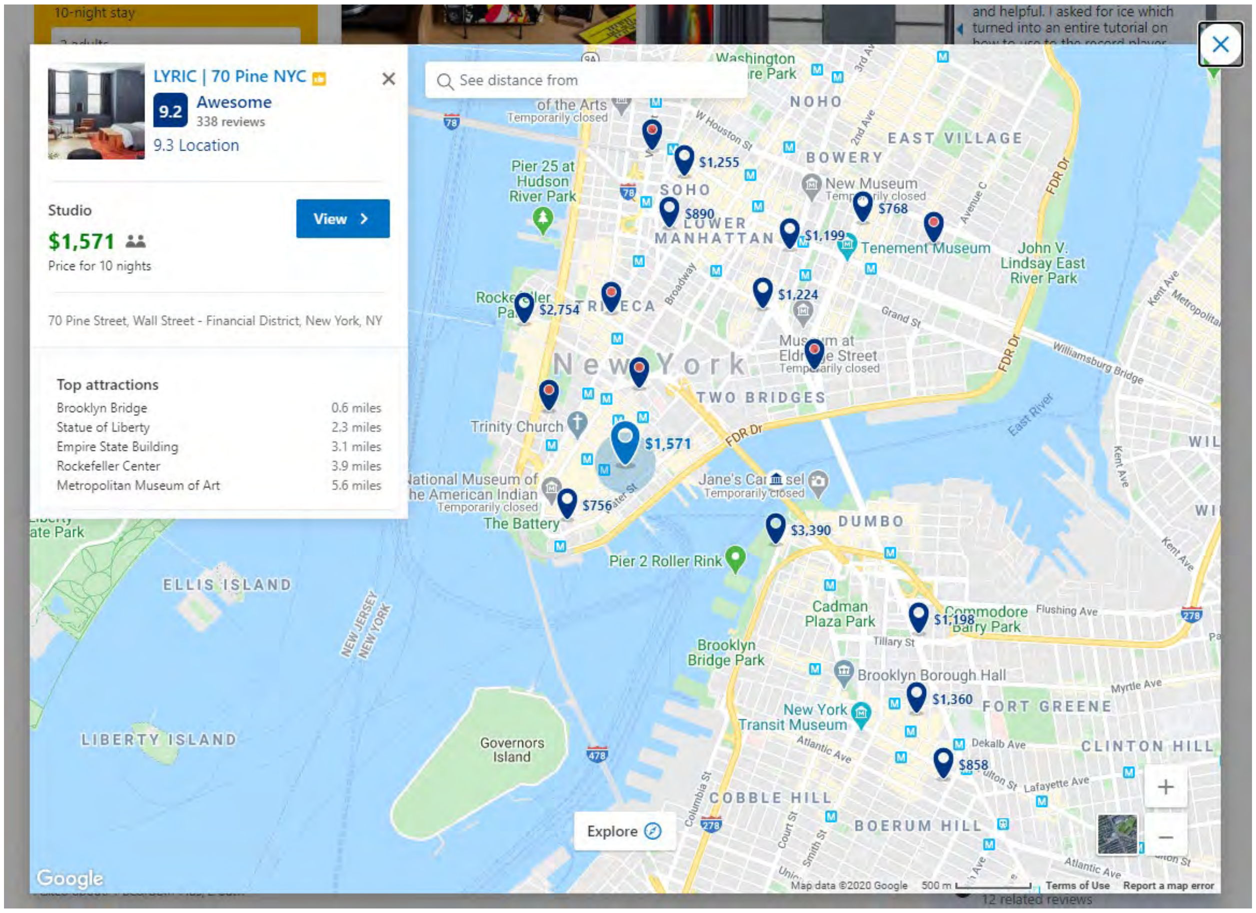
286. On information and belief, the computer memory where data for the property listings are stored also includes addresses for the web services where the input symbolic names and output symbolic names can be sent to and received from. For example, the screenshot below shows a request URL including an address of the Google Maps web service when a user inputs an address for Geocoding service in the Google Maps API. In particular, the user-input property address "800 Connecticut Avenue. Norwalk, CT 06854" is embedded in the request URL and sent to the Google Maps API via an HTTP GET request. The Geocoding service in the Google Maps API then responds with geographic coordinates for the user-input address in the format of JSON data and places a marker on the map indicating the property's location. The geographic coordinates and the user input address are stored in the Booking's database for further use.

Request URL:
https://maps.googleapis.com/maps/api/js/GeocodeService.Search?4s800%20Connecticut%20Avenue%2C%20Norwalk%2C%20CT%2C%2006854%2C%20United%20States&7sUS&9sen-US&callback=_xdc_._8gjqbx&key=AIzaSyBBac1sA8oIOlgBJHNT3th8ad5PlaSutkA&token=118053


Pin the location of your property
This is the location we'll show to guests on our site. Drag the map so the pin matches the exact location of your place.
Map
Satellite

```
General
  Request URL: https://maps.googleapis.com/maps/api/js/GeocodeService.Search?4s800%20Connecticut%20Avenue%2C%20Norwalk%2C%20CT%2C%2006854%2C%20United%20States&7sUS&9sen-US&callback=_xdc_._8gjqbx&key=AIzaSyBBac1sA8oIOlgBJHNT3th8ad5PlaSutkA&token=118053
  Request Method: GET
  Status Code: 200
  Remote Address: 142.250.31.95:443
  Referrer Policy: no-referrer-when-downgrade
Response Headers
  alt-svc: h3-29=":443"; ma=2592000,h3-27=":443"; ma=2592000,h3-T050=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"
  cache-control: no-cache, must-revalidate
  content-disposition: attachment
  content-encoding: gzip
  content-length: 526
  content-type: text/javascript; charset=UTF-8
  date: Thu, 06 Aug 2020 14:57:35 GMT
  expires: Fri, 01 Jan 1990 00:00:00 GMT
  pragma: no-cache
  server: mafe
  server-timing: gfet4t7; dur=15
  status: 200
  x-frame-options: SAMEORIGIN
  x-xss-protection: 0
Request Headers
  :authority: maps.googleapis.com
  :method: GET
  :path: /maps/api/js/GeocodeService.Search?4s800%20Connecticut%20Avenue%2C%20Norwalk%2C%20CT%2C%2006854%2C%20United%20States&7sUS&9sen-US&callback=_xdc_._8gjqbx&key=AIzaSyBBac1sA8oIOlgBJHNT3th8ad5PlaSutkA&token=118053
  :scheme: https
  accept: */*
  accept-encoding: gzip, deflate, br
  accept-language: en-US,en;q=0.9
  referer: [illegible]
```

287. The Booking Platform defines a UI object for presentation on the display, where the UI object corresponds to a web component included in the computer memory, and the web component is selected from a group consisting of an input or output for an API described above (i.e., a web service). By way of example, when the Google Maps API is embedded onto a Booking website, UI objects for input and output web components such as a map image, a pin indicating the property's location, zoom-in and zoom-out buttons, and other nearby attractions and points of interest, are defined for the Google Maps web service. Each of these UI objects is automatically selected by the Booking Platform as a preferred UI object.


LYRIC | 70 Pine NYC
9.2 Awesome
338 reviews
9.3 Location

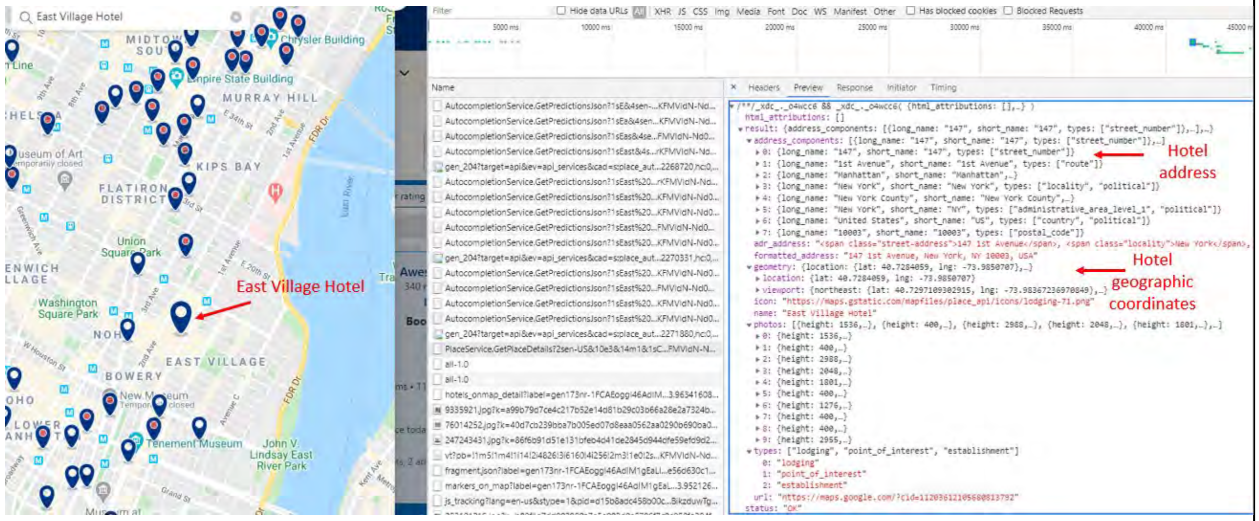
Studio
$1,571
Price for 10 nights
View

70 Pine Street, Wall Street - Financial District, New York, NY

Top attractions

Brooklyn Bridge	0.6 miles
Statue of Liberty	2.3 miles
Empire State Building	3.1 miles
Rockefeller Center	3.9 miles
Metropolitan Museum of Art	5.6 miles

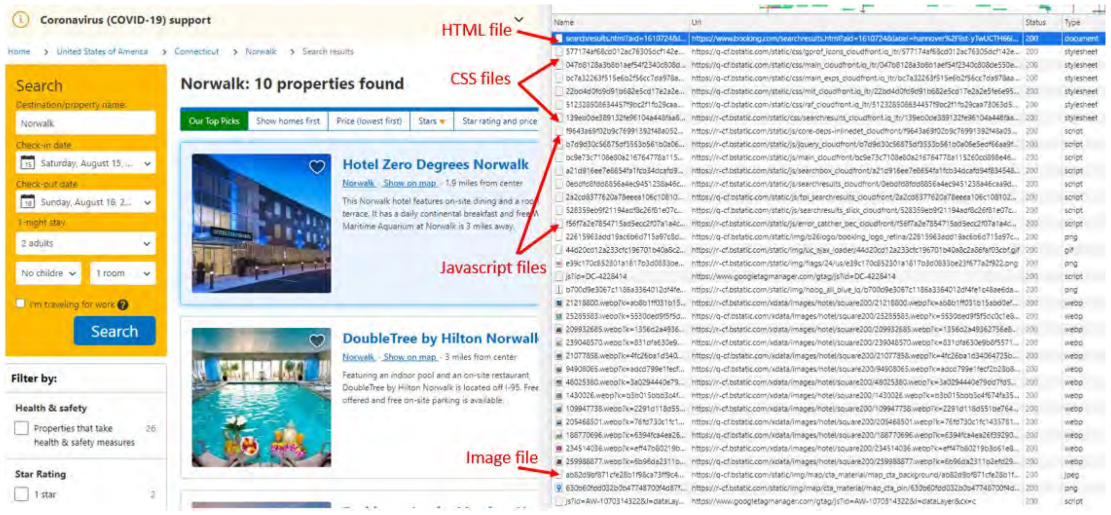
288. When a UI object is so defined, a symbolic name (e.g., JSON data) from the web component corresponding to the defined UI object is selected and associated with the defined UI object. The symbolic name is only available to UI objects that support the defined data format associated with that symbolic name. The example below demonstrates how a user can search for "East Village Hotel" in the Google Maps API on a property listing. When the hotel name "East Village Hotel" in entered into the search box of the Google Maps API, the Google Maps web service responds by sending a JSON dataset to the Booking Platform. In particular, the symbolic names "address_components" and "formatted_address" associated with the hotel address, and the symbolic name "geometry" associated with the geographic coordinates of the hotel are used to display the location of the property on the Google Maps API. In the example below, a large pin marker object associated with the address and geographic coordinates of the hotel is dropped on the map to indicate the location of the "East Village Hotel." The large pin marker object is a preferred UI object to represent the target hotel, which is distinguishable from other pins indicating nearby hotels and points of interest.



```
/**/_xdc_._o4wcc6 && _xdc_._o4wcc6( {html_attributions: [],…} )
  html_attributions: []
  result: {address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…],…}
    address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…]
      0: {long_name: "147", short_name: "147", types: ["street_number"]}
      1: {long_name: "1st Avenue", short_name: "1st Avenue", types: ["route"]}
      2: {long_name: "Manhattan", short_name: "Manhattan",…}
      3: {long_name: "New York", short_name: "New York", types: ["locality", "political"]}
      4: {long_name: "New York County", short_name: "New York County",…}
      5: {long_name: "New York", short_name: "NY", types: ["administrative_area_level_1", "political"]}
      6: {long_name: "United States", short_name: "US", types: ["country", "political"]}
      7: {long_name: "10003", short_name: "10003", types: ["postal_code"]}
    adr_address: "<span class="street-address">147 1st Avenue</span>, <span class="locality">New York</span>,
    formatted_address: "147 1st Avenue, New York, NY 10003, USA"
    geometry: {location: {lat: 40.7284059, lng: -73.9850707},…}
      location: {lat: 40.7284059, lng: -73.9850707}
      viewport: {northeast: {lat: 40.7297109302915, lng: -73.98367236970849},…}
    icon: "https://maps.gstatic.com/mapfiles/place_api/icons/lodging-71.png"
    name: "East Village Hotel"
    photos: [{height: 1536,…}, {height: 400,…}, {height: 2988,…}, {height: 2048,…}, {height: 1801,…},…]
      0: {height: 1536,…}
      1: {height: 400,…}
      2: {height: 2988,…}
      3: {height: 2048,…}
      4: {height: 1801,…}
      5: {height: 400,…}
      6: {height: 1276,…}
      7: {height: 400,…}
      8: {height: 400,…}
      9: {height: 2955,…}
    types: ["lodging", "point_of_interest", "establishment"]
      0: "lodging"
      1: "point_of_interest"
      2: "establishment"
    url: "https://maps.google.com/?cid=11203612105680813792"
  status: "OK"
```

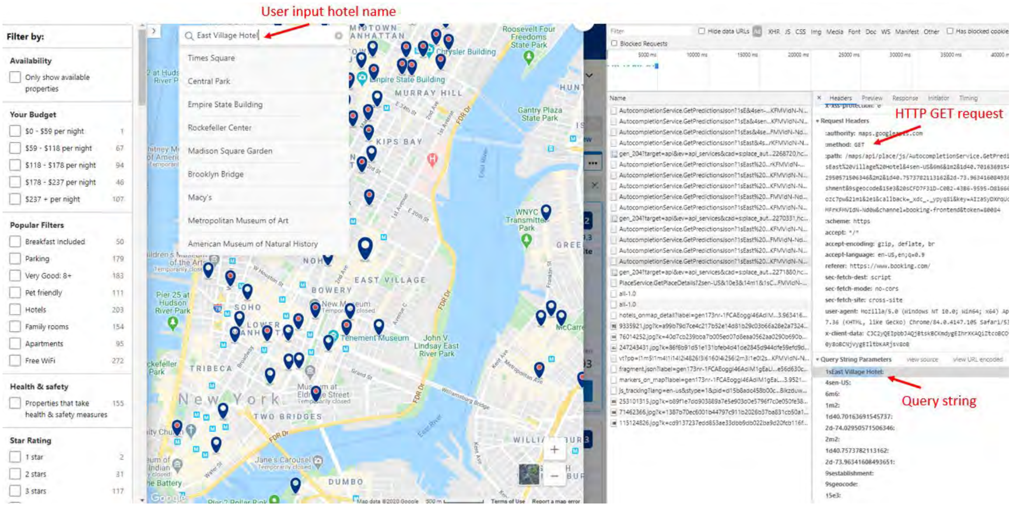
289. The Booking Platform stores information representative of the defined UI object and related settings (e.g., JSON data, along with standard HTML, CSS, and Javascript code) in the MySQL database described above.

290. To generate the property listing website, the Booking Platform retrieves the information for the UI objects stored in the MySQL database, and builds an application consisting of one or more web pages views from at least a portion of the database using the player. The player (e.g., a browser engine or operating system) utilizes the data stored in the database to generate for the display at least a portion of a web page. In the example below, the Booking Platform converts HTML, CSS, Javascript, image, and other files into an active property website.


Coronavirus (COVID-19) support
Home > United States of America > Connecticut > Norwalk > Search results
Search
Norwalk: 10 properties found
Our Top Picks | Show homes first | Price (lowest first) | Stars | Star rating and price
Hotel Zero Degrees Norwalk
DoubleTree by Hilton Norwalk
Filter by:
Health & safety
Star Rating
HTML file
CSS files
Javascript files
Image file
Name
Url
Status
Type

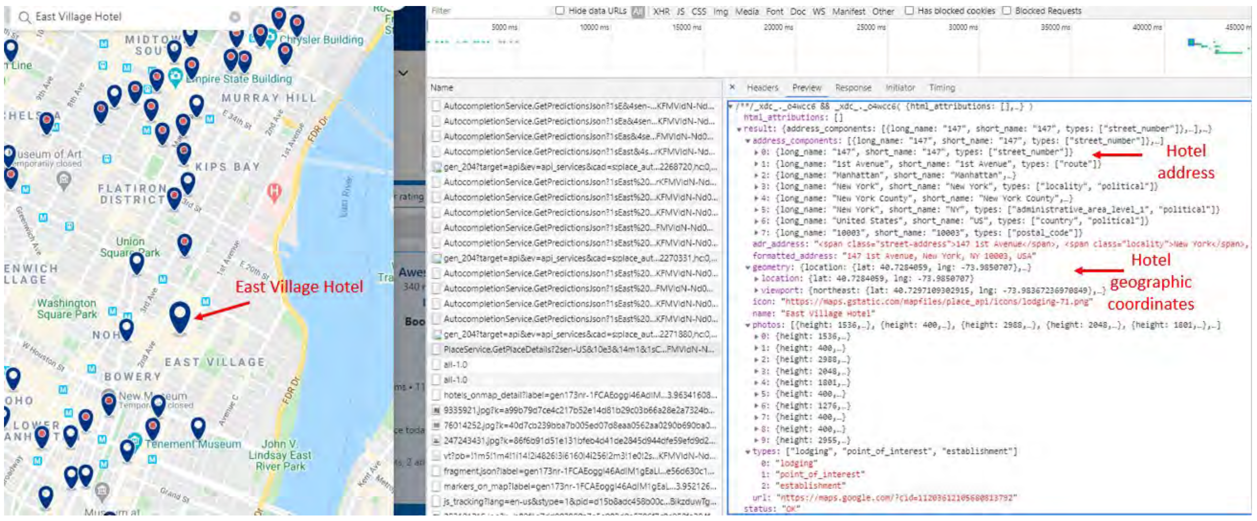
291. When the application and player are executed on the device, an input value can be provided by a user. The device then provides the input value and a corresponding input symbolic name to the web service, which in turn generates an output value and a corresponding symbolic name for the output. By way of example, when the Google Maps API accepts an input value (e.g., a hotel name, a zoom-in click, or a click on a nearby hotel) from a user into the API, a corresponding input symbolic name (e.g., a query string or JSON data for the input) is transmitted to the Google Maps web service through an HTTP request protocol, such as a GET method call. The example below demonstrates how a user can search for "East Village Hotel" in the Google Maps API on a property listing. When the user types "East Village Hotel," a query string with

parameters including the symbolic name “1s” and associated value “East Village Hotel” is sent to the Google Maps web service to initiate an autocompletion service with an HTTP GET request.



```
▾ Request Headers
  :authority: maps.googleapis.com
  :method: GET
  :path: /maps/api/place/js/AutocompletionService.GetPredi
  sEast%20Village%20Hotel&4sen-US&6m6&1m2&1d40.701636915
  29505715063462m2&1d40.7573782113162&2d-73.963416084936
  shment&9sgeocode&15e3&20sCFD7F31D-C0B2-43B6-9595-D8166
  ozC7pu&21m1&2e1&callback=_xdc_._ypyq8i&key=AIzaSyDXrqU
  HFrKFHVIdN-Nd0w&channel=booking-frontend&token=80084
  :scheme: https
  accept: */*
  accept-encoding: gzip, deflate, br
  accept-language: en-US,en;q=0.9
  referer: https://www.booking.com/
  sec-fetch-dest: script
  sec-fetch-mode: no-cors
  sec-fetch-site: cross-site
  user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) Ap
  7.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/5
  x-client-data: CJC2yQEIpbbJAQjBtskBCKmdygEIhrXKAQiZtcoBCO
  0y8oBCNjVygEIltbKARjsv8oB
▾ Query String Parameters      view source      view URL encoded
  1sEast Village Hotel:
  4sen-US:
  6m6:
  1m2:
  1d40.70163691545737:
  2d-74.02950571506346:
  2m2:
  1d40.7573782113162:
  2d-73.96341608493651:
  9sestablishment:
  9sgeocode:
  15e3:
```

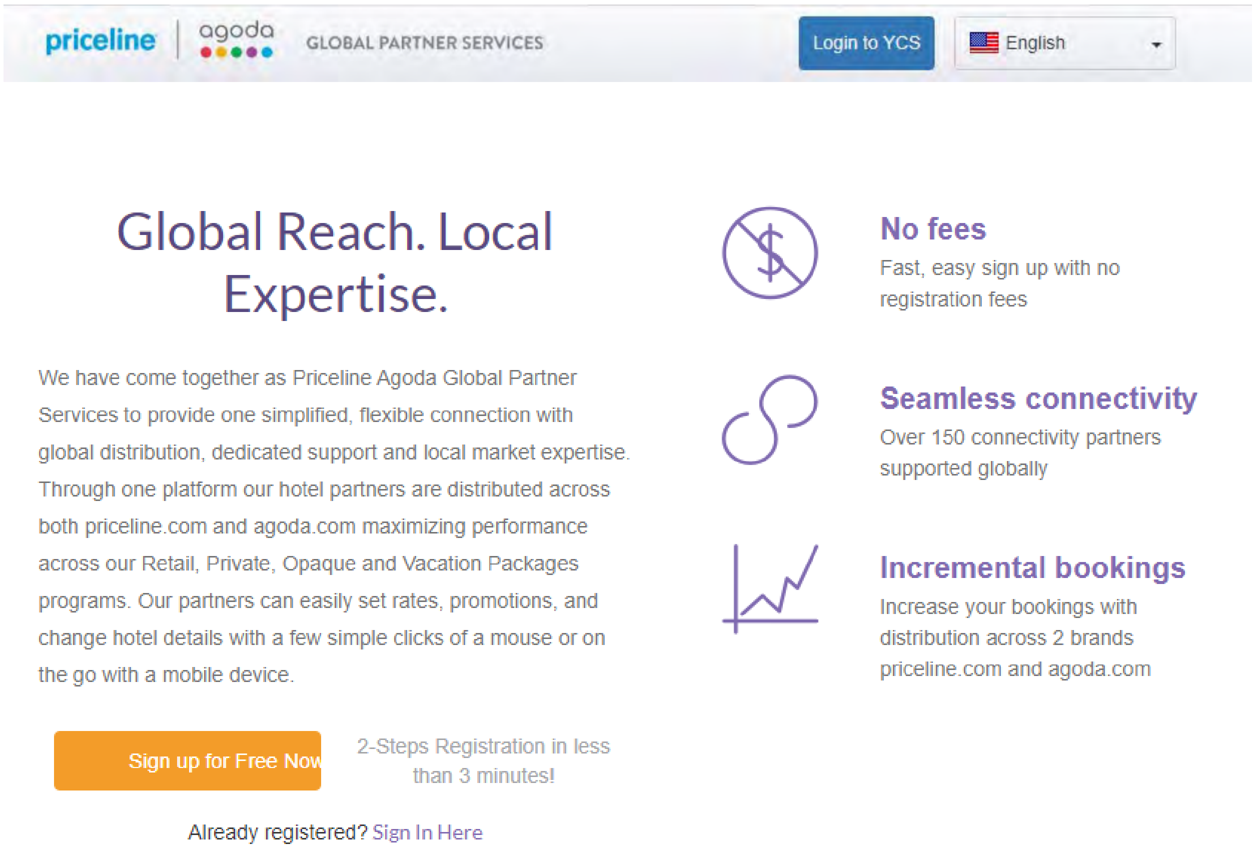
292. The input symbolic name is utilized by the Booking server to generate an output value with an associated output symbolic name (e.g., JSON data for the output). The player then provides instructions to the display of the device to present and display the output in the Google Maps API. For example, when a user searches “East Village Hotel” and selects an autocomplete suggestion from the Google Maps API, the Google Maps web service generates and transmits a corresponding hotel address and geographic coordinates in the format of JSON data to the web browser and displays a pin indicating the location of the selected hotel on the Google Maps API.



```
▾ /**/_xdc_._o4wcc6 && _xdc_._o4wcc6( {html_attributions: [],…} )
   html_attributions: []
 ▾ result: {address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…],…}
   ▾ address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…]
     ▸ 0: {long_name: "147", short_name: "147", types: ["street_number"]}
     ▸ 1: {long_name: "1st Avenue", short_name: "1st Avenue", types: ["route"]}
     ▸ 2: {long_name: "Manhattan", short_name: "Manhattan",…}
     ▸ 3: {long_name: "New York", short_name: "New York", types: ["locality", "political"]}
     ▸ 4: {long_name: "New York County", short_name: "New York County",…}
     ▸ 5: {long_name: "New York", short_name: "NY", types: ["administrative_area_level_1", "political"]}
     ▸ 6: {long_name: "United States", short_name: "US", types: ["country", "political"]}
     ▸ 7: {long_name: "10003", short_name: "10003", types: ["postal_code"]}
     adr_address: "<span class="street-address">147 1st Avenue</span>, <span class="locality">New York</span>,
     formatted_address: "147 1st Avenue, New York, NY 10003, USA"
   ▾ geometry: {location: {lat: 40.7284059, lng: -73.9850707},…}
     ▸ location: {lat: 40.7284059, lng: -73.9850707}
     ▸ viewport: {northeast: {lat: 40.7297109302915, lng: -73.98367236970849},…}
     icon: "https://maps.gstatic.com/mapfiles/place_api/icons/lodging-71.png"
     name: "East Village Hotel"
   ▾ photos: [{height: 1536,…}, {height: 400,…}, {height: 2988,…}, {height: 2048,…}, {height: 1801,…},…]
     ▸ 0: {height: 1536,…}
     ▸ 1: {height: 400,…}
     ▸ 2: {height: 2988,…}
     ▸ 3: {height: 2048,…}
     ▸ 4: {height: 1801,…}
     ▸ 5: {height: 400,…}
     ▸ 6: {height: 1276,…}
     ▸ 7: {height: 400,…}
     ▸ 8: {height: 400,…}
     ▸ 9: {height: 2955,…}
   ▾ types: ["lodging", "point_of_interest", "establishment"]
       0: "lodging"
       1: "point_of_interest"
       2: "establishment"
     url: "https://maps.google.com/?cid=11203612105680813792"
   status: "OK"
```

The YCS Platform:

293. The YCS infringers infringe at least claim 15 of the '044 patent through a combination of features in the YCS Platform that collectively practice each claimed limitation of claim 15. By way of example, the YCS infringers provide the YCS Platform for creating property listing websites for hotels and other lodgings where travelers can view and book accommodations.



Global Reach. Local Expertise.

We have come together as Priceline Agoda Global Partner Services to provide one simplified, flexible connection with global distribution, dedicated support and local market expertise. Through one platform our hotel partners are distributed across both priceline.com and agoda.com maximizing performance across our Retail, Private, Opaque and Vacation Packages programs. Our partners can easily set rates, promotions, and change hotel details with a few simple clicks of a mouse or on the go with a mobile device.

Sign up for Free Now

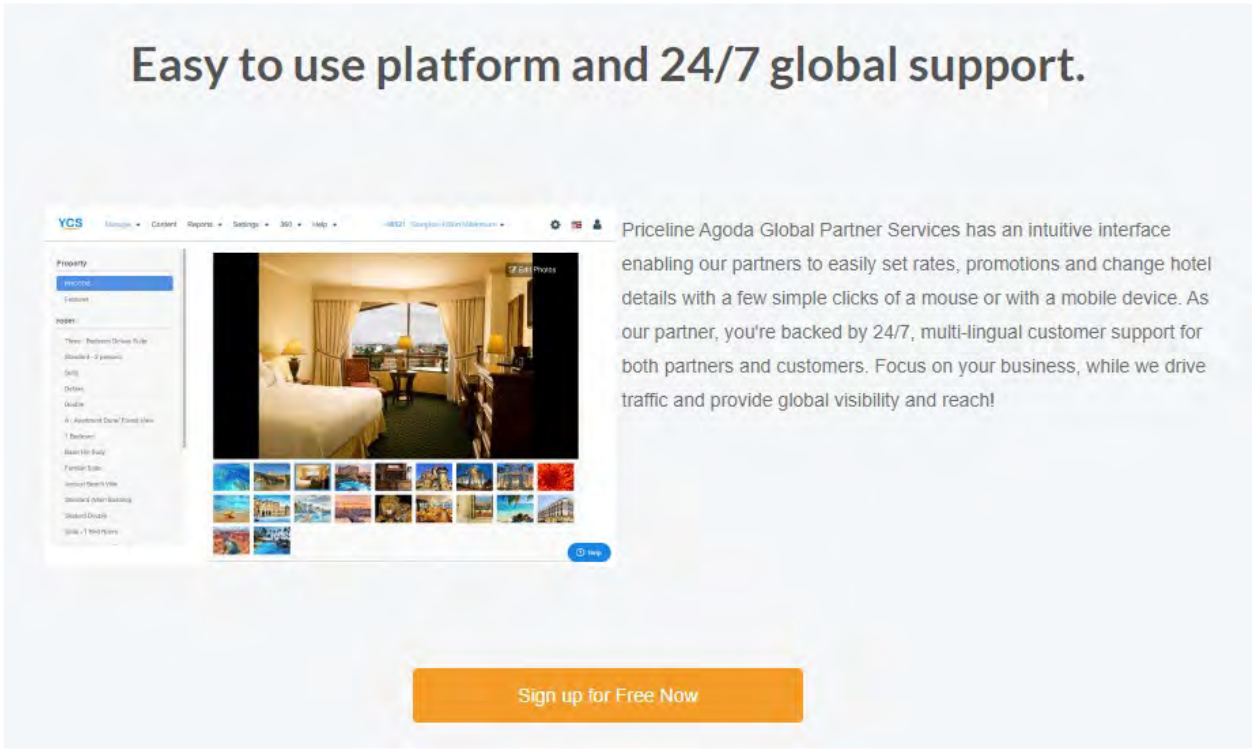
2-Steps Registration in less than 3 minutes!

Already registered? Sign In Here

No fees
Fast, easy sign up with no registration fees

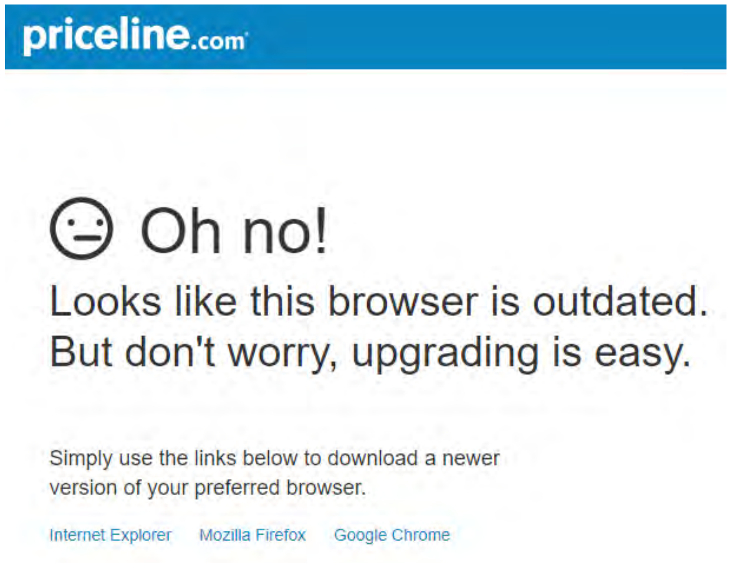
Seamless connectivity
Over 150 connectivity partners supported globally

Incremental bookings
Increase your bookings with distribution across 2 brands priceline.com and agoda.com


Easy to use platform and 24/7 global support.

Priceline Agoda Global Partner Services has an intuitive interface enabling our partners to easily set rates, promotions and change hotel details with a few simple clicks of a mouse or with a mobile device. As our partner, you're backed by 24/7, multi-lingual customer support for both partners and customers. Focus on your business, while we drive traffic and provide global visibility and reach!

294. The YCS Platform provides a platform for displaying content on a display of a device having a player, such as a device-dependent code for a browser engine for a specific browser, or an operating system or application for a particular kind of device. *See, e.g.*, *Shopify Inc. v. Express Mobile, Inc.*, Case No. 1:19-cv-00439-RGA, D. Del., D.I. 137. For example, Priceline property listings are displayed through modern browsers such as Internet Explorer, Mozilla Firefox, and Google Chrome.


priceline.com

Oh no!

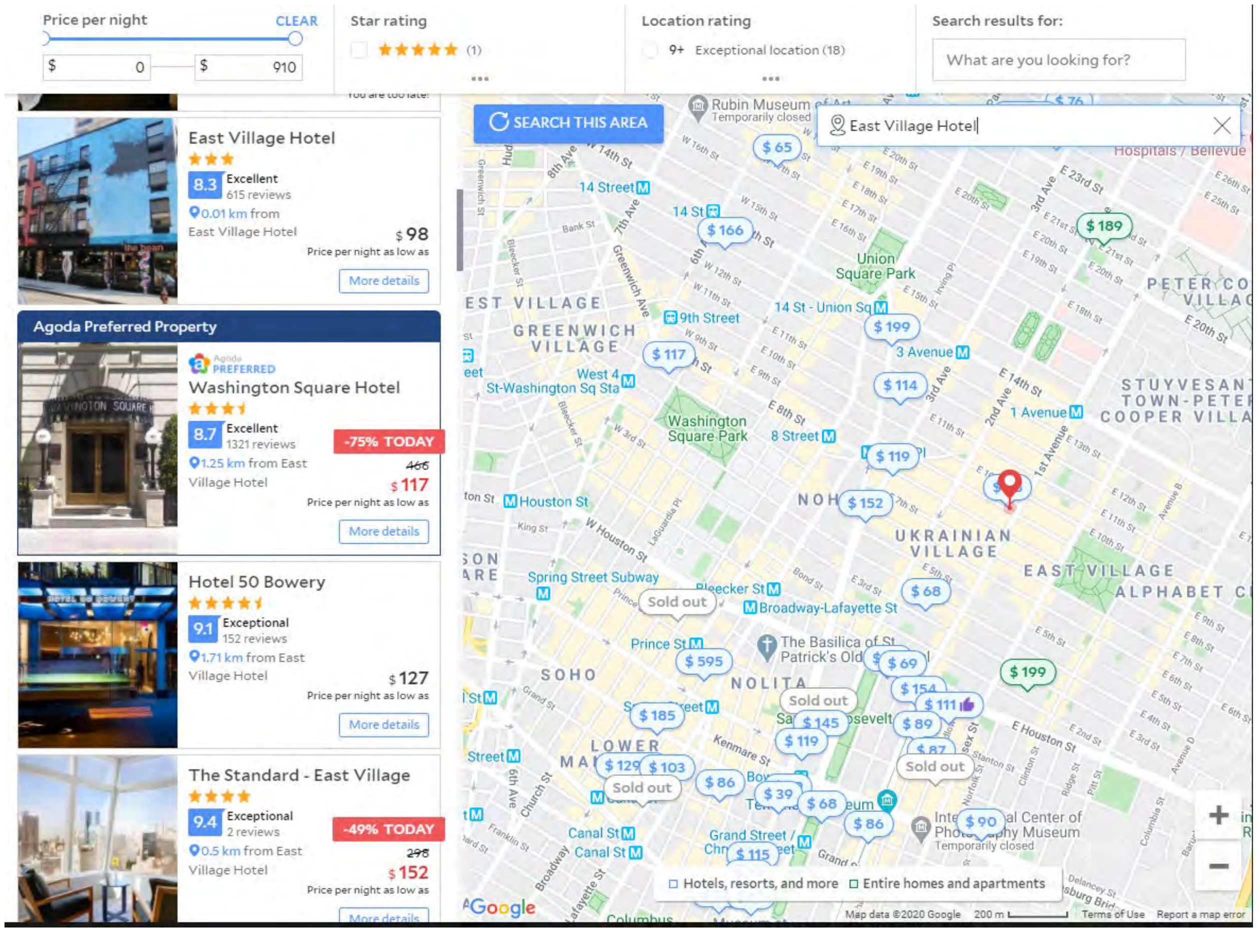
Looks like this browser is outdated.
But don't worry, upgrading is easy.

Simply use the links below to download a newer version of your preferred browser.

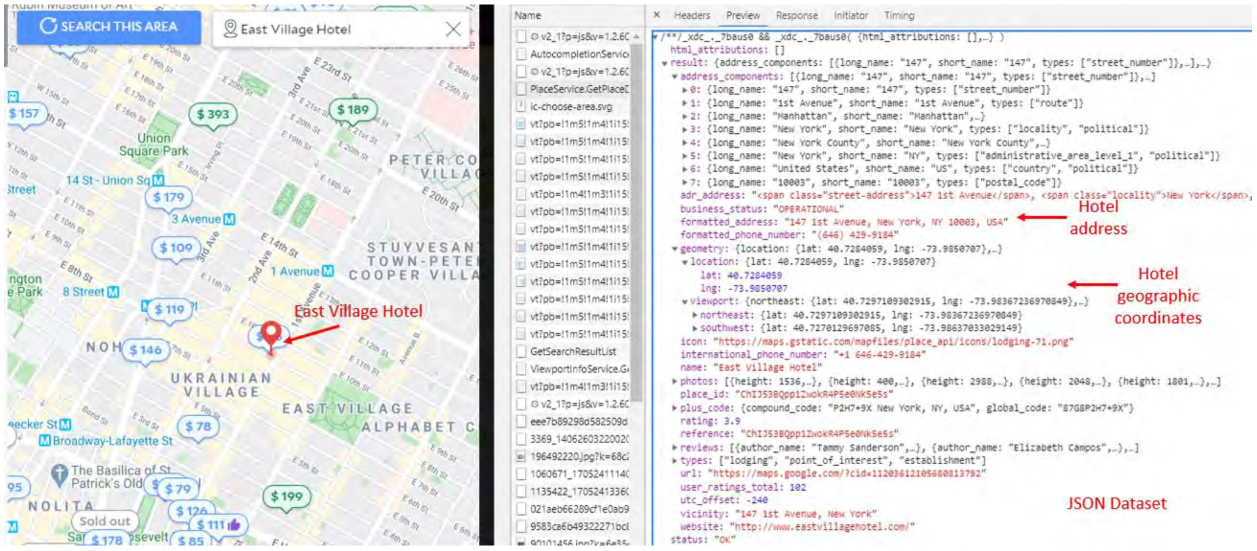
Internet Explorer Mozilla Firefox Google Chrome

https://www.priceline.com/static-pages/browser-upgrade.html

295. On information and belief, the YCS Platform includes a non-volatile computer memory for storing symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network. The YCS Platform is supported by a server database, and databases conventionally use computer memories to store data. On information and belief, a non-volatile computer memory in the YCS Platform stores data for the property listings, including symbolic names for evoking inputs and outputs of Application Programming Interfaces (APIs), i.e., web services, such the Google Maps API.

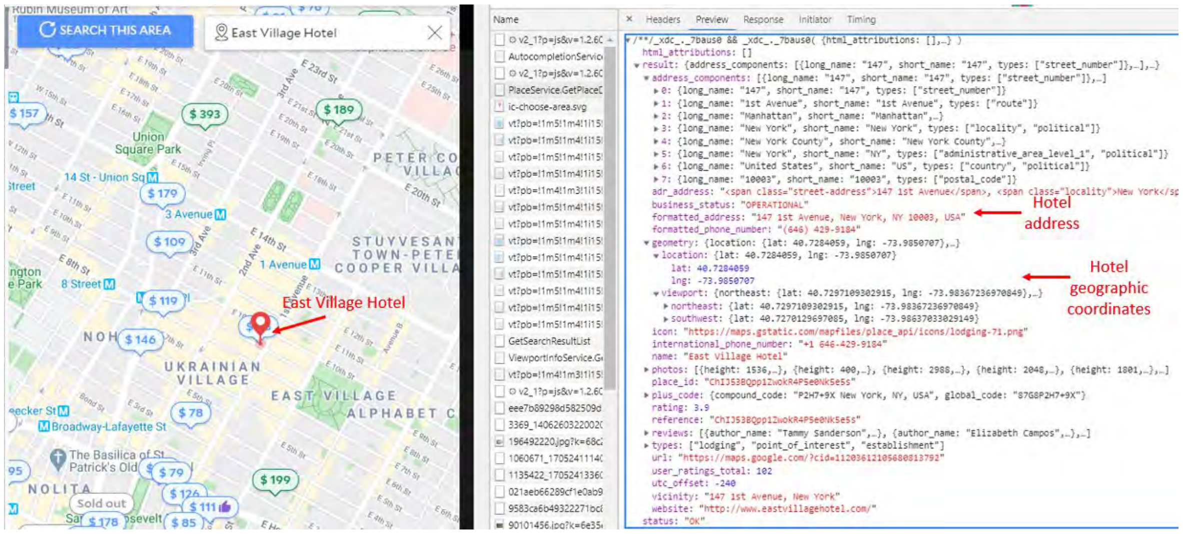

Price per night CLEAR
$ 0
$ 910
Star rating
Location rating
9+ Exceptional location (18)
Search results for:
What are you looking for?
East Village Hotel
8.3 Excellent 615 reviews
0.01 km from East Village Hotel
$98 Price per night as low as
More details
Agoda Preferred Property
Washington Square Hotel
8.7 Excellent 1321 reviews
-75% TODAY
1.25 km from East Village Hotel
466 $117 Price per night as low as
More details
Hotel 50 Bowery
9.1 Exceptional 152 reviews
1.71 km from East Village Hotel
$127 Price per night as low as
More details
The Standard - East Village
9.4 Exceptional 2 reviews
-49% TODAY
0.5 km from East Village Hotel
298 $152 Price per night as low as
SEARCH THIS AREA
East Village Hotel
GREENWICH VILLAGE
Washington Square Park
Union Square Park
UKRAINIAN VILLAGE
EAST VILLAGE
SOHO
NOLITA
LOWER
STUYVESANT TOWN-PETER COOPER VILLAGE
Sold out
Hotels, resorts, and more
Entire homes and apartments
Google
Map data ©2020 Google

296. In order to store and transmit data for the APIs, the YCS Platform employs JSON data to evoke web components and communicate inputs and outputs for the APIs. JSON data comprise key/value pairs that are essentially symbolic names constituting character strings with no persistent address or pointer. In the example below, a JSON dataset is used to store a hotel search result when a user enters "East Village Hotel" into the Google Maps API embedded on an Agoda website. The hotel search result consists of symbolic names and associated values that can be used to provide hotel information, such as the hotel name, address, geographic coordinates, photos, website, and contact numbers.



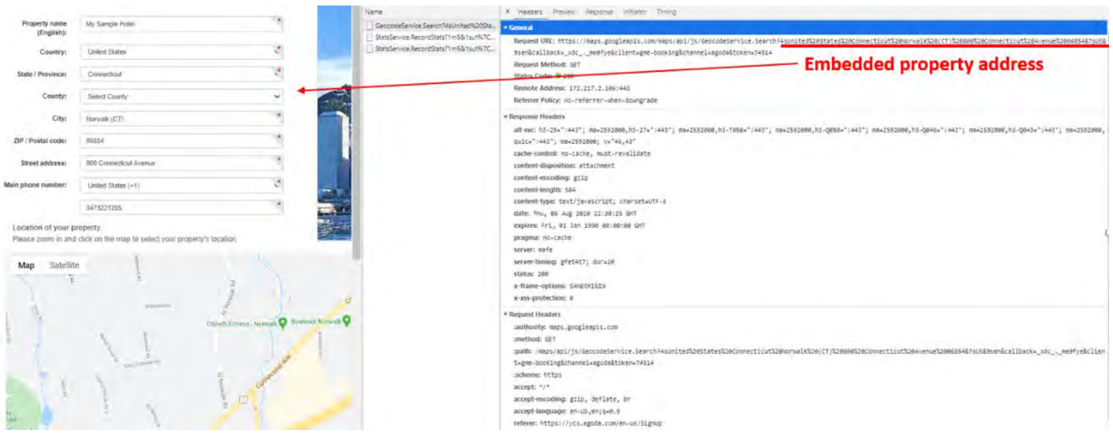
```
/**/_xdc_._7baus0 && _xdc_._7baus0( {html_attributions: [],…} )
  html_attributions: []
  result: {address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…],…}
    address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…]
      0: {long_name: "147", short_name: "147", types: ["street_number"]}
      1: {long_name: "1st Avenue", short_name: "1st Avenue", types: ["route"]}
      2: {long_name: "Manhattan", short_name: "Manhattan",…}
      3: {long_name: "New York", short_name: "New York", types: ["locality", "political"]}
      4: {long_name: "New York County", short_name: "New York County",…}
      5: {long_name: "New York", short_name: "NY", types: ["administrative_area_level_1", "political"]}
      6: {long_name: "United States", short_name: "US", types: ["country", "political"]}
      7: {long_name: "10003", short_name: "10003", types: ["postal_code"]}
    adr_address: "<span class="street-address">147 1st Avenue</span>, <span class="locality">New York</span>,
    business_status: "OPERATIONAL"
    formatted_address: "147 1st Avenue, New York, NY 10003, USA"
    formatted_phone_number: "(646) 429-9184"
    geometry: {location: {lat: 40.7284059, lng: -73.9850707},…}
      location: {lat: 40.7284059, lng: -73.9850707}
        lat: 40.7284059
        lng: -73.9850707
      viewport: {northeast: {lat: 40.7297109302915, lng: -73.98367236970849},…}
        northeast: {lat: 40.7297109302915, lng: -73.98367236970849}
        southwest: {lat: 40.7270129697085, lng: -73.98637033029149}
    icon: "https://maps.gstatic.com/mapfiles/place_api/icons/lodging-71.png"
    international_phone_number: "+1 646-429-9184"
    name: "East Village Hotel"
    photos: [{height: 1536,…}, {height: 400,…}, {height: 2988,…}, {height: 2048,…}, {height: 1801,…},…]
    place_id: "ChIJ53BQpp1ZwokR4P5e0Nk5e5s"
    plus_code: {compound_code: "P2H7+9X New York, NY, USA", global_code: "87G8P2H7+9X"}
    rating: 3.9
    reference: "ChIJ53BQpp1ZwokR4P5e0Nk5e5s"
    reviews: [{author_name: "Tammy Sanderson",…}, {author_name: "Elizabeth Campos",…},…]
    types: ["lodging", "point_of_interest", "establishment"]
    url: "https://maps.google.com/?cid=11203612105680813792"
    user_ratings_total: 102
    utc_offset: -240
    vicinity: "147 1st Avenue, New York"
    website: "http://www.eastvillagehotel.com/"
  status: "OK"
```

297. Each symbolic name has an associated data format class type corresponding to a subclass of UI objects that support the data format type of the symbolic name and has the preferred UI object. In the example below, the symbolic names "address_components" and "formatted_address" are associated with the address for the "East Village Hotel," and the symbolic name "geometry" is associated with the geographic coordinates for the "East Village Hotel." A pin object associated with the address and geographic coordinates of the hotel is dropped on the map to indicate the location of the hotel. The pin object is a preferred UI object indicating the target hotel, which is distinguishable from bubbles indicating the locations and prices of other nearby hotels.



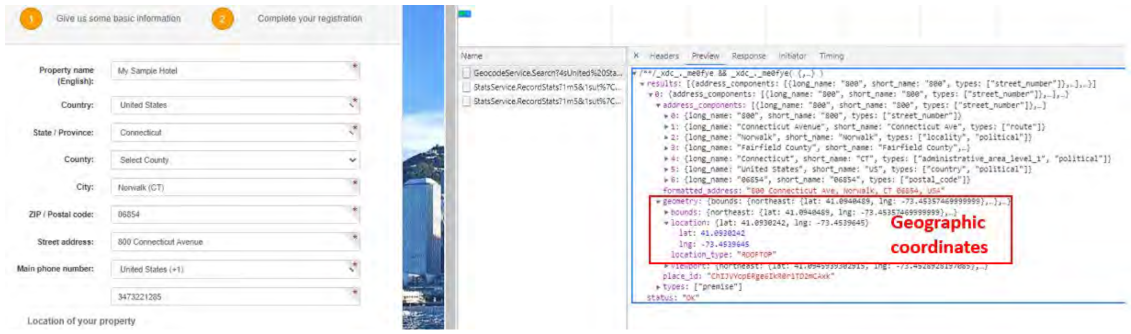
```
/**/_xdc_._7baus0 && _xdc_._7baus0( {html_attributions: [],…} )
  html_attributions: []
  result: {address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…],…}
    address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…]
      0: {long_name: "147", short_name: "147", types: ["street_number"]}
      1: {long_name: "1st Avenue", short_name: "1st Avenue", types: ["route"]}
      2: {long_name: "Manhattan", short_name: "Manhattan",…}
      3: {long_name: "New York", short_name: "New York", types: ["locality", "political"]}
      4: {long_name: "New York County", short_name: "New York County",…}
      5: {long_name: "New York", short_name: "NY", types: ["administrative_area_level_1", "political"]}
      6: {long_name: "United States", short_name: "US", types: ["country", "political"]}
      7: {long_name: "10003", short_name: "10003", types: ["postal_code"]}
    adr_address: "<span class="street-address">147 1st Avenue</span>, <span class="locality">New York</sp
    business_status: "OPERATIONAL"
    formatted_address: "147 1st Avenue, New York, NY 10003, USA"
    formatted_phone_number: "(646) 429-9184"
    geometry: {location: {lat: 40.7284059, lng: -73.9850707},…}
      location: {lat: 40.7284059, lng: -73.9850707}
        lat: 40.7284059
        lng: -73.9850707
      viewport: {northeast: {lat: 40.7297109302915, lng: -73.98367236970849},…}
        northeast: {lat: 40.7297109302915, lng: -73.98367236970849}
        southwest: {lat: 40.7270129697085, lng: -73.98637033029149}
    icon: "https://maps.gstatic.com/mapfiles/place_api/icons/lodging-71.png"
    international_phone_number: "+1 646-429-9184"
    name: "East Village Hotel"
    photos: [{height: 1536,…}, {height: 400,…}, {height: 2988,…}, {height: 2048,…}, {height: 1801,…},…]
    place_id: "ChIJ53BQpp1ZwokR4P5e0Nk5e5s"
    plus_code: {compound_code: "P2H7+9X New York, NY, USA", global_code: "87G8P2H7+9X"}
    rating: 3.9
    reference: "ChIJ53BQpp1ZwokR4P5e0Nk5e5s"
    reviews: [{author_name: "Tammy Sanderson",…}, {author_name: "Elizabeth Campos",…},…]
    types: ["lodging", "point_of_interest", "establishment"]
    url: "https://maps.google.com/?cid=11203612105680813792"
    user_ratings_total: 102
    utc_offset: -240
    vicinity: "147 1st Avenue, New York"
    website: "http://www.eastvillagehotel.com/"
  status: "OK"
```

298. The computer memory where data for the property listings are stored also includes addresses for the web services where the input symbolic names and output symbolic names can be sent to and received from. For example, the screenshot below shows a request URL including an address of the Google Maps web service. In particular, the user-input property address "800 Connecticut Avenue. Norwalk, CT 06854" is embedded in the request URL and sent to the Google Maps API via an HTTP GET request for Geocoding service. The Geocoding service in the Google Maps API then converts the property address to geographic coordinates and sends the geographic coordinates with other property information in the format of JSON data to the Agoda website. The geographic coordinates and the user input address are stored in Booking's database for further use.



Property name (English): My Sample Hotel
Country: United States
State / Province: Connecticut
County: Select County
City: Norwalk (CT)
ZIP / Postal code: 06854
Street address: 800 Connecticut Avenue
Main phone number: United States (+1)
3473221285
Location of your property
Please zoom in and click on the map to select your property's location

```
Name
GeocodeService.Search?4sUnited%20Sta...
StatsService.RecordStats?1m5&1sut%7C...
StatsService.RecordStats?1m5&1sut%7C...

X  Headers  Preview  Response  Initiator  Timing
▾ General
  Request URL: https://maps.googleapis.com/maps/api/js/GeocodeService.Search?4sUnited%20States%20Connecticut%20Norwalk%20(CT)%20800%20Connecticut%20Avenue%2006854&7sUS&
  9sen&callback=_xdc_._me0fye&client=gme-booking&channel=agoda&token=74514
  Request Method: GET
  Status Code: ● 200
  Remote Address: 172.217.2.106:443
  Referrer Policy: no-referrer-when-downgrade
▾ Response Headers
  alt-svc: h3-29=":443"; ma=2592000,h3-27=":443"; ma=2592000,h3-T050=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,
  quic=":443"; ma=2592000; v="46,43"
  cache-control: no-cache, must-revalidate
  content-disposition: attachment
  content-encoding: gzip
  content-length: 584
  content-type: text/javascript; charset=UTF-8
  date: Thu, 06 Aug 2020 22:30:25 GMT
  expires: Fri, 01 Jan 1990 00:00:00 GMT
  pragma: no-cache
  server: mafe
  server-timing: gfet4t7; dur=20
  status: 200
  x-frame-options: SAMEORIGIN
  x-xss-protection: 0
▾ Request Headers
  :authority: maps.googleapis.com
  :method: GET
  :path: /maps/api/js/GeocodeService.Search?4sUnited%20States%20Connecticut%20Norwalk%20(CT)%20800%20Connecticut%20Avenue%2006854&7sUS&9sen&callback=_xdc_._me0fye&clien
  t=gme-booking&channel=agoda&token=74514
  :scheme: https
  accept: */*
  accept-encoding: gzip, deflate, br
  accept-language: en-US,en;q=0.9
  referer: https://ycs.agoda.com/en-us/Signup
```

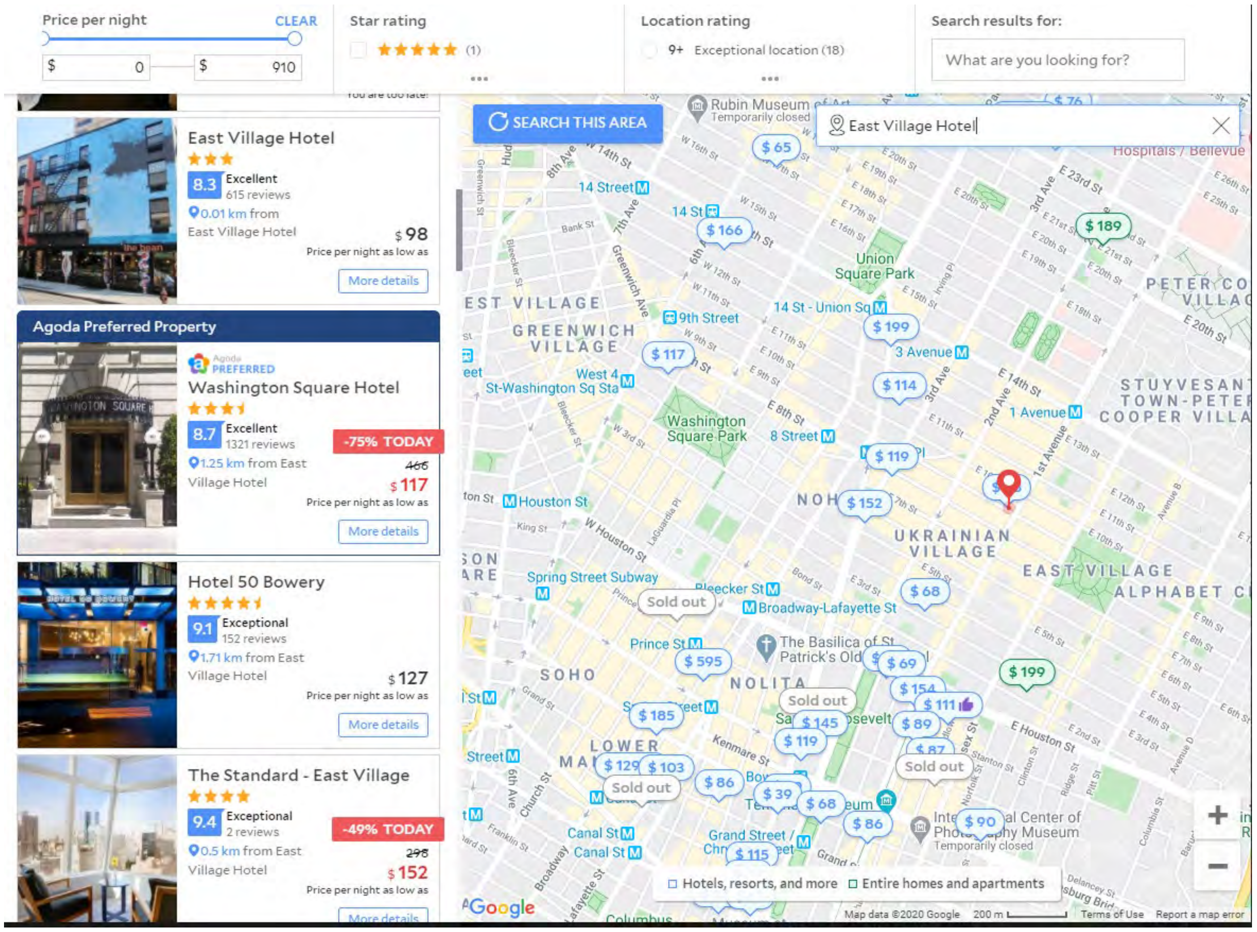


Property name (English): My Sample Hotel
Country: United States
State / Province: Connecticut
County: Select County
City: Norwalk (CT)
ZIP / Postal code: 06854
Street address: 800 Connecticut Avenue
Main phone number: United States (+1)
3473221285
Location of your property

```
Name
GeocodeService.Search?4sUnited%20Sta...
StatsService.RecordStats?1m5&1sut%7C...
StatsService.RecordStats?1m5&1sut%7C...

X  Headers  Preview  Response  Initiator  Timing
▾ /**/_xdc_._me0fye && _xdc_._me0fye( {,…} )
  ▾ results: [{address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…],…}]
    ▾ 0: {address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…],…}
      ▾ address_components: [{long_name: "800", short_name: "800", types: ["street_number"]},…]
        ▸ 0: {long_name: "800", short_name: "800", types: ["street_number"]}
        ▸ 1: {long_name: "Connecticut Avenue", short_name: "Connecticut Ave", types: ["route"]}
        ▸ 2: {long_name: "Norwalk", short_name: "Norwalk", types: ["locality", "political"]}
        ▸ 3: {long_name: "Fairfield County", short_name: "Fairfield County",…}
        ▸ 4: {long_name: "Connecticut", short_name: "CT", types: ["administrative_area_level_1", "political"]}
        ▸ 5: {long_name: "United States", short_name: "US", types: ["country", "political"]}
        ▸ 6: {long_name: "06854", short_name: "06854", types: ["postal_code"]}
      formatted_address: "800 Connecticut Ave, Norwalk, CT 06854, USA"
      ▾ geometry: {bounds: {northeast: {lat: 41.0940489, lng: -73.45357469999999},…},…}
        ▸ bounds: {northeast: {lat: 41.0940489, lng: -73.45357469999999},…}
        ▾ location: {lat: 41.0930242, lng: -73.4539645}
            lat: 41.0930242
            lng: -73.4539645
          location_type: "ROOFTOP"
        ▸ viewport: {northeast: {lat: 41.0945939302915, lng: -73.4526928197085},…}
      place_id: "ChIJVVopERge6IkR0r1TD2mCAxk"
      ▸ types: ["premise"]
  status: "OK"
```

299. The YCS Platform defines a UI object corresponding to a web component for an input or output of a web service, for presentation on the display. For example, when the Google Maps API is embedded on a Priceline or Agoda property listing website, UI objects for web components such as zoom-in and zoom-out buttons, a search field, a search button, a slide bar for

"Price per night," a check box for "Star rating," and a radio button for "Location rating" are automatically selected by the YCS Platform as preferred UI objects.


Price per night CLEAR $ 0 $ 910 | Star rating ★★★★★ (1) | Location rating 9+ Exceptional location (18) | Search results for: What are you looking for?

East Village Hotel ★★★ 8.3 Excellent 615 reviews 0.01 km from East Village Hotel $ 98 Price per night as low as More details

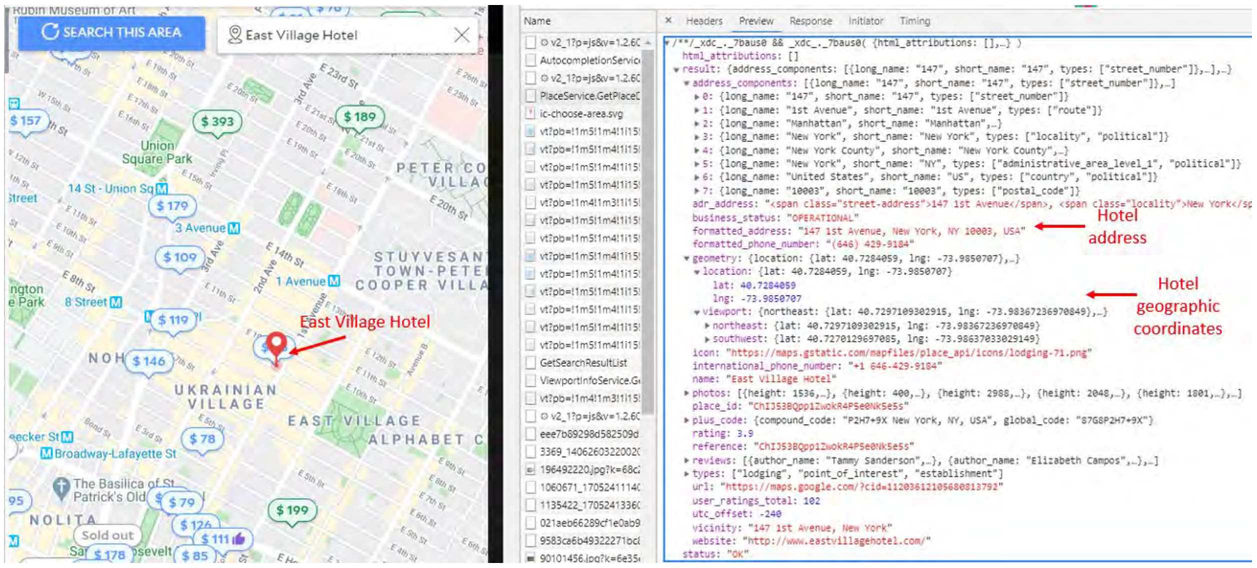
Agoda Preferred Property
Agoda PREFERRED Washington Square Hotel ★★★ 8.7 Excellent 1321 reviews 1.25 km from East Village Hotel -75% TODAY ~~466~~ $ 117 Price per night as low as More details

Hotel 50 Bowery ★★★★ 9.1 Exceptional 152 reviews 1.71 km from East Village Hotel $ 127 Price per night as low as More details

The Standard - East Village ★★★★ 9.4 Exceptional 2 reviews 0.5 km from East Village Hotel -49% TODAY ~~298~~ $ 152 Price per night as low as More details

300. When a UI object is so defined, a symbolic name (e.g., JSON data) from the web component corresponding to the defined UI object is selected and associated with the defined UI object. The symbolic name is only available to UI objects that support the defined data format associated with that symbolic name. The example below demonstrates a user searching for "East Village Hotel" on the Google Maps API. When the user types "East Village Hotel" into the search field of the Google Maps API, the Google Maps web service responds by sending a JSON dataset to the YCS Platform. In particular, the symbolic names "address_components" and "formatted_address" are associated with the hotel address, and the symbolic name "geometry" is associated with the geographic coordinates of the hotel. This information is further used to display the property's location on the Google Maps API. In the example below, a pin object associated

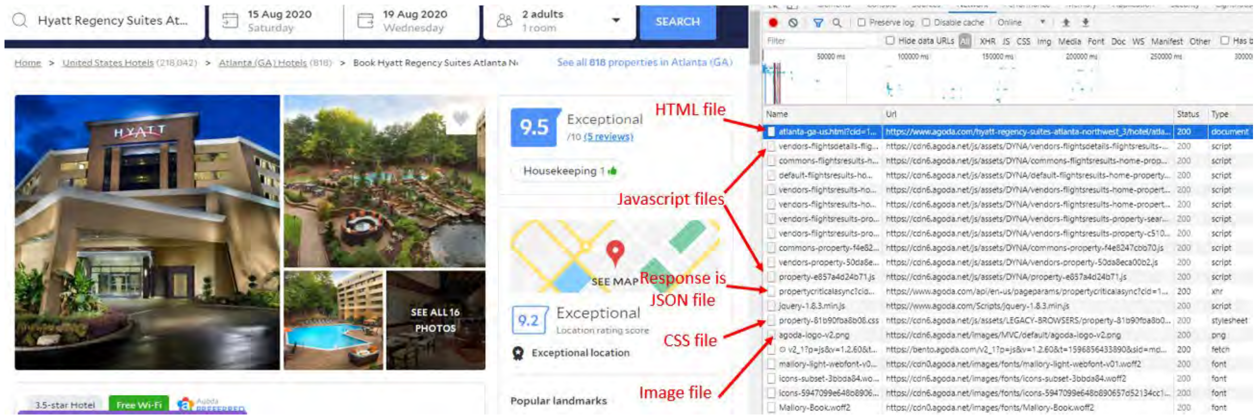
with the address and geographic coordinates of the property is dropped on the map to indicate the location of the "East Village Hotel." The pin object is a preferred UI object indicating the target hotel, distinguishable from bubbles indicating locations and prices for other nearby hotels.



```
/**/_xdc_._7baus0 && _xdc_._7baus0( {html_attributions: [],…} )
  html_attributions: []
 ▾result: {address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…],…}
  ▾address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…]
   ▸0: {long_name: "147", short_name: "147", types: ["street_number"]}
   ▸1: {long_name: "1st Avenue", short_name: "1st Avenue", types: ["route"]}
   ▸2: {long_name: "Manhattan", short_name: "Manhattan",…}
   ▸3: {long_name: "New York", short_name: "New York", types: ["locality", "political"]}
   ▸4: {long_name: "New York County", short_name: "New York County",…}
   ▸5: {long_name: "New York", short_name: "NY", types: ["administrative_area_level_1", "political"]}
   ▸6: {long_name: "United States", short_name: "US", types: ["country", "political"]}
   ▸7: {long_name: "10003", short_name: "10003", types: ["postal_code"]}
   adr_address: "<span class="street-address">147 1st Avenue</span>, <span class="locality">New York</sp
   business_status: "OPERATIONAL"
   formatted_address: "147 1st Avenue, New York, NY 10003, USA"
   formatted_phone_number: "(646) 429-9184"
  ▾geometry: {location: {lat: 40.7284059, lng: -73.9850707},…}
   ▾location: {lat: 40.7284059, lng: -73.9850707}
      lat: 40.7284059
      lng: -73.9850707
   ▾viewport: {northeast: {lat: 40.7297109302915, lng: -73.98367236970849},…}
    ▸northeast: {lat: 40.7297109302915, lng: -73.98367236970849}
    ▸southwest: {lat: 40.7270129697085, lng: -73.98637033029149}
   icon: "https://maps.gstatic.com/mapfiles/place_api/icons/lodging-71.png"
   international_phone_number: "+1 646-429-9184"
   name: "East Village Hotel"
  ▸photos: [{height: 1536,…}, {height: 400,…}, {height: 2988,…}, {height: 2048,…}, {height: 1801,…},…]
   place_id: "ChIJ53BQpp1ZwokR4P5e0Nk5e5s"
  ▸plus_code: {compound_code: "P2H7+9X New York, NY, USA", global_code: "87G8P2H7+9X"}
   rating: 3.9
   reference: "ChIJ53BQpp1ZwokR4P5e0Nk5e5s"
  ▸reviews: [{author_name: "Tammy Sanderson",…}, {author_name: "Elizabeth Campos",…},…]
  ▸types: ["lodging", "point_of_interest", "establishment"]
   url: "https://maps.google.com/?cid=11203612105680813792"
   user_ratings_total: 102
   utc_offset: -240
   vicinity: "147 1st Avenue, New York"
   website: "http://www.eastvillagehotel.com/"
  status: "OK"
```

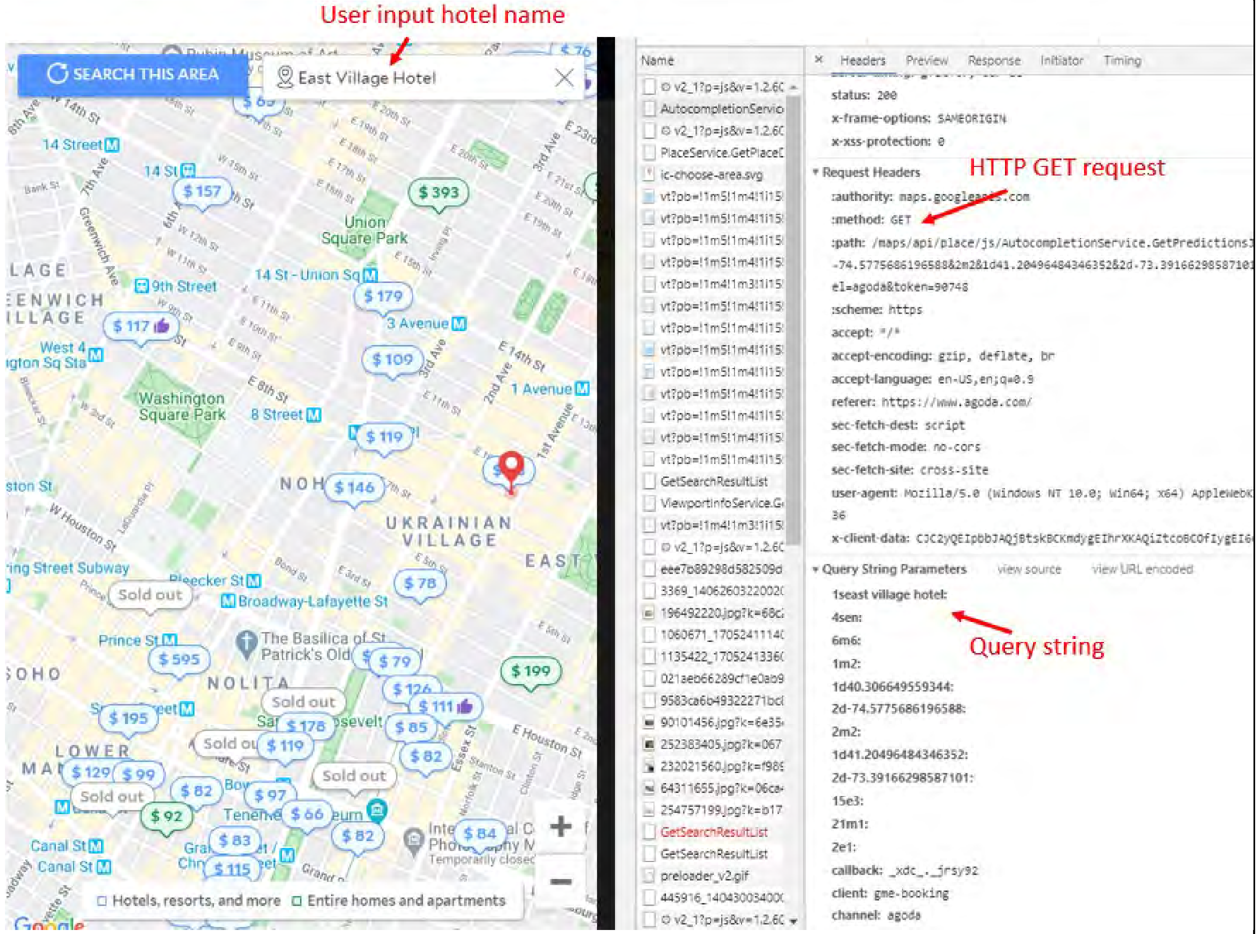
301. On information and belief, the YCS Platform stores information representative of the defined UI object and related settings (e.g., JSON data, along with standard HTML, CSS, and Javascript code) in a database and/or CDN as described above.

302. To generate the property listing, the YCS Platform retrieves the information for the UI objects stored in the database, and builds an application consisting of one or more web pages views from at least a portion of the database using the player. The player (e.g., a browser engine or operating system) utilizes the data stored in the database to generate for the display at least a portion of a web page. In the example below, the YCS Platform converts HTML, CSS, Javascript, JSON, and other files into an active property listing website for "Hyatt Regency Suites Atlanta."



Name	Url	Status	Type
atlanta-ga-us.html?cid=1...	https://www.agoda.com/hyatt-regency-suites-atlanta-northwest_3/hotel/atla...	200	document
vendors-flightsdetails-flig...	https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsdetails-flightsresults-...	200	script
commons-flightsresults-h...	https://cdn6.agoda.net/js/assets/DYNA/commons-flightsresults-home-prop...	200	script
default-flightsresults-ho...	https://cdn6.agoda.net/js/assets/DYNA/default-flightsresults-home-property...	200	script
vendors-flightsresults-ho...	https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsresults-home-propert...	200	script
vendors-flightsresults-ho...	https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsresults-home-propert...	200	script
vendors-flightsresults-pro...	https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsresults-property-sear...	200	script
vendors-flightsresults-pro...	https://cdn6.agoda.net/js/assets/DYNA/vendors-flightsresults-property-c510...	200	script
commons-property-f4e82...	https://cdn6.agoda.net/js/assets/DYNA/commons-property-f4e8247cbb70.js	200	script
vendors-property-50da8e...	https://cdn6.agoda.net/js/assets/DYNA/vendors-property-50da8eca00b2.js	200	script
property-e857a4d24b71.js	https://cdn6.agoda.net/js/assets/DYNA/property-e857a4d24b71.js	200	script
propertycriticalasync?cid...	https://www.agoda.com/api/en-us/pageparams/propertycriticalasync?cid=1...	200	xhr
jquery-1.8.3.min.js	https://www.agoda.com/Scripts/jquery-1.8.3.min.js	200	script
property-81b90fba8b08.css	https://cdn6.agoda.net/js/assets/LEGACY-BROWSERS/property-81b90fba8b0...	200	stylesheet
agoda-logo-v2.png	https://cdn6.agoda.net/images/MVC/default/agoda-logo-v2.png	200	png
v2_1?p=js&v=1.2.60&t...	https://bento.agoda.com/v2_1?p=js&v=1.2.60&t=1596856433890&sid=md...	200	fetch
mallory-light-webfont-v0...	https://cdn0.agoda.net/images/fonts/mallory-light-webfont-v01.woff2	200	font
icons-subset-3bbda84.wo...	https://cdn6.agoda.net/images/fonts/icons-subset-3bbda84.woff2	200	font
icons-5947099e648b8906...	https://cdn6.agoda.net/images/fonts/icons-5947099e648b890657d52134cc1...	200	font
Mallory-Book.woff2	https://cdn0.agoda.net/images/fonts/Mallory-Book.woff2	200	font

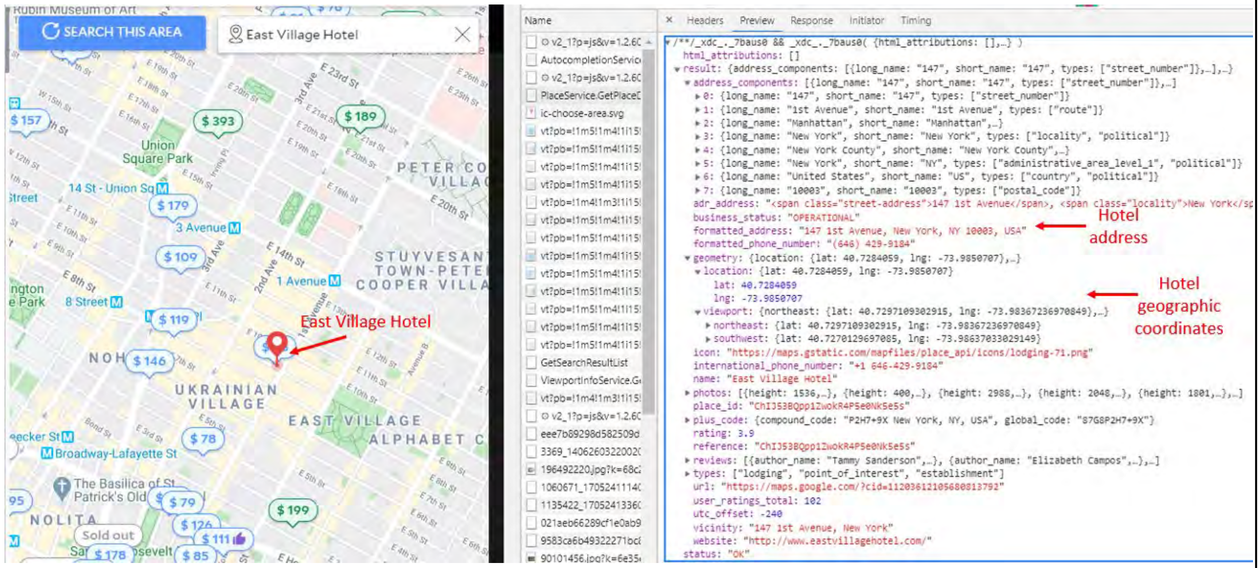
303. When the application and player are executed on the device, an input value can be provided by a user. The device then provides the input value and a corresponding input symbolic name to the web service, which in turn generates an output value and a corresponding symbolic name for the output. By way of example, when the Google Maps API accepts an input value (e.g., an input hotel name, a zoom-in click, or a click on the "Explore" tab) from a user, a corresponding input symbolic name (e.g., a query string or JSON data for the input) is transmitted to the Google Maps web service through an HTTP request protocol, such as a GET method call. The example below demonstrates a user searching for "East Village Hotel" on the Google Maps API on an Agoda website. When the user types "East Village Hotel," a query string with parameters including the symbolic name "1s" and the associated value "East Village Hotel" is sent to the Google Maps web service to initiate an autocompletion service with an HTTP GET request.



```
Name
v2_1?p=js&v=1.2.6(
AutocompletionServic
v2_1?p=js&v=1.2.6(
PlaceService.GetPlace(
ic-choose-area.svg
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m4!1m3!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
vt?pb=!1m5!1m4!1i15
GetSearchResultList
ViewportInfoService.G
vt?pb=!1m4!1m3!1i15
v2_1?p=js&v=1.2.6(
eee7b89298d582509d
3369_14062603220020
196492220.jpg?k=68c
1060671_17052411140
1135422_17052413360
021aeb66289cf1e0ab9
9583ca6b49322271bcl
90101456.jpg?k=6e35
252383405.jpg?k=067
232021560.jpg?k=f98
64311655.jpg?k=06ca
254757199.jpg?k=b17
GetSearchResultList
GetSearchResultList
preloader_v2.gif
445916_14043003400
v2_1?p=js&v=1.2.6(
```

```
× Headers Preview Response Initiator Timing
status: 200
x-frame-options: SAMEORIGIN
x-xss-protection: 0
▾ Request Headers
:authority: maps.googleapis.com
:method: GET
:path: /maps/api/place/js/AutocompletionService.GetPredictionsJ
-74.5775686196588&2m2&1d41.20496484346352&2d-73.391662985871O1
el=agoda&token=90748
:scheme: https
accept: */*
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
referer: https://www.agoda.com/
sec-fetch-dest: script
sec-fetch-mode: no-cors
sec-fetch-site: cross-site
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebK
36
x-client-data: CJC2yQEIpbbJAQjBtskBCKmdygEIhrXKAQiZtcoBCOfIygEI6
▾ Query String Parameters    view source    view URL encoded
1seast village hotel:
4sen:
6m6:
1m2:
1d40.306649559344:
2d-74.5775686196588:
2m2:
1d41.20496484346352:
2d-73.39166298587101:
15e3:
21m1:
2e1:
callback: _xdc_._jrsy92
client: gme-booking
channel: agoda
```

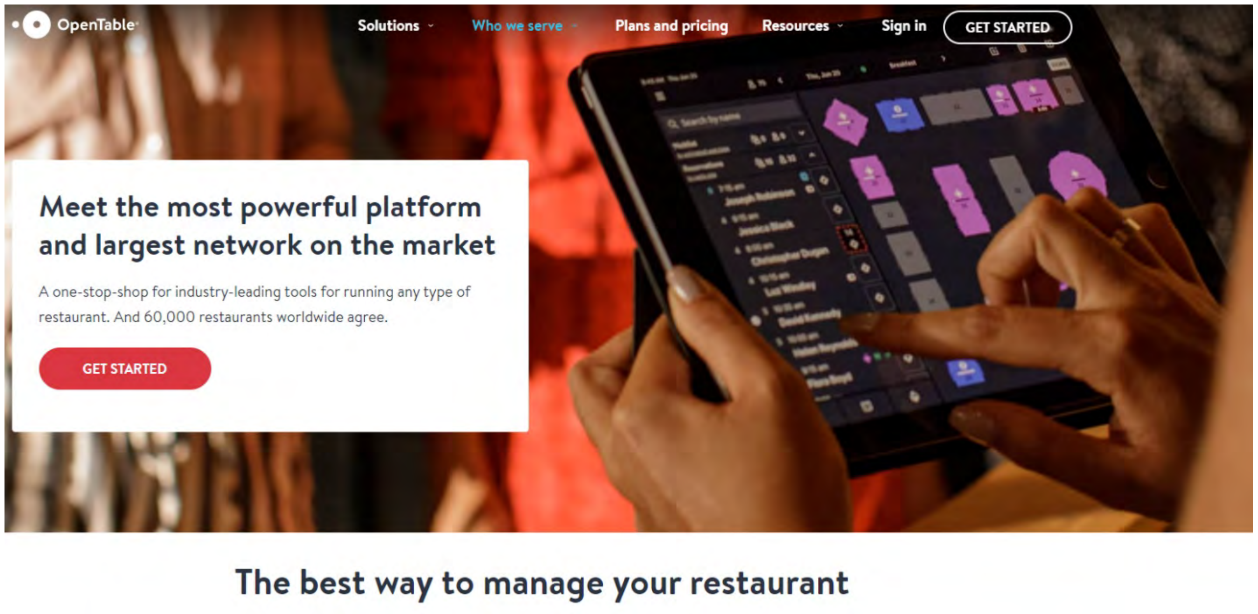
304. The input symbolic name is utilized by the Google Map web service to generate an output value with an associated output symbolic name (e.g., JSON data for the output). The player then provides instructions to the display of the device to present the output value in the API. For example, when a user searches "East Village Hotel" and selects an autocomplete suggestion from the Google Maps API, the Google Maps web service generates and transmits a corresponding hotel address and geographic coordinates in the format of JSON data to the web browser and displays a pin indicating the location of the hotel on the Google Maps API.



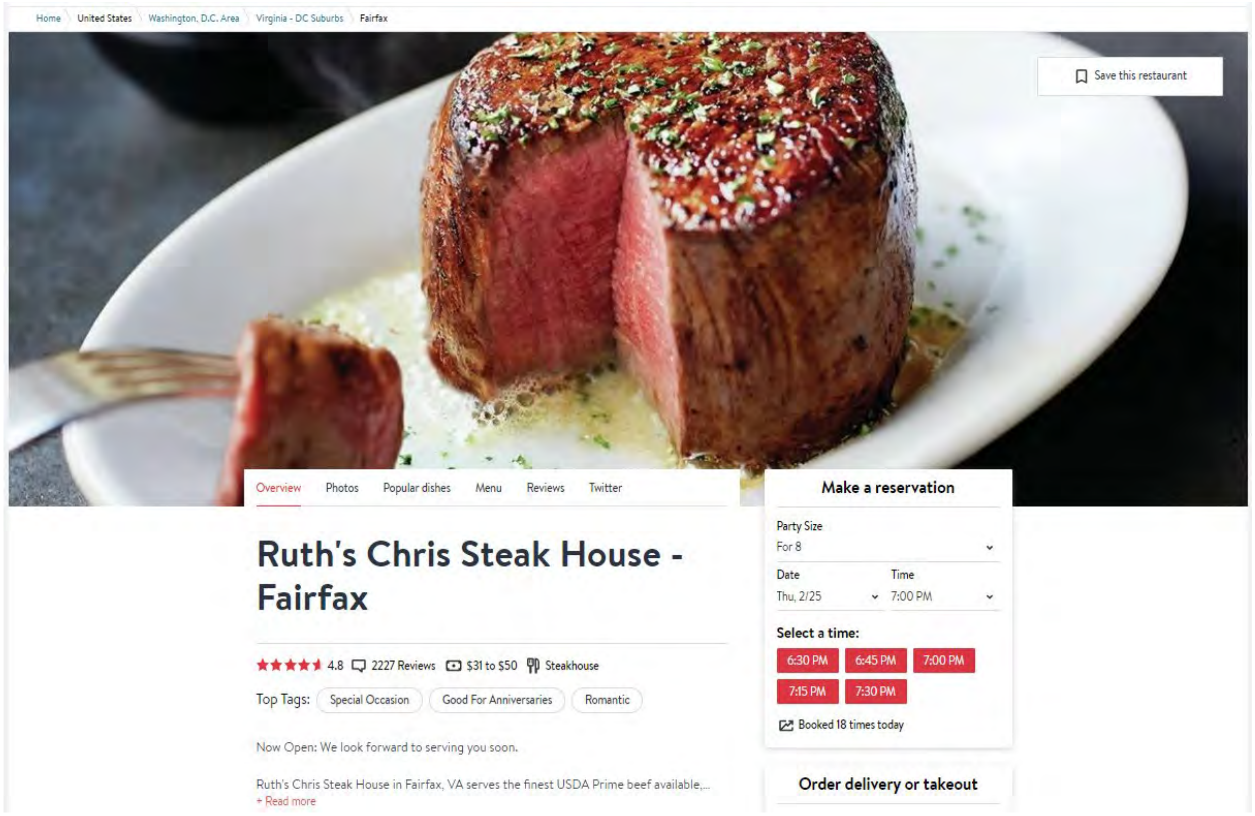
```
/**/_xdc_._7baus0 && _xdc_._7baus0( {html_attributions: [],…} )
  html_attributions: []
  result: {address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…],…}
    address_components: [{long_name: "147", short_name: "147", types: ["street_number"]},…]
      0: {long_name: "147", short_name: "147", types: ["street_number"]}
      1: {long_name: "1st Avenue", short_name: "1st Avenue", types: ["route"]}
      2: {long_name: "Manhattan", short_name: "Manhattan",…}
      3: {long_name: "New York", short_name: "New York", types: ["locality", "political"]}
      4: {long_name: "New York County", short_name: "New York County",…}
      5: {long_name: "New York", short_name: "NY", types: ["administrative_area_level_1", "political"]}
      6: {long_name: "United States", short_name: "US", types: ["country", "political"]}
      7: {long_name: "10003", short_name: "10003", types: ["postal_code"]}
    adr_address: "<span class="street-address">147 1st Avenue</span>, <span class="locality">New York</sp
    business_status: "OPERATIONAL"
    formatted_address: "147 1st Avenue, New York, NY 10003, USA"
    formatted_phone_number: "(646) 429-9184"
    geometry: {location: {lat: 40.7284059, lng: -73.9850707},…}
      location: {lat: 40.7284059, lng: -73.9850707}
        lat: 40.7284059
        lng: -73.9850707
      viewport: {northeast: {lat: 40.7297109302915, lng: -73.98367236970849},…}
        northeast: {lat: 40.7297109302915, lng: -73.98367236970849}
        southwest: {lat: 40.7270129697085, lng: -73.98637033029149}
    icon: "https://maps.gstatic.com/mapfiles/place_api/icons/lodging-71.png"
    international_phone_number: "+1 646-429-9184"
    name: "East Village Hotel"
    photos: [{height: 1536,…}, {height: 400,…}, {height: 2988,…}, {height: 2048,…}, {height: 1801,…},…]
    place_id: "ChIJ53BQpp1ZwokR4P5e0Nk5e5s"
    plus_code: {compound_code: "P2H7+9X New York, NY, USA", global_code: "87G8P2H7+9X"}
    rating: 3.9
    reference: "ChIJ53BQpp1ZwokR4P5e0Nk5e5s"
    reviews: [{author_name: "Tammy Sanderson",…}, {author_name: "Elizabeth Campos",…},…]
    types: ["lodging", "point_of_interest", "establishment"]
    url: "https://maps.google.com/?cid=11203612105680813792"
    user_ratings_total: 102
    utc_offset: -240
    vicinity: "147 1st Avenue, New York"
    website: "http://www.eastvillagehotel.com/"
  status: "OK"
```

<u>The OpenTable Platform:</u>

305. OpenTable infringes at least claim 15 of the '044 patent through a combination of features in the OpenTable Platform that collectively practice each limitation of claim 15. By way of example, OpenTable provides the OpenTable Platform, a browser-based platform for, *inter alia*, creating restaurant profile websites that can be displayed on a display of a device.


OpenTable
Solutions
Who we serve
Plans and pricing
Resources
Sign in
GET STARTED
Meet the most powerful platform and largest network on the market
A one-stop-shop for industry-leading tools for running any type of restaurant. And 60,000 restaurants worldwide agree.
GET STARTED
The best way to manage your restaurant

https://restaurant.opentable.com/why-opentable/


Home > United States > Washington, D.C. Area > Virginia - DC Suburbs > Fairfax

Save this restaurant

Overview Photos Popular dishes Menu Reviews Twitter

Ruth's Chris Steak House - Fairfax

★★★★★ 4.8 2227 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Romantic

Now Open: We look forward to serving you soon.

Ruth's Chris Steak House in Fairfax, VA serves the finest USDA Prime beef available,...
+ Read more

Make a reservation

Party Size: For 8

Date: Thu, 2/25 Time: 7:00 PM

Select a time:

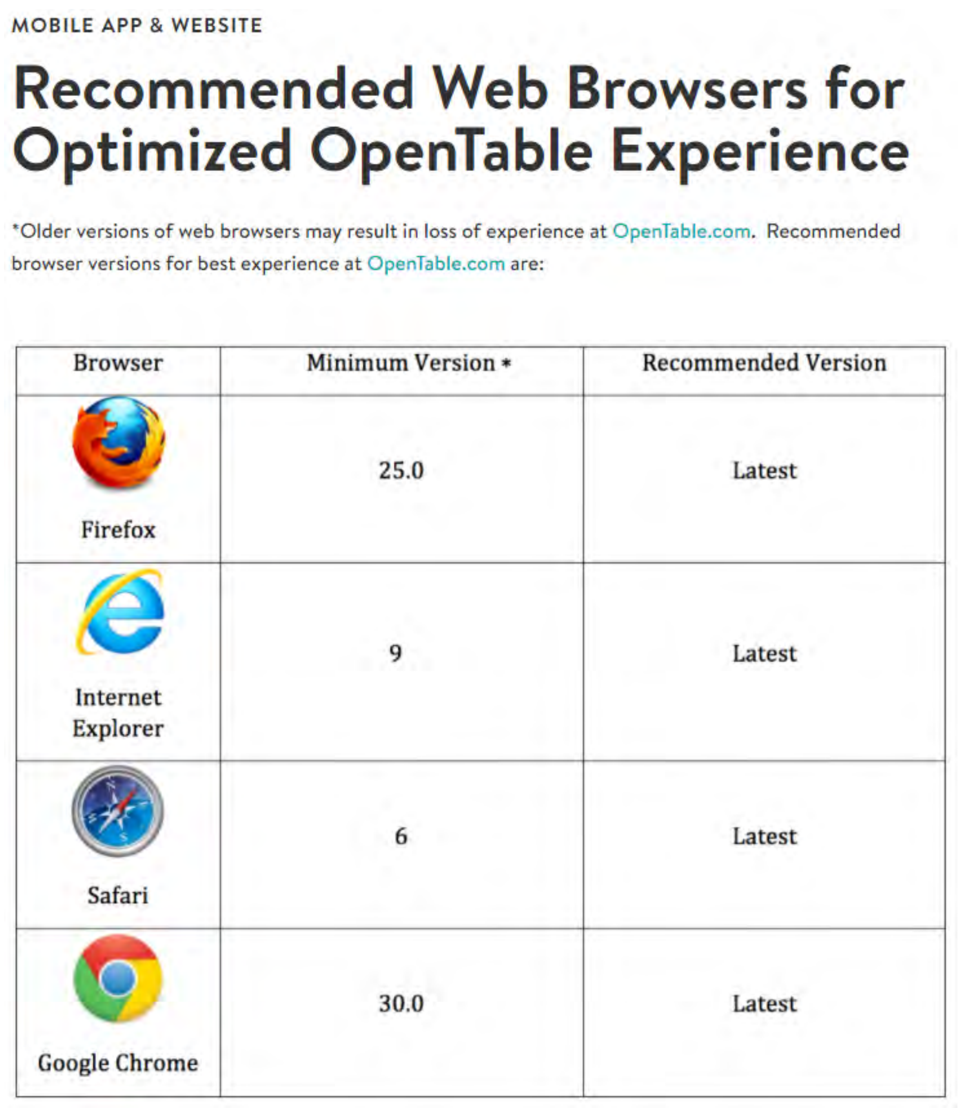
6:30 PM 6:45 PM 7:00 PM 7:15 PM 7:30 PM

Booked 18 times today

Order delivery or takeout

https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8eae-7ada9d88b68e&corrid=dfc6cee2-c8f8-4054-8eae-7ada9d88b68e&avt=eyJ2IjoyLCJtIjoxLCJwIjowLCJzIjowLCJuIjowfQ

307. The OpenTable platform displays restaurant profile on a device having a player, such as a device-dependent code for a browser engine for a browser, or for an operating system or application of a device. See, e.g., Shopify Inc. v. Express Mobile, Inc., Case No. 1:19-cv-00439-RGA, D. Del., D.I. 137. For example, the OpenTable platform can display a restaurant profile on a Google Chrome, Mozilla Firefox, Apple Safari, or Microsoft Internet Explorer using the player.


MOBILE APP & WEBSITE

Recommended Web Browsers for Optimized OpenTable Experience

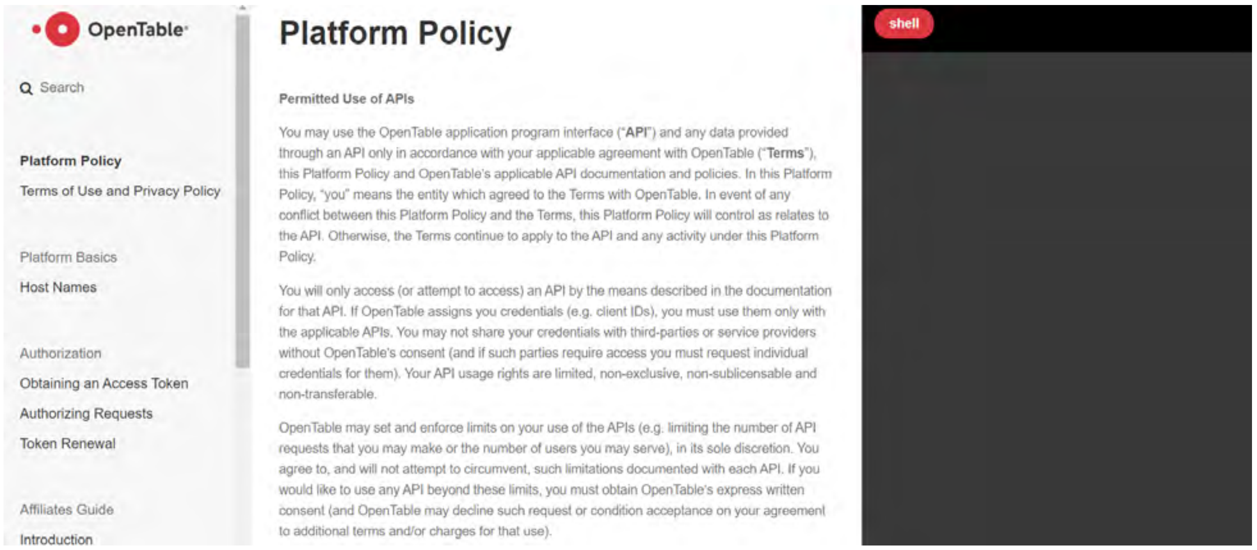
*Older versions of web browsers may result in loss of experience at OpenTable.com. Recommended browser versions for best experience at OpenTable.com are:

Browser	Minimum Version *	Recommended Version
Firefox	25.0	Latest
Internet Explorer	9	Latest
Safari	6	Latest
Google Chrome	30.0	Latest

https://help.opentable.com/s/article/Recommended-Web-Browsers-for-Optimized-OpenTable-Experience-1505260708133?language=en_US

308. On information and belief, the OpenTable platform displays restaurant profile on a device having a non-volatile computer memory storing symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network. The symbolic names are present in the form of JSON data, and are used to evoke web components for inputs and outputs of web services, such as the OpenTable Application Programming Interface (API). In particular, the data is sent and received in JSON format.

309.



Platform Policy

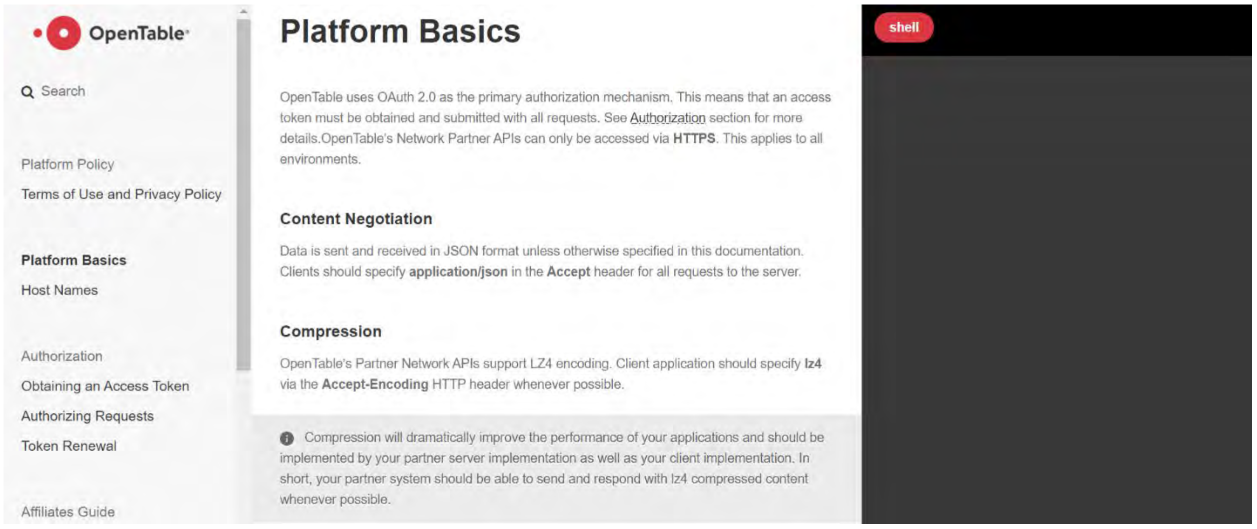
Permitted Use of APIs

You may use the OpenTable application program interface ("**API**") and any data provided through an API only in accordance with your applicable agreement with OpenTable ("**Terms**"), this Platform Policy and OpenTable's applicable API documentation and policies. In this Platform Policy, "you" means the entity which agreed to the Terms with OpenTable. In event of any conflict between this Platform Policy and the Terms, this Platform Policy will control as relates to the API. Otherwise, the Terms continue to apply to the API and any activity under this Platform Policy.

You will only access (or attempt to access) an API by the means described in the documentation for that API. If OpenTable assigns you credentials (e.g. client IDs), you must use them only with the applicable APIs. You may not share your credentials with third-parties or service providers without OpenTable's consent (and if such parties require access you must request individual credentials for them). Your API usage rights are limited, non-exclusive, non-sublicensable and non-transferable.

OpenTable may set and enforce limits on your use of the APIs (e.g. limiting the number of API requests that you may make or the number of users you may serve), in its sole discretion. You agree to, and will not attempt to circumvent, such limitations documented with each API. If you would like to use any API beyond these limits, you must obtain OpenTable's express written consent (and OpenTable may decline such request or condition acceptance on your agreement to additional terms and/or charges for that use).

https://platform.opentable.com/documentation/#platform-policy



Platform Basics

OpenTable uses OAuth 2.0 as the primary authorization mechanism. This means that an access token must be obtained and submitted with all requests. See Authorization section for more details.OpenTable's Network Partner APIs can only be accessed via **HTTPS**. This applies to all environments.

Content Negotiation

Data is sent and received in JSON format unless otherwise specified in this documentation. Clients should specify **application/json** in the **Accept** header for all requests to the server.

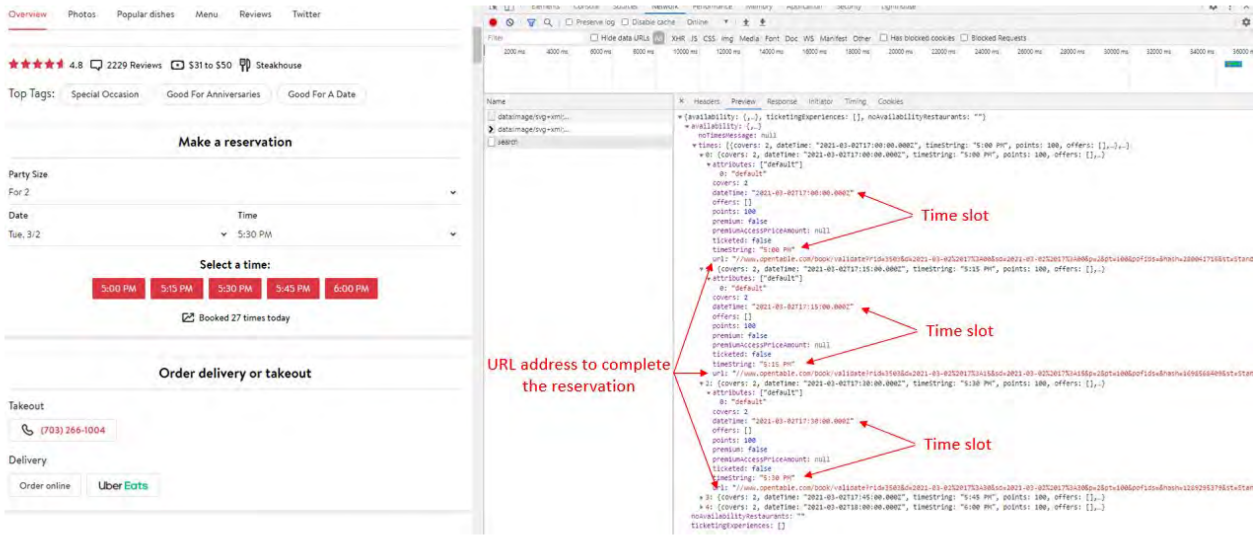
Compression

OpenTable's Partner Network APIs support LZ4 encoding. Client application should specify **lz4** via the **Accept-Encoding** HTTP header whenever possible.

Compression will dramatically improve the performance of your applications and should be implemented by your partner server implementation as well as your client implementation. In short, your partner system should be able to send and respond with lz4 compressed content whenever possible.

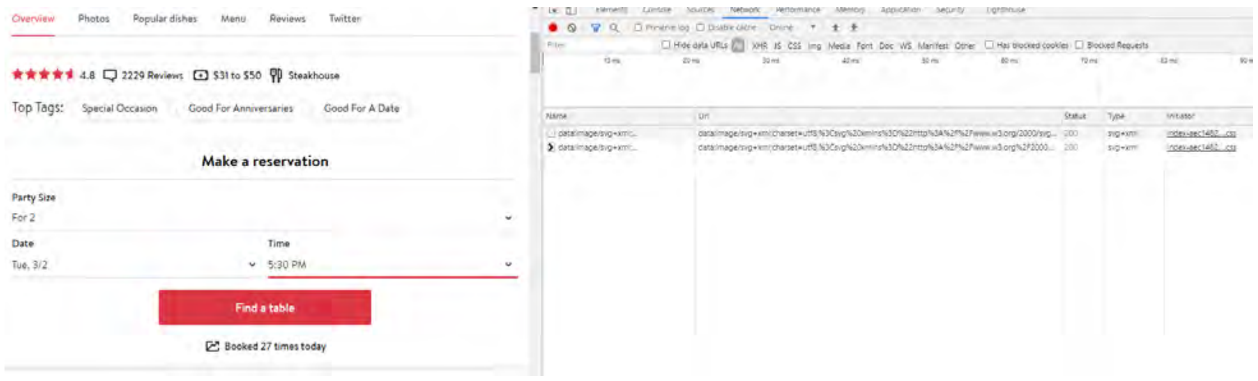
https://platform.opentable.com/documentation/#platform-basics

310. The JSON data used to evoke web components and to communicate inputs and outputs for the applications comprise key/value pairs that are essentially symbolic names constituting character strings with no persistent address or pointer. The example below shows JSON data for available time slots in key/value pairs comprising character strings with no persistent address or pointer.

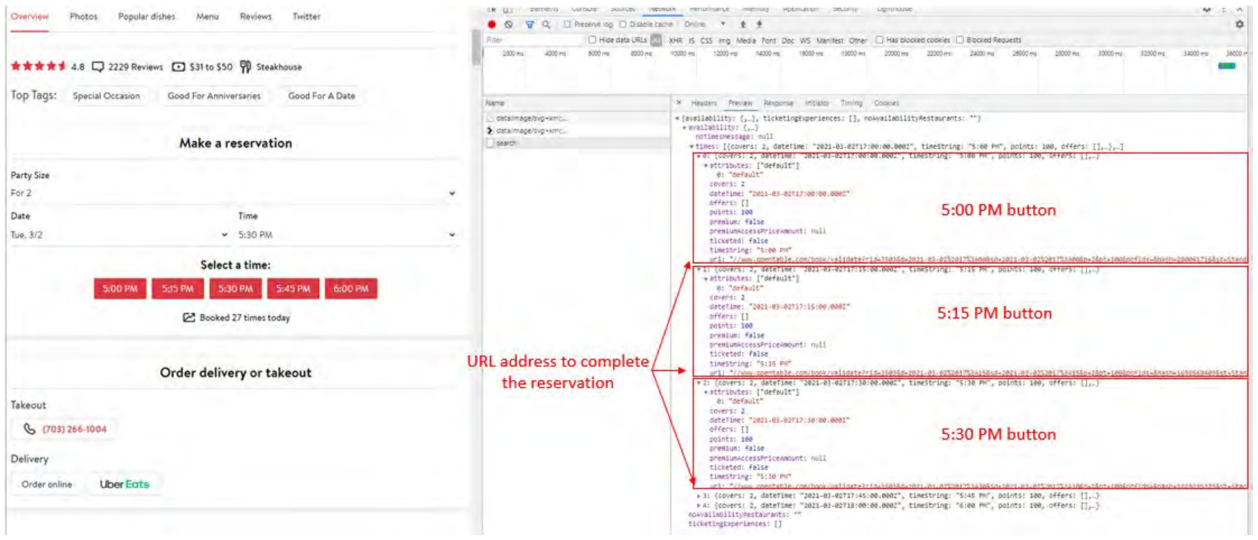

Overview Photos Popular dishes Menu Reviews Twitter
4.8 2229 Reviews $31 to $50 Steakhouse
Top Tags: Special Occasion Good For Anniversaries Good For A Date
Make a reservation
Party Size For 2
Date Tue, 3/2 Time 5:30 PM
Select a time:
5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM
Booked 27 times today
Order delivery or takeout
Takeout (703) 266-1004
Delivery Order online Uber Eats
Headers Preview Response Initiator Timing Cookies
Time slot
Time slot
Time slot
URL address to complete the reservation

```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
    noTimesMessage: null
   ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
    ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:00:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:00 PM"
      url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A00&sd=2021-03-02%2017%3A00&p=2&pt=100&pofids=&hash=2800417168st=Stand
    ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:15:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:15 PM"
      url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A15&sd=2021-03-02%2017%3A15&p=2&pt=100&pofids=&hash=1698568409&st=Stan
    ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:30:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:30 PM"
      url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A30&sd=2021-03-02%2017%3A30&p=2&pt=100&pofids=&hash=1289295379&st=Stan
    ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
    ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```

311. Each symbolic name has an associated data format class type corresponding to a subclass of User Interface (UI) objects that support the data format type of the symbolic name, and has a preferred UI object. By way of example, when "Find a table" button is pressed, the OpenTable platform creates and displays a subclass of UI objects, e.g., five button objects, that represent five available time slots. In particular, symbolic name "0" associates with the "5:00 PM" button and the button object is the preferred UI object. The symbolic name "0" also includes an URL address of the OpenTable reservation web service to complete the reservation in the specified time slot.

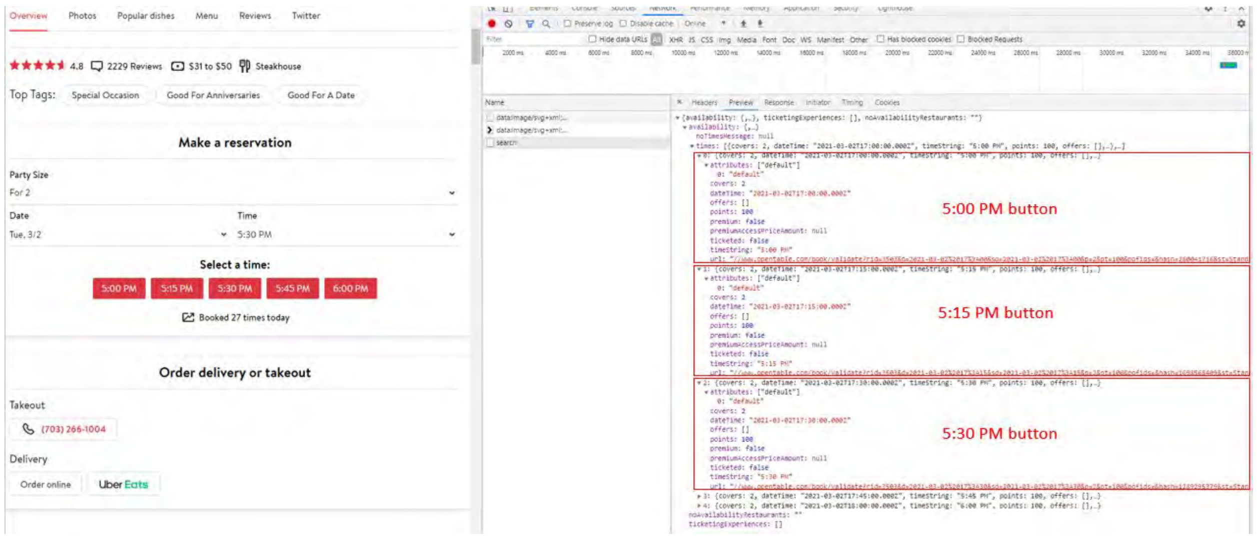

Overview Photos Popular dishes Menu Reviews Twitter
4.8 2229 Reviews $31 to $50 Steakhouse
Top Tags: Special Occasion Good For Anniversaries Good For A Date
Make a reservation
Party Size For 2
Date Tue, 3/2 Time 5:30 PM
Find a table
Booked 27 times today
Elements Console Sources Network Performance Memory Application Security Lighthouse
Name Url Status Type Initiator

Name	Url	Status	Type	Initiator
data:image/svg+xml;…	data:image/svg+xml;charset=utf8,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org/2000/svg…	200	svg+xml	index-aec1482….css
data:image/svg+xml;…	data:image/svg+xml;charset=utf8,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000…	200	svg+xml	index-aec1482….css



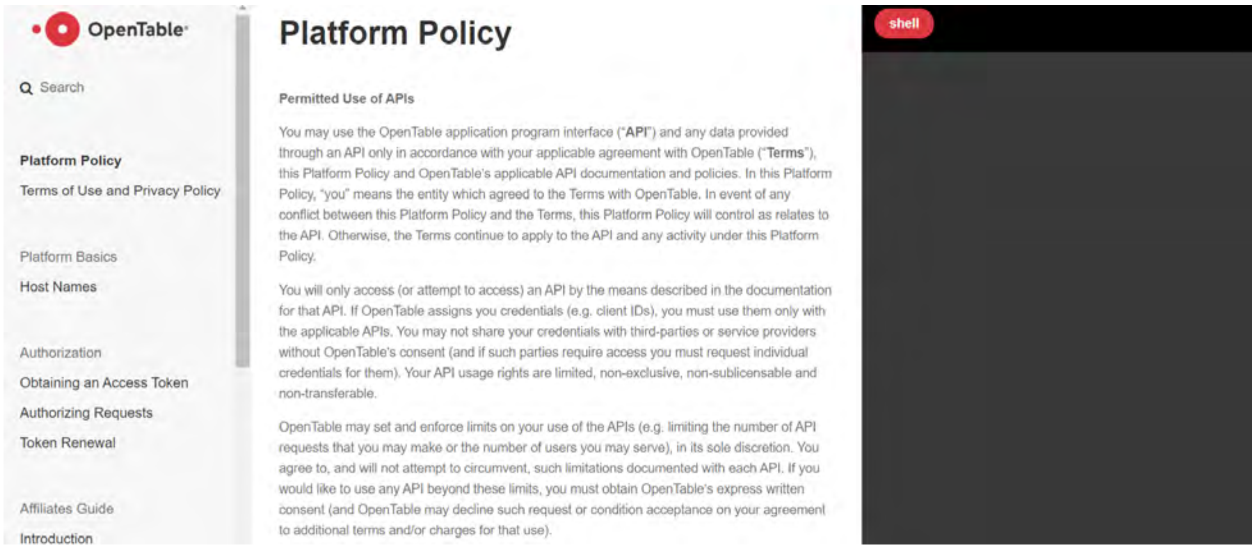
```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
     noTimesMessage: null
    ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
      ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
        ▾attributes: ["default"]
           0: "default"
         covers: 2
         dateTime: "2021-03-02T17:00:00.000Z"
         offers: []
         points: 100
         premium: false
         premiumAccessPriceAmount: null
         ticketed: false
         timeString: "5:00 PM"
         url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A00&sd=2021-03-02%2017%3A00&p=2&pt=100&pofIds=&hash=[illegible]&st=Stand
      ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
        ▾attributes: ["default"]
           0: "default"
         covers: 2
         dateTime: "2021-03-02T17:15:00.000Z"
         offers: []
         points: 100
         premium: false
         premiumAccessPriceAmount: null
         ticketed: false
         timeString: "5:15 PM"
         url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A15&sd=2021-03-02%2017%3A15&p=2&pt=100&pofIds=&hash=[illegible]&st=Stan
      ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
        ▾attributes: ["default"]
           0: "default"
         covers: 2
         dateTime: "2021-03-02T17:30:00.000Z"
         offers: []
         points: 100
         premium: false
         premiumAccessPriceAmount: null
         ticketed: false
         timeString: "5:30 PM"
         url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A30&sd=2021-03-02%2017%3A30&p=2&pt=100&pofIds=&hash=[illegible]&st=Stan
      ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
      ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
     noAvailabilityRestaurants: ""
     ticketingExperiences: []
```

312. The OpenTable platform defines a UI object corresponding to a web component in the registry for an input or output of a web service, for presentation on the display. By way of example, the authoring tool defines UI objects for web components such as a button to represent the available time slot.



```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
     noTimesMessage: null
    ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
      ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
        ▾attributes: ["default"]
           0: "default"
         covers: 2
         dateTime: "2021-03-02T17:00:00.000Z"
         offers: []
         points: 100
         premium: false
         premiumAccessPriceAmount: null
         ticketed: false
         timeString: "5:00 PM"
         url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A00&sd=2021-03-02%2017%3A00&p=2&pt=100&pofIds=&hash=[illegible]&st=Stand
      ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
        ▾attributes: ["default"]
           0: "default"
         covers: 2
         dateTime: "2021-03-02T17:15:00.000Z"
         offers: []
         points: 100
         premium: false
         premiumAccessPriceAmount: null
         ticketed: false
         timeString: "5:15 PM"
         url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A15&sd=2021-03-02%2017%3A15&p=2&pt=100&pofIds=&hash=[illegible]&st=Stan
      ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
        ▾attributes: ["default"]
           0: "default"
         covers: 2
         dateTime: "2021-03-02T17:30:00.000Z"
         offers: []
         points: 100
         premium: false
         premiumAccessPriceAmount: null
         ticketed: false
         timeString: "5:30 PM"
         url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A30&sd=2021-03-02%2017%3A30&p=2&pt=100&pofIds=&hash=[illegible]&st=Stan
      ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
      ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
     noAvailabilityRestaurants: ""
     ticketingExperiences: []
```

313. The OpenTable platform provides various web services through the OpenTable application programming interface (API). A client can use the OpenTable API to access the OpenTable platform for data and services, such as make a reservation. In particular, the data is sent and received in JSON format.



Platform Policy

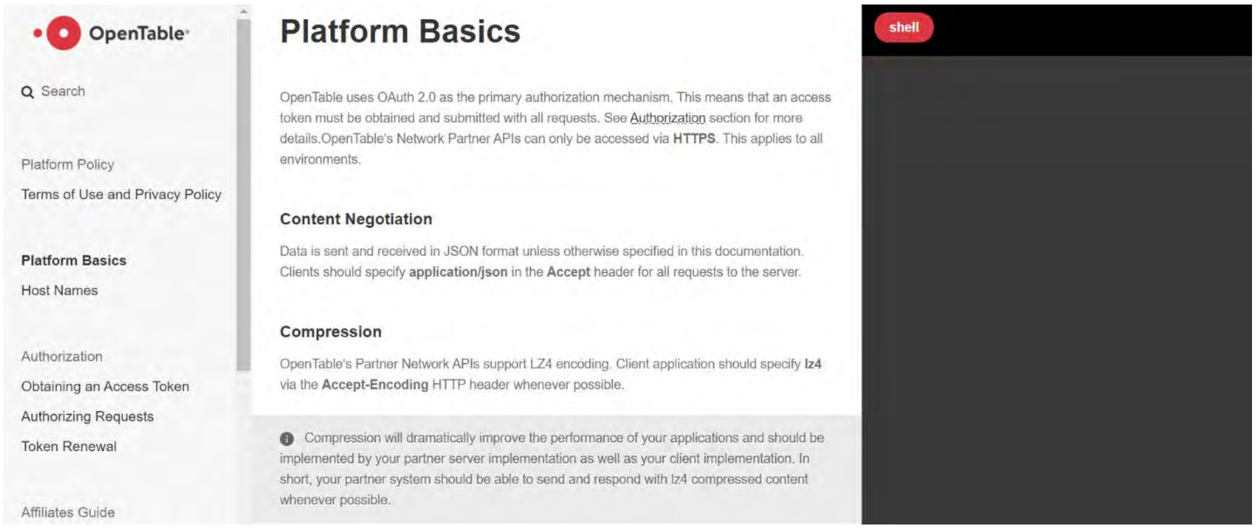
Permitted Use of APIs

You may use the OpenTable application program interface ("**API**") and any data provided through an API only in accordance with your applicable agreement with OpenTable ("**Terms**"), this Platform Policy and OpenTable's applicable API documentation and policies. In this Platform Policy, "you" means the entity which agreed to the Terms with OpenTable. In event of any conflict between this Platform Policy and the Terms, this Platform Policy will control as relates to the API. Otherwise, the Terms continue to apply to the API and any activity under this Platform Policy.

You will only access (or attempt to access) an API by the means described in the documentation for that API. If OpenTable assigns you credentials (e.g. client IDs), you must use them only with the applicable APIs. You may not share your credentials with third-parties or service providers without OpenTable's consent (and if such parties require access you must request individual credentials for them). Your API usage rights are limited, non-exclusive, non-sublicensable and non-transferable.

OpenTable may set and enforce limits on your use of the APIs (e.g. limiting the number of API requests that you may make or the number of users you may serve), in its sole discretion. You agree to, and will not attempt to circumvent, such limitations documented with each API. If you would like to use any API beyond these limits, you must obtain OpenTable's express written consent (and OpenTable may decline such request or condition acceptance on your agreement to additional terms and/or charges for that use).

https://platform.opentable.com/documentation/#platform-policy



Platform Basics

OpenTable uses OAuth 2.0 as the primary authorization mechanism. This means that an access token must be obtained and submitted with all requests. See Authorization section for more details.OpenTable's Network Partner APIs can only be accessed via **HTTPS**. This applies to all environments.

Content Negotiation

Data is sent and received in JSON format unless otherwise specified in this documentation. Clients should specify **application/json** in the **Accept** header for all requests to the server.

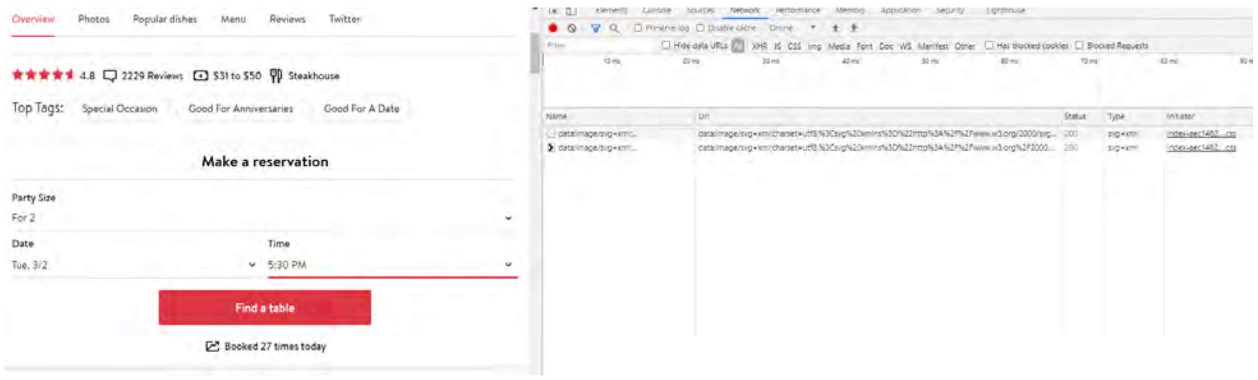
Compression

OpenTable's Partner Network APIs support LZ4 encoding. Client application should specify **lz4** via the **Accept-Encoding** HTTP header whenever possible.

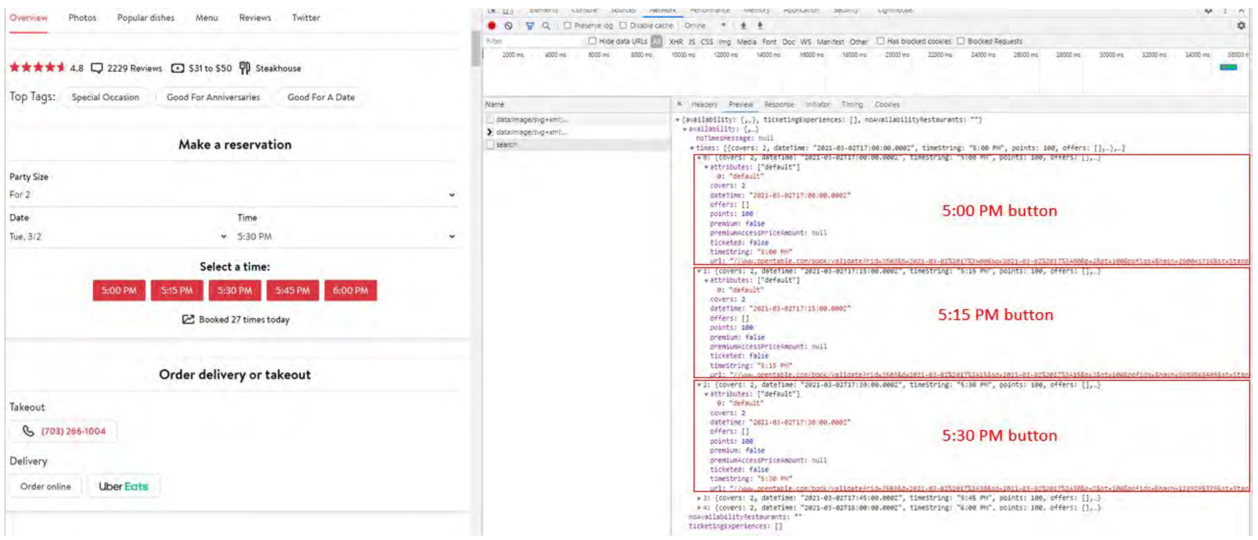
Compression will dramatically improve the performance of your applications and should be implemented by your partner server implementation as well as your client implementation. In short, your partner system should be able to send and respond with lz4 compressed content whenever possible.

https://platform.opentable.com/documentation/#platform-basics

314. Each defined UI object is either selected by a user of an authoring tool or automatically selected by the OpenTable platform as the preferred UI object corresponding to the symbolic name of the web component selected by the user of the authoring tool. For example, the OpenTable platform automatically selects the button as preferred UI objects.

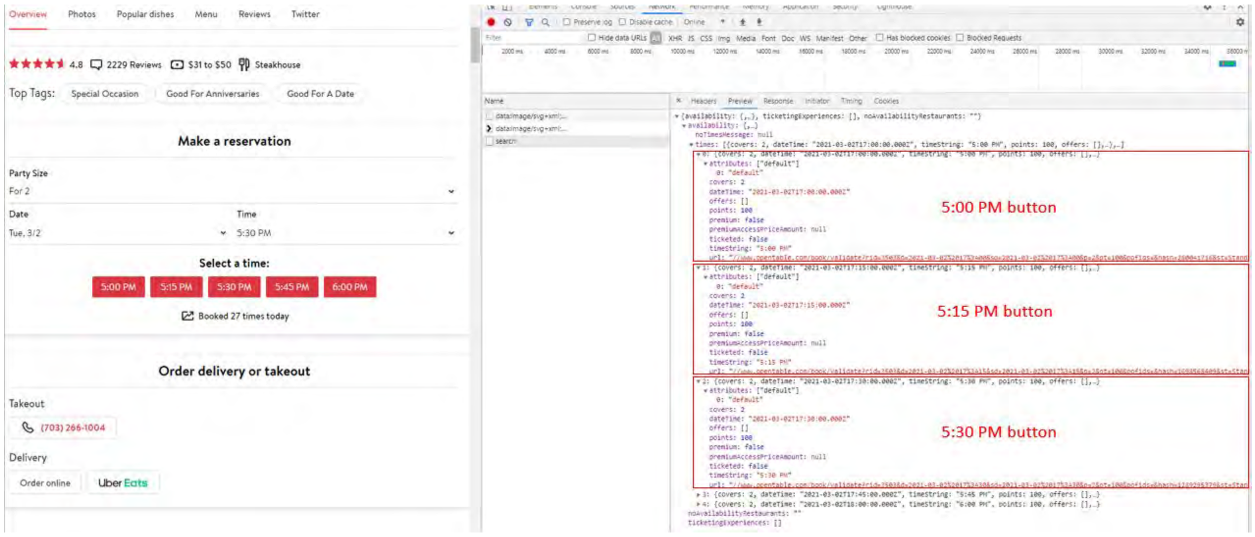


```
data:image/svg+xml;...   data:image/svg+xml;charset=utf8,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org/2000/svg...   200   svg+xml   index-aec1482...css
data:image/svg+xml;...   data:image/svg+xml;charset=utf8,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000...   200   svg+xml   index-aec1482...css
```



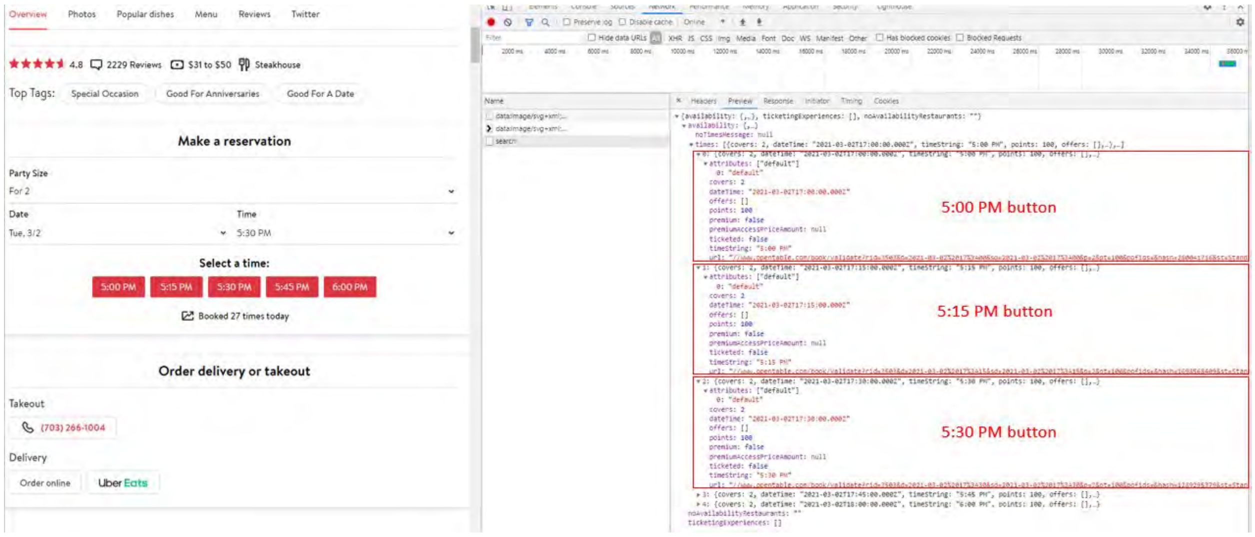
```
▾ {availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾ availability: {,…}
      noTimesMessage: null
    ▾ times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
      ▾ 0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
        ▾ attributes: ["default"]
            0: "default"
          covers: 2
          dateTime: "2021-03-02T17:00:00.000Z"
          offers: []
          points: 100
          premium: false
          premiumAccessPriceAmount: null
          ticketed: false
          timeString: "5:00 PM"
          url: "//www.opentable.com/book/validate?rid=[illegible]"
      ▾ 1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
        ▾ attributes: ["default"]
            0: "default"
          covers: 2
          dateTime: "2021-03-02T17:15:00.000Z"
          offers: []
          points: 100
          premium: false
          premiumAccessPriceAmount: null
          ticketed: false
          timeString: "5:15 PM"
          url: "//www.opentable.com/book/validate?rid=[illegible]"
      ▾ 2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
        ▾ attributes: ["default"]
            0: "default"
          covers: 2
          dateTime: "2021-03-02T17:30:00.000Z"
          offers: []
          points: 100
          premium: false
          premiumAccessPriceAmount: null
          ticketed: false
          timeString: "5:30 PM"
          url: "//www.opentable.com/book/validate?rid=[illegible]"
      ▸ 3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
      ▸ 4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
    noAvailabilityRestaurants: ""
    ticketingExperiences: []
```

315. On information and belief, the authoring tool selects a symbolic name (e.g., JSON data) from the web component corresponding to the defined UI object. In the example below, symbolic name "0" corresponds to the "5:00 PM" button UI object; symbolic name "1" corresponds to the "5:15 PM" button UI object; symbolic name "2" corresponds to the "5:30 PM" button UI object.

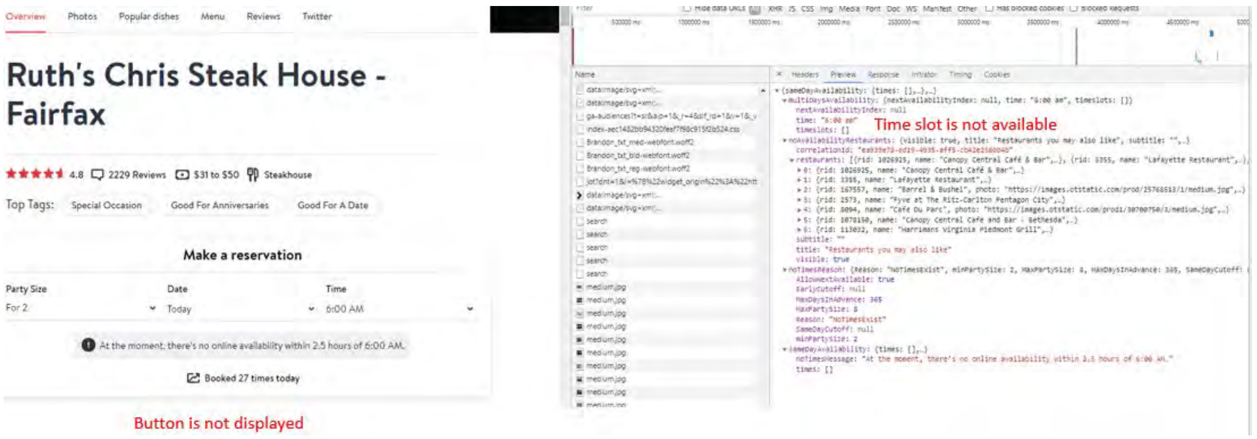


```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
    noTimesMessage: null
   ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
    ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:00:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:00 PM"
      url: "//www.opentable.com/book/validate?rid=[illegible]
    ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:15:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:15 PM"
      url: "//www.opentable.com/book/validate?rid=[illegible]
    ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:30:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:30 PM"
      url: "//www.opentable.com/book/validate?rid=[illegible]
    ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
    ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```

316. The authoring tool associates the selected symbolic name from the web component corresponding to the defined UI object with the defined UI object. In the example below, the authoring tool associates "5:00 PM" button UI object with the symbolic name "0". In particular, the symbolic name "timeString" associates with the value "5:00 PM" that is displayed on the button UI object. When the time slot is not available, the button object is not displayed and the symbolic name "timeString" is not available.



```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
    noTimesMessage: null
   ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
    ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:00:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:00 PM"
      url: "//www.opentable.com/book/validate?rid=[illegible]
    ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:15:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:15 PM"
      url: "//www.opentable.com/book/validate?rid=[illegible]
    ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
     ▾attributes: ["default"]
        0: "default"
      covers: 2
      dateTime: "2021-03-02T17:30:00.000Z"
      offers: []
      points: 100
      premium: false
      premiumAccessPriceAmount: null
      ticketed: false
      timeString: "5:30 PM"
      url: "//www.opentable.com/book/validate?rid=[illegible]
    ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
    ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```


[redacted]

Overview Photos Popular dishes Menu Reviews Twitter

Ruth's Chris Steak House - Fairfax

★★★★★ 4.8 2229 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Good For A Date

Make a reservation

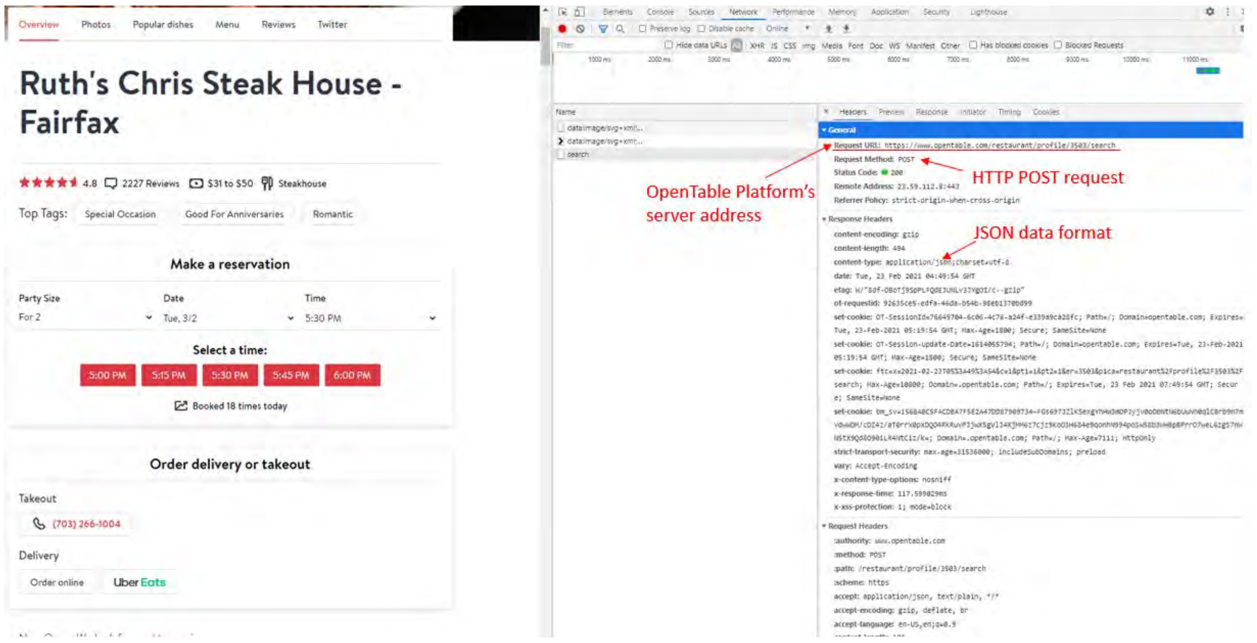
Party Size: For 2 Date: Today Time: 6:00 AM

At the moment, there's no online availability within 2.5 hours of 6:00 AM.

Booked 27 times today

```
▾{sameDayAvailability: {times: [],…},…}
  ▾multiDaysAvailability: {nextAvailabilityIndex: null, time: "6:00 am", timeslots: []}
    nextAvailabilityIndex: null
    time: "6:00 am"
    timeslots: []
  ▾noAvailabilityRestaurants: {visible: true, title: "Restaurants you may also like", subtitle: "",…}
    correlationId: "ea939e7d-ed19-4935-aff5-cb42e2580040"
   ▾restaurants: [{rid: 1026925, name: "Canopy Central Café & Bar",…}, {rid: 3355, name: "Lafayette Restaurant",…},
    ▸0: {rid: 1026925, name: "Canopy Central Café & Bar",…}
    ▸1: {rid: 3355, name: "Lafayette Restaurant",…}
    ▸2: {rid: 167557, name: "Barrel & Bushel", photo: "https://images.otstatic.com/prod/25768513/1/medium.jpg",…}
    ▸3: {rid: 2573, name: "Fyve at The Ritz-Carlton Pentagon City",…}
    ▸4: {rid: 8094, name: "Cafe Du Parc", photo: "https://images.otstatic.com/prod1/30700750/3/medium.jpg",…}
    ▸5: {rid: 1078150, name: "Canopy Central Cafe and Bar - Bethesda",…}
    ▸6: {rid: 113032, name: "Harrimans Virginia Piedmont Grill",…}
    subtitle: ""
    title: "Restaurants you may also like"
    visible: true
  ▾noTimesReason: {Reason: "NoTimesExist", minPartySize: 2, MaxPartySize: 8, MaxDaysInAdvance: 365, SameDayCutoff:
    AllowNextAvailable: true
    EarlyCutoff: null
    MaxDaysInAdvance: 365
    MaxPartySize: 8
    Reason: "NoTimesExist"
    SameDayCutoff: null
    minPartySize: 2
  ▾sameDayAvailability: {times: [],…}
    noTimesMessage: "At the moment, there's no online availability within 2.5 hours of 6:00 AM."
    times: []
```

317. Upon information and belief, the OpenTable platform stored information representative of the selected setting in a database that supporting the restaurant profile. In the example below, the OpenTable platform sends the user selected settings for a reservation, such as party size, reservation date and time through a "search" HTTP POST request to the OpenTable platform's server and stores the information representative of the said user selected settings in an OpenTable platform's database. In particular, the user selected settings are transferred and stored in JSON data format.


[redacted]

Overview Photos Popular dishes Menu Reviews Twitter

Ruth's Chris Steak House - Fairfax

★★★★★ 4.8 2227 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Romantic

Make a reservation

Party Size: For 2 Date: Tue, 3/2 Time: 5:30 PM

Select a time: 5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM

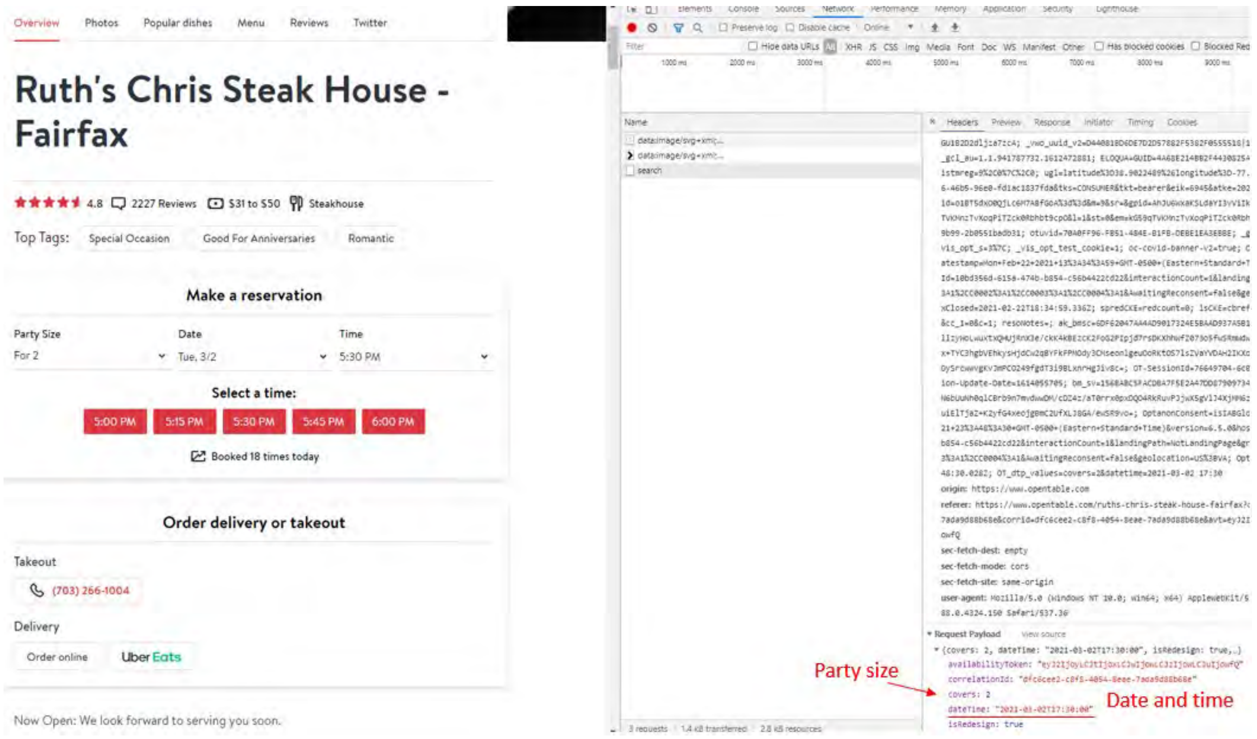
Booked 18 times today

Order delivery or takeout

Takeout: (703) 266-1004

Delivery: Order online Uber Eats

```
▾General
  Request URL: https://www.opentable.com/restaurant/profile/3503/search
  Request Method: POST
  Status Code: ● 200
  Remote Address: 23.59.112.8:443
  Referrer Policy: strict-origin-when-cross-origin
▾Response Headers
  content-encoding: gzip
  content-length: 494
  content-type: application/json;charset=utf-8
  date: Tue, 23 Feb 2021 04:49:54 GMT
  etag: W/"8df-OBoTj9SpPL+QdEJUNLv3JYgOI/c--gzip"
  ot-requestid: 92635ce5-edfa-46da-b54b-98eb1370bd99
  set-cookie: OT-SessionId=76649704-6c06-4c76-a24f-e339a9ca28fc; Path=/; Domain=opentable.com; Expires=
  Tue, 23-Feb-2021 05:19:54 GMT; Max-Age=1800; Secure; SameSite=None
  set-cookie: OT-Session-Update-Date=1614055794; Path=/; Domain=opentable.com; Expires=Tue, 23-Feb-2021
  05:19:54 GMT; Max-Age=1800; Secure; SameSite=None
  set-cookie: ftc=x=2021-02-23T05%3A49%3A54&c=1&pt1=1&pt2=1&er=3503&p1ca=restaurant%2Fprofile%2F3503%2F
  search; Max-Age=10800; Domain=.opentable.com; Path=/; Expires=Tue, 23 Feb 2021 07:49:54 GMT; Secur
  e; SameSite=None
  set-cookie: bm_sv=156BABC5FACDBA7F5E2A47DD87909734~FGS6973ZlKSexgYhHw3mDPJyjV0oDeNtN6bUAVh0qlCBrb9n7m
  VdwWDM/cDZ4Z/aT0rrX0pXDQO4#KAuVPJjwXSgVlJ4XjMM6I7CjZ9KoO3H684e9QonhN994poSW58b3H4Bg8FrrO7weL6ZgS7mW
  NStX9Qd8O90iLR4NtCiz/k=; Domain=.opentable.com; Path=/; Max-Age=7111; HttpOnly
  strict-transport-security: max-age=31536000; includeSubDomains; preload
  vary: Accept-Encoding
  x-content-type-options: nosniff
  x-response-time: 117.599029ms
  x-xss-protection: 1; mode=block
▾Request Headers
  :authority: www.opentable.com
  :method: POST
  :path: /restaurant/profile/3503/search
  :scheme: https
  accept: application/json, text/plain, */*
  accept-encoding: gzip, deflate, br
  accept-language: en-US,en;q=0.9
```



Ruth's Chris Steak House - Fairfax

★★★★★ 4.8 2227 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Romantic

Make a reservation

Party Size: For 2
Date: Tue, 3/2
Time: 5:30 PM

Select a time:
5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM

Booked 18 times today

Order delivery or takeout

Takeout
(703) 266-1004

Delivery
Order online Uber Eats

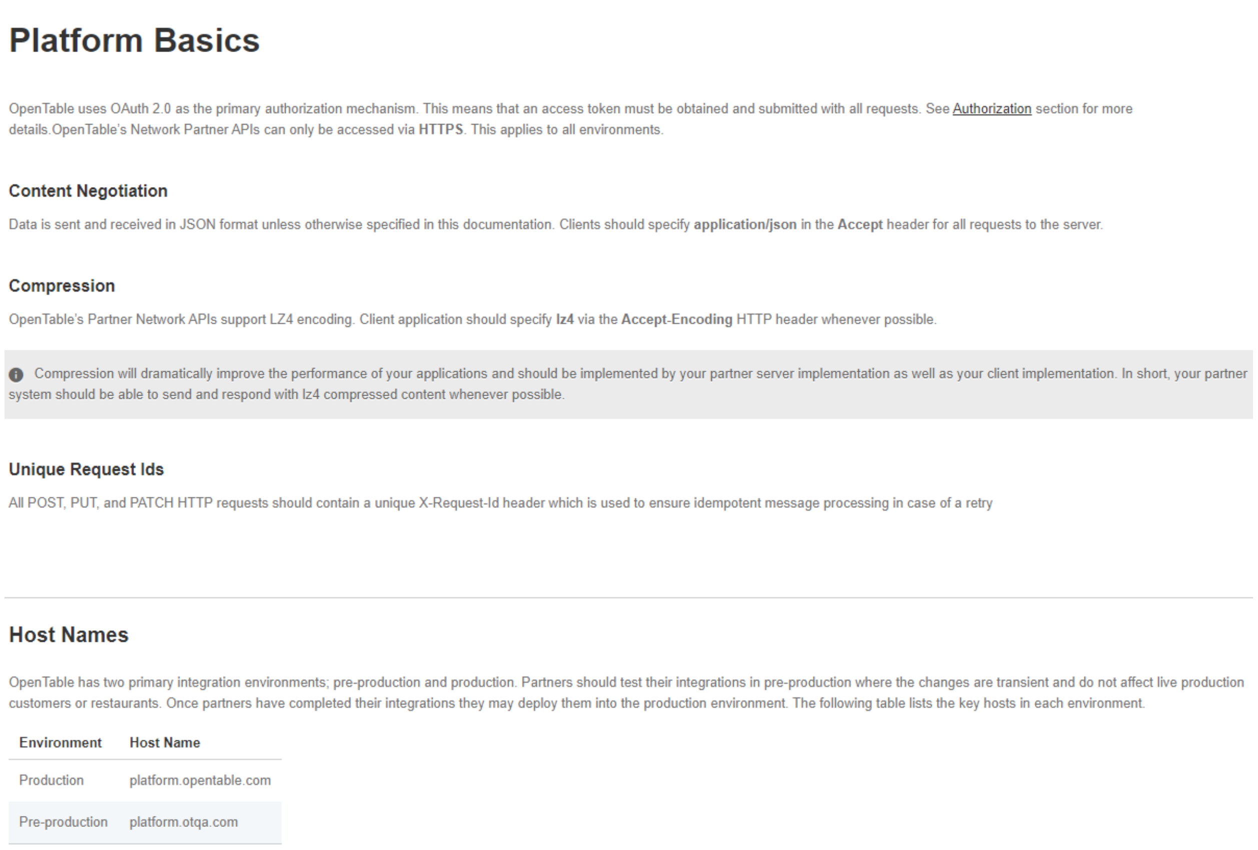
Now Open: We look forward to serving you soon.

[redacted]

```
data:image/svg+xml;...
data:image/svg+xml;...
search
```

```
GU1B2D2dljza7zcA; _vwo_uuid_v2=D44081BD6DE7D2D57882F5382F0555516|1
_gcl_au=1.1.941787732.1612472881; ELOQUA=GUID=4A68E214B82F443082S4
istmreg=9%2C0%7C%2C0; ugl=latitude%3D38.9022489%26longitude%3D-77.
6-46b5-96e0-fd1ac1837fda&tks=CONSUMER&tkt=bearer&eik=6945&atke=202
id=018T5dXO0QjLC6M7A8fGOA%3d%3d&m=9&sr=&gpid=AhJU6WXaKSLdaYI3VV1Ik
TVKMnzTvXoqPiTZck0Rbhbt9cpO&l=1&st=0&em=kG59qTVKMnzTvXoqPiTZck0Rbh
9b99-2b0551badb31; otuvid=70A0FF96-FB51-484E-81FB-DEBE1EA3EBBE; _g
vis_opt_s=3%7C; _vis_opt_test_cookie=1; oc-covid-banner-v2=true; C
atestamp=Mon+Feb+22+2021+13%3A34%3A59+GMT-0500+(Eastern+Standard+T
Id=10bd356d-6158-474b-b854-c56b4422cd22&interactionCount=1&landing
3A1%2CC0002%3A1%2CC0003%3A1%2CC0004%3A1&AwaitingReconsent=false&ge
xClosed=2021-02-22T18:34:59.336Z; spredCKE=redcount=0; lsCKE=cbref
&cc_1=0&c=1; resoNotes=; ak_bmsc=6DF62047AA4AD9017324E5BAAD937A5B1
lIzyHoLwuKtXQHUjRnX3e/cKK4kBEzCK2FoG2PIpjd7rsDKXhhwfZ0730SfwSRmadw
x+TYC3hgbVEhkysHjdCw2q8YFkFPMOdy3CHseon1geuDoRKtOS71sZVaYVDAH2IKXc
OySrcwwVgKVJmPCO249fgdT3i9BLxnrHgJivBc=; OT-SessionId=76649704-6c8
ion-Update-Date=1614055705; bm_sv=1568A8C5FACD8A7F5E2A47DD87909734
N6bUuNh0qlCBrb9n7mvdwwDM/cDZ4z/aT0rrx0pxDQO4RkRuvPJjwX5gVlJ4XjMM6z
uiElTjaZ+K2yfG4xeojgBmC2UfXLJ8GA/ewSR9vo=; OptanonConsent=isIABGlo
21+23%3A48%3A30+GMT-0500+(Eastern+Standard+Time)&version=6.5.0&hos
b854-c56b4422cd22&interactionCount=1&landingPath=NotLandingPage&gr
3%3A1%2CC0004%3A1&AwaitingReconsent=false&geolocation=US%3BVA; Opt
48:30.028Z; OT_dtp_values=covers=2&datetime=2021-03-02 17:30
origin: https://www.opentable.com
referer: https://www.opentable.com/ruths-chris-steak-house-fairfax?
7ada9d88b68e&corrid=dfc6cee2-c8f8-4054-8eae-7ada9d88b68e&avt=eyJ2I
owfQ
sec-fetch-dest: empty
sec-fetch-mode: cors
sec-fetch-site: same-origin
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/5
88.0.4324.150 Safari/537.36
▾ Request Payload    view source
  ▾ {covers: 2, dateTime: "2021-03-02T17:30:00", isRedesign: true,…}
      availabilityToken: "eyJ2IjoyLCJtIjoxLCJwIjowLCJzIjowLCJuIjowfQ"
      correlationId: "dfc6cee2-c8f8-4054-8eae-7ada9d88b68e"
      covers: 2
      dateTime: "2021-03-02T17:30:00"
      isRedesign: true
```

318. The OpenTable platform retrieves information representative of one or more UI object settings stored in the database. As shown in the example below, the OpenTable platform could load the restaurant profile on a browser by sending an HTTP GET request to fetch the stored settings information (e.g., overview, photos, popular dishes, reservation, menu, and reviews of the restaurant) from the database in the OpenTable's server. In particular, the OpenTable platform stored the images of the restaurant (e.g., restaurant profile image) along with other user selected settings, which it in turn communicated from the server as a response to an HTTP GET request. On information and belief, these functionalities were present during relevant time periods of infringement.



Platform Basics

OpenTable uses OAuth 2.0 as the primary authorization mechanism. This means that an access token must be obtained and submitted with all requests. See Authorization section for more details.OpenTable's Network Partner APIs can only be accessed via **HTTPS**. This applies to all environments.

Content Negotiation

Data is sent and received in JSON format unless otherwise specified in this documentation. Clients should specify **application/json** in the **Accept** header for all requests to the server.

Compression

OpenTable's Partner Network APIs support LZ4 encoding. Client application should specify **lz4** via the **Accept-Encoding** HTTP header whenever possible.

Compression will dramatically improve the performance of your applications and should be implemented by your partner server implementation as well as your client implementation. In short, your partner system should be able to send and respond with lz4 compressed content whenever possible.

Unique Request Ids

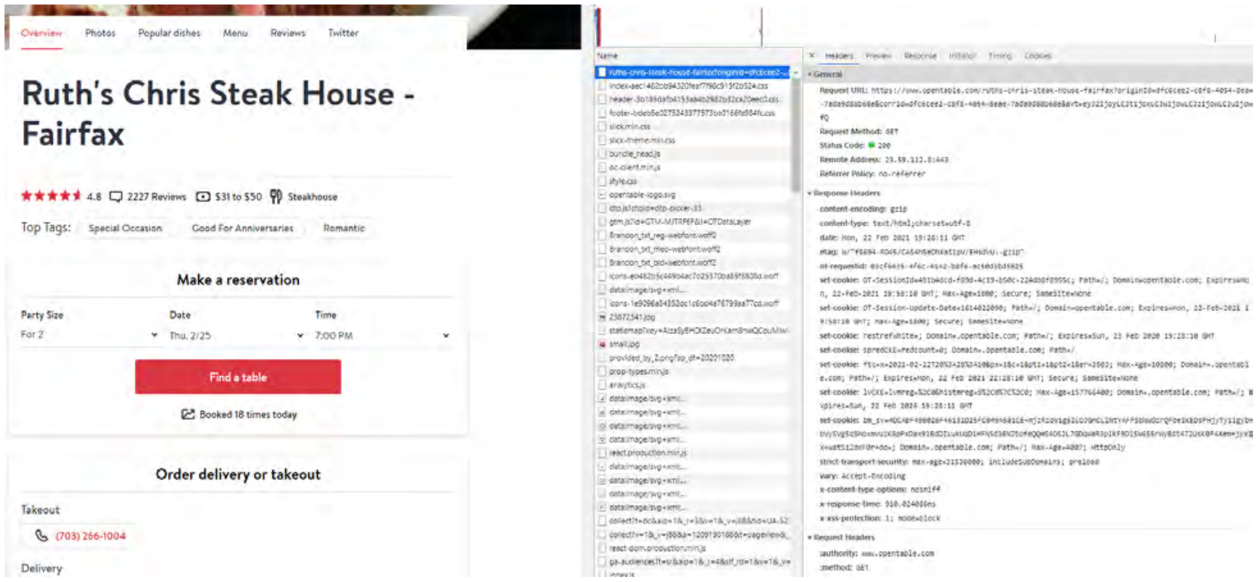
All POST, PUT, and PATCH HTTP requests should contain a unique X-Request-Id header which is used to ensure idempotent message processing in case of a retry

Host Names

OpenTable has two primary integration environments; pre-production and production. Partners should test their integrations in pre-production where the changes are transient and do not affect live production customers or restaurants. Once partners have completed their integrations they may deploy them into the production environment. The following table lists the key hosts in each environment.

Environment	Host Name
Production	platform.opentable.com
Pre-production	platform.otqa.com

https://platform.opentable.com/documentation/?shell#platform-basics



Ruth's Chris Steak House - Fairfax

★★★★★ 4.8 · 2227 Reviews · $31 to $50 · Steakhouse

Top Tags: Special Occasion · Good For Anniversaries · Romantic

Make a reservation

Party Size	Date	Time
For 2	Thu, 2/25	7:00 PM

Booked 18 times today

Order delivery or takeout

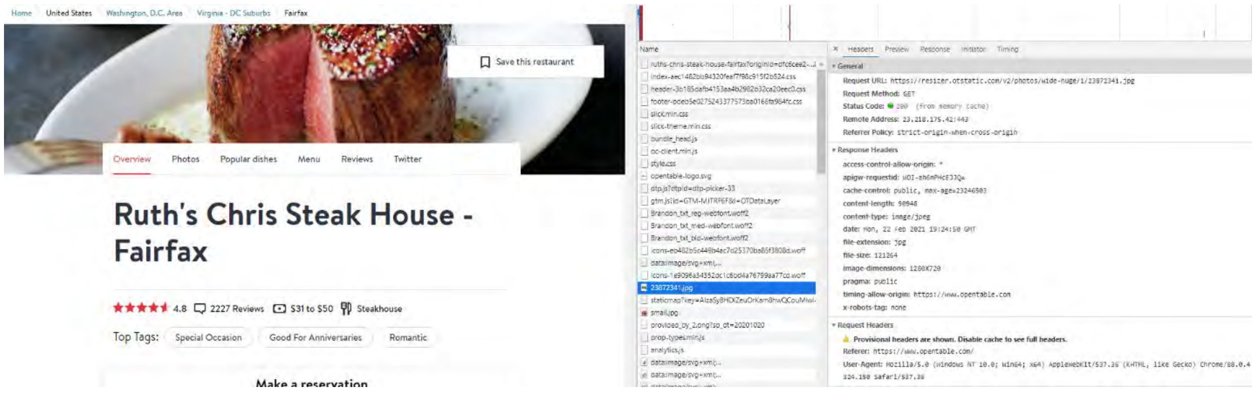
Takeout

(703) 266-1004

Delivery

```
Name
ruths-chris-steak-house-fairfax?originId=dfc6cee2-...
index-aec1482cb94320feaf7f90c915f2b524.css
header-3b186daf54153ab4b2982b32ca20eec3.css
footer-bdeb5e327524337 7573ba0166fa984fc.css
slick.min.css
slick-theme.min.css
bundle_head.js
oc-client.min.js
style.css
opentable-logo.svg
dtp.js?dtpid=dtp-picker-33
gtm.js?id=GTM-MJTRPEF&l=OTDataLayer
Brandon_txt_reg-webfont.woff2
Brandon_txt_med-webfont.woff2
Brandon_txt_bld-webfont.woff2
icons-e0482b5c449b4ac7c25370ba89f5828d.woff
data:image/svg+xml...
icons-1e9096a84352ac1c6bd4a76799aa77cd.woff
23872341.jpg
staticmap?key=AIzaSyBHDXZeuOrKam8wQCouMiw-
small.jpg
provided_by_2.png?so_dt=20201020
prop-types.min.js
analytics.js
data:image/svg+xml...
data:image/svg+xml...
data:image/svg+xml...
data:image/svg+xml...
react.production.min.js
data:image/svg+xml...
data:image/svg+xml...
data:image/svg+xml...
data:image/svg+xml...
collect?t=dc&aip=1&_r=3&v=1&_v=j88&tid=UA-52
collect?v=1&_v=j88&a=1209190188&t=pageview&_
react-dom.production.min.js
ga-audiences?t=sr&aip=1&_r=4&slf_rd=1&v=1&_v=
index.js
```

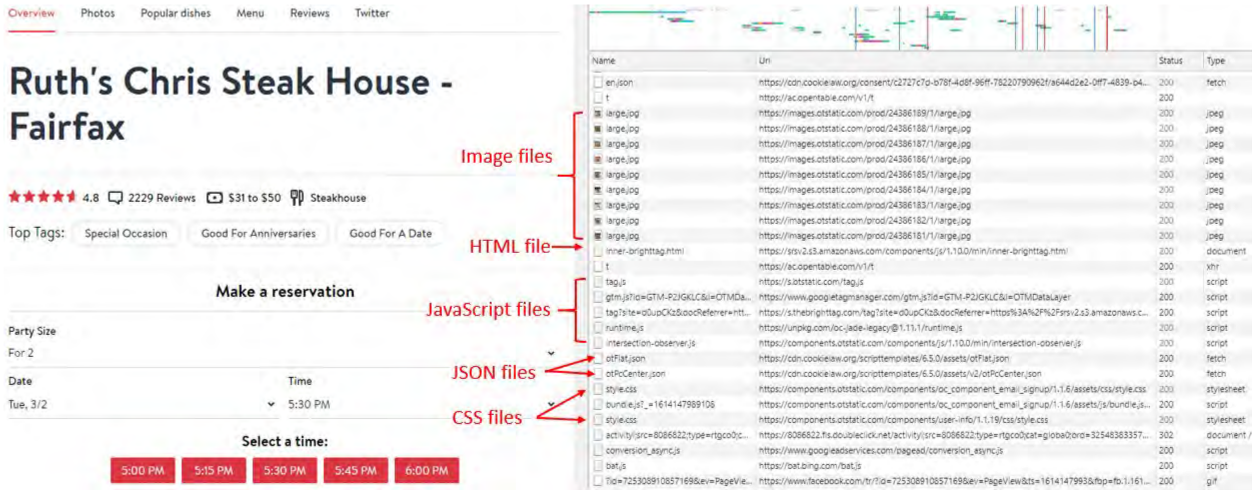
```
▾ General
  Request URL: https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8ea
  -7ada9d88b68e&corrid=dfc6cee2-c8f8-4054-8eae-7ada9d88b68e&avt=eyJ2IjoyLCJtIjoxLCJwIjowLCJ2IjowLCJuIjow
  fQ
  Request Method: GET
  Status Code: ● 200
  Remote Address: 23.59.112.8:443
  Referrer Policy: no-referrer
▾ Response Headers
  content-encoding: gzip
  content-type: text/html;charset=utf-8
  date: Mon, 22 Feb 2021 19:28:11 GMT
  etag: W/"f8694-RO45/CAS4n6eOhkatIpV/EHsdVU.-gzip"
  ot-requestid: e3cf6e35-4f6c-4142-b8f6-ac5d3bd5925
  set-cookie: OT-SessionId=401b4dcd-f09d-4c19-b50c-224d80f0955c; Path=/; Domain=opentable.com; Expires=Mo
  n, 22-Feb-2021 19:58:10 GMT; Max-Age=1800; Secure; SameSite=None
  set-cookie: OT-Session-Update-Date=1614022090; Path=/; Domain=opentable.com; Expires=Mon, 22-Feb-2021 1
  9:58:10 GMT; Max-Age=1800; Secure; SameSite=None
  set-cookie: restrefWhite=; Domain=.opentable.com; Path=/; Expires=Sun, 23 Feb 2020 19:28:10 GMT
  set-cookie: spredCKI=redcount=0; Domain=.opentable.com; Path=/
  set-cookie: ftc=x=2021-02-22T20%3A28%3A10&px=1&c=1&pt1=1&pt2=1&er=3503; Max-Age=10800; Domain=.opentabl
  e.com; Path=/; Expires=Mon, 22 Feb 2021 22:28:10 GMT; Secure; SameSite=None
  set-cookie: lvCKI=lvmreg=%2C0&histmreg=0%2C0%7C%2C0; Max-Age=157766400; Domain=.opentable.com; Path=/; E
  xpires=Sun, 22 Feb 2026 19:28:11 GMT
  set-cookie: bm_sv=4DCABF498028F46131D25FC0494581CE~mjZRzoV1g8ILO7GMCLlNTYAFFSblwDzrQFoeIkEDsPHjyTy11gyb
  bVySVg5q9Hoxmvu2K8aPxDax91BdDIuAUaDiHFN5d38N7to#eQQm540S2L7GDqwaR3plkF9OlSw6S5rWyBdT4T2UsK0F4Xem+jyX
  X+u8tS12dvF0r+0o=; Domain=.opentable.com; Path=/; Max-Age=4087; HttpOnly
  strict-transport-security: max-age=31536000; includeSubDomains; preload
  vary: Accept-Encoding
  x-content-type-options: nosniff
  x-response-time: 910.024080ms
  x-xss-protection: 1; mode=block
▾ Request Headers
  :authority: www.opentable.com
  :method: GET
```



```
Name
ruths-chris-steak-house-fairfax?originId=dfc6cee2-...
index-aec1482bb94320feaf7f98c915f2b524.css
header-3b185dafb4153aa4b2982b32ca20eec0.css
footer-odeb5e0275243377573ba0166fa984fc.css
slick.min.css
slick-theme.min.css
bundle_head.js
oc-client.min.js
style.css
opentable-logo.svg
dtp.js?dtpid=dtp-picker-33
gtm.js?id=GTM-MJTRF6F&l=OTDataLayer
Brandon_txt_reg-webfont.woff2
Brandon_txt_med-webfont.woff2
Brandon_txt_bld-webfont.woff2
icons-eb482b5c449b4ac7d25370ba85f3808d.woff
data:image/svg+xml;...
icons-1e9096a34352dc1c6bd4a76799aa77cd.woff
23872341.jpg
staticmap?key=AIzaSyBHDlZeuOrKam8hwQCouMiwl-
small.jpg
provided_by_2.png?sp_dt=20201020
prop-types.min.js
analytics.js
data:image/svg+xml;...
data:image/svg+xml;...
```

```
Headers   Preview   Response   Initiator   Timing
▾ General
  Request URL: https://resizer.otstatic.com/v2/photos/wide-huge/1/23872341.jpg
  Request Method: GET
  Status Code: ● 200 (from memory cache)
  Remote Address: 23.218.175.42:443
  Referrer Policy: strict-origin-when-cross-origin
▾ Response Headers
  access-control-allow-origin: *
  apigw-requestid: WOI-ah6nPHcEJJQ=
  cache-control: public, max-age=23246583
  content-length: 98948
  content-type: image/jpeg
  date: Mon, 22 Feb 2021 19:24:58 GMT
  file-extension: jpg
  file-size: 121264
  image-dimensions: 1280X720
  pragma: public
  timing-allow-origin: https://www.opentable.com
  x-robots-tag: none
▾ Request Headers
  ⚠ Provisional headers are shown. Disable cache to see full headers.
  Referer: https://www.opentable.com/
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4
  324.150 Safari/537.36
```

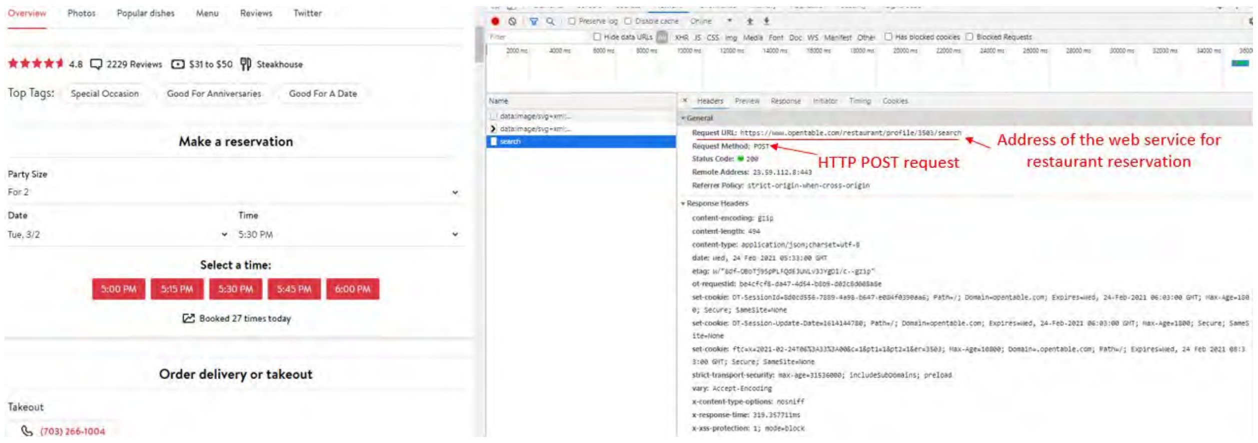
319. The OpenTable platform builds an application including, e.g., the JSON data as well as standard HTML, CSS and Javascript code, consisting of one or more web page views from at least a portion of the database using at least one player, such as a code for a browser engine for a browser, or an operating system or application of a device. See, e.g., Shopify Inc. v. Express Mobile, Inc., Case No. 1:19-cv-00439-RGA, D. Del., D.I. 137. In the example below, the OpenTable platform generates an application for a restaurant profile, in which the OpenTable platform fetches its source code, such as HTML files and run time files (including CSS files, and Javascript files) from the OpenTable platform's database to build a restaurant profile for display.



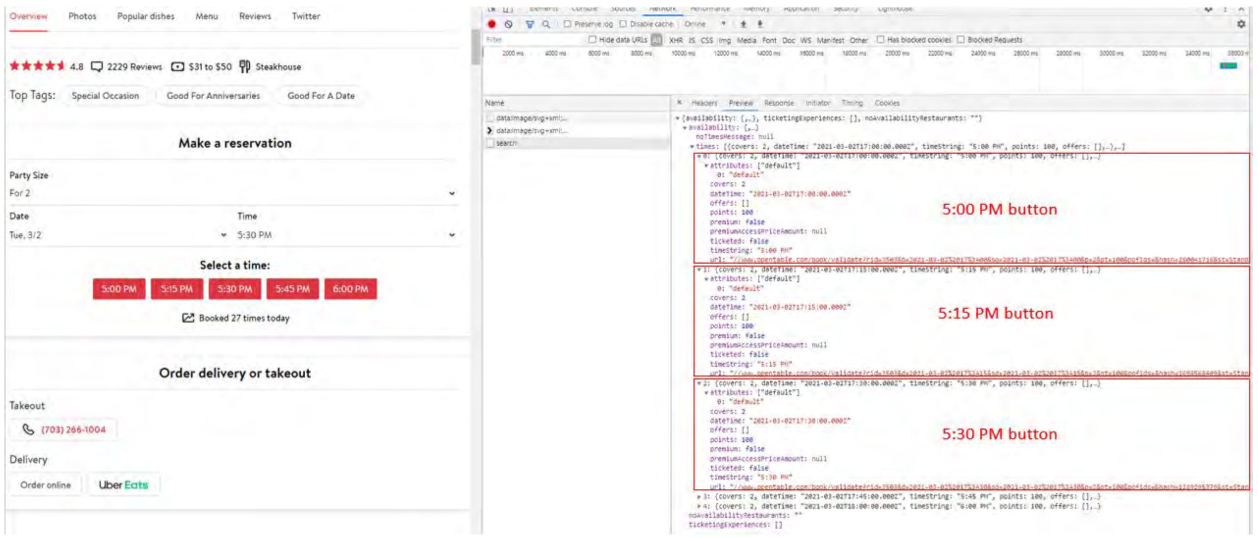
Name	Url	Status	Type
en.json	https://cdn.cookielaw.org/consent/c2727c7d-b78f-4d8f-96ff-78220790962f/a644d2e2-0ff7-4839-b4...	200	fetch
t	https://ac.opentable.com/v1/t	200	
large.jpg	https://images.otstatic.com/prod/24386189/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386188/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386187/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386186/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386185/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386184/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386183/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386182/1/large.jpg	200	jpeg
large.jpg	https://images.otstatic.com/prod/24386181/1/large.jpg	200	jpeg
inner-brighttag.html	https://srsv2.s3.amazonaws.com/components/js/1.10.0/min/inner-brighttag.html	200	document
t	https://ac.opentable.com/v1/t	200	xhr
tag.js	https://s.otstatic.com/tag.js	200	script
gtm.js?id=GTM-P2JGKLC&l=OTMDa...	https://www.googletagmanager.com/gtm.js?id=GTM-P2JGKLC&l=OTMDataLayer	200	script
tag?site=d0upCKz&docReferrer=htt...	https://s.thebrighttag.com/tag?site=d0upCKz&docReferrer=https%3A%2F%2Fsrsv2.s3.amazonaws.c...	200	script
runtime.js	https://unpkg.com/oc-jade-legacy@1.11.1/runtime.js	200	script
intersection-observer.js	https://components.otstatic.com/components/js/1.10.0/min/intersection-observer.js	200	script
otFlat.json	https://cdn.cookielaw.org/scripttemplates/6.5.0/assets/otFlat.json	200	fetch
otPcCenter.json	https://cdn.cookielaw.org/scripttemplates/6.5.0/assets/v2/otPcCenter.json	200	fetch
style.css	https://components.otstatic.com/components/oc_component_email_signup/1.1.6/assets/css/style.css	200	stylesheet
bundle.js?_=1614147989106	https://components.otstatic.com/components/oc_component_email_signup/1.1.6/assets/js/bundle.js...	200	script
style.css	https://components.otstatic.com/components/user-info/1.1.19/css/style.css	200	stylesheet
activity;src=8086822;type=rtgco0;c...	https://8086822.fls.doubleclick.net/activity;src=8086822;type=rtgco0;cat=globa0;ord=32548383357...	302	document /
conversion_async.js	https://www.googleadservices.com/pagead/conversion_async.js	200	script
bat.js	https://bat.bing.com/bat.js	200	script
?id=725308910857169&ev=PageVie...	https://www.facebook.com/tr/?id=725308910857169&ev=PageView&ts=1614147993&fbp=fb.1.161...	200	gif

320. The player utilizes the information stored in the database to generate for the display at least a portion of a web page for the restaurant profile. In the example below, the player utilizes

the information relating to available time slots stored in the database to generate a display of a portion of a web page for a restaurant reservation.



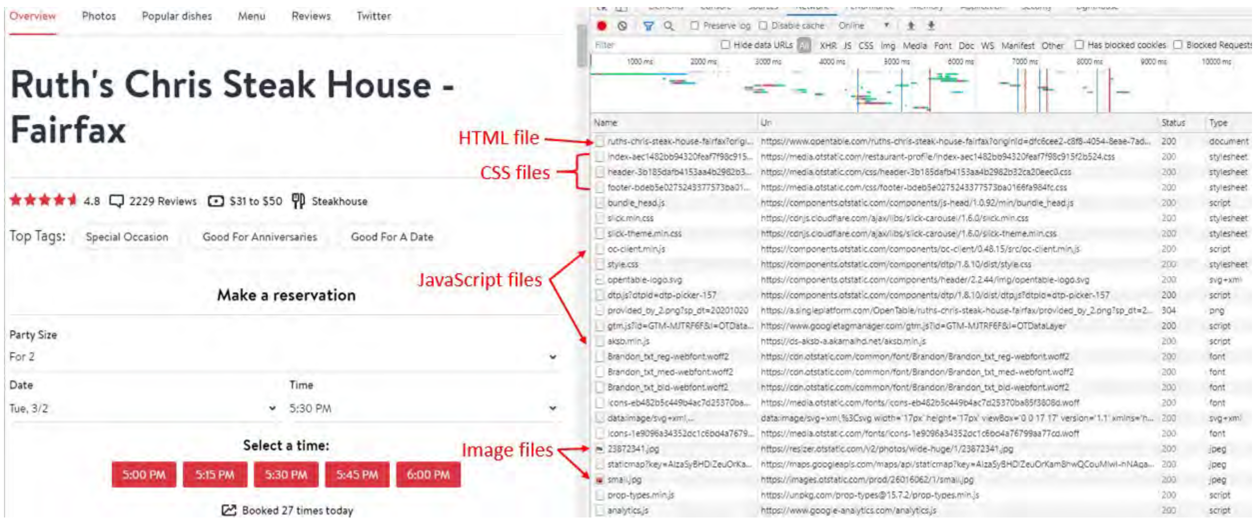
```
▾ General
  Request URL: https://www.opentable.com/restaurant/profile/3503/search
  Request Method: POST
  Status Code: ● 200
  Remote Address: 23.59.112.8:443
  Referrer Policy: strict-origin-when-cross-origin
▾ Response Headers
  content-encoding: gzip
  content-length: 494
  content-type: application/json;charset=utf-8
  date: Wed, 24 Feb 2021 05:33:00 GMT
  etag: W/"8df-OBoTJ9SpPLFQdEJUNLV3JYgDI/c--gzip"
  ot-requestid: be4cfcf8-da47-4d54-b8b9-d02c6d088a6e
  set-cookie: OT-SessionId=8d0cd556-7889-4a98-b647-e084f0390ea6; Path=/; Domain=opentable.com; Expires=Wed, 24-Feb-2021 06:03:00 GMT; Max-Age=180
  0; Secure; SameSite=None
  set-cookie: OT-Session-Update-Date=1614144780; Path=/; Domain=opentable.com; Expires=Wed, 24-Feb-2021 06:03:00 GMT; Max-Age=1800; Secure; SameS
  ite=None
  set-cookie: ftc=x=2021-02-24T06%3A33%3A00&c=1&pt1=1&pt2=1&er=3503; Max-Age=10800; Domain=.opentable.com; Path=/; Expires=Wed, 24 Feb 2021 08:3
  3:00 GMT; Secure; SameSite=None
  strict-transport-security: max-age=31536000; includeSubDomains; preload
  vary: Accept-Encoding
  x-content-type-options: nosniff
  x-response-time: 319.357711ms
  x-xss-protection: 1; mode=block
```



```
▾ {availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾ availability: {,…}
      noTimesMessage: null
    ▾ times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
      ▾ 0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
        ▾ attributes: ["default"]
            0: "default"
          covers: 2
          dateTime: "2021-03-02T17:00:00.000Z"
          offers: []
          points: 100
          premium: false
          premiumAccessPriceAmount: null
          ticketed: false
          timeString: "5:00 PM"
          url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A00&sd=2021-03-02%2017%3A00&p=2&pt=100&pofids=&hash=[illegible]&st=Stand
      ▾ 1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
        ▾ attributes: ["default"]
            0: "default"
          covers: 2
          dateTime: "2021-03-02T17:15:00.000Z"
          offers: []
          points: 100
          premium: false
          premiumAccessPriceAmount: null
          ticketed: false
          timeString: "5:15 PM"
          url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A15&sd=2021-03-02%2017%3A15&p=2&pt=100&pofids=&hash=[illegible]&st=Stan
      ▾ 2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
        ▾ attributes: ["default"]
            0: "default"
          covers: 2
          dateTime: "2021-03-02T17:30:00.000Z"
          offers: []
          points: 100
          premium: false
          premiumAccessPriceAmount: null
          ticketed: false
          timeString: "5:30 PM"
          url: "//www.opentable.com/book/validate?rid=3503&d=2021-03-02%2017%3A30&sd=2021-03-02%2017%3A30&p=2&pt=100&pofids=&hash=[illegible]&st=Stan
      ▸ 3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
      ▸ 4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
    noAvailabilityRestaurants: ""
    ticketingExperiences: []
```

321. The OpenTable platform can execute the application and the player on a device to produce the contents for the restaurant profile, where a user can input a value associated with an input symbolic name to an input of a defined UI object. In the example below, the OpenTable platform converts HTML, CSS, Javascript, image, and other files from an application into an active property listing using a player. The active restaurant profile accepts a search input value (e.g., party size, date, and time). For example, a client can make a reservation for party size of " for 2", date of "Tue, 3/2", and time of "5:30 PM". An input symbolic name "covers" is associated with party size and another input symbolic name "dateTime" is associated with the date and time. These

symbolic names along with the input values are transmitted to the OpenTable API through an HTTP request protocol such as a POST method call.



Ruth's Chris Steak House - Fairfax

4.8 2229 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Good For A Date

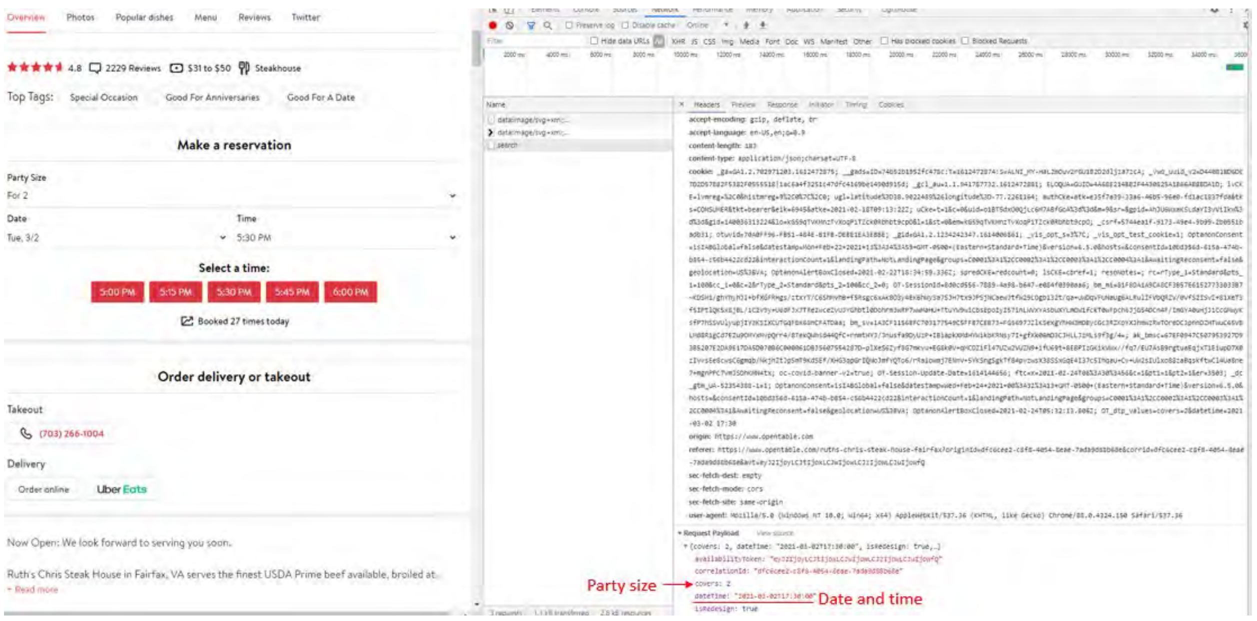
Make a reservation

Party Size: For 2
Date: Tue, 3/2
Time: 5:30 PM

Select a time: 5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM

Booked 27 times today

Name	Url	Status	Type
ruths-chris-steak-house-fairfax?origi...	https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8eae-7ad...	200	document
index-aec1482bb94320feaf7f98c915...	https://media.otstatic.com/restaurant-profile/index-aec1482bb94320feaf7f98c915f2b524.css	200	stylesheet
header-3b185dafb4153aa4b2982b3...	https://media.otstatic.com/css/header-3b185dafb4153aa4b2982b32ca20eec0.css	200	stylesheet
footer-bdeb5e027524337757 3ba01...	https://media.otstatic.com/css/footer-bdeb5e0275243377573ba0166fa984fc.css	200	stylesheet
bundle_head.js	https://components.otstatic.com/components/js-head/1.0.92/min/bundle_head.js	200	script
slick.min.css	https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick.min.css	200	stylesheet
slick-theme.min.css	https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick-theme.min.css	200	stylesheet
oc-client.min.js	https://components.otstatic.com/components/oc-client/0.48.15/src/oc-client.min.js	200	script
style.css	https://components.otstatic.com/components/dtp/1.8.10/dist/style.css	200	stylesheet
opentable-logo.svg	https://components.otstatic.com/components/header/2.2.44/img/opentable-logo.svg	200	svg+xml
dtp.js?dtpId=dtp-picker-157	https://components.otstatic.com/components/dtp/1.8.10/dist/dtp.js?dtpId=dtp-picker-157	200	script
provided_by_2.png?sp_dt=20201020	https://a.singleplatform.com/OpenTable/ruths-chris-steak-house-fairfax/provided_by_2.png?sp_dt=2...	304	png
gtm.js?id=GTM-MJTRF6F&l=OTData...	https://www.googletagmanager.com/gtm.js?id=GTM-MJTRF6F&l=OTDataLayer	200	script
aksb.min.js	https://ds-aksb-a.akamaihd.net/aksb.min.js	200	script
Brandon_txt_reg-webfont.woff2	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_reg-webfont.woff2	200	font
Brandon_txt_med-webfont.woff2	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_med-webfont.woff2	200	font
Brandon_txt_bld-webfont.woff2	https://cdn.otstatic.com/common/font/Brandon/Brandon_txt_bld-webfont.woff2	200	font
icons-eb482b5c449b4ac7d25370ba...	https://media.otstatic.com/fonts/icons-eb482b5c449b4ac7d25370ba85f3808d.woff	200	font
data:image/svg+xml,...	data:image/svg+xml,%3Csvg width='17px' height='17px' viewBox='0 0 17 17' version='1.1' xmlns='h...	200	svg+xml
icons-1e9096a34352dc1c6bd4a7679...	https://media.otstatic.com/fonts/icons-1e9096a34352dc1c6bd4a76799aa77cd.woff	200	font
23872341.jpg	https://resizer.otstatic.com/v2/photos/wide-huge/1/23872341.jpg	200	jpeg
staticmap?key=AIzaSyBHDIZeuOrKa...	https://maps.googleapis.com/maps/api/staticmap?key=AIzaSyBHDIZeuOrKamBhwQCouMlwI-nNAqa...	200	jpeg
small.jpg	https://images.otstatic.com/prod/26016062/1/small.jpg	200	jpeg
prop-types.min.js	https://unpkg.com/prop-types@15.7.2/prop-types.min.js	200	script
analytics.js	https://www.google-analytics.com/analytics.js	200	script



4.8 2229 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Good For A Date

Make a reservation

Party Size: For 2
Date: Tue, 3/2
Time: 5:30 PM

Select a time: 5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM

Booked 27 times today

Order delivery or takeout

Takeout: (703) 266-1004

Delivery: Order online Uber Eats

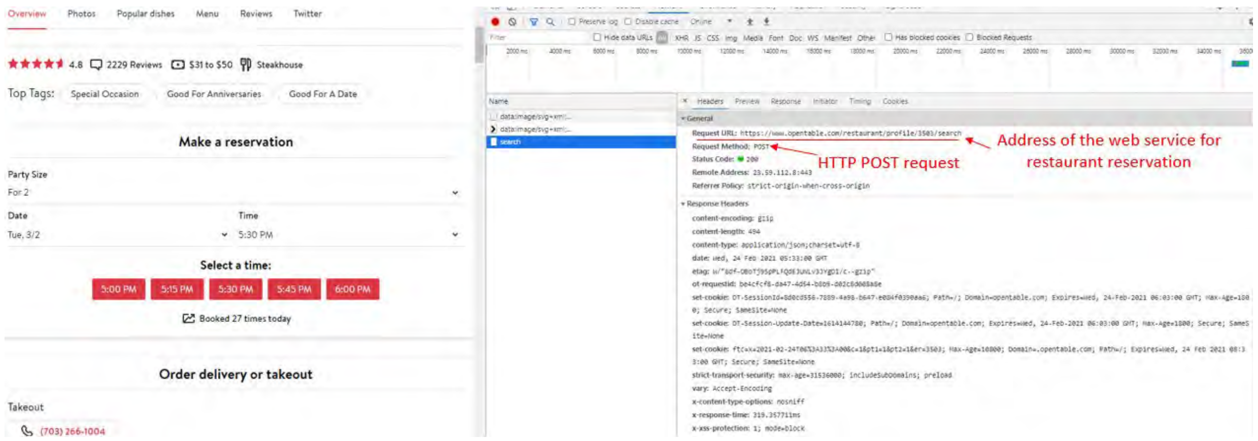
Now Open: We look forward to serving you soon.

Ruth's Chris Steak House in Fairfax, VA serves the finest USDA Prime beef available, broiled at...
+ Read more

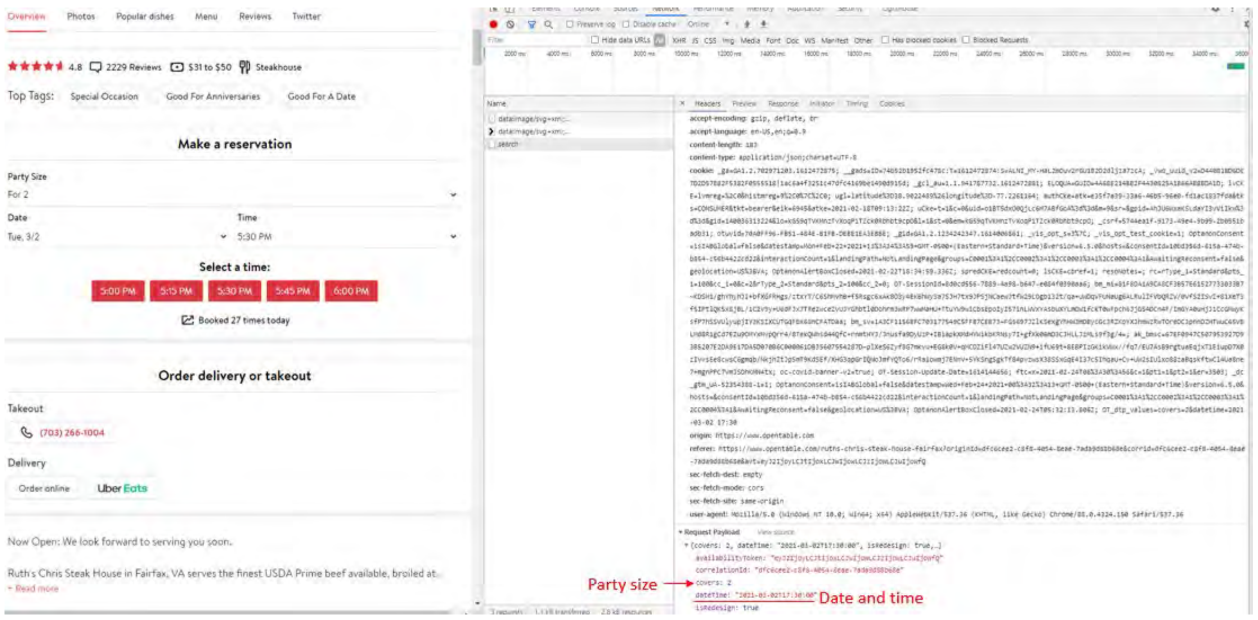
```
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
content-length: 183
content-type: application/json;charset=UTF-8
cookie: _ga=GA1.2.702971203.1612472075; [illegible]
origin: https://www.opentable.com
referer: https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8eae-7ada9d88b68e&corrid=dfc6cee2-c8f8-4054-8eae-7ada9d88b68e&avt=eyJ2IjoyLCJtIjoxLCJwIjowLCJzIjowLCJuIjowfQ
sec-fetch-dest: empty
sec-fetch-mode: cors
sec-fetch-site: same-origin
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36
▾ Request Payload    view source
▾ {covers: 2, dateTime: "2021-03-02T17:30:00", isRedesign: true,…}
    availabilityToken: "eyJ2IjoyLCJtIjoxLCJwIjowLCJzIjowLCJuIjowfQ"
    correlationId: "dfc6cee2-c8f8-4054-8eae-7ada9d88b68e"
    covers: 2
    dateTime: "2021-03-02T17:30:00"
    isRedesign: true
```

322. When the OpenTable platform executes the application and player on the device, the OpenTable platform can cause the device to provide an input value and corresponding input symbolic name to a web service. By way of example, the OpenTable platform causes the device to transmit the input value and a corresponding input symbolic name (e.g., JSON data for the input) to the OpenTable's reservation web service through an HTTP request protocol, such as a POST method call. In the example below, when a client wants to make a reservation for party size of "

for 2", date of "Tue, 3/2", and time of "5:30 PM", the OpenTable platform causes the device to send a JSON data including the symbolic name "covers" associating with value "2" and the symbolic name "covers" associating with the value "2021-03-02T17:30:00" to the OpenTable's reservation web service with an HTTP POST request.



```
Headers  Preview  Response  Initiator  Timing  Cookies
▾ General
   Request URL: https://www.opentable.com/restaurant/profile/3503/search
   Request Method: POST
   Status Code: ● 200
   Remote Address: 23.59.112.8:443
   Referrer Policy: strict-origin-when-cross-origin
▾ Response Headers
   content-encoding: gzip
   content-length: 494
   content-type: application/json;charset=utf-8
   date: Wed, 24 Feb 2021 05:33:00 GMT
   etag: W/"1df-OBoTj9SpPLfQdEJUNLv3JYgDI/c--gzip"
   ot-requestid: be4cfcf8-da47-4d54-b8d6-d02c8d008a5e
   set-cookie: OT-SessionId=8d0cd556-7889-4a98-b647-e084f0390ea6; Path=/; Domain=opentable.com; Expires=Wed, 24-Feb-2021 06:03:00 GMT; Max-Age=180
   0; Secure; SameSite=None
   set-cookie: OT-Session-Update-Date=1614144780; Path=/; Domain=opentable.com; Expires=Wed, 24-Feb-2021 06:03:00 GMT; Max-Age=1800; Secure; SameS
   ite=None
   set-cookie: ftc=x=2021-02-24T06%3A33%3A00&c=1&pt1=1&pt2=1&er=3503; Max-Age=10800; Domain=.opentable.com; Path=/; Expires=Wed, 24 Feb 2021 08:3
   3:00 GMT; Secure; SameSite=None
   strict-transport-security: max-age=31536000; includeSubDomains; preload
   vary: Accept-Encoding
   x-content-type-options: nosniff
   x-response-time: 319.357711ms
   x-xss-protection: 1; mode=block
```

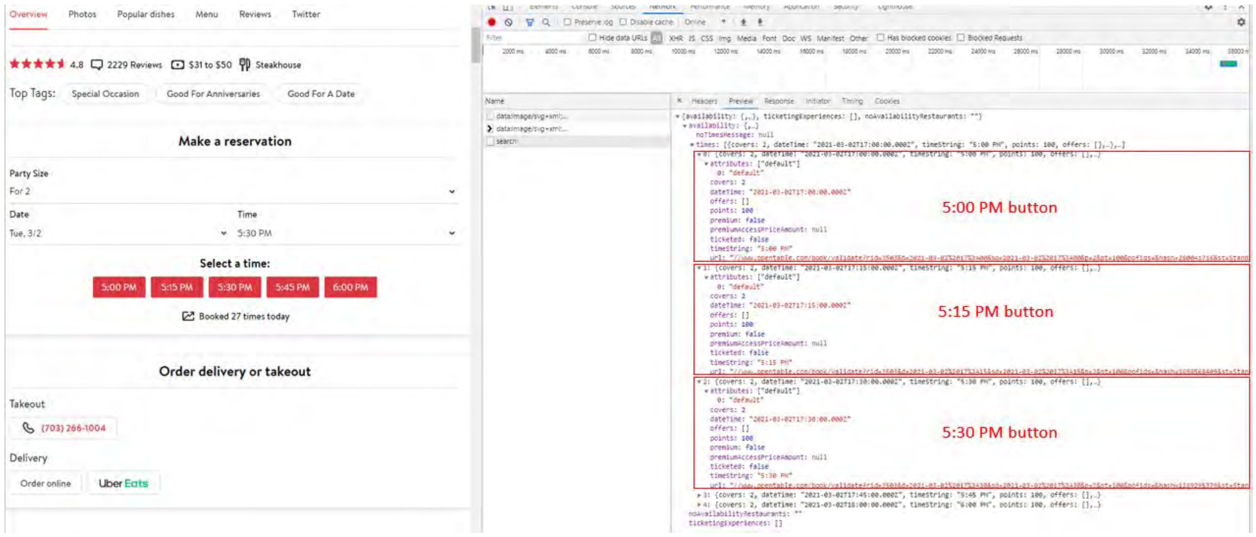


Now Open: We look forward to serving you soon.

Ruth's Chris Steak House in Fairfax, VA serves the finest USDA Prime beef available, broiled at...
+ Read more

```
Headers  Preview  Response  Initiator  Timing  Cookies
   accept-encoding: gzip, deflate, br
   accept-language: en-US,en;q=0.9
   content-length: 183
   content-type: application/json;charset=UTF-8
   cookie: [illegible] OT_dtp_values=covers=2&datetime=2021
   -03-02 17:30
   origin: https://www.opentable.com
   referer: https://www.opentable.com/ruths-chris-steak-house-fairfax?originId=dfc6cee2-c8f8-4054-8eae-7ada9d8b68e&corrid=dfc6cee2-c8f8-4054-8eae
   -7ada9d8b68e&avt=eyJ2IjoyLCJtIjoxLCJwIjowLCJ1IjowLCJuIjowfQ
   sec-fetch-dest: empty
   sec-fetch-mode: cors
   sec-fetch-site: same-origin
   user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36
▾ Request Payload    view source
  ▾ {covers: 2, dateTime: "2021-03-02T17:30:00", isRedesign: true,…}
     availabilityToken: "eyJ2IjoyLCJtIjoxLCJwIjowLCJ1IjowLCJuIjowfQ"
     correlationId: "dfc6cee2-c8f8-4054-8eae-7ada9d8b68e"
     covers: 2
     dateTime: "2021-03-02T17:30:00"
     isRedesign: true
```

323. The OpenTable platform provides the input value and a corresponding input symbolic name to the web service, and in turn causes the web service to generate an output value and a corresponding symbolic name for the output. In the example described below, when a search for "2021-03-02T17:30:00" is input, the OpenTable platform causes the OpenTable's reservation web service to generate five corresponding output available time slots around the date and time in the form of JSON data.



4.8 2229 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Good For A Date

Make a reservation

Party Size: For 2

Date: Tue, 3/2 Time: 5:30 PM

Select a time: 5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM

Booked 27 times today

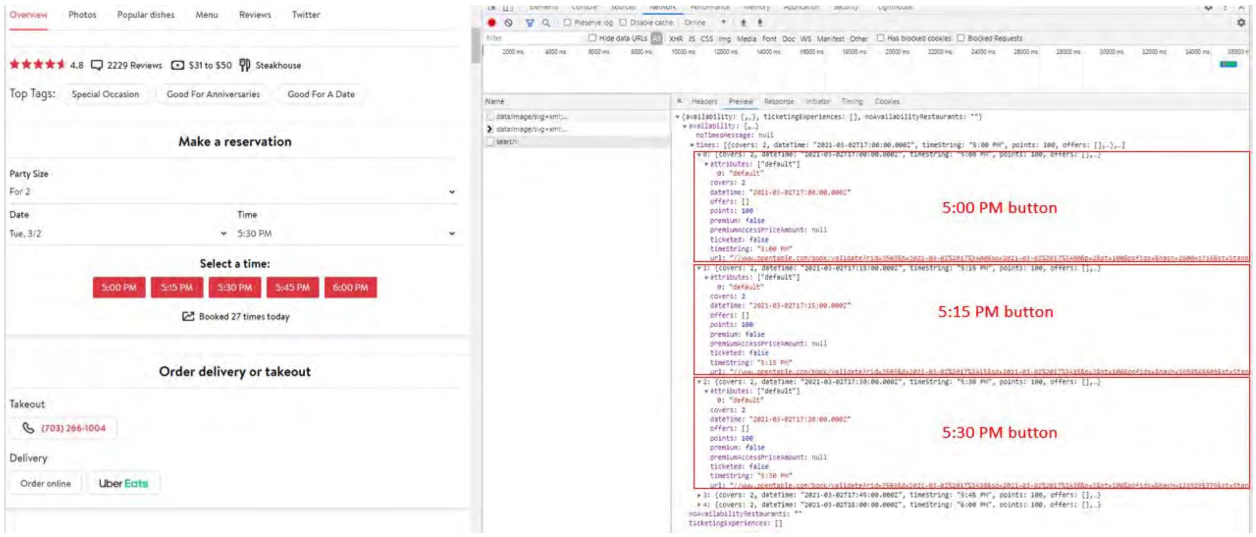
Order delivery or takeout

Takeout: (703) 266-1004

Delivery: Order online Uber Eats

```
▾{availability: {,…}, ticketingExperiences: [], noAvailabilityRestaurants: ""}
  ▾availability: {,…}
    noTimesMessage: null
   ▾times: [{covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…},…]
    ▾0: {covers: 2, dateTime: "2021-03-02T17:00:00.000Z", timeString: "5:00 PM", points: 100, offers: [],…}
      ▾attributes: ["default"]
         0: "default"
        covers: 2
        dateTime: "2021-03-02T17:00:00.000Z"
        offers: []
        points: 100
        premium: false
        premiumAccessPriceAmount: null
        ticketed: false
        timeString: "5:00 PM"
        url: "//www.opentable.com/book/validate?rid=[illegible]
    ▾1: {covers: 2, dateTime: "2021-03-02T17:15:00.000Z", timeString: "5:15 PM", points: 100, offers: [],…}
      ▾attributes: ["default"]
         0: "default"
        covers: 2
        dateTime: "2021-03-02T17:15:00.000Z"
        offers: []
        points: 100
        premium: false
        premiumAccessPriceAmount: null
        ticketed: false
        timeString: "5:15 PM"
        url: "//www.opentable.com/book/validate?rid=[illegible]
    ▾2: {covers: 2, dateTime: "2021-03-02T17:30:00.000Z", timeString: "5:30 PM", points: 100, offers: [],…}
      ▾attributes: ["default"]
         0: "default"
        covers: 2
        dateTime: "2021-03-02T17:30:00.000Z"
        offers: []
        points: 100
        premium: false
        premiumAccessPriceAmount: null
        ticketed: false
        timeString: "5:30 PM"
        url: "//www.opentable.com/book/validate?rid=[illegible]
    ▸3: {covers: 2, dateTime: "2021-03-02T17:45:00.000Z", timeString: "5:45 PM", points: 100, offers: [],…}
    ▸4: {covers: 2, dateTime: "2021-03-02T18:00:00.000Z", timeString: "6:00 PM", points: 100, offers: [],…}
  noAvailabilityRestaurants: ""
  ticketingExperiences: []
```

324. The player receives the output symbolic name and output value and provides instructions to the display of the device to present the output value in the API. In the example below, the player provides instructions to the browser of the device to display five output button UI objects.


Overview Photos Popular dishes Menu Reviews Twitter

4.8 2229 Reviews $31 to $50 Steakhouse

Top Tags: Special Occasion Good For Anniversaries Good For A Date

Make a reservation

Party Size: For 2

Date: Tue, 3/2 Time: 5:30 PM

Select a time: 5:00 PM 5:15 PM 5:30 PM 5:45 PM 6:00 PM

Booked 27 times today

Order delivery or takeout

Takeout: (703) 266-1004

Delivery: Order online Uber Eats

5:00 PM button
5:15 PM button
5:30 PM button

325. The presence of the above referenced features is demonstrated, by way of example, by testing the Accused Instrumentalities for investigative purposes on https://www.booking.com/, https://join.booking.com/, https://partner.booking.com/en-us, and/or https://ycs.agoda.com/en-us/kipp/public/home, and by reference to publicly available information, including the following:

- https://www.booking.com/,

- https://join.booking.com/,
- https://partner.booking.com/en-us,
- https://partner.booking.com/en-us/help/working-booking/how-do-i-join-bookingcom,
- https://partner.booking.com/en-us/help/working-booking/how-can-i-set-my-property-easily,
- https://ycs.agoda.com/en-us/kipp/public/home,
- https://www.agoda.com/info/ycs-online-registration.html?cid=1844104,
- https://www.agoda.com/info/privacy.html?cid=1844104,
- https://partners.agoda.com/en-us/faq.html,
- https://agodapropertyhelp.zendesk.com/hc/en-us,
- https://partners.agoda.com/,
- https://agodapropertyhelp.zendesk.com/hc/en-us/articles/115009545508-How-do-I-list-my-property-on-Agoda-com-,
- https://www.opentable.com/,
- https://support.opentable.com/s/?language=en_US,
- https://platform.opentable.com/documentation/, and
- https://restaurant.opentable.com/.

326. On information and belief, Defendants have had knowledge of the ‘044 patent and their infringement thereof at least as early as October 3, 2019, and no later than November 30, 2020, when Plaintiff provided notice of the ‘044 patent and Defendants' infringement of the ‘044 patent. Furthermore, Defendants have been aware of the ‘044 patent and their infringement thereof since at least the filing of the original Complaint, D.I. 1, on December 1, 2020.

327. On information and belief, Defendants have contributed and are contributing to the infringement of the ‘044 patent because Defendants know that the infringing aspects of their infringing products and services, including but not limited to the Accused Instrumentalities, are made for use in an infringement, and are not staple articles of commerce suitable for substantial

non-infringing uses.

328. On information and belief, Defendants have induced and are inducing the infringement of the ‘044 patent, with knowledge of the ‘044 patent and that their acts, including without limitation using, offering to sell, selling within, and importing into the United States, the Accused Instrumentalities, would aid and abet and induce infringement by customers, clients, partners, developers, and end users of the foregoing.

329. In particular, Defendants actions that aid and abet others such as customers, clients, partners, developers, and end users to infringe include advertising and distributing the Accused Instrumentalities, providing instructional materials, training, and other services regarding the Accused Instrumentalities, and providing free listings for the Accused Instrumentalities. Defendants actively encouraged the adoption of the Accused Instrumentalities and provided support sites for the vast network of developers working with the Accused Instrumentalities, emphasizing the simple and user-friendly nature of the Accused Instrumentalities, for example, explaining that “Registration can take as little as 15 minutes to complete—get started today” and that Defendants provides “24/7 support by phone or email” (*see, e.g.,* https://join.booking.com/), that “Through one platform our hotel partners are distributed across both priceline.com and agoda.com maximizing performance across our Retail, Private, Opaque and Vacation Packages programs. Our partners can easily set rates, promotions, and change hotel details with a few simple clicks of a mouse or on the go with a mobile device” (*see, e.g.,* https://ycs.agoda.com/en-us/kipp/public/home), and that “From online ordering and takeout to powerful marketing and experiences, make more money when you access our network of millions” and that restaurants can “[g]et discovered and capture the business of the millions of people, around the world and in your neighborhood, searching on OpenTable” (*see, e.g.,* https://restaurant.opentable.com/?utm_source=dinersite&utm_medium=referral&utm_campaign=topnav&Lead.LeadSource=DinerSite&Lead.Marketing_ID__c=topnav). On information and belief, Defendants have engaged in such actions with specific intent to cause infringement or with willful blindness to the resulting infringement because Defendants have had actual knowledge of

the ‘044 patent and knowledge that their acts were inducing infringement of the ‘044 patent since at least the date Defendants received notice that their activities infringed the ‘044 patent.

330. Defendants' acts of infringement have caused damage to Plaintiff, and Plaintiff is entitled to recover damages from Defendants in an amount subject to proof at trial.

331. Defendants' infringement of Plaintiff’s rights under the ‘044 patent will continue to damage Plaintiff’s business, causing irreparable harm, for which there is no adequate remedy at law, unless enjoined by this Court.

332. On information and belief, Defendants have acted with disregard of Plaintiff’s patent rights, without any reasonable basis for doing so, and have willfully infringed and do willfully infringe the ‘044 patent.

333. The foregoing is illustrative of Defendants' infringement of the ‘044 patent. Plaintiff reserves the right to identify additional claims and Accused Instrumentalities in accordance with the Court’s local rules and applicable scheduling orders.

<u>PRAYER FOR RELIEF</u>

WHEREFORE, Plaintiff prays for the following relief:

A. A judgment that U.S. Patent Nos. 6,546,397, 7,594,168, 9,063,755, 9,471,287, and 9,928,044 are valid and enforceable;

B. A judgment that Defendants have directly infringed, contributorily infringed, and/or induced the infringement of U.S. Patent Nos. 6,546,397, 7,594,168, 9,063,755, 9,471,287, and 9,928,044;

C. A judgment that Defendants' infringement of U.S. Patent Nos. 6,546,397, 7,594,168, 9,063,755, 9,471,287, and 9,928,044 has been willful;

D. An award of attorneys’ fees incurred in prosecuting this action, on the basis that this is an exceptional case;

E. A judgment and order requiring Defendants to pay Plaintiff damages under 35 U.S.C. § 284, including supplemental damages for any continuing post-verdict infringement up until entry of the final judgment, with an accounting, as needed, and treble damages for willful infringement as provided by 35 U.S.C. § 284;

F. A judgment and order requiring Defendants to pay Plaintiff the costs of this action (including all disbursements);

G. A judgment and order requiring Defendants to pay Plaintiff pre-judgment and post-judgment interest on the damages awarded;

H. A judgment and order requiring that Plaintiff be awarded a compulsory ongoing license fee; and

I. Such other relief as the Court may deem just and proper.

Dated: February 26, 2021

Respectfully submitted,

/s/ Ramon A. Miyar

Steven J. Rizzi (*pro hac vice* forthcoming)
Ramy Hanna (*pro hac vice* forthcoming)
Ryan A. Schmid (*pro hac vice* forthcoming)
Ramon A. Miyar

Attorneys for Plaintiff Express Mobile, Inc.

DEMAND FOR JURY TRIAL

Pursuant to Rule 38(b) of the Federal Rules of Civil Procedure and Civil L.R. 3-6(a), Express Mobile respectfully demands a trial by jury on all issues triable by Jury.

Dated: February 26, 2021

Respectfully submitted,

/s/ Ramon A. Miyar

Steven J. Rizzi (admitted *pro hac vice*)
Ramy Hanna (*pro hac vice* forthcoming)
Ryan A. Schmid (*pro hac vice* forthcoming)
Ramon A. Miyar

Attorneys for Plaintiff Express Mobile, Inc.

EXHIBIT A

US006546397B1

(12) **United States Patent**
Rempell

(10) **Patent No.: US 6,546,397 B1**
(45) **Date of Patent: Apr. 8, 2003**

(54) **BROWSER BASED WEB SITE GENERATION TOOL AND RUN TIME ENGINE**

(76) Inventor: **Steven H. Rempell**, 38 Washington St., Novato, CA (US) 94947

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/454,061**

(22) Filed: **Dec. 2, 1999**

(51) **Int. Cl.**^{7} **G06F 17/00**; G06T 13/00

(52) **U.S. Cl.** **707/102**; 707/104.1; 707/501.1; 345/700; 345/473

(58) **Field of Search** 707/102, 103, 707/104, 501, 513, 517, 530, 104.1, 501.1; 345/333, 967, 326, 339, 348, 352, 700, 473

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,428,731	A	*	6/1995	Powers, III	707/501
5,842,020	A	*	11/1998	Faustini	395/701
5,870,767	A	*	2/1999	Kraft, IV	707/501
6,035,119	A	*	3/2000	Massena et al.	717/100
6,081,263	A	*	6/2000	LeGall et al.	345/327
6,083,276	A	*	7/2000	Davidson et al.	717/1
6,148,311	A	*	11/2000	Wishnie et al.	707/513
6,191,786	B1	*	2/2001	Eyzaguirre et al.	345/853

OTHER PUBLICATIONS

Piero Fraternali, "Tools and Approaches for Developing Data–Intensive Web Applications: A Survey", ACM Sep. 1999, pp. 226–263.*

Balusubramanian et al A Large–Scale Hypermedia Application using Document Management and Web Technologies, ACM 1997, pp. 134–145.*

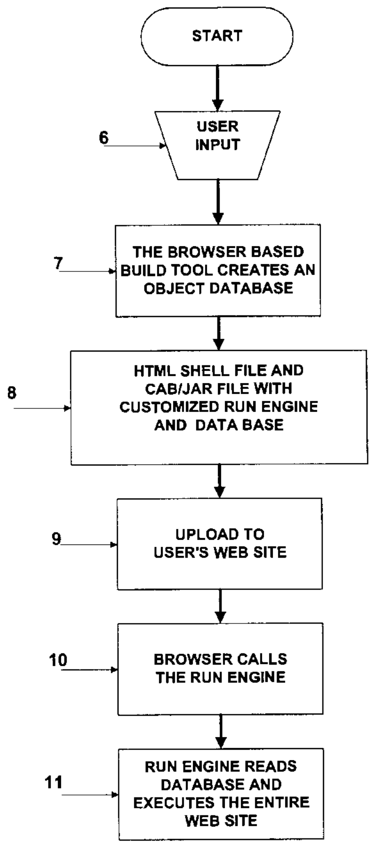
* cited by examiner

Primary Examiner—Safet Metjahic
Assistant Examiner—Uyen Le
(74) *Attorney, Agent, or Firm*—Coudert Brothers LLP

(57) **ABSTRACT**

A method and apparatus for designing and building a web page. The apparatus includes a browser based build engine including build tools and a user interface. The build tools are operable to construct a single run time file and an associated database that describe, and when executed, produce the web page. The user interface includes a build frame and a panel. The build frame is operable to receive user input and present a WYSIWIG representation of the web page. The panel includes one or more menus for controlling the form of content to be placed on the web page.

39 Claims, 68 Drawing Sheets


START
6
USER INPUT
7
THE BROWSER BASED BUILD TOOL CREATES AN OBJECT DATABASE
8
HTML SHELL FILE AND CAB/JAR FILE WITH CUSTOMIZED RUN ENGINE AND DATA BASE
9
UPLOAD TO USER'S WEB SITE
10
BROWSER CALLS THE RUN ENGINE
11
RUN ENGINE READS DATABASE AND EXECUTES THE ENTIRE WEB SITE

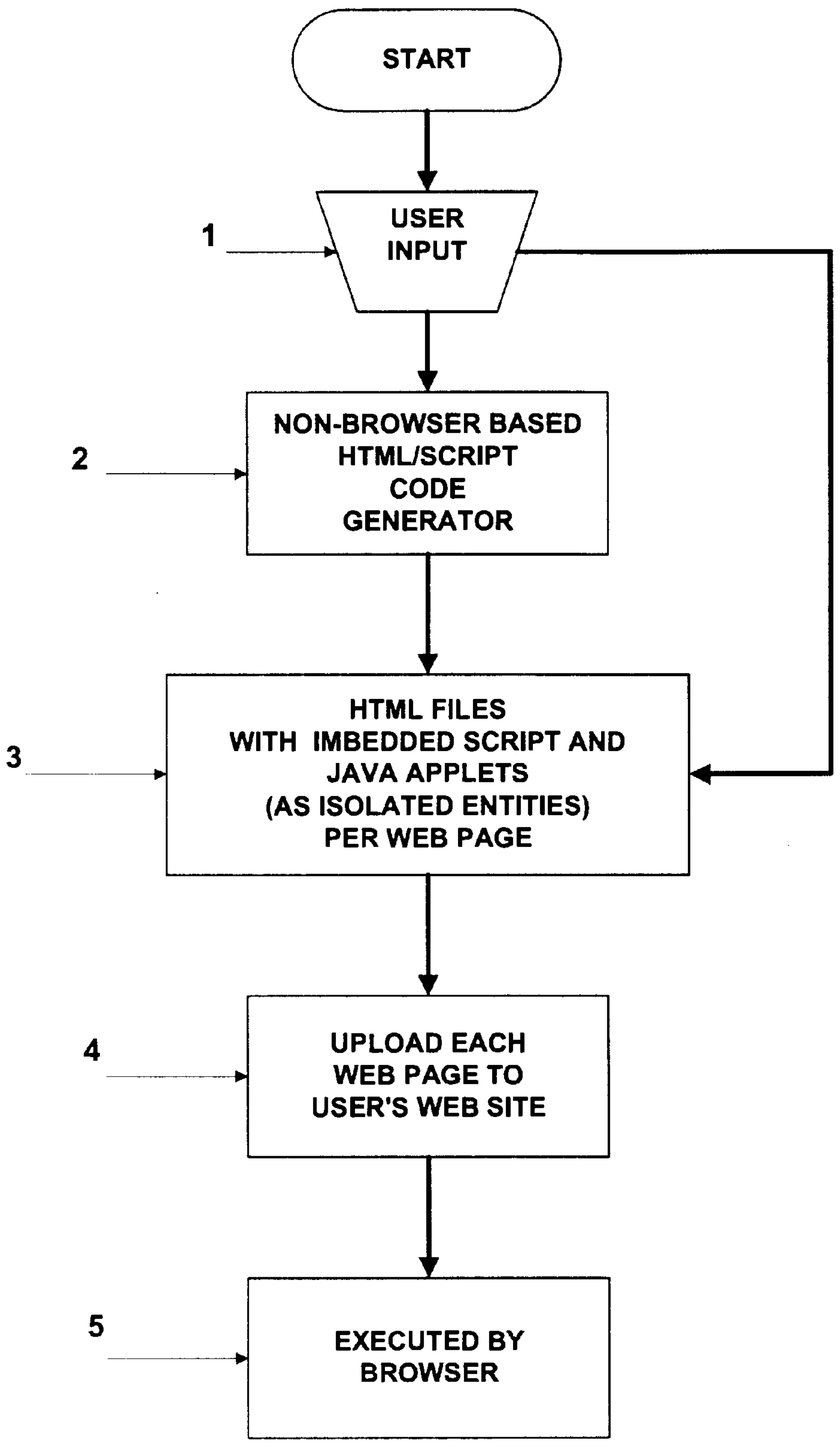

START
USER INPUT
1
NON-BROWSER BASED HTML/SCRIPT CODE GENERATOR
2
HTML FILES WITH IMBEDDED SCRIPT AND JAVA APPLETS (AS ISOLATED ENTITIES) PER WEB PAGE
3
UPLOAD EACH WEB PAGE TO USER'S WEB SITE
4
EXECUTED BY BROWSER
5

Fig. 1 PRIOR ART

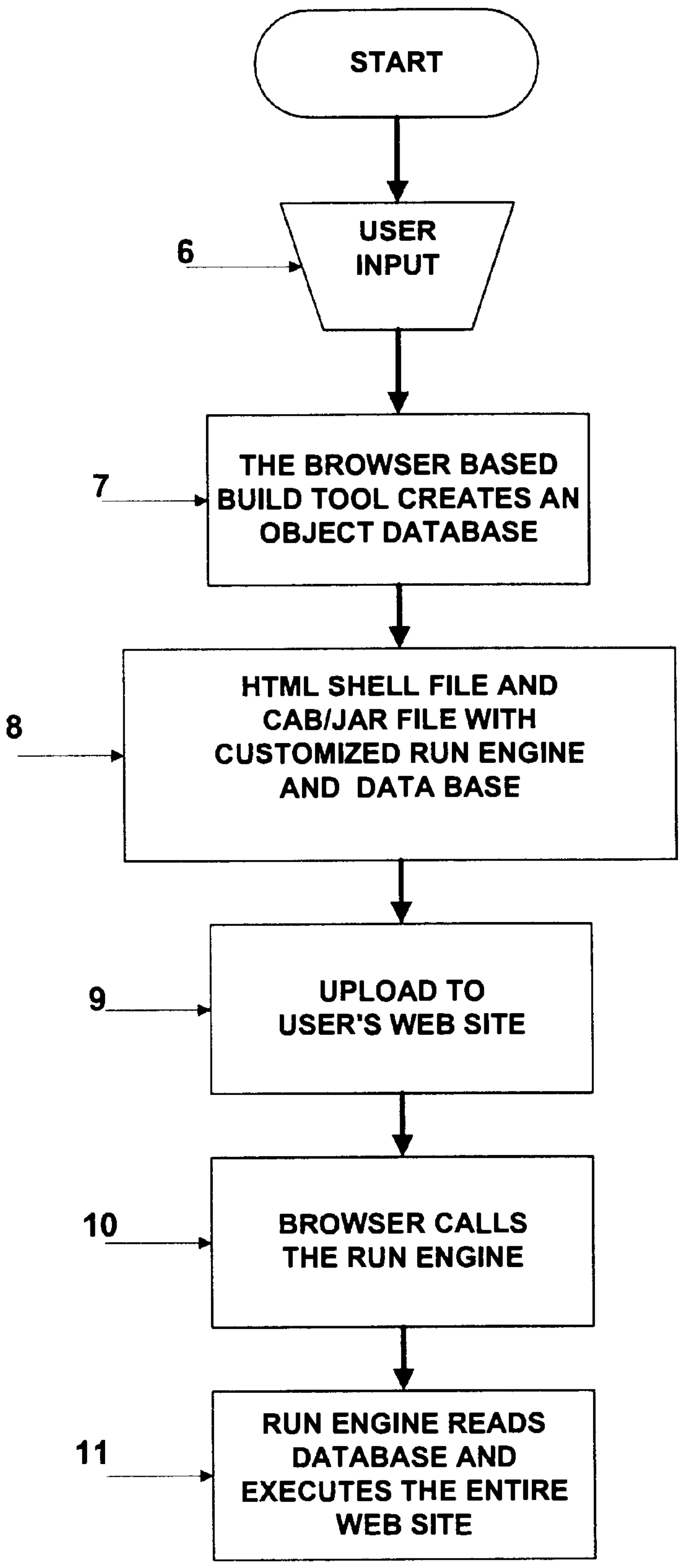

START
6
USER INPUT
7
THE BROWSER BASED BUILD TOOL CREATES AN OBJECT DATABASE
8
HTML SHELL FILE AND CAB/JAR FILE WITH CUSTOMIZED RUN ENGINE AND DATA BASE
9
UPLOAD TO USER'S WEB SITE
10
BROWSER CALLS THE RUN ENGINE
11
RUN ENGINE READS DATABASE AND EXECUTES THE ENTIRE WEB SITE

Fig. 2

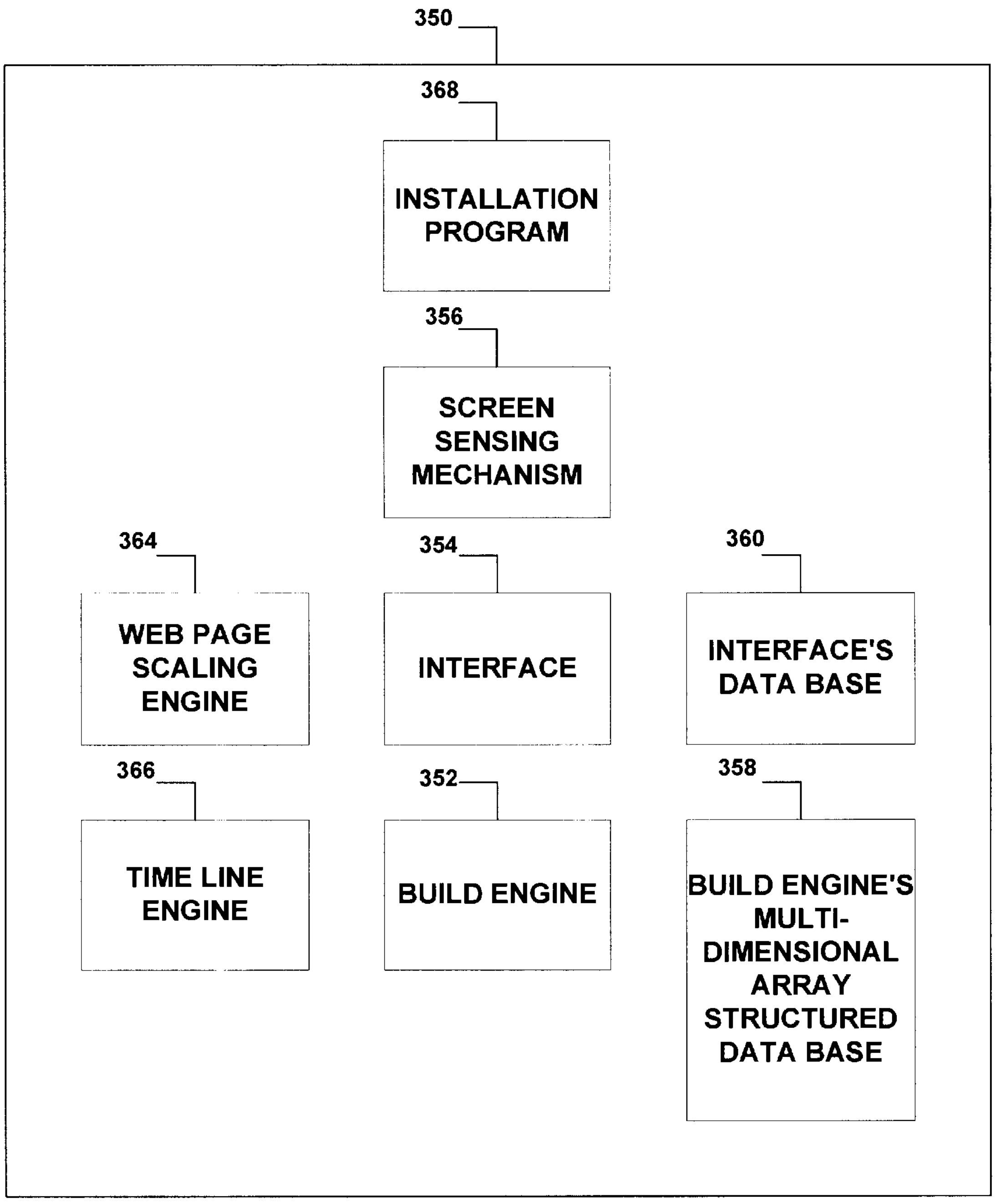


Fig. 3a

BUILD TOOL COMPONENTS

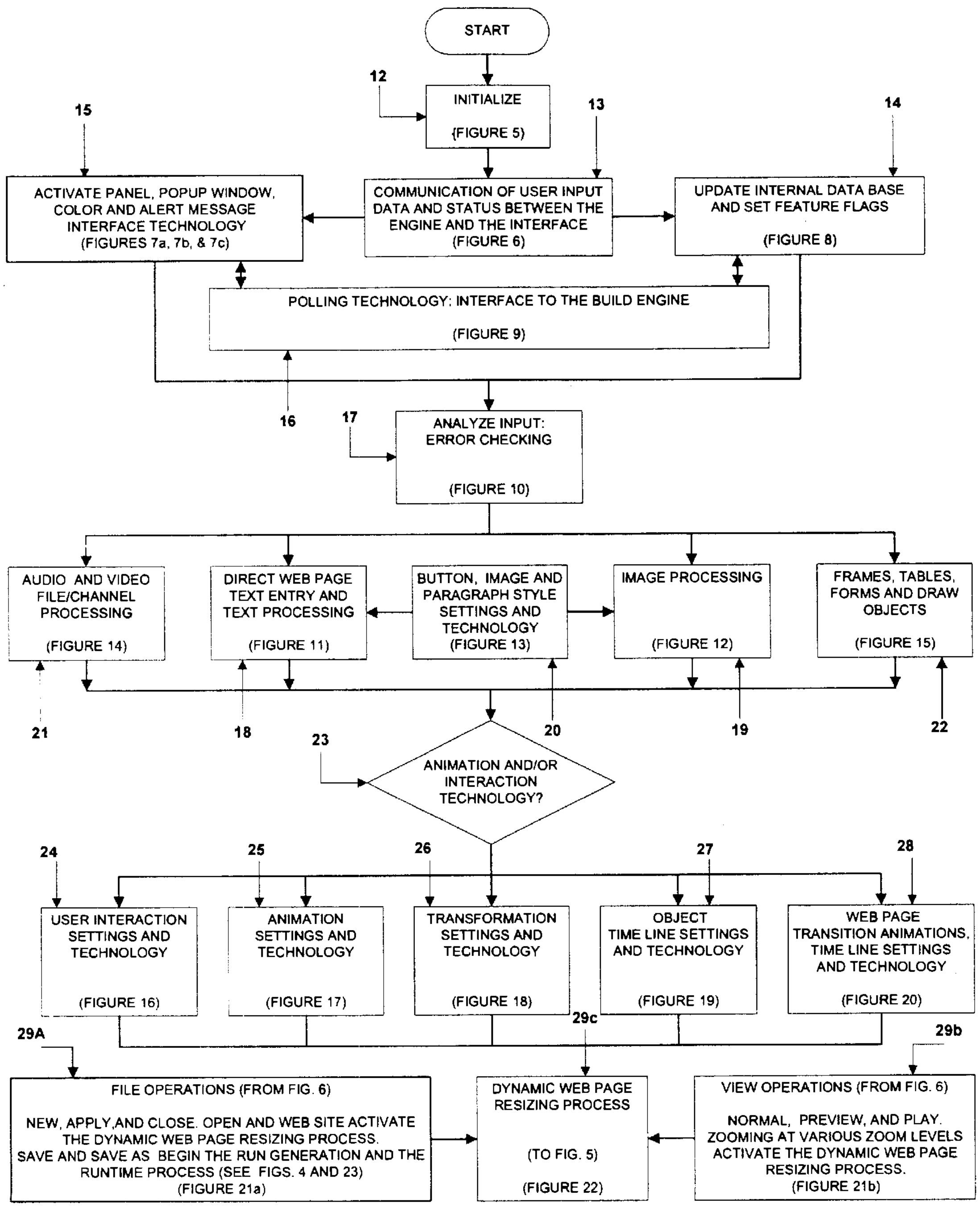

START
12
INITIALIZE (FIGURE 5)
13
14
15
ACTIVATE PANEL, POPUP WINDOW, COLOR AND ALERT MESSAGE INTERFACE TECHNOLOGY (FIGURES 7a, 7b, & 7c)
COMMUNICATION OF USER INPUT DATA AND STATUS BETWEEN THE ENGINE AND THE INTERFACE (FIGURE 6)
UPDATE INTERNAL DATA BASE AND SET FEATURE FLAGS (FIGURE 8)
POLLING TECHNOLOGY: INTERFACE TO THE BUILD ENGINE (FIGURE 9)
16
17
ANALYZE INPUT: ERROR CHECKING (FIGURE 10)
AUDIO AND VIDEO FILE/CHANNEL PROCESSING (FIGURE 14)
DIRECT WEB PAGE TEXT ENTRY AND TEXT PROCESSING (FIGURE 11)
BUTTON, IMAGE AND PARAGRAPH STYLE SETTINGS AND TECHNOLOGY (FIGURE 13)
IMAGE PROCESSING (FIGURE 12)
FRAMES, TABLES, FORMS AND DRAW OBJECTS (FIGURE 15)
21
18
23
20
19
22
ANIMATION AND/OR INTERACTION TECHNOLOGY?
24
25
26
27
28
USER INTERACTION SETTINGS AND TECHNOLOGY (FIGURE 16)
ANIMATION SETTINGS AND TECHNOLOGY (FIGURE 17)
TRANSFORMATION SETTINGS AND TECHNOLOGY (FIGURE 18)
OBJECT TIME LINE SETTINGS AND TECHNOLOGY (FIGURE 19)
WEB PAGE TRANSITION ANIMATIONS, TIME LINE SETTINGS AND TECHNOLOGY (FIGURE 20)
29A
29c
29b
FILE OPERATIONS (FROM FIG. 6) NEW, APPLY,AND CLOSE. OPEN AND WEB SITE ACTIVATE THE DYNAMIC WEB PAGE RESIZING PROCESS. SAVE AND SAVE AS BEGIN THE RUN GENERATION AND THE RUNTIME PROCESS (SEE FIGS. 4 AND 23) (FIGURE 21a)
DYNAMIC WEB PAGE RESIZING PROCESS (TO FIG. 5) (FIGURE 22)
VIEW OPERATIONS (FROM FIG. 6) NORMAL, PREVIEW, AND PLAY. ZOOMING AT VARIOUS ZOOM LEVELS ACTIVATE THE DYNAMIC WEB PAGE RESIZING PROCESS. (FIGURE 21b)

Fig. 3b

THE BUILD TOOL & BUILD PROCESS

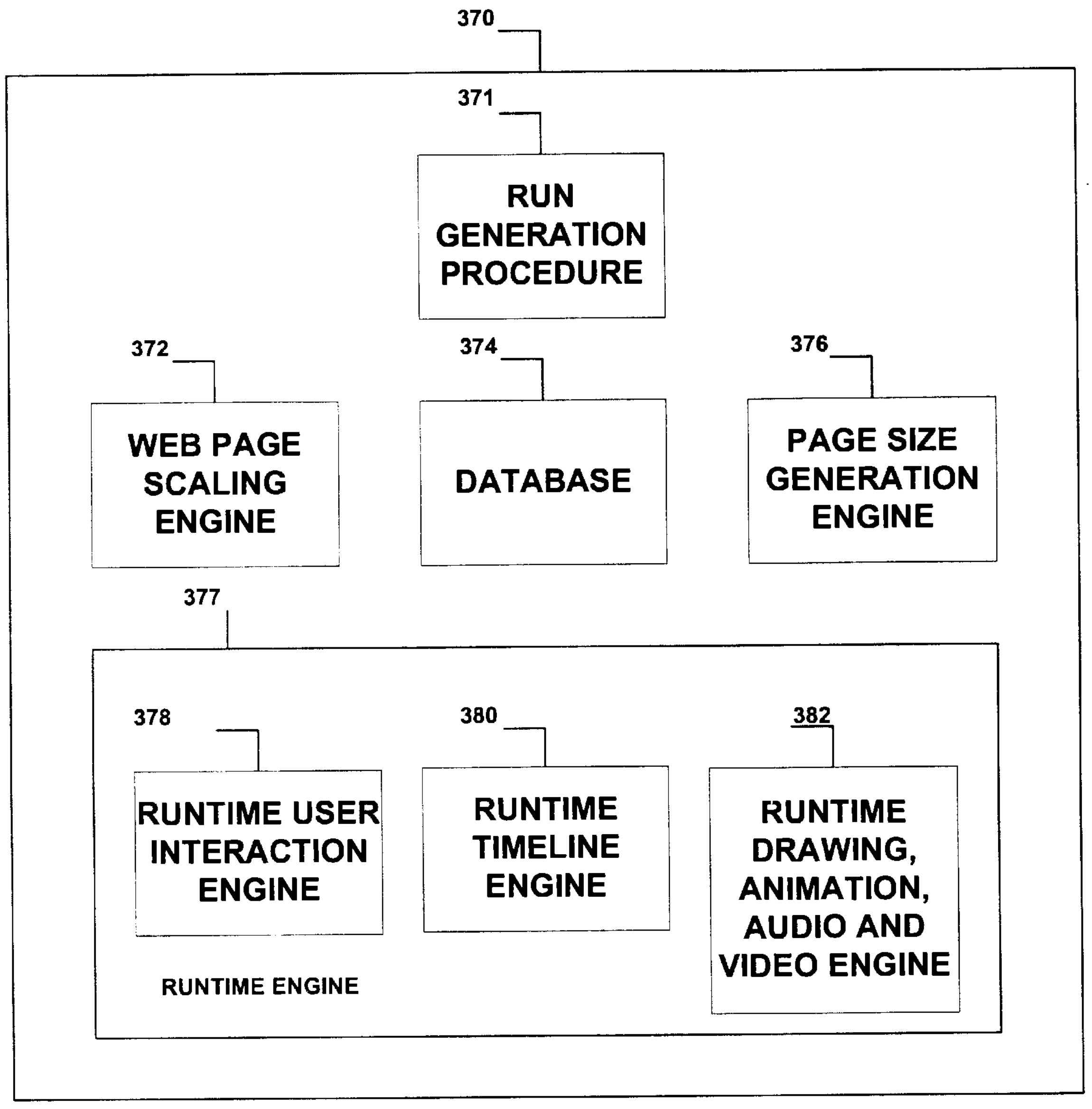


Fig. 4a

RUN GENERATION AND RUNTIME COMPONENTS

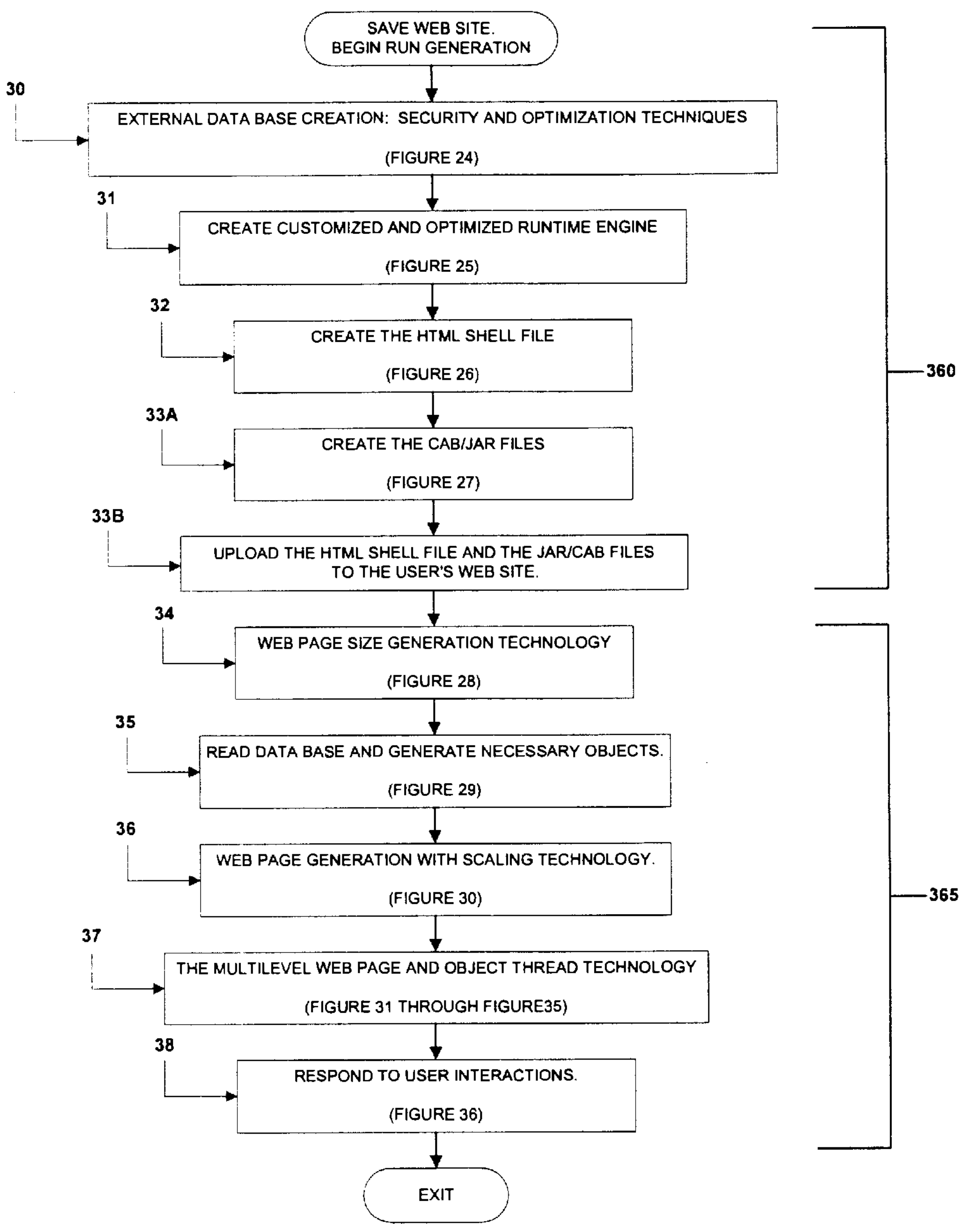

SAVE WEB SITE. BEGIN RUN GENERATION
30
EXTERNAL DATA BASE CREATION: SECURITY AND OPTIMIZATION TECHNIQUES (FIGURE 24)
31
CREATE CUSTOMIZED AND OPTIMIZED RUNTIME ENGINE (FIGURE 25)
32
CREATE THE HTML SHELL FILE (FIGURE 26)
33A
CREATE THE CAB/JAR FILES (FIGURE 27)
33B
UPLOAD THE HTML SHELL FILE AND THE JAR/CAB FILES TO THE USER'S WEB SITE.
360
34
WEB PAGE SIZE GENERATION TECHNOLOGY (FIGURE 28)
35
READ DATA BASE AND GENERATE NECESSARY OBJECTS. (FIGURE 29)
36
WEB PAGE GENERATION WITH SCALING TECHNOLOGY. (FIGURE 30)
365
37
THE MULTILEVEL WEB PAGE AND OBJECT THREAD TECHNOLOGY (FIGURE 31 THROUGH FIGURE35)
38
RESPOND TO USER INTERACTIONS. (FIGURE 36)
EXIT

Fig. 4b

RUN GENERATION & THE RUNTIME PROCESS

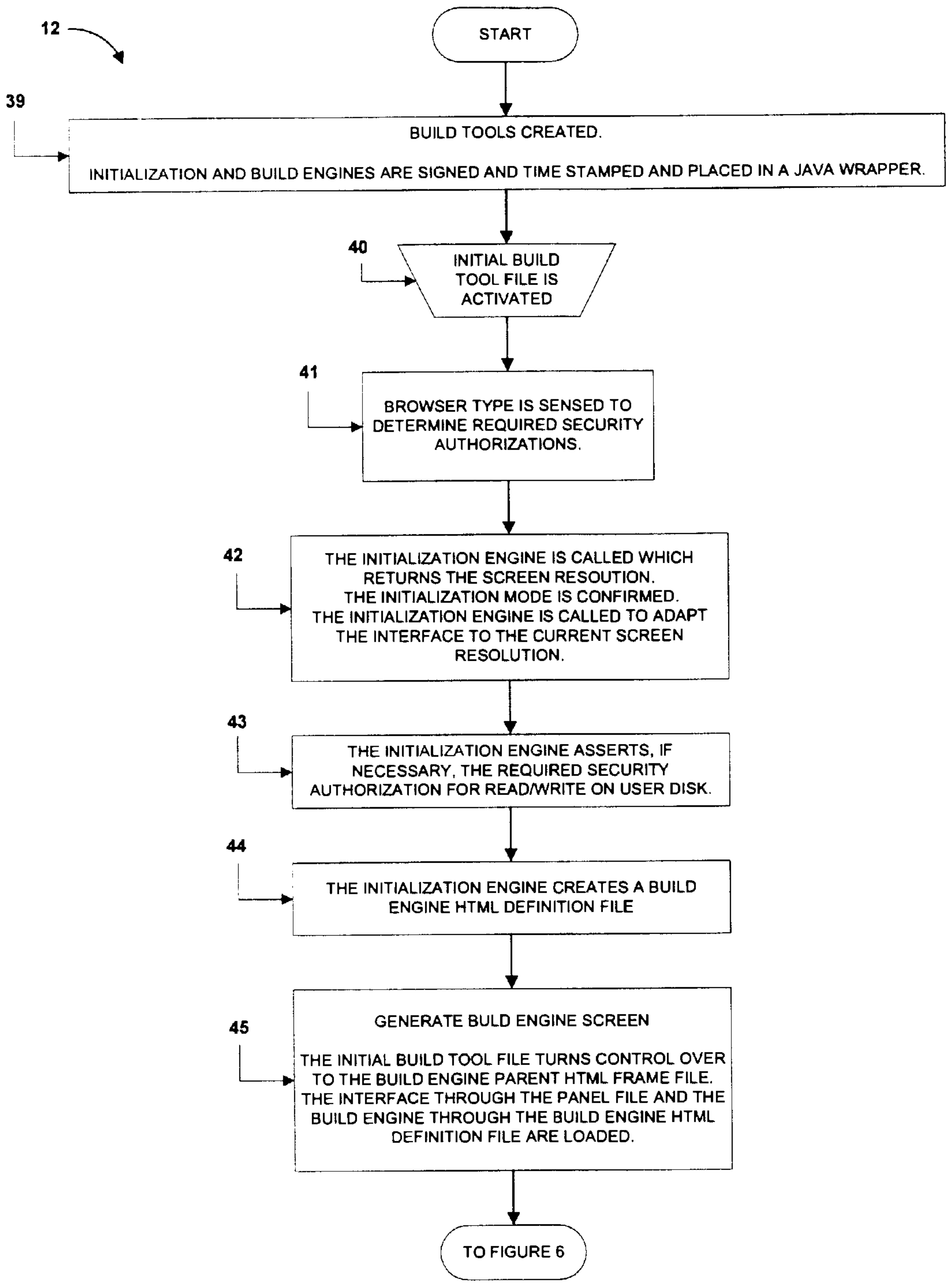

12
START
39
BUILD TOOLS CREATED.
INITIALIZATION AND BUILD ENGINES ARE SIGNED AND TIME STAMPED AND PLACED IN A JAVA WRAPPER.
40
INITIAL BUILD TOOL FILE IS ACTIVATED
41
BROWSER TYPE IS SENSED TO DETERMINE REQUIRED SECURITY AUTHORIZATIONS.
42
THE INITIALIZATION ENGINE IS CALLED WHICH RETURNS THE SCREEN RESOUTION. THE INITIALIZATION MODE IS CONFIRMED. THE INITIALIZATION ENGINE IS CALLED TO ADAPT THE INTERFACE TO THE CURRENT SCREEN RESOLUTION.
43
THE INITIALIZATION ENGINE ASSERTS, IF NECESSARY, THE REQUIRED SECURITY AUTHORIZATION FOR READ/WRITE ON USER DISK.
44
THE INITIALIZATION ENGINE CREATES A BUILD ENGINE HTML DEFINITION FILE
45
GENERATE BULD ENGINE SCREEN
THE INITIAL BUILD TOOL FILE TURNS CONTROL OVER TO THE BUILD ENGINE PARENT HTML FRAME FILE. THE INTERFACE THROUGH THE PANEL FILE AND THE BUILD ENGINE THROUGH THE BUILD ENGINE HTML DEFINITION FILE ARE LOADED.
TO FIGURE 6

Fig . 5 **INITIALIZATION**

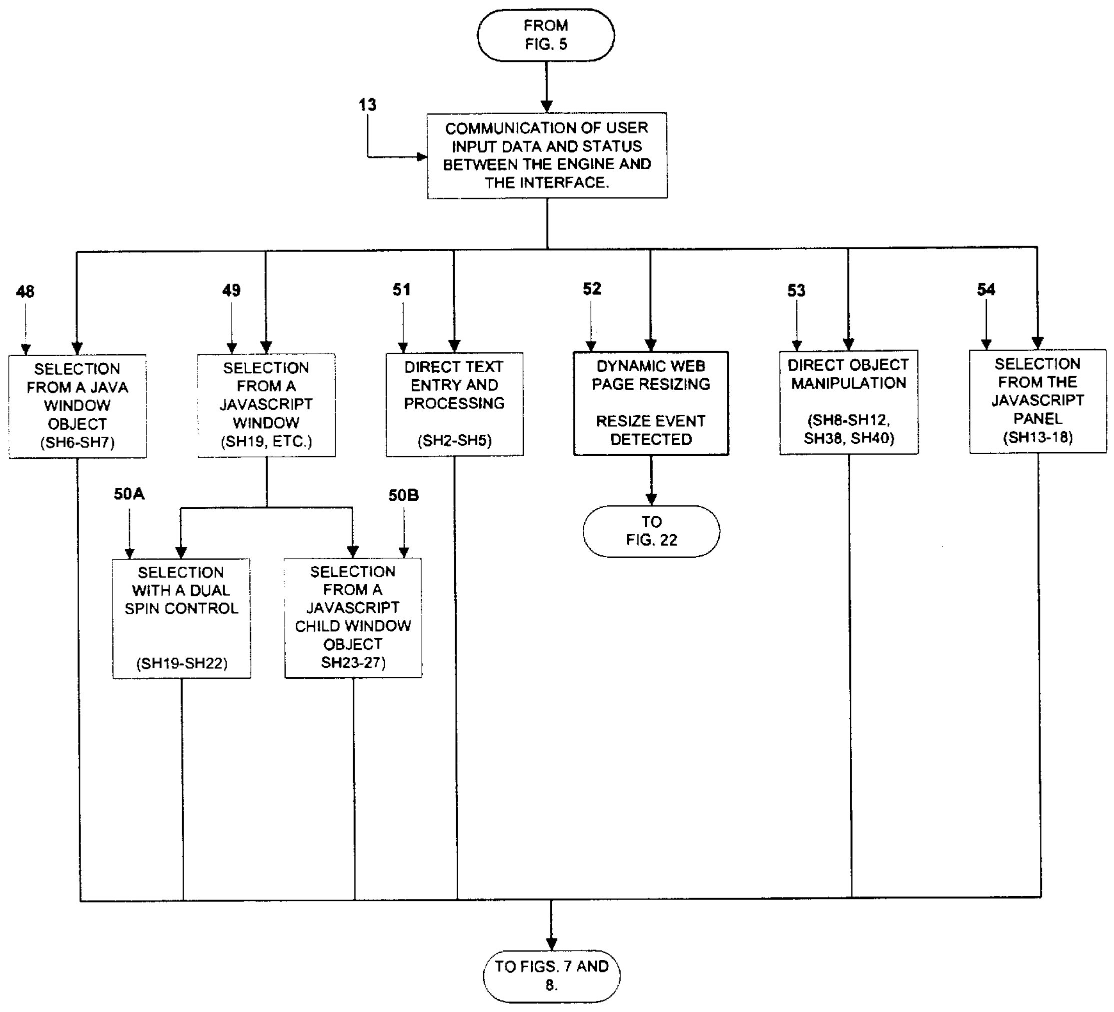

FROM FIG. 5
13
COMMUNICATION OF USER INPUT DATA AND STATUS BETWEEN THE ENGINE AND THE INTERFACE.
48
SELECTION FROM A JAVA WINDOW OBJECT (SH6-SH7)
49
SELECTION FROM A JAVASCRIPT WINDOW (SH19, ETC.)
51
DIRECT TEXT ENTRY AND PROCESSING (SH2-SH5)
52
DYNAMIC WEB PAGE RESIZING RESIZE EVENT DETECTED
53
DIRECT OBJECT MANIPULATION (SH8-SH12, SH38, SH40)
54
SELECTION FROM THE JAVASCRIPT PANEL (SH13-18)
50A
SELECTION WITH A DUAL SPIN CONTROL (SH19-SH22)
50B
SELECTION FROM A JAVASCRIPT CHILD WINDOW OBJECT SH23-27)
TO FIG. 22
TO FIGS. 7 AND 8.

Fig. 6

COMMUNICATION OF USER INPUT DATA AND STATUS BETWEEN THE ENGINE AND THE INTERFACE.

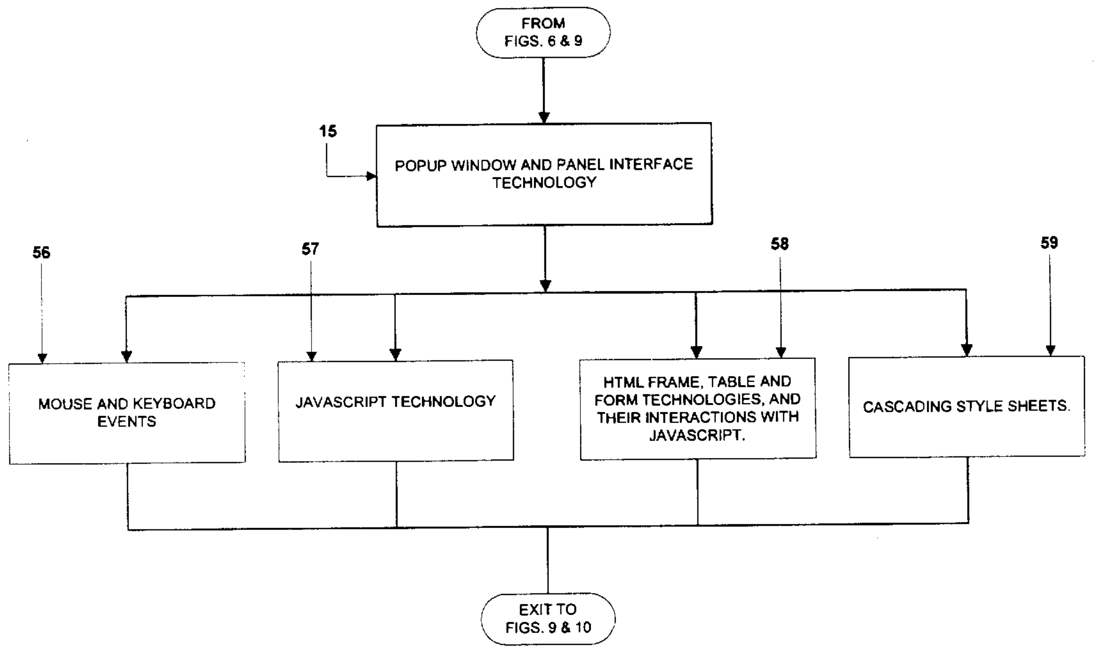

FROM FIGS. 6 & 9
15
POPUP WINDOW AND PANEL INTERFACE TECHNOLOGY
56
57
58
59
MOUSE AND KEYBOARD EVENTS
JAVASCRIPT TECHNOLOGY
HTML FRAME, TABLE AND FORM TECHNOLOGIES, AND THEIR INTERACTIONS WITH JAVASCRIPT.
CASCADING STYLE SHEETS.
EXIT TO FIGS. 9 & 10

Fig. 7a

POPUP WINDOW AND PANEL INTERFACE AND COLOR TECHNOLOGY

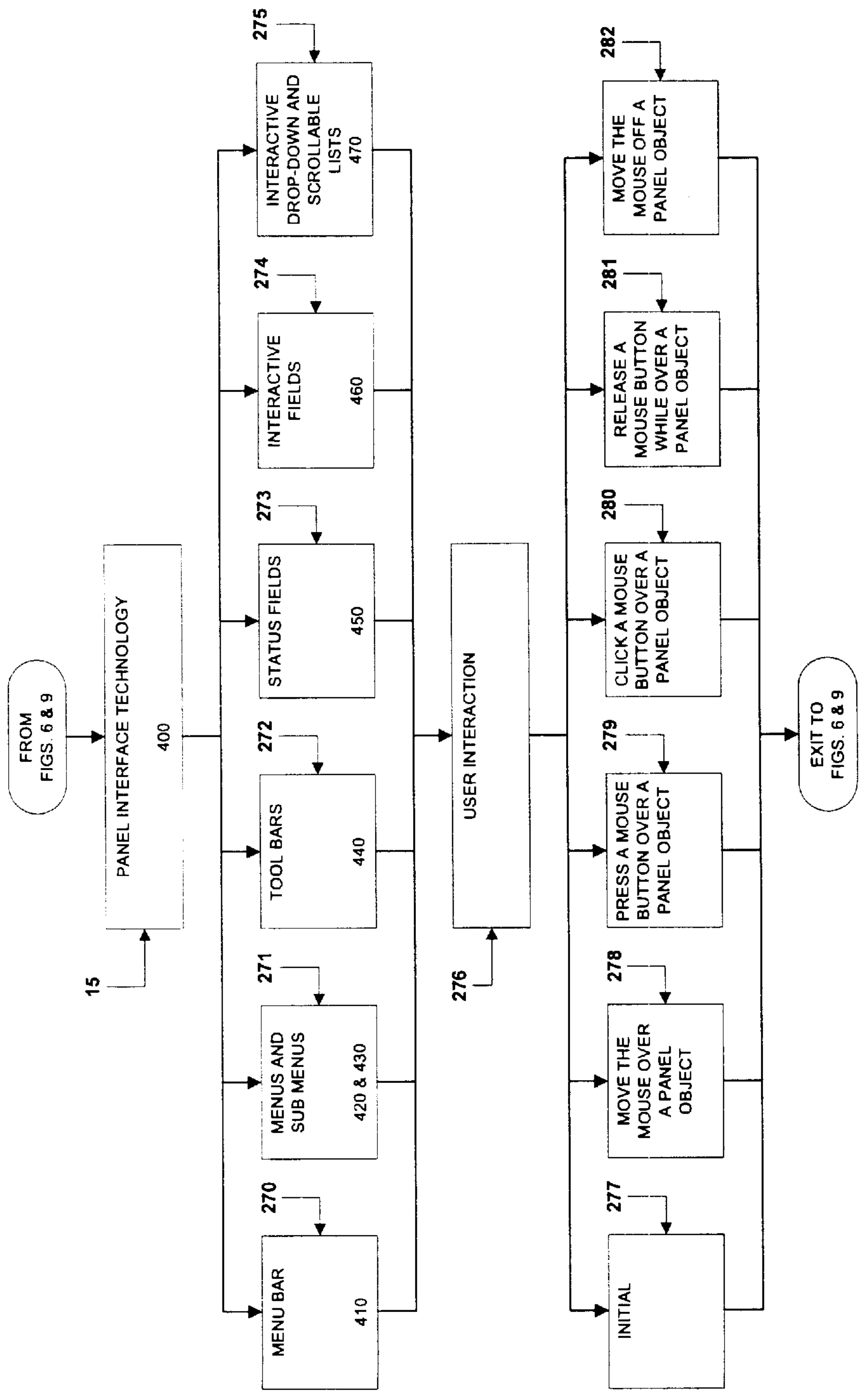

FROM FIGS. 6 & 9
15
PANEL INTERFACE TECHNOLOGY 400
270
MENU BAR 410
271
MENUS AND SUB MENUS 420 & 430
272
TOOL BARS 440
273
STATUS FIELDS 450
274
INTERACTIVE FIELDS 460
275
INTERACTIVE DROP-DOWN AND SCROLLABLE LISTS 470
276
USER INTERACTION
277
INITIAL
278
MOVE THE MOUSE OVER A PANEL OBJECT
279
PRESS A MOUSE BUTTON OVER A PANEL OBJECT
280
CLICK A MOUSE BUTTON OVER A PANEL OBJECT
281
RELEASE A MOUSE BUTTON WHILE OVER A PANEL OBJECT
282
MOVE THE MOUSE OFF A PANEL OBJECT
EXIT TO FIGS. 6 & 9

Fig. 7b

IMPLEMENTATION OF PANEL INTERFACE OBJECTS

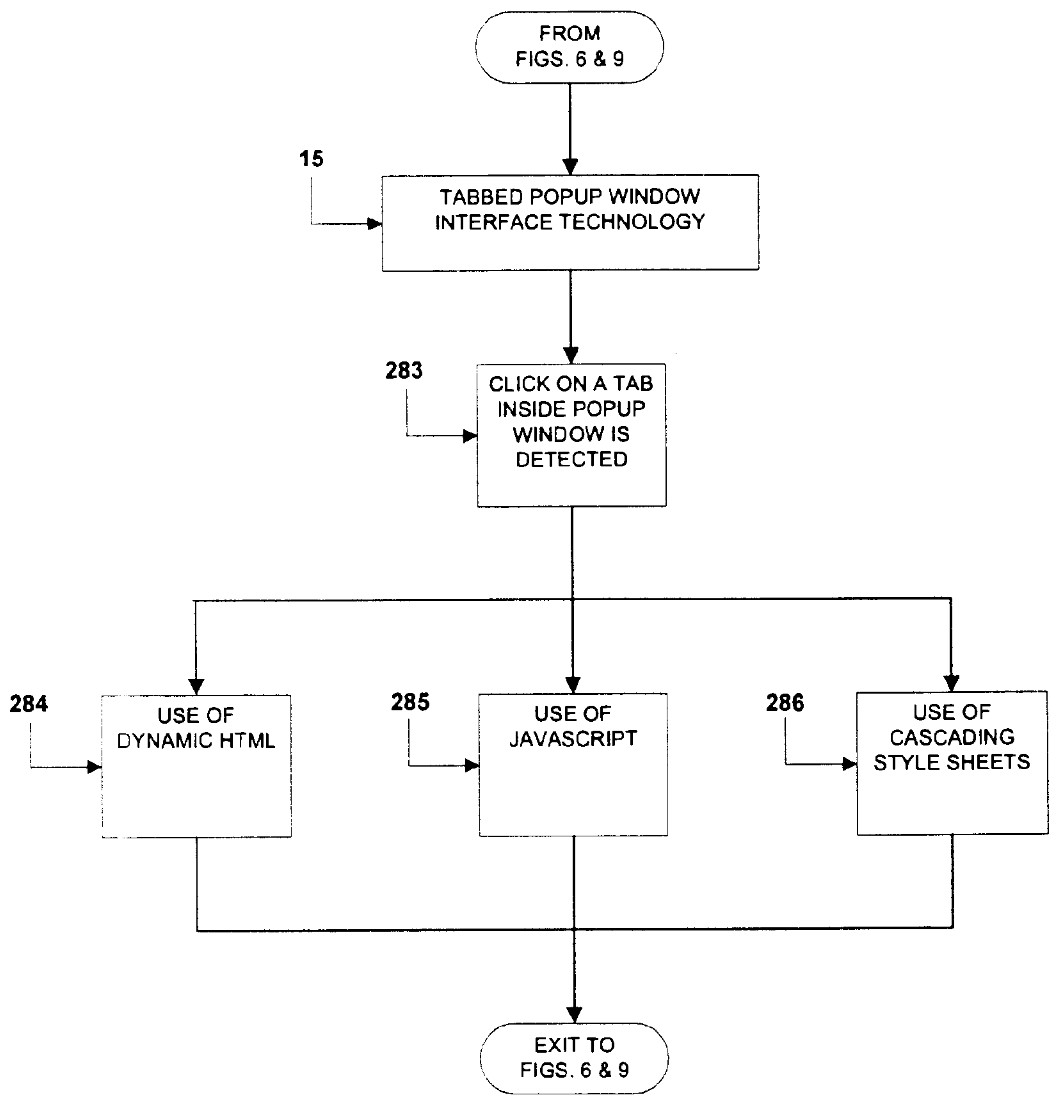

FROM FIGS. 6 & 9
15
TABBED POPUP WINDOW INTERFACE TECHNOLOGY
283
CLICK ON A TAB INSIDE POPUP WINDOW IS DETECTED
284
USE OF DYNAMIC HTML
285
USE OF JAVASCRIPT
286
USE OF CASCADING STYLE SHEETS
EXIT TO FIGS. 6 & 9

Fig. 7c

IMPLEMENTATION OF TABBED POPUP WINDOWS

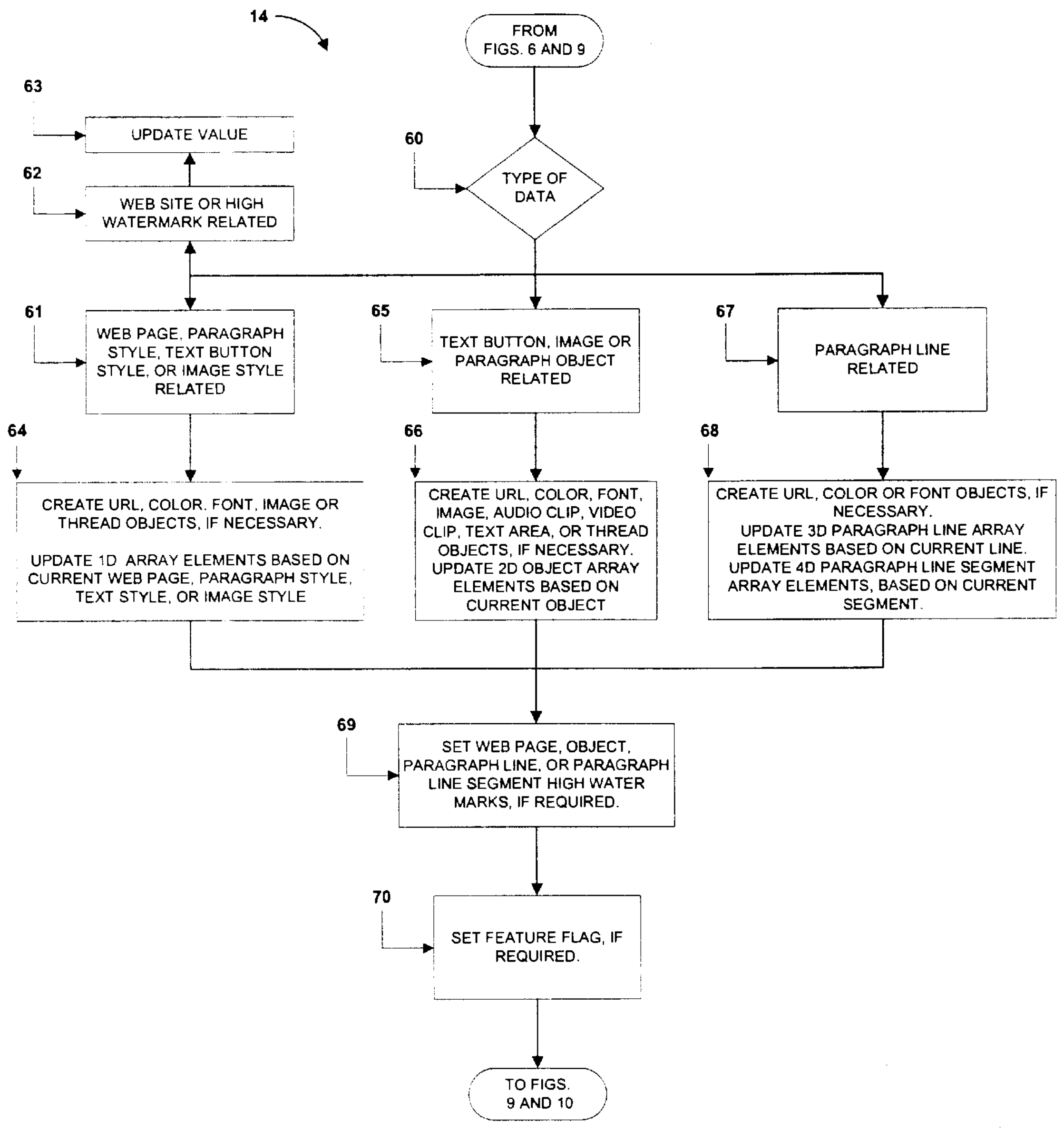

14
FROM FIGS. 6 AND 9
60
TYPE OF DATA
63
UPDATE VALUE
62
WEB SITE OR HIGH WATERMARK RELATED
61
WEB PAGE, PARAGRAPH STYLE, TEXT BUTTON STYLE, OR IMAGE STYLE RELATED
65
TEXT BUTTON, IMAGE OR PARAGRAPH OBJECT RELATED
67
PARAGRAPH LINE RELATED
64
CREATE URL, COLOR, FONT, IMAGE OR THREAD OBJECTS, IF NECESSARY.
UPDATE 1D ARRAY ELEMENTS BASED ON CURRENT WEB PAGE, PARAGRAPH STYLE, TEXT STYLE, OR IMAGE STYLE
66
CREATE URL, COLOR, FONT, IMAGE, AUDIO CLIP, VIDEO CLIP, TEXT AREA, OR THREAD OBJECTS, IF NECESSARY. UPDATE 2D OBJECT ARRAY ELEMENTS BASED ON CURRENT OBJECT
68
CREATE URL, COLOR OR FONT OBJECTS, IF NECESSARY. UPDATE 3D PARAGRAPH LINE ARRAY ELEMENTS BASED ON CURRENT LINE. UPDATE 4D PARAGRAPH LINE SEGMENT ARRAY ELEMENTS, BASED ON CURRENT SEGMENT.
69
SET WEB PAGE, OBJECT, PARAGRAPH LINE, OR PARAGRAPH LINE SEGMENT HIGH WATER MARKS, IF REQUIRED.
70
SET FEATURE FLAG, IF REQUIRED.
TO FIGS. 9 AND 10

Fig. 8

UPDATE INTERNAL DATA BASE AND SET FEATURE FLAGS

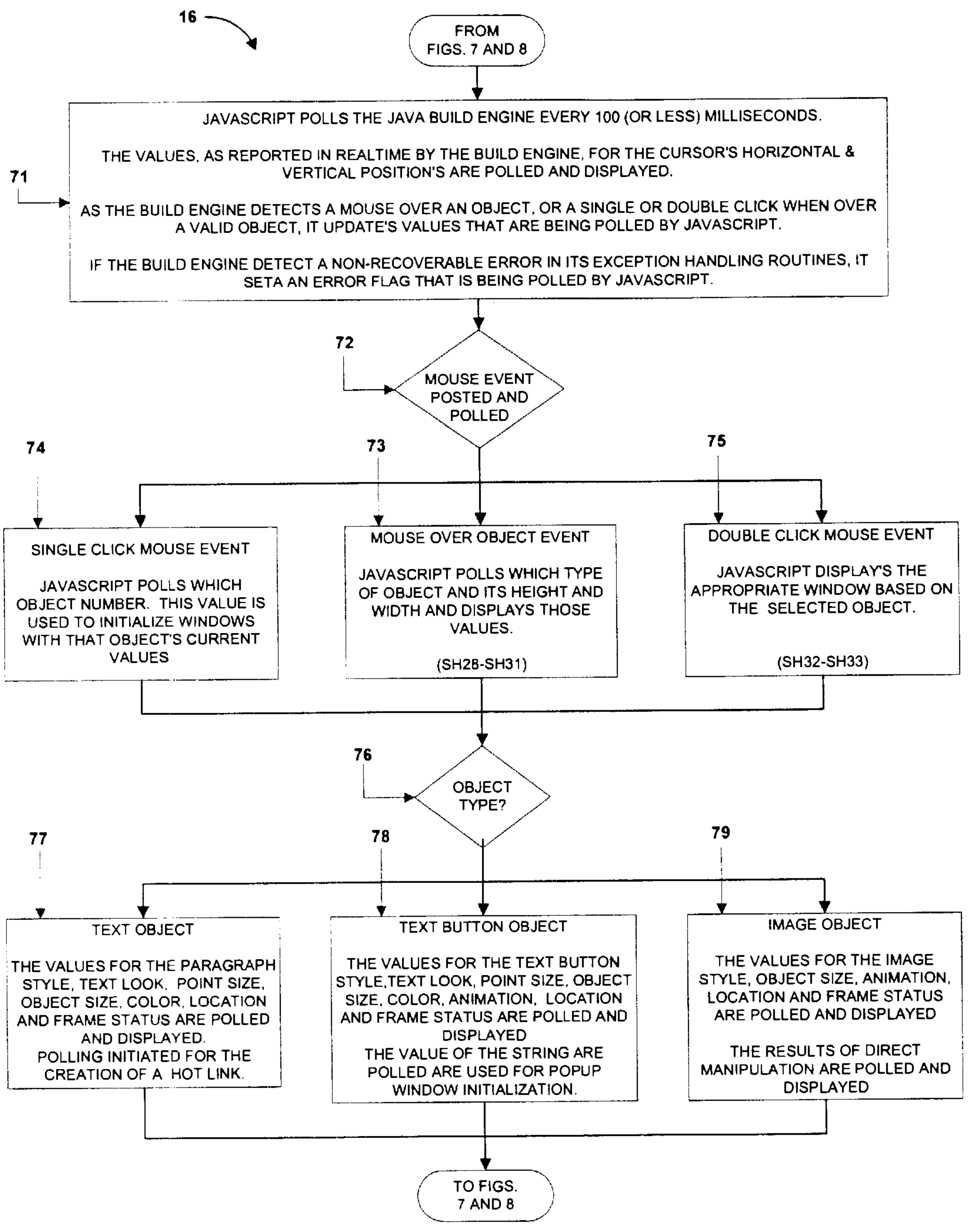

16
FROM FIGS. 7 AND 8
71
JAVASCRIPT POLLS THE JAVA BUILD ENGINE EVERY 100 (OR LESS) MILLISECONDS.
THE VALUES, AS REPORTED IN REALTIME BY THE BUILD ENGINE, FOR THE CURSOR'S HORIZONTAL & VERTICAL POSITION'S ARE POLLED AND DISPLAYED.
AS THE BUILD ENGINE DETECTS A MOUSE OVER AN OBJECT, OR A SINGLE OR DOUBLE CLICK WHEN OVER A VALID OBJECT, IT UPDATE'S VALUES THAT ARE BEING POLLED BY JAVASCRIPT.
IF THE BUILD ENGINE DETECT A NON-RECOVERABLE ERROR IN ITS EXCEPTION HANDLING ROUTINES, IT SETA AN ERROR FLAG THAT IS BEING POLLED BY JAVASCRIPT.
72
MOUSE EVENT POSTED AND POLLED
74
73
75
SINGLE CLICK MOUSE EVENT
JAVASCRIPT POLLS WHICH OBJECT NUMBER. THIS VALUE IS USED TO INITIALIZE WINDOWS WITH THAT OBJECT'S CURRENT VALUES
MOUSE OVER OBJECT EVENT
JAVASCRIPT POLLS WHICH TYPE OF OBJECT AND ITS HEIGHT AND WIDTH AND DISPLAYS THOSE VALUES.
(SH28-SH31)
DOUBLE CLICK MOUSE EVENT
JAVASCRIPT DISPLAY'S THE APPROPRIATE WINDOW BASED ON THE SELECTED OBJECT.
(SH32-SH33)
76
OBJECT TYPE?
77
78
79
TEXT OBJECT
THE VALUES FOR THE PARAGRAPH STYLE, TEXT LOOK, POINT SIZE, OBJECT SIZE, COLOR, LOCATION AND FRAME STATUS ARE POLLED AND DISPLAYED.
POLLING INITIATED FOR THE CREATION OF A HOT LINK.
TEXT BUTTON OBJECT
THE VALUES FOR THE TEXT BUTTON STYLE,TEXT LOOK, POINT SIZE, OBJECT SIZE, COLOR, ANIMATION, LOCATION AND FRAME STATUS ARE POLLED AND DISPLAYED
THE VALUE OF THE STRING ARE POLLED ARE USED FOR POPUP WINDOW INITIALIZATION.
IMAGE OBJECT
THE VALUES FOR THE IMAGE STYLE, OBJECT SIZE, ANIMATION, LOCATION AND FRAME STATUS ARE POLLED AND DISPLAYED
THE RESULTS OF DIRECT MANIPULATION ARE POLLED AND DISPLAYED
TO FIGS. 7 AND 8

Fig. 9 **POLLING METHODS**

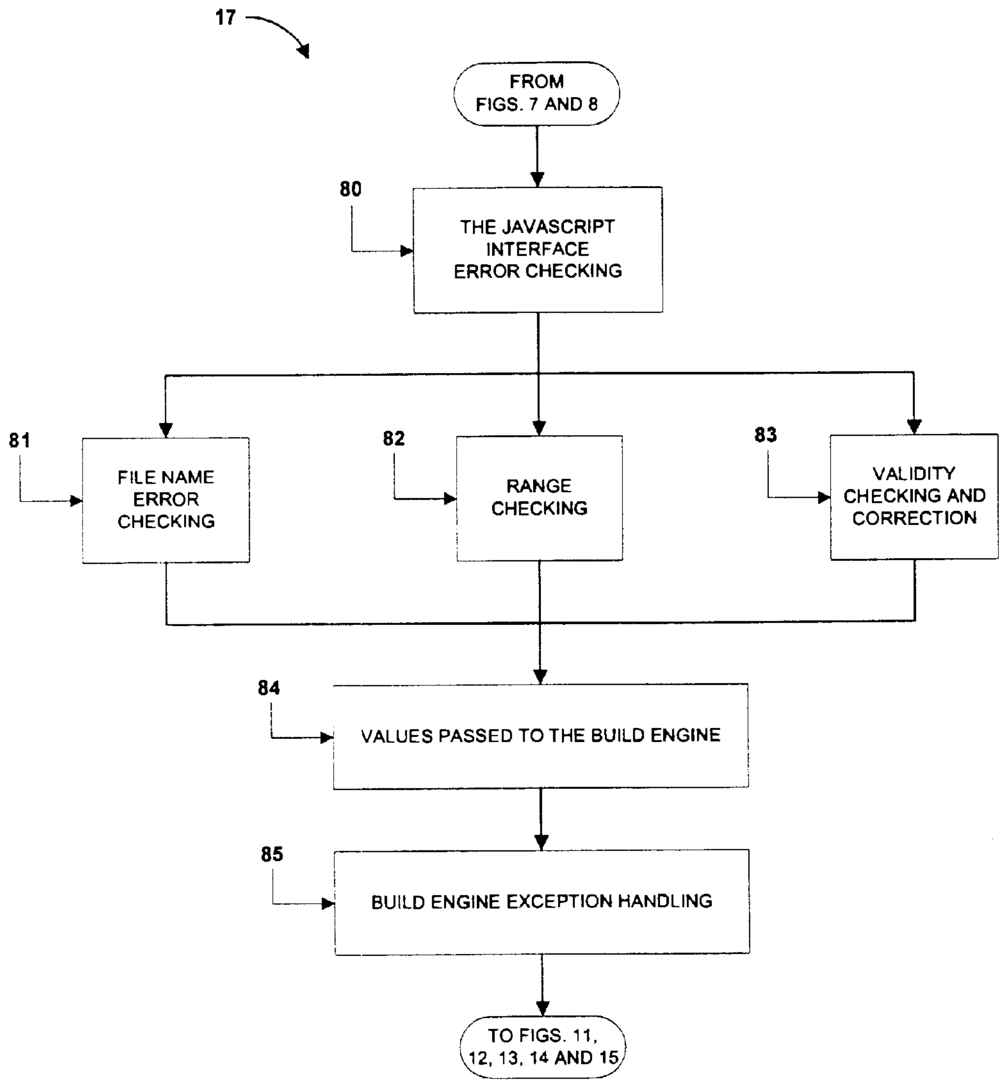

17
FROM FIGS. 7 AND 8
80
THE JAVASCRIPT INTERFACE ERROR CHECKING
81
FILE NAME ERROR CHECKING
82
RANGE CHECKING
83
VALIDITY CHECKING AND CORRECTION
84
VALUES PASSED TO THE BUILD ENGINE
85
BUILD ENGINE EXCEPTION HANDLING
TO FIGS. 11, 12, 13, 14 AND 15

Fig. 10

ANALYZE INPUT: ERROR CHECKING

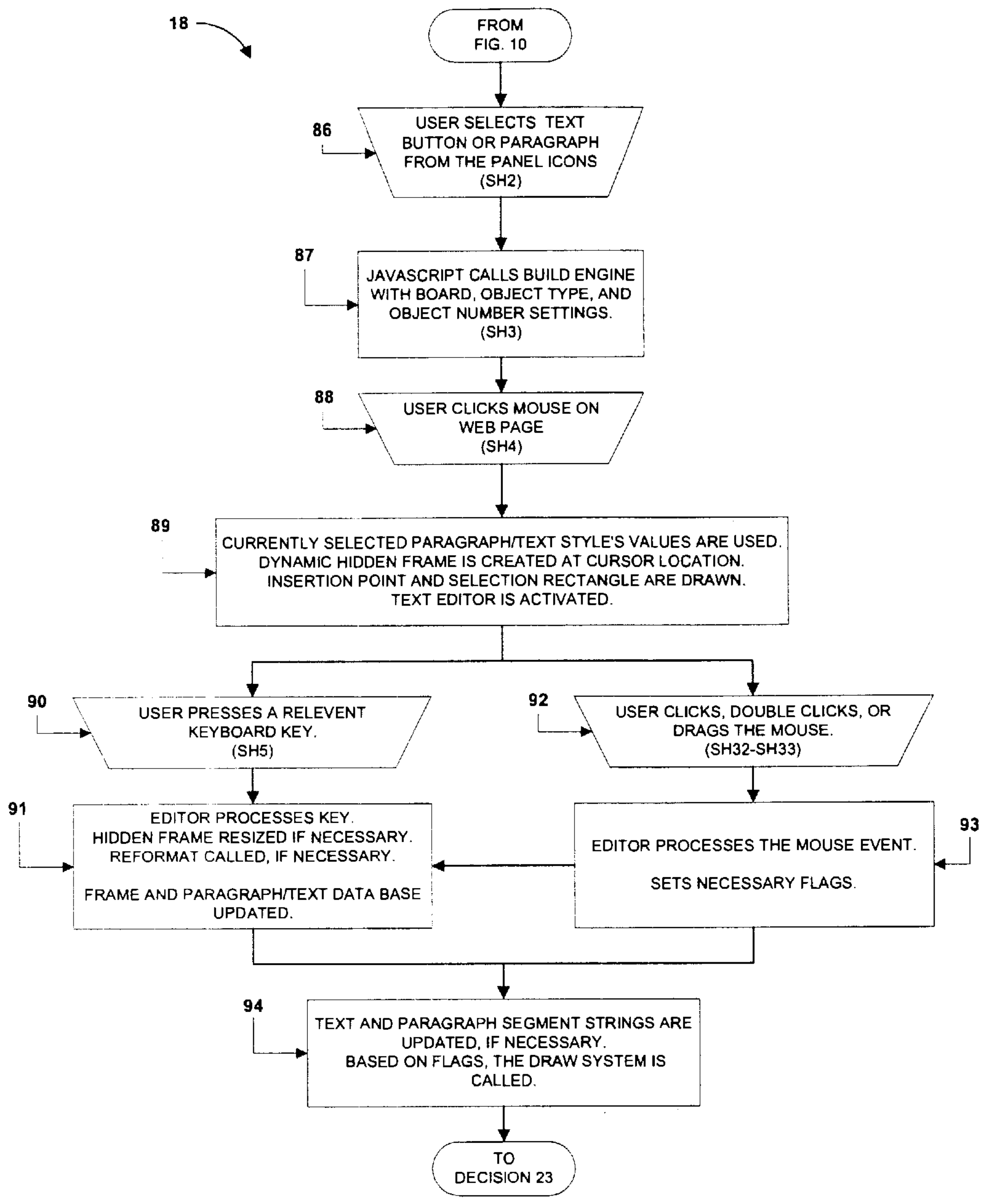

18
FROM FIG. 10
86
USER SELECTS TEXT BUTTON OR PARAGRAPH FROM THE PANEL ICONS (SH2)
87
JAVASCRIPT CALLS BUILD ENGINE WITH BOARD, OBJECT TYPE, AND OBJECT NUMBER SETTINGS. (SH3)
88
USER CLICKS MOUSE ON WEB PAGE (SH4)
89
CURRENTLY SELECTED PARAGRAPH/TEXT STYLE'S VALUES ARE USED. DYNAMIC HIDDEN FRAME IS CREATED AT CURSOR LOCATION. INSERTION POINT AND SELECTION RECTANGLE ARE DRAWN. TEXT EDITOR IS ACTIVATED.
90
USER PRESSES A RELEVENT KEYBOARD KEY. (SH5)
92
USER CLICKS, DOUBLE CLICKS, OR DRAGS THE MOUSE. (SH32-SH33)
91
EDITOR PROCESSES KEY. HIDDEN FRAME RESIZED IF NECESSARY. REFORMAT CALLED, IF NECESSARY. FRAME AND PARAGRAPH/TEXT DATA BASE UPDATED.
93
EDITOR PROCESSES THE MOUSE EVENT. SETS NECESSARY FLAGS.
94
TEXT AND PARAGRAPH SEGMENT STRINGS ARE UPDATED, IF NECESSARY. BASED ON FLAGS, THE DRAW SYSTEM IS CALLED.
TO DECISION 23

Fig. 11

DIRECT WEB PAGE DATA ENTRY AND TEXT PROCESSING

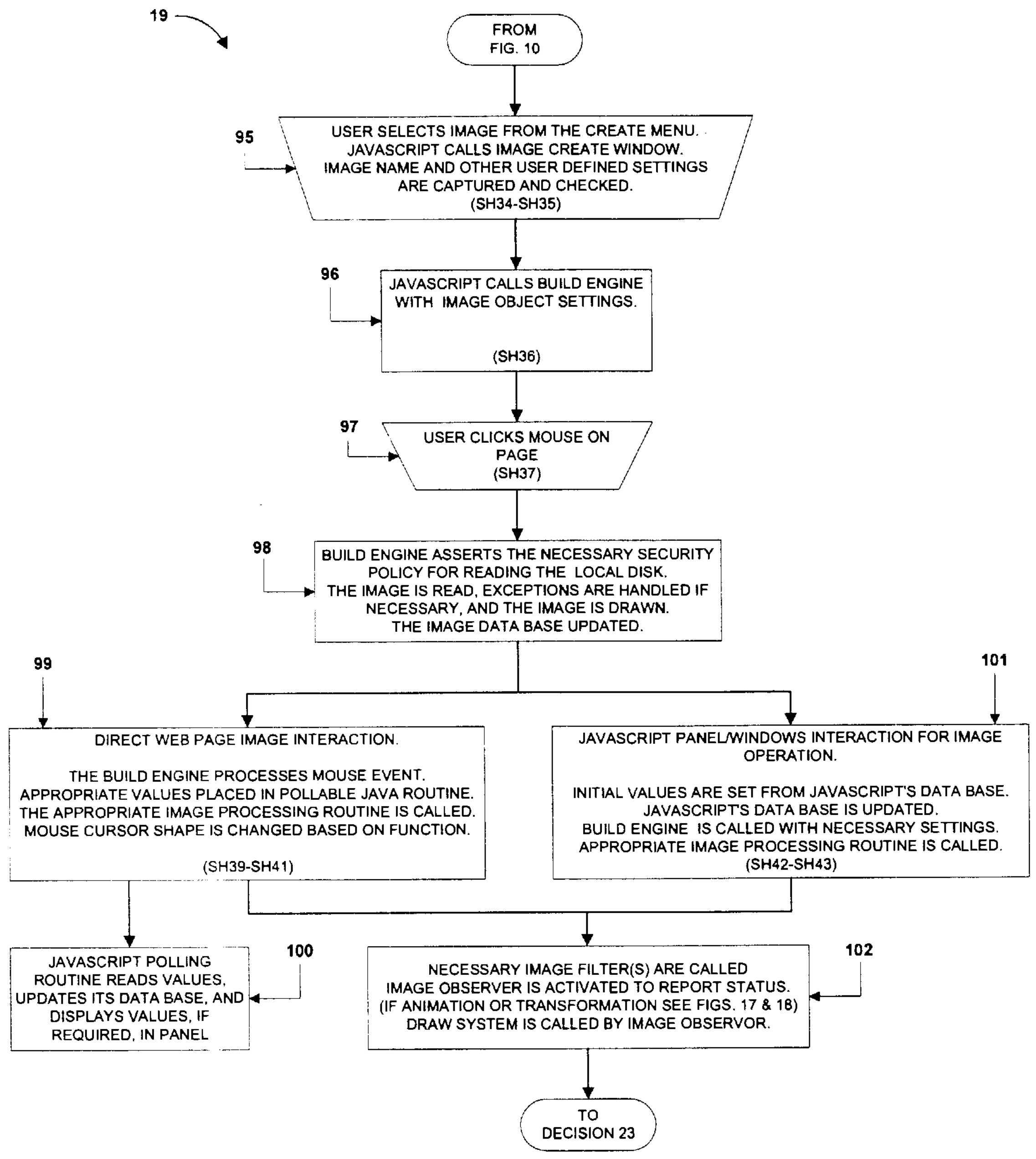

19
FROM FIG. 10
95
USER SELECTS IMAGE FROM THE CREATE MENU. JAVASCRIPT CALLS IMAGE CREATE WINDOW. IMAGE NAME AND OTHER USER DEFINED SETTINGS ARE CAPTURED AND CHECKED. (SH34-SH35)
96
JAVASCRIPT CALLS BUILD ENGINE WITH IMAGE OBJECT SETTINGS. (SH36)
97
USER CLICKS MOUSE ON PAGE (SH37)
98
BUILD ENGINE ASSERTS THE NECESSARY SECURITY POLICY FOR READING THE LOCAL DISK. THE IMAGE IS READ, EXCEPTIONS ARE HANDLED IF NECESSARY, AND THE IMAGE IS DRAWN. THE IMAGE DATA BASE UPDATED.
99
DIRECT WEB PAGE IMAGE INTERACTION.
THE BUILD ENGINE PROCESSES MOUSE EVENT. APPROPRIATE VALUES PLACED IN POLLABLE JAVA ROUTINE. THE APPROPRIATE IMAGE PROCESSING ROUTINE IS CALLED. MOUSE CURSOR SHAPE IS CHANGED BASED ON FUNCTION. (SH39-SH41)
101
JAVASCRIPT PANEL/WINDOWS INTERACTION FOR IMAGE OPERATION.
INITIAL VALUES ARE SET FROM JAVASCRIPT'S DATA BASE. JAVASCRIPT'S DATA BASE IS UPDATED. BUILD ENGINE IS CALLED WITH NECESSARY SETTINGS. APPROPRIATE IMAGE PROCESSING ROUTINE IS CALLED. (SH42-SH43)
100
JAVASCRIPT POLLING ROUTINE READS VALUES, UPDATES ITS DATA BASE, AND DISPLAYS VALUES, IF REQUIRED, IN PANEL
102
NECESSARY IMAGE FILTER(S) ARE CALLED IMAGE OBSERVER IS ACTIVATED TO REPORT STATUS. (IF ANIMATION OR TRANSFORMATION SEE FIGS. 17 & 18) DRAW SYSTEM IS CALLED BY IMAGE OBSERVOR.
TO DECISION 23

Fig. 12 **IMAGE PROCESSING**

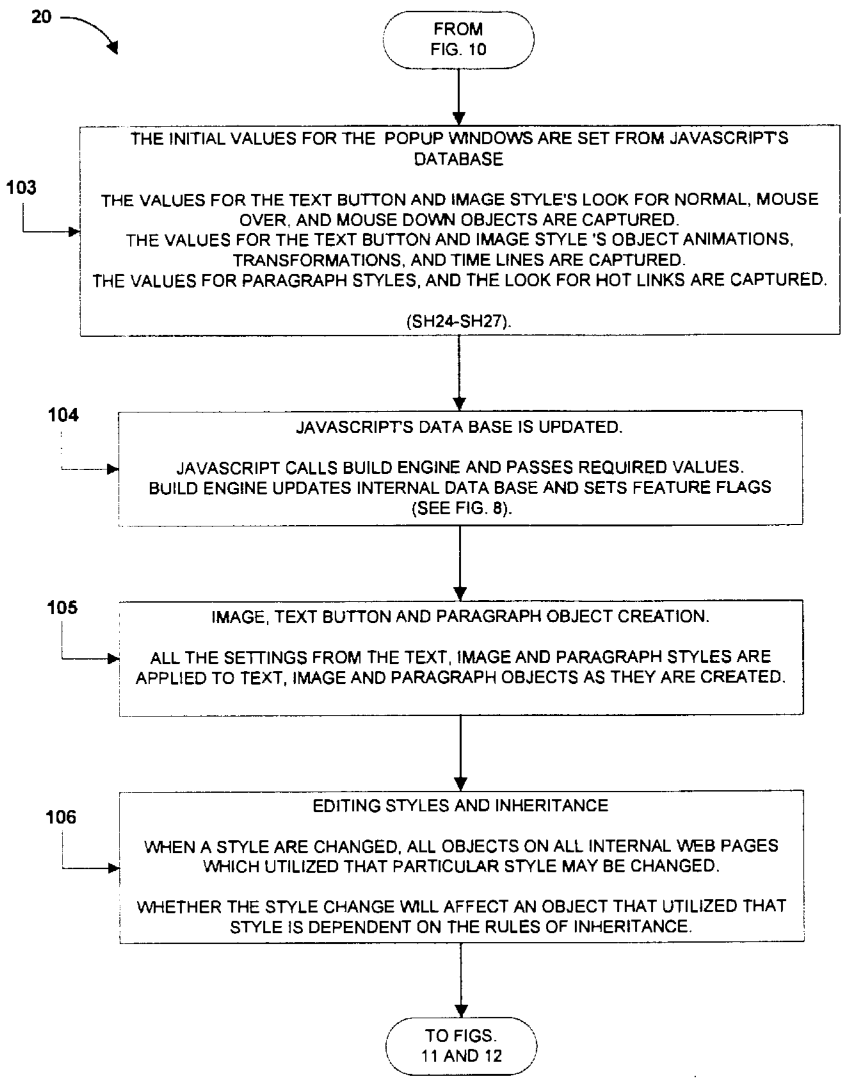

20
FROM FIG. 10
103
THE INITIAL VALUES FOR THE POPUP WINDOWS ARE SET FROM JAVASCRIPT'S DATABASE
THE VALUES FOR THE TEXT BUTTON AND IMAGE STYLE'S LOOK FOR NORMAL, MOUSE OVER, AND MOUSE DOWN OBJECTS ARE CAPTURED.
THE VALUES FOR THE TEXT BUTTON AND IMAGE STYLE 'S OBJECT ANIMATIONS, TRANSFORMATIONS, AND TIME LINES ARE CAPTURED.
THE VALUES FOR PARAGRAPH STYLES, AND THE LOOK FOR HOT LINKS ARE CAPTURED.
(SH24-SH27).
104
JAVASCRIPT'S DATA BASE IS UPDATED.
JAVASCRIPT CALLS BUILD ENGINE AND PASSES REQUIRED VALUES. BUILD ENGINE UPDATES INTERNAL DATA BASE AND SETS FEATURE FLAGS (SEE FIG. 8).
105
IMAGE, TEXT BUTTON AND PARAGRAPH OBJECT CREATION.
ALL THE SETTINGS FROM THE TEXT, IMAGE AND PARAGRAPH STYLES ARE APPLIED TO TEXT, IMAGE AND PARAGRAPH OBJECTS AS THEY ARE CREATED.
106
EDITING STYLES AND INHERITANCE
WHEN A STYLE ARE CHANGED, ALL OBJECTS ON ALL INTERNAL WEB PAGES WHICH UTILIZED THAT PARTICULAR STYLE MAY BE CHANGED.
WHETHER THE STYLE CHANGE WILL AFFECT AN OBJECT THAT UTILIZED THAT STYLE IS DEPENDENT ON THE RULES OF INHERITANCE.
TO FIGS. 11 AND 12

Fig. 13

BUTTON, IMAGE AND PARAGRAPH STYLE SETTINGS AND TECHNOLOGY

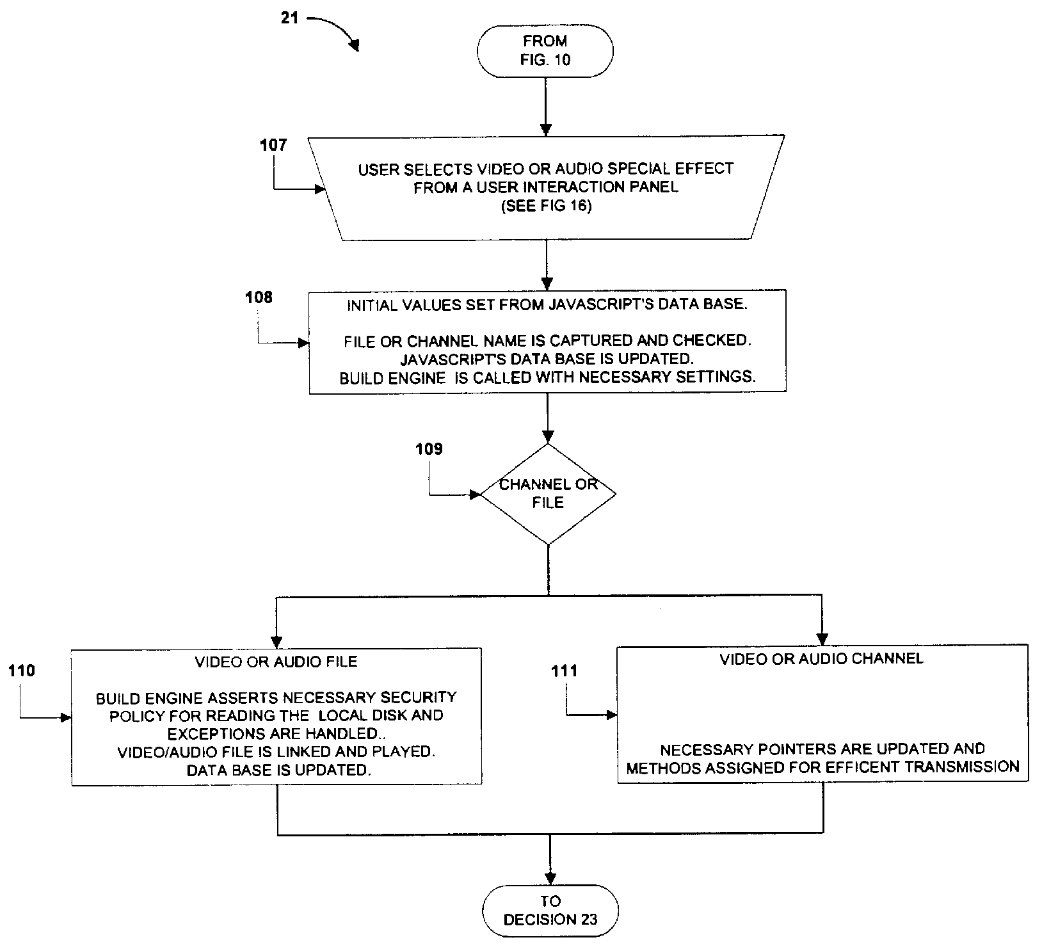

21
FROM FIG. 10
107
USER SELECTS VIDEO OR AUDIO SPECIAL EFFECT FROM A USER INTERACTION PANEL (SEE FIG 16)
108
INITIAL VALUES SET FROM JAVASCRIPT'S DATA BASE.
FILE OR CHANNEL NAME IS CAPTURED AND CHECKED. JAVASCRIPT'S DATA BASE IS UPDATED. BUILD ENGINE IS CALLED WITH NECESSARY SETTINGS.
109
CHANNEL OR FILE
110
VIDEO OR AUDIO FILE
BUILD ENGINE ASSERTS NECESSARY SECURITY POLICY FOR READING THE LOCAL DISK AND EXCEPTIONS ARE HANDLED.. VIDEO/AUDIO FILE IS LINKED AND PLAYED. DATA BASE IS UPDATED.
111
VIDEO OR AUDIO CHANNEL
NECESSARY POINTERS ARE UPDATED AND METHODS ASSIGNED FOR EFFICENT TRANSMISSION
TO DECISION 23

Fig. 14

VIDEO AND AUDIO FILE/CHANNEL PROCESSING

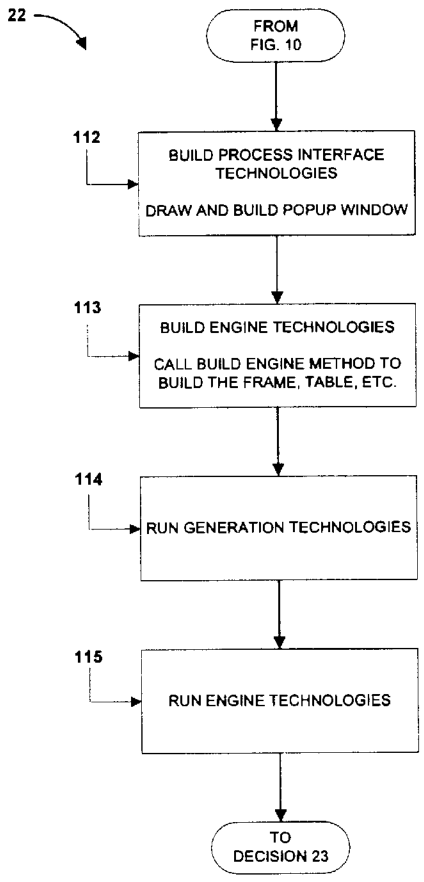

22
FROM FIG. 10
112
BUILD PROCESS INTERFACE TECHNOLOGIES
DRAW AND BUILD POPUP WINDOW
113
BUILD ENGINE TECHNOLOGIES
CALL BUILD ENGINE METHOD TO BUILD THE FRAME, TABLE, ETC.
114
RUN GENERATION TECHNOLOGIES
115
RUN ENGINE TECHNOLOGIES
TO DECISION 23

Fig. 15

FRAMES, TABLES, FORMS AND DRAW OBJECTS

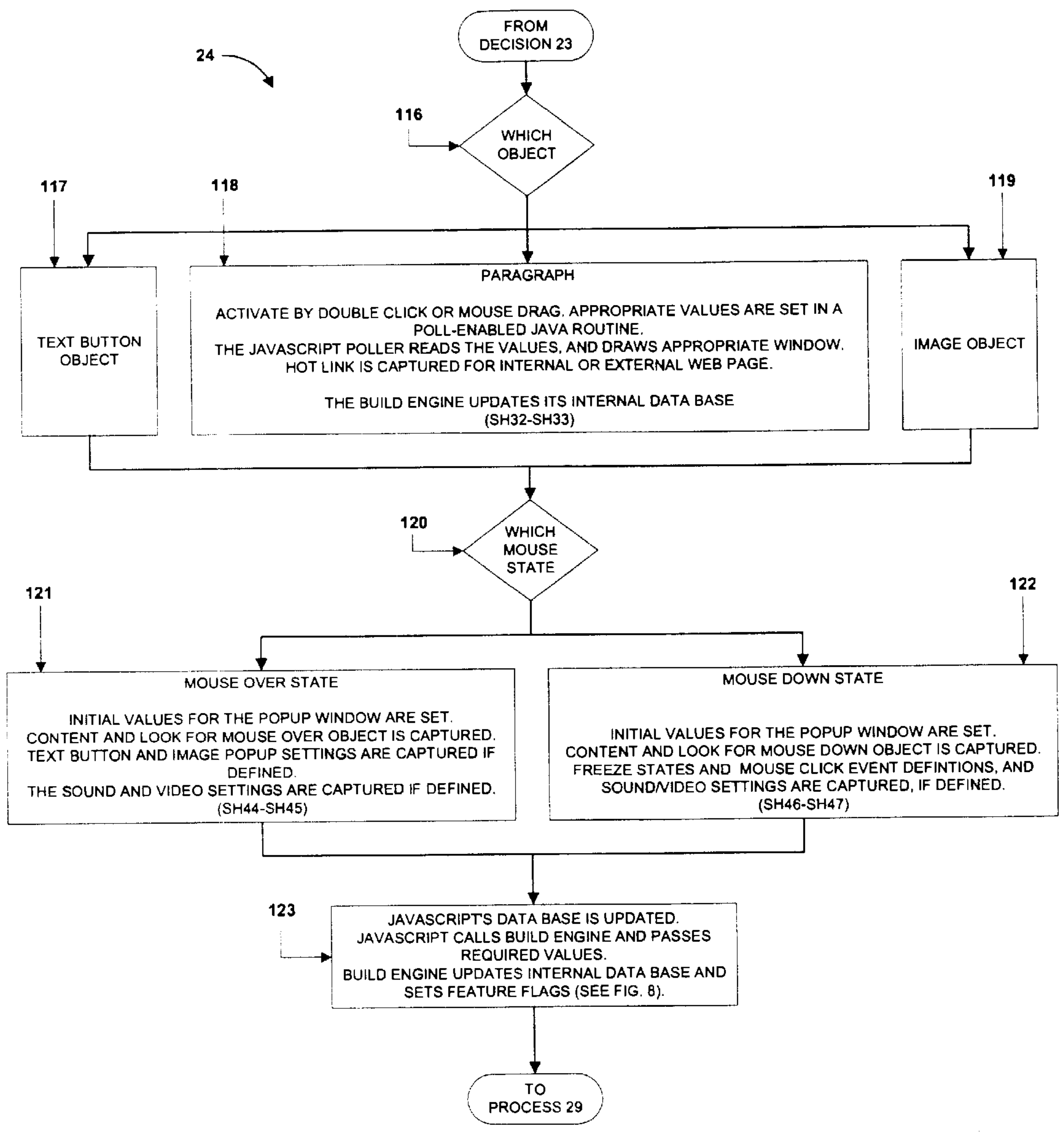

24
FROM DECISION 23
116
WHICH OBJECT
117
118
119
TEXT BUTTON OBJECT
PARAGRAPH
ACTIVATE BY DOUBLE CLICK OR MOUSE DRAG. APPROPRIATE VALUES ARE SET IN A POLL-ENABLED JAVA ROUTINE.
THE JAVASCRIPT POLLER READS THE VALUES, AND DRAWS APPROPRIATE WINDOW. HOT LINK IS CAPTURED FOR INTERNAL OR EXTERNAL WEB PAGE.
THE BUILD ENGINE UPDATES ITS INTERNAL DATA BASE (SH32-SH33)
IMAGE OBJECT
120
WHICH MOUSE STATE
121
122
MOUSE OVER STATE
INITIAL VALUES FOR THE POPUP WINDOW ARE SET. CONTENT AND LOOK FOR MOUSE OVER OBJECT IS CAPTURED. TEXT BUTTON AND IMAGE POPUP SETTINGS ARE CAPTURED IF DEFINED.
THE SOUND AND VIDEO SETTINGS ARE CAPTURED IF DEFINED. (SH44-SH45)
MOUSE DOWN STATE
INITIAL VALUES FOR THE POPUP WINDOW ARE SET. CONTENT AND LOOK FOR MOUSE DOWN OBJECT IS CAPTURED. FREEZE STATES AND MOUSE CLICK EVENT DEFINTIONS, AND SOUND/VIDEO SETTINGS ARE CAPTURED, IF DEFINED. (SH46-SH47)
123
JAVASCRIPT'S DATA BASE IS UPDATED. JAVASCRIPT CALLS BUILD ENGINE AND PASSES REQUIRED VALUES.
BUILD ENGINE UPDATES INTERNAL DATA BASE AND SETS FEATURE FLAGS (SEE FIG. 8).
TO PROCESS 29

Fig. 16

USER INTERACTION SETTINGS AND TECHNOLOGY

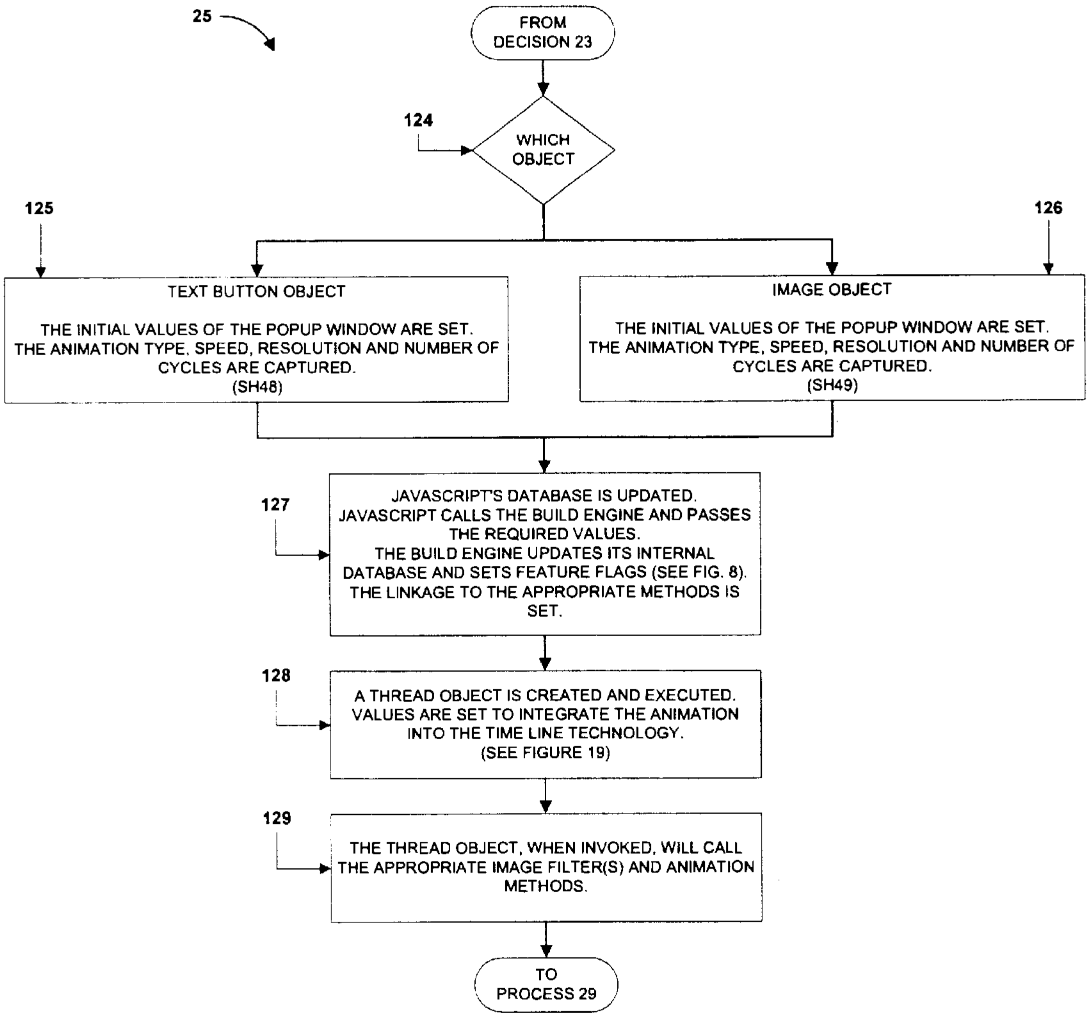

25
FROM DECISION 23
124
WHICH OBJECT
125
126
TEXT BUTTON OBJECT
THE INITIAL VALUES OF THE POPUP WINDOW ARE SET. THE ANIMATION TYPE, SPEED, RESOLUTION AND NUMBER OF CYCLES ARE CAPTURED. (SH48)
IMAGE OBJECT
THE INITIAL VALUES OF THE POPUP WINDOW ARE SET. THE ANIMATION TYPE, SPEED, RESOLUTION AND NUMBER OF CYCLES ARE CAPTURED. (SH49)
127
JAVASCRIPT'S DATABASE IS UPDATED. JAVASCRIPT CALLS THE BUILD ENGINE AND PASSES THE REQUIRED VALUES. THE BUILD ENGINE UPDATES ITS INTERNAL DATABASE AND SETS FEATURE FLAGS (SEE FIG. 8). THE LINKAGE TO THE APPROPRIATE METHODS IS SET.
128
A THREAD OBJECT IS CREATED AND EXECUTED. VALUES ARE SET TO INTEGRATE THE ANIMATION INTO THE TIME LINE TECHNOLOGY. (SEE FIGURE 19)
129
THE THREAD OBJECT, WHEN INVOKED, WILL CALL THE APPROPRIATE IMAGE FILTER(S) AND ANIMATION METHODS.
TO PROCESS 29

Fig. 17

ANIMATION SETTINGS AND TECHNOLOGY

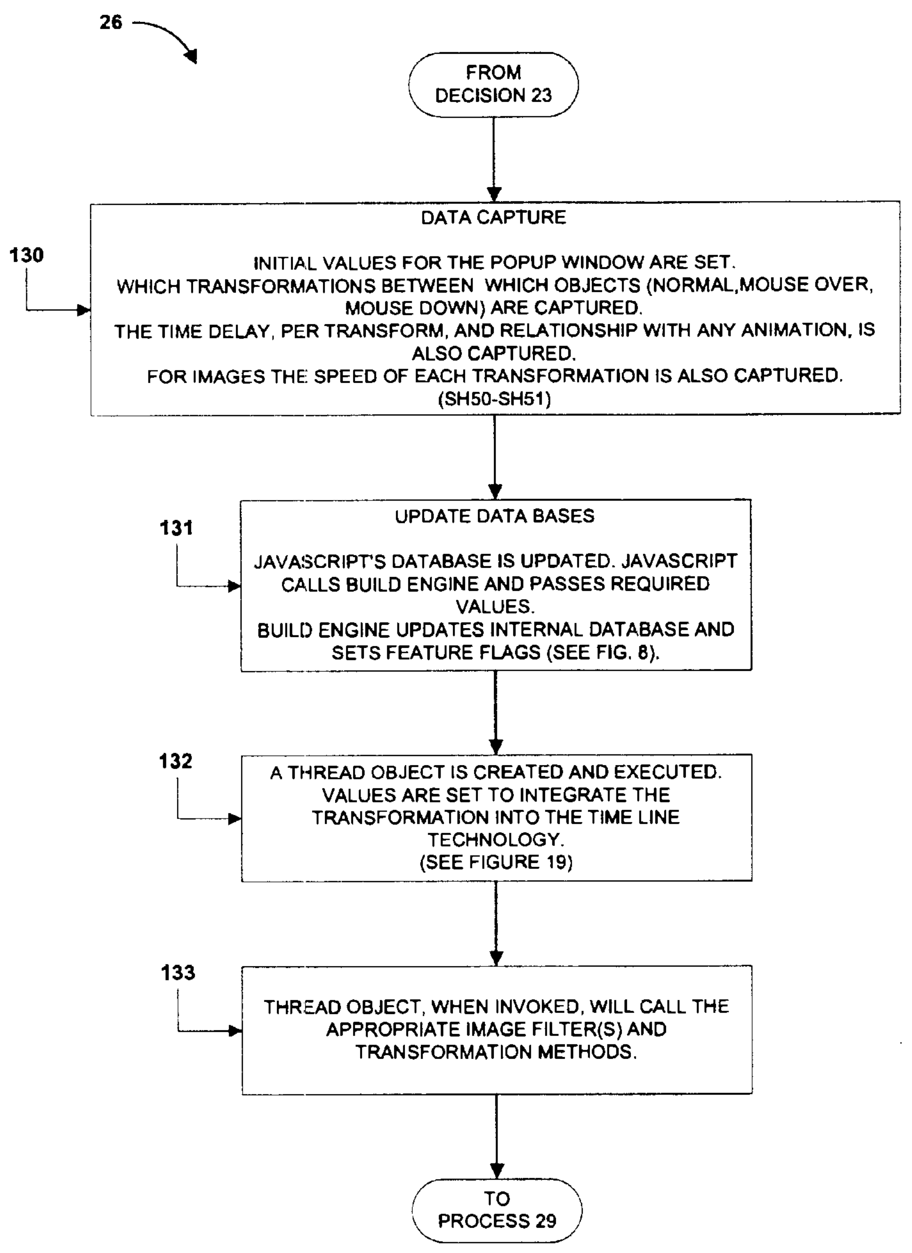
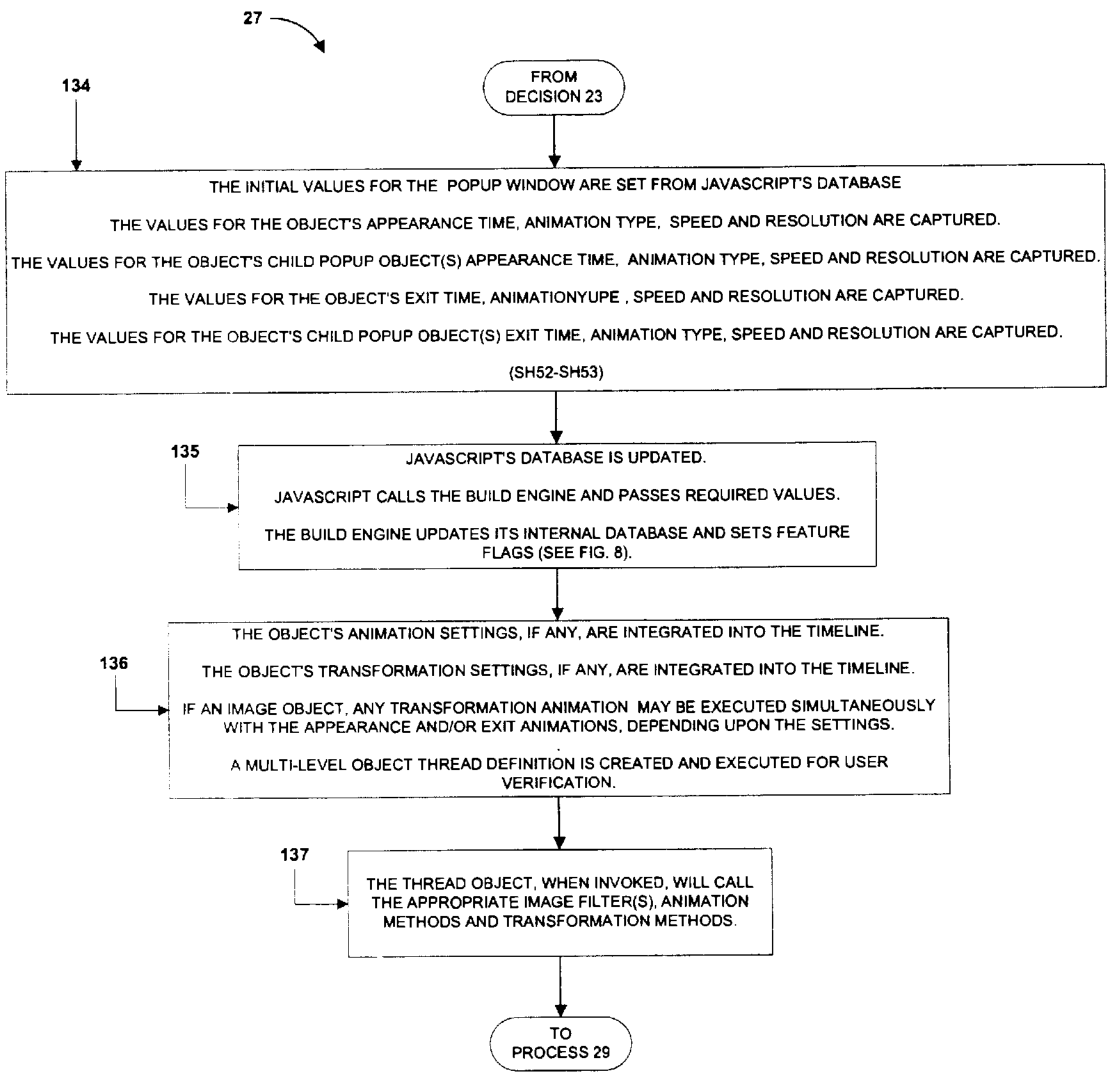

26
FROM DECISION 23
130
DATA CAPTURE
INITIAL VALUES FOR THE POPUP WINDOW ARE SET.
WHICH TRANSFORMATIONS BETWEEN WHICH OBJECTS (NORMAL,MOUSE OVER, MOUSE DOWN) ARE CAPTURED.
THE TIME DELAY, PER TRANSFORM, AND RELATIONSHIP WITH ANY ANIMATION, IS ALSO CAPTURED.
FOR IMAGES THE SPEED OF EACH TRANSFORMATION IS ALSO CAPTURED.
(SH50-SH51)
131
UPDATE DATA BASES
JAVASCRIPT'S DATABASE IS UPDATED. JAVASCRIPT CALLS BUILD ENGINE AND PASSES REQUIRED VALUES.
BUILD ENGINE UPDATES INTERNAL DATABASE AND SETS FEATURE FLAGS (SEE FIG. 8).
132
A THREAD OBJECT IS CREATED AND EXECUTED. VALUES ARE SET TO INTEGRATE THE TRANSFORMATION INTO THE TIME LINE TECHNOLOGY.
(SEE FIGURE 19)
133
THREAD OBJECT, WHEN INVOKED, WILL CALL THE APPROPRIATE IMAGE FILTER(S) AND TRANSFORMATION METHODS.
TO PROCESS 29

Fig. 18

TRANSFORMATION SETTINGS AND TECHNOLOGY



27
134
FROM DECISION 23
THE INITIAL VALUES FOR THE POPUP WINDOW ARE SET FROM JAVASCRIPT'S DATABASE
THE VALUES FOR THE OBJECT'S APPEARANCE TIME, ANIMATION TYPE, SPEED AND RESOLUTION ARE CAPTURED.
THE VALUES FOR THE OBJECT'S CHILD POPUP OBJECT(S) APPEARANCE TIME, ANIMATION TYPE, SPEED AND RESOLUTION ARE CAPTURED.
THE VALUES FOR THE OBJECT'S EXIT TIME, ANIMATIONYUPE , SPEED AND RESOLUTION ARE CAPTURED.
THE VALUES FOR THE OBJECT'S CHILD POPUP OBJECT(S) EXIT TIME, ANIMATION TYPE, SPEED AND RESOLUTION ARE CAPTURED.
(SH52-SH53)
135
JAVASCRIPT'S DATABASE IS UPDATED.
JAVASCRIPT CALLS THE BUILD ENGINE AND PASSES REQUIRED VALUES.
THE BUILD ENGINE UPDATES ITS INTERNAL DATABASE AND SETS FEATURE FLAGS (SEE FIG. 8).
136
THE OBJECT'S ANIMATION SETTINGS, IF ANY, ARE INTEGRATED INTO THE TIMELINE.
THE OBJECT'S TRANSFORMATION SETTINGS, IF ANY, ARE INTEGRATED INTO THE TIMELINE.
IF AN IMAGE OBJECT, ANY TRANSFORMATION ANIMATION MAY BE EXECUTED SIMULTANEOUSLY WITH THE APPEARANCE AND/OR EXIT ANIMATIONS, DEPENDING UPON THE SETTINGS.
A MULTI-LEVEL OBJECT THREAD DEFINITION IS CREATED AND EXECUTED FOR USER VERIFICATION.
137
THE THREAD OBJECT, WHEN INVOKED, WILL CALL THE APPROPRIATE IMAGE FILTER(S), ANIMATION METHODS AND TRANSFORMATION METHODS.
TO PROCESS 29

Fig. 19

OBJECT TIME LINES AND TECHNOLOGY

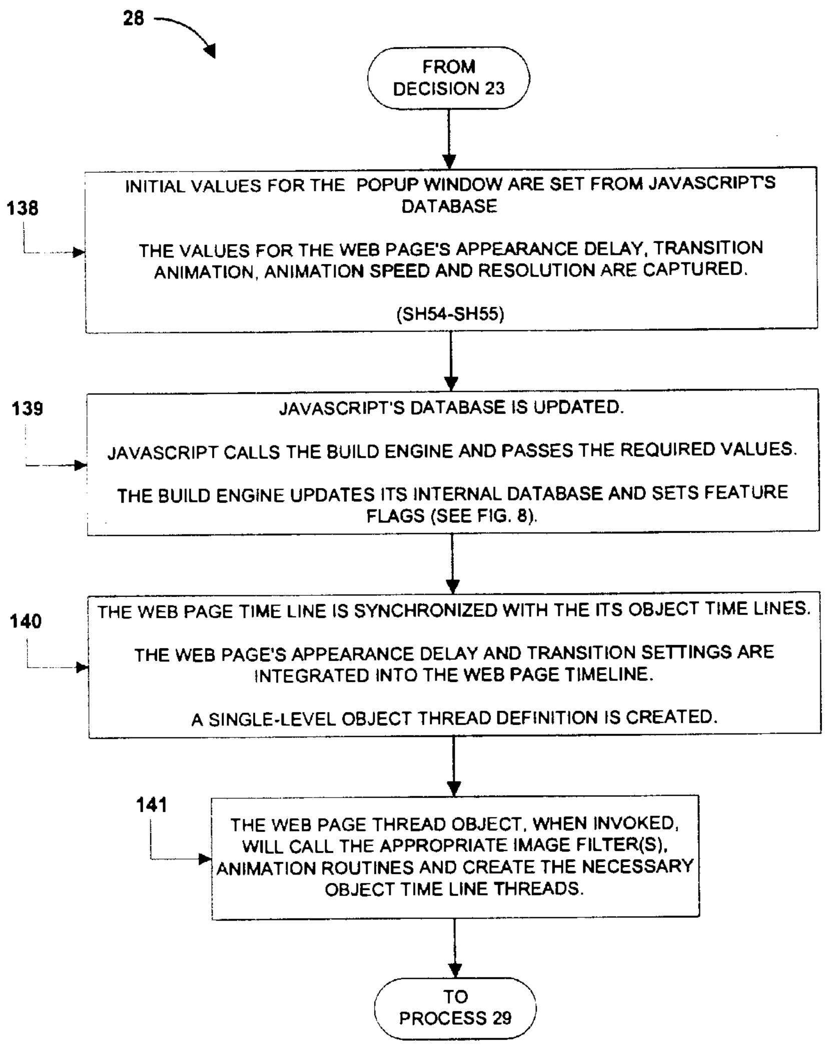

28
FROM DECISION 23
138
INITIAL VALUES FOR THE POPUP WINDOW ARE SET FROM JAVASCRIPT'S DATABASE
THE VALUES FOR THE WEB PAGE'S APPEARANCE DELAY, TRANSITION ANIMATION, ANIMATION SPEED AND RESOLUTION ARE CAPTURED.
(SH54-SH55)
139
JAVASCRIPT'S DATABASE IS UPDATED.
JAVASCRIPT CALLS THE BUILD ENGINE AND PASSES THE REQUIRED VALUES.
THE BUILD ENGINE UPDATES ITS INTERNAL DATABASE AND SETS FEATURE FLAGS (SEE FIG. 8).
140
THE WEB PAGE TIME LINE IS SYNCHRONIZED WITH THE ITS OBJECT TIME LINES.
THE WEB PAGE'S APPEARANCE DELAY AND TRANSITION SETTINGS ARE INTEGRATED INTO THE WEB PAGE TIMELINE.
A SINGLE-LEVEL OBJECT THREAD DEFINITION IS CREATED.
141
THE WEB PAGE THREAD OBJECT, WHEN INVOKED, WILL CALL THE APPROPRIATE IMAGE FILTER(S), ANIMATION ROUTINES AND CREATE THE NECESSARY OBJECT TIME LINE THREADS.
TO PROCESS 29

Fig. 20

WEB PAGE TRANSITION ANIMATIONS, TIME LINE SETTINGS AND TECHNOLOGY

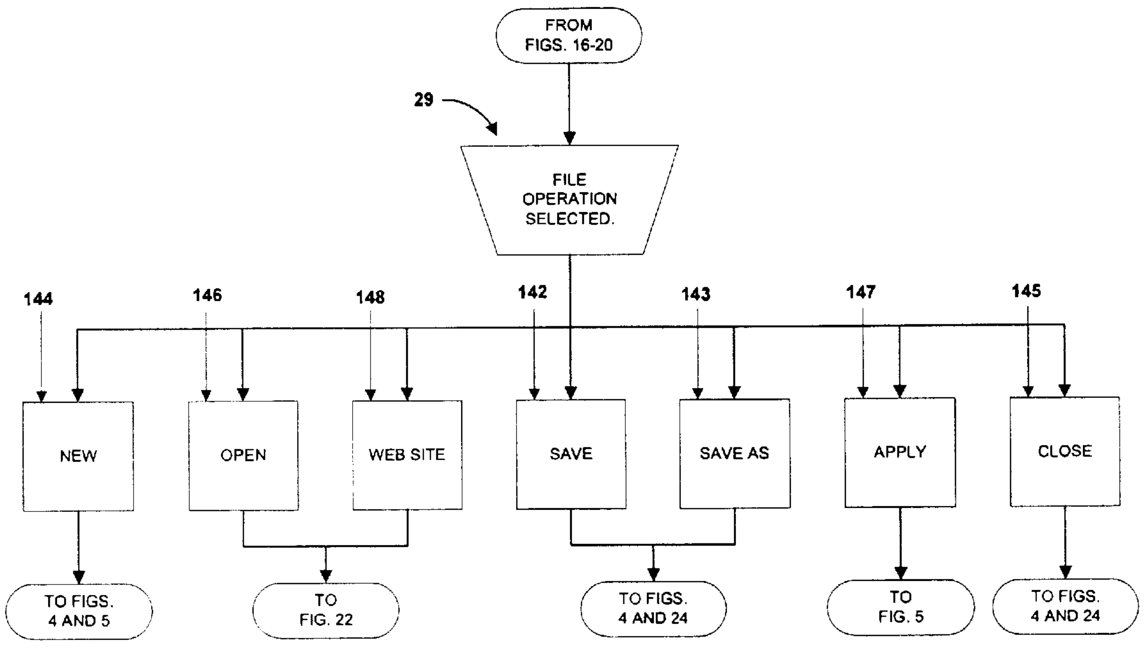


fig. 21a

FILE OPERATIONS

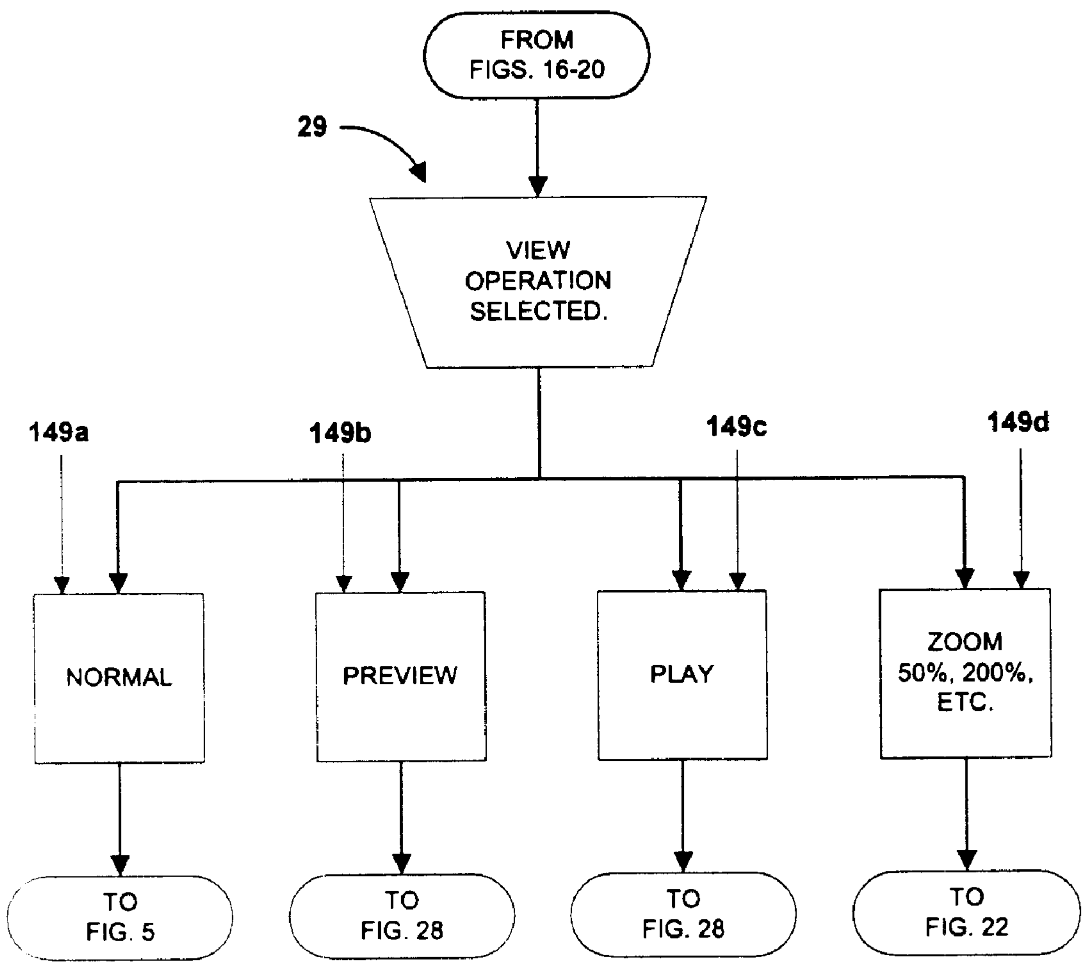


fig. 21b

VIEW OPERATIONS

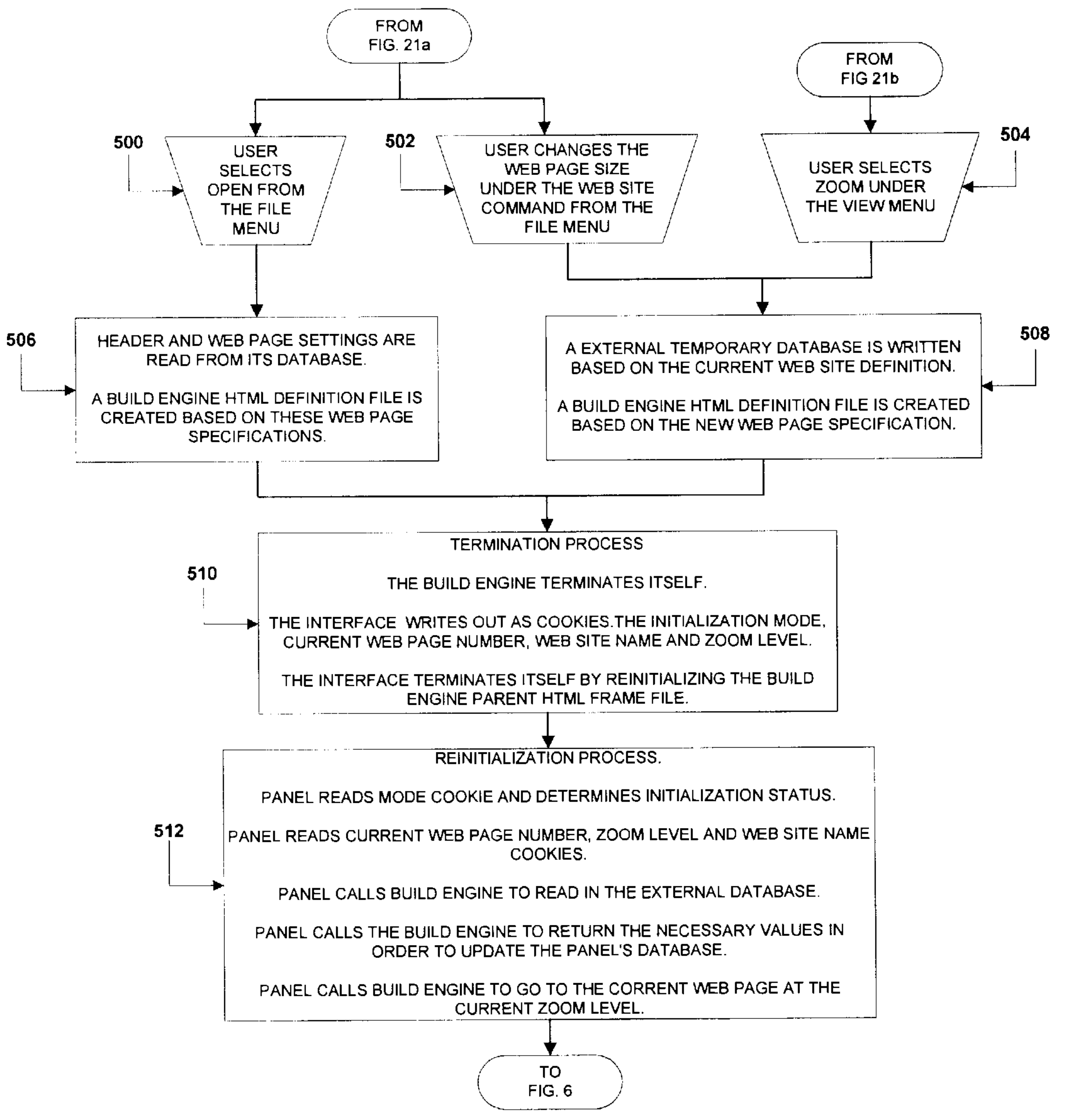

FROM FIG. 21a
FROM FIG 21b
500
USER SELECTS OPEN FROM THE FILE MENU
502
USER CHANGES THE WEB PAGE SIZE UNDER THE WEB SITE COMMAND FROM THE FILE MENU
504
USER SELECTS ZOOM UNDER THE VIEW MENU
506
HEADER AND WEB PAGE SETTINGS ARE READ FROM ITS DATABASE.
A BUILD ENGINE HTML DEFINITION FILE IS CREATED BASED ON THESE WEB PAGE SPECIFICATIONS.
508
A EXTERNAL TEMPORARY DATABASE IS WRITTEN BASED ON THE CURRENT WEB SITE DEFINITION.
A BUILD ENGINE HTML DEFINITION FILE IS CREATED BASED ON THE NEW WEB PAGE SPECIFICATION.
510
TERMINATION PROCESS
THE BUILD ENGINE TERMINATES ITSELF.
THE INTERFACE WRITES OUT AS COOKIES.THE INITIALIZATION MODE, CURRENT WEB PAGE NUMBER, WEB SITE NAME AND ZOOM LEVEL.
THE INTERFACE TERMINATES ITSELF BY REINITIALIZING THE BUILD ENGINE PARENT HTML FRAME FILE.
512
REINITIALIZATION PROCESS.
PANEL READS MODE COOKIE AND DETERMINES INITIALIZATION STATUS.
PANEL READS CURRENT WEB PAGE NUMBER, ZOOM LEVEL AND WEB SITE NAME COOKIES.
PANEL CALLS BUILD ENGINE TO READ IN THE EXTERNAL DATABASE.
PANEL CALLS THE BUILD ENGINE TO RETURN THE NECESSARY VALUES IN ORDER TO UPDATE THE PANEL'S DATABASE.
PANEL CALLS BUILD ENGINE TO GO TO THE CORRENT WEB PAGE AT THE CURRENT ZOOM LEVEL.
TO FIG. 6

Fig. 22

DYNAMIC WEB PAGE RESIZING PROCESS

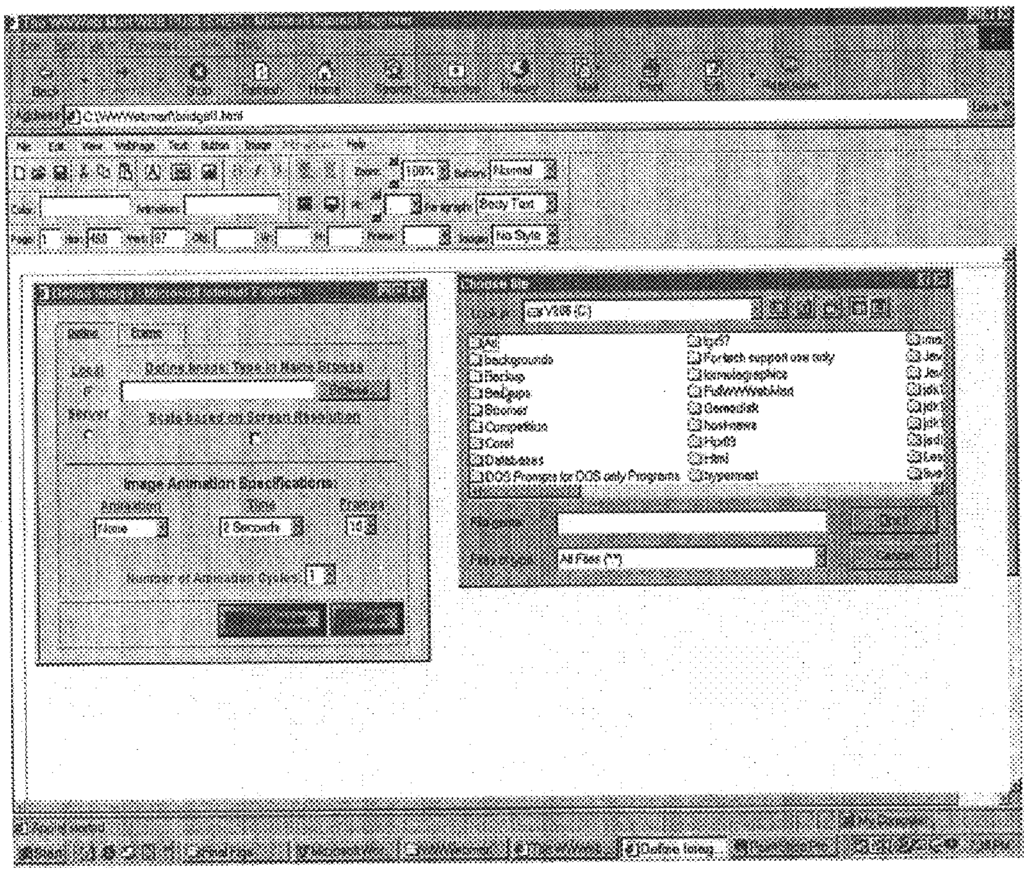

Address C:\WWWebmart\bridge0.html
Zoom 100%
Body Text
No Style
Define Image: Type in Name Browse
Scale based on Screen Resolution
Image Animation Specifications
Animation
Time
Frames
None
2 Seconds
10
Number of Animation Cycles 1
Choose file
(C:)
backgrounds
Backup
Borland
Competition
Corel
Databases
DOS Prompts for DOS only Programs
Fortech support use only
Gamedisk
hyperwest
File name
Files of type
All Files (*.*)
Define Imag

Fig. 23

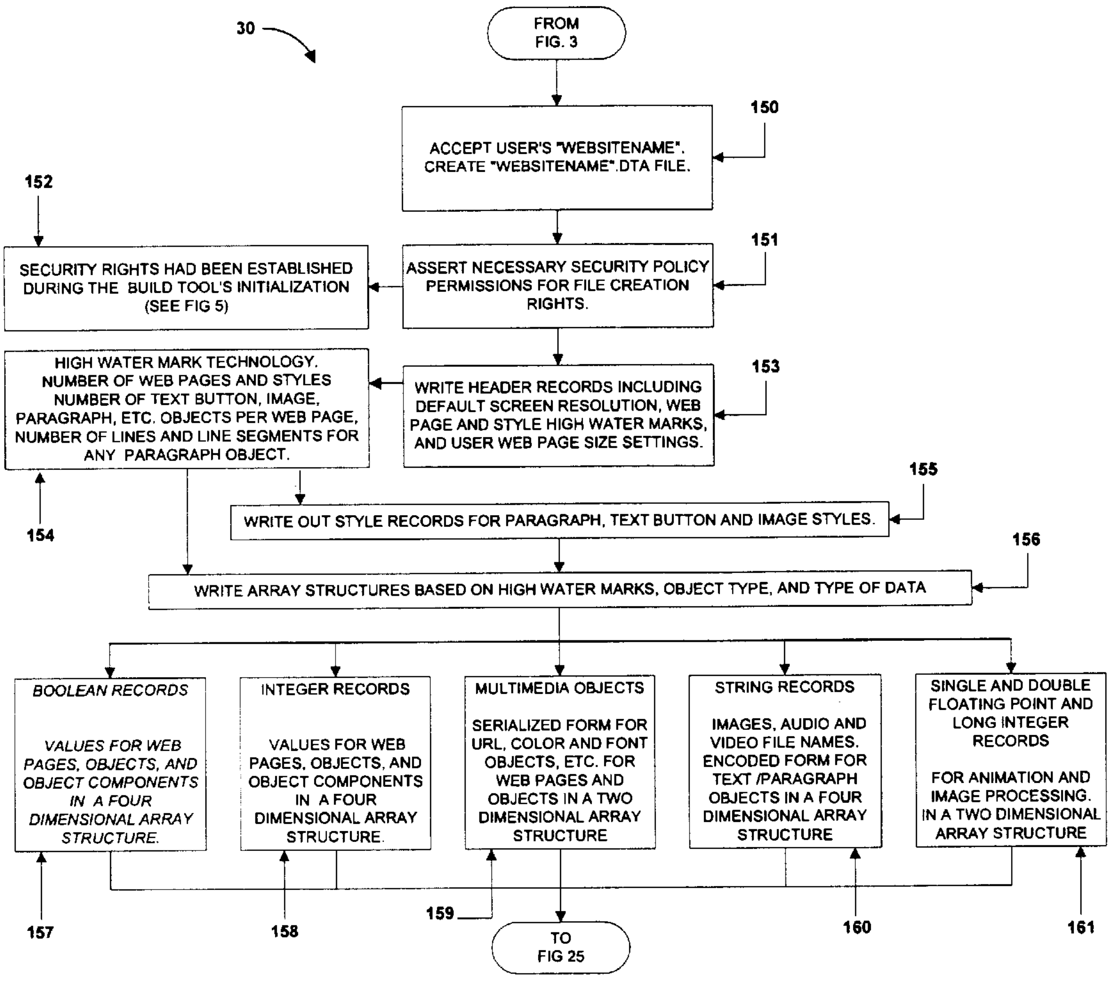

30
FROM FIG. 3
ACCEPT USER'S "WEBSITENAME". CREATE "WEBSITENAME".DTA FILE.
150
152
SECURITY RIGHTS HAD BEEN ESTABLISHED DURING THE BUILD TOOL'S INITIALIZATION (SEE FIG 5)
ASSERT NECESSARY SECURITY POLICY PERMISSIONS FOR FILE CREATION RIGHTS.
151
HIGH WATER MARK TECHNOLOGY. NUMBER OF WEB PAGES AND STYLES NUMBER OF TEXT BUTTON, IMAGE, PARAGRAPH, ETC. OBJECTS PER WEB PAGE, NUMBER OF LINES AND LINE SEGMENTS FOR ANY PARAGRAPH OBJECT.
WRITE HEADER RECORDS INCLUDING DEFAULT SCREEN RESOLUTION, WEB PAGE AND STYLE HIGH WATER MARKS, AND USER WEB PAGE SIZE SETTINGS.
153
155
154
WRITE OUT STYLE RECORDS FOR PARAGRAPH, TEXT BUTTON AND IMAGE STYLES.
156
WRITE ARRAY STRUCTURES BASED ON HIGH WATER MARKS, OBJECT TYPE, AND TYPE OF DATA
BOOLEAN RECORDS VALUES FOR WEB PAGES, OBJECTS, AND OBJECT COMPONENTS IN A FOUR DIMENSIONAL ARRAY STRUCTURE.
INTEGER RECORDS VALUES FOR WEB PAGES, OBJECTS, AND OBJECT COMPONENTS IN A FOUR DIMENSIONAL ARRAY STRUCTURE.
MULTIMEDIA OBJECTS SERIALIZED FORM FOR URL, COLOR AND FONT OBJECTS, ETC. FOR WEB PAGES AND OBJECTS IN A TWO DIMENSIONAL ARRAY STRUCTURE
STRING RECORDS IMAGES, AUDIO AND VIDEO FILE NAMES. ENCODED FORM FOR TEXT /PARAGRAPH OBJECTS IN A FOUR DIMENSIONAL ARRAY STRUCTURE
SINGLE AND DOUBLE FLOATING POINT AND LONG INTEGER RECORDS FOR ANIMATION AND IMAGE PROCESSING. IN A TWO DIMENSIONAL ARRAY STRUCTURE
157
158
159
TO FIG 25
160
161

Fig. 24

EXTERNAL DATA BASE CREATION:
SECURITY AND OPTIMIZATION TECHNIQUES

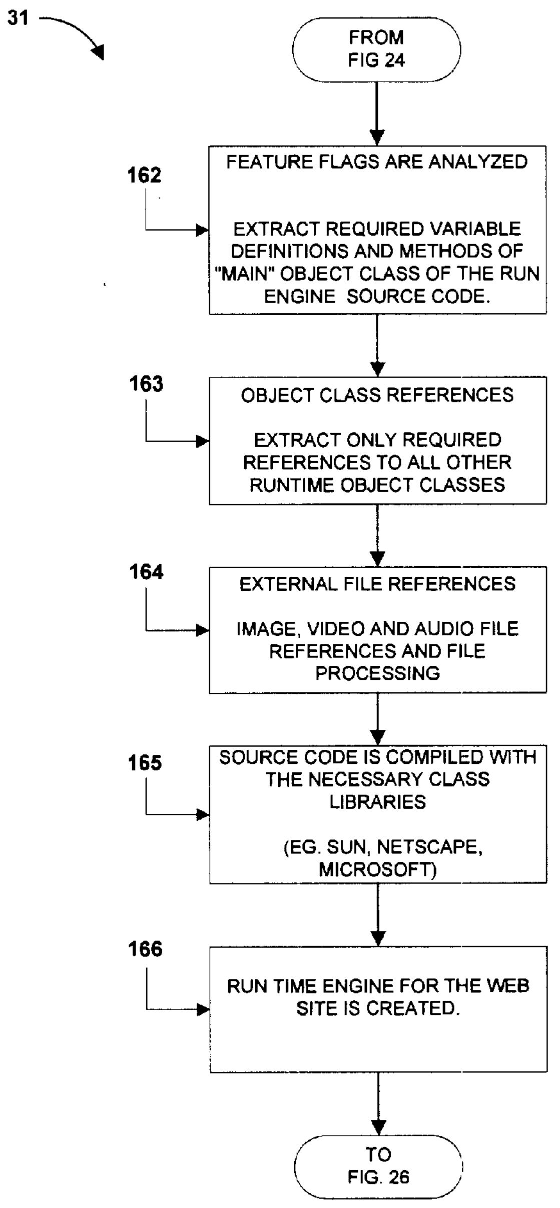

31
FROM FIG 24
162
FEATURE FLAGS ARE ANALYZED
EXTRACT REQUIRED VARIABLE DEFINITIONS AND METHODS OF "MAIN" OBJECT CLASS OF THE RUN ENGINE SOURCE CODE.
163
OBJECT CLASS REFERENCES
EXTRACT ONLY REQUIRED REFERENCES TO ALL OTHER RUNTIME OBJECT CLASSES
164
EXTERNAL FILE REFERENCES
IMAGE, VIDEO AND AUDIO FILE REFERENCES AND FILE PROCESSING
165
SOURCE CODE IS COMPILED WITH THE NECESSARY CLASS LIBRARIES
(EG. SUN, NETSCAPE, MICROSOFT)
166
RUN TIME ENGINE FOR THE WEB SITE IS CREATED.
TO FIG. 26

Fig. 25

CREATE CUSTOMIZED AND OPTIMIZED RUNTIME ENGINE

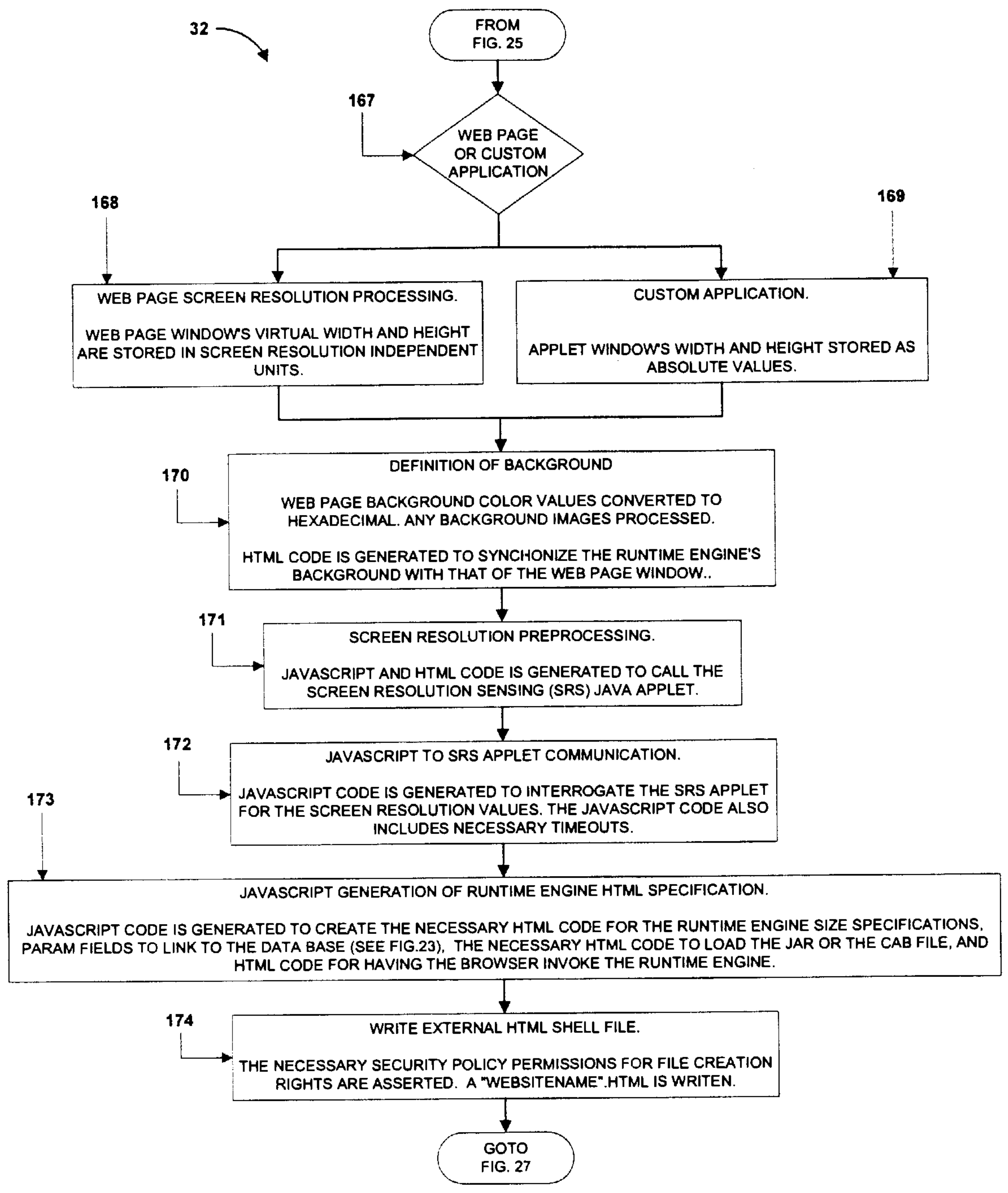

32
FROM FIG. 25
167
WEB PAGE OR CUSTOM APPLICATION
168
169
WEB PAGE SCREEN RESOLUTION PROCESSING.
WEB PAGE WINDOW'S VIRTUAL WIDTH AND HEIGHT ARE STORED IN SCREEN RESOLUTION INDEPENDENT UNITS.
CUSTOM APPLICATION.
APPLET WINDOW'S WIDTH AND HEIGHT STORED AS ABSOLUTE VALUES.
170
DEFINITION OF BACKGROUND
WEB PAGE BACKGROUND COLOR VALUES CONVERTED TO HEXADECIMAL. ANY BACKGROUND IMAGES PROCESSED.
HTML CODE IS GENERATED TO SYNCHONIZE THE RUNTIME ENGINE'S BACKGROUND WITH THAT OF THE WEB PAGE WINDOW..
171
SCREEN RESOLUTION PREPROCESSING.
JAVASCRIPT AND HTML CODE IS GENERATED TO CALL THE SCREEN RESOLUTION SENSING (SRS) JAVA APPLET.
172
JAVASCRIPT TO SRS APPLET COMMUNICATION.
JAVASCRIPT CODE IS GENERATED TO INTERROGATE THE SRS APPLET FOR THE SCREEN RESOLUTION VALUES. THE JAVASCRIPT CODE ALSO INCLUDES NECESSARY TIMEOUTS.
173
JAVASCRIPT GENERATION OF RUNTIME ENGINE HTML SPECIFICATION.
JAVASCRIPT CODE IS GENERATED TO CREATE THE NECESSARY HTML CODE FOR THE RUNTIME ENGINE SIZE SPECIFICATIONS, PARAM FIELDS TO LINK TO THE DATA BASE (SEE FIG.23), THE NECESSARY HTML CODE TO LOAD THE JAR OR THE CAB FILE, AND HTML CODE FOR HAVING THE BROWSER INVOKE THE RUNTIME ENGINE.
174
WRITE EXTERNAL HTML SHELL FILE.
THE NECESSARY SECURITY POLICY PERMISSIONS FOR FILE CREATION RIGHTS ARE ASSERTED. A "WEBSITENAME".HTML IS WRITEN.
GOTO FIG. 27

Fig. 26

CREATE THE HTML SHELL FILE

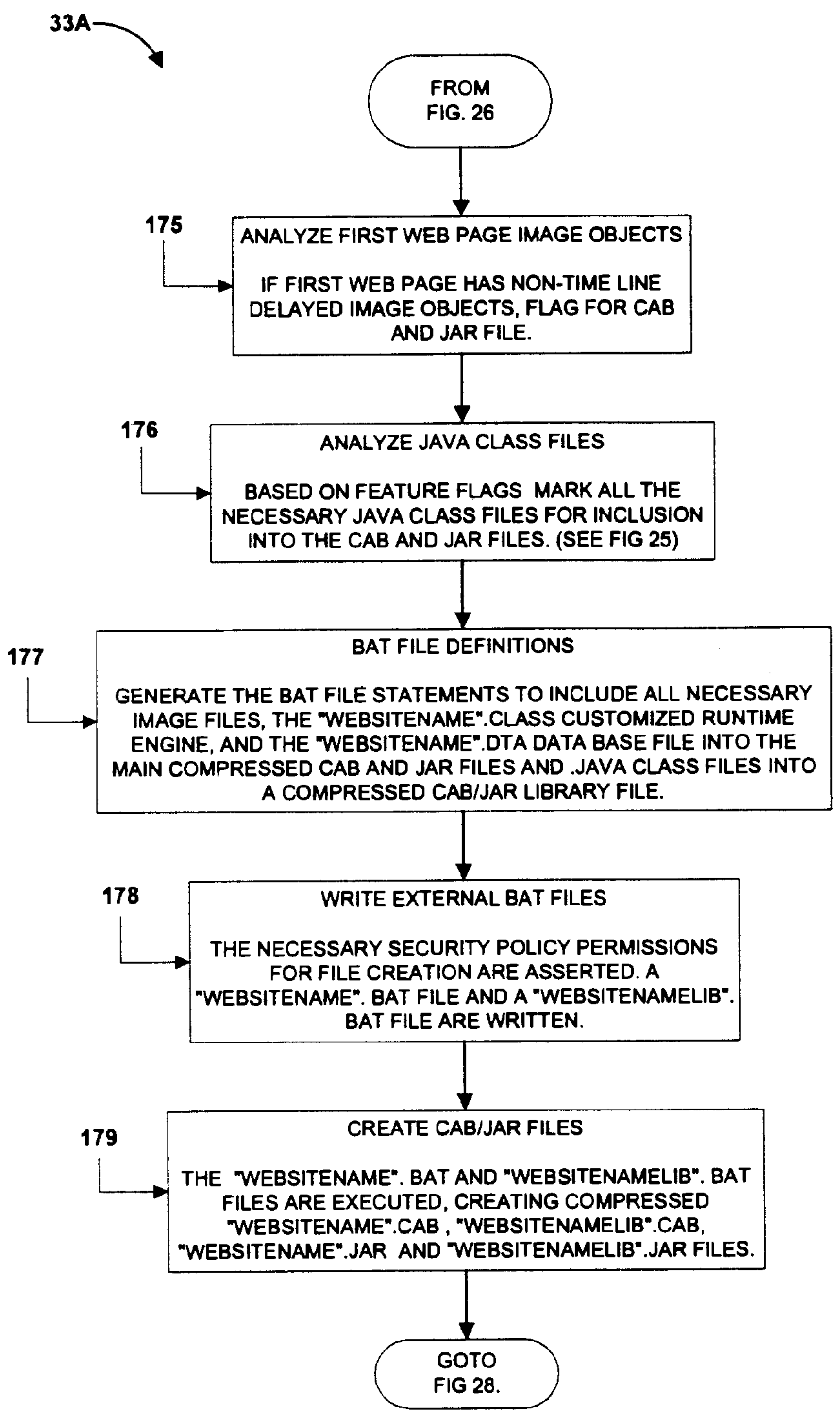

33A
FROM FIG. 26
175
ANALYZE FIRST WEB PAGE IMAGE OBJECTS
IF FIRST WEB PAGE HAS NON-TIME LINE DELAYED IMAGE OBJECTS, FLAG FOR CAB AND JAR FILE.
176
ANALYZE JAVA CLASS FILES
BASED ON FEATURE FLAGS MARK ALL THE NECESSARY JAVA CLASS FILES FOR INCLUSION INTO THE CAB AND JAR FILES. (SEE FIG 25)
177
BAT FILE DEFINITIONS
GENERATE THE BAT FILE STATEMENTS TO INCLUDE ALL NECESSARY IMAGE FILES, THE "WEBSITENAME".CLASS CUSTOMIZED RUNTIME ENGINE, AND THE "WEBSITENAME".DTA DATA BASE FILE INTO THE MAIN COMPRESSED CAB AND JAR FILES AND .JAVA CLASS FILES INTO A COMPRESSED CAB/JAR LIBRARY FILE.
178
WRITE EXTERNAL BAT FILES
THE NECESSARY SECURITY POLICY PERMISSIONS FOR FILE CREATION ARE ASSERTED. A "WEBSITENAME". BAT FILE AND A "WEBSITENAMELIB". BAT FILE ARE WRITTEN.
179
CREATE CAB/JAR FILES
THE "WEBSITENAME". BAT AND "WEBSITENAMELIB". BAT FILES ARE EXECUTED, CREATING COMPRESSED "WEBSITENAME".CAB , "WEBSITENAMELIB".CAB, "WEBSITENAME".JAR AND "WEBSITENAMELIB".JAR FILES.
GOTO FIG 28.

Fig. 27

CREATE THE CAB/JAR FILES

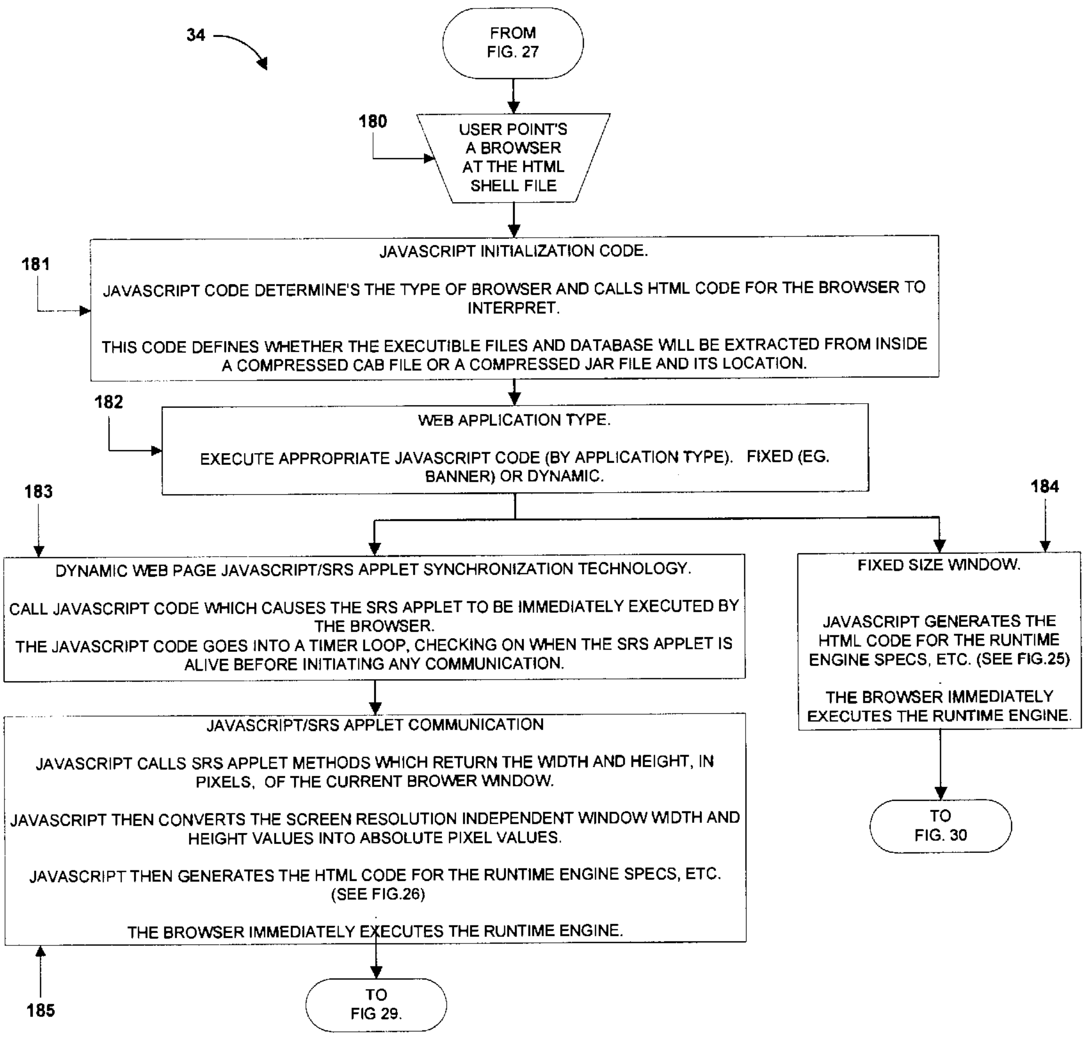

34
FROM FIG. 27
180
USER POINT'S A BROWSER AT THE HTML SHELL FILE
181
JAVASCRIPT INITIALIZATION CODE.
JAVASCRIPT CODE DETERMINE'S THE TYPE OF BROWSER AND CALLS HTML CODE FOR THE BROWSER TO INTERPRET.
THIS CODE DEFINES WHETHER THE EXECUTIBLE FILES AND DATABASE WILL BE EXTRACTED FROM INSIDE A COMPRESSED CAB FILE OR A COMPRESSED JAR FILE AND ITS LOCATION.
182
WEB APPLICATION TYPE.
EXECUTE APPROPRIATE JAVASCRIPT CODE (BY APPLICATION TYPE). FIXED (EG. BANNER) OR DYNAMIC.
183
DYNAMIC WEB PAGE JAVASCRIPT/SRS APPLET SYNCHRONIZATION TECHNOLOGY.
CALL JAVASCRIPT CODE WHICH CAUSES THE SRS APPLET TO BE IMMEDIATELY EXECUTED BY THE BROWSER.
THE JAVASCRIPT CODE GOES INTO A TIMER LOOP, CHECKING ON WHEN THE SRS APPLET IS ALIVE BEFORE INITIATING ANY COMMUNICATION.
184
FIXED SIZE WINDOW.
JAVASCRIPT GENERATES THE HTML CODE FOR THE RUNTIME ENGINE SPECS, ETC. (SEE FIG.25)
THE BROWSER IMMEDIATELY EXECUTES THE RUNTIME ENGINE.
JAVASCRIPT/SRS APPLET COMMUNICATION
JAVASCRIPT CALLS SRS APPLET METHODS WHICH RETURN THE WIDTH AND HEIGHT, IN PIXELS, OF THE CURRENT BROWER WINDOW.
JAVASCRIPT THEN CONVERTS THE SCREEN RESOLUTION INDEPENDENT WINDOW WIDTH AND HEIGHT VALUES INTO ABSOLUTE PIXEL VALUES.
JAVASCRIPT THEN GENERATES THE HTML CODE FOR THE RUNTIME ENGINE SPECS, ETC. (SEE FIG.26)
THE BROWSER IMMEDIATELY EXECUTES THE RUNTIME ENGINE.
TO FIG. 30
185
TO FIG 29.

Fig. 28

WEB PAGE SIZE GENERATION TECHNOLOGY

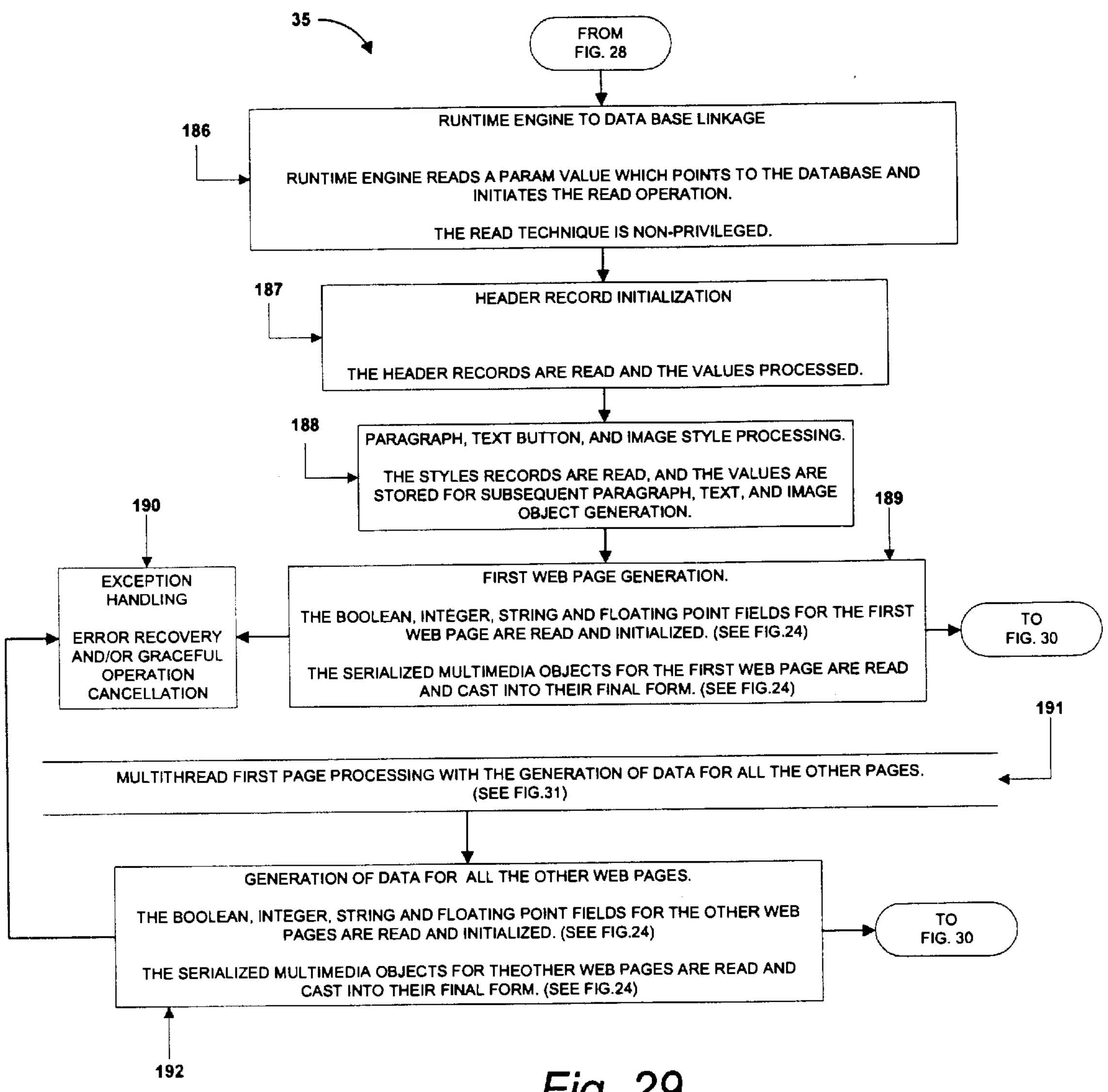

35
FROM FIG. 28
186
RUNTIME ENGINE TO DATA BASE LINKAGE
RUNTIME ENGINE READS A PARAM VALUE WHICH POINTS TO THE DATABASE AND INITIATES THE READ OPERATION.
THE READ TECHNIQUE IS NON-PRIVILEGED.
187
HEADER RECORD INITIALIZATION
THE HEADER RECORDS ARE READ AND THE VALUES PROCESSED.
188
PARAGRAPH, TEXT BUTTON, AND IMAGE STYLE PROCESSING.
THE STYLES RECORDS ARE READ, AND THE VALUES ARE STORED FOR SUBSEQUENT PARAGRAPH, TEXT, AND IMAGE OBJECT GENERATION.
190
189
EXCEPTION HANDLING
ERROR RECOVERY AND/OR GRACEFUL OPERATION CANCELLATION
FIRST WEB PAGE GENERATION.
THE BOOLEAN, INTEGER, STRING AND FLOATING POINT FIELDS FOR THE FIRST WEB PAGE ARE READ AND INITIALIZED. (SEE FIG.24)
THE SERIALIZED MULTIMEDIA OBJECTS FOR THE FIRST WEB PAGE ARE READ AND CAST INTO THEIR FINAL FORM. (SEE FIG.24)
TO FIG. 30
191
MULTITHREAD FIRST PAGE PROCESSING WITH THE GENERATION OF DATA FOR ALL THE OTHER PAGES. (SEE FIG.31)
GENERATION OF DATA FOR ALL THE OTHER WEB PAGES.
THE BOOLEAN, INTEGER, STRING AND FLOATING POINT FIELDS FOR THE OTHER WEB PAGES ARE READ AND INITIALIZED. (SEE FIG.24)
THE SERIALIZED MULTIMEDIA OBJECTS FOR THEOTHER WEB PAGES ARE READ AND CAST INTO THEIR FINAL FORM. (SEE FIG.24)
TO FIG. 30
192

Fig. 29

READ DATA BASE AND GENERATE NECESSARY OBJECTS.

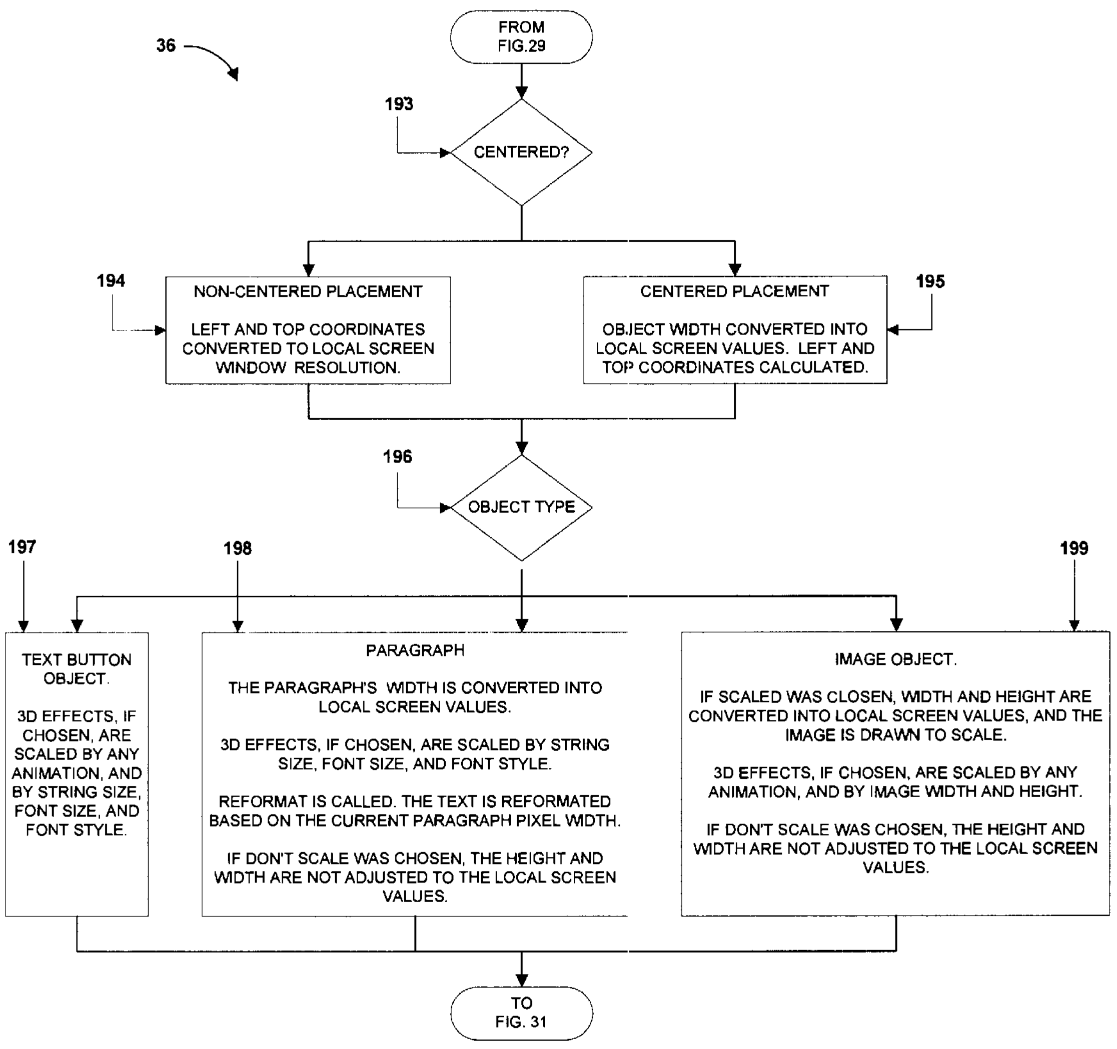

36
FROM FIG.29
193
CENTERED?
194
NON-CENTERED PLACEMENT
LEFT AND TOP COORDINATES CONVERTED TO LOCAL SCREEN WINDOW RESOLUTION.
CENTERED PLACEMENT
OBJECT WIDTH CONVERTED INTO LOCAL SCREEN VALUES. LEFT AND TOP COORDINATES CALCULATED.
195
196
OBJECT TYPE
197
198
199
TEXT BUTTON OBJECT.
3D EFFECTS, IF CHOSEN, ARE SCALED BY ANY ANIMATION, AND BY STRING SIZE, FONT SIZE, AND FONT STYLE.
PARAGRAPH
THE PARAGRAPH'S WIDTH IS CONVERTED INTO LOCAL SCREEN VALUES.
3D EFFECTS, IF CHOSEN, ARE SCALED BY STRING SIZE, FONT SIZE, AND FONT STYLE.
REFORMAT IS CALLED. THE TEXT IS REFORMATED BASED ON THE CURRENT PARAGRAPH PIXEL WIDTH.
IF DON'T SCALE WAS CHOSEN, THE HEIGHT AND WIDTH ARE NOT ADJUSTED TO THE LOCAL SCREEN VALUES.
IMAGE OBJECT.
IF SCALED WAS CLOSEN, WIDTH AND HEIGHT ARE CONVERTED INTO LOCAL SCREEN VALUES, AND THE IMAGE IS DRAWN TO SCALE.
3D EFFECTS, IF CHOSEN, ARE SCALED BY ANY ANIMATION, AND BY IMAGE WIDTH AND HEIGHT.
IF DON'T SCALE WAS CHOSEN, THE HEIGHT AND WIDTH ARE NOT ADJUSTED TO THE LOCAL SCREEN VALUES.
TO FIG. 31

Fig. 30

WEB PAGE GENERATION WITH SCALING TECHNOLOGY

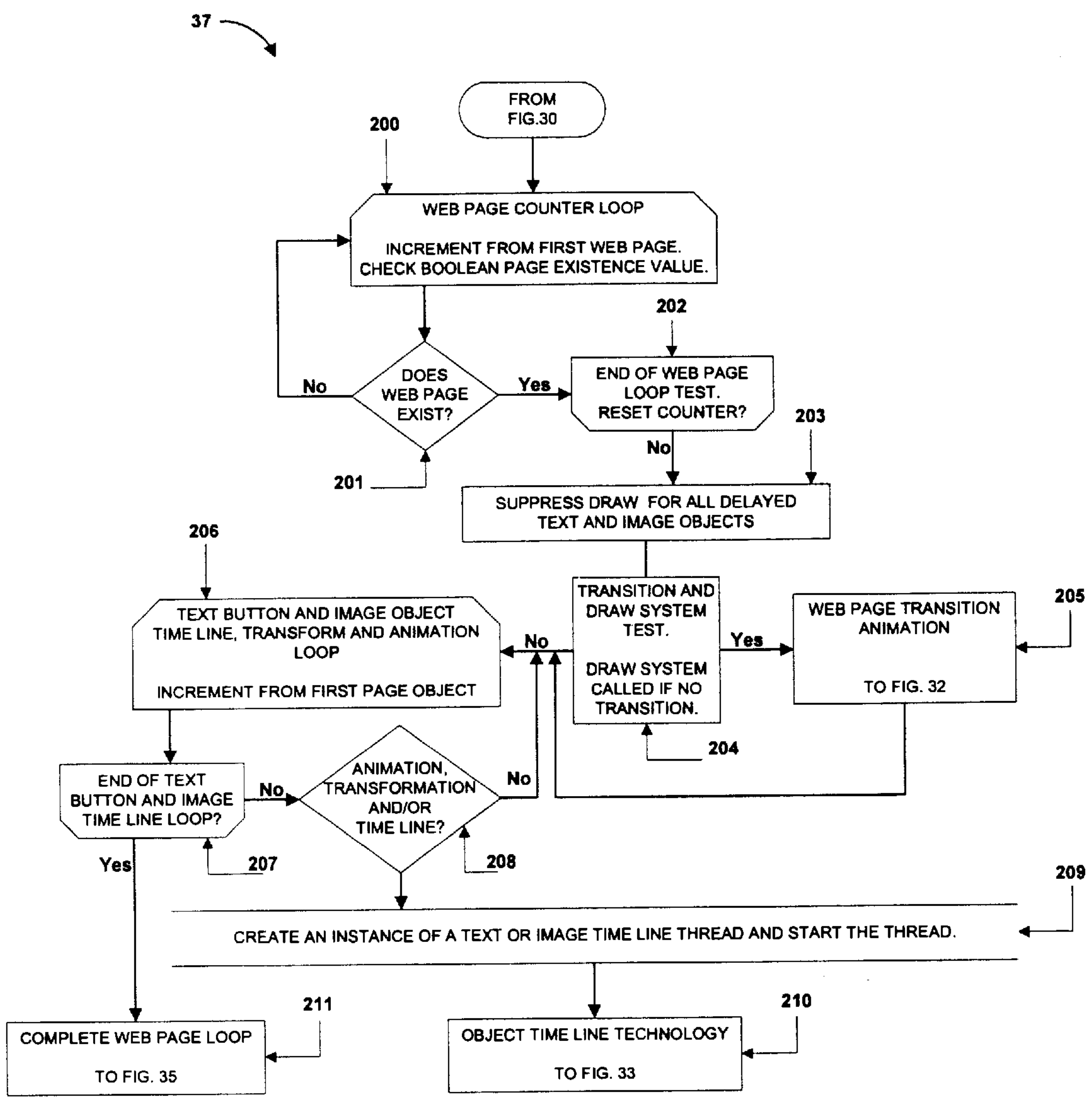

37
FROM FIG.30
200
WEB PAGE COUNTER LOOP
INCREMENT FROM FIRST WEB PAGE. CHECK BOOLEAN PAGE EXISTENCE VALUE.
DOES WEB PAGE EXIST?
No
Yes
201
202
END OF WEB PAGE LOOP TEST. RESET COUNTER?
No
203
SUPPRESS DRAW FOR ALL DELAYED TEXT AND IMAGE OBJECTS
206
TEXT BUTTON AND IMAGE OBJECT TIME LINE, TRANSFORM AND ANIMATION LOOP
INCREMENT FROM FIRST PAGE OBJECT
TRANSITION AND DRAW SYSTEM TEST.
DRAW SYSTEM CALLED IF NO TRANSITION.
No
Yes
WEB PAGE TRANSITION ANIMATION
TO FIG. 32
205
204
END OF TEXT BUTTON AND IMAGE TIME LINE LOOP?
No
ANIMATION, TRANSFORMATION AND/OR TIME LINE?
No
Yes
207
208
209
CREATE AN INSTANCE OF A TEXT OR IMAGE TIME LINE THREAD AND START THE THREAD.
211
COMPLETE WEB PAGE LOOP
TO FIG. 35
210
OBJECT TIME LINE TECHNOLOGY
TO FIG. 33

Fig. 31

THE MULTILEVEL WEB PAGE AND OBJECT THREAD TECHNOLOGY

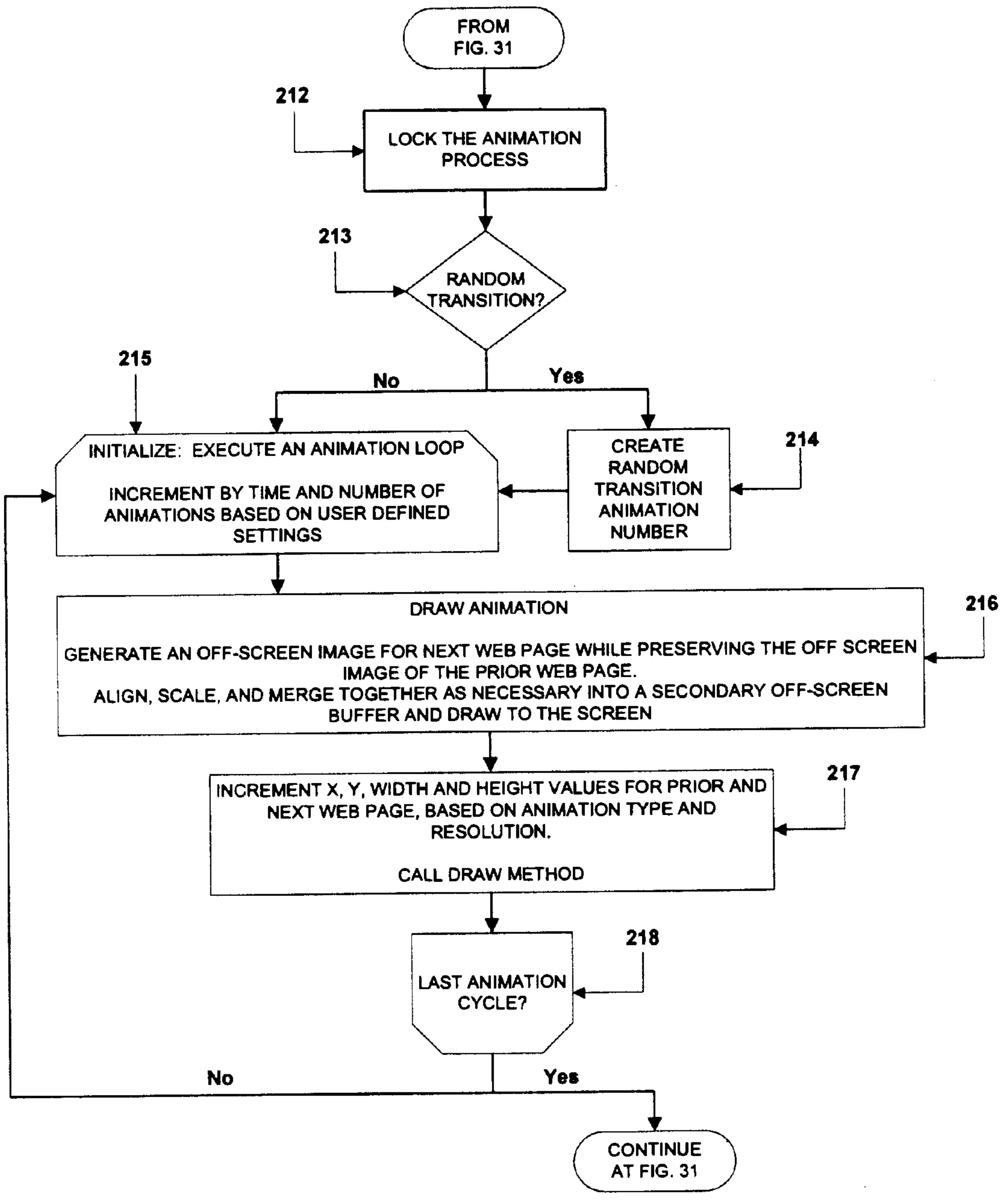

FROM FIG. 31
212
LOCK THE ANIMATION PROCESS
213
RANDOM TRANSITION?
215
No
Yes
INITIALIZE: EXECUTE AN ANIMATION LOOP
INCREMENT BY TIME AND NUMBER OF ANIMATIONS BASED ON USER DEFINED SETTINGS
CREATE RANDOM TRANSITION ANIMATION NUMBER
214
DRAW ANIMATION
GENERATE AN OFF-SCREEN IMAGE FOR NEXT WEB PAGE WHILE PRESERVING THE OFF SCREEN IMAGE OF THE PRIOR WEB PAGE.
ALIGN, SCALE, AND MERGE TOGETHER AS NECESSARY INTO A SECONDARY OFF-SCREEN BUFFER AND DRAW TO THE SCREEN
216
INCREMENT X, Y, WIDTH AND HEIGHT VALUES FOR PRIOR AND NEXT WEB PAGE, BASED ON ANIMATION TYPE AND RESOLUTION.
CALL DRAW METHOD
217
LAST ANIMATION CYCLE?
218
No
Yes
CONTINUE AT FIG. 31

Fig. 32

WEB PAGE TRANSITION ANIMATION

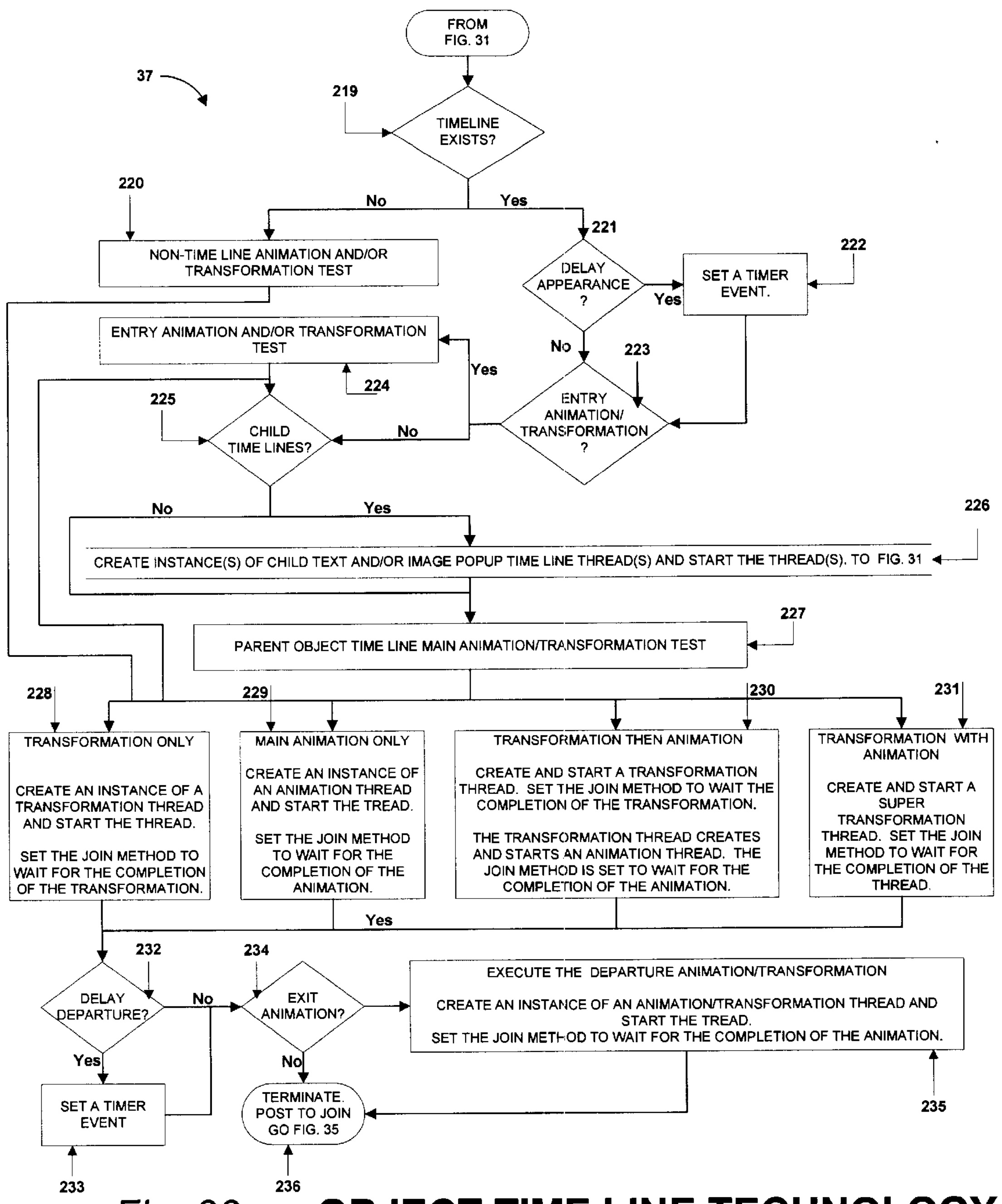

FROM FIG. 31
37
219
TIMELINE EXISTS?
220
No
Yes
221
NON-TIME LINE ANIMATION AND/OR TRANSFORMATION TEST
DELAY APPEARANCE ?
Yes
SET A TIMER EVENT.
222
ENTRY ANIMATION AND/OR TRANSFORMATION TEST
Yes
No
223
224
225
CHILD TIME LINES?
No
ENTRY ANIMATION/ TRANSFORMATION ?
No
Yes
226
CREATE INSTANCE(S) OF CHILD TEXT AND/OR IMAGE POPUP TIME LINE THREAD(S) AND START THE THREAD(S). TO FIG. 31
227
PARENT OBJECT TIME LINE MAIN ANIMATION/TRANSFORMATION TEST
228
229
230
231
TRANSFORMATION ONLY
CREATE AN INSTANCE OF A TRANSFORMATION THREAD AND START THE THREAD.
SET THE JOIN METHOD TO WAIT FOR THE COMPLETION OF THE TRANSFORMATION.
MAIN ANIMATION ONLY
CREATE AN INSTANCE OF AN ANIMATION THREAD AND START THE TREAD.
SET THE JOIN METHOD TO WAIT FOR THE COMPLETION OF THE ANIMATION.
TRANSFORMATION THEN ANIMATION
CREATE AND START A TRANSFORMATION THREAD. SET THE JOIN METHOD TO WAIT THE COMPLETION OF THE TRANSFORMATION.
THE TRANSFORMATION THREAD CREATES AND STARTS AN ANIMATION THREAD. THE JOIN METHOD IS SET TO WAIT FOR THE COMPLETION OF THE ANIMATION.
TRANSFORMATION WITH ANIMATION
CREATE AND START A SUPER TRANSFORMATION THREAD. SET THE JOIN METHOD TO WAIT FOR THE COMPLETION OF THE THREAD.
Yes
232
234
DELAY DEPARTURE?
No
EXIT ANIMATION?
EXECUTE THE DEPARTURE ANIMATION/TRANSFORMATION
CREATE AN INSTANCE OF AN ANIMATION/TRANSFORMATION THREAD AND START THE TREAD.
SET THE JOIN METHOD TO WAIT FOR THE COMPLETION OF THE ANIMATION.
Yes
No
SET A TIMER EVENT
TERMINATE. POST TO JOIN GO FIG. 35
235
233
236

Fig. 33 **OBJECT TIME LINE TECHNOLOGY**

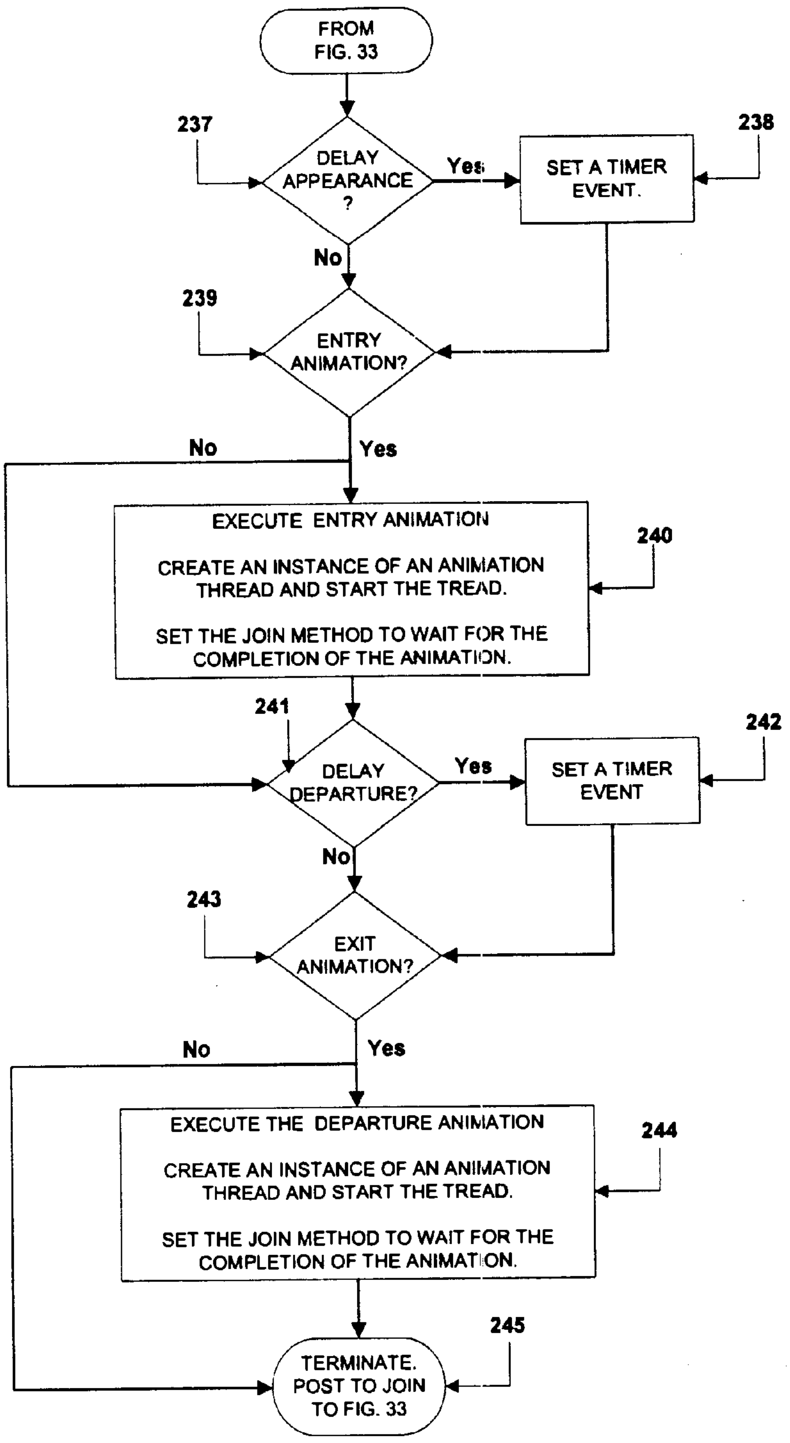

FROM FIG. 33
237
DELAY APPEARANCE ?
Yes
SET A TIMER EVENT.
238
No
239
ENTRY ANIMATION?
No
Yes
EXECUTE ENTRY ANIMATION
CREATE AN INSTANCE OF AN ANIMATION THREAD AND START THE TREAD.
SET THE JOIN METHOD TO WAIT FOR THE COMPLETION OF THE ANIMATION.
240
241
DELAY DEPARTURE?
Yes
SET A TIMER EVENT
242
No
243
EXIT ANIMATION?
No
Yes
EXECUTE THE DEPARTURE ANIMATION
CREATE AN INSTANCE OF AN ANIMATION THREAD AND START THE TREAD.
SET THE JOIN METHOD TO WAIT FOR THE COMPLETION OF THE ANIMATION.
244
245
TERMINATE. POST TO JOIN TO FIG. 33

Fig. 34

CHILD TIME LINES FOR TEXT BUTTON AND IMAGE OBJECTS

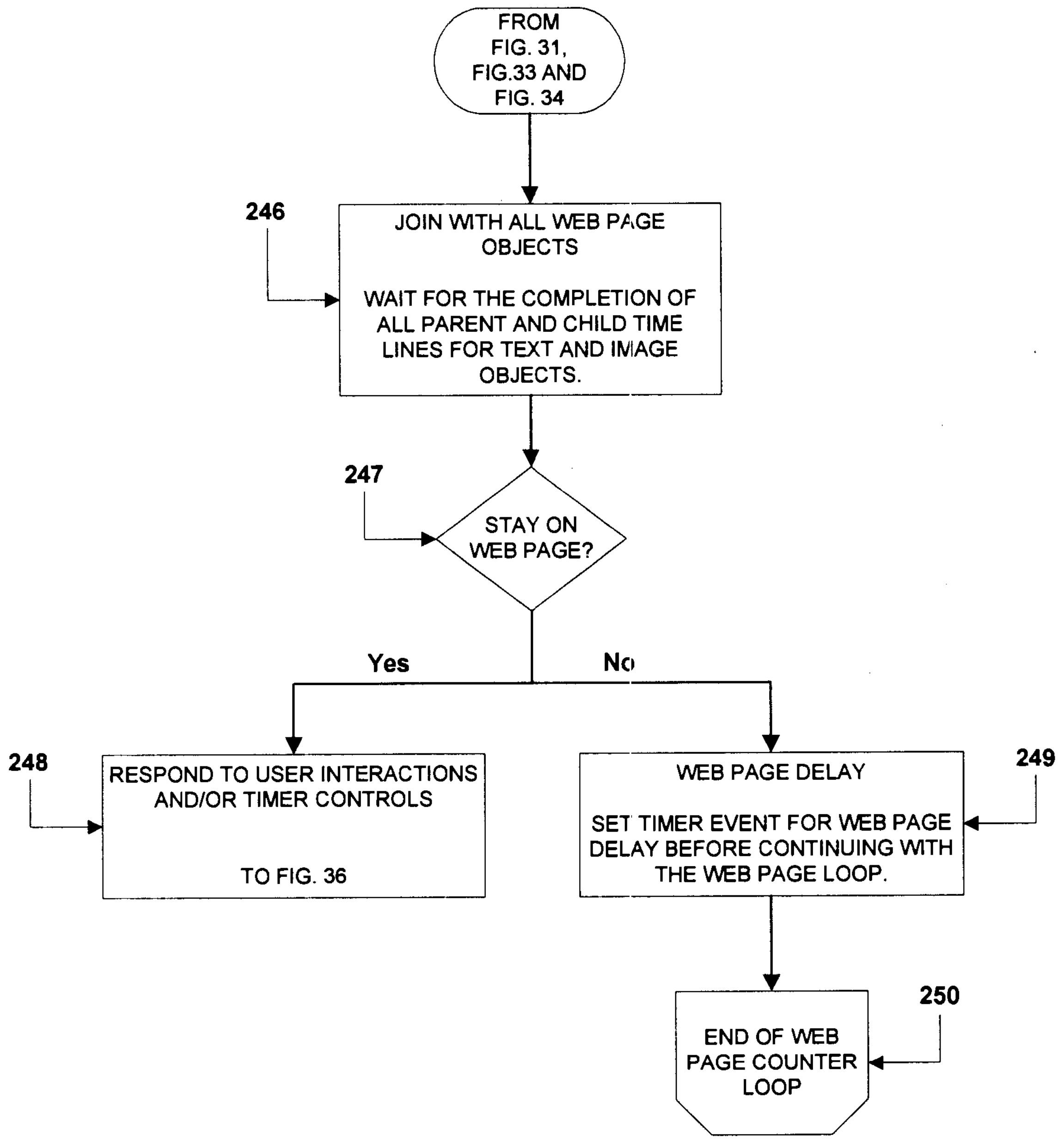

FROM FIG. 31, FIG.33 AND FIG. 34
246
JOIN WITH ALL WEB PAGE OBJECTS
WAIT FOR THE COMPLETION OF ALL PARENT AND CHILD TIME LINES FOR TEXT AND IMAGE OBJECTS.
247
STAY ON WEB PAGE?
Yes
No
248
RESPOND TO USER INTERACTIONS AND/OR TIMER CONTROLS
TO FIG. 36
249
WEB PAGE DELAY
SET TIMER EVENT FOR WEB PAGE DELAY BEFORE CONTINUING WITH THE WEB PAGE LOOP.
250
END OF WEB PAGE COUNTER LOOP

Fig. 35

COMPLETE WEB PAGE AND OBJECT THREAD LOOP

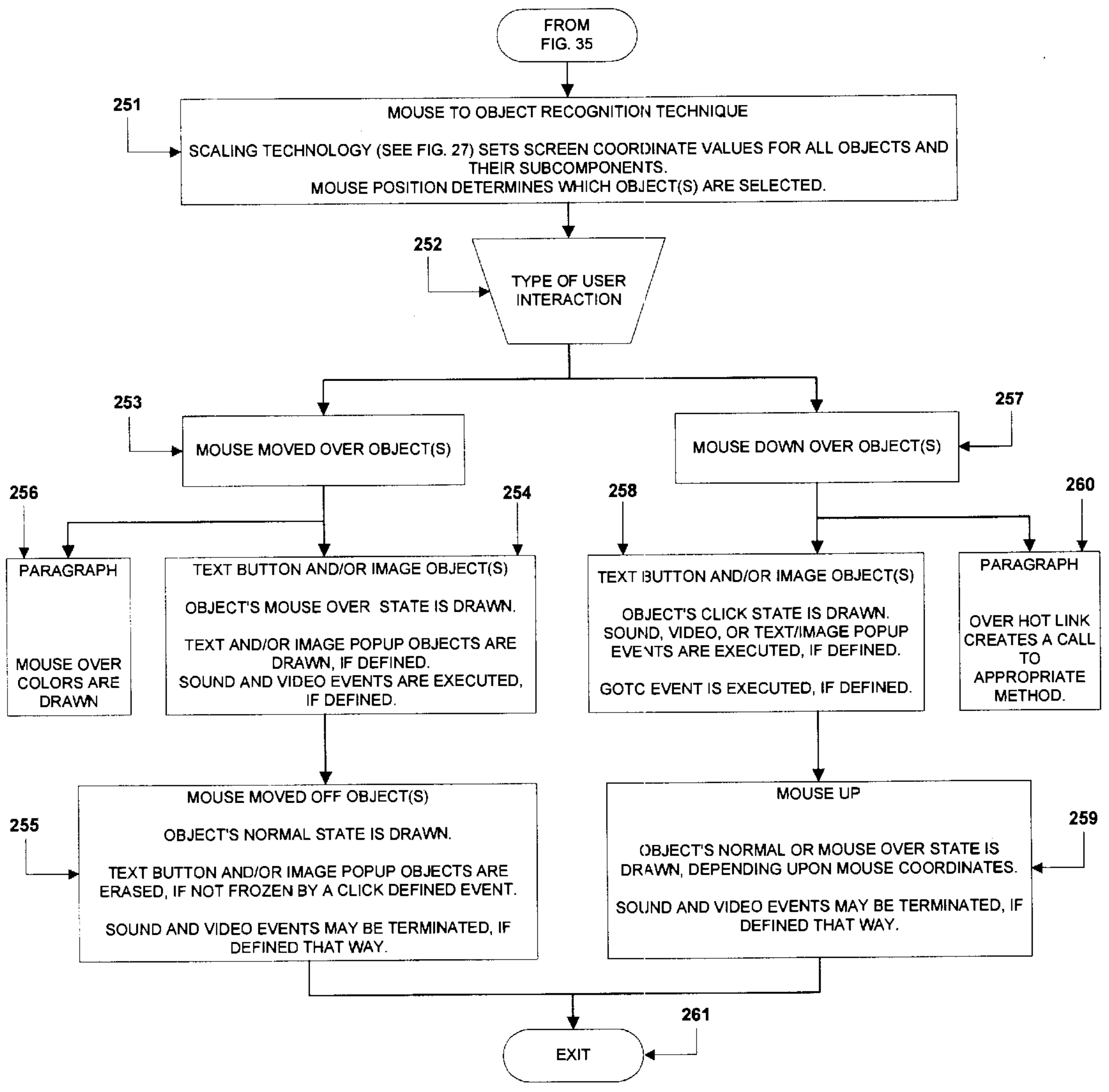
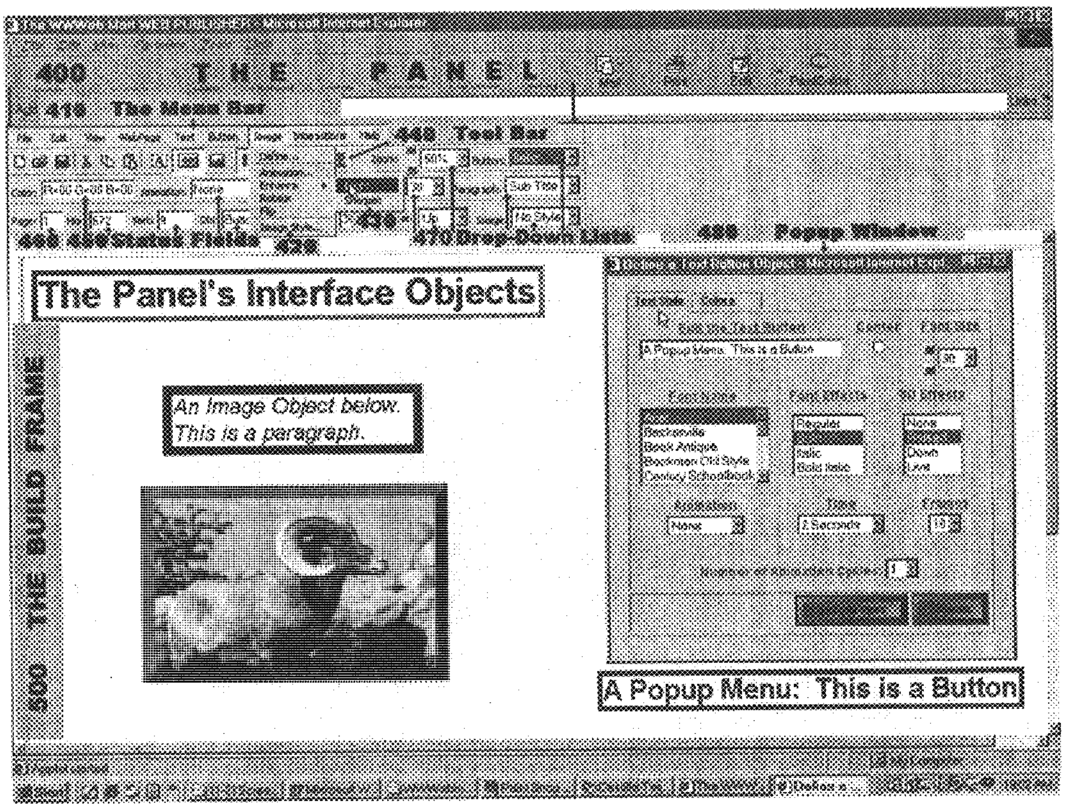

FROM FIG. 35
251
MOUSE TO OBJECT RECOGNITION TECHNIQUE
SCALING TECHNOLOGY (SEE FIG. 27) SETS SCREEN COORDINATE VALUES FOR ALL OBJECTS AND THEIR SUBCOMPONENTS.
MOUSE POSITION DETERMINES WHICH OBJECT(S) ARE SELECTED.
252
TYPE OF USER INTERACTION
253
MOUSE MOVED OVER OBJECT(S)
257
MOUSE DOWN OVER OBJECT(S)
256
254
258
260
PARAGRAPH
MOUSE OVER COLORS ARE DRAWN
TEXT BUTTON AND/OR IMAGE OBJECT(S)
OBJECT'S MOUSE OVER STATE IS DRAWN.
TEXT AND/OR IMAGE POPUP OBJECTS ARE DRAWN, IF DEFINED.
SOUND AND VIDEO EVENTS ARE EXECUTED, IF DEFINED.
TEXT BUTTON AND/OR IMAGE OBJECT(S)
OBJECT'S CLICK STATE IS DRAWN.
SOUND, VIDEO, OR TEXT/IMAGE POPUP EVENTS ARE EXECUTED, IF DEFINED.
GOTO EVENT IS EXECUTED, IF DEFINED.
PARAGRAPH
OVER HOT LINK CREATES A CALL TO APPROPRIATE METHOD.
255
MOUSE MOVED OFF OBJECT(S)
OBJECT'S NORMAL STATE IS DRAWN.
TEXT BUTTON AND/OR IMAGE POPUP OBJECTS ARE ERASED, IF NOT FROZEN BY A CLICK DEFINED EVENT.
SOUND AND VIDEO EVENTS MAY BE TERMINATED, IF DEFINED THAT WAY.
MOUSE UP
259
OBJECT'S NORMAL OR MOUSE OVER STATE IS DRAWN, DEPENDING UPON MOUSE COORDINATES.
SOUND AND VIDEO EVENTS MAY BE TERMINATED, IF DEFINED THAT WAY.
261
EXIT

Fig. 36

RESPOND TO USER INTERACTIONS



400 THE PANEL
410 The Menu Bar
440 Tool Bar
460 480 Status Fields
420
430
470 Drop-Down Lists
490 Popup Window
The Panel's Interface Objects
An Image Object below.
This is a paragraph.
500 THE BUILD FRAME
A Popup Menu: This is a Button

Fig. 37

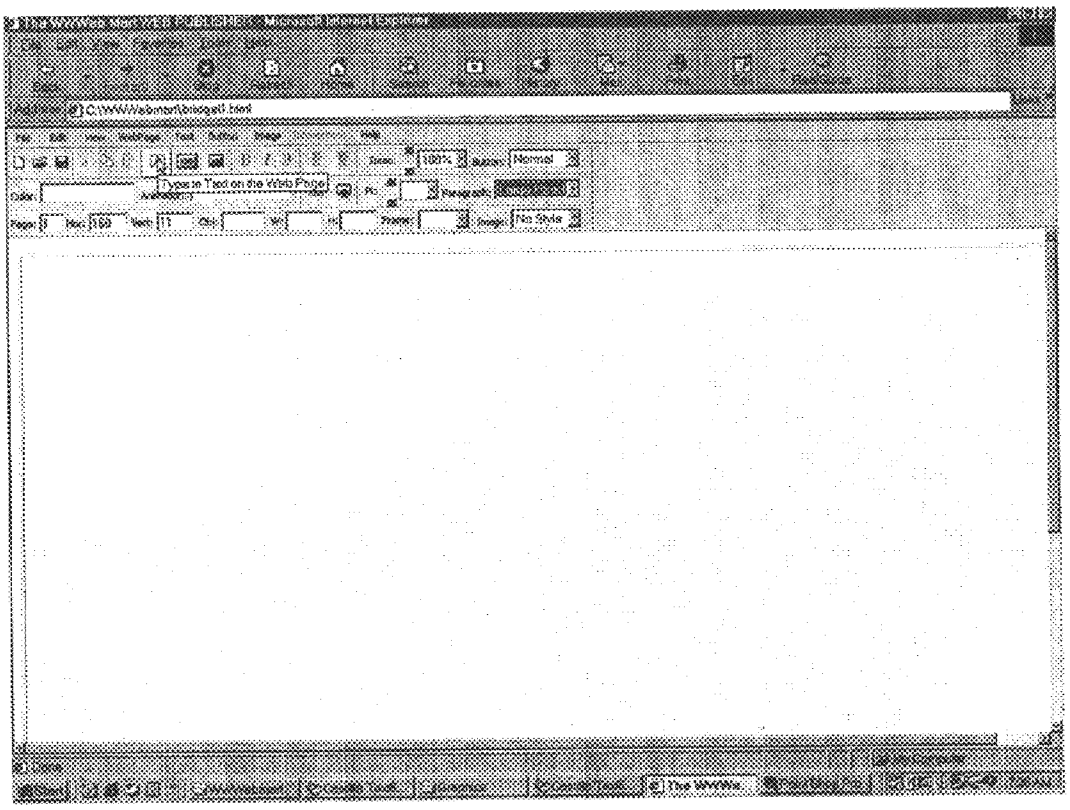

Address C:\WWWebmstr\bridge0.html
Types in Text on the Web Page
100%
Normal
No Style
Done

Fig. 38

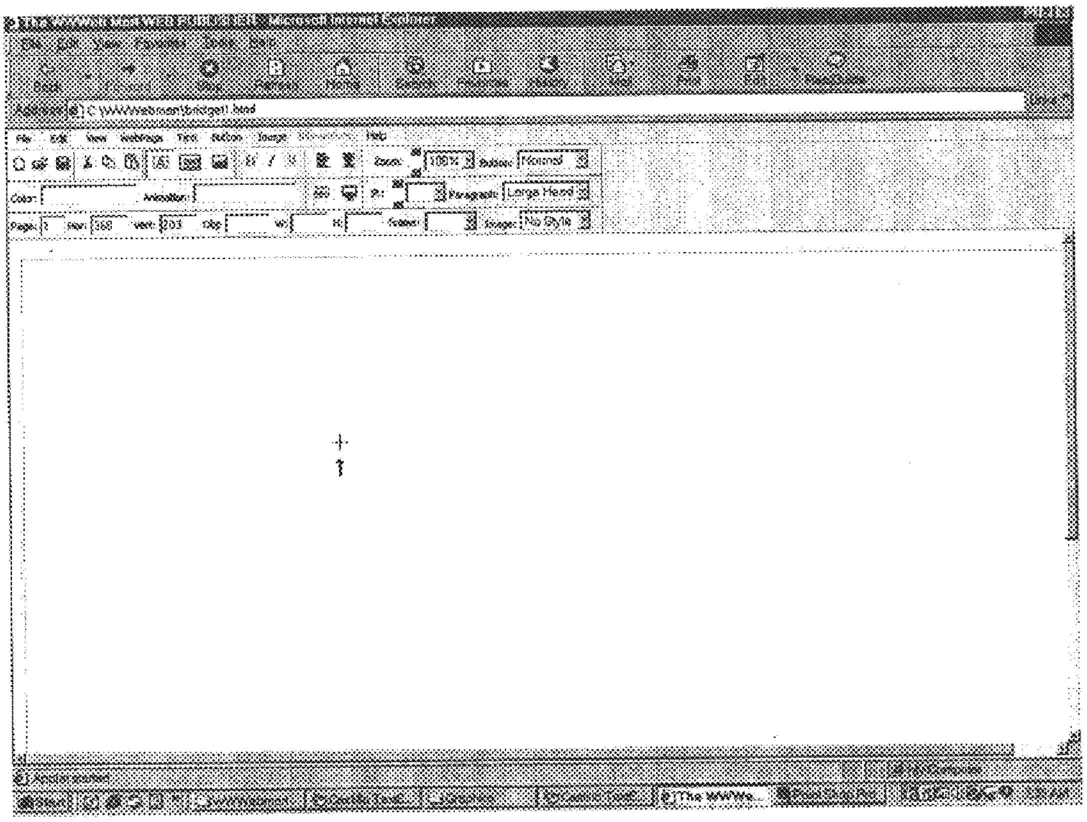

Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Address C:\WWWebman\bridge11.html
File Edit View WebPage Text Button Image Help
Zoom: 100%
Button: Normal
Color:
Animation:
Paragraph: Large Head
Page: 1 Horz: 368 Vert: 203
Image: No Style
Start

Fig. 39

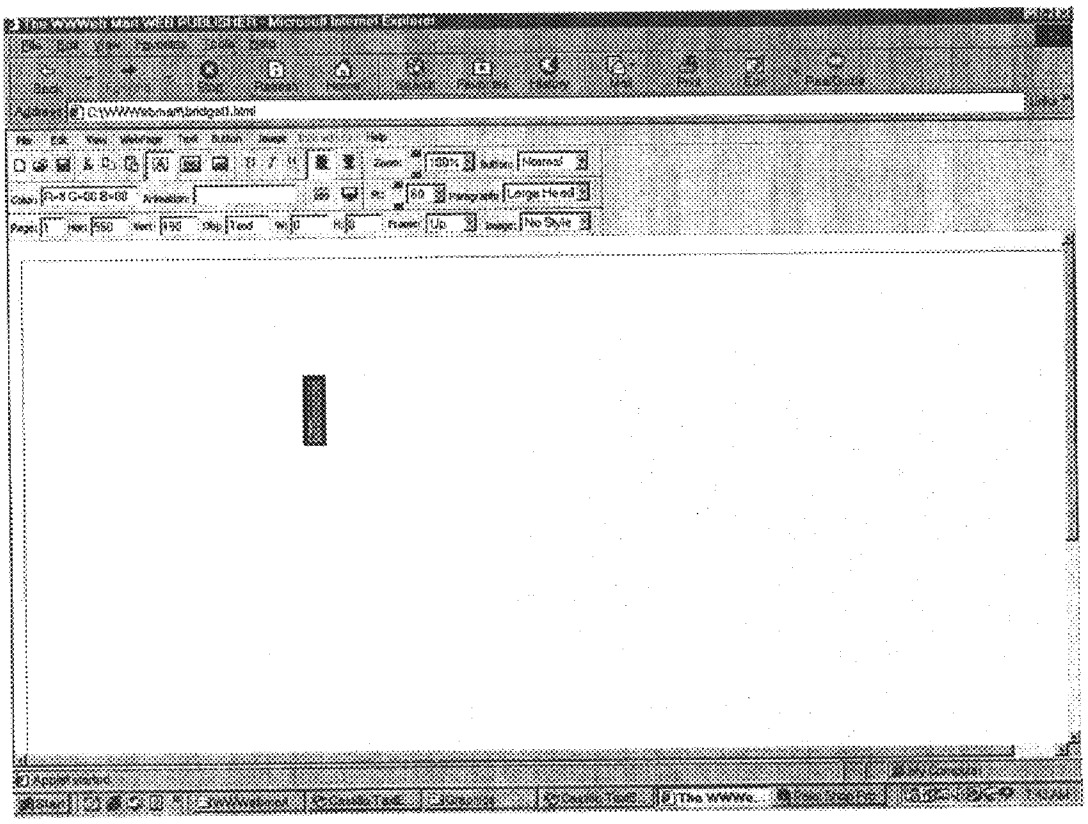

C:\WWWebmaster\bridge1.html
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Zoom: 100%
Button: Normal
Paragraph: Large Head
Frame: Up
Image: No Style
Applet started.
Start

Fig. 40

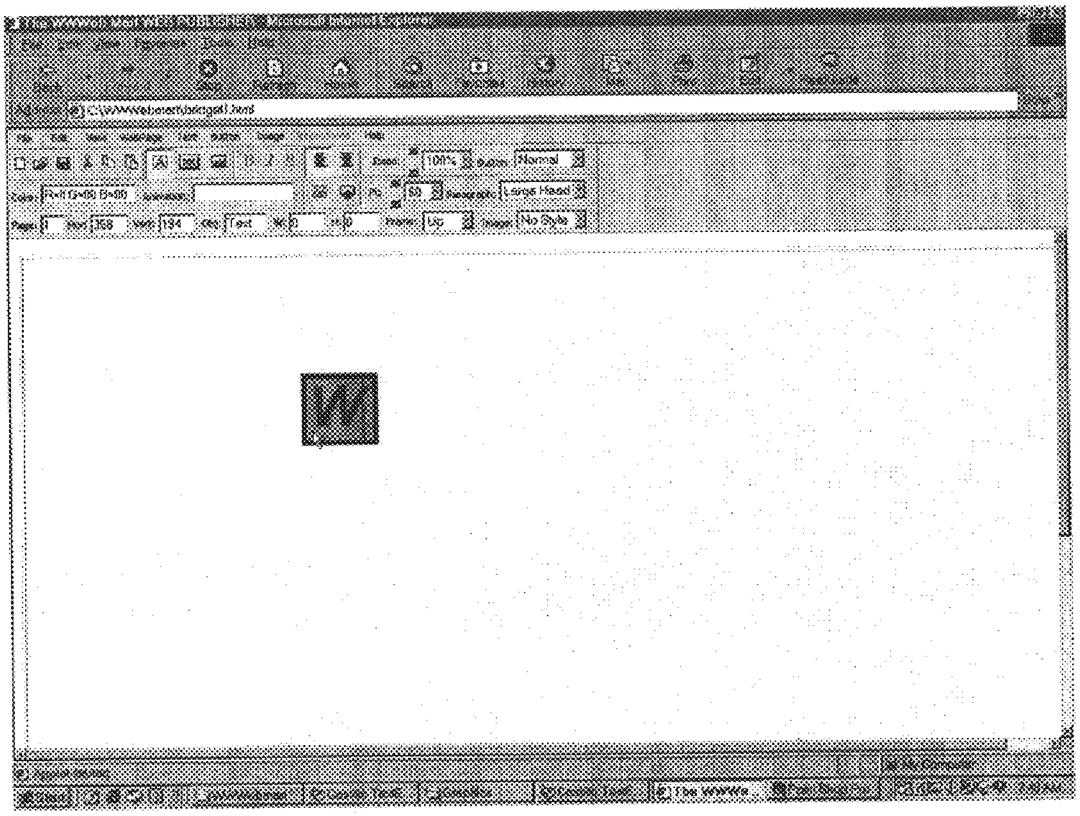

C:\WWWebelert\bridge41.html
100%
Normal
Large Head
Text
Up
No Style
Start

Fig. 41

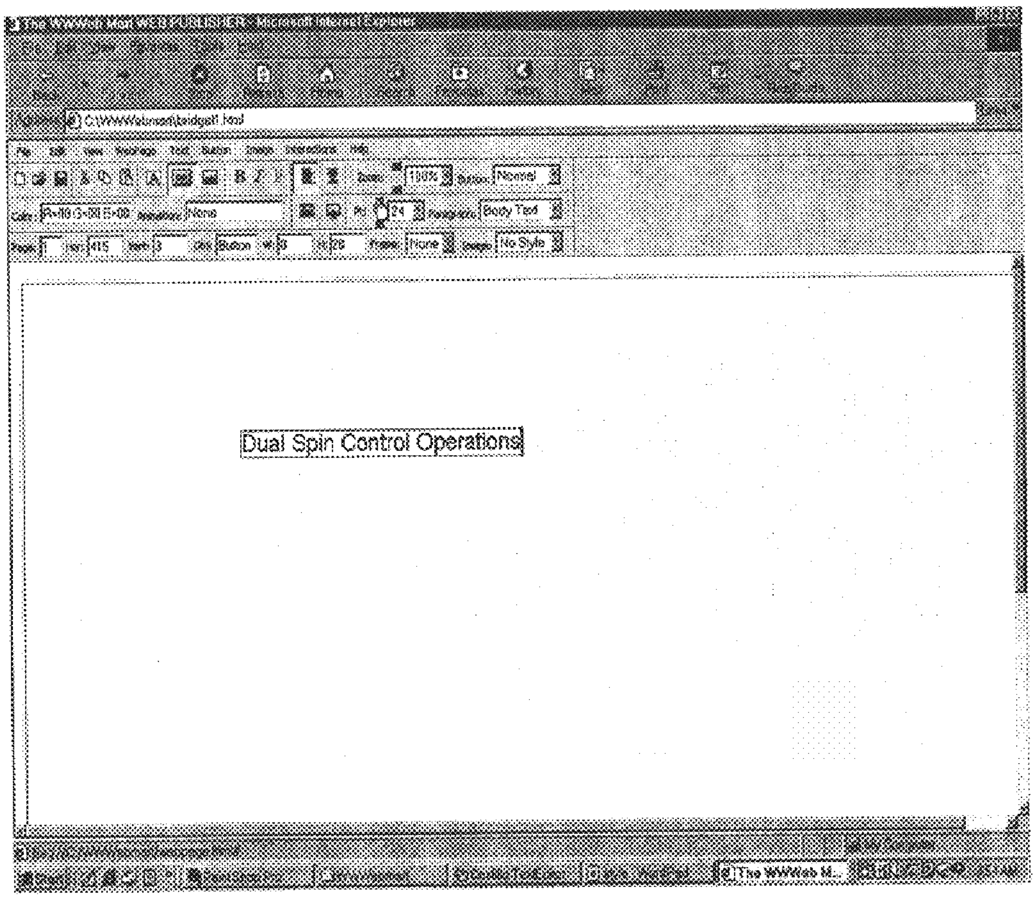

The WWWeb Host WEB PUBLISHER - Microsoft Internet Explorer
C:\WWWebmart\bridge1.html
File Edit View Webpage Text Button Image Interactions Help
100%
Normal
Body Text
None
No Style
Button
Dual Spin Control Operations
Start

Fig. 42

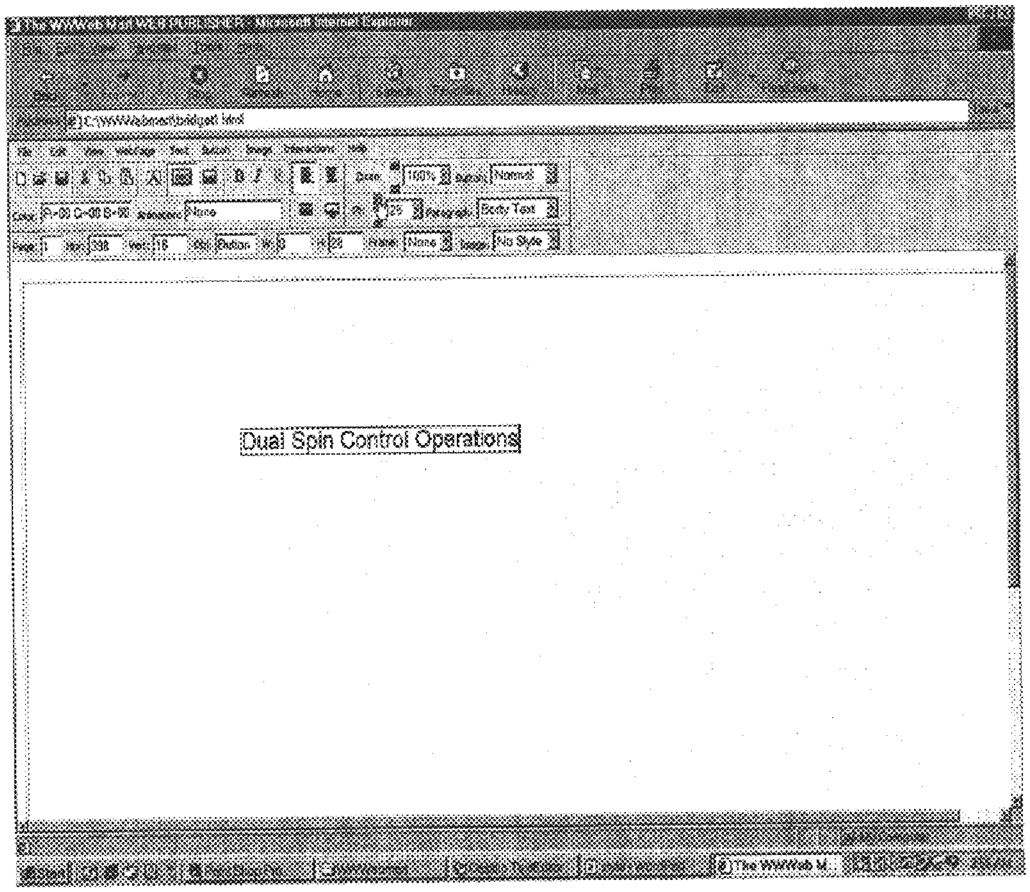


Fig. 43

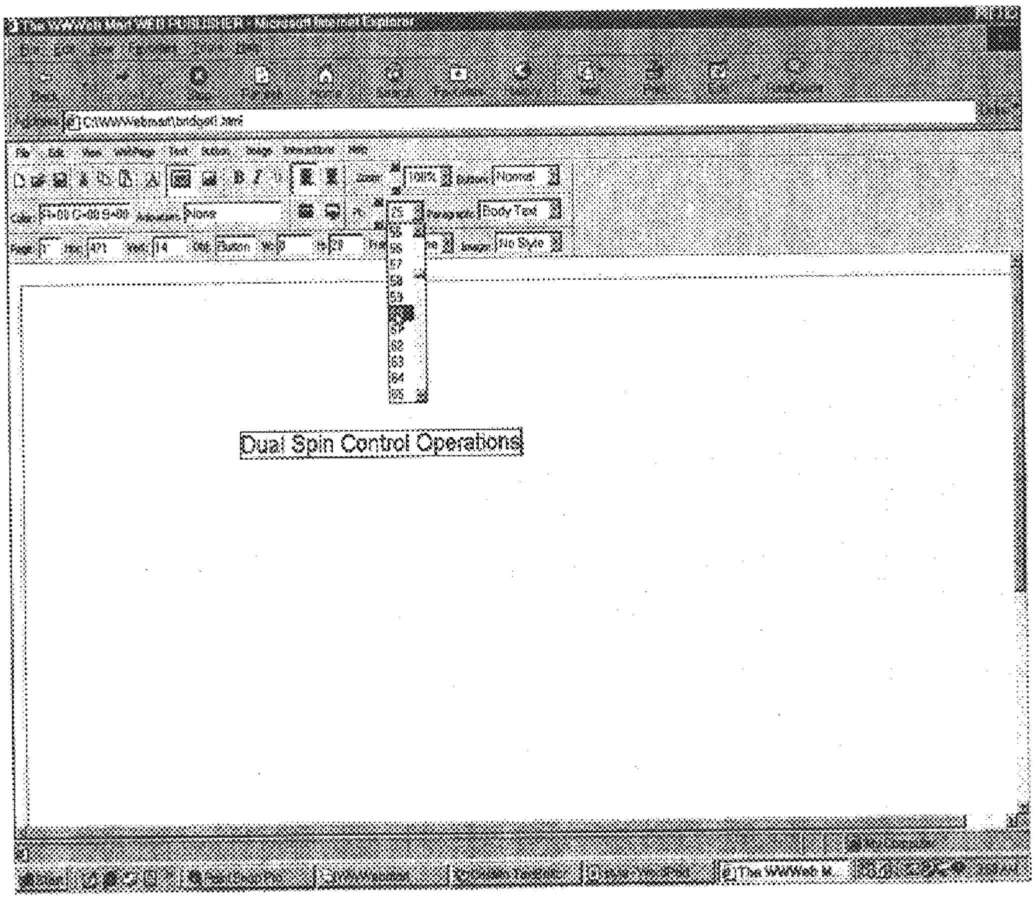

The WWWeb Man WEB PUBLISHER - Microsoft Internet Explorer
C:\WWWebman\bridge40.html
Zoom 100%
Button Normal
Paragraph Body Text
Adjustment None
Image No Style
Button
55
56
57
58
59
61
62
63
64
65
Dual Spin Control Operations

Fig. 44

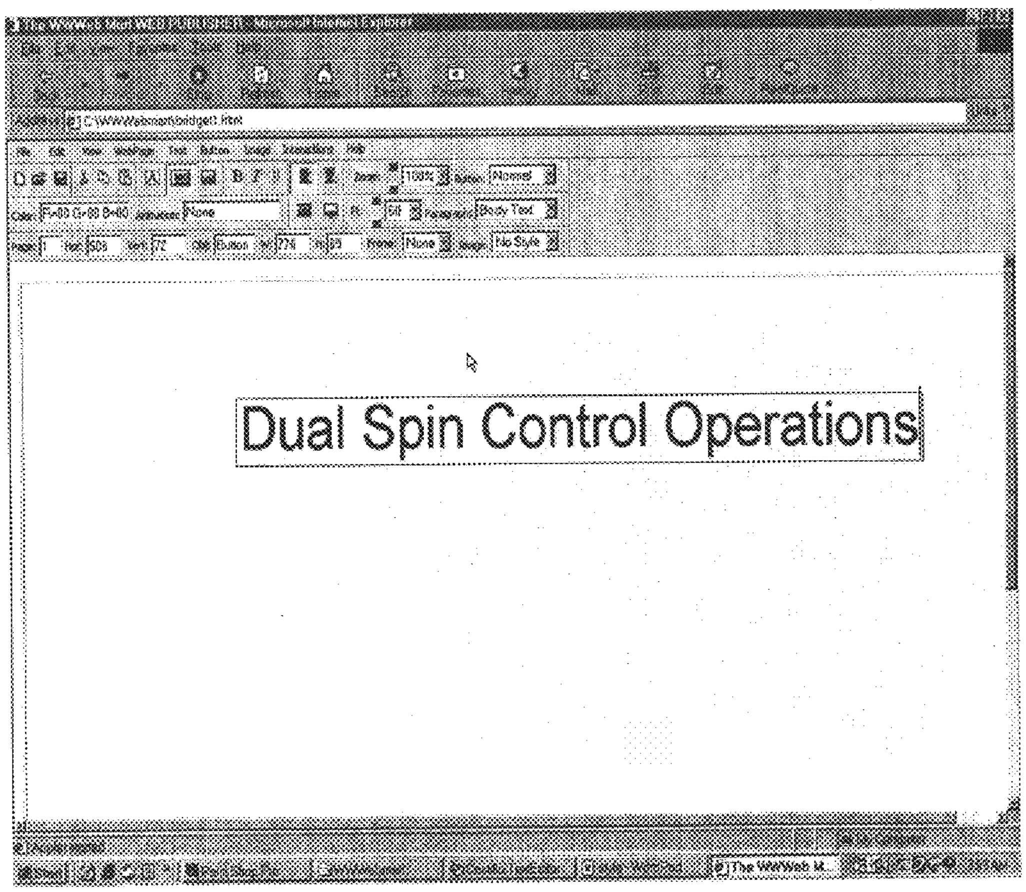

The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
C:\WWWebmart\bridget1.html
Dual Spin Control Operations

Fig. 45

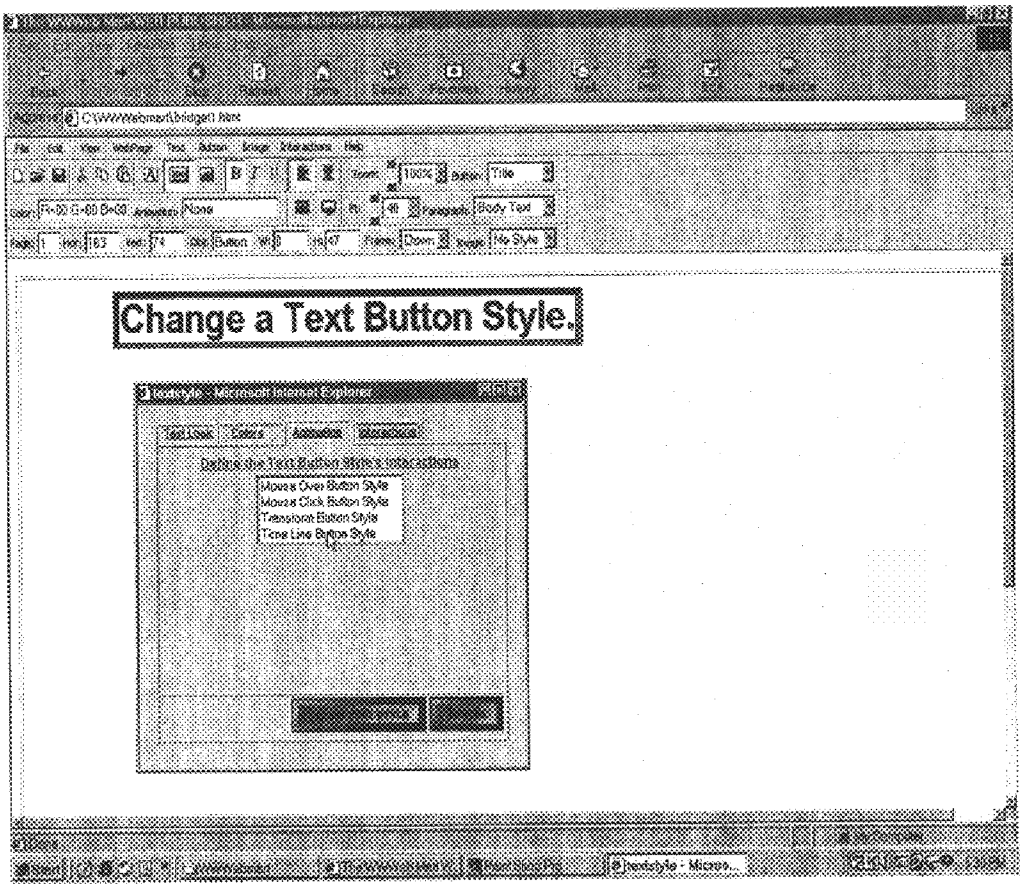

C:\WWWebmart\bridget1.html
Change a Text Button Style.
textstyle - Microsoft Internet Explorer
Text Look
Colors
Activation
Interactions
Define the Text Button Style's Interactions
Mouse Over Button Style
Mouse Click Button Style
Transform Button Style
Time Line Button Style
None
Body Text
Button
Down
No Style
Title
100%
Start
textstyle - Micros...

Fig. 46

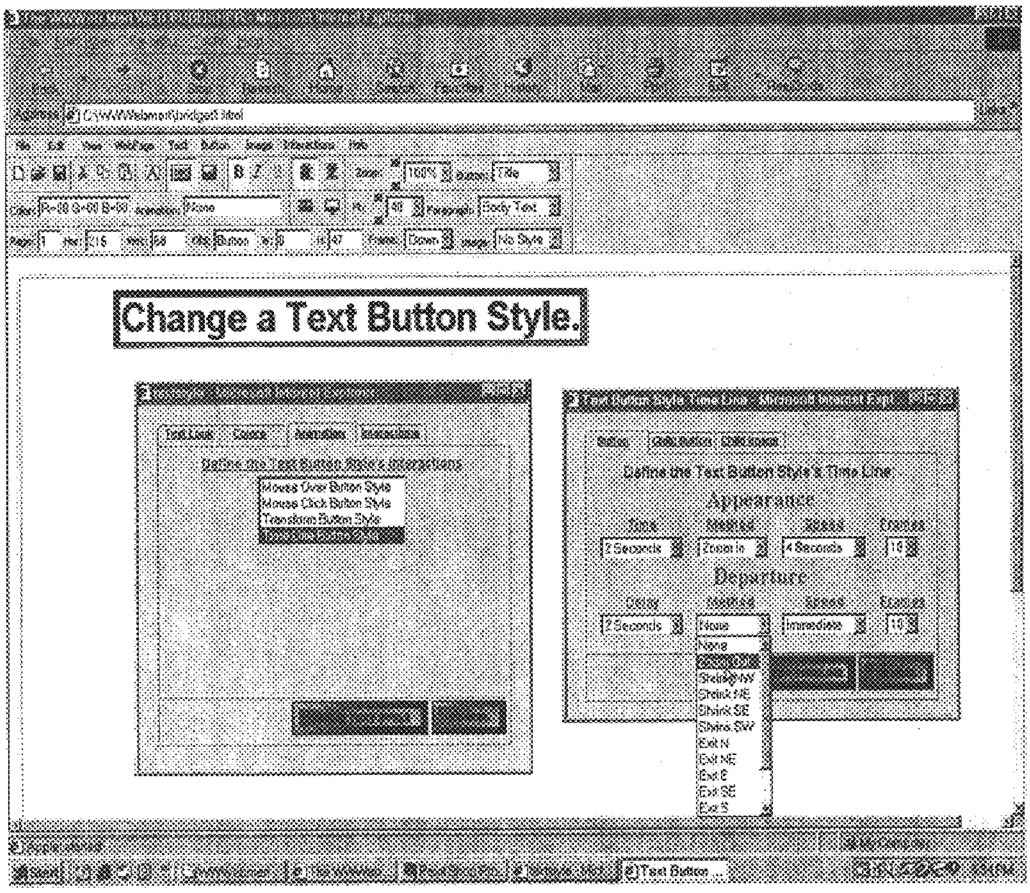

Address C:\WWWebmart\bridget1.html
Change a Text Button Style.
Text Look
Colors
Animation
Interactions
Define the Text Button Style's Interactions
Mouse Over Button Style
Mouse Click Button Style
Transform Button Style
Text Button Style Time Line - Microsoft Internet Expl
Define the Text Button Style's Time Line
Appearance
Time
Method
Speed
Frames
2 Seconds
Zoom In
4 Seconds
10
Departure
Delay
Method
Speed
Frames
2 Seconds
None
Immediate
10
None
Shrink NW
Shrink NE
Shrink SE
Shrink SW
Exit N
Exit NE
Exit E
Exit SE
Exit S
Start
Text Button ...

Fig. 47

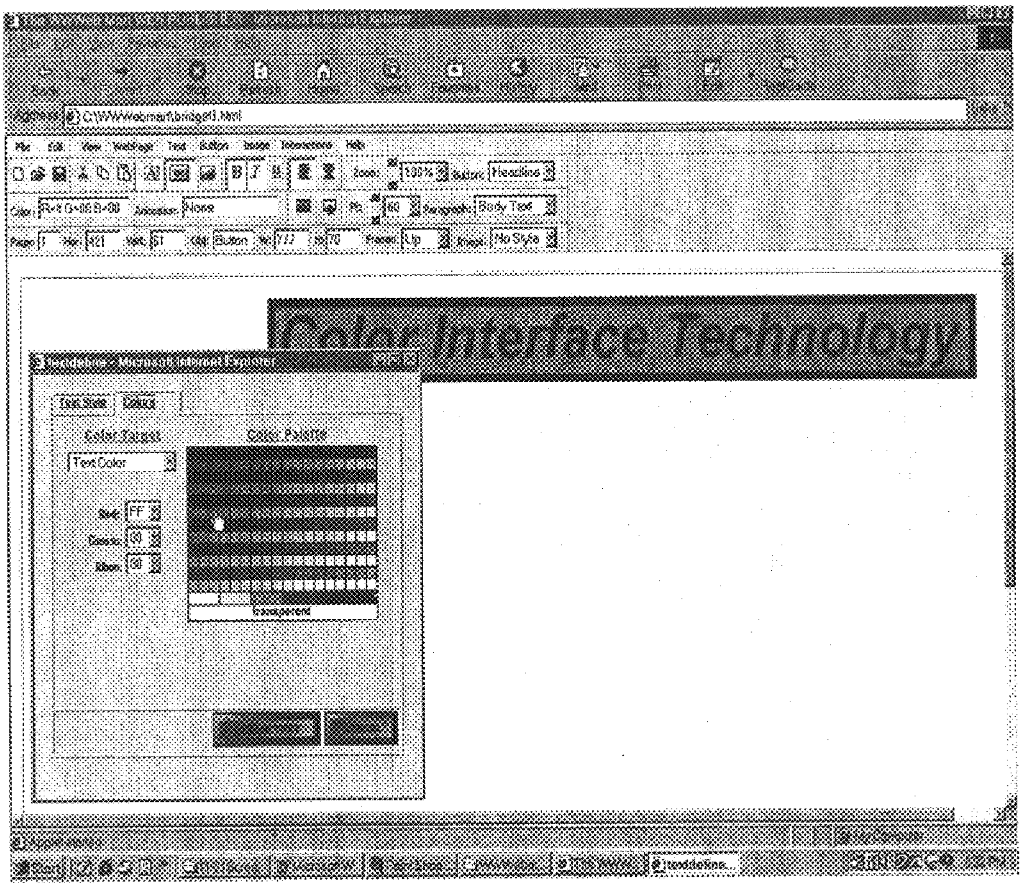
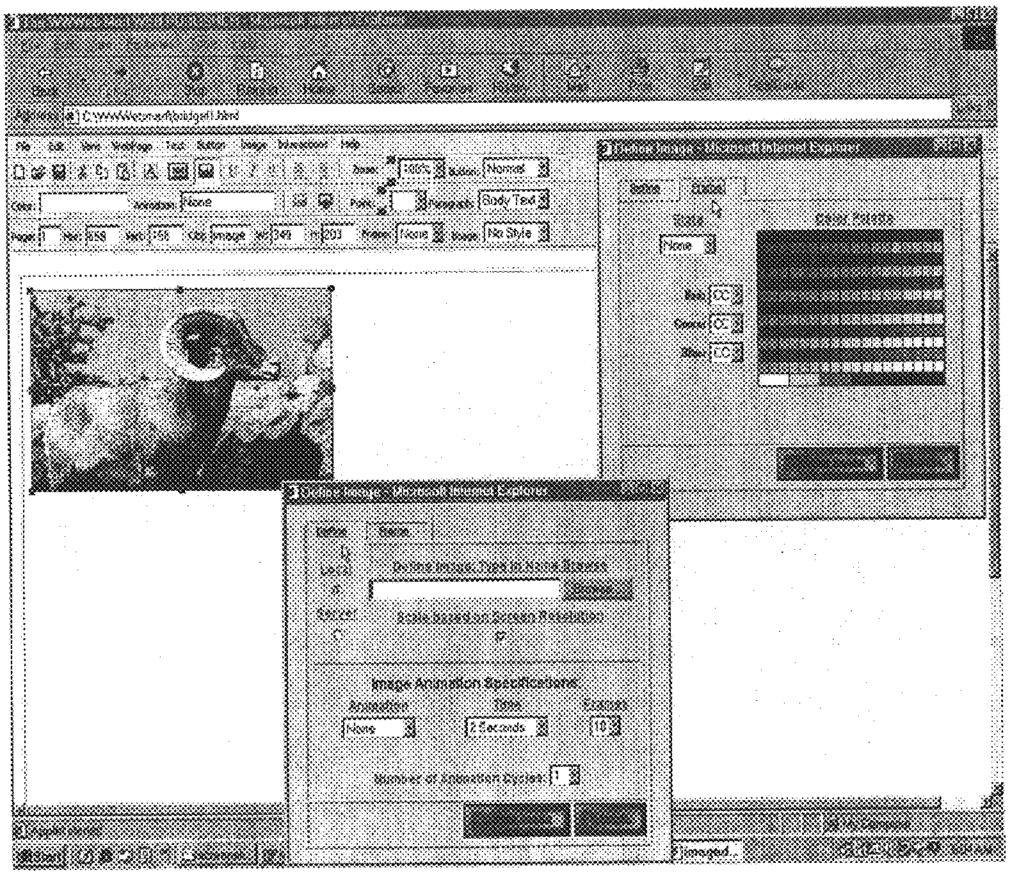

C:\WWWebmart\bridge43.html
File Edit View WebPage Text Button Image Interactions Help
Zoom: 100%
Button: Headline
Color: R=f G=80 B=80
Animation: None
Pt: 60
Paragraph: Body Text
Page: 1
Hor: 421
Vert: 61
Obj: Button
W: 777
H: 70
Preset: Up
Image: No Style
Color Interface Technology
Text Color
Color Target
Color Palette
Red: FF
Green: 80
Blue: 80
transparent

Fig. 48



File Edit View WebPage Text Button Image Interactivity Help
Zoom: 100%
Button: Normal
Animation: None
Paragraph: Body Text
Obj: Image
Frames: None
Image: No Style
Define Image - Microsoft Internet Explorer
Define
Status
Color Palette
None
Define Image - Microsoft Internet Explorer
Define Image: Type in Name Browse
Browse
Scale based on Screen Resolution
Image Animation Specifications
Animation
Time
Frames
None
2 Seconds
10
Number of Animation Cycles: 1

Fig. 49

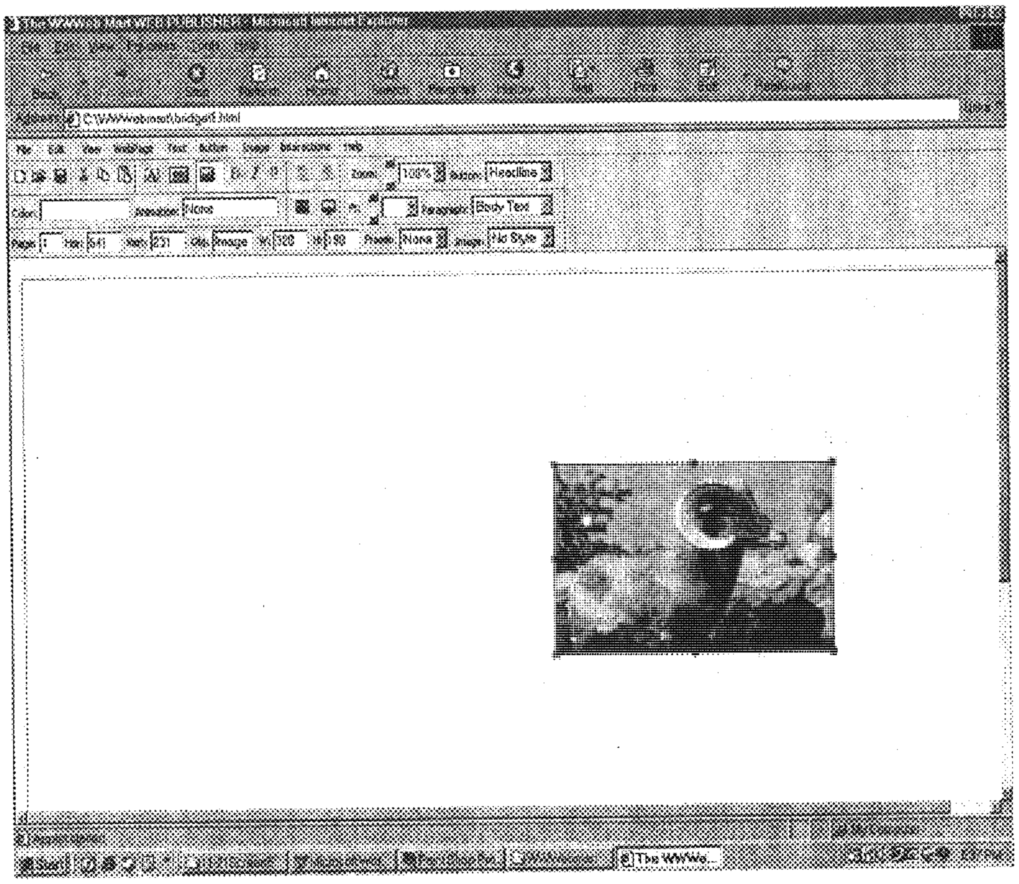

Zoom: 100%
Button: Headline
Attribute: None
Paragraph: Body Text
Image
None
No Style
Start

Fig. 50

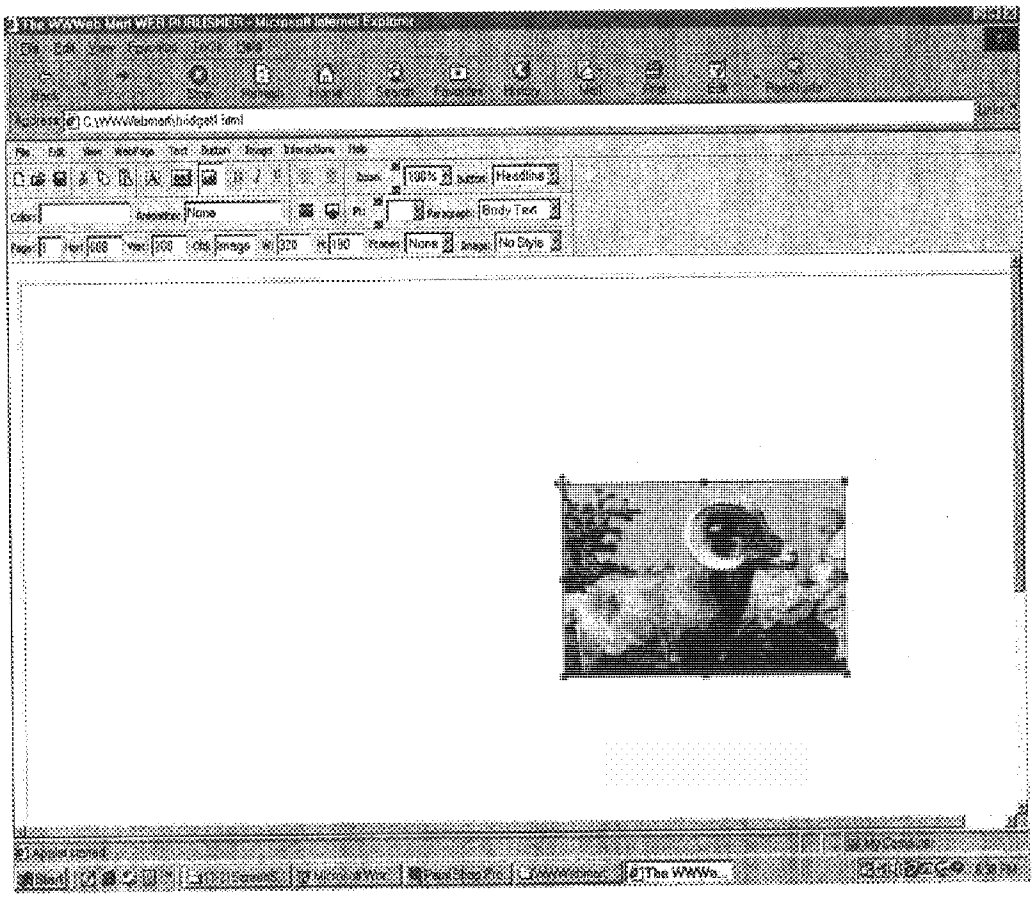

The WWWeb Man WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Address C:\WWWebman\bridget.html
File Edit View WebPage Text Button Image Interactions Help
Zoom: 100% Button: Headline
Color: Animation: None Paragraph: Body Text
Page: 1 Hor: 388 Ver: 300 Obj: Image W: 320 H: 190 Frame: None Image: No Style
Applet started
My Computer
Start
The WWWe...

Fig. 51

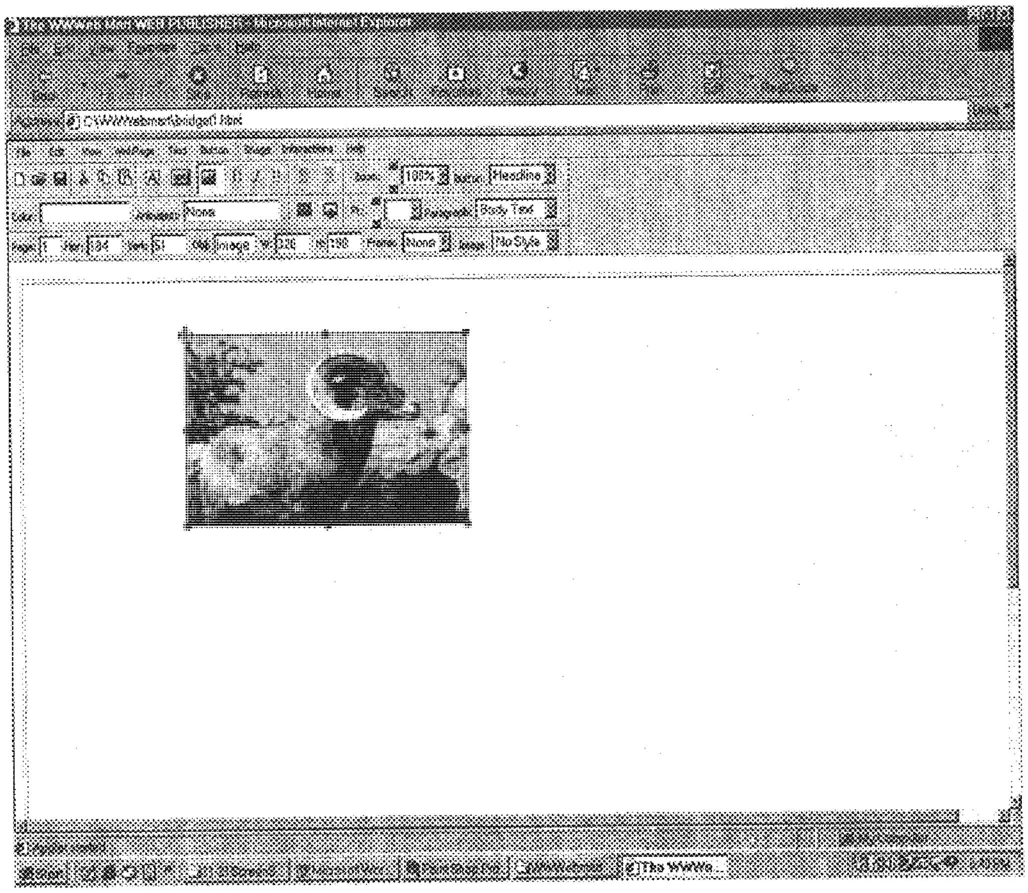

The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
C:\WWWebmart\bridget\.html
100%
Headline
None
Body Text
Image
None
No Style
The WWWe...

Fig. 52

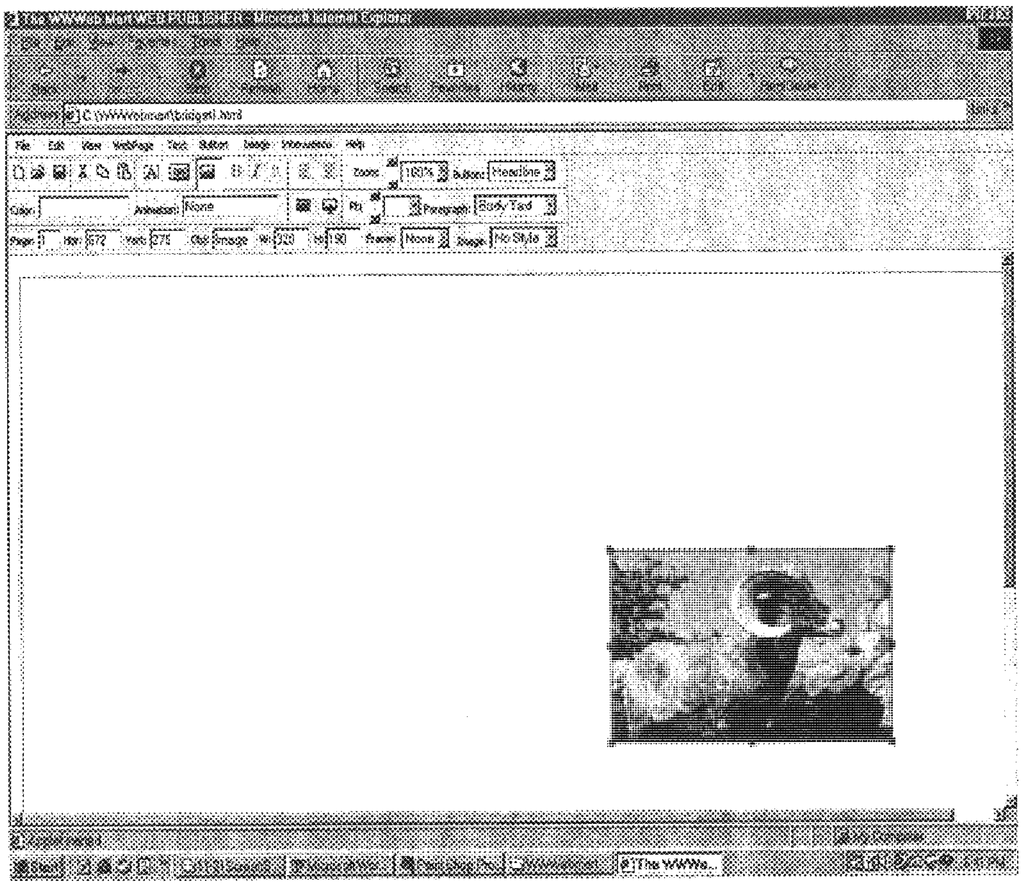
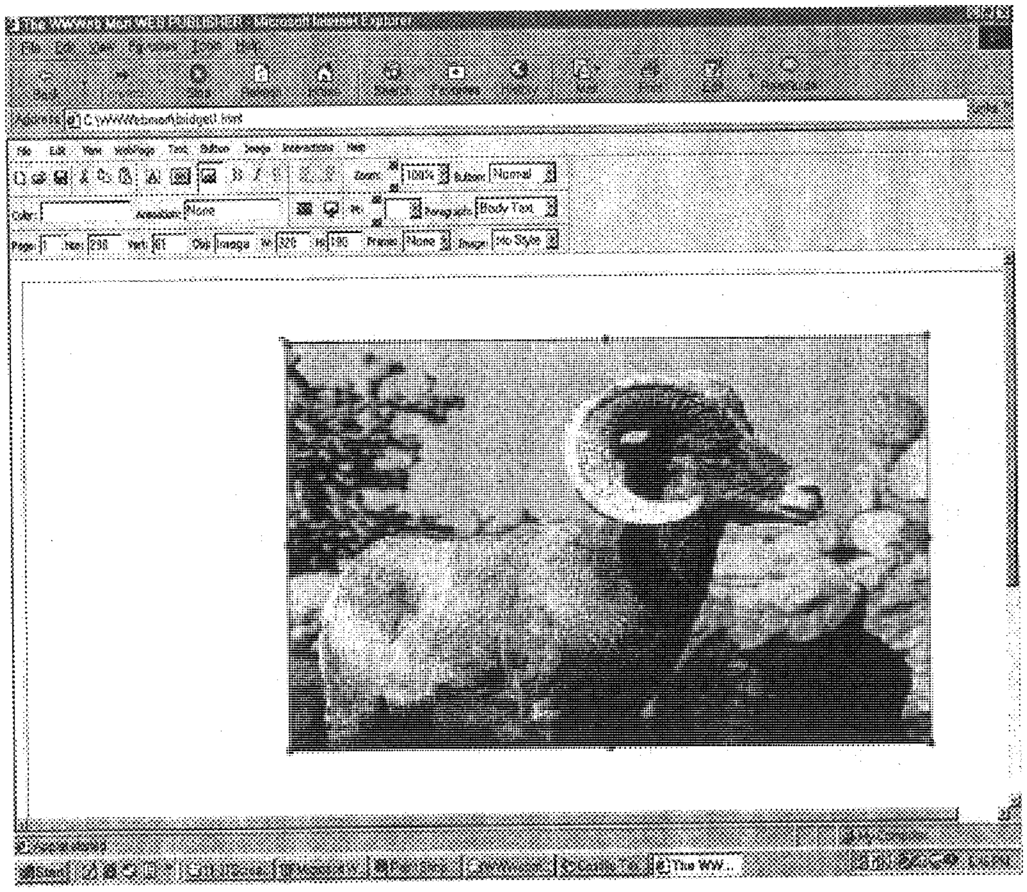

The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
Address C:\WWWebmart\bridget1.html
File Edit View WebPage Text Button Image Miscellaneous Help
Zoom 100%
Button Headline
Color
Animation None
Paragraph Body Text
Page 1
Hor 572
Vert 275
Obj Image
W 320
Ht 190
Frame None
Image No Style
Start
The WWWe...

Fig. 53

Fig. 54


File Edit View WebPage Text Button Image Interactivity Help
Zoom: 100%
Button: Normal
Color:
Animation: None
Paragraph: Body Text
Page: 1
Horz: 298
Vert: 81
Obj: Image
Frame: None
Image: No Style
The WW...

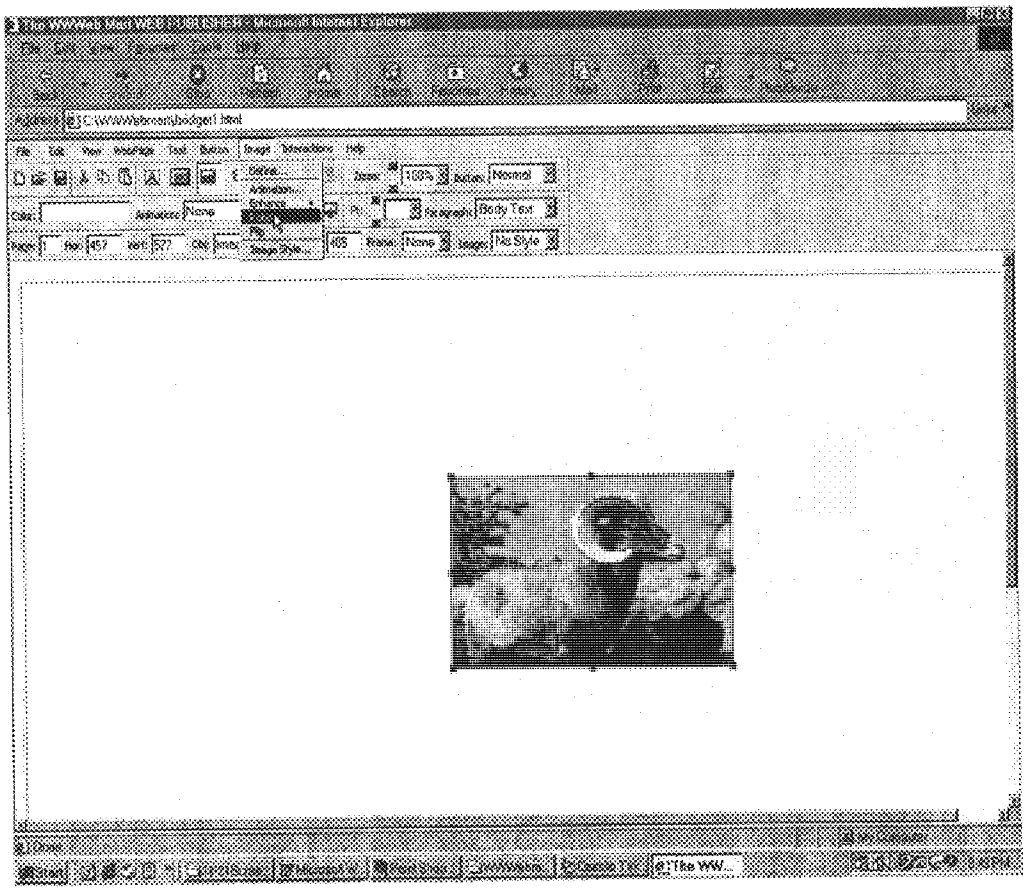

The WWWeb Med WEB PUBLISHER - Microsoft Internet Explorer
Address C:\WWWebmed\badger1.html
File Edit View WebPage Text Button Image Interactions Help
Zoom 100%
Button Normal
Animation None
Paragraph Body Text
Image No Style
Done
Start

Fig. 55

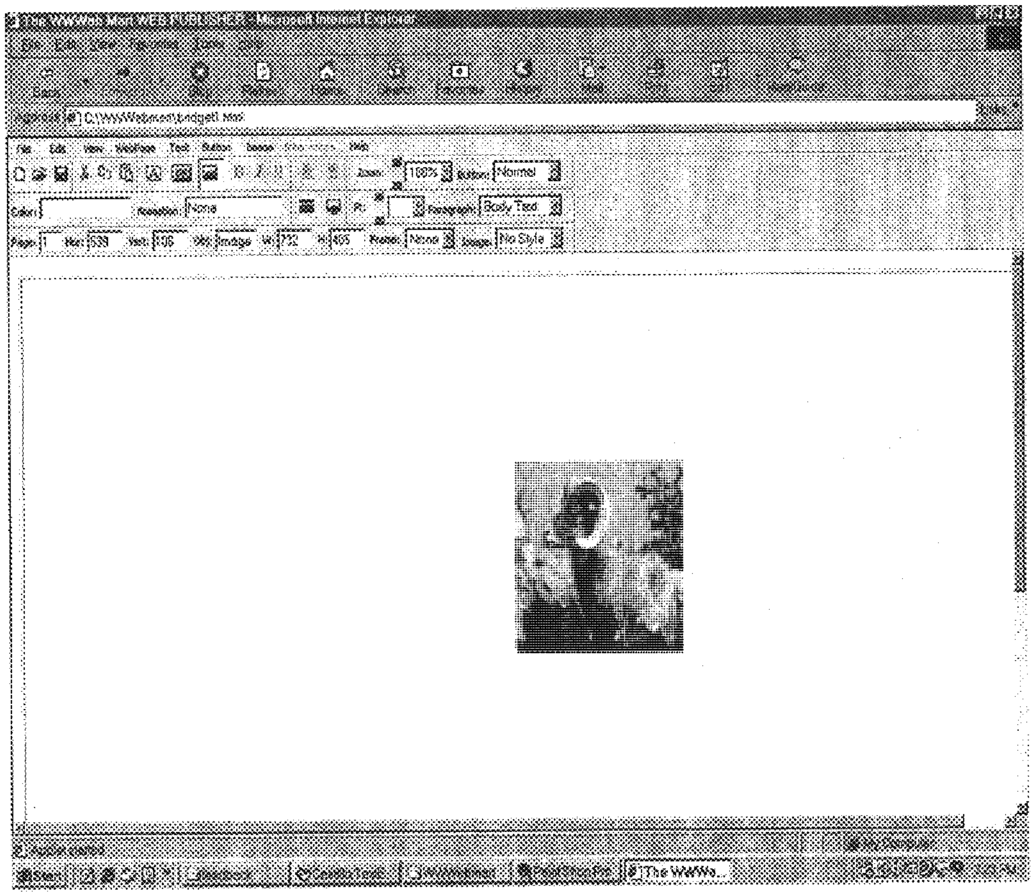

The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
C:\WWWebmart\gadget1.html
Zoom: 100%
Button: Normal
Paragraph: Body Text
Page: 1
Hor: 539
Vert: 106
Obj: Image
W: 732
H: 405
Frame: None
Image: No Style
Color:
Rotation: None
Start

Fig. 56

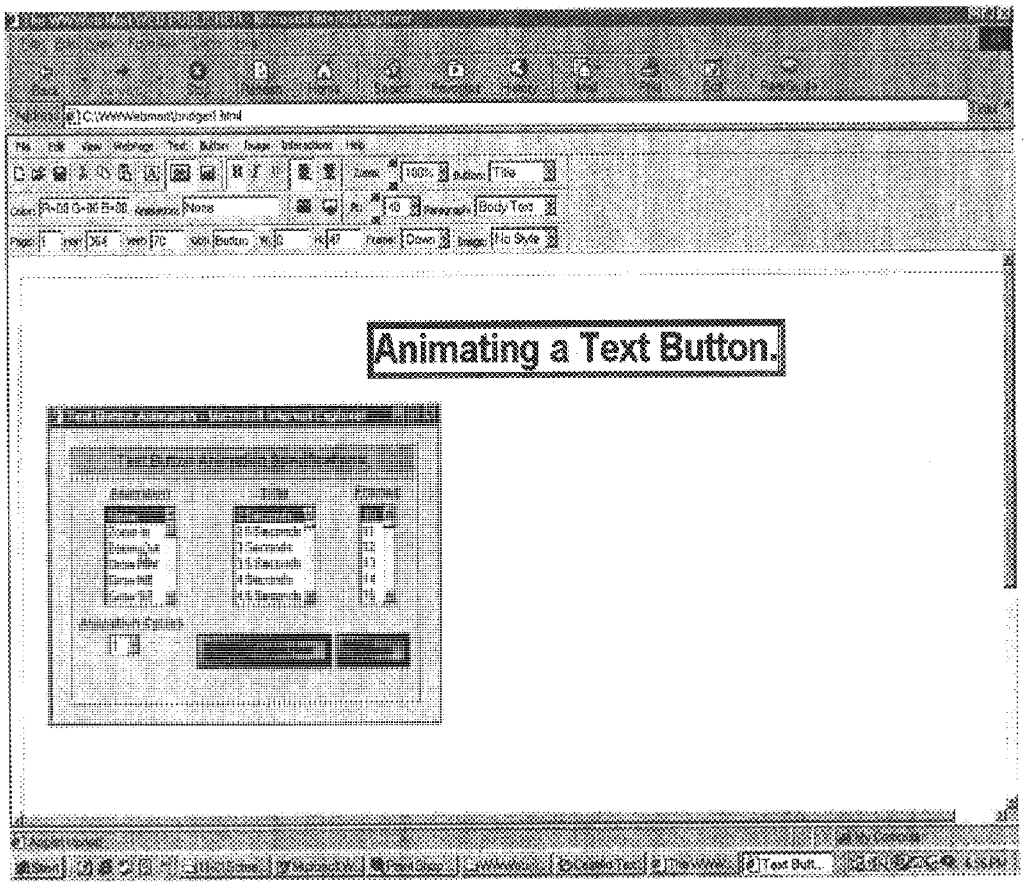
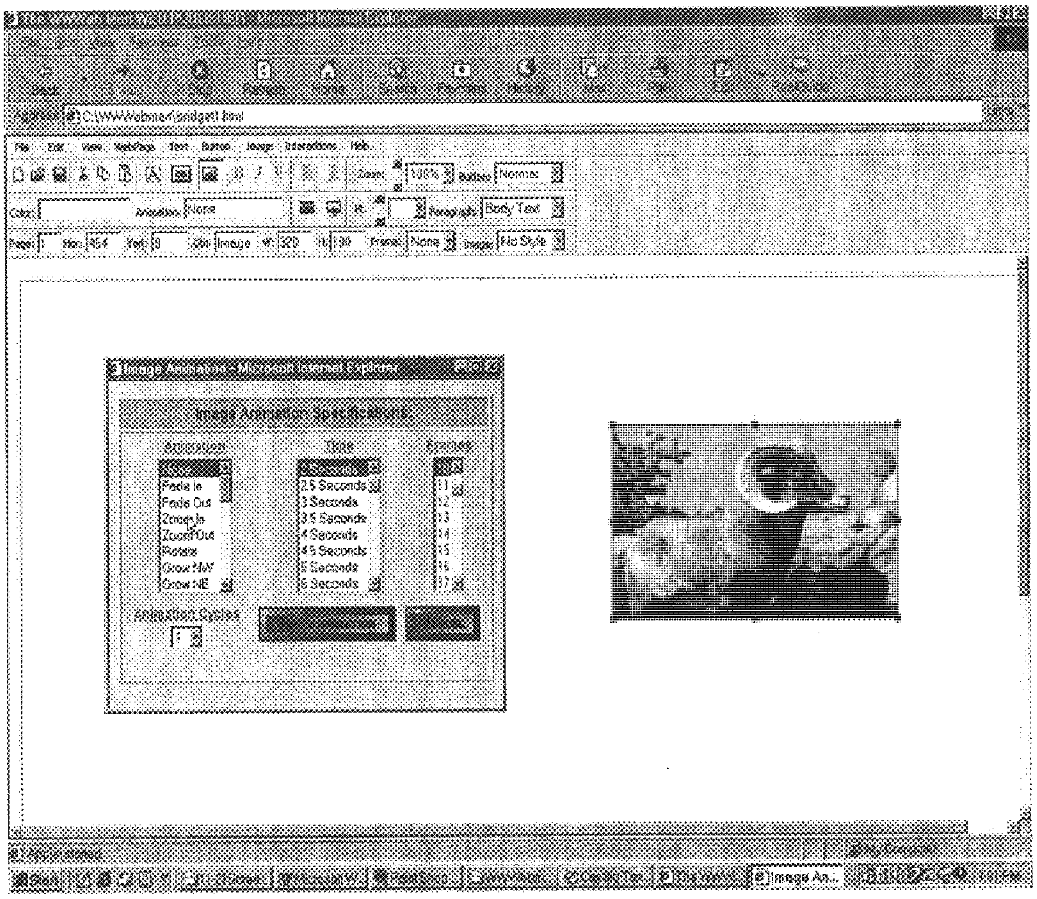

Animating a Text Button.
Text Button Animation Specifications
Animation
Title
Frames
Animation Speed

Fig. 57



Address C:\WWWebmaker\gadget1.html
File Edit View WebPage Text Button Image Interactions Help
Zoom: 100%
Button: Normal
Color:
Animation: None
Paragraph: Body Text
Page: 1
Hor: 354
Vert: 8
Obj: Image
W: 320
H: 180
Frame: None
Image: No Style
Image Animation - Microsoft Internet Explorer
Image Animation Specifications
Animation
None
Fade In
Fade Out
Zoom In
Zoom Out
Rotate
Grow NW
Grow NE
Time
2 Seconds
2.5 Seconds
3 Seconds
3.5 Seconds
4 Seconds
4.5 Seconds
5 Seconds
6 Seconds
Frames
10
11
12
13
14
15
16
17
Animation Cycles
1
Image An...

Fig. 58

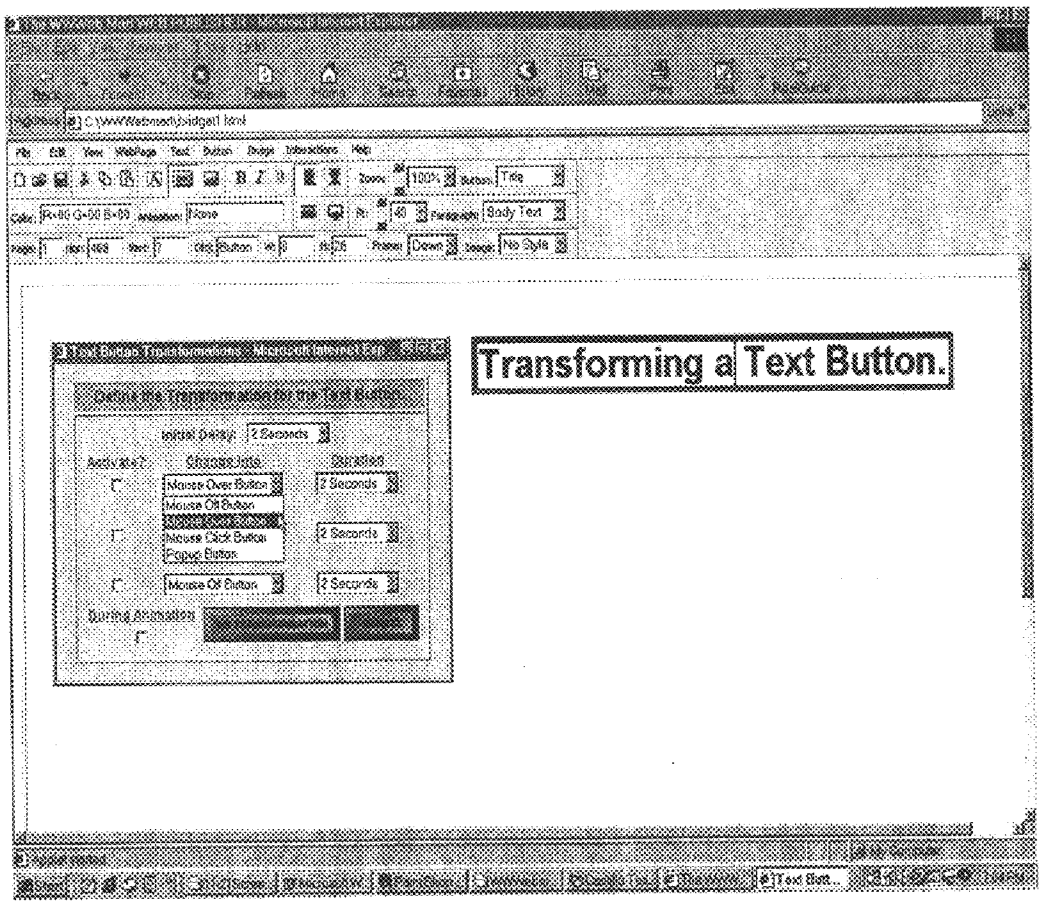

C:\WWWebmest\bidgat1.html
File Edit View WebPage Text Button Image Interactions Help
Zoom: 100%
Button: Title
Color: R=00 G=00 B=00
Animation: None
Paragraph: Body Text
Frames: Down
Image: No Style
Text Button Transformations - Microsoft Internet Exp
Define the Transformations for the Text Button
Initial Delay: 2 Seconds
Activate?
Change Into
Duration
Mouse Over Button
2 Seconds
Mouse Off Button
Mouse Click Button
Popup Button
2 Seconds
Mouse Off Button
2 Seconds
During Animation
Transforming a Text Button.

Fig. 59

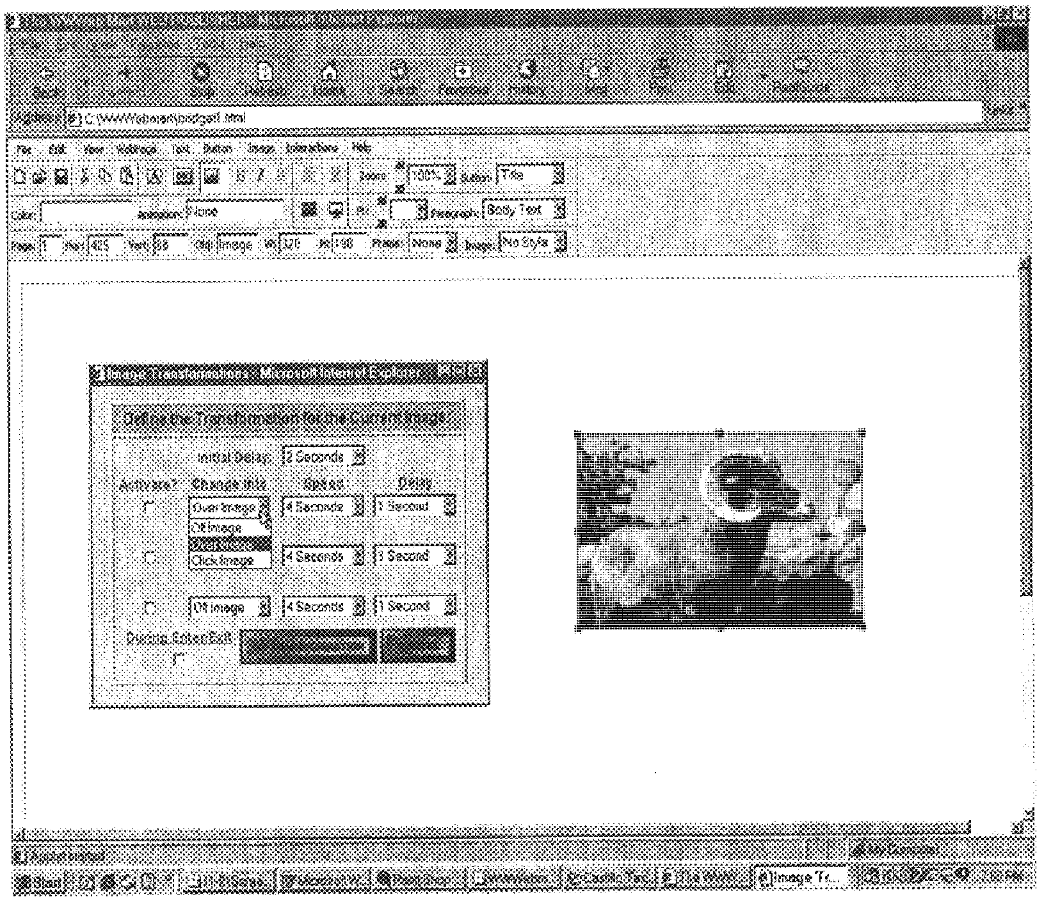
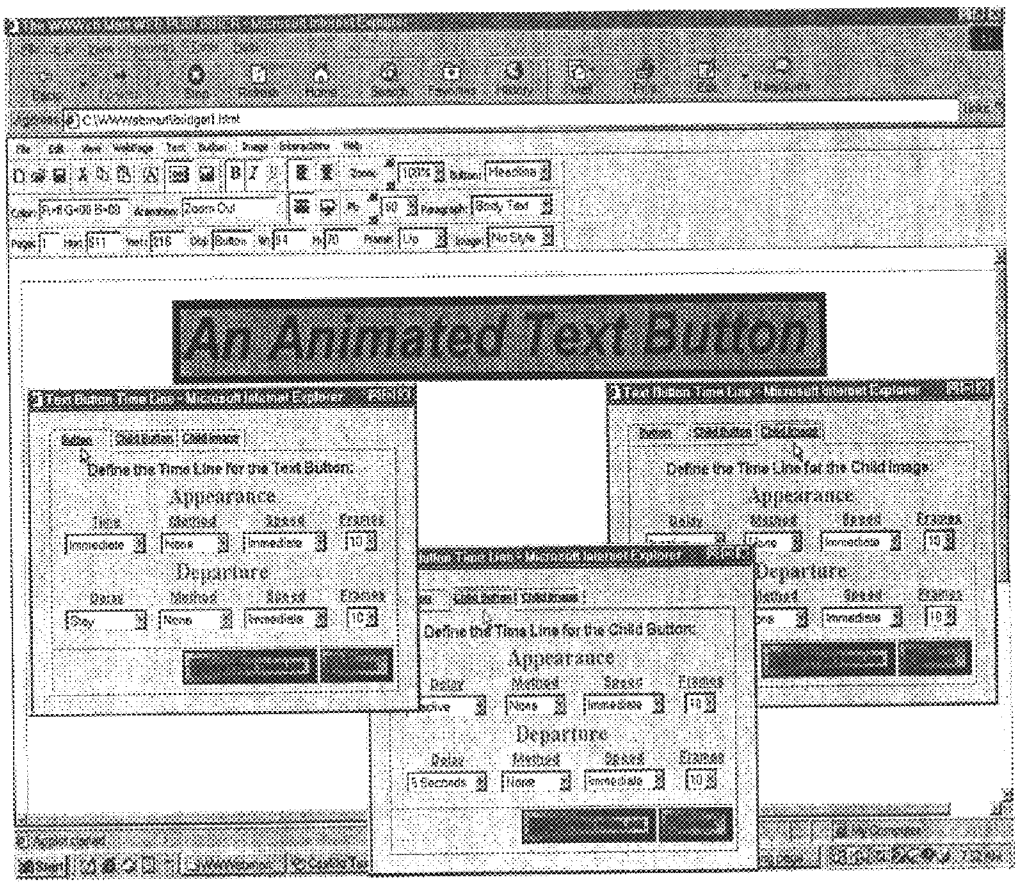

Address C:\WWWebsite\bridget1.html
File Edit View WebPage Text Button Image Interactions Help
Zoom: 100%
Button: Title
Color:
Animation: None
Paragraph: Body Text
Page: 1
Horz: 425
Vert: 66
Obj: Image
Wd: 326
Ht: 198
Frame: None
Image: No Style
Image Transformations - Microsoft Internet Explorer
Define the Transformation for the Current Image
Initial Delay: 2 Seconds
Activate?
Change Into
Speed
Delay
Over Image
4 Seconds
1 Second
Off Image
Click Image
4 Seconds
1 Second
Off Image
4 Seconds
1 Second
During Enter/Exit
Start
Image Tr...

Fig. 60



Define the Time Line for the Text Button:

Appearance

Time	Method	Speed	Frames
Immediate	None	Immediate	10

Departure

Delay	Method	Speed	Frames
Stay	None	Immediate	10

Define the Time Line for the Child Button:

Appearance

Delay	Method	Speed	Frames
[illegible]	None	Immediate	10

Departure

Delay	Method	Speed	Frames
5 Seconds	None	Immediate	10

Define the Time Line for the Child Image:

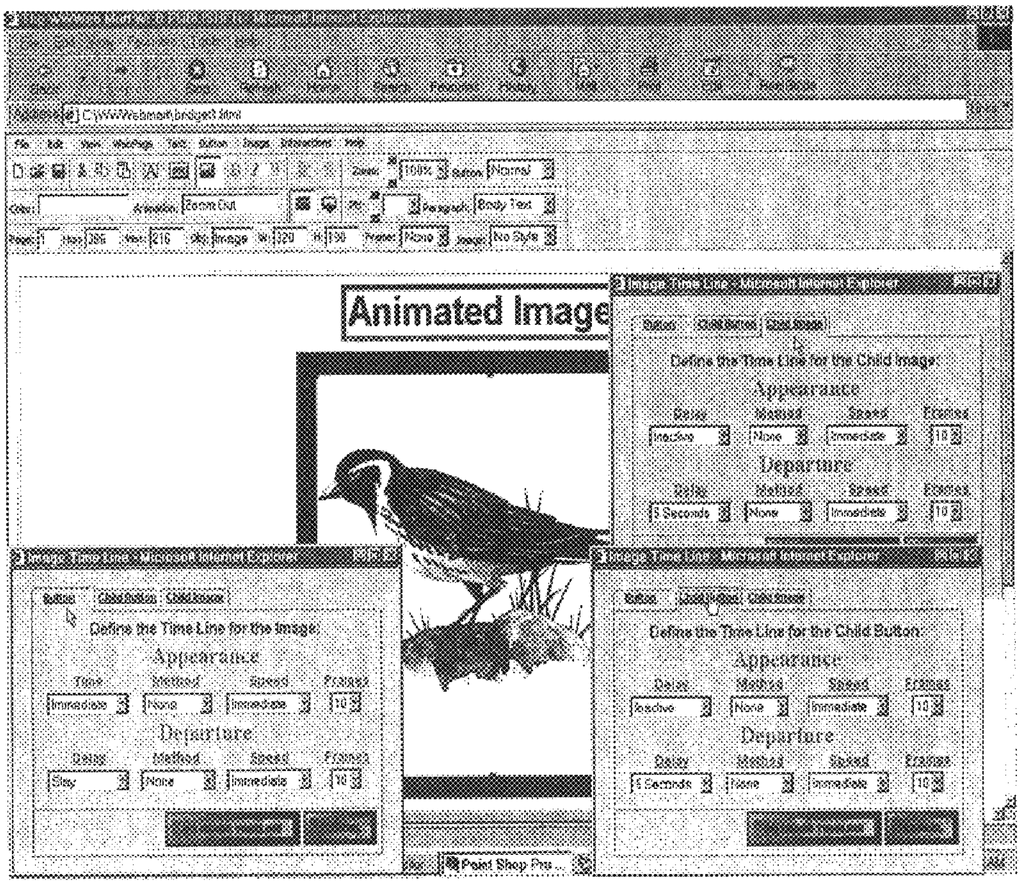
Appearance

Delay	Method	Speed	Frames
[illegible]	None	Immediate	10

Departure

Delay	Method	Speed	Frames
[illegible]	None	Immediate	10

Fig. 61



C:\WWWebmart\bridget.html

Zoom: 100% Button: Normal
Animation: Zoom Out Paragraph: Body Text
Page: 1 Hpos: 386 Vpos: 216 Obj: Image W: 320 H: 190 Frame: None Image: No Style

Image Time Line - Microsoft Internet Explorer

Button | Child Button | Child Image

Define the Time Line for the Image:

Appearance

Time	Method	Speed	Frames
Immediate	None	Immediate	10

Departure

Delay	Method	Speed	Frames
Stay	None	Immediate	10

Image Time Line - Microsoft Internet Explorer

Button | Child Button | Child Image

Define the Time Line for the Child Image:

Appearance

Delay	Method	Speed	Frames
Inactive	None	Immediate	10

Departure

Delay	Method	Speed	Frames
5 Seconds	None	Immediate	10

Image Time Line - Microsoft Internet Explorer

Button | Child Button | Child Image

Define the Time Line for the Child Button:

Appearance

Delay	Method	Speed	Frames
Inactive	None	Immediate	10

Departure

Delay	Method	Speed	Frames
5 Seconds	None	Immediate	10

Fig. 62

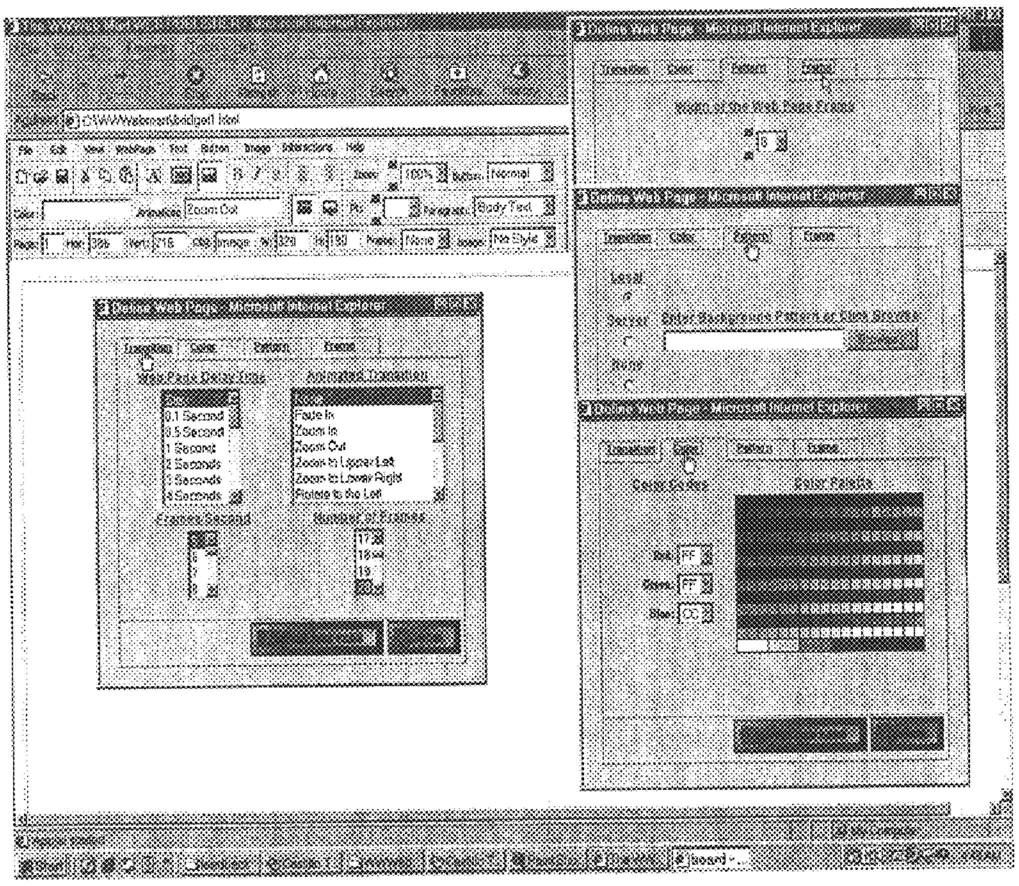

Define Web Page - Microsoft Internet Explorer
Transition
Color
Pattern
Frame
Width of the Web Page Frame
Local
Server
None
Enter Background Pattern or Click Browse
Color Sliders
Color Palette
Red
Green
Blue
Web Page Delay Time
Animated Transition
0.1 Second
0.5 Second
1 Second
2 Seconds
3 Seconds
4 Seconds
Fade In
Zoom In
Zoom Out
Zoom to Upper Left
Zoom to Lower Right
Rotate to the Left
Frames/Second
Number of Frames
C:\WWWebmart\bridget1.html
Zoom Out
Body Text
No Style
Normal

Fig. 63

BROWSER BASED WEB SITE GENERATION TOOL AND RUN TIME ENGINE

FIELD OF THE INVENTION

The present application is directed to computing systems, and more particularly to methods and apparatus for building a web site using a browser-based build engine.

BACKGROUND

Conventional web site construction tools operate on traditional operating system platforms and generate as output HTML (hyper text mark-up language) and Script Code (e.g., JavaScript). A browser draws a web page associated with the web site by interpreting the HTML and JavaScript Code. However, conventional mark-up and scripting languages include numerous inherent limitations. Conventional mark-up and scripting languages have not been designed for serious multimedia applications. They have almost no file handling ability and very little computational power. In addition, they are remarkably slow and inefficient.

As such it is virtually impossible to write a web publishing application in HTML and JavaScript. All conventional implementations must, and do, utilize a full-featured programming language, such as C++ or Visual Basic. Since the current popular browsers do not support these languages, by necessity, conventional web publishing applications run on platforms other than the World Wide Web (WWW) and its browsers. Therefore, at best, a conventional web publishing application can offer only a crude preview capability of what a real web page will look like.

Although C++ and Visual Basic are very capable languages, the conventional web publishing applications written in these languages are still necessarily limited by the limitations inherent in their form of output, which as described above is typically HTML and scripting code. As such, a conventional web publishing application written in one of these languages suffers from the severe performance problems inherent in these languages.

For example, HTML and JavaScript are incapable of reformatting text and scaling buttons or images dynamically. In addition, most conventional web publishing applications design a web page layout to fit into a 640 pixel wide screen. This means that the ability for higher resolution screens to display more data horizontally is lost. Since capability is wasted on the horizontal plane, unnecessary vertical scrolling may be required. Further, on higher output resolution devices (screens), unsightly extra white space or background may be prevalent.

SUMMARY

In one aspect the invention includes a Browser Based build engine that is written entirely in a web based full featured programming language (e.g., JAVA). A Browser Based Interface (the "Interface") between the web designer and the build engine is included. The browser-based interface can be written in the World Wide Web's (WWW) Hypertext Markup Language (HTML) and its Extensions (Dynamic HTML, JavaScript and Cascading Style Sheets). The Interface includes a unique set of communication techniques. One technique allows for effective two-way communications between a JAVA engine and JavaScript. Another technique allows for communications between a JAVA applet object inside a JavaScript window, with the JAVA engine, which permits the implementation of advanced intelligent interface objects, such as a "slider" or a "dial".

In one aspect the invention includes a screen resolution sensing mechanism that causes a build engine (i.e. build tools) to adopt its interface to the web designer's screen resolution.

In one aspect, the invention includes a multi-dimensional array structured database, that, in addition to storing the numeric and string data found in conventional databases, also stores multi-dimensional arrays of various multimedia objects. They include colors, fonts, images, audio clips, video clips, text areas, URLs and thread objects. The invention includes a run time generation procedure that creates a compressed web site specific customized run time engine program file, with its associated database and a build engine generated HTML shell file.

The invention can include web page scaling technology, so that when the web site/web page is accessed on the WWW, the web pages and all the objects within them can be scaled to the user's screen resolution and to the then current browser window size.

In one aspect, the invention includes a proprietary multi-level program animation model (threads) that responds to multiple user interactions and time sensitive operations simultaneously.

In one aspect, the invention includes a mechanism for the dynamic resizing of the build engine's web page size during various editing operations.

In one aspect, the invention includes techniques for creating browser based interface objects that visually and behaviorally are identical to those of the MS Window's standard.

Aspects of the invention can include one or more of the following features. A browser based build engine is provided that includes a browser based interface. The entire web site build process is WYSIWYG (what you see is what you get), with the web designer working directly on and with the final web page. The data produced by the build engine is processed and ultimately placed into a multi-dimensional array structured database, and stored in an external file. A run time generation procedure creates a compressed program customized run time engine file, with an associated database and a build engine generated HTML Shell File.

When the web site/web page is accessed on the WWW, web page scaling technology can be accessed to generate web pages that are scaled to the user's screen resolution. A technique is provided so that an applet's size (height and width) can be set in real time under the control of either the interface or the build engine. At the same time a multi-level program animation model (threads) is activated for user interactions and time sensitive operations.

The browser based interface technologies create a set of interface objects with a look and feel that is identical to that of MS Windows, yet includes technologies that equal or occasionally surpass those of high end word processors, desk top publishers, and image processing software programs, particularly in the areas of interaction, animation, and timeline technologies. The run time engine includes multimedia capabilities often rivaling the digital processing capabilities seen on television and in the movies.

Because of the implementation of a variety of performance and file reduction techniques, the entire run time environment can range from as low as 12K, and no larger than 50K. This depends upon the features selected by the web designer. Although the compressed image, audio, and/or video files must also be downloaded, with a reasonable web site design, web pages should load quickly. The initial run time environment is no larger than 25K, thus the initial web page should generally load in less than 2 seconds, and subsequent web pages in less than 1 second with a 56K modem, even with numerous image files.

The present invention provides a real time, dynamic linkage between JAVA and HTML including two-way communications, in real time, between JAVA and JavaScript.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing aspects and many of the attendant advantages of the invention will become more readily appreciated through the following drawings and their associated screen shots, referred to throughout the detailed description, wherein:

FIG. **1** is a flow chart depicting a prior art conceptual overview of how a user and a web browser interface.

FIG. **2** is flow chart depicting a conceptual overview of how a user interfaces with a web browser when implementing the present invention to construct a web site.

FIG. **3***a* is a schematic diagram showing the main components of a build tool in accordance with one implementation of the present invention.

FIG. **3***b* is a process flow diagram showing a build process in accordance with one implementation of the present invention.

FIG. **4***a* is schematic diagram showing the main components of a run generation tool in accordance with one implementation of the present invention.

FIG. **4***b* is process flow diagram showing a run time process in accordance with one implementation of the present invention.

FIG. **5** is a flow chart, with its attendant screen shot shown in FIG. **37**, that depicts a detailed view of a build time initialization procedure in accordance with one implementation of the present invention.

FIG. **6** is a flow chart, with its attendant screenshots shown in FIGS. **38**–**48**, that depicts a detailed view of the build time supported user input techniques and techniques for communication of data and status between the build engine and the interface in accordance with one implementation of the present invention.

FIG. **7***a* is a flow chart that shows an overview of the build time techniques for implementation of pop-up windows (usually called "dialog boxes" in MS Windows), the panel interface, and interface for color selection.

FIG. **7***b* is a flow chart, with its attendant screenshots shown in FIGS. **37**–**38**, that shows a detailed view of the build time techniques for implementation of panel interface objects, including the menu bar, menus and sub-menus, the tool bars, status fields, interactive fields, and interactive pull down lists, in accordance with one implementation of the present invention.

FIG. **7***c* is a flow chart, with its attendant screenshots shown in FIG. **37** and FIG. **63**, that shows a detailed view of the build time techniques for implementation of tabbed pop-up windows (also called "dialog boxes" in MS Windows).

FIG. **8** is a flow chart that shows a detailed view of the build time techniques for updating the internal databases and the setting of feature flags for run time optimization purposes.

FIG. **9** is a flow chart, with its attendant screenshot shown in FIG. **37**, that shows a detailed view of the build time polling methods used to facilitate communication from the JAVA build engine to the interface.

FIG. **10** is a flow chart that shows a detailed view of the build time techniques for analyzing user input for error checking and data integrity.

FIG. **11** is a flow chart, with its attendant screenshot shown in FIGS. **38**–**41**, that shows a detailed view of the build time methods for direct text entry at an arbitrary cursor position and text editor implementation methods.

FIG **12** is a flow chart, with its attendant screenshot shown in FIGS. **49**–**56**, that shows a detailed view of the build time techniques for reading external image files, creating them on a web page, and then manipulating them through either direct mouse interaction or through the interface's panel/windows.

FIG. **13** is a flow chart that shows a detailed view of the build time implementation of text, button and image styles in accordance with one implementation of the present invention.

FIG. **14** is a flow chart that shows a detailed view of the video and audio processing in accordance with one implementation of the present invention.

FIG. **15** is a flow chart that shows a detailed view of the frame, table, forms, and draw objects processing and technology in accordance with one implementation of the present invention.

FIG. **16** is a flow chart that shows a detailed view of the build time methods for supporting various user interactions at run time.

FIG. **17** is a flow chart, with its attendant screenshots shown in FIGS. **57**–**58**, that shows a detailed view of the build time methods for text button and image object animation.

FIG. **18** is a flow chart, with its attendant screenshots shown in FIGS. **59**–**60**, that shows a detailed view of the build time methods for text button and image transformations.

FIG. **19** is a flow chart, with its attendant screenshots shown in FIGS. **61**–**62**, that shows a detailed view of the build time methods for text button and image time lines.

FIG. **20** is a flow chart with its attendant screenshot shown in FIG. **63**, that shows a detailed view of the build time web page transition animations and time lines.

FIG. **21***a* is a flow chart that shows a detailed view of file operations performed in accordance with one implementation of the present invention.

FIG. **21***b* is a flow chart that shows a detailed view of the view operations performed in accordance with one implementation of the present invention.

FIG. **22** is a flow chart that shows a detailed view of a dynamic web resizing process that is activated by the "Open" and "Web Site" commands under the "File" menu and the "Zoom" command under the "View" menu.

FIG. **23** is a screen shot showing a file selection window operation in accordance with one implementation of the present invention.

FIG. **24** is a flow chart showing a detailed view of an external database in accordance with one implementation of the present invention and also shows the security and optimization techniques that can be employed.

FIG. **25** is a flow chart showing a detailed view of a method for creating a customized and optimized run time engine in accordance with one implementation of the present invention.

FIG. **26** is a flow chart showing a detailed view of the methods for creating an HTML shell file in accordance with one implementation of the present invention.

FIG. **27** is a flow chart showing a detailed view of the methods for creating compressed CAB and JAR files in accordance with one implementation of the present invention.

FIG. **28** is a flow chart showing a detailed view of the technology for dynamic web page size creation in accordance with one implementation of the present invention.

FIG. **29** is a flow chart showing a detailed view of the methods for reading the multimedia database and generating the necessary objects in accordance with one implementation of the present invention.

FIG. **30** is a flow chart showing a detailed view of the methods for dynamically scaling the web page object(s) to different screen resolutions and window sizes in accordance with one implementation of the present invention.

FIG. **31** is a flow chart showing a detailed view of the methods for executing a multi-level web page and object thread architecture in accordance with one implementation of the present invention.

FIG. **32** is a schematic diagram that shows a detailed view of the web page transition animation architecture in accordance with one implementation of the present invention.

FIG. **33** is a schematic diagram that shows a detailed view of the parent object time line architecture in accordance with one implementation of the present invention.

FIG. **34** is a schematic diagram that shows a detailed view of the child object time line architecture in accordance with one implementation of the present invention.

FIG. **35** completes the flow chart begun at FIG. **31**.

FIG. **36** is a flow chart showing a detailed view of the methods for responding to user interactions in accordance with one implementation of the present invention.

FIGS. **37–63** are screen shots of the user interface presented by the build process in accordance with one implementation of the present invention.

DETAILED DESCRIPTION

Referring to FIG. **1**, in a prior art process for creating and displaying a web site, the user either directly writes HTML and Script Code providing user input at **1** or operates a related prior art product at **2**, which generates the HTML and Script Code at **3**. A separate file, with its attendant HTML and Script Code is uploaded for each separate web page in the web site at **4**, which is then interpreted by a browser when accessed at **5**.

FIG. **2** shows a process for creating and displaying a web site in accordance with one aspect of the invention in which, a user operates a build tool at **6**, working directly with one or more of the final web pages in a full WYSIWYG mode. The build tool accepts the user input and creates a multi-dimensional embedded multimedia object database at **7**. A run time generation process is then invoked to create the necessary run time files at **8** (including HTML shell, CAB/JAR files and a customized runtime engine) which are then loaded to a user's web site at **9**. The web page(s), when viewed by a web surfer, are activated by the browser calling the customized run time engine at **10**. The run time engine then begins to read the database and down load image, audio and video files, while simultaneously drawing the first web page for viewing or user interaction at **11**.

Build Tool and Process

FIG. **3***a* shows a build tool **350** at the detailed component level. The build tool includes a build engine **352**, interface **354**, screen sensing mechanism **356**, multi-dimensional array structured database **358**, interface's database **360**, web

page scaling engine **364**, time line engine **366** and installation Program **368**. The operation and use of each of these components is described in greater detail below.

FIG. **3***b* is a flow of the build process executed by the build tool to create a web page/web site. Referring to FIGS. **3***a* and **3***b*, the process begins with an initialization (**12**) and continues through to a point where a web site has been defined and stored in the build engine's internal database (**29**).

The build tool **350** includes plural individual tools that are created and initialized at (**12**). The processes for creating and initializing build tools are described in greater detail below in association with FIG. **5**. After the build tools are created and initialized at **12**, the build tool **350** interacts with the user, receiving user commands (actions), for example, to build a web site. The build tool **350** processes user responses and communicates the same and status information to both the build engine **352** and interface **354** at **13**. The processes for interacting with the user are described in greater detail below in association with FIG. **6**.

In one implementation, the interface includes a panel (and its objects, including a menu bar, menus and sub-menus, tool bars, status fields, interactive fields and interactive pull down lists), pop-up windows (called "dialog boxes" in MS Windows), color and alert message interface technologies, built with HTML, Dynamic HTML (DHTML), JavaScript, and Cascading Style Sheets (CSS). Interface **354** responds to the user input and may display a pop-up window, update the interface objects, or display alert messages, as shown at **15**. The operation of the interface **354** is described in greater detail below in association with FIG. **7***a*, FIG. **7***b* and FIG. **7***c*.

As the build engine **352** receives data and status information, it updates an internal database (part of multi-dimensional array structured database **358**) and sets feature flags at **14**. The processes for updating the internal database and setting flags are described in greater detail below in association with FIG. **8**. To enable effective two-way communication between the interface and the build engine, polling technology is included as shown at **16**. The details of the polling process are described in greater detail below in association with FIG. **9**.

Whenever user input is received, the build tool **350** analyzes the input including error checking at **17**. In one implementation, the input is analyzed and then processed by object type (class). The process for analyzing input to determine type is described in greater detail below in association with FIG. **10**. In one implementation, the number of different object processing technology classes are four, and include direct text entry (**18**), image processing (**19**), video or audio files and links (**21**) and frames, tables, forms and draw objects (**22**). The build tool **350** processes the user input based on class. The processes invoked for direct text entry are described in greater detail below in association with FIG. **11**. The processes invoked for image processing is described in greater detail below in association with FIG. **12**. The processes invoked by the text button, paragraph, and image style technologies are described in greater detail below in association with FIG. **13**. The processes invoked for processing audio and video files and channels are described in greater detail below in association with FIG. **14**. The processes invoked for processing frames, tables, forms and draw objects are described in greater detail below in association with FIG. **15**. When an image, text button or paragraph object is to be inserted in the web page, the current style that is selected in the panel defines the initial settings used when creating the object in the web page. As such, button, image and paragraph style setting and technology will be invoked at **20** depending on the user input. The processes invoked by the paragraph style setting and technology is described in greater detail below in association with FIG. **13**.

After the input is processed as described above, a check is made to determine if one or more animation or transformation (interaction) techniques are to be invoked at **23**. The run time engine provided in accordance with the teachings of the present invention support various user interactions, including support for numerous animation and transformation techniques, and both web page and object time lines. Depending on the user selections, one or more technologies may be invoked. In the implementation shown, the build tool **350** is configured to check to determine if the input data is related to plural technologies including: user interaction technology (**24**), animation technology (**25**), transformation technology (**26**), object time line technology (**27**) and web page transition animation technology (**28**). The processes invoked for user interaction technology are described in greater detail below in association with FIG. **16**. The processes invoked for animation technologies are described in greater detail below in association with FIG. **17**. The processes invoked for transformation technologies are described in greater detail below in association with FIG. **18**. The processes invoked for object timeline technologies are described in greater detail below in association with FIG. **19**. The processes invoked for web page transition animation technologies are described in greater detail below in association with FIG. **20**.

After the build tool **350** has processed the user input, one or more file operations can be invoked at **29***a*. In one implementation, the file operations are "save", "save as", "new", "close", "open", "apply" and "web site". If "open" or "web site" are selected, the build tool **350** initiates the dynamic web page resizing process at **29***c* (See FIG. **22**). If "save" or "save as" are selected, the build tool **350** initiates a run generation process (See FIG. **4** and FIG. **24**). File operations "close", "open", and "new" can also initiate the run generation process, based on the state of the build process and user action.

At any time during the processing of user input, one or more view operations can be invoked at **29***b*. In one implementation, the view operations supported are "normal", "preview", "play", and "zoom" (at various zoom percentages). If any of the "zoom" levels are selected, the build tool initiates the dynamic web page resizing process at **29***c* (See FIG. **22**). If the "preview" or "play" view operations are selected they will initiate the run time process (See FIGS. **28** through **36**). FIG. **4***a* shows a run generation and runtime tool **370** at the detailed component level. The run generation and runtime tool **370** includes a run generation procedure **371**, web scaling engine **372**, a database **374** and a (web) page size generation engine **376** and run time engine **377** including a runtime user interaction engine **378**, a runtime timeline engine **380** and a runtime drawing, animation, audio, and video engine **382**. In one implementation, run time engine **377** includes plural engines, each of which may in themselves include plural engines.

FIG. **4***b* shows the run processes including methods for creating the run time files, including the external database, the web site specific customized run time engine, the HTML shell file, and the compressed CAB/JAR file. The run processes also include methods for scaling each web page to the web surfer's then current screen resolution and web browser window size. After a web page has been scaled, a

run time engine executes a multi-level thread technology, which presents to the viewer web pages that can operate under time lines that may include animated transitions. Associated with the web page time lines can be object time lines that may define entrance, main and exit animations, transformations, and synchronized time lines for child objects. Each object can have multiple object states, responsive to various user interactions, which can result in numerous types of visual and audio responses and actions.

Referring now to FIGS. **4***a* and **4***b*, a run generation process **360** begins by invoking the run generation procedure **377**. The run generation procedure **371** begins by creating the external database (part of database **374**) at **30**. The external database may include references to image, video and audio files, and video and audio channels. The process for creating the external database is described in greater detail below in association with FIG. **24**. A customized and optimized run time engine (run time engine **377**) is created at **31**. The customized and optimized run time engine (run time engine **377**) generates the web pages for the web site and is activated from the user's server. The process for creating the run time engine **377** is described in greater detail below in association with FIG. **25**. The HTML shell file is created at **32**, and then the CAB and JAR files are created at **33***a*. The HTML shell file includes JavaScript Code to activate and interrogate the page size generation engine **376**, and to activate the entire runtime engine. The CAB and JAR files both include the runtime engine and database in compressed executable form. The CAB file(s) will be activated by the HTML shell file if it senses the browser as being Microsoft Explorer, otherwise it will activate the JAR file(s). The processes for creating the HTML shell file and the CAB and JAR files are described in greater detail below in association with FIG. **26** and FIG. **27**, respectively. The run generation process portion of the run processes is completed as the HTML shell file and the CAB and JAR files are uploaded to the user's web site at **33***b*.

After the upload, the run time process **365** portion begins with the run time engine **377** invoking a web page size generation technology (engine) **376** at **34**. The web page size generation technology can be used to determine the screen resolution and the current browser window size. The process for invoking and initializing the web page size generation technology is described in greater detail below in association with FIG. **28**. The external database is read and the necessary objects generated at **35** from their stored external references. These objects include image, audio, and video objects. The processes for generating the necessary objects are described in greater detail below in association with FIG. **29**. A web page generation and scaling technology (web page scaling engine **372**) is then invoked at **36**. The web page scaling engine **372** can be used to reformat and scale objects that had been placed in a web page during the build process. The processes employed by the web page generation and scaling technology are described in greater detail below in association with FIG. **30**. The run time engine then, as necessary, executes a multilevel web page and object thread technology at **37** while the runtime user interaction portion **378** of run time engine **371** responds to user interactions at **38**. The processes invoked by the multilevel web page and object thread technology are described in greater detail below in association with FIGS. **31**–**35**. The processes invoked by the run time engine to respond to user interactions are described in greater detail below in association with FIG. **36**.

Detailed Build Processes

Referring now to FIGS. **3***a* and **5** through FIG. **22** the build tool **350** and its associated build process are described.

Referring first to FIGS. **3***a* and **5**, initialization methods are shown. At **39** the build tools are created as part of the execution of the installation program **368**. They can include:

1: Initial build tool HTML/JavaScript file (IBTF)

2: An initialization engine (IE).

3: A build engine.

4: The build engine parent HTML frame file. (PFF).

5: A "Control Panel and Status Line" HTML/JavaScript File ("panel") for;

Controlling the JavaScript database.

Calling and initializing all pop-up windows.

Reading all pop-up window values, and updating a JavaScript database

Calling the build engine and passing all necessary data and status information.

Polling the build engine for two-way JAVA/JavaScript communication.

Displaying and updating the status of its interface objects.

Issuing alert messages.

Processing direct user interactions with the panel's interface objects.

6: Numerous HTML/JavaScript files, one for each pop-up window.

7: JAVA applets, embedded in HTML/JavaScript pop-up window files.

8: A build engine HTML definition file that is created and modified dynamically.

The initialization and build engines can be placed in a JAVA wrapper so that JavaScript code may receive and process return values from JAVA methods. The initialization and build engines are also created in a "Signed" CAB file, and assigned the necessary security rights, so that the engines can assert the necessary permissions, if permitted by a given browser's security manager, when read or write operations are required. In one implementation, an installation program is run prior to the first use of the build tools. After installing all of the files, the installation program can install the necessary class libraries required by the run generation process in which the customized and optimized run time engine is created (See FIG. **25**). The installation program can also set the necessary environmental variables and installation options.

At **40** the web surfer points a browser at (i.e. calls) an initial build tool HTML/JavaScript file (IBTF). At **41** the IBTF identifies the current browser type and version number. Presently, each browser has different security manager implementations. In one implementation, the invention supports the following three categories:

1: With appropriate signing and time stamping, and with appropriate assertions of permissions, the browser will permit local read/write operations.

2: With appropriate signing and time stamping, and with appropriate assertions of permissions, the browser will permit local read operations, but write is only legal if sent to a server.

3: Local read/write operations are illegal, but are permitted on the server. The IBTF can include a flag that can be set to indicate which security implementation is supported, so that all subsequent read/write operations will comply with the current browser's security manager.

At **42**, the IBTF causes the browser to execute the IE so as to sense the screen resolution and for adapting the interface to the user's screen resolution. In one implementation, after entering a delay loop and waiting for the IE to report it is fully loaded and initialized, the IBTF calls two IE methods, which return the width and height of the current screen and browser window. The IBTF then checks for the presence and value of a "mode cookie", to determine whether this is an initialization process, a web site open command process, or a dynamic web page resizing process. If the mode cookie is set to initialize, or it doesn't exist, the IBTF calls the IE to generate the build engine's HTML definition file. At **43** the IE then asserts the required security permission and at **44** creates a build engine HTML definition file and writes this file to the local disk (as appropriate). At **45** the IBTF then turns control over to the PFF for activating the "paner" and build engine and displaying the build engine user interface screen.

The build engine user interface screen includes a "panel" portion and a build engine portion, each of which are loaded into their respective frames, after which the web site page(s) build process can begin. Screen shot FIG. **37** shows a representation of the user interface presented by the build tool. The user interface includes a panel **400** and build frame **500**. Panel **400** includes a menu bar **410**, menus **420** and sub-menus **430**, tool bars **440**, status fields **450**, interactive fields **460**, interactive pull down lists **470** and operational pop-up windows **480**. The menu bar **410** can be used for selecting a menu command that will cause a menu to be drawn. The menu (one or menus **420**) can be used to select a feature command that could cause an operational pop-up window to be drawn, a direct user input technique or object manipulation technique to be activated, or a sub-menu **430** to be drawn. A sub-menu (one of sub-menu **430**) can cause the same type of events as that of a menu. The tool bars **440** include various icons that are shortcuts to feature commands that are also available through the menu bar and its menus. In addition, the tool bar **440** can be used to show the current state of a feature. Status fields **450** show the current value of a certain setting. Interactive fields **460** also show the current value of a setting, but can also be directly changed by the user by typing into the field, with the result immediately processed by the build engine **352** and displayed in the build frame **500**. Interactive pull-down lists **470** also show the current value of a setting, but, if selected with a mouse click, will drop down a selection list, which may have an elevator attached. The user can click on an item in the selection list, which will become the current setting with the result immediately processed by the build engine **352** and displayed in the build frame. Operational pop-up windows **480** can have tabs assigned if the number of choices within the pop-up window is large. One or more settings can be changed through a pop-up window, with the results immediately processed by the build engine **352** and displayed in the build frame **500**. These interface techniques are described in greater detail below in the build process.

The build frame **500** is used to present the actual web page as constructed by a user. The user can directly enter text, import images, video and audio for display/playback and create animations and transformations that can be viewed in the build frame. FIG. **6**, with its attendant screen shots FIG. **38** through **48**, shows the user input techniques supported in one implementation of the invention. In one implementation, the user inputs supported include: selection from a JAVA window object (**48**); selection from a JavaScript window (**49**) including selection with dual spin control (**50***a*) or selection from a JavaScript child window object (**50***b*); direct text entry (**51**); page resizing (**52**); direct object manipulation (**53**); and, selection from a JavaScript panel (**54**).

In the implementation shown, of the six user input techniques sensed at **13**, the code for supporting selections from a JavaScript pop-up window at **49** and selections from the "panel" at **54** were implemented entirely in HTML/JavaScript Code, while support for direct text entry at **51** and direct web page object manipulation at **53** were implemented entirely in JAVA (or any other browser-based full featured programming language). In one implementation, code for supporting selections from a JAVA Window object at **48** and dynamic web page resizing at **52** are implemented using both HTML/JavaScript and JAVA. Those of ordinary skill will recognize that, JAVA could have been used more extensively to implement the methods described at **48**, **49** and **54**. However, in order to achieve the most intuitive and MS Windows like interface, and because effective two-way communication between JavaScript and JAVA had been achieved (See FIG. **9**), the languages proposed appear to best support the particular user input technique.

For example, FIG. **23** shows an actual file selection window **2300**, implemented by the invention. This type of file selection window is available in JavaScript/HTML, but not supported by JAVA for applets. File selection window **2300** greatly enhances the interface for the user, as the image, sound clip, or video clip names need not be memorized. File selection window **2300** further eliminates possible operator error when typing in a pathname or filename. The present invention utilized the strengths of JavaScript/HTML with the power of JAVA to create a unique browser based interface solution. In one implementation, the HTML form element "INPUT type=file" was embedded in a JavaScript pop-up window to create the file selection window. The file selection window returns a string value of the image (or other file type) pathname to the pop-up window. The pop-up window's JavaScript then could be used to call a JavaScript function in the panel (panel **400**) which:

1: Reads the pathname value in the pop-up window.

2: Creates a string version of a valid URL by adding the correct URL protocol to the string.

3: Updates the panel's database (interface's database **360**).

4: Calls a JAVA method in the build engine, which casts the string value of the URL into a URL object, creates an image object which is then drawn on the screen, and updates its internal database.

User inputs that are a selection from a JAVA window object (**48**) permit the implementation of a vast array of intelligent user input interface objects, from sliders to dials, which are extremely intuitive and significantly enhance the user's ergonomic experience. In one implementation, user input interface objects are supported as follows. When a selection from a JAVA window is detected, a pop-up window (applet) is presented (associated with the feature being manipulated, e.g., color, volume) and an engine method is called to begin two-way communication (for passing as arguments any necessary status information). The engine begins polling a JAVA abstract object waiting for a static variable's value to change. The pop-up applet processes the value as defined by a user interaction event, and updates the static variable in that same JAVA abstract object with the new value. Upon detecting a change in the polled static variable, the engine calls the necessary methods to process that new value. These methods include can include a brightness filter that is applied to the image bitmap utilizing techniques very similar to that of that employed by the "fade in" and "fade out" animations, described in association with FIG. **33**

User inputs for a selection from a JavaScript pop-up window (**49**) can be made in a manner identical to that of

making a selection from a dialog box under MS Windows, including the use of tabbed JavaScript pop-up windows. In one implementation when a selection from a JavaScript pop-up window is detected, the panel's (panel **400**) JavaScript opens a pop-up window. The pop-up window's initial values are set from a JavaScript database defined in the panel or by the panel calling the engine for the current values and then setting the initial values. In a tabbed JavaScript window, clicking on a tab will call the pop-up window's JavaScript in order to change the state and appearance of the tabbed JavaScript window in the expected way. The pop-up window's JavaScript calls the panel's JavaScript when a completion event occurs. The panel's JavaScript reads or the pop-up window's JavaScript writes the pop-up window's field values, causing the panel's database to be updated, and the panel then calls the appropriate build engine **352** method, passing as arguments the necessary data and status conditions. Initializing the pop-up window's values and updating the panel's database upon completion can alternatively be implemented by JavaScript functions executed within the pop-up window's HTML file.

In addition, there are interface extensions that can extend beyond the usual MS Windows implementations. One is support for a selection from a dual spin control at **50**A. Screen shots FIGS. **42**–**45** show a visualization of an implementation of this interface technique. Screen shot FIG. **42** shows the mouse placed over an upper spin control. Screen shot FIG. **43** shows the result after the user clicked once on the upper spin control. Notice that the value has been incremented by 1, and the text button object is now at a larger point size. Screen shot FIG. **44** shows a combo box list selected by the mouse with the user about to select a significantly larger point size. Screen shot FIG. **45** shows the result of that selection, including the effect on the text button object.

In one implementation, dual spin controls are supported as follows. Each spin control has three visual states, so that when the user places the mouse over the control it appears to light up, and when the mouse button is depressed (pressed down), the spin control is modified to give the appearance of being pressed. JavaScript methods are called in the panel (panel **400**) to:

1: process each mouse click event over either spin control,

2: range check as necessary,

3: update the value in the HTML frame object residing in the pop-up window,

4: update the JavaScript (panel **400**) database,

5: call the build engine **352**, if necessary, passing the necessary value and status.

If the mouse is clicked on a combo box, the selection window opens in the usual way. If a mouse click in that window is detected, another JavaScript method in the panel **400** is called to update the JavaScript database, and call the build engine **352**, if necessary, passing the necessary value and status as function call arguments.

Another interface extension is selection from a JavaScript child window at **50**B. This technique helps simplify the number of choices given to the user in a complex pop-up window operation. A selection from a JavaScript child window can be supported as follows. The panel's (panel **400**) JavaScript opens the pop-up window. The pop-up window and its child pop-up windows' initial values are set from the JavaScript database defined in the panel **400**. The pop-up window's JavaScript opens the child pop-up window and sets its initial values. The child pop-up window's JavaScript calls the pop-up window's JavaScript when a completion event occurs. The pop-up window's JavaScript reads the child pop-up window's values, sets those values to its own internally defined variables, and calls the panel's JavaScript. The panel's JavaScript reads the pop-up window's values (which include the settings for its own fields as well as those of its child windows), updates its database, and calls the appropriate build engine **352** method, passing as arguments the necessary data and status conditions. Screen shots FIGS. **46**–**47** show a visualization of an implementation of a JavaScript child window. Screen shot FIG. **46** show a change text button style pop-up window. Screen shot FIG. **47** shows the result after the user selected the "Define the Mouse Down Text Button Style" child pop-up window.

Direct text entry is supported at any arbitrary cursor location. In one implementation, "text areas" are utilized in an unconventional way, in order to support full text entry, text editing, text button and paragraph styles, and reformat. Direct text entry can be defined at any arbitrary cursor location, and then text can be dragged to any other arbitrary location.

Text areas are objects that are utilized by JAVA primarily as an interface object for the implementation of a form and are generally "added" to the screen at the initialization time of a JAVA applet. Text areas are decidedly not WYSIWYG. The present invention creates text areas dynamically. Screen shots FIG. **38** through FIG. **41** show a visualization for an implementation of this technique. Screen shot FIG. **38** shows the user selecting a text object from the create text icon object from a tool bar of the panel (panel **400**). When the text icon object is selected, the cursor shape is changed to indicate the selection while the text icon object is in the select state. Screen Shot FIG. **39** shows that the cursor has changed shape and that the user has placed the cursor at an arbitrary location on the web page. Screen shot FIG. **40** shows the result after the user has clicked the mouse. A text insertion point and a selection rectangle are drawn at the arbitrary web page location. Screen shot FIG. **41** shows the result after the user has pressed the letter "W" on the keyboard. As can be seen in screen shot FIG. **41**, a draw method associated with the build process immediately hides the text area. However, text editor methods associated with the build process continue to utilize the text area as a hidden, dynamically resizing frame, whose size is subject to text button or paragraph style settings, by the amount of text, by the text's orientation (vertical or horizontal) and by the text's font style(s) and font size(s). As the build engine **352** detects a relevant mouse event or keyboard event, the build engine **352** updates the necessary variables that are defined as return values in specified build engine methods. Polling technology (see FIG. **9**) retrieves the relevant values and calls the necessary JavaScript method for processing. In one implementation, these same techniques (text area techniques) are used in the scaling technology (See FIG. **30**). Since the direct text entry and editing processes bypass completely the interface and the JavaScript code, the polling technology (See FIG. **9**) is used to pass the text string values back to the JavaScript database, in order for the interface's pop-up windows to be correctly initialized for subsequent text operations.

Direct text processing at **51** begins with the build engine **352** detecting a "Mouse Drag" or a "Mouse Double Click" event. In one implementation of the present invention, if a mouse drag event is detected, the entire initial anchor word (assuming the "mouse down" event placed the text insertion point within a word) is selected as well as the entire closing anchor word. If a double click event occurs over a word, the entire word is selected. If a double click event is detected over a special hot zone (for example, just to the left of a paragraph line), then an integral number of words are selected. Appropriate four-dimensional variables are set, and a draw system is called. The draw system paints the selected line segment in the marked text background and text color. The build engine **352** then sets a return flag to be read by the polling technology (FIG. **9**). A panel JavaScript poller (FIG. **9**) detects this flag and redraws the panel's "Text" menu object showing the choices available when text is selected. In one implementation, the "Text" menu includes choices of "Text Style", "Hot Link", "Preferences", and "Format". The states for the tool bar icon objects of "Bold", "Italic" and "Underline" are set appropriately as is the setting for the point size interactive drop-down list. The panel's JavaScript then calls an appropriate build engine method that resets the flag. If the panel's JavaScript detects the user selecting the "Text Style", "Hot Link", "Preferences" or "Format" choices, it creates the appropriate pop-up window. Upon detecting a user completion event, the panel's JavaScript reads the data settings in the pop-up window, closes that pop-up window, and sends this data to an appropriate build engine method for processing (See FIG. **11**). Dynamic web page resizing at **52** is invoked when the build engine **352** detects a user initiated web page resize event. This could be caused by the "Open" or "Web Site" commands from the "File" menu, or from a "Zoom" command from the "View" menu. This technology is explained in detail below in association with FIG. **22**.

Direct object manipulation at **53** includes dragging of any object, resizing of non-text objects, rotation and other image manipulation functions, as required. The processing for direct object manipulation begins by analyzing the type of object selected and the state of the object, as set by the interface based on a user's panel selection. The build engine **352** then changes the mouse cursor's appearance, and the type of selection rectangle, including which attachment points, if any, should be drawn and activated. (See FIG. **10** for the mouse event processing technology and FIG. **12** for image processing technology). In one implementation, the same direct object manipulation polling technology is used as described above with regard to direct text entry.

If a selection of an interactive field, interactive drop-down list object, or a toll bar icon object from the JavaScript panel is detected at **54**, then the following steps can be invoked, depending on the selection. The point size of a paragraph, a marked text range inside a paragraph or text button object can be changed. The state of an object's 3D frame can be changed. In one implementation, three states for an object frame are supported. The 3D frame can be drawn as a "raised" 3D object, as a "depressed" 3D object, or as a "raised" 3D object that turns into a "depressed" 3D object if a mouse down event is detected over the object to which the 3D frame is assigned. An object's style can be changed. The current web page can be changed. Finally, any other operation that has been defined by a tool bar icon object in the panel can be invoked. This includes the "file" menu choices of new, open and save, the "edit" menu choices of cut, copy and paste, inserting a text, button or image object onto the web page, applying or removing the bold, italic, and underline text attributes for a text or button object, centering or uncentering any web page object, setting the animation for a button or image object, changing the zoom level of the web page, or previewing the web site.

As each new user input is received and processed in accordance with the steps shown in FIG. **6**, at all times the internal databases of the JavaScript panel and the build engine **352** are maintained completely in synchronization.

Synchronization is maintained so that: all status information displayed by the panel is current and correct; all data and status information passed to the build engine **352** from the interface are consistent with the build engine's state at any given time; the values in all pop-up windows are correctly initialized. In order to meet these requirements, all of the variables in the JavaScript panel database are explicitly "typed", to be compliant with the strict variable typing methodology generally imposed in all full featured programming languages such as Java. As JavaScript does not explicitly type anything, where using JavaScript herein, all string, Boolean, and integer variables are typed. Full two-way real time communication support between the JavaScript/HTML interface and the JAVA build engine **352** is provided as described below in association with FIG. **9**.

FIG. **7***a* shows four tools utilized for an implementation of the pop-up window and panel interface technology (**15** of FIG. **3**). The panel and pop-up windows make extensive use of JavaScript mouse events, including onMouseDown, onMouseUp, onMouseOver, onMouseOut, onClick and onchange methods (**56**). The pop-up windows make extensive use of the JavaScript onLoad and onUnLoad methods. In one implementation, when a pop-up window is loaded by the panel, the panel goes into a wait loop, set for 5 times a second using the JavaScript setTimeout method, interrogating in each loop whether the pop-up window's status flag has been set. Meanwhile the pop-up window, when loaded by the browser, executes the onLoad method in order to set a flag in the panel informing the panel that the pop-up window is now loaded. Upon detecting the load event completion, the panel then proceeds to initialize the fields in the pop-up window. The panel will always close a pop-up window after detecting its completion event. However, if the user has closed the pop-up window in a non-standard way, the pop-up window executes the onUnLoad JavaScript method, which sets a flag in the panel notifying it that the pop-up window has been closed.

The JavaScript code in the panel and in all pop-up windows make extensive use of JavaScript method onKeyDown for the following operations:

1: When the focus is on the icon representing completion ("OK" is used in many MS Windows applications) causing the enter key to initiate a pop-up window/panel completion event.

2: When the focus is on the icon representing cancellation ("cancel" is used in many MS Windows applications) causing the Esc key to initiate a pop-up window/panel cancellation event.

3: When the focus is on any pop-up window or panel object, such as a data entry field, a check box, a radio button, a drop-down and scrollable list, a scrollable list, an icon, or a DHTML tab object (discussed below), the navigation keys are captured by the onKeyDown method, a JavaScript function is called, and the appropriate change is made. For all pop-up window and panel objects, when the Tab key or the combination of the Tab key with the Shift key are detected, the onFocus JavaScript method is employed and the focus moves to the appropriate pop-up window object. If the pop-up window or panel object is a data entry field, drop-down list or a scrollable list, all cursor key operations are detected and the insertion point is adjusted accordingly. If the pop-up window or panel object is a check box, radio button a icon, or a DHTML tab object, and a cursor key (up, down, left, right, home and end keys, with or without the Ctrl or Shift keys) is detected, the onFocus JavaScript method is employed and the focus moves to the appropriate pop-up window object.

One methodology for this feature requires that all keyboard events be monitored, at all times. When the scan code for the enter key is detected, the appropriate JavaScript function is called to close a pop-up window and to call the appropriate JavaScript function for processing of the relevant data (updated in the window) and communicating, as necessary, with the build engine **352**. In another implementation, rather than the panel going into a wait loop awaiting notification from the pop-up window for data initialization purposes the pop-up window, when loaded, executes the onLoad JavaScript method, and reads the required data values directly from the panel's database, utilizing the JavaScript "opener.fieldname.value" technique. Similarly, the pop-up window, when detecting its completion event, updates the panel's database with the revised values from its own fields and then calls the appropriate JavaScript function in the panel for further processing. Both implementations, and any combinations, assure that the pop-up windows are correctly initialized, the panel's database is correctly updated, and the data is successfully sent to the build engine **352** for processing.

Extensive use of JavaScript technology is employed to enhance the user interface and for communication between the various HTML frames and/or files, within a given HTML frame or file, between an HTML frame and the JAVA engine, and as a bridge between two different JAVA applets (**57**). Extensive use is made of JavaScript arrays to store the values of all page and object attributes, to initialize the correct values in all pop-up windows, and to pass data and status to the engine. Various JavaScript techniques are employed to "type" all variables (JavaScript does not explicitly type anything as described above) as a prerequisite for passing values to the build engine **352**. Variables that should be typed as strings, integers and Booleans are typed through the use of "Eval" and "New" JavaScript functions. The choice of color, found in most pop-up windows to define one or more color elements, can be implemented utilizing several innovative JavaScript techniques. They include:

1: Defining a complex image map through a JavaScript function utilizing arrays. Screen shot FIG. **48** shows a visualization of an image map. A JavaScript computational loop utilizing arrays can be used to define each individual rectangle in this color palette with its appropriate RGB value and a function call to the appropriate JavaScript method.

2: Limiting the color choices from the image map to only those colors that are designated as safe colors. Safe Colors are the subset of all colors that are browser independent, assuring a consistent color look across all browsers.

3: Supporting a dual color selection technology. The user can be presented with a color palette and can click on a particular color in the color palette. Image map technology can call a JavaScript function, which converts that choice into a RGB numeric definition. This definition updates the RGB values shown in screen shot FIG. **48**, as well as passing those values, though an appropriate function call, to a build engine JAVA method. The build engine **352** will then draw the actual color immediately on the web page. Alternatively, the user can select a value from Red, Green or Blue selection lists, which can be implemented using an HTML drop-down list form object. The value selected is then processed by an appropriate JavaScript function call to a build engine method, which converts the RGB to a JAVA compliant value, and then draws the actual color on the web page.

4: Supporting True Transparency. For appropriate color elements, such as the background for a text button object, the user can choose, either from the color palette by clicking on a "transparency" rectangle, as described above, or by selecting "TIR" from a Red, Green or Blue selection list. This choice is then processed by an appropriate JavaScript function call to a build engine method, that in turn sets a particular flag for the draw system (of the Build Tool) to not draw a background color for that object.

Innovative techniques are used to enable JavaScript to dynamically create HTML code based on real time conditions. Cookies can be used for data communication between HTML frames and HTML files, some of which were created in real time. Many unique combinations of HTML elements, including frames, forms, and tables, enhanced by JavaScript code, are utilized to create a extensions beyond that of the MS Windows interface (**58**). For example, a dual combo box/spin control for both small and large numeric incremental jumps can be implemented by a combination of form and table elements, mouse events, and JavaScript methods.

Extensive use of Cascading Style Sheets (CSS) was employed to create a consistent look for all pop-up windows, and for precision placement of various HTML elements (**59**).

FIG. **7***b* shows a detailed view of the build-time techniques for implementation of panel interface objects, including the menu bar, menus and sub-menus, the tool bars, status fields, interactive fields, and interactive pull down lists, in accordance with one implementation of the present invention (**15** of FIG. **3**). These techniques create panel interface objects that have the same look and feel of those which are implemented under the various MicroSoft Windows Operating Systems. In one implementation of the present invention, the status fields, interactive fields, and interactive drop-down lists are defined as HTML form objects (text boxes and lists) embedded within DHTML objects. The menu bar, menus and sub-menus, and the tool bars can be defined as pure DHTML objects. However, Cascading Style Sheets can be used for all panel interface objects; although more extensively with DHTML objects as will be described below. In an alternative implementation of the present invention, the status fields and interactive drop-down lists are defined as pure DHTML objects.

In one implementation of the present invention the menu bar at **270** is defined as sets of DHTML objects, each set corresponding to a menu command. Each set consists of four DHTML objects with absolute screen positioning, one defining the DHTML object in the Mouse Over state at **278**, the second for the Mouse Down state at **279**, the third for the Active state, and the fourth for the Inactive state. Each state has a different CSS style assigned, which defines the visual appearance of that state. When the build tool is initialized at FIG. **5**, the appropriate menu commands are initialized as active or inactive at **277**. If the menu command is defined to be inactive, that DHTML inactive object is assigned by a JavaScript function to the "visible" style attribute, while the other three DHTML objects are assigned the "hidden" style attribute. Screen shot FIG. **38** shows a visualization of the "Interactions" menu command in the inactive state. In the inactive state all user interactions are ignored. If the menu command is defined to be active, that DHTML active object is assigned by a JavaScript function to the "visible" style attribute, while the other three DHTML objects are assigned the "hidden" style attribute. While in the active state, the JavaScript functions for "onMouseDown", "onMouseUp", "onMouseOver" and "onMouseOut" are implemented. If a Mouse Down user interaction event is detected over an active menu DHTML object at **279**, a menu command specific JavaScript function is called. This function sets the DHTML object for the Mouse Down state to the "visible" style attribute, calls a generalized JavaScript function to reset the visibility states all the other appropriate DHTML objects, set certain status variables, and set the DHTML object which defines the menu associated with that menu command to the "visible" style attribute. Screen shot FIG. **37** shows a visualization of the "Image" menu command after having received a mouse down event, with its associated menu **420** having been set to the "visible" style attribute. If a mouse up user interaction event is detected over an active menu DHTML object at **281**, a generalized JavaScript function is called in which the DHTML object defining the mouse over state is passed as a function call argument. This function sets the DHTML object defining the mouse over state to the "hidden" style attribute thus resulting in the appearance as shown for the image menu command in screen shot FIG. **37**, even when the mouse has been moved off the menu object. If a mouse over user interaction event is detected over an active menu DHTML object at **278**, a generalized JavaScript function is called in which three DHTML objects are passed as function call arguments as well as a menu command name. These DHTML objects are the ones defining the mouse over state, the mouse down state, and the associated menu. This JavaScript function first tests to see if a menu has been activated by a previous mouse down event and is still visible. If so, a generalized "reset visiblity states" function is called, then both the mouse down and associated menu objects are set to visible. Finally the same menu specific function is called as with the mouse down event. If no menu is visible, then the object associated with mouse over state is set to visible. If a mouse off user interaction event is detected over an active menu DHTML object at **281**, a generalized JavaScript function is called in which the mouse over DHTML object and the menu command name are sent as arguments. Logic tests are made to determine which menu command object has been left, as well as whether any menus are currently visible. Depending upon the results, the mouse over DHTML object may be set to hidden.

In one implementation of the present invention the menus and sub-menus at **271** are defined as a set of DHTML objects, one for each menu choice, nested inside an DHTML object that defines the entire menu. Each menu object is given absolute positioning, while the menu items are given absolute positioning relative the menu objects origin. Both the entire menu and each choice are assigned CSS styles to define their visual appearances. For each menu choice the JavaScript functions for "onClick", "onMouseOver" and "onMouseOut" are implemented. If a mouse click event is detected at **280** and no sub-menu is defined, a feature specific JavaScript function is called. First the menu bar and the menus are set to their appropriate visibility states. Then setting their visibility attribute style to "visible" activates the appropriate tool bar icon DHTML objects. Finally the feature specific JavaScript code is executed as discussed herewithin, which may cause a pop-up window to be displayed, the Panel's database to be updated, and/or the build engine **352** to be called. If a mouse over event is detected at **278** and no, sub-menu is defined, a generalized JavaScript function is called in which the menu choice object is passed as an argument. This function first calls a generalized JavaScript function to close any pop-up windows, then set a status variable and finally executes DHTML commands to set the correct text and background colors for the object. If a mouse off event is detected at **282** and no sub-menu is defined for a menu choice either immediately above or below, a generalized JavaScript function is called in which the menu choice object is passed as an argument. A status variable is set and DHTML commands are executed to set the correct text and background colors for the object. If a sub-menu is defined for a menu choice object, then the same sub-menu specific JavaScript function are called for both mouse click or mouse over events. This function performs the same steps as that of the generalized function that was called for a mouse over event, as well as setting the sub-menu object and its menu choice objects to the visible state. Screen shot FIG. **37** shows a visualization of the menu bar's "Image" command having been activated, the drawing of its associated menu **420**, the selection of the "Enhance" menu choice, and the drawing of the "Enhance" sub-menu **430**. In the event that the cursor is moved to an adjacent menu choice under the "Image" menu, such as "Animation . . . " or "Rotate", then a specific JavaScript function is called which, in addition to the functions executed by the generalized JavaScript mouse over function, also hides the "Enhance" sub-menu.

In one implementation of the present invention, the tool bars at **272** are defined as a DHTML objects, and a set of DHTML objects are defined for a tool icon. The tool is given absolute positioning and is assigned a CSS style in order to define is visual appearance. Each tool icon is assigned a set of three DHTML objects all with absolute screen positioning. The first DHTML object defines the mouse over state at **278**, the second for the mouse down state at **279**, and the third for the active state. Each state has a different CSS style assigned, which defines the visual appearance of that state. For each tool icon active state object the JavaScript functions for "onClick", "onMouseDown", "onMouseUp", "onMouseOver" and "onMouseOut" are implemented. GIF images are defined for the tool bar DHTML objects, and may be always visible. The inactive "grayed out" representations for each toll icon can be drawn on this image. When the build tool is initialized at FIG. **5**, the appropriate tool icon objects are defined as active or inactive at **277**. The inactive state for an tool icon is represented when all three of its associated objects are assigned the visibility style of "hidden". Screen shot FIG. **38** shows a visualization for several inactive tool icons including the icon commands for bold, italic, underline, left and centered. All user interaction events are ignored in the inactive state. If the tool icon, based on the state of the build engine and based on the polling technology described below, is set to an active state, then the tool icon's active state object is set to the visibility style of "visible". If a mouse click event is then detected at **280**, a feature specific JavaScript function is called in a manner identical to that for a mouse click event over a menu choice object as described above. If mouse down or mouse up events are detected at **279** or **281**, then respective generalized JavaScript functions are called, in which the DHTML object defining the mouse down state is passed as a function call argument. If a mouse down event was detected, then the generalized function sets the tool icon's mouse down object to the "visible" state. If a mouse up event was detected, then the generalized function sets the tool icon's mouse down object to the "hidden" state. If mouse over or mouse out events are detected at **278** or **282**, then respective generalized JavaScript functions are called, in which the DHTML object defining the mouse over state is passed as a function call argument. If a mouse over event was detected, then the generalized function sets the tool icon's mouse over object to the "visible" state. If a mouse off event was detected, then the generalized function

sets the tool icon's mouse over object to the "hidden" state. Screen shot FIG. **37** shows a visualization of the button tool icon with both its associated the mouse down and active objects set to "visible". Screen shot FIG. **38** shows a visualization of the text tool icon with both its associated the mouse over and active objects set to "visible".

In one implementation of the present invention, the status fields at **273** and the interactive fields at **274** are defined as HTML text boxes. In an alternative implementation status fields are defined as DHTML objects. For both of these panel interface object types, under certain conditions, the panel draws status information into these panel interface objects. This information can result from user input as discussed in FIG. **6**, or through the polling and two-way communication technology between the interface and the build engine **352** as discussed below. In one implementation of the present invention the status fields are:

1: The color of the selected web page object, in which the red, green and blue settings are shown.

2: The animation state of the selected button or image object.

3: The zoom level for the current web page.

4: The point size for the selected text or button object.

5: The horizontal position, in pixels, of the mouse cursor.

6: The vertical position, in pixels, of the mouse cursor.

7: The type of web page object (text, button, image, table, form object, draw object, etc.,) if selected. The type of object that the mouse is over, if no object is selected.

8: The width, in pixels, of web page object (text, button, image, table, form object, draw object, etc.,) if selected. The width of the object that the mouse is over, if no object is selected.

9: The height, in pixels, of web page object (text, button, image, table, form object, draw object, etc.,) if selected. The height of the object that the mouse is over, if no object is selected.

Screen shot FIG. **38** shows a visualization of the status fields in one implementation of the invention **450**. In an alternate implementation using DHTML objects, the status fields will appear two-dimensional rather than the three-dimensional look currently shown.

There is one interactive field defined in one implementation of the present invention. Screen shot FIG. **37** at **460** shows a visualization of the page number interactive field. In addition to the current web page being displayed, either as a number in one implementation or as a user defined name in an alternative implementation, the user can place the cursor into this field and enter the desired page to go to. A click at **280** or Enter Key event will execute this function.

In one implementation of the present invention, the interactive drop-down lists at **275** are defined as HTML form lists. In an alternative implementation, status fields are defined as DHTML objects. For both of these panel interface object types, under certain conditions, the panel draws status information into these panel interface objects. The interactive drop-down lists behave in a manner very similar to that of interactive fields, except that when selected, a selection list drops down for selection. Depending upon the number of entries in the list, an elevator object may be drawn. The operations of selecting the interactive pull down list, the selecting of a list item, or the operation of the elevator is identical to that of comparable MS Windows objects. In one implementation of the present invention the interactive pull down list are:

1: Zoom. This interface object has dual spin controls as described above and is always selectable except when in a preview mode. It shows the current zoom level.

2: Button Style. This interface object is always selectable except when in preview. It shows the button style of the currently selected button, if any. Changing the button style will change the style of the currently selected button, and/or define the style of the next button to be created.

3: Point Size. This interface object has dual spin controls as described above and is selectable when a text or button object is selected. It shows the point size of the currently selected text or button object, if any. Changing the point size will change the point size of the currently selected text or button object.

4: Paragraph Style. This interface object is always selectable except when in preview. It shows the paragraph style of the currently selected paragraph, if any. Changing the paragraph style will change the style of the currently selected paragraph, and/or define the style of the next paragraph to be created.

5: Frame State: The state of the 3D frame (none, raised, pressed or live) of the currently selected text, button, or image object.

6: Image Style. This interface object is always selectable except when in preview. It shows the image style of the currently selected image, if any. Changing the image style will change the style of the currently selected image, and/or define the style of the next image to be created.

Screen shot FIG. **37** shows a visualization of interactive drop-down lists **470**. In an alternate implementation using DHTML objects, the interactive drop-down lists will appear two-dimensional rather than the three dimensional look currently shown.

Tool bar icon objects, status fields, interactive fields, and interactive pull down lists all show feedback of the current build engine state. The technology utilized by one implementation of the invention is described below.

FIG. **7***c* shows a detailed view of the of the build time techniques for implementation of tabbed pop-up windows (**15** of FIG. **3**). These techniques create a pop-up window interface that visually and behaviorally is identical to that which is implemented as dialog boxes under the various MicroSoft Windows Operating Systems. Pop-up windows can be non-tabbed as described in FIG. **7***a*, or can have from two to as many as **10** or more tabs, depending upon the complexity of the choices available to the user for a given feature. In one implementation of the present invention each tab at **283** is defined as a DHTML object at **284**. The tab is given absolute positioning and is assigned a CSS style at **286** in order to define is visual appearance. When a click is detected through the JavaScript "onClick" function, a tab specific JavaScript function at **285** is called within the pop-up window's HTML file. This function sets the display style attribute for the DHTML objects that define the settings for all the non-selected tabs to the display style attribute of "none". The DHTML objects that define the GIF image of the non-selected tab file representations are also set to the display style attribute of "none". The display style attribute for the DHTML objects that define the settings of the currently selected tab and the GIF image that depicts the selected tab file representation is set to the display style attribute of "". If there is to a change of the focus of the selected field within the now to be visible tab specific choices, the focus attribute for that object is executed. Screen shot FIG. **37** shows a visualization of a tabbed pop-up window, and screen shot FIG. **63** shows a collage of four views of a tabbed pop-up window with four tabs. Notice that each state of the tabbed pop-up window has a different tab file representation, showing the selected tab as being in the forefront.

The interface technology of the invention, in addition to its utilization as part of a web-based web site generation tool, can be used to provide a general purpose interface for all web-based applications that want a MS Windows compliant interface.

A process for updating the internal database of the build engine **352** is shown schematically in FIG. **8**. The database is compact and efficient in order to meet the performance requirements for the run time process. The database handles a wide selection of data objects, including multi media objects such as colors, fonts, images, sound clips, URLs, threads, and video. The database supports a multi level animation, transformation, and time line model (discussed in greater detail below). The database complies with the differing rules imposed by the various popular browser security managers.

The process begins by determining the type of data to be updated at **60**. Data that defines generic web site settings (See FIG. **21***a*), screen resolution values (See FIG. **21***a* and FIG. **24**), and the web page high watermark setting (See FIG. **24**) can be stored in a header record as boolean and integer variables and URL and color objects at **62** and **63**. Data that defines web page, paragraph, text button, and image style and text button, image and paragraph high watermark settings can be stored in one-dimensional arrays as boolean, integer and string variables and URL, font, image or thread objects at **61** and **64**. The URL, color, font, image and thread objects can also be created as required at **64**.

Data that defines text button, image, paragraph, or other parent objects and paragraph line high watermark settings can be stored in two-dimensional arrays (by web page and by object number) as boolean, integer, string, floating point variables and URLs at **65** and **66**. Again, the URL, color, font, image, audio clip, video clip, text area and thread objects can also be created as required at **66**. Data that defines a paragraph line and paragraph line segment high watermarks can be stored in three-dimensional arrays (by web page, by paragraph number, and by line number) as Boolean, integer or string variables at **67** and **68**. Again, the URL, color or font objects can be created as required at **68**. Data that defines a paragraph line segment can be stored into four-dimensional arrays (by web page, by paragraph number, by line number and by line number segment) as Boolean, integer or string variables or URL, color and font objects at **67** and **68**. As a data field is added, changed or deleted, a determination is made at **69** on whether a value for a given high watermark needs to be changed. If so, it is updated. As a specific method in the build engine is called, a determination is made at **70** on whether a feature flag needs to be set. For example, if a particular JAVA method is called, which requires an instance of a certain JAVA Class to be executed by the run time engine, then that JAVA Class is flagged, as well as any supporting methods, variables and/or object definitions. The use of these flags is described in greater detail below in association with FIG. **25** and FIG. **27** to create a compact and efficient customized run time environment.

In one implementation, the header record, the style record, the web page record, and the object records, are carefully defined in a specific order, written in that order, and explicitly cast by object type when read by the run time engine. Exception handling can be implemented to recover from any errors. This helps assure that data integrity is maintained throughout the build and run time processes.

FIG. **9** details the polling process (**16** of FIG. **3**). The polling technology is essential for creating the necessary two-way real time communication between the JavaScript/HTML interface and the JAVA build engine. Since there is no particular difficulty for JavaScript to be able to call and pass values directly to JAVA methods, the technological challenge is to find a reasonable technique to enable JAVA to communicate back to JavaScript. The polling technology is generic, and workable across all the current browsers. The polling technology is flexible, as there are no real constraints as to what data could be communicated from the build engine to the interface, and this communication can occur at any time. The polling technology is reasonably efficient, so that the performance of the build process is not significantly affected.

In one implementation, two different techniques were utilized to implement this capability. The first was to place the build engine inside a JAVA wrapper. The JAVA wrapper accepts direct communication from JavaScript function calls, interrogates a particular JAVA build engine method, and returns that method's return value back to the calling JavaScript function. The second technique was more unconventional. A polling loop is defined in the panel's (panel **400**) JavaScript that creates a near continuous, at least from a human perception point of view, dynamic real time link, in order to monitor events occurring inside the build engine. The result is a real time retrieval (from an ergonomic perception point of view) of necessary data and status settings from the build engine back to the interface.

Upon the loading of the panel HTML file, a JavaScript function at **71** (the poller) is immediately called which initiates a polling loop. In one implementation, the polling loop is set at a poll rate of once every 100 milliseconds or less. The polling routine, operating through the JAVA wrapper, calls the build engine in order to read the current horizontal and vertical coordinates of the mouse cursor, and displays them in the panel's status fields (FIG. **37** at **450**). The polling routine also polls the build engine in order to detect whether the mouse has moved over a valid object or, by inference, whether a mouse single click, or double click event has occurred. The poller is also constantly requesting the JAVA wrapper to return the status of an error flag in order to inform the user, if necessary, of an unrecoverable error condition, and the reason for it. (See FIG. **10**). The poller then calls a panel JavaScript function that, in turn, calls the build engine to reset the flag. The poller constantly requests that the JAVA Wrapper return the status of whether the mouse cursor is over a valid object, and, if so, that object's number, type, height and width. The poller also constantly requests the JAVA wrapper to return the status of whether an object is selected, and, if so, the type and number of that selected object, as well as the objects height, width, and 3D frame state (and the point size of the object's current font if the object is a text button or paragraph object). In addition, if the object is a paragraph, the poller constantly requests the JAVA wrapper to return a flag if a double click or drag mouse event has occurred.

At **72** the polling routine detects a mouse event based on analyzing the return values received. The poller can infer that the mouse has either moved off or moved on to a valid object at **73** if the mouse over object state has changed or the mouse over object number has changed. If so, the poller updates the relevant interface objects of the panel as appropriate and displays them as necessary, and, depending upon whether the object is a text button object, a paragraph, image object, etc., at **75**, begins polling their unique values.

The poller can infer that a single click mouse event has occurred at **74** if the selection state has changed, or the selected object changed. The poller updates the menu bar

(FIG. **37** at **410**) as appropriate, making the appropriate menu commands either active or inactive. The poller also sets the necessary status variables, and, depending upon whether the newly selected object is a text button object, a text object, image object, etc., at **74**, begins polling their unique values. The poller also activates the appropriate menu choice objects inside the "Edit" menu, the "Text" menu, the "Button" menu, the "Image" menu, and the "Interactions" menu objects (FIG. **37** at **420** and **430**), depending upon whether an web page object is selected or not, which type of web page object is selected, or, if the selected web page object is a text object, whether text is marked through a drag or double click event. In a similar manner, the poller also sets the values for the interactive field objects (FIG. **37** at **460**) and the interactive drop-down list objects (FIG. **37** at **470**). More specifically, JavaScript can poll the web page object number. The value of the web page object number can be used to initialize pop-up windows with that object's web page current values, either from the panel's database or, if necessary, by interrogating the build engine's database.

The poller can infer that a double click or mouse drag operation has occurred if the flag indicating a double click or mouse drag operation is detected at **75**. The poller calls a panel JavaScript function that, in turn, calls the build engine to reset the flag. The poller then calls a panel JavaScript function to display the appropriate panel menu choices. For example, if the double click or mouse drag event occurs within a text object, then the "Text Style and "Hot Link" menu choice objects become active under the panel's "Text" menu object.

Depending on the object type (**76**), the polling technology performs various functions. If the object is a text object at **77**, the values for the paragraph style, point size, object height and width, text color, and the 3D frame status are polled and displayed. The panel's menu objects and the menu choice objects within that are active for a text object are set to the active state, and the non-text menu choice objects are set to the inactive state, which visually means they are unavailable and are "grayed out". In addition, polling can be initiated for the creation of a hot link. If the object is a text button object at **78**, the values for the text button style, point size, object height, width, text color, animation state, and 3D frame status are polled and displayed. The menu choice objects inside the panel's menu objects that are active for a text button object are set to the active state, and the non-text button menu choice objects are set to the inactive state, which visually means they are unavailable and are "grayed out". The value of the text button object string is also polled and saved in the panel's database for use when initializing relevant pop-up windows. If the object is an image object at **79**, the values for the image style, object height, width, frame color, animation state, and 3D frame status are polled and displayed. Again, the menu choice objects inside the panel's menu objects that are active for a image object are set to the active state and the non-text button menu choice objects are set to the inactive state. In addition, the results of any relevant direct object manipulation are polled and displayed.

FIG. **10** describes a two level error correction technology (**17** of FIG. **3**) employed by the build process. Initial error checking occurs during the interactions between the user and the interface with the JavaScript error checking code at **80**. Any file name, selected by the user through the file selection window or typed in a file pathname (See FIG. **6** at **49**) is checked by the panel's JavaScript to assure that it has the correct file type suffix (gif, jpg, au, etc.) at **81**.

The panel's JavaScript Code performs range checking at **82** to prevent user error or to prevent the breaking of any internal limits imposed by the build engine. These can include: going to a non-existent web page; exceeding any limit with the dual spin control (i.e. attempting to increment or decrement a point size outside of the legal range, or trying to illegally decrement a value to zero or a minus number; typing in a numeric value that is outside a legal range; and, implicitly creating an object that exceeds a limit imposed by the build engine).

The panel's JavaScript code also checks the file pathname to make sure it contains a valid address, and makes necessary additions or conversions, if necessary, at **83**. For example, if the user selected a file from the local disk, the correct URL protocol is appended to the file name in order to make it a valid string representation of a URL address. Any illegal characters for a pathname or a null file pathname entry are also caught at **83**. In addition to file pathname validity checking there are other validity checking functions that can be employed by the JavaScript at **83**. They include the attempt by the user to enter a non-numeric character into a numeric field, or leaving an essential fill-in field empty.

The panel's JavaScript then passes these values to the build engine though the arguments of a JAVA method function call at **84**. The build engine can utilize the extensive exception handling capability of JAVA at **85** (or that of any other full featured programming language used) to attempt to recover from any processing error. If recovery is not possible, the build engine sets an error flag, utilizing the polling technology (See FIG. **9** at **71**). The poller, upon detecting this flag, informs the user, for example, through an alert JavaScript pop-up message, what non-recoverable error has occurred, from which operation, and what actions, if any, the user should take. For example, if the user had selected a corrupted image file, the exception handling technology can inform the user of this fact so that user corrective action can resolve this very common problem. In one implementation, error handling and exception recovery support is provided for a malformed URL, an input or output error, a security manager violation, and a null pointer error.

FIG. **11** shows a process for text entry and text processing (**18** of FIG. **3**). The process begins when the panel's JavaScript detects the user selecting either "Button" or "Text" icon objects from the panel's tool bar or from their equvalent menu choices under the "Button" or "Text" menus, and calls the appropriate JavaScript function at **86**. The JavaScript function, after performing a range check to assure that no internal limits of the build engine are being broken, updates its database, and sets the necessary status variables. The panel's JavaScript then calls the appropriate build engine method, passing the necessary arguments, including the current board number, the internal number to be assigned to the object, the object type, and the current text button or paragraph style at **87**. The build engine then updates its internal database and sets the necessary status variables. The build engine also changes the mouse cursor shape to that of a text entry symbol. In one implementation, the mouse cursor is shaped like a crosshair, and can be moved onto the web page (the build frame **402**) at an arbitrary location.

The build engine detects a mouse click event through its "mouseDown" method at **88**. This method reports to the build engine the exact horizontal and vertical coordinates of the crosshair mouse cursor at the moment the mouse button is pressed. The build engine places these values into its internal database. The polling process is also supported, as discussed in FIG. **9**, by placing the necessary return values in the appropriate poll enabled methods.

The build engine-creates a dynamically resizable frame utilizing JAVA's "Text Area" object class, whose coordinates and size coincide with that of the draw system for the object as defined below. Other full-featured programming languages, if used by the invention, also possess similar object types. The text area is immediately overdrawn by the draw system's background paint routine. The build engine, utilizing the font metrics as defined by the selected text button or paragraph style, and utilizing the crosshair cursor's coordinates, calls the draw system. The draw system paints the background and then paints an insertion point and a selection rectangle, in the appropriate colors, and with the appropriate height and width, into the appropriate web page location at **89**. If the text button or paragraph style has a 3D frame selected, this intelligent ornamental object would also be drawn, in the appropriate color, dimensions, and thickness. Screen shot FIG. **41** shows a visualization of this process. The text insert point is in black, surrounded by a red selection rectangle, and surrounded by a blue 3D frame, as defined by the selected style. The text editor is then initialized by setting the necessary status variables.

The build engine waits until a keyboard keystroke is detected. The scan code is interpreted, and if it is a text entry key, the text editor's methods are called at **90**. The text editor processes the key event at **91**. The build engine employs frame (Text Area) processing methods and draw methods to implement the text entry and text processing functions. As a keyboard key for a text character is pressed, the build engine passes this value to the editor's text entry method, which updates both the text area's frame definition, and the draw system's database. The width of the text area is dynamically resized as necessary. If the object was a paragraph, a check is made on whether a reformat event should occur, based on the paragraph style's definition and the width of the current line's text string. If so, the appropriate text editor reformat method is called, which may cause the text area's vertical dimension to also be resized. A high watermark variable may also be set, for optimization purposes. After the final state of the text area is determined for the text entry keyboard event, the internal database for the text area, and for the paragraph or text button object, are updated. The draw system is called, and the results of the text entry event are drawn on the web page at **94**.

In one implementation, the build engine also supports the usual text processing functions found in MS Windows or Macintosh based Word Processors or Desktop Publishers at **92** and **93**. For example, if the user single clicks the mouse when over an unselected paragraph or text button object, that object is selected, a selection rectangle is drawn, the mouse cursor shape is changed to a crosshair, and the poller reports the necessary information to the panel's JavaScript. If a mouse click occurs over a selected paragraph or text button object, the editor's "Set Text Insertion Point" method is called. Based on the coordinates of the mouse cursor, and based on a calculation by the build engine as to the nearest line, and the nearest character on that line, the text insertion point can be drawn appropriately, and the necessary status variables are updated. Text entry is then processed as discussed at **91**.

If a double click or mouse drag mouse event is detected over a paragraph, an appropriate "text string selection" method is called (See FIG. **6**). Based on the coordinates of the mouse cursor, and based on a calculation by the build engine as to what text string should be selected, the internal database in updated, appropriate status variables are set, and the draw system is called for marking the text string at **94**. The polling technology is activated as discussed in FIG. **9**.

The build engine's reformat methods for paragraphs can utilize a "Clean Text Stream" model for calculating line breaks and for updating four-dimensional variables utilized by the draw system in order to draw each paragraph, each paragraph line, and each paragraph line segment in the correct location, with the correct font type, font style, font size, font effect, and background and text string color. Font style refers to a font format such as Normal, Bold, Italic, or Bold Italic. Font effect refers to style overrides such as Underline, Double Underline, Small Caps, Cross Out, Superscript, Subscript, etc. The "Clean Text Stream Model" implemented by the build engine maintains multi-dimensional array pointers and records for every paragraph line and line segment external to the text string defined within the text area. Three-dimensional and four-dimensional variables are updated after each text entry or text editing and processing event in order to assure that the pointers into the paragraph text stream, defined in the text area, are current. The three-dimensional variables that the build engine has implemented can include soft and hard line end pointers for each paragraph line. Their values can be the absolute character positions within the text area text string for that line end. Hard line breaks can be created by the user pressing the enter key. Soft line breaks can be created by a reformat method based on a calculation defined below.

The four-dimensional variables can be absolute pointers into the text area text string for the beginning and end of every style override, associated with each paragraph line segment. These style overrides can include hot links, font type, font style, font size, numerous font effects, and text and background colors. For each style override there is an associated style override record that maintains all the font and color settings for that paragraph line segment. Also positional and size data such as start and end pointers into the paragraph text stream, a left offset relative to the paragraph's left origin, a top offset relative to the paragraph's top origin, and the line height. The style override record is created when the build engine detects a mouse drag or mouse double click event within a selected paragraph. When the mouse button is initially pressed, the current paragraph line and current word on that line are calculated in a manner identical to that for calculating the location of the text insertion point on a mouse click operation. The entire word becomes one anchor for the paragraph line segment, while the word defined by the mouse coordinates when the mouse button is released becomes the other anchor. Up to two other paragraph line segments can be implicitly created by the word oriented selection method. If there is text to the left of the first anchor word, and that paragraph line had not previously had a style override defined in it, the text string from the beginning of the paragraph line to the first anchor point has a style override record created for it. The values are set to that of the underlying paragraph.

If style overrides had already been created on that paragraph line, and the anchor word is inside one of them, then that style override's end pointer is adjusted to the start of the anchor word. All other style overrides, if any, to the right of the anchor word are deleted, as overlapping style overrides are not permitted. In a similar manner, the text string, if any, to the right of the last anchor point, up to the line or paragraph end, can also be defined as a style override. If a mouse click occurs before a "text style" operation, then these pointers will be reset. If the panel's JavaScript detects a user selection of "text style" from the "Text" menu, the appropriate pop-up window is drawn and its values initialized from the JavaScript database. Upon detecting a user completion event (i.e., the depressing of the enter key), the

panel's JavaScript database is updated and a call is made to an appropriate build engine method, with the necessary data and status information passed as function call arguments. The build engine updates its internal database and calls the reformat method if necessary. The draw system utilizes these four-dimensional variables in order to paint the paragraph line segment style override.

The calculation for the creation or updating of a soft line break begins with the maximum paragraph width, which is set at a percentage of the browser screen width. This percentage is converted to an absolute pixel number based on the web designer's screen resolution. When any text entry or text editing and processing event occurs, a build engine method is called which calculates the width, in pixels, for the current paragraph line, based on the character string in the text area that exists between the previous line end pointer and the current line end pointer. The font definition(s) that are related to this character string are applied, and a string width is calculated. If the string width exceeds that of the maximum paragraph width, an "OverFlow" reformat method is called. The overflow reformat method calls a method to determine the settings for the last word on that line, and that word overflows to the following paragraph line. All pointers for the current line, and subsequent lines are updated as necessary, as are all pointers and records to paragraph line segments. If the string width is less than that of the maximum paragraph width, and the text processing operation was not text entry, then an "UnderFlow" Reformat method is called. The underflow reformat method calls a method to determine the width, in pixels, for the first word on the next line. If that word will fit on the current line it is placed there. As before, all pointers for the current line, and subsequent lines are updated as necessary, as are all pointers and records to paragraph line segments. The word oriented selection technique, and the reformat, database, and draw technologies that support it, greatly simplify the text editor and produce a run time engine that is smaller, faster and more reliable.

FIG. **12** shows the operation of the image processing technology utilized by the build engine (**19** at FIG. **3**). The process begins when the panel's JavaScript detects the user selecting the "Image" icon from the panel's tool bar or the comparable menu choice under the "hnage" menu. The appropriate JavaScript function is called at **95**, which draws the define image pop-up window. The user then selects an image from the file selection window with the browser, types in the image pathname for the image file on the local disk, or types in the URL for the image that exists on a server. The user could also define a 3D frame for the selected image at this time. Screen shot FIG. **49** shows a visualization of a collage for the define image pop-up window and the user's selection choices under each tab setting. The user can complete the selection process by either pressing the Enter Key or clicking on the "Create Image" icbn in the pop-up window. If the Enter Key is pressed, the pop-up window's JavaScript Code utilizes the onKeyDown function, or if a mouse click, the onClick function, as described in FIG. **7**, to recognize the completion event. An appropriate error checking JavaScript function is called, which performs a file name error check, a filename validity check, and a range check to assure that no internal limits of the build engine are being broken. If the error checking tests are successful another JavaScript function is called to update the panel's database, and set the necessary status variables.

The panel's JavaScript then calls the appropriate build engine method, passing the necessary arguments, including the current internal web page number, the internal number to be assigned to the image object, the object type, and the current image style at **96**. The build engine then updates its internal database and sets the necessary status variables. It also changes the mouse cursor shape to that of an "Image Create" symbol. In one implementation, the mouse cursor is shaped like an arrow. The build engine detects a mouse click event through its "mouseDQwn" Method at **97**. This method reports to the build engine the exact horizontal and vertical coordinates of the arrow mouse cursor at the time the mouse button was pressed, and places these values into its internal database. The polling process is also handled, as discussed in FIG. **9**. The build engine then asserts the necessary security permission for reading from the local disk, if necessary, and attempts to create the necessary image object at the current mouse coordinates at **98**, while checking for any exception conditions as described in FIG. **10**. If no unrecoverable exceptions are reported, the internal database is updated and the draw system is called.

The image processing technology supports direct web page image object interactions at **99**, utilizing the communication technology described in FIG. **6**. The build engine first processes the mouse event as described in FIG. **7**, and places the appropriate values into a poll enabled JAVA method as described in FIG. **9**. There are two types of direct web page image object interactions. s The first occurs by simply selecting the image object with a single mouse click. A red selection rectangle is drawn around the image, as are eight attachment points. When the user has pressed the mouse cursor, the mouse cursor's shape changes to that of an anchor, which is a symbol that can be used when dragging or moving an object. The mouse's location will jump to the origin for the image. In an alternative implementation, the anchor can be defined by the mouse location at the time of the mouse drag operation. In either case, while the mouse is being dragged, the build engine updates its internal database. The build engine also updates its poll-enabled methods for communication with the interface's polling technology at **100**. The JavaScript poller reads these values, updates the panel JavaScript database, and updates the panel's interface objects. In a similar way, placing the mouse cursor over an attachment point and dragging will result in an image resizing operation. Screen shots FIG. **50** through FIG. **52** show a visualization of an image dragging operation. Screen shot FIG. **50** shows the cursor over an unselected image. Screen shot FIG. **51** shows the screen state after the left mouse button has been pressed. Notice that the image is now selected and the cursor shape has changed to the drag state. Screen shot FIG. **52** shows the screen state after the mouse has been dragged to the northwest. Notice that the image stayed selected and moved with the mouse. Screen shots FIG. **53** and FIG. **54** show a visualization of an image resizing operation for a normal image. Notice that all eight-attachment points are drawn and active for the selected image. Screen shot FIG. **53** shows the cursor over the upper left attachment point. Notice that the cursor shape has changed to a northwest to southeast resize cursor shape. Screen shot FIG. **54** shows the result after the left mouse button has been pressed over the upper left attachment point and dragged to the northwest. Notice that the image's upper left corner is still under the cursor, the image has resized, and the cursor shape remained unchanged. For image resizing operations with the mouse over and mouse down objects, only the east, southeast, and south attachment points are drawn and active.

The second type of direct web page image object interaction occurs when the panel's JavaScript code detects that the user has selected an image object interaction feature from the panel's "Image" menu. The appropriate JavaScript function is called, which sets the necessary status variables, and then calls the appropriate JAVA method, passing the necessary values as arguments. The JAVA method then sets its necessary status variables, changes the mouse cursor shape as appropriate, depending upon the type of direct image operation, and awaits a direct mouse operation on the image object. Image rotation is an example of this type of direct image interaction. In one implementation, direct image object rotation is realized by utilizing the image rotation technology described in association with FIG. **33** below. Screen shots FIG. **55** and FIG. **56** show a visualization of an image rotation for a normal image. Screen shot FIG. **55** shows the user selecting the rotate command from the "Image" menu. Immediately the cursor's shape changes to the rotate (a dual left/right arrow) cursor, and the selected image's attachment points disappear. Placing the cursor on the image and dragging will cause the image to rotate on an east/west and/or north south axis. Screen shot FIG. **56** shows the result after the mouse was dragged on an east/west plane.

Image object interactions are invoked by selecting from the JavaScript panel, selecting from a JavaScript pop-up window, and by selecting from a JAVA window object at **101**, as described in FIG. **6**. The initial values in the pop-up window are set from JavaScript's database. After any user interaction, JavaScript's database is updated and the appropriate method in the build engine is called with the necessary settings. The build engine, after updating its internal database, calls the appropriate image processing method. The image processing routine then calls the required image filter(s), which then perform the necessary processing on the image bitmap at **102**.

An image filter is a body of code, usually consisting of one or more digital image processing algorithms, which operate on an image bitmap, and create a transformed image bitmap. An image observer can be invoked by the image filter, which then reports when the image bitmap processing has been completed. An image observer is a independent process that monitor's a particular image processing event, such as the execution of an image filter or the reading in of an image file, and reports the status of that process when queried. When the image observer reports a successful completion, the image filter can call the build engine's draw system to display the transformed image bitmap. This interaction between the build engine's image processing method, the image filter(s), the image observer, and the draw system can occur many times, depending upon the image processing operation chosen. Image animations and image transformations, which are technologies that rely heavily on image filters, and the image observer are discussed in greater detail below in association with FIG. **16** and FIG. **17**.

FIG. **13** shows a process for implementing text button, image and paragraph style settings (**20** of FIG. **3**). The initial values for all the settings inside a parent pop-up window and associated child pop-up windows, for a particular style, can be set from the JavaScript database at **103**. The settings can include: image object styles, text button object styles and paragraph object styles.

The following settings can be initialized and changed for image object styles. The following settings are initialized for all image object states (Normal, mouse Over, mouse Down) and can be changed:

(1) resize factor.

(2) rotation factor.

(3) main animation type, speed, number of animation steps (resolution) and number of cycles.

(4) image processing factors. (brightness, contrast, etc.)

(5) 3d effects and their color values.
(6) web page centering attribute.
(7) web page scaling attribute.

b) The following actions are initialized and can be changed.
(1) sound effects and audio channels.
(2) video files and video channels
(3) text button and image pop ups and their attributes (See 1.a above and 2.a below.)
(4) click events.

c) The following transformation settings are initialized and can be changed.
(1) the initial delay
(2) up to three transformations can be defined with the following settings:
(a) which image states should the transformation be from and into.
(b) the speed of the transformation.
(c) any delay before the next transformation.
(3) whether the transformation(s) should occur simultaneously with the enter and exit time line animation or after the enter and before the exit animations.

d) The following time line settings are initialized and can be changed.
(1) the initial delay before the image object's appearance.
(2) the enter animation type, speed, and animation resolution.
(3) the delay after the enter animation and the main animation.
(4) the exit animation type, speed, and animation resolution.
(5) the initial delay, after the entrance of the parent object, before the child text button and image object's appearance(s).
(6) the child object(s) enter animation type, speed, and animation resolution.
(7) the delay after the child object(s) enter animation.
(8) the child object(s) exit animation type, speed, and animation resolution. The following settings can be initialized and changed for text button object styles.

e) The following attributes are initialized for all text button object states (normal, mouse over, mouse down) and can be changed:
(1) all font specifications.
(2) vertical state.
(3) all color specifications.
(4) 3d effects and their color values.
(5) web page centering attribute.
(6) font processing attributes (available in java 2)
(7) scale, shear, and rotate (available in java 2)

f) The following actions are initialized and can be changed.
(1) sound effects and audio channels.
(2) video files and video channels
(3) text button and image pop ups
(4) click events.

g) The following transformation settings are initialized and can be changed.
(1) the initial delay
(2) up to three transformations can be defined with the following settings:
(a) which image states should the transformation be from and into.
(b) the delay before the next transformation.
(3) whether the transformation(s) should occur simultaneously with the enter and exit time line animation or after the enter and before the exit animations.

h) The time line settings are the same as those defined for image objects. They also are initialized and can be changed.

The following settings can be initialized and changed for paragraph styles.

The following attributes are and can be changed:

i) all font specifications.

j) all color specifications.

k) 3d effects and their color values.

l) web page centering attribute.

m) the look of hot links, including the text and background colors when the link is active and when the mouse is over the link.

The reference to JAVA 2 under text button object styles refer to the most recent version of JAVA released by Sun Microsystems. This version supports a far more robust two-dimensional processing capability than JAVA 1.6, including significant font processing capabilities and the scaling, shearing, and rotation of objects. Currently, most conventional browsers only support JAVA 1.6. Provisions are made in the invention so that as the then popular browsers support more robust versions of programming languages, those new capabilities can be employed to further enhance the capability of the invention.

Referring again to FIG. **13**, upon detecting the completion of editing an image, text button or paragraph style, the panel's JavaScript calls a build engine method and passes the required values. The build engine updates its internal database and sets any necessary feature flags at **104**. When an image, text button or paragraph object is created, all the style settings for the currently selected style are applied by the build engine as part of the definition for the newly created object at **105**.

If a style is changed, all objects on all internal web pages that are utilizing that style are candidates for being changed to those new values at **106**. Flags are kept for every possible style setting for each object. If a given object is edited through the text button, image, or interaction menus or other interface objects of the panel **400**, the flags are set for any setting that are changed. If that style is subsequently changed, only those settings that have not had their flags set will be changed for any given object.

FIG. **14** describes the video and audio file and video and audio channel processing techniques employed by the build engine (**21** of FIG. **3**). A user can select a video or audio special effect (i.e. user input is provided at **107** that indicates a video or audio special effect). The method for activating a video file or video channel is defined in the text button and image object "mouse over" interactive pop-up windows described later at FIG. **16**. Methods for defining a video object as a pop-up, or a frozen object, are described with reference to the text button and image object "mouse down" interactive pop-up window also described at FIG. **16**. Audio files and audio channels can be defined in both the "mouse over" and "mouse down" interactive pop-up windows also described at FIG. **16**. The pop-up or a frozen object settings for audio are also set in the object "mouse down" interactive pop-up windows discussed therein.

As before, the panel JavaScript code initializes any pop-up windows (where the initial values are set from the JavaScript database), captures a file or channel name (from the user input) and performs file and validity error checking upon detecting a user completion action at **108**. The build engine is then called, receiving the necessary data and status as function call arguments. The build engine determines if the audio and video definition is a file pathname or the URL of a live channel at **109**, and thereafter initiates its exception handling. If the video or audio definition is a file, the build engine performs the relevant file exception handling checks, and asserts the necessary security permissions. If there were no errors, or the exception handling error was recoverable, the build engine reads and links the video/audio file to the database, and plays the file for user verification at **110**. If the video or audio definition was a channel, the necessary pointers are updated in the database, and methods are assigned for efficient transmission, at run time by the run time engine, at **111**. The ability of the run time engine to play multiple synchronized audio and video files and channels simultaneously will be described at FIGS. **31–35**.

FIG. **15** describes the frames, tables, forms and draw objects technologies employed by the build engine (**22** of FIG. **3**) in one implementation of the invention. When the panel JavaScript code detects a user action to create a "frame", "table", "form" or "draw object" from an appropriate panel interface object, it draws and initializes the appropriate pop-up window at **112**. Upon detecting a user completion action by the pop-up window's JavaScript code, a panel JavaScript function is called to perform the necessary error checking and updating of the panel's database. Panel JavaScript thereafter calls the appropriate build engine method(s) passing the necessary data and status values as function call arguments at **113**.

The build engine updates its internal database, sets the necessary status values, and initializes, as necessary, appropriate methods for run time processing. In one implementation, the build engine includes definitions to map a given object into a relational database. Also available are a full array of database operations. Support for popular databases (such as Oracle, Informix, Sybase and DB2) are available on a real time interactive basis.

The run generation technologies, as described later in FIGS. **24–27**, are also implemented for a given frames, table, form and draw object at **114**. The run time technologies, as described later in FIGS. **28–36**, are also implemented for a given frame, table, form and draw object at **115**. FIG. **16** describes the user interaction settings and technology employed by the build engine (**24** of FIG. **3**). Depending upon the type of object currently selected at **116** (if no object is selected no user interaction choices will be available) the panel JavaScript Code draws an appropriate pop-up window. If the selected object was a text buttoh object at **117**, or an image object at **119**, both "mouse over" and "mouse down" choices will be available from the panel's "Interactions" menu. If the selected object is a paragraph, user interaction definitions can be activated by a double click or a mouse drag event being detected by the build engine at **118**.

More specifically, appropriate values are set in a poll-enabled JAVA routine. The JavaScript poller reads the values, and draws the appropriate panel menu choices. The "Text Style", "Hot Link", "Preferences" and "Format" pop-up windows can be chosen. If the hot link choice under the panel's "Text" menu is selected and executed, the hot link definition for internal or external web pages is captured by an appropriate JavaScript function and file pathname error and validity checking is performed. If either the "Text Style", "Hot Link", Preferences" or "Format" choices under the panel's "Text" menu are selected, the panel's JavaScript draws the appropriate pop-up window. Upon detecting a user completion event, the panel's JavaScript reads the values in the pop-up window and passes the font specification parameters to an appropriate build engine method as function call parameters. The build engine then processes this data, calls

a reformat method, updates its internal database, and sets the necessary four-dimensional variables for communication with the draw system.

The normal and "mouse over" foreground and background colors for the hot link, which were defined in a link look pop-up window (available under the "Text" menu of the panel), are utilized by the build engine to draw the hot link. The build engine performs the necessary exception handling, and then updates its internal database.

Based on the panel's JavaScript Code detecting whether the user chose the "mouse over" or "mouse down" choice under the "Interactions" menu at **120**, as well as based on whether an image or text button object is currently selected, the panel's JavaScript code draws the appropriate pop-up window. Initial values for the pop-up windows are set from the panel's database at **121** and **122**. In one implementation, the following user interaction's for the "mouse over" and "mouse down" states for text button and image objects are supported:

1: 3D Frame, in a specified color, and selected for a specified 3D appearance, can be defined for text button and image object's "mouse over" and "mouse down" states, as well as for their text, image and video pop-ups.

2: The font typeface, font style, font size, font effect(s), text color, and text background color can be defined for a text button object's "mouse over" and "mouse down" states, as well as for the text pop-ups associated from both text button and image objects.

3: Text, image, and video pop-ups can be defined for the text button and image object's "mouse over" state.

4: A sound track (file) can be defined for the text button and image object's "mouse over" state with the following choices:
  a. play once when a "mouse over" event occurs.
  b. play until a click event while on the object.
  c. play until the mouse moves off the object.

5: A sound track (file) can be defined for the text button and image object's "mouse down" state with the following choices:
  a. play once when a mouse click event occurs when over the object.
  b. play until a second click event while on the object.
  c. play until the mouse moves off the object.

6: Both video and sounds can be defined as channels as well as files.

7: The text, image, and video pop-ups can be frozen (i.e. not disappear when the mouse moves off the object after a mouse click event, for both text button and image objects).

8: Text button and image objects can have one of the following click events defined:
  a. go to a specific internal web page.
  b. go to the next internal web page.
  c. return to the parent (calling) web page.
  d. go to an external web age. That web page will replace the current web page.
  e. go to an external web page. That web page will be launched into a new window so that both web pages will be visible and accessible.

After a user completion action is detected, the panel JavaScript code performs the necessary file error and validity checking, updates its database and sets necessary status values, and then calls the appropriate build engine method, passing the necessary data values and status as function call arguments at **123**. The build engine updates its internal database, sets the necessary status variables, then draws the appropriate "mouse over" or mouse down" text button or image object states. The build engine also plays the sound or video file for user verification. The run time technology behind the user interactions will be described in greater detail in association with FIG. **36**.

FIG. **17** describes the image and text button object animation settings and technology employed by the build engine (**25** of FIG. **3**). The panel's JavaScript code determines which type of object, and which object number, from the currently selected object, as reported by the poller at **124**. When the panel's JavaScript detects a user selection of "Define Image" or "Animate" from the panel's "Image" menu, or a user selection of "Define Button" or "Animate" from the panel's "Button" menu, it draws the appropriate pop-up window and initializes the pop-up window's values from its database at **125** and **126**. Screen shot FIG. **57** shows a visualization of one implementation of the "Text Button Animation Specifications" pop-up window and the animation settings available to the user. Screen shot FIG. **58** shows a visualization of one implementation of the "image animation specifications" pop-up window and the animation settings available to the user.

When a user completion event is detected, the panel's JavaScript code captures the values from the pop-up window for the animation type, speed, resolution, and number of animation cycles at **125** and **126**, respectively, and updates its database at **127**. The panel's JavaScript then calls the appropriate build engine method, passing the necessary data and status values as function call arguments. The build engine updates its internal database and sets the necessary feature flags (See FIG. **8**.) Linkage to the appropriate animation method(s) is also set.

A thread object (a thread is an independent asynchronous program that is multiprogrammed with other threads, are defined and executed by the invention, by a JAVA Virtual Machine and by the browser) is created and executed for user verification at **128**. Values are set to integrate the given animation thread with the object time line technology (See FIG. **19**). Values are set at **129** so that when the thread object is invoked by the run time engine, the appropriate image filter(s) and animation methods are called. The run time technology behind image and text button object animations is described in greater detail in association with FIG. **31** through FIG. **35**.

FIG. **18** describes the transformation settings and technology utilized by the build engine (**26** of FIG. **3**). A transformation is defined as the changing of an object from one state to another based on a timer control, subject to user settings. In one implementation, the available states for text button and image objects are their "normal", "mouse over", "mouse down" and "pop-up" definitions. For text button objects, a transformation is implemented as the instantaneous drawing of one object state while erasing the previous object state. For images, a transformation is the gradual fading out of the previous object state, while, simultaneously, fading into the next object state.

Prior to any user menu selection, the panel's JavaScript code already knows the status of any selected object through the poller mechanism (**124** of FIG. **17**). This includes what type of object and the object's internal identifying number. When the panel's JavaScript detects a user selection of "Transform" from the panel's "Interactions" menu, it draws an appropriate pop-up window and initializes the pop-up window's values from its database at **130**. Screen shot FIG. **59** shows a visualization of one implementation of a "define the transformation for the text button object" pop-up window and the transformation settings available to the user. Screen shot FIG. **60** shows a visualization of one implementation of a "define the transformation for the image object" pop-up window and the transformation settings available to the user. When a user completion event is detected, the panel's JavaScript Code captures the values from the pop-up window based on the object type.

In one implementation, the following settings for text button objects can be specified:

1. The initial delay.
2. Up to three transformations can be defined with the following settings:
   a. Which image states should the transformation be from and into.
   b. The delay before the next transformation.
3. Whether the transformation(s) should occur simultaneously with the enter and exit time line animation or after the enter and before the exit animations.

In one implementation, the following settings for image objects can be specified:

1. The initial delay.
2. Up to three transformations can be defined with the following settings:
   a. Which image states should the transformation be from and into.
   b. The speed of the transformation.
   c. The resolution of the transformation.
   d. Any delay before the next transformation.
3. Whether the transformation(s) should occur simultaneously with the enter and exit time line animation or after the enter and before the exit animations.

The panel's JavaScript updates its database at **131**. The panel's JavaScript then calls the appropriate build engine method, passing the necessary data and status values as function call arguments. The build engine updates its internal database and sets the necessary feature flags (See FIG. **8**.) Linkage to the appropriate transformation method(s) is also set.

A thread object is created and executed for user verification at **132**. Values are set to integrate this transformation thread with the object time line technology (See FIG. **19**). Values are set at **133** so that when the run time engine invokes the thread object, the appropriate image filter(s) and transformation methods are called. The run time technology behind image and text button object transformations is described in greater detail below in association with FIG. **31** through FIG. **35**. FIG. **19** describes the text button and image time lines and technology utilized by the build engine (**27** of FIG **3**). A time line is an independent asynchronous process that defines the existence of a given text button or image object. An object's time line begins at the time a given web page makes its appearance, either through an immediate draw or through a transition animation. In one implementation, an object time line can be created as an instance of a class, which has a threadable interface. This instance has its own data structures, which define the animations, and transitions associated with the time line definition. An image or text button object time line can spawn child time lines, at a designated moment. A complete description of time line technology, and how they integrate the animation and transformation technologies, will be described below in association with FIG. **31** through FIG. **35**.

The build process begins the time line definition process by having the panel's JavaScript determine what is the currently selected object, utilizing the polling technology at

134. That is, values for the object' appearance time, animation type, speed and resolution are captured. When the panel's JavaScript detects a user selection of "time line" from the panel's "Interactions" menu, it draws the appropriate pop-up window and initializes the pop-up window's values from its database. Screen shot FIG. **61** shows a visualization of a collage of one implementation of a "define the time line for the text button object" tabbed pop-up window and the time line settings available to the user under each tab. Screen shot FIG. **62** shows a visualization of a collage of one implementation of a "define the time line for the image object" pop-up window and the time line settings available to the user under each tab. When a user completion event is detected, the panel's JavaScript captures the values from the pop-up window based on the object type. The currently available settings, for both text button and image objects, are:

1: The initial delay before the image object's appearance.

2: The enter animation type, speed, and animation resolution.

3: The delay after the enter animation and the main animation.

4: The exit animation type, speed, and animation resolution.

5: The initial delay, after the entrance of the parent object, before the child text button and image object's appearance(s).

6: The child object(s) enter animation type, speed, and animation resolution.

7: The delay after the child object(s) enter animation.

8: The child object(s) exit animation type, speed, and animation resolution.

The panel's JavaScript updates its database at **135**. The panel's JavaScript then calls the appropriate build engine method, passing the necessary data and status values as function call arguments. The build engine updates its internal database and sets the necessary feature flags (See FIG. **8**.) A build engine method then processes all the data related to this object. The object's animation settings, if any, are integrated into the timeline at **136**. The object's transformation settings, if any, are also integrated into the timeline. If an image object, any transformation animation may be executed simultaneously with the appearance and/or exit animations, depending upon the settings. Finally, a multi-level object thread definition is created and executed for user verification. Values are set at **137** so that when the run time engine invokes the thread object, the appropriate image filter(s), animation methods, and transformation methods are called.

FIG. **20** describes the web page transition animations, time line settings and technology utilized by the build engine (**28** of FIG. **3**). When the panel's JavaScript detects a user selection of "Define" from the panel's "Webpage" menu, it draws the appropriate pop-up window and initializes the pop-up window's values for the current web page from its database at **138**. Screen shot FIG. **63** shows a visualization of one implementation of the "define the current web page settings" pop-up window and the web page settings available to the user. In the implementation shown, the choices supported include:

1: The web page delay time (which is the delay, after the completion of the last object time line, to the loading of the next web Page).

2: The transition animation, which can include a random animation choice. This is the animation applied to the web page when it is loaded and to the previous web page as it departs.

3: The number of animation frames per second, which effectively is the resolution of the transition animation.

4: The number of animation frames, which effectively defines the time expected for the transition animation to complete.

5: The web page's background color. This setting will override the generic setting for the web site, defined in FIG. **21***a*.

6: A web page boarder. This boarder, if selected, will also override the setting for the web site, defined in FIG. **21***a*. The boarder can be drawn with a 3D effect, taking the background color, and applying a transformation so that, to the human eye, a lighter and darker shade of that color will be drawn appropriately to create a 3D effect.

7: The web page's background pattern. This setting will override the generic setting for the web site, defined in FIG. **21***a*.

The panel's JavaScript updates its database at **139**. The panel's JavaScript code then calls the appropriate build engine method, passing the necessary data and status values as function call arguments. The build engine updates its internal database and sets the necessary feature flags (See FIG. **8**.). The web page time line is synchronized with its object time lines by an appropriate build engine method at **140**. The web page's appearance delay and transition settings are integrated into the web page time line. Thereafter, a single-level object thread definition is created. Values are set at **141** so that when the thread object is invoked by the run time engine, the appropriate animation methods and object time line threads are called. Again, the run time technology behind web page transition animations and web page time lines is described in greater detail below in association with FIG. **31** through FIG. **35**.

FIG. **21***a* describes the file operations supported by the build engine (**29***a* of FIG. **3**). In one implementation, the file operations supported include:

1: "Save" at **142** or "Save As" at **143**. If the selection from the panel's "File" menu is to "Save" as a web page, the current browser screen height percentage value is sent to the build engine. The build engine updates its internal database and the build process is completed. Thereafter, the run generation process is executed. (See FIG. **24** through FIG. **27**.) If the selection is to "Save As" a template for the run generation process is also executed but the generated files are placed in the template directory. If the selection is to save as a banner or custom application, those absolute screen dimensions are sent to the build engine and its internal database is updated and the run generation process is executed.

2: "New" at **144**. A test is made by the panel's JavaScript code to see if any user input has been processed and not saved. If so, the user is asked whether this data should be saved. If so, and if the selection is to "Save" as a web page, the build process is completed and the run generation process is executed as described above. If the selection is to "Save" as a template the run generation process is executed but the generated files are placed in the template directory as described above. The panel's JavaScript code then reinitializes its database and calls a build engine method that reinitializes the build engine database.

3: "Close" at **145**. A test is made by the panel's JavaScript to see if any user input has been processed and not saved. If so, the user is asked whether this data should be saved. If so, and if the selection is to "Save As" a web page, the build process is completed and the run generation process is executed. If the selection is to "Save As" a template the run generation process is executed but the generated files are placed in the template directory. The panel's JavaScript then terminates the build process.

4: "Open" at **146**. A test is made by the panel's JavaScript to see if any user input has been processed and not saved. If so, the user is asked whether this data should be saved. If so, and if the selection is to "Save As" a web Page, the build process is completed and the run generation process is executed. If the selection is to "Save As" a template the run generation process is executed but the generated files are placed in the template directory. The panel then initiates the dynamic web page resizing technology as described in FIG. **22** below for the open re-initialization mode.

5: "Apply" at **147**. A template is applied to the existing web site that is being processed by the build engine. The web page and style record definitions of the template replace those of the existing web site. The web page objects of the template are added to the web page objects of the existing web site.

6: "Web Site at **148**. The web designer can define settings that will be applied to all web pages in the web site. In one implementation, the web site applications supported include:

a: web page. The web page height can be set, as a percentage, larger than the browser window for long vertically scrolled web pages.

b: Standard banner sizes.

c: Custom. (The user can define any arbitrary web page size and resolution) Screen shot FIG. **63** shows the generic web site setting choices presented to the user in one implementation of the invention.

FIG. **21***b* describes the view operations supported by the build engine (**29***a* of FIG. **3**). In one implementation, the file operations supported include:

1: "Normal" at **149***a*. This is the default file mode in which the interface and the build engine are processing user input as described in FIG. **5** through FIG. **23** above.

2: "Preview" at **149***b*. The build engine runs the web site off its internal database. The web site will perform in an identical manner as if it had gone through the entire run generation and run time process, but it is being executed on the web designer's computer.

3: "Play" at **149***c*. The build engine runs the web site off an external database in a separate browser window. The web site will perform in an identical manner as if it had gone through the entire run generation and run time process, but it is being executed on the web designer's computer.

4: "Zoom" at **149***d*. The dynamic web page resizing technology (see FIG. **22** below) is first executed. When the engine is fully reinitialized, and the engine has gone to the current web page, the page and all its objects are drawn to the scale as defined by the zoom level. All object coordinates and sizes are automatically scaled appropriately because they are always defined with virtual screen values, even when the web page is being draw in the "normal" view.

FIG. **22** describes the dynamic web page resizing technology supported by the build engine. If a user selection of the "Open" command from the "File" menu is detected by the panel at **500**, the panel calls an engine method to read selected contents from that web site's external database file.

In one implementation of the invention at **506**, the engine reads the web page width and length fields, as well as the background color or background image definition for the first web page of the Web Site. The engine then creates a build engine HTML definition file based on these specifications, and writes this file either to the local disk or the server, depending upon the origination of the build tool.

At **502**, if a user completion event occurs inside the web site JavaScript pop-up window, which had been activated when the user selected the "Web Site" command from the "File" menu, the panel determines if the web site page size has been changed. If so, the panel calls an engine method for processing.

Similarly, at **504**, if a user selection of a "Zoom" command from the "View" menu is detected by the panel at **504**, the panel also calls an engine for processing.

In both the cases at **502** or **504**, in one implementation of the invention, the engine writes out a checkpoint record at **508** that is similar to that of a "Websitename".dta "database file (See FIG. **24**). But is given the temporary checkpoint Websitename. The engine then creates a build engine HTML definition file based on these specifications, and writes this file either to the local disk or the server, depending upon the origination of the build tool.

In one implementation of the invention at **510** the engine terminates itself, by stopping all of its threads. Meanwhile the interface writes out four cookies onto the local disk which define the following:

1: The re-initialization mode. (Either Open or Checkpoint).

2: The current web page number when the resizing event occurred.

3: The Web Site Name. (The checkpoint name if in checkpoint mode)

4: The zoom level.

The interface then terminates itself by executing the JavaScript "parent.location.href" command, which causes the build engine parent HTML frame file (PFF) to be reloaded (See FIG. **5**).

In one implementation of the invention at **512** the re-initialization process begins. The PFF cause both the panel and the build engine to be reloaded and activated. The panel then reads the mode cookie. If the mode is either open or checkpoint, the interface reads the web site name, page number and zoom level cookies, then resets the mode cookie to the initialize state for subsequent operations. The interface then calls an engine method to read the external database, and then to return the necessary values from that database in order to update the interface's database. Finally the engine calls two engine methods in order for the engine to go to the correct current web page and draw that page at the now current zoom level. Normal processing can then resume.

Run Generation Process

FIG. **24** through FIG. **27** describe the run generation process. This process starts when the build process detects that the user is finished defining the web site (user has saved the web site and invokes the run generation process), and concludes with the actual uploading of all the necessary web site run time files to the user's server.

FIG. **24** describes the techniques employed by the build engine for the creation of the external database, and the security and optimization techniques that support this process (**30** of FIG. **4**).

When the panel's JavaScript Code detects a user selection of "Save" or "Save As" from the panel's "File" menu, it draws the appropriate pop-up window and initializes the pop-up window's values for the current web page size as had been defined at FIG. **5** and passed to the build engine. The panel's JavaScript in the "save the web page/template" pop-up window detects a user completion event at **150** (i.e., the designation of a user's web site name followed by the enter key), and calls the appropriate panel JavaScript function. More specifically, after completing the appropriate validity checks, the function calls the appropriate JAVA build engine method, passing as a function call argument the user defined "Websitename". The build engine method checks for the existence of a "Websitename".dta file, and, if so, posts that result into a poll-enabled method return value. The poller checks that value, and if set to true, calls a JavaScript function which draws a pop-up window asking the user to confirm whether the existing web site definition should be overwritten or not. This JavaScript function also calls an appropriate build engine method to reset the return value to false in order to be initialized for the next possible "Save" operation.

Once this verification process is completed the build engine begins the external database creation process at **151**, which will vary depending upon the security manager of a given browser at **152**. See FIG. **5** for a detailed description of the browser security manager alternatives. If the browser's security manager allows for local disk file creation, the build engine calls a method, which asserts the necessary security policy permissions to create and write a file. If not, the build engine calls the necessary method to create and write a file on the user's server.

The external database contains, as its first record, a "Header" record, which contains can include the following information:

1: A file format version number, used for upgrading database in future releases.

2: The default screen resolution, in virtual pixels, for both the screen width and height. This is usually set to the web designer's screen resolution, unless overwritten by the user at FIG. **5**.

3: Whether the application is a web site.

4: Virtual web page size settings. A calculation is performed by the build engine method, in order to calculate what the maximum web page length is, after reformatting all paragraphs on all internal web pages, based on the default screen resolution.

5: Web page and styles high watermarks.

6: The Websitename.

The header records are written at step **153**.

During the build process, as new web pages or new objects are created by the user, or as text is added to or deleted from a paragraph, or as new styles are created or deleted, appropriate high watermarks are set, in order to show the current number of each of these entities. Thus, at **154**, the values for the number of active web pages and the number of text button, image, paragraph or other styles are written as high watermarks in the header. The high watermarks for the number of text button, image, paragraph or other objects that exist for each web page, the number of lines for each paragraph object, and the number of line segments for each paragraph line are written within the body of the external database as described at **156**, and used as settings for each of the loops in the four-dimensional data structure. Because no structural limits are set on the number of web pages, objects per web page, styles, or paragraph size, these high watermarks greatly reduce the external database file size, and the time it takes for the run time engine to process the data stored in its database.

The settings for all paragraph, text button and image styles are then written as a style record at **155** based on their high watermark. This data includes Boolean and integer variables, and font and color objects, written as a one-dimensional array, based on the high watermark values for the number of styles that exist. The font and color objects are serialized as is discussed in greater detail below (See **159** below).

The body of the external database is then written at **156**. All Boolean values are written inside a four-dimensional loop at **157**. The outside loop contains the Boolean values used to define web pages (i.e. a one-dimensional array definition) as well as the high watermarks for the number of text button, image, paragraph or other objects per web page, with the loop set at the high watermark which defines the number of existing web pages for this web site structure. The second level consists of three or more two dimensional loops with the loops set to the high watermarks defining the actual number of text button, image, and paragraph or other objects that appear on any given web page and contains the values used to define web page objects ((i.e. a two-dimensional array definition; web page number by object number). Included within the loop for paragraph objects are the high watermarks for the number of lines for each paragraph object. The third loop is set by the high watermark defining the actual number of paragraph lines that for all paragraphs on any web page and contains the values used to define paragraph lines (i.e. a three-dimensional array definition; web page number by object number by paragraph line.) Included within the loop for paragraph lines are the high watermarks for the number of line segments for each paragraph line. The inner most loop is set by the high watermarks defining the number of line segments per paragraph line and contains the values used to define paragraph line segments (i.e. a four-dimensional array definition; web page number by object number by paragraph line by paragraph line segment.).

All integer values are written inside a four-dimensional loop at **158**. Their four loops are controlled by the same high watermark settings as used for the Boolean records, and they describe the same logical entities.

Multimedia objects are written inside a two-dimensional loop at **159**. They include URL, color, and font objects, and can include other types of objects. A URL object is the encoded form of a URL Address, used by a browser or a JAVA method to access files and web addresses. All multimedia objects must be serialized before they can be written. This means that the objects are converted into a common external definition format that can be understood by the appropriate deserialization technique when they are read back in and cast into their original object structure. The outside loop contains web page related objects, and the inner loop contains image, text button, paragraph, etc. related URL, color, and font objects. The outer loop is defined by the web page high watermark and the inner loops by the high watermarks for the actual number of text button, image, paragraph or other objects on a web page.

String records are written inside a four-dimensional loop at **160**. The outer loop may be empty. The second loop can include the string values for text button objects, audio and video filenames, and audio and video channel names. The third loop contains values for paragraph line related data, and the innermost loop contains the values for paragraph line segment definitions. The string records are controlled by the same high watermarks as those used for Boolean and integer records. String records are stored utilizing an appropriate field delimiter technology. In one implementation, a UTF encoding technology that is supported by JAVA is utilized.

Single and double floating-point, and long integer records are written inside a two-dimensional loop at **161**. The outer loop may be empty. The inner loop contains mathematical values required for certain animations and image processing algorithms. The single and double floating-point, and long integer records are controlled by the same high watermarks as those used for Boolean and integer records.

FIG. **25** describes the techniques used to create a customized and optimized run time engine by the build engine (**31** of FIG. **4**). A versionizing program analyzes the feature flags, and only those variable definitions, defined in the "Main" object class, relating to the object classes and methods that will be executed at run time, are extracted at **162**. All references to object classes that will be called at run time are extracted, creating the source code for the run engine "Main" object class that is ready for compilation at **163**.

All external image, video and audio files are resolved at **164**. The external references can be copied to designated directories at **164**, either on the user's local disk or file server. The file Pathnames can be changed to reflect these new locations. During the installation of the build tools, the necessary class libraries from Sun, Microsoft and Netscape are either installed on the local system (See FIG. **5**) or made available on the server where the build tools can be optionally located. The necessary environmental variables are set to permit normal access to the required class libraries.

The customized run engine and a library of the referenced run time classes are compiled and converted into byte code at **165**. Finally, the run time engine for the web site is created at **166**. The required set of class objects required at run time is flagged for inclusion into the CAB/JAR file (See FIG. **27**).

FIG. **26** shows the techniques used to create the HTML Shell File (HSF) (**32** of FIG. **4**). The first step of the process at **167** is to determine whether the dynamic web page and object resizing is desired by testing the application setting, set by the user at FIG. **21***a*, or possibly reset at FIG. **24**. If the application was a web page, and thus requiring dynamic web page and object resizing, virtual screen resolution settings, calculated at FIG. **24** at **153**, are placed in an appropriate HTML compliant string at **168**. If the application is a banner or other customized application, the absolute values for the run time object (applet size) height and width are placed in an appropriate HTML compliant string as absolute width and height values at **169**.

An analysis is made for the background definition for the first internal web page at **170**. If a background pattern is defined, an appropriate HTML compliant string for setting the HTML "background" to the same background image is generated. If the first web page definition is a color instead, then the RGB values from those colors are converted to hexadecimal and an appropriate HTML compliant String is generated setting the "bgcolor" to the required hexadecimal value. This process synchronizes the web page background with the background that will be drawn by the browser when it first interprets the HSF.

Thereafter, a JAVA method generates HTML and JavaScript compliant strings, that when executed by a browser, generate additional sets of HTML and JavaScript compliant strings that are again executed by the browser. More specifically, if the application required dynamic web page and object resizing (See **167**) then JavaScript and HTML compliant strings are generated at **171** so that, when interpreted by the browser at the time the HTML Shell File is initialized, the screen resolution sensing JAVA applet (SRS) will be executed. JavaScript code is generated at **172** in order to enable JavaScript to SRS applet communication. In one implementation, the code is generated by performing the following functions:

1: Determine the current browser type.

2: Load the SRS from either a JAR or CAB File, based on browser type.

3: Enter a timing loop, interrogating when the SRS is loaded.

4: When the SRS returns an "available" status, interrogate the SRS, which will return the current screen and window's actual height and width.

5: Convert the virtual screen resolution settings into appropriate absolute screen width and height values.

Strings defining additional JavaScript code are generated at **173** that perform the following steps at the time the HSF is initialized by the browser:

1: Generate HTML compliant strings that set the run time engine's applet size to the appropriate values.

2: Generate an HTML complaint string that contains a "param" definition for linking the run time engine to the External Database created at FIG **24**.

3: Generate an HTML complaint string, dependent upon the type of browser, which causes the current browser to load either the JAR or the CAB File(s).

4: Generate JavaScript Code compliant strings that create and dynamically write the applet size defining HTML strings utilizing the JavaScript "document.write" function. This dynamically created code causes the browser to execute the run time engine, in the correctly sized window, from the correct JAR or CAB file, and linked to the external database.

At **174**, writing out the above-generated HTML and JavaScript compliant strings creates the HSF. The necessary security policy permissions are asserted, and a "Websitename".html file is created.

FIG. **27** describes the processes for creating the CAB and JAR Files (**33***a* of FIG. **4**). The image objects, if any, which were defined on the first internal web page are analyzed at **175**. If they are set to draw immediately upon the loading of the first web page, then they are flagged for compression and inclusion in the CAB and JAR Files. The feature flags are analyzed at **176** to determine which JAVA classes have been compiled (See FIG. **25**). These class files are flagged for compression and inclusion in the library CAB and JAR Files. Strings that are BAT compliant definitions are created at **177** that will, when executed in DOS, create compressed CAB and JAR Files. These CAB and JAR Files contain the compressed versions of all necessary JAVA class files, image files, the "Websitename".class, customized run time engine file, and the "Websitename".dta database file. In one implementation of the invention, two BAT files are created. The first, when executed, will create a CAB/JAR file with the "Websitename".dta database file and the customized "main" run time engine, excluding all the image and button object animation, transformation, and image processing code. The second BAT file, when executed, will create a CAB/JAR file with all the library of all the referenced image and button object animation, transformation, and image processing code.

The necessary security policy permissions for file creation are then asserted, and "Websitename".bat and "Websitenamelib".bat files are written at **178**. The "Websitename".bat and "Websitename".bat files are then executed under DOS, creating compressed "Websitename".cab and "Websitenamelib".cab files and compressed "Websitename".jar and "Websitenamelib".jar files at **179**. The HTML Shell File and the JAR and CAB files are then, either as an automatic process, or manually, uploaded to the user's web site. This completes the run generation processes.

Run Time Process

The run time process is shown in FIG. **28** through FIG. **36**.

FIG. **28** shows the web page size generation technology utilized by the run time engine. A web surfer points a browser at the HTML shell file (HSF) at **180**. The browser begins to interpret the HTML and JavaScript code in the HSF that was created (See FIG. **26**). The browser draws either the background color or background image pattern, as defined by the HTML complaint code (See FIG. **26**) at **170**. The browser then executes the HSF's JavaScript initialization code, which "sniffs" the current browser at **181** to determine its type, and then generates the appropriate HTML code for that particular browser to interpret. This code defines whether the executable files and database will be extracted from inside a compressed CAB file or a compressed JAR file and its location.

Based on the user application (defined at FIG. **21***a*, or possibly reset at FIG. **24**), the HSF at **182** will then execute an appropriate JavaScript function (as created in FIG. **26** at **167**). If the application required dynamic resizing of the web page's dimensions, JavaScript code is called which generates HTML code using the JavaScript "document.write" function, which causes the SRS applet to be immediately executed by the browser at **183**. The JavaScript code then goes into a timer loop, checking on when the SRS applet is alive before initiating any communication. After detecting that the SRS has been initialized, a JavaScript function calls the appropriate SRS applet methods at **185**, which return the width and height, in pixels, of the current browser window. JavaScript Code is then called which converts the screen resolution independent window width and height values into absolute pixel values. A JavaScript function is then called which use the JavaScript "document.write" function to generate HTML code that define the run time engine specifications, etc. (see FIG. **26**) and cause the browser to immediately execute the run time engine. If the application had not required dynamic resizing of the web page's dimensions, then a JavaScript function is called which generates HTML code using the JavaScript "document-.write" function that defines the fixed dimensions for the web page size and cause the browser to immediately execute the run time engine at **184**.

FIG. **29** shows the techniques employed by the run time engine to read the external database and to generate the necessary web page objects (**35** of FIG. **4**). The run time engine reads a "PARAM" value at **186**, from HTML Code that was generated above (see FIG. **26**), which points to the "Websitename".dta external database that is compressed into the JAR or CAB File (that was loaded and accessed in FIG. **28**). The run time engine then initiates the read operation. In one implementation, the read technique is always non-privileged. If permitted by the current browser as a non-privileged operation, the "Websitename".dta file will be extracted and read from the CAB/JAR file residing in temporary local storage. If not, the run time engine will read the "Websitename". dta file directly from the server. The header record is read at **187**. Any objects, such as fonts and colors, are cast into their original form. The high watermark values, as they are encountered in the header and in the body of the database, are immediately used for setting the limits for the subsequent multilevel read loops for reading in the style record and the web page(s) and object(s) definitions. The virtual screen resolution values are read for the subsequent dynamic resizing of the web page objects.

The style record is read based on its high watermarks, and processed at **188**. The definitions for all paragraph, text

button, image or other styles are read and stored for subsequent initialization and processing of all paragraph, text button, image or other objects. The data representing the values for the first web page and all its objects is read at **189**. The Boolean, integer, string and floating point fields for the first web page are initialized. The serialized multimedia objects for the first web page are read and cast into their final form. (See FIG. **24**)

If external files, such as image, audio and video files, must be read as part of the first web page's generation, exception handling routines are executed at **190**, as necessary, in the event of any processing errors. In one implementation, error recovery at this stage places the highest priority on a graceful operation cancellation, rather than a web page crash. In the worst case, a particular image, sound or video file may not be available to the web surfer. All other aspects of the web page will likely be available even in this error scenario.

Process step **191** is executed simultaneously with the generation of all the other web pages at **192** by means of multiprogramming utilizing thread technology. Thus the first web page will be drawn and active for user viewing and user interaction long before the data for all the other web pages have been read, processed, and initialized. The data representing the values for the subsequent web pages and all their objects are read at **192**. The Boolean, integer, string and floating point fields for these web pages are initialized. The serialized multimedia objects for these web pages are read and cast into their final form. (See FIG. **24**)

FIG. **30** shows the scaling techniques employed by the run time engine for web page generation (**36** of FIG. **4**). The first step in the scaling process is to calculate the coordinates that define the origin for the placement of each object for a given web page. (This is usually the upper left corner of the object, defined in actual screen pixels.) A test is made at **193** to determine if the centering attribute is set for the object. If not, the left and top coordinates are converted from the virtual screen values to the local screen values, based on the local screen window resolution at **194**. In one implementation, multiplying the virtual coordinate by the local screen window resolution and dividing by the virtual screen resolution determine the conversion.

If the centering attribute is on, then a calculation for the object's width is performed. See processes **197**, **198**, and **199** below for a description of this calculation. Based on this calculated width, and based on the local screen window resolution, the left coordinate is calculated at **195**. One algorithm that can be used is to subtract the screen width, as calculated in **197**–**199** below, from the local screen window resolution, and divide that result by 2. The top coordinate is calculated the same as in process **194** above.

Based on the object type, determined at **196**, a different scaling technology is employed. If the object is a text button object at **197**, the text button object itself, including its background, is not scaled. The virtual width and the local screen width remain the same. However, if a 3D Frame effect is defined, it is scaled based on the following algorithm: if the text string's orientation is Left to Right, the inner width of the 3D Frame, and its placement relative to the text string, is calculated as the length of the text string, plus $\frac{1}{8}$ of an "n" space on each side, plus an additional offset appended to the right of the inner width to compensate for the italic font style, if defined for the font of that text string. The italic offset can be defined as the font size for the text string, divided by 10, plus 1. The inner height of the 3D Frame can be defined as the font height plus 2 pixels. The font height equals the font's leading plus its ascent plus its descent specifications. The inner height origin can equal the text string origin. The style of the 3D effect (i.e., either a 3D raised look or a 3D depressed look), plus the inner width and height, is sent to a 3D frame build method for the construction of the 3D frame. The width of the 3D frame in pixels can be calculated as the inner width divided by 10 plus 3.

If the text string's orientation is vertical, the inner width of the 3D Frame is an "m" space. The inner height of the 3D Frame can be calculated as the font height times the number of characters in the text string. Both the left and top placement of the 3D frame can be set to the left and top origin of the text string. The width of the 3D Frame can then be calculated as the inner height divided by 10, plus 3.

If an animation is assigned to the text string, the font size used for the initial calculation of the 3D frame is the same as that used to define the animation's initialization value. If the object is a paragraph at **198**, and the scaling attribute is on, the maximum width for the paragraph can be defined by the attached paragraph style (or paragraph override) as a percentage of the screen width. This screen width percentage can be converted into an actual width in pixels, based on the local screen's window resolution. If the current screen resolution is the same as that used by the web designer, then the paragraph line end values (just read from the external database) are used without adjustment, bypassing the entire paragraph reformat process. If the current screen resolution is different than that of the virtual screen resolution, then a very compact method of reformat is called (relative to the build engine reformat methods at FIG. **6** and at FIG. **18**), and the text for the paragraph is reformatted based on this width.

The run time engine's reformat technology begins by creating one paragraph line for the entire text string assigned to the paragraph text area. All the style overrides are renumbered sequentially with the style records or the non-marked text strings ignored. A simplified "Overflow" reformat method can be called, which chops up the single paragraph line first into paragraph line segments, where each word is defined as a line segment. Because of the word oriented style override architecture, the style overrides have a one-for-one correspondence with the line segments. Each paragraph line break can be calculated by relying on the simplified word oriented style override technology described above. The paragraph line can be built inside a tight word-by-word loop, with a simple logic check for a style override or hard line break. The paragraph width is then derived as the width of the longest line of the reformatted paragraph, while the paragraph height is defined as the font height times the number of lines. If a 3D frame was defined for the paragraph, it can be scaled based on the following algorithm:

The inner width is defined as the same as that of a text string, but the width of the text string for the longest line is used. The same "n" space and italic offset calculations are used. The inner height is calculated as the font height times the number of lines plus 2 pixels.

If the object is a paragraph, and the scaling attribute is off, then the paragraph is treated the same as a text button object, with the only exceptions that there is no vertical orientation, and the height and width of the 3D frame, if defined, is calculated using the same algorithm as was used for the scaled paragraph above.

If the object is an image at **199**, and the scaling attribute is on, the image width can be calculated as the virtual width times the local screen window width divided by the virtual screen width. The image height can be calculated as the virtual height times the local screen widow height divided by the virtual screen height. If the image had been resized or

rotated, then the virtual width and height of the image would differ from that of that of the original image. If a 3D frame is defined for the image, it can be scaled based on the following algorithm:

The inner width and the inner height of the 3D frame will coincide exactly with the outer edges of the image, after the image had been scaled. Adding the scaled image height to the scaled image width and dividing the result by **40** and adding **3** can calculate the width in pixels of the 3D frame.

If an animation is assigned to the image, then the animation's initialization values for the image's width and height can be used to calculate and draw the initial 3D frame. The coordinates and sizes for the backgrounds for text button, image and paragraph objects can be calculated using the same algorithms as was employed for the calculation and placement of the inner width and inner height for the 3d frame for each object.

FIG. **31** through FIG. **35** shows the multilevel web page and object thread technology employed by the run time engine. The description includes all the animation technologies, transformation technologies, time line technologies and drawing technologies that support this multilevel architecture.

FIG. **31** describes the initial processes for the invention's multilevel web page and object thread technology employed by the run time engine (**37** of FIG. **4**). Upon the completion of the processing of all the data definitions for the first internal web page (FIG. **30**), the main web page thread is created and executed. This causes the run method for the main run time engine class to be executed simultaneously with the reading, processing, and scaling of the data for the subsequent web pages (See FIG. **29**). In addition, the reading of any image files defined for the first web page is also performed simultaneously, under the control of an image observer (See FIG. **12**). The main run method enters a web page counter loop at **200**, the loop being defined from the first internal web page to the high watermark that was set to the number of existing internal web pages for the web site.

A check is made at **201** to see if the current web page exists. If the web page does not exist, and the current web page number is less than that of the high watermark, then the web page counter is incremented by one and the web page counter loop is reentered. If the current web page number equals the high watermark at **202**, then the web page counter is reinitialized to the first web page, so that the web page loop may repeat itself, from the first internal web page, depending upon the delay setting for the last web page.

A test is then made on all objects defined for this web page at **203**, utilizing a loop whose range is defined by the number of objects per web page high watermarks. More specifically, within this universe of possible objects, if the object exists, and it is defined by a time line in which there is a delayed entrance, then a boolean flag is set for those objects that causes the draw system to suppress drawing these objects during the web page transition as defined below.

A test is then made to determine if the web page has a transition animation defined at **204**. If not, the draw system is called for the first time. The draw system for a given web page utilizes a loop whose range is defined by the number of objects per web page high watermarks. The draw system can also employ technology so that the draw process generates a screen image in one or more off-screen buffers, only drawing to the screen when the screen image, or the clipping area for the screen, has been fully generated. This greatly reduces, if not totally eliminates, any screen flicker, and creates visually smooth animation effects.

The first draw function is to draw the web page background into the primary off-screen buffer. The web page background color is drawn, as defined initially at FIG. **21***a*, or modified for that particular web page at FIG. **20**. A test is then made to determine if the web page has a background image pattern, as defined initially at FIG. **21***a*, or modified for that particular web page at FIG. **20**. If it does, and the image observer reports that the image is ready to be drawn, a background image draw loop is executed, defined by the height and width of the background image, and the screen resolution of the current browser window. In the unlikely event that the background image pattern is not yet available, there is a delay until the image observer reports the completion of the image processing operation. The tiled background image pattern is also drawn into the primary off-screen buffer, completely overdrawing the background color. The backgrounds for all non-suppressed (See **203**) parent web page text button and paragraph objects are then drawn into the primary off-screen buffer, unless a background transparency flag has been set (See FIG. **7**).

The text strings for non-suppressed parent web page text button and paragraph objects are then drawn into the primary off-screen buffer. These text strings are drawn based on their font name, style, size, effect(s), and color. If a paragraph line string, the string may have multiple string segments, each with their own font name, style, etc. If the text button object has its vertical attribute set to true, then the draw system executes a loop defined by the number of characters defined in the text button object. The top and left origin coordinates were set in the usual way (See FIG. **30**), but the top coordinate is adjusted by the font height for each iteration of this draw loop. The intelligent 3D Frame, if defined, is then drawn into the primary off-screen buffer for the paragraph and text button objects (See FIG. **30**). The primary image objects for the web page are then processed by the draw system. If the image observer reports that the image is ready to be drawn, it is drawn into the primary off-screen buffer, based on the coordinates and size as defined in FIG. **30**. If not ready, there is a delay until the image observer reports the completion of the image processing operation. The Intelligent 3D frame, if defined, is then drawn into the primary off-screen buffer for the image objects (See FIG. **30**).

The draw system is responsive to two other technologies at this stage. The first is user interaction based on the location of the mouse cursor and any user initiated mouse event. This subject will be described in greater detail below in association with FIG. **36**. The second is object animation for non-delayed web page objects. This subject will be described in greater detail below in association with FIG. **33**.

If the web page transition test at **204** was true, then the run time engine's main run method executes the web page transition animation technology at **205**.

FIG. **32** describes the web page transition animation technology. First a lock is placed on this method at **212**, as a safety precaution to prevent any interference from other threads during the animation. A test is then made on whether the transition animation setting (See FIG. **20**) for the web page is random at **213**. If so, a random transition number is generated at **214**. The web page thread then begins a particular animation loop at **215**, depending upon the random number that was generated at **214** or by the transition animation that was set previously (at FIG. **20**). In one implementation, 13 different transition animations plus random are supported including. They are: Fade In, Zoom In, Zoom Out, Zoom to Upper Left, Zoom to Lower Right, Rotate to the Left, Rotate to the Right, Rotate Bottom to Top, Rotate Top to Bottom, Slide to the Left, Slide to the Right, Slide Bottom to Top, and Slide Top to Bottom.

For all web page transition animations, the X and Y animation increment values are calculated by dividing the current browser's screen width and height by the user defined animation resolution at **215**. In all animation and draw loops, the number of loops can equal the number of animation frames as set at FIG. **20**. The timer delay for all animations, in milliseconds, can be calculated by dividing the number of frames per second (See FIG. **20**) into 1,000.

For a description of "Fade In" Technology see FIG. **33**. A "Zoom In" algorithm sets the initial scaled width and height for the current web page image to zero and the prior web page image to its full size. In each animation and draw loop the previous web page's final image state is drawn into a secondary off-screen buffer at **216**. (If this is the first occurrence of the first web page, then the secondary off-screen buffer is set to the background of the first web page.) The upper left hand corner (origin) of the current web page can be calculated based on the following formula: browser screen width minus the scaled width divided by two.

The scaled image of the current web page is then drawn into the secondary off-screen buffer at the calculated origin, using the current scaled width and height for the web page image. This merged image of the prior and scaled version of the current web page is then drawn to the screen. A timer delay then occurs as defined at **215**, after which the X and Y animation increment values are added to the scaled width and height for the current web page image. The animation loop is then repeated to its conclusion at **218**.

The other eleven web page transition animations follow a similar methodology, but have quite different calculations, which are based on the following variables:

1: Order of drawing of the prior and current web pages.

2: Initialization values for the X and Y origin coordinates for the current and prior web pages.

3: The initial values for the scaled width and height for the current and prior web pages.

4: Whether X and Y origin coordinates for the current and prior web pages increment, decrement, or remain the same.

5: Whether the values for the scaled width and height for the current and prior web pages increment, decrement, or remain the same.

For the "Zoom Out" animation, the current page is drawn first and always drawn at 100%. The prior web page is initialized also at 100%, but its X and Y origin coordinates are incremented and its scaled width and height values are decremented, by the appropriate values, for each animation iteration.

For the "Zoom to Upper Left", "Zoom to Lower Right", "Rotate to the Left", "Rotate to the Right", "Rotate Bottom to Top" and "Rotate Top to Bottom" animations, a common data initialization and data increment strategy is implemented.

1: The X and Y variables for page image one is set to zero.

2: The X and Y variables for page image two is set to the right and bottom edges of the browser window.

3: The scaled width and height variables for page image one is set to 100% of the browser window's resolution.

4: The scaled width and height variables for page image two is set to zero.

5: For each loop iteration, the scaled width and height variables for page image one are decremented by the X and Y animation increment values defined at **215**.

6: For each loop iteration, the scaled width and height variables for page image two are incremented by the X and Y Animation increment values defined at **215**.

For the "Zoom to Upper Left" animation, the prior web page is drawn first; with its X and Y origin coordinates always set to zero. (upper left corner of the browser window) Its scaled width and height values are always set to the current values for scaled width and height variables for page image one. The X and Y origin coordinates for the current web page can be calculated by subtracting the current values of image two's scaled width and height variables from the initial values of the X and Y variables for page image two. The scaled width and height values for the current web page can be set to the current values for the scaled width and height variables for page image two.

For the "Zoom to Lower Right" animation, the current web page is drawn first; with its X and Y origin coordinates always set to zero. Its scaled width and height values are always set to the current values for scaled width and height variables for page image two. The X and Y origin coordinates for the prior web page are set to current values of image two's scaled width and height variables. The scaled width and height values for the prior web page are set to the current values for the scaled width and height variables for page image one.

For the "Rotate to the Left" animation, the prior web page is drawn first; with its X and Y origin coordinates always set to zero. Its scaled width value is set to current value of image one's scaled width variable. Its scaled height value is always set to the bottom of the browser's window. The X origin coordinate for the current web page can be calculated by subtracting the current value for image two's scaled width variable from the initial value for image two's X origin coordinate. The Y origin coordinate for the current web page is always set to zero. Its scaled width value is set to current value of image two's scaled width variable. Its scaled height value is always set to the bottom of the browser's window.

For the "Rotate Bottom to Top" animation, the prior web page is drawn first; with its X and Y origin coordinates always set to zero. Its scaled width value is set to the width of the browser window. Its scaled height value is set to current value of image one's scaled height variable. The current web page's X origin coordinate is always set to zero. The Y origin coordinate is calculated by subtracting the current value of image two's scaled height variable from the initial value for image two's Y origin coordinate. Its scaled width value is always set to the right edge of the browser's window. Its scaled height value is set to current value of image two's scaled height variable.

For the "Rotate Top to Bottom" animation, the current web page is drawn first; with its X and Y origin coordinates always set to zero. Its scaled width value is set to the width of the browser window. Its scaled height value is set to current value of image two's scaled height variable. The prior web page's X origin coordinate is always set to zero. The Y origin coordinate is set to the current value of image two's scaled height. Its scaled width value is always set to the right edge of the browser's window. Its scaled height value is set to current value of image one's scaled height variable.

For the "Slide to the Left", "Slide to the Right", "Slide Bottom to Top" and "Slide Top to Bottom" transition animations, a common data initialization and data increment strategy is implemented. The strategy includes

1: The X and Y variables for page image one is set to zero.

2: The X and Y variables for page image two is set to the right and bottom edges of the browser window.

3: For each loop iteration, the X and Y variables for page image one are incremented by the X and Y animation increment values defined at **215**.

4: For each loop iteration, the X and Y variables for page image two are decremented by the X and Y animation increment values defined at **215**.

5: The scaled width and height values always remain at 100% of the browser windows width and height.

For the "Slide to the Left" animation, the prior web page is drawn first; with its X and Y origin coordinates always set to zero. The current web page's X origin coordinate is set to the current value of page image two's X variable. Its Y origin coordinate is always set to zero. For the "Slide to the Right" Animation, the current web page is drawn first, with its X and Y origin coordinates always set to zero. The prior web page's X origin coordinate is set to the current value of page image one's X variable. Its Y origin coordinate is always set to zero.

For the "Slide Bottom to Top" animation, the prior web page is drawn first; with its X and Y origin coordinates always set to zero. The current web page's Y origin coordinate is set to the current value of page image two's Y variable. Its X origin coordinate is always set to zero.

For the "Slide Top to Bottom" animation, the current web page is drawn first; with its X and Y origin coordinates always set to zero. The prior web page's Y origin coordinate is set to the current value of page image one's Y variable. Its X origin coordinate is always set to zero.

After the last animation cycle is completed for any of the transition animations at **218**, the animation process is unlocked, and process step **206** shown in FIG. **31** is then executed.

Returning to FIG. **31**, the main web page thread's run method then executes a text button and image object time line, transformation and animation loop at **206**. This range loop is defined from the first object on the given web page to the high watermark for the number of those objects on a web page for this web site. A test is made on each object on whether an animation, transformation and/or time line has been assigned at **208**.

If so, an "instance" of the time line class for that particular object type is created at **209**. An "instance" of a class is a fundamental aspect of object oriented programming (OOP). Each time, the line class is implemented with a "runnable" interface, so that they can be executed as independent threads. Communication of data, between the "instance" of a class and the main run engine class can be accomplished in OOP using several different techniques. In one implementation, this construction, passed as an argument, is used to permit different objects to address each other's variables and databases. A thread is then created, utilizing a two-dimensional object internal database architecture (web page number by internal object number). This methodology is convenient for permitting all object time lines for a given web page to be managed and synchronized. The object's thread is then "started".

The result of this process at **209** is that an independent thread has been created for each appropriate object on a given web page, all executing simultaneously with each other and with the main run time engine web page thread, subject to the definitions of their independent time lines at **210**. See FIG. **33** for a description of the time line technology. When the main web page thread has finished the text and image loop at **207**, the draw system is activated; the run time engine can now respond to user interactions, and the main web page thread transitions into a "Join" loop at **211**. See FIG. **35** for a description of this process.

FIG. **33** shows the time line technology used by the run time engine. The techniques and algorithms employed to create this technology permit each web page object to have an independent yet synchronized existence with each other, with the main web page thread, and with child objects that each main or parent object may spawn. Furthermore, each object and each of their child objects are capable of performing multiple animations and transformations, either serially or simultaneously. Database initialization is first accomplished for each object thread. This assures that the object thread's database is set to the correct initial values as required for that particular object, and that the references to the main web page thread's database are established.

A test is then made to determine if the object has a time line definition assigned to it at **219**. If not, a test is made at **220** on certain two-dimensional object definition variables in order to determine which of the following four states have been defined for the object: animation without a transformation; transformation without animation; animation, with the transformation occurring simultaneously with the animation; and animation and transformations occurring in a serial manner.

If the test shows that the object has an animation defined, but no transformation, then certain two-dimensional status variables are set, and an "instance" of the "animation class" for that particular object type is created at **229**. Each "animation class" is also implemented with a "runnable" interface. An object animation thread is then created, utilizing the two-dimensional object internal database architecture (See FIG. **8**). This object animation thread is then "started". Communication between the object animation thread, the parent time line thread, and its parent, and the main web page thread, are accomplished as discussed in process **209**. The object time line thread then executes a "Join" method. This puts the object time line thread in a "wait state". When the thread it is waiting for is completed, this child thread "joins" the parent object time line thread, and the object time line thread then continues its process. Other forms of synchronization between two independent threads could have been implemented as is known in the art.

The techniques employed at **229** to implement object animation vary by object type. In one implementation, for text button object animations, 26 different animations are supported including: Zoom In, Zoom Out, Grow NW, Grow NE, Grow SE, Grow SW, Shrink SE, Shrink SW, Shrink NW, Shrink NE, Enter N, Enter NE, Enter E, Enter SE, Enter S, Enter SW, Enter W, Enter NW, Exit N, Exit NE, Exit E, Exit SE, Exit S, Exit SW, Exit W and Exit NW. In one implementation, for image object animations, 29 different animations are supported including: Fade In, Fade out, Rotate, Zoom In, Zoom Out, Grow NW, Grow NE, Grow SE, Grow SW, Shrink SE, Shrink SW, Shrink NW, Shrink NE, Enter N, Enter NE, Enter E, Enter SE, Enter S, Enter SW, Enter W, Enter NW, Exit N, Exit NE, Exit E, Exit SE, Exit S, Exit SW, Exit W and Exit NW.

As discussed above with regard to FIG. **17** each animation type has a defined speed, resolution, and number of animation cycles. These settings are stored in the main web page class, and are passed to the particular animation thread through a two-dimensional object internal database architecture as discussed in process step **209** above during the animation thread's initialization process. The animation thread then executes, in its run method, a main animation loop that has the number of iterations set to the end number of animation cycles, as assigned to that particular text button object.

Text button animations are currently implemented in three logical groups. Group One includes "Zoom In", "Grow NW", "Grow NE", "Grow SE", and "Grow SW". Group Two includes "Zoom Out", "Shrink SF", "Shrink SW", "Shrink NW", and "Shrink NE". Group Three includes "Enter N", "Enter NE", "Enter E", "Enter SE", "Enter SW", "Enter W", "Enter NW", "Exit N", "Exit NE", "Exit E", "Exit SE", "Exit S", "Exit SW", "Exit W" and "Exit NW".

For Group One text button animations, the animation font size is initialized at a very small value, and in one implementation is set at 4 Points. The animation point size increment can be derived by dividing the resolution (number of animation frames) into the font size for that text button object. The run method then executes a secondary animation loop, which will terminate when the animation font size equals the text button object point size. For each secondary animation loop, the length of the current animated text string is calculated, a new font object is created for the current animation point size, and the font metrics for that new font are created. If the text button object has a vertical orientation, the animated text button object's width is calculated to be the width of an "m" space, in the current animated font. The animated text button object's height is calculated to be current animated font height times the number of characters in the text string. If the text had a horizontal orientation, the animated text button object's width is calculated to be the width of the text string in the current animated font. The animated text button object's height can be calculated to be the font height of the current animated font. The calculations for X and Y coordinates for the animated text button object depend upon which animation was defined within the Group One-text button animations. The X and Y animation increments can be calculated utilizing the height and width, in pixels, of the text button object scaled to the current browser's window, utilizing the text button animation resolution, and considering whether the animating text button object is being centered during the animation ("Zoom Out") or not. These calculations are similar to those for the web page transition animations discussed with regard to FIG. **32**.

The draw system is then called. Based on the values of the two-dimensional status variables that had been set initially, the draw system executes the appropriate animation draw routine utilizing, through the data communication techniques already discussed, the current animation font point size, and the current animation X and Y coordinates. If a text background is to be drawn, the same algorithm as defined in FIG. **31** is used. If a 3D Frame is assigned, the current animated string width and height are passed to the appropriate 3D frame generation method, and the frame is drawn with the same algorithm as defined in FIG. **31**, but utilizing the current animation X and Y coordinates. The text button object's orientation is also handled by the draw system with the same algorithms as defined in FIG. **31**.

The text button animation thread then executes a timer delay, whose value had been defined in FIG. **17**. When the timer reactivates the text button animation thread after the appropriate delay, an animation cycle completion test is made to see if the text button object's point size minus the animation point size is less than the animation point size increment. This type of testing methodology permits the invention to utilize integer values, as opposed to floating point values, for the text button animation. This improves the execution of the animation considerably.

If the above test is true, the animation point size is set equal to the object point size and a final call is made to the draw system for that animation cycle. A test is then made to see if the current animation cycle equals the total number of animation cycles as defined in FIG. **17**. If not, a new animation cycle is initiated, with the animation values reinitialized. If this was the last animation cycle the text button animation thread calls its "stop" method, which sets the required status variables as appropriate, then terminates itself This causes the parent text button time line thread to be reactivated through the "join" mechanism.

If the results of animation cycle completion test are false, the current animation point size is increased by the animation point size increment. A new font object is created for the now current animation point size, and new font metrics for that new font are created. If the text button object has a vertical orientation, the animated text button object's width is calculated to be the width of an "m" space, in the now current animated font. The animated text button object's height is calculated to be the now current animated font height times the number of characters in the text string. If the text has a horizontal orientation, the animated text button object's width is calculated to be the width of the text string in the now current animated font. The animated text button object's height is calculated to be the font height of the now current animated font. The calculations for the new X and Y coordinates for the animated text button object are then completed, as appropriate, and the draw system is called again.

The algorithms for Group Two text button animations are very similar to those of Group One. The differences are just in what are the initial animation values, and whether the animation point size increments and the animation X and Y coordinate increments are added or subtracted from the then current animation point size and the then current X and Y coordinates for the animating text button object.

For Group Three text button animations, the distance that the text button animation will move is calculated, in pixels, from its initial location to its final location in the current browser window. The X and Y animation increments are calculated by dividing that distance by the resolution of the text button animation. All the other algorithms for Group Three text button animations are generally a subset of those for Group One, and similar to the web page slide transition animations defined with reference to FIG. **31**.

Referring again to FIG. **33**, image animations at process step **229** can currently be grouped into five logical classes. As with text button animations, Group One includes "Zoom In", "Grow NW", "Grow NE", "Grow SE", and "Grow SW". Group Two includes "Zoom Out", "Shrink SE", "Shrink SW", "Shrink NW", and "Shrink NE". Group Three includes "Enter N", "Enter NE", "Enter E", "Enter SE", "Enter S", "Enter SW", "Enter W", "Enter NW", "Exit N", "Exit NE", "Exit E", "Exit SE", "Exit S", "Exit SW", "Exit W" and "Exit NW". In addition, image animations have a Group Four, which includes "Fade In" and "Fade Out". Group **5** image animations include the "Rotate" Animation.

For Group One mage animations, the animation width and height increments are calculated by dividing the image object's width and height by the resolution (number of animation frames) as set in FIG. **17**. The initial animation width and animation height values are set to a very small number, currently equal to the animation width and height increment values just calculated. The calculations for X and Y coordinates for the animated image object depends upon which animation was defined within the Group One text button animations. The X and Y animation increments are calculated utilizing the height and width, in pixels, of the image object scaled to the current browser's window, utilizing the image animation resolution, and considering whether the animating image object is being centered during the animation ("Zoom Out") or not. These calculations are similar to those for the web page Transition Animations discussed above with regard to FIG. **32**.

The run method then executes a secondary animation loop, which will terminate when the animation width equals the image object's width. The algorithms employed by the invention to change the animating object's height, width, X coordinate, and Y coordinate are very similar to those employed for Group One text button animations, and will not be repeated here. The techniques to utilize the draw system for drawing the image animation, the time delay technique, and the post draw logic tests and actions are also very similar.

The algorithms for Group Two image animations are very similar to those of Group One. The differences are just in what are the initial animation values, and whether the animation width and height increments and the animation X and Y coordinate increments are added or subtracted from the then current animation width and height and the then current X and Y coordinates for the animating image object.

For Group Three image animations, the algorithms are identical to those of Group Three text button animations. For Group Four image animations, the "alpha" value of a given image object is utilized in order to implement "Fade In" and "fade Out" animations. The alpha value can range from 0 to 255, depending upon the image strength desired. The value for an alpha animation increment variable can be calculated by dividing the resolution of the animation into 255, after making the necessary adjustments to keep the data in integer form, without losing resolution due to integer rounding errors. For a "Fade In" animation the value of an alpha animation variable is set to zero. The run method then executes a secondary animation loop, which will not terminate until 255 minus the then current value of the alpha animation variable is less than the value alpha animation increment variable. A "Fade In" image filter can be created for each iteration of the animation loop, using the current setting of the alpha animation variable. An image producer can also be created with pointers to the last image bitmap produced for the image object in the last animation loop and to the image filter that has just been created. The image producer, under the control of a media tracker then creates a new image bitmap. The animation thread then "waits" for the completion of this image-processing event using the media tracker. Upon completion, the draw system is called which draws the then current state of the image object. The image animation thread goes into a timer delay of some preset value (in one implementation 500 milliseconds), to permit a smooth visual animation effect. The value for the alpha animation increment is added to alpha animation variable and the loop is then repeated until the loop condition is met. Then the "stop" method is called, certain status variables are set, and the image animation thread terminates itself This causes the parent image time line thread to be reactivated through the "Join" mechanism.

The "Fade Out" animation employs very similar technology, except that:

1: the alpha animation variable is set to zero,

2: the value for the alpha animation increment is subtracted, and

3: the loop termination test is when the value for the alpha animation variable is less than the value for the alpha animation increment.

For the Group Five image rotate animation, a different bitmap for the image object is created for each animation frame through the use of a progression of standard geometrical transformations on the original image bitmap. A secondary animation loop is then executed as defined by the number of animation frames. In each loop iteration, an image object is created from an appropriate image bitmap selected from among the set just created, the necessary two-dimensional variables are set to communicate with the draw system, and the draw system is then called. The image animation thread then executes a timer delay method based on the delay setting as defined above with reference to FIG **17**. When the timer reactivates the image animation thread after the appropriate delay, the next iteration of the secondary animation loop is repeated until the loop condition is met. Then the "stop" method is called, certain status variables are set, and the image animation thread terminates itself. This causes the parent image time line thread to be reactivated through the "join" mechanism.

Returning to process step **220** shown in FIG. **33**, if the object had a transformation, but not an animation, then certain two-dimensional status variables are set, and an "instance" of the "transformation class" for that particular object type is created at **228**. Each "transformation class" is also implemented with a "runnable" interface. An object transformation thread is then created, utilizing the invention's two-dimensional object internal database architecture. This object transformation thread is then "started". The inter-thread communication technology and the "join" technology employed for object transformations is the same as for object animations.

If the transformation is being applied to a text button object at **228**, then a timer delay method is executed based on the delay setting as described in association with FIG. **18**. When the timer reactivates the text button transformation thread after the appropriate delay, the appropriate two-dimensional status variables are set to inform the draw system which state of the current text button object to draw. The draw system is called and,-based on the settings for the above mentioned two-dimensional status variables, either the "normal", mouse over", mouse down" or "pop-up" states of the text button object's background, if any, the text button object's string, and the 3D frame, if any, are drawn. If additional transformations are defined (FIG. **18**), the above process is repeated, based on the timer delay and object states defined for the subsequent transformations. When the last transformation is completed, the "stop" method is called, which sets the required status variables as appropriate. This causes the parent text button time line thread to be reactivated through the "join" mechanism.

If the transformation is being applied to an image object at **228**, then a timer delay method is executed based on the delay setting (as defined in FIG. **18**). When the timer reactivates the image transformation thread after the appropriate delay, image transformation technology is executed. In one implementation, the image transformation technology utilizes the "alpha" value of a given image object state in order to fade in and fade out images. The alpha value can range from 0 to 255, depending upon the image strength desired. The value for an alpha transformation increment variable is calculated by dividing the resolution of the transformation into 255, after making the necessary adjustments to keep the data in integer form, without losing resolution due to integer rounding errors. The value of an alpha transformation variable is set to zero. Depending upon the settings as defined in FIG. **18**, the bitmap for one image object state is initialized to an alpha value of zero, while another is initialized to an alpha value of 255. The appropriate two-dimensional status variables are set for communication with the draw system.

A transformation loop is then executed, until 255 minus the then current value of the alpha transformation variable is less than the value alpha transformation increment variable. This methodology again keeps all calculations in the form of

integers, as opposed to floating point, thus speeding up the transformation process.

Two "Fade In" image filters are created for each iteration of the transformation loop. The first uses an alpha value calculated at the current setting of the alpha transformation variable. The second uses an alpha value calculated at 255 minus the current setting of the alpha transformation variable. Two image producers are also created with pointers to the last image bitmap produced for each image object state in the last transformation loop and to the two image filters that had just been created. The two image producers under the control of two media trackers then create two new image bitmaps. The transformation thread then "waits" for the completion of these two image processing events using the media trackers. Upon completion, the draw system is called which draws the then current state of the two image object states, in the correct order, and in the correct location. The image transformation thread goes into a timer delay of some preset value (500 milliseconds in one implementation), to permit a smooth visual transformation effect. The loop is then repeated until the loop condition is met. Then the "stop" method is called, certain status variables are set, and the image transformation thread terminates itself. This causes the parent image time line thread to be reactivated through the "join" mechanism.

Returning to process step **220** in FIG. **33**, if the object was defined with an animation and transformation that would execute in a serial manner, then certain two-dimensional status variables are set, and an "instance" of the "transformation" class for that particular object type is created at **230**. An object transformation thread is then created, utilizing the two-dimensional object internal database architecture. This object transformation thread is then "started" and the parent object time line thread "waits" to be "joined".

If a text button object, then a primary loop is executed, with the number of iterations set to the number of transformations. After the execution and return from a timer delay event, if any, an "instance" of the text button animation class is created, and then a text button animation thread is created and "started". The parent text button transformation thread then waits to be "joined". This causes the text button animation thread to be executed, in the manner described at **229**. When the text button animation thread completes its execution, it calls its "stop" method, which sets the necessary status variables and then terminates itself. This causes the text button animation thread to "join" the parent text button animation thread, causing that thread to resume processing. The first text button transformation is then executed, in the manner described at **228**. After the execution and return from another timer delay event, if any, another "instance" of the text button animation class is created, and then another text button animation thread is created and "started". The parent text button transformation thread again waits to be "joined". This causes the text button animation thread to be executed again with the animation being executed, based on the definition set at FIG. **18**, on a different text button object state. The loop is then repeated until the last text button transformation is completed. Then the text button transformation thread calls its "stop" method, certain status variables are set, and the text button transformation thread terminates itself This causes the parent text button time line thread to be reactivated through the "join" mechanism.

If an image object, the mechanism of the image transformation thread spawns image animation threads, before each transformation, and is the same as that of a text button object. The actual image transformation process is identical to that described at **228**. When completed the "stop" method is called, certain status variables are set, and the image transformation thread terminates itself. This causes the parent image time line thread to be reactivated through the "join" mechanism.

Returning to process step **220** in FIG. **33**, if the object was defined with a simultaneous animation and transformation, then certain two-dimensional status variables are set, and an "instance" of the "super transformation class" for that particular object type is created at **231**. In one implementation, the animation, transformation, and super transformation classes are integrated into one structure in order to reduce code size and increase execution speed. Each "super transformation class" is also implemented with a "runnable" interface. An object super transformation thread is then created, utilizing the two-dimensional object internal database architecture. This object super transformation thread is then "started". The inter-thread communication technology and the "join" technology employed for object super transformations is the same as for object transformations.

If a text button object, a calculation is made in order to prorate the text button animation process across the defined text button transformation process. The calculation is driven by the number of text button animation frames, and prorates from that total the number of frames that should be assigned to each transformation state. This can be done by dividing the sum of all the transformation times by each individual transformation time, and multiplying that result by the number of frames, making necessary adjustments to prevent integer rounding error. After these calculations are completed, the text button animation is executed in a similar manner as was defined at **229**. However, when the appropriate number of animation frames had been drawn, certain two-dimensional status variables are set prior to calling the draw system for the next animation frame, so that the correct text button object state is drawn, in the correct size and with the correct coordinates, by the draw system. When the super transformation process is completed the "stop" method is called, certain status variables are set, and the text button super transformation thread terminates itself. This causes the parent text button time line thread to be reactivated through the "join" mechanism.

If an image object, a calculation is made in order to prorate the image animation process across the defined image transformation process. The calculation is driven by the number of image transformation events that would occur (where each one can be set at approximately 500 milliseconds) over the entire animation event. A calculation is performed in order to calculate how many image transformation events should be assigned to each transformation state. This is done by dividing the sum of all the transformation times by each individual transformation time, and multiplying that result by the total number of transformation events, making necessary adjustments to prevent integer rounding error. A calculation is then made to allocate the number of animation frames to each image transformation event. After these calculations are completed, the image animation is executed in a similar manner as was defined at **229**. However, when the appropriate number of animation frames had been drawn, the image transformation technology is called to perform the next transformation event. The alpha transformation increment can be defined by dividing 255 by the number of transformation events assigned to that transformation. The draw system is then called. When the number of image transformation events assigned to a given image transformation is reached, then certain two-dimensional status variables are set prior to calling the draw

system for the next animation frame, so that the correct image states, in the correct size and with the correct coordinates, are utilized by the draw system. This entire animation/transformation process will be repeated by the number of image animation cycles. When the super transformation process is completed the "stop" method is called, certain status variables are set, and the text button super transformation thread terminates itself. This causes the parent text button time line thread to be reactivated through the "join" mechanism.

Returning to process step **219** in FIG. **33**, if the object had a time line, then a test is made at **221** on whether an appearance delay had been defined in FIG. **19**. If so, a timer event is set at **222**. When the timer reactivates the object time line thread after the appropriate delay, a test is made on whether an entry animation/transformation has been defined for this object time line at **224**, as described FIG. **19**. If so, based which animation/transformation process was defined, it is created and executed at **228**, **229**, **230**, or **231**. In one implementation, 13 entry animations are supported for both text button and image objects, and an additional "Fade In" entry animation is supported for image objects. The 13 common entry animations supported include Zoom In, Grow NW, Grow NE, Grow SE, Grow SW, Enter N, Enter NE, Enter E, Enter SE, Enter S, Enter SW, Enter W and Enter NW.

If no entry animation/transformation is defined, or when the entry animation/transformation has "joined" the object time line thread, a test is made to determine if any child time lines have been defined at **225**, as described in FIG. **19**, for this parent object time line. If so, an "instance" of the "child time line class" for that particular object type is created at **226**. Each "child time line class" is also implemented with a "runnable" interface. An object child time line thread is then created, utilizing the two-dimensional object internal database architecture. This object child time line thread is then "started". The inter-thread communication technology and the "join" technology employed for object child time lines is the same as for object time lines. Either a text button child time line thread or an image child time line thread, or both, can be spawned at this time. Simultaneous with the execution of any spawned text button child time line threads, the parent object thread then executes the defined main animation and or transformation. As with non-time line object threads, a test is made on certain two-dimensional object definition variables in order to determine which of the following four states have been defined for the object at **227**: animation without a transformation; transformation without animation; animation, with the transformation occurring simultaneously with the animation; and, animation and transformations occurring in a serial manner.

Based on the results of this test, an appropriate "instance" of an appropriate animation, transformation, or super transformation class is created, and an appropriate animation, transformation, or super transformation thread is created and "started". This results in the execution of process steps **228**, **229**, **230**, or **231**, as defined above.

The parent object time line thread then executes a "join" method. This again puts the object time line thread in a "wait state". When the thread it is waiting for is completed, the child thread "joins" the parent object time line thread, and the object time line thread then continues its process. The object time line thread then checks to see if there is a departure delay defined at **232**. If so, it sets a timer event at **233**. When the timer reactivates the object time line thread after the appropriate delay, a test is made at **234** on whether an exit animation/transformation has been defined for this object time line, as described in FIG. **19**. If so, it is created at **235**, and performed as discussed with reference to processes **228**, **229**, **230**, or **231**. In one implementation, 13 exit animations supported for both text button and image objects, and an additional "Fade Out" exit animation is supported for image objects. The 13 common exit animations include: Zoom Out, Shrink NW, Shrink NE, Shrink SE, Shrink SW, Exit N, Exit NE, Exit E, Exit SE, Exit S, Exit SW, Exit W and Exit NW.

If no exit animation/transformation is defined, or when the exit animation/transformation has "joined" the object time line thread, the parent object time line thread then executes a "join" method if it had spawned any child time lines. This again puts the object time line thread in a "wait state". Finally, when then the child time line threads, if any, "join" the parent object time line, the "stop" method for the parent time line is called. Certain status variables are set, and the parent object time line thread terminates itself This causes the main web page time line that had been in a "join" loop at **211** of FIG. **31**, since the invocation of the object time lines, to be "joined" by this particular object time line thread.

FIG. **34** shows the technology employed by the run time engine for implementing child time lines for text button and image objects. Child text button object time lines and child image object time lines are subsets of their parent object time lines. First a test is made at **237** on whether an appearance delay had been defined (See FIG. **19**). If so, a timer event is set at **238**. When the timer reactivates the child object time line thread after the appropriate delay, a test is made on whether an entry animation has been defined for this child object time line at **239** (as described FIG. **19**). If so, it is created and executed at **240** in a manner identical to that described at process step **229** in FIG. **33**. The same **13** entry animations supported for parent object time lines are also supported for both child text button and image objects, and the additional "Fade In" entry animation is supported for child image objects. The "join" mechanism described in FIG. **33** is employed in an identical manner at **240** to synchronize the child time line thread with its entry animation thread.

After being "joined" and reactivated, the child object time line performs a test at **241** on whether an exit delay had been defined (See FIG. **19**). If so, a timer event is set at **242**. When the timer reactivates the child object time line thread after the appropriate delay, a test is made on whether an exit animation has been defined for this child object time line at **243**, as described in association with FIG. **19**. If so, it is created and executed at **244** in a manner identical to that described above at process step **229** in FIG. **33**. The same **13** exit animations supported for parent object time lines are also supported for both child text button and image objects, and the additional "Fade Out" exit animation is supported for child image objects. The "join" mechanism described above in association with FIG. **33** is employed in an identical manner at **245** to synchronize the child time line thread with its parent object time line thread. As discussed at process step **236** in FIG. **35**, the parent object time lines "wait" until all their child time lines have terminated, before they in turn terminate and "join" the main web page time line at FIG. **35**.

FIG. **35** describes technology employed by the run time engine for the web page and object thread loop. As noted in FIG. **31** at process step **211**, after all the text button and image time line threads for the current web page had been launched, the main web page thread executed a "join" loop, waiting for the completion of all the parent object time line

threads. Because each parent object time line thread waited for their child object time line threads to be "joined", as well as any other spawned animation threads, transformation threads, and/or super transformation threads, the effect of this "join" loop at **246** is that the web page thread will not resume processing until all parent time line threads have completed and that of all of their spawned threads.

Upon resuming its processing after the "join" process at **246** has been completed, the main web page thread checks at **247** to see whether the current web page has an automatic termination, based on a timer delay, or whether the web page will wait for a user interaction before terminating. If the web page has a time delay based termination setting, then a timer method is called at **249**, and the web page goes to "sleep" awaiting the completion of the timer event.

When the timer event occurs, the web page thread resumes processing by incrementing the web page counter by one, and the entire web page process, which began at process step **200** in FIG. **31**, is repeated. If the current web page termination setting was to set to wait until user interaction, then web page thread is placed in a "pause" state, and the run time engine waits to respond to any mouse, keyboard or other user initiated event.

FIG. **36** describes the technology employed by the run time engine for responding to user interactions. As mentioned in association with process step **204** of FIG. **31**, as soon as the draw system has been activated, the run time engine will respond to any user interactions that have been defined (See FIG. **16**). This is also true during any object time line events, as with respect to process step **207** of FIG. **31**. The run time engine currently responds to "mouse over" and "mouse down" events for text button, image, and paragraph objects. For form objects, the run time engine will also respond to keyboard events. As the full-featured programming languages supported by browsers evolve, the run time engine may be configured to respond to other user interactions, including but not limited to single and double clicks from both the left and right mouse button, voice commands, eye focusing technologies, touch screen technologies, and push technologies originating from a server.

The run time engine invokes a "dynamic mouse to object recognition" technology at **251** in order to be responsive to the following elements:

1: The location of objects will vary based on the viewer's screen resolution and browser window size as discussed above with regard to FIG. **27**.

2: Objects may move or resize themselves based on time lines and animations.

3: Objects may have different sizes based on the state they are being displayed in based on time lines and transformations.

4: More than one object can occupy the same screen location, and which objects occupy that location may change over time based on time lines, animations, and transformations.

The run time engine maintains, in its internal database, the object's current X and Y origin coordinates, and the object's current width and height, in pixels, based on the current viewer's screen and browser window size. This can be accomplished by first converting all coordinates and sizes to the current viewer environment with the scaling technology as discussed above with regard to FIG. **27**. Every time line, animation, and transformation thread updates, in real time, the run time engine's internal database positional and size variables of the objects they define, utilizing the data communication techniques described above with reference to FIG. **33**.

The run time engine employs mouseEnter, mouseMove, mouseDown, mouseDrag, mouseUp, and mouseExit methods to constantly monitor the state of the mouse at all times. The onClick method (to detect a single click) and a specialized method to detect a double click event are also employed. The onKeyDown method, with processing the returned scan code, permits the run time engine to process all keyboard events. The mouseEnter, mouseMove, mouseDown, mouseDrag, mouseUp, and mouseExit methods return to the run time engine the exact X and Y coordinates of the mouse cursor at the instant that particular mouse event occurred. Thus for each supported mouse based user interaction technique supported by the run time engine, a two-dimensional loop exists (web page number by object number) in which the current bounding rectangle for every object on a given web page is being compared to the current mouse cursor location at all times. The bounding rectangle is simply the current X and Y origin coordinates of an object, extended by its current width and height. In this way, the run time engine is informed if the current mouse cursor location falls within one or more bounding rectangles. Parenthetically, the run time engine also knows when the current mouse cursor falls outside the bounding rectangle of a given object.

Based on the type of mouse user interaction at **252**, the run time engine employs different techniques and executes different methods. If the viewer moves the mouse at **253**, the mouseMove method informs the run time engine immediately of this event and the current mouse cursor coordinates.

If this user mouse move action caused the mouse to move into one or more bounding rectangles of any text button or image object(s) at **254**, or out of one or more bounding rectangles of any text button or image object(s) at **255**, then appropriate two-dimensional status variables are set and the draw system is called. The draw system interprets the relevant two-dimensional variables for the existing text button and image object(s) on the current web page in its draw loop as described above with reference to FIG. **16**, and draws the correct backgrounds, text strings, images, and 3D frames based on the state of each object, as just set by the "mouseMove" computational two-dimensional loop. If the mouse has entered into or out of the bounding rectangles of any image and/or text button object that has a defined text button, image and/or video pop-up object (See FIG. **16**), then the draw system paints or effectively erases the appropriate background, text string, images and/or 3D frame for these pop-up objects. If the location of any of these text button and image objects or child pop-up objects is changing over time because of time line, animation, or transformation threads, the draw system, as described in association with FIG. **33** and FIG. **34**, is aware of these dynamics, and redraws the screen as these real time events occur. If any sound or video events were defined (See FIG. **16**) for any text button or image objects, then the run time engine plays those sound and/or video files or channels as defined. As multiple objects can be defined that each have associated sound (and even video) files, and these objects can be overlaid on each other, either completely or partially, very interesting synchronized multiple sound tracks can be constructed, in which certain designated sounds are played, or stopped, based purely on user mouse movement.

If the viewer moves the mouse into or out of the bounding rectangle for a paragraph hot link at **256**, then appropriate four-dimensional status variables (web page by paragraph number, by paragraph line number by paragraph line segment number) are set or reset and the draw system is called. The draw system paints the background color behind the hot link text string, and the color for the hot link text string in the hot link "mouse over" or the hot link normal colors. The text string may be underlined or in a bold font, depending on the settings.

If the viewer presses a mouse button at **257**, the mouse-down method informs the run time engine immediately of this event and the current mouse cursor coordinates. If the viewer releases the mouse at **259**, the mouse-up method informs the run time engine immediately of this event. Either way, if the original "mouse down" event had occurred inside one or more bounding rectangles of any text button or image object(s) at **258**, then appropriate two-dimensional status variables are set and the draw system is called. The "mouse up" event will cause the run time engine to reset those appropriate two-dimensional status variables, and then call the draw system. If the mouse was inside of the bounding rectangles of any image and/or text button object that had a defined text button, image and/or video pop-up object (See FIG. **16**) with the freeze attribute, the appropriate two-dimensional status variables are set so that the draw system will not erase these pop-up objects when the mouse moves outside the bounding rectangle(s) of the parent text button or image objects. The draw system interprets the relevant two-dimensional variables for the existing text button and image object(s) on the current web page in its draw loop described above with reference to FIG. **16**, and draws the correct backgrounds, text strings, images, and 3D frames based on the state of each object, as just set by the "mouseDown" computational two-dimensional loop.

If the 3D frame had been previously defined (See FIG. **16**) to have the "live" setting, then the 3D effect is changed from a "raised" effect to a "depressed" effect. If the location of any of these text button and image objects or child pop-up objects is changing over time because of time line, animation, or transformation threads, the draw system, as described in association with FIG. **33** and FIG. **34**, is aware of these dynamics, and redraws the screen as these real time events occur, but does not recognize any new "mouse down" or "mouse up" Events. In one implementation, only "mouse over" events are recognized dynamically by the draw system.

If any sound or video events were defined (See FIG. **16**) for any text button or image objects, then the run time engine plays those sound and/or video files or channels as defined. As multiple objects can be defined that each have associated sound (and even video) files, and these objects can be overlaid on each other, either completely or partially, very interesting synchronized multiple sound tracks can be constructed, in which certain designated sounds are played, or stopped, based purely on the user pressing a mouse button.

If a mouse down event as described previously in association with FIG. **16** was defined for one or more effected text button or image objects, only the object that was drawn last will have its event processed. If the event was to go to a different internal web page, then the run time engine sets the web page counter described in association with FIG. **31** for the "current" page to be one less than that of the desired web page. The current web page's object time lines, if any, complete normally, and the desired web page is then executed. If the "mouse down" event was to go to an external web page in a different window, then the run time engine creates a new browser window. In JAVA this can be accomplished with the "getAppletContext( ).showDocument(theURL, "_blank")" method, where "theURL" is the URL address for the external web page. The run time engine, however, continues executing.

If the mouse down event was to go to an external web page in the same browser window, then the run time engine terminates by turning control over to the designated external web page. In JAVA this can be accomplished with the "getAppletContext( ).showDocument(theURL)" method, where "theURL" is the URL address for the external web page.

If the viewer presses a mouse button while inside the bounding rectangle for a paragraph hot link at **260**, then appropriate four-dimensional status variables (web page by paragraph number, by paragraph line number, by paragraph line segment number) are set. If the hot link setting at FIG. **16** was to go to a different internal web page, then the run time engine sets the web page counter described previously in association with FIG. **31** for the "current" page to be one less than that of the desired web page. The current web page's object time lines, if any, complete normally, and the desired web page is then executed. If the hot link setting was to go to an external web page in a different window, then the run time engine creates a new browser window and loads the external web page. The run time engine, however, continues executing. If the hot link setting was to go to an external web page in the same browser window, then the run time engine terminates by turning control over to the designated external web page. This completes the detailed description of the run time process at **261**.

As is obvious from the descriptions of the various features and processes described above, it will be apparent to one skilled in the art that variations in form and detail may be made in the preferred implementation and methods without varying from the spirit and scope of the invention as defined in the claims or as interpreted under the doctrine of equivalents. It should also be clear that terms such as "browser", "mouse", "server", "web", etc., while adequate to describe the current state of interactive systems such as the World Wide Web, may evolve into new and more powerful entities. This evolution of terminology and technology is independent of the preferred implementation and methods of the invention as defined in the claims or as interpreted under the doctrine of equivalents. The preferred implementation and methods are thus provided for purposes of explanation and illustration, but not limitation.

I claim:

1. A method to allow users to produce Internet websites on and for computers having a browser and a virtual machine capable of generating displays, said method comprising:

(a) presenting a viewable menu having a user selectable panel of settings describing elements on a website, said panel of settings being presented through a browser on a computer adapted to accept one or more of said selectable settings in said panel as inputs therefrom, and where at least one of said user selectable settings in said panel corresponds to commands to said virtual machine;

(b) generating a display in accordance with one or more user selected settings substantially contemporaneously with the selection thereof;

(c) storing information representative of said one or more user selected settings in a database;

(d) generating a website at least in part by retrieving said information representative of said one or more user selected settings stored in said database; and

(e) building one or more web pages to generate said website from at least a portion of said database and at least one run time file, where said at least one run time file utilizes information stored in said database to generate virtual machine commands for the display of at least a portion of said one or more web pages.

2. An apparatus for producing Internet websites on and for computers having a browser and a virtual machine capable of generating displays, said apparatus comprising:

(a) an interface to present a viewable menu of a user selectable panel of settings to describe elements on a website, said panel of settings being presented through a browser on a computer adapted to accept one or more of said selectable settings in said panel as inputs therefrom, and where at least one of said user selectable settings in said panel corresponds to commands to said virtual machine;

(b) a browser to generate a display in accordance with one or more user selected settings substantially contemporaneously with the selection thereof;

(c) a database for storing information representative of said one or more user selected settings; and

(d) a build tool having at least one run time file for generating one or more web pages, said run time file operating to utilize information stored in said database to generate commands to said virtual machine for generating the display of at least a portion of said one or more web pages.

3. The apparatus of claim **2**, wherein said database is a multi-dimensional array structured database.

4. The apparatus of claim **3**, wherein said representative information is Boolean data, numeric data, string data or multi-dimensional arrays of various multimedia objects.

5. The apparatus of claim **4**, wherein said elements include multimedia objects selected from the group consisting of a color, a font, an image, an audio clip, a video clip, a text area and a URL.

6. The apparatus of claim **2**, wherein said elements are selected from the group consisting of a button, an image, a paragraph, a frame, a table, a form and a vector object.

7. The apparatus of claim **2**, wherein said one or more web pages of a website is two or more web pages of a website, wherein said elements include web pages, and wherein said description of elements is a transition or an animation between two of said two or more web pages.

8. The apparatus of claim **2**, wherein said elements include one or more objects on a web page, and wherein said description of elements are a transition or an animation of at least one of said elements on a web page.

9. The apparatus of claim **2**, wherein said elements include a button or an images, wherein said selectable settings includes the selection of an element style, and wherein said build engine includes means for storing information representative of selected style in said database.

10. The apparatus of claim **9**, wherein said elements are described by multiple object states.

11. The apparatus of claim **9**, wherein said elements are described by a transformation or a timelines of said selected styles.

12. The apparatus of claim **9**, wherein at least one of said elements is a child element, and wherein said element style is a child element style.

13. The apparatus of claim **12**, wherein at least one of said element is described by timelines of said child elements.

14. The apparatus of claim **2**, wherein said elements include buttons or images, wherein said description of elements is a transition or a timeline which is selected according to input from a mouse, and wherein said build engine includes means for storing information representative of said selected description of elements in said database.

15. The apparatus of claim **14**, wherein at least one of said description of elements is a timeline or an animation.

16. The apparatus of claim **2**, wherein said elements include one or more objects on two or more web page, wherein at least one of said description of elements is a transition or a timeline, and wherein said build engine includes means to synchronize said description of said one or more objects on one web page between said two or more web pages.

17. The apparatus of claim **2**, wherein one or more of said elements is a button or an image, wherein said description of elements is a transition, an animation or a timeline, and wherein said build engine includes means to synchronize said description of said one or more elements.

18. The apparatus of claim **2**, wherein at least one of said elements is an object elements selected from the group consisting of a button and an image, and web pages as elements of a website, wherein said description of said object elements is a transition, an animation or a timeline, wherein said description of said web pages is a transition or a timeline, and wherein said build engine includes means to synchronize said description of said object elements and said description of said web pages.

19. The apparatus of claim **2**, wherein said elements include an object and a child object, wherein said description of said elements is a timeline or an animation, and wherein said build engine activates said description of said elements according to input from a mouse.

20. The apparatus of claim **2**, wherein at least one of said elements is a child button or a child object, wherein said description of said elements is a timeline, a transition or an animation, and wherein said build engine includes means for defining said description of said element.

21. The apparatus of claim **17**, wherein said elements further includes at least two child object elements, and said means to synchronize includes means for synchronizing said description of at least two of said at least two child object elements.

22. The apparatus of claim **18**, wherein said elements further includes a child object element, and said means to synchronize includes means for synchronizing said description of said child elements.

23. The apparatus of claim **19**, wherein said description of elements is a transition or a timeline which is selected according to input from a mouse, and wherein said build engine includes means for storing information representative of said selected description in said database.

24. The apparatus of claim **2**, wherein said run time files include one compressed website specific, customized run time engine program file and one compressed website specific, customized run time engine library file.

25. The apparatus of claim **24**, wherein said run time files include a dynamic web page scaling mechanism, whereby each of said one or more generated web pages is scaled for viewing on said display.

26. The apparatus of claim **2**, wherein said run time files includes one compressed website specific, customized run time engine program file and one compressed website specific, customized run time engine library file, wherein said run time files include a dynamic web page scaling mechanism, and wherein the build engine generates an HTML shell file for each set of one or more run time files which instructs said browser to display a background in said browser window identical to that which the run time engine will draw, invokes said dynamic web page scaling mechanism, and invokes said run time engine.

27. The apparatus of claim **2**, wherein said run time files include a multi-level program animation model that presents multiple user interactions and time sensitive operations simultaneously.

28. The apparatus of claim **2**, wherein at least one of said elements includes individual ones of multiple audio tracks, and wherein said build engine includes means for associating at least one of said individual audio tracks with a web page or an object on a web page and means for synchronizing multiple audio tracks.

29. The apparatus of claim **2**, further including file size reduction means for reducing the total size of said run time files to less than approximately 50K and for reducing the total size of said the run time file of the first web page of said website to less than 25K.

30. The apparatus of claim **29**, wherein said total size of said run time files is from approximately 12K to approximately 50K.

31. The apparatus of claim **29**, wherein said elements include a web page, an object, a paragraph, or combinations thereof, and wherein said file size reduction means includes means for tracking multiple high water marks for said elements to minimize the size of said at least a portion of said database generated by said build tool.

32. The apparatus of claim **29**, wherein said file size reduction means includes means for identify website specific run time files.

33. The apparatus of claim **2**, wherein said website has a first web page, wherein said run time files includes a main run time file and main database information corresponding to said first web page, and further includes a run time routine for simultaneously obtaining from the Internet and displaying said first web page and processing said database corresponding to others of said web pages.

34. The apparatus of claim **33**, wherein at least one of said elements includes one or more images, and wherein the run time files includes an image observer operable to permit said one or more images of said website to be downloaded simultaneously with the execution of said run time files.

35. The apparatus of claim **2**, wherein the build engine includes dynamic resizing means operable to redefine a size of a web page upon being display.

36. The apparatus of claim **35**, wherein the dynamic resizing apparatus can be invoked in real time during the build process when a new web site file is opened, when the web page size of the existing web site is changed, or when the web page is zoomed to a different size.

37. An apparatus for producing Internet websites having one or more web pages on and for a computer having a browser and a virtual machine capable of generating a display, said apparatus comprising:

(a) an interface configured for building a website through control of website elements, said interface being operable through the browser on the computer to:

present a viewable menu of a user selectable panel of settings, accept a plurality of settings from said user selectable panel of settings to form an assembly of settings, and

generate the display in accordance with said assembly of settings contemporaneously with the acceptance thereof, at least one of said user selectable settings of said panel of settings being operable to generate said display through commands to said virtual machine;

(b) an internal database associated with said interface for storing information representative of one or more of said assembly of settings for controlling elements of the website; and

(c) a build tool to construct one or more web pages of said website having:

an external database containing data corresponding to said information stored in said internal database, and one or more run time files, where said run time files utilize information stored in said external database to generate virtual machine commands for the display of at least a portion of said one or more web pages.

38. The apparatus of claim **37**, wherein said user selectable panel of settings arc selected from the group consisting of construction, manipulation, animation and transition of elements.

39. An apparatus to allow users to produce Internet websites on and for computers having a browser and a virtual machine capable of generating displays, said apparatus comprising:

(a) means for presenting a viewable menu of a user selectable panel of settings to describe elements on a website, said panel of settings being presented through a browser on a computer adapted to accept one or more of said selectable settings in said panel as inputs

therefrom, and where at least one of said user selectable settings in said panel corresponds to commands to said virtual machine;

(b) means for generating a display in accordance with one or more user selected settings substantially contemporaneously with the selection thereof;

(c) means for storing information in a database, said information being representative of a plurality of said one or more user selected settings; and

(d) means for building one or more web pages for generating said website from at least a portion of said database and at least one run time file, where said at least one run time file utilizes information stored in said database to generate virtual machine commands for the display of at least a portion of said one or more web pages.

* * * * *

EXHIBIT B

US007594168B2

(12) **United States Patent**
Rempell

(10) **Patent No.: US 7,594,168 B2**
(45) **Date of Patent: Sep. 22, 2009**

(54) **BROWSER BASED WEB SITE GENERATION TOOL AND RUN TIME ENGINE**

(75) Inventor: **Steven H. Rempell**, Novato, CA (US)

(73) Assignee: **Akira Technologies, Inc.**

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1100 days.

(21) Appl. No.: **10/351,182**

(22) Filed: **Jan. 24, 2003**

(65) **Prior Publication Data**

US 2004/0148307 A1 Jul. 29, 2004

Related U.S. Application Data

(63) Continuation of application No. 09/454,061, filed on Dec. 2, 1999, now Pat. No. 6,546,397.

(51) **Int. Cl.**
G06F 17/00 (2006.01)

(52) **U.S. Cl.** **715/234**; 715/238; 715/762

(58) **Field of Classification Search** 715/234, 715/238, 762
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,428,731 A		6/1995	Powers, III	
5,842,020 A		11/1998	Faustini	
5,870,767 A		2/1999	Kraft, IV	
6,026,433 A	*	2/2000	D'Arlach et al.	709/217
6,081,263 A		6/2000	LeGall et al.	
6,083,276 A		7/2000	Davidson et al.	
6,148,311 A		11/2000	Wishnie et al.	
6,262,729 B1	*	7/2001	Marcos et al.	715/744
6,313,835 B1	*	11/2001	Gever et al.	715/846
6,424,979 B1	*	7/2002	Livingston et al.	715/206
6,585,779 B1	*	7/2003	Becker	715/237
6,675,382 B1	*	1/2004	Foster	717/177
6,684,369 B1	*	1/2004	Bernardo et al.	715/205
7,127,501 B1	*	10/2006	Beir et al.	709/219
7,152,207 B1	*	12/2006	Underwood et al.	715/207
2001/0011287 A1	*	8/2001	Goto et al.	707/513
2001/0042083 A1	*	11/2001	Saito et al.	707/517
2003/0058277 A1	*	3/2003	Bowman-Amuah	345/765
2005/0198087 A1	*	9/2005	Bremers	707/204
2005/0223320 A1	*	10/2005	Brintzenhofe et al.	715/517
2009/0094327 A1	*	4/2009	Shuster et al.	709/203

OTHER PUBLICATIONS

Tyler, Denise, Microsoft Frontpage 98, Nov. 1997, pp. 276-281, 321-327, 558-561.*
"Photo Album Applet Installation and Customization Instructions" © 1997 AgenX Corp, http://pages.prodigy.net/larzman/applets/phalbum.html.*

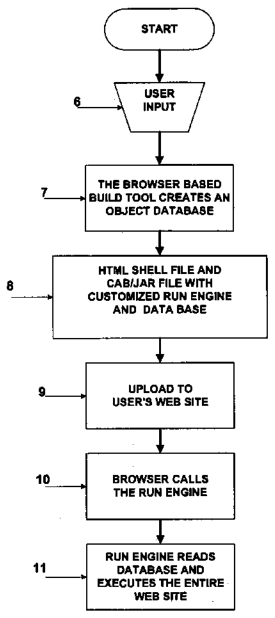
* cited by examiner

Primary Examiner—Adam M Queler
(74) *Attorney, Agent, or Firm*—Steven R. Vosen

(57) **ABSTRACT**

A method and apparatus for designing and building a web page. The apparatus includes a browser based build engine including build tools and a user interface. The build tools are operable to construct a single run time file and an associated database that describe, and when executed, produce the web page. The user interface includes a build frame and a panel. The build frame is operable to receive user input and present a WYSIWIG representation of the web page. The panel includes one or more menus for controlling the form of content to be placed on the web page.

6 Claims, 68 Drawing Sheets


START
6
USER INPUT
7
THE BROWSER BASED BUILD TOOL CREATES AN OBJECT DATABASE
8
HTML SHELL FILE AND CAB/JAR FILE WITH CUSTOMIZED RUN ENGINE AND DATA BASE
9
UPLOAD TO USER'S WEB SITE
10
BROWSER CALLS THE RUN ENGINE
11
RUN ENGINE READS DATABASE AND EXECUTES THE ENTIRE WEB SITE

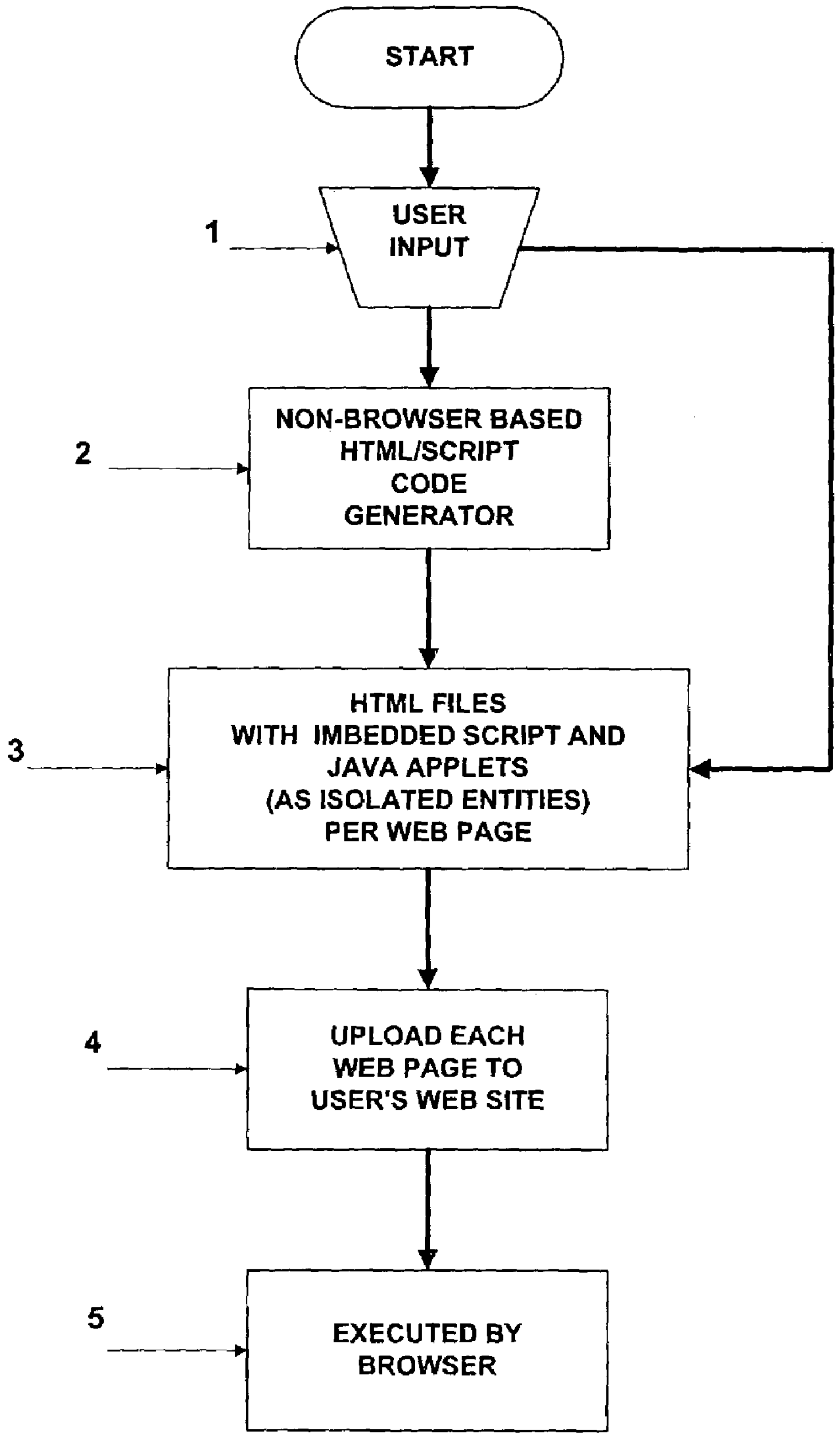

START
USER INPUT
1
NON-BROWSER BASED HTML/SCRIPT CODE GENERATOR
2
HTML FILES WITH IMBEDDED SCRIPT AND JAVA APPLETS (AS ISOLATED ENTITIES) PER WEB PAGE
3
UPLOAD EACH WEB PAGE TO USER'S WEB SITE
4
EXECUTED BY BROWSER
5

Fig. 1 PRIOR ART

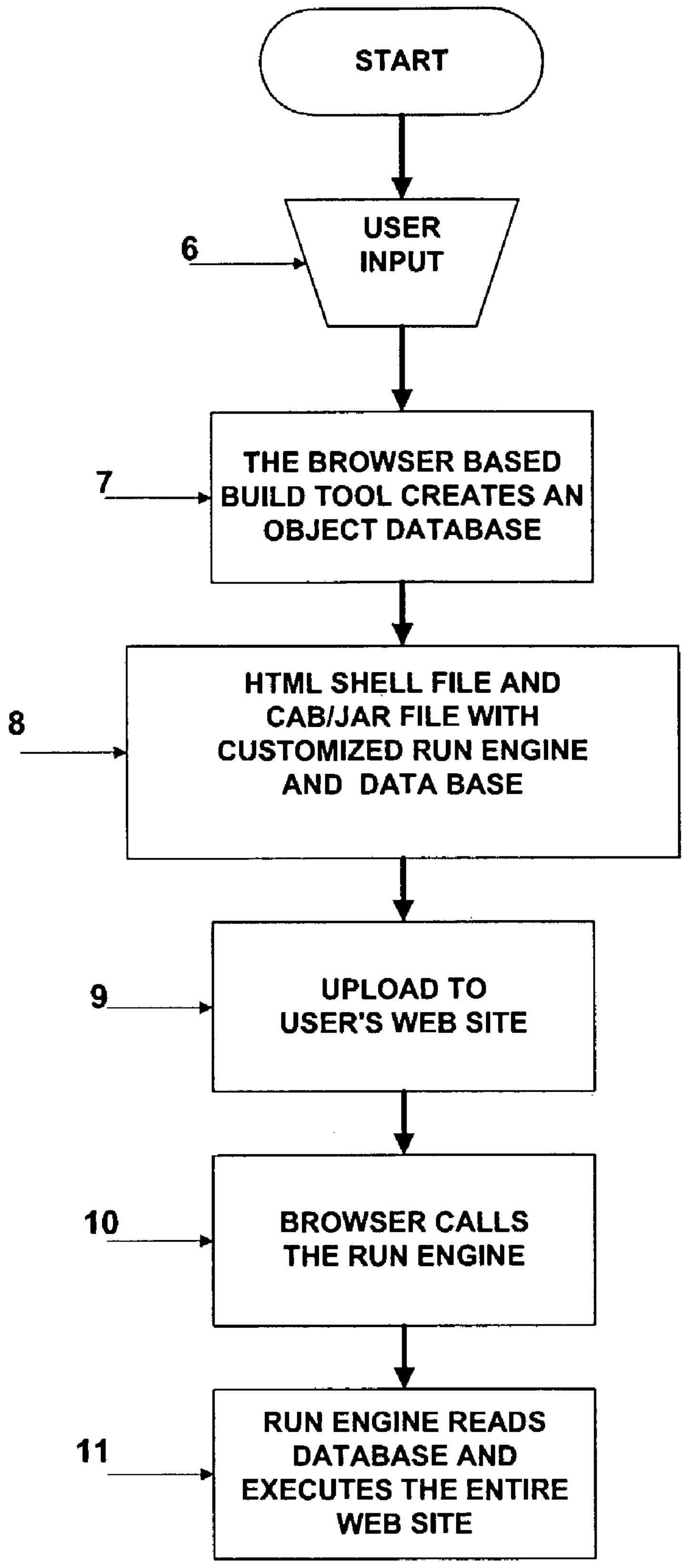

START
6
USER INPUT
7
THE BROWSER BASED BUILD TOOL CREATES AN OBJECT DATABASE
8
HTML SHELL FILE AND CAB/JAR FILE WITH CUSTOMIZED RUN ENGINE AND DATA BASE
9
UPLOAD TO USER'S WEB SITE
10
BROWSER CALLS THE RUN ENGINE
11
RUN ENGINE READS DATABASE AND EXECUTES THE ENTIRE WEB SITE

Fig. 2

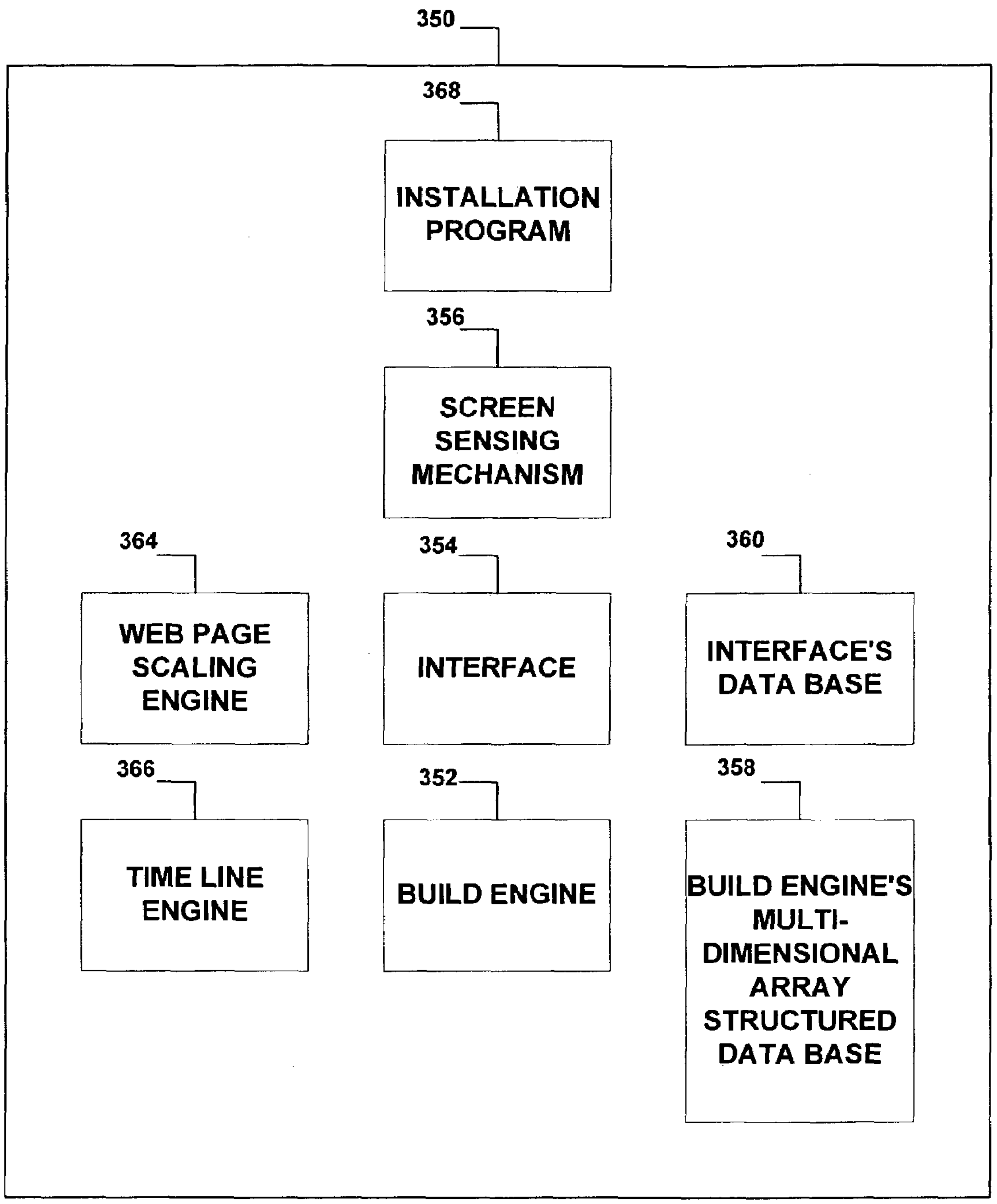


Fig. 3a

BUILD TOOL COMPONENTS

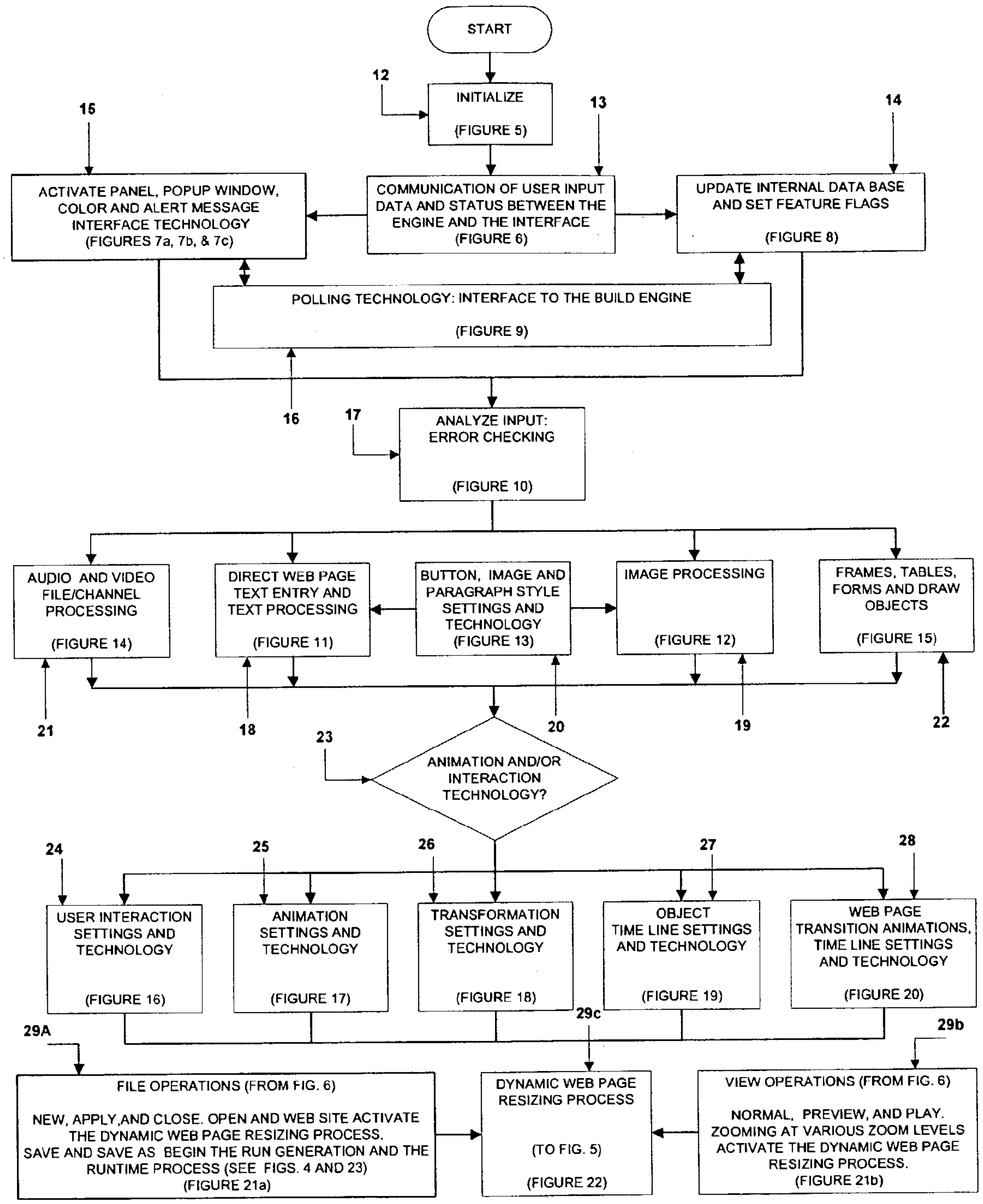

START
12
INITIALIZE (FIGURE 5)
13
14
15
ACTIVATE PANEL, POPUP WINDOW, COLOR AND ALERT MESSAGE INTERFACE TECHNOLOGY (FIGURES 7a, 7b, & 7c)
COMMUNICATION OF USER INPUT DATA AND STATUS BETWEEN THE ENGINE AND THE INTERFACE (FIGURE 6)
UPDATE INTERNAL DATA BASE AND SET FEATURE FLAGS (FIGURE 8)
POLLING TECHNOLOGY: INTERFACE TO THE BUILD ENGINE (FIGURE 9)
16
17
ANALYZE INPUT: ERROR CHECKING (FIGURE 10)
AUDIO AND VIDEO FILE/CHANNEL PROCESSING (FIGURE 14)
DIRECT WEB PAGE TEXT ENTRY AND TEXT PROCESSING (FIGURE 11)
BUTTON, IMAGE AND PARAGRAPH STYLE SETTINGS AND TECHNOLOGY (FIGURE 13)
IMAGE PROCESSING (FIGURE 12)
FRAMES, TABLES, FORMS AND DRAW OBJECTS (FIGURE 15)
21
18
23
20
19
22
ANIMATION AND/OR INTERACTION TECHNOLOGY?
24
25
26
27
28
USER INTERACTION SETTINGS AND TECHNOLOGY (FIGURE 16)
ANIMATION SETTINGS AND TECHNOLOGY (FIGURE 17)
TRANSFORMATION SETTINGS AND TECHNOLOGY (FIGURE 18)
OBJECT TIME LINE SETTINGS AND TECHNOLOGY (FIGURE 19)
WEB PAGE TRANSITION ANIMATIONS, TIME LINE SETTINGS AND TECHNOLOGY (FIGURE 20)
29A
29c
29b
FILE OPERATIONS (FROM FIG. 6) NEW, APPLY,AND CLOSE. OPEN AND WEB SITE ACTIVATE THE DYNAMIC WEB PAGE RESIZING PROCESS. SAVE AND SAVE AS BEGIN THE RUN GENERATION AND THE RUNTIME PROCESS (SEE FIGS. 4 AND 23) (FIGURE 21a)
DYNAMIC WEB PAGE RESIZING PROCESS (TO FIG. 5) (FIGURE 22)
VIEW OPERATIONS (FROM FIG. 6) NORMAL, PREVIEW, AND PLAY. ZOOMING AT VARIOUS ZOOM LEVELS ACTIVATE THE DYNAMIC WEB PAGE RESIZING PROCESS. (FIGURE 21b)

Fig. 3b **THE BUILD TOOL & BUILD PROCESS**

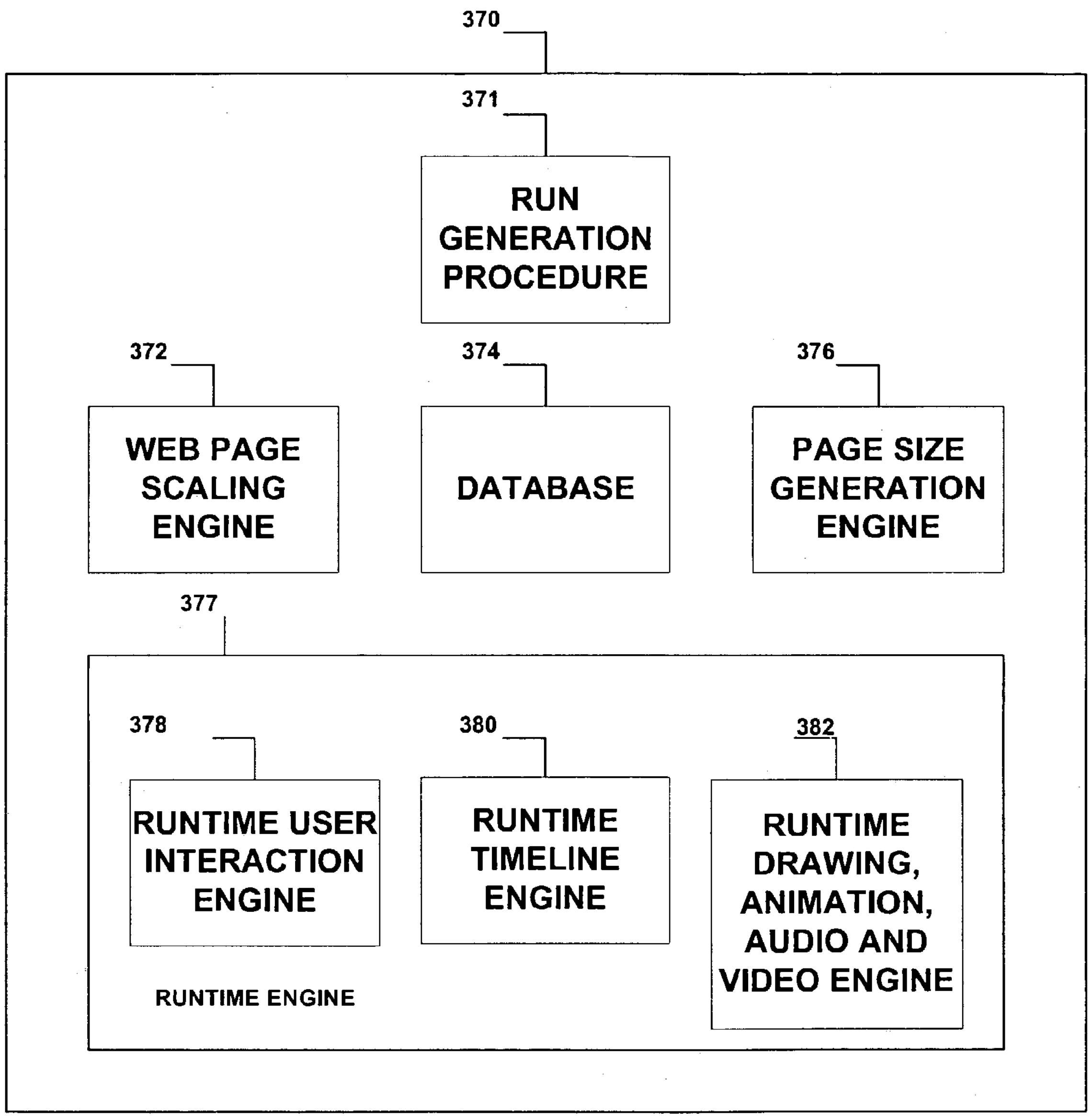


Fig. 4a

RUN GENERATION AND RUNTIME COMPONENTS

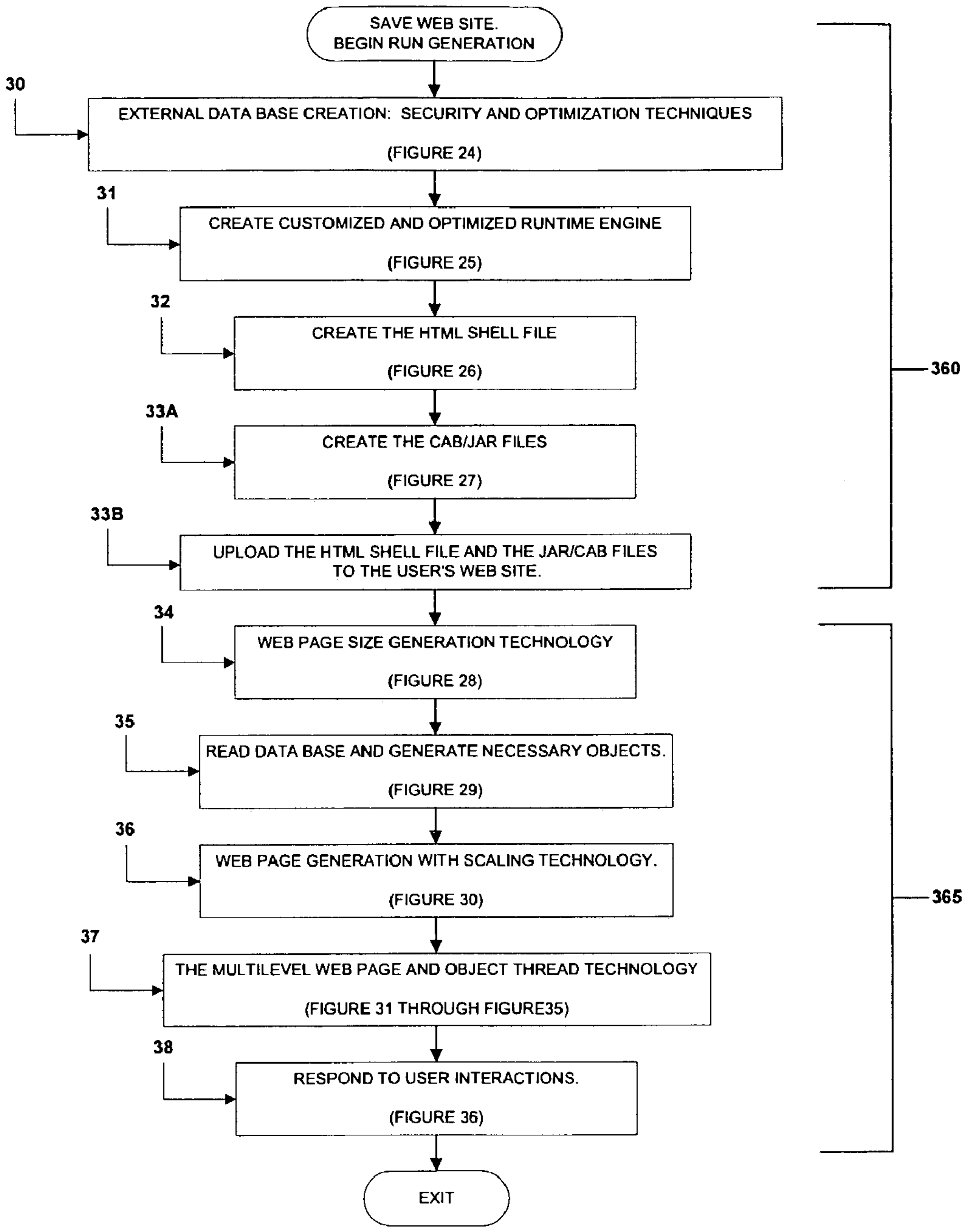

SAVE WEB SITE. BEGIN RUN GENERATION
30
EXTERNAL DATA BASE CREATION: SECURITY AND OPTIMIZATION TECHNIQUES (FIGURE 24)
31
CREATE CUSTOMIZED AND OPTIMIZED RUNTIME ENGINE (FIGURE 25)
32
CREATE THE HTML SHELL FILE (FIGURE 26)
33A
CREATE THE CAB/JAR FILES (FIGURE 27)
33B
UPLOAD THE HTML SHELL FILE AND THE JAR/CAB FILES TO THE USER'S WEB SITE.
360
34
WEB PAGE SIZE GENERATION TECHNOLOGY (FIGURE 28)
35
READ DATA BASE AND GENERATE NECESSARY OBJECTS. (FIGURE 29)
36
WEB PAGE GENERATION WITH SCALING TECHNOLOGY. (FIGURE 30)
37
THE MULTILEVEL WEB PAGE AND OBJECT THREAD TECHNOLOGY (FIGURE 31 THROUGH FIGURE35)
38
RESPOND TO USER INTERACTIONS. (FIGURE 36)
365
EXIT

Fig. 4b

RUN GENERATION & THE RUNTIME PROCESS

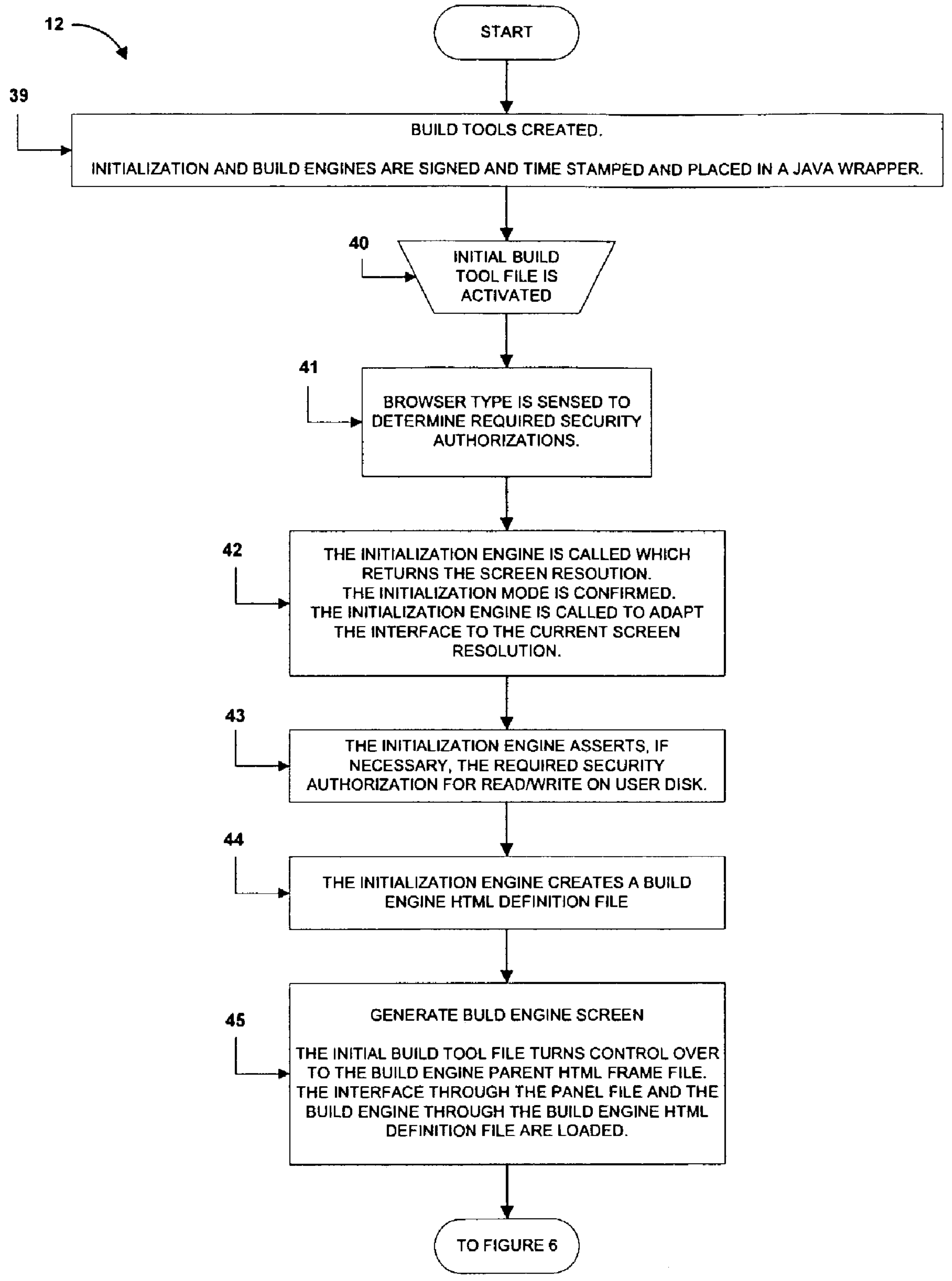

12
START
39
BUILD TOOLS CREATED.
INITIALIZATION AND BUILD ENGINES ARE SIGNED AND TIME STAMPED AND PLACED IN A JAVA WRAPPER.
40
INITIAL BUILD TOOL FILE IS ACTIVATED
41
BROWSER TYPE IS SENSED TO DETERMINE REQUIRED SECURITY AUTHORIZATIONS.
42
THE INITIALIZATION ENGINE IS CALLED WHICH RETURNS THE SCREEN RESOUTION. THE INITIALIZATION MODE IS CONFIRMED. THE INITIALIZATION ENGINE IS CALLED TO ADAPT THE INTERFACE TO THE CURRENT SCREEN RESOLUTION.
43
THE INITIALIZATION ENGINE ASSERTS, IF NECESSARY, THE REQUIRED SECURITY AUTHORIZATION FOR READ/WRITE ON USER DISK.
44
THE INITIALIZATION ENGINE CREATES A BUILD ENGINE HTML DEFINITION FILE
45
GENERATE BULD ENGINE SCREEN
THE INITIAL BUILD TOOL FILE TURNS CONTROL OVER TO THE BUILD ENGINE PARENT HTML FRAME FILE. THE INTERFACE THROUGH THE PANEL FILE AND THE BUILD ENGINE THROUGH THE BUILD ENGINE HTML DEFINITION FILE ARE LOADED.
TO FIGURE 6

Fig . 5 **INITIALIZATION**

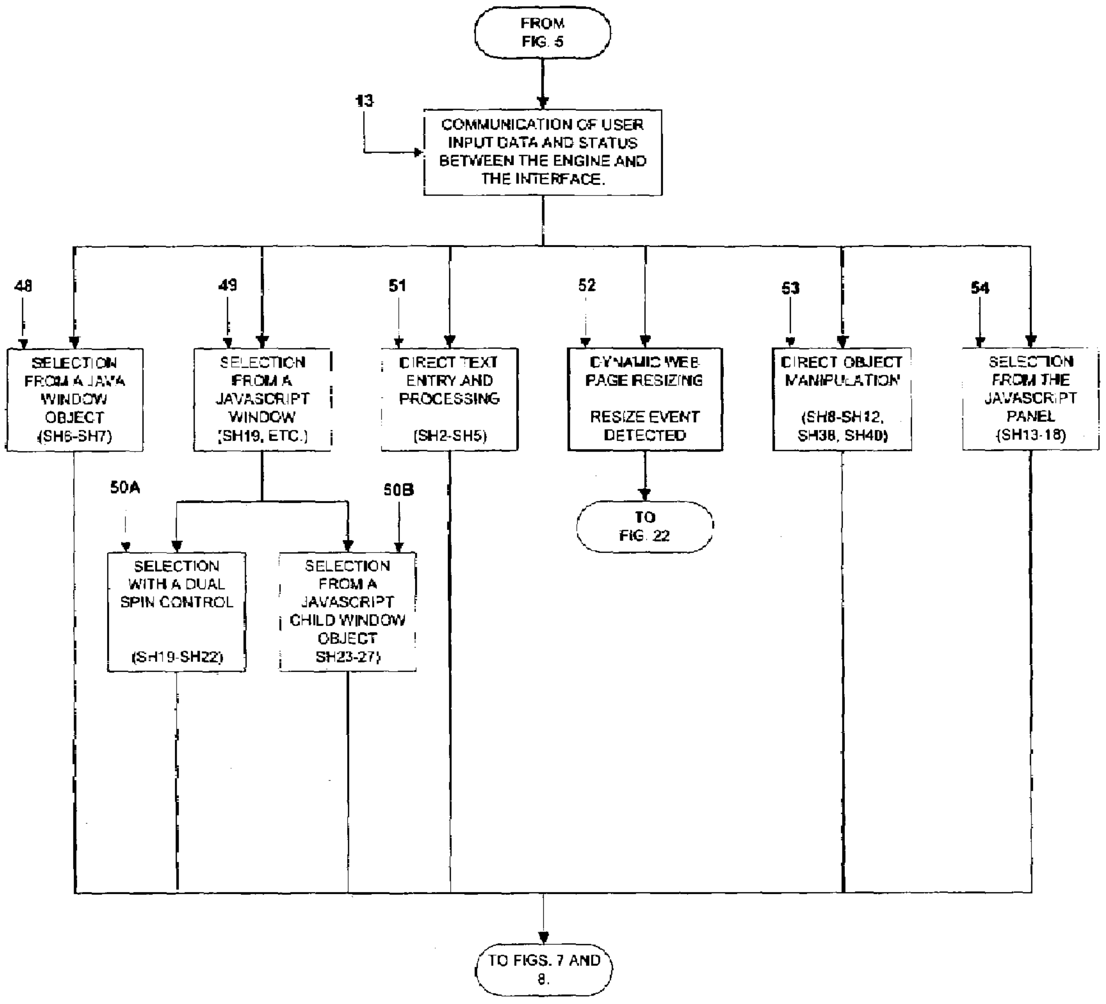

FROM FIG. 5
13
COMMUNICATION OF USER INPUT DATA AND STATUS BETWEEN THE ENGINE AND THE INTERFACE.
48
SELECTION FROM A JAVA WINDOW OBJECT (SH6-SH7)
49
SELECTION FROM A JAVASCRIPT WINDOW (SH19, ETC.)
51
DIRECT TEXT ENTRY AND PROCESSING (SH2-SH5)
52
DYNAMIC WEB PAGE RESIZING RESIZE EVENT DETECTED
53
DIRECT OBJECT MANIPULATION (SH8-SH12, SH38, SH40)
54
SELECTION FROM THE JAVASCRIPT PANEL (SH13-18)
50A
SELECTION WITH A DUAL SPIN CONTROL (SH19-SH22)
50B
SELECTION FROM A JAVASCRIPT CHILD WINDOW OBJECT SH23-27)
TO FIG. 22
TO FIGS. 7 AND 8.

Fig. 6

COMMUNICATION OF USER INPUT DATA AND STATUS BETWEEN THE ENGINE AND THE INTERFACE.

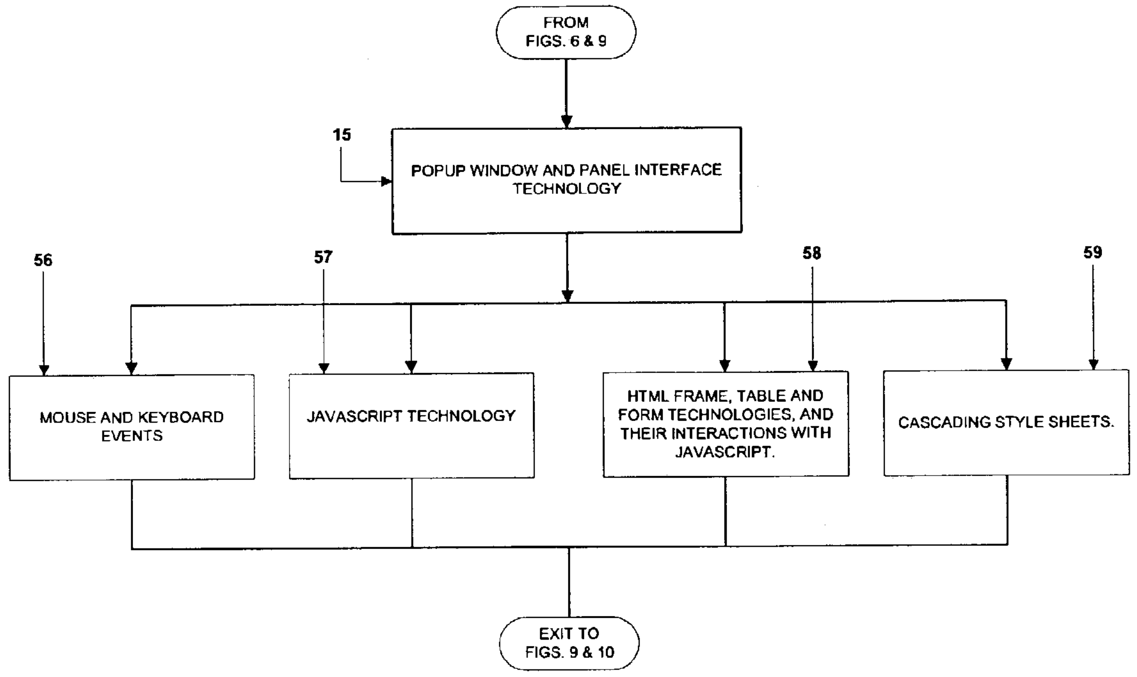

FROM FIGS. 6 & 9
15
POPUP WINDOW AND PANEL INTERFACE TECHNOLOGY
56
57
58
59
MOUSE AND KEYBOARD EVENTS
JAVASCRIPT TECHNOLOGY
HTML FRAME, TABLE AND FORM TECHNOLOGIES, AND THEIR INTERACTIONS WITH JAVASCRIPT.
CASCADING STYLE SHEETS.
EXIT TO FIGS. 9 & 10

Fig. 7a

POPUP WINDOW AND PANEL INTERFACE AND COLOR TECHNOLOGY

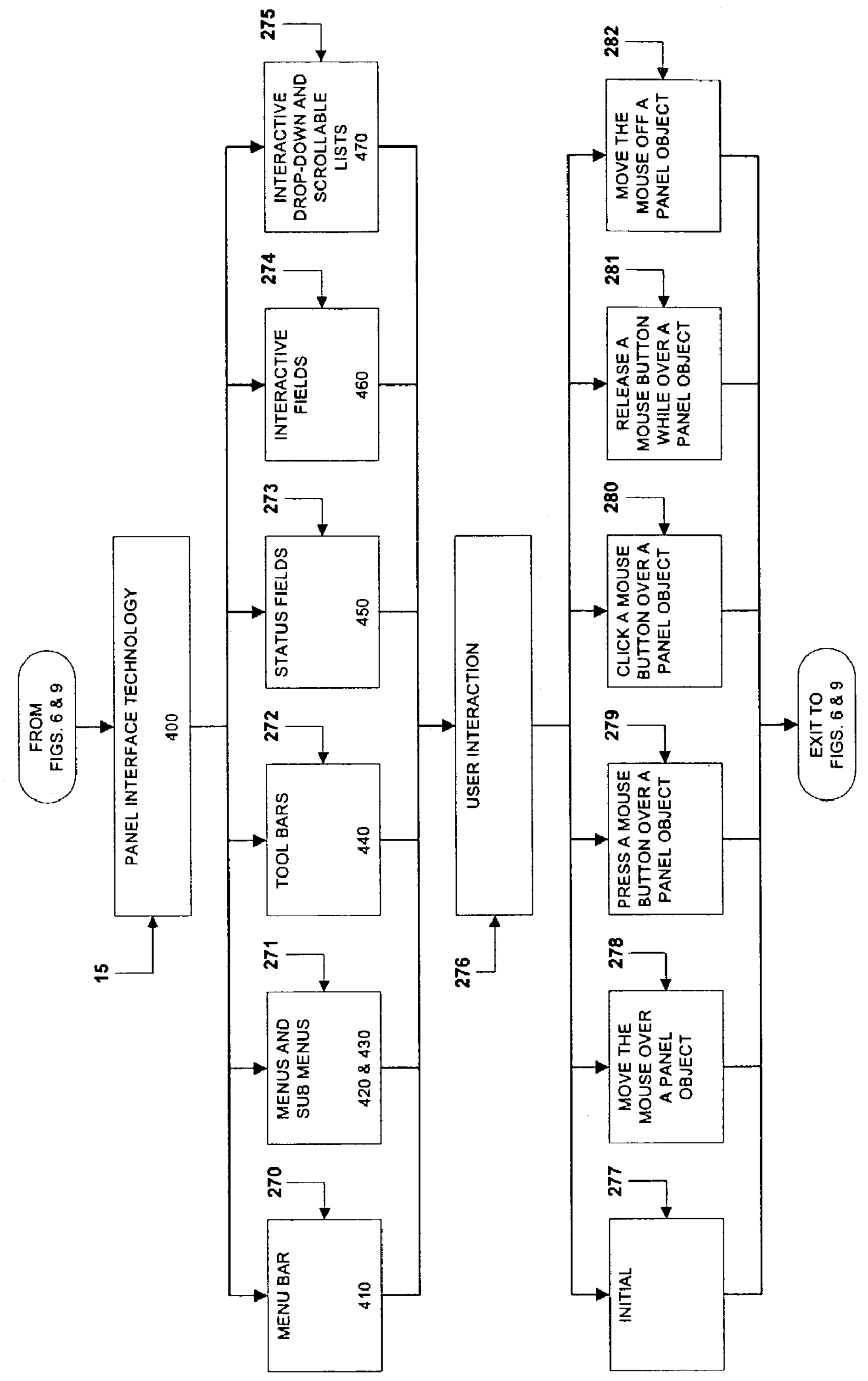

FROM FIGS. 6 & 9
15
PANEL INTERFACE TECHNOLOGY 400
270
MENU BAR 410
271
MENUS AND SUB MENUS 420 & 430
272
TOOL BARS 440
273
STATUS FIELDS 450
274
INTERACTIVE FIELDS 460
275
INTERACTIVE DROP-DOWN AND SCROLLABLE LISTS 470
276
USER INTERACTION
277
INITIAL
278
MOVE THE MOUSE OVER A PANEL OBJECT
279
PRESS A MOUSE BUTTON OVER A PANEL OBJECT
280
CLICK A MOUSE BUTTON OVER A PANEL OBJECT
281
RELEASE A MOUSE BUTTON WHILE OVER A PANEL OBJECT
282
MOVE THE MOUSE OFF A PANEL OBJECT
EXIT TO FIGS. 6 & 9

Fig. 7b

IMPLEMENTATION OF PANEL INTERFACE OBJECTS

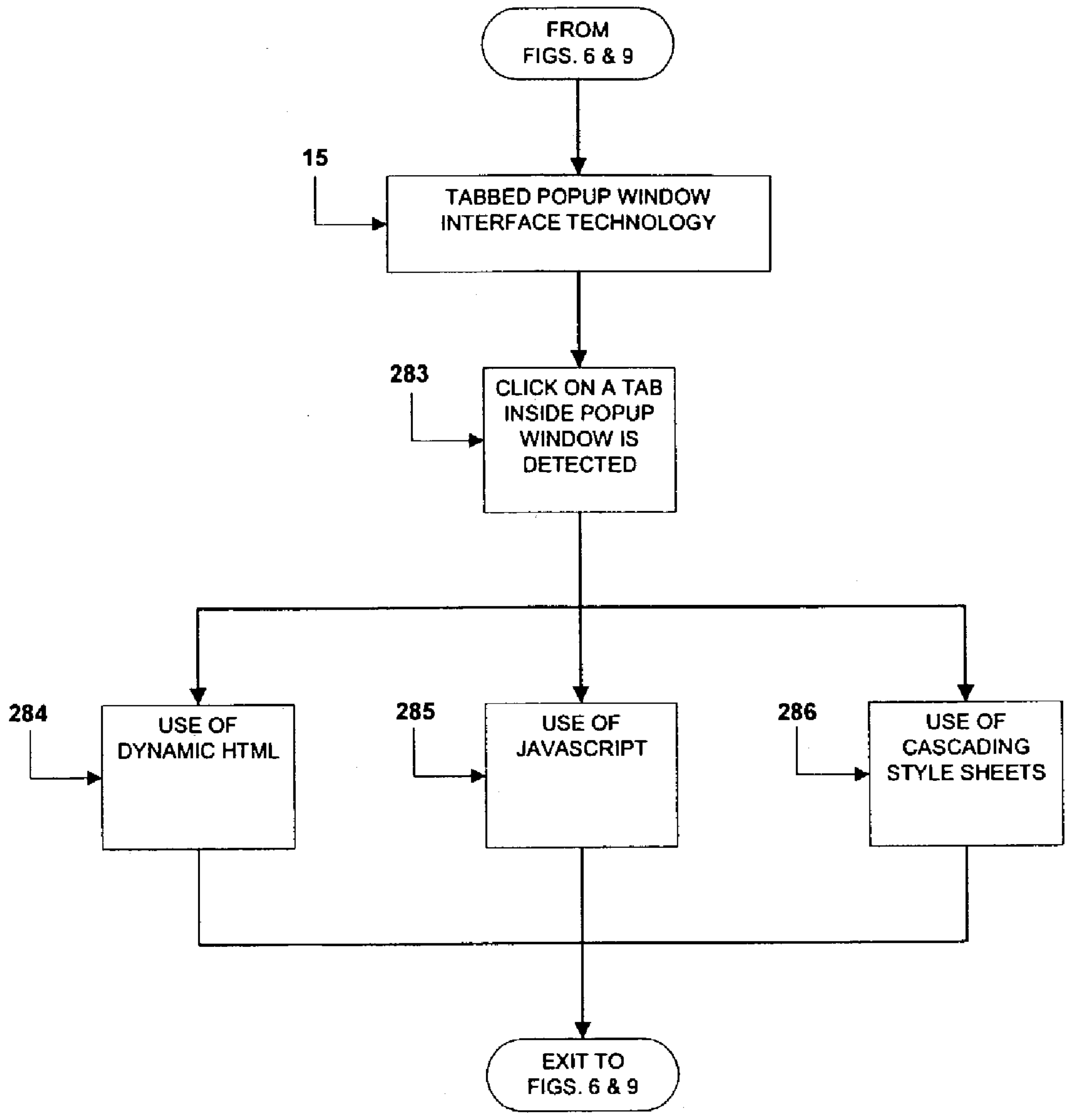

FROM FIGS. 6 & 9
15
TABBED POPUP WINDOW INTERFACE TECHNOLOGY
283
CLICK ON A TAB INSIDE POPUP WINDOW IS DETECTED
284
USE OF DYNAMIC HTML
285
USE OF JAVASCRIPT
286
USE OF CASCADING STYLE SHEETS
EXIT TO FIGS. 6 & 9

Fig. 7c

IMPLEMENTATION OF TABBED POPUP WINDOWS

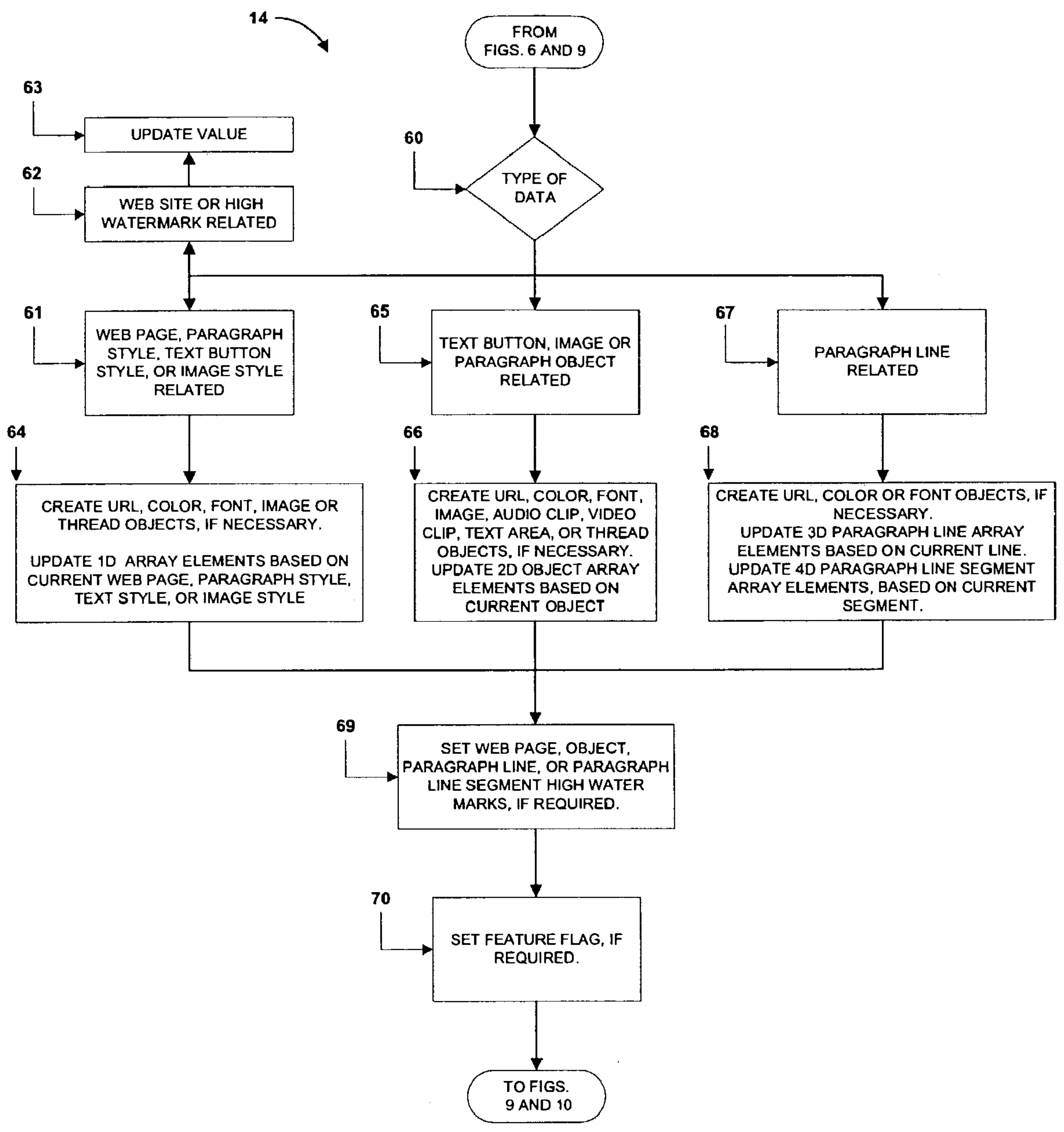

14
FROM FIGS. 6 AND 9
60
TYPE OF DATA
63
UPDATE VALUE
62
WEB SITE OR HIGH WATERMARK RELATED
61
WEB PAGE, PARAGRAPH STYLE, TEXT BUTTON STYLE, OR IMAGE STYLE RELATED
65
TEXT BUTTON, IMAGE OR PARAGRAPH OBJECT RELATED
67
PARAGRAPH LINE RELATED
64
CREATE URL, COLOR, FONT, IMAGE OR THREAD OBJECTS, IF NECESSARY.
UPDATE 1D ARRAY ELEMENTS BASED ON CURRENT WEB PAGE, PARAGRAPH STYLE, TEXT STYLE, OR IMAGE STYLE
66
CREATE URL, COLOR, FONT, IMAGE, AUDIO CLIP, VIDEO CLIP, TEXT AREA, OR THREAD OBJECTS, IF NECESSARY.
UPDATE 2D OBJECT ARRAY ELEMENTS BASED ON CURRENT OBJECT
68
CREATE URL, COLOR OR FONT OBJECTS, IF NECESSARY.
UPDATE 3D PARAGRAPH LINE ARRAY ELEMENTS BASED ON CURRENT LINE.
UPDATE 4D PARAGRAPH LINE SEGMENT ARRAY ELEMENTS, BASED ON CURRENT SEGMENT.
69
SET WEB PAGE, OBJECT, PARAGRAPH LINE, OR PARAGRAPH LINE SEGMENT HIGH WATER MARKS, IF REQUIRED.
70
SET FEATURE FLAG, IF REQUIRED.
TO FIGS. 9 AND 10

Fig. 8

UPDATE INTERNAL DATA BASE AND SET FEATURE FLAGS

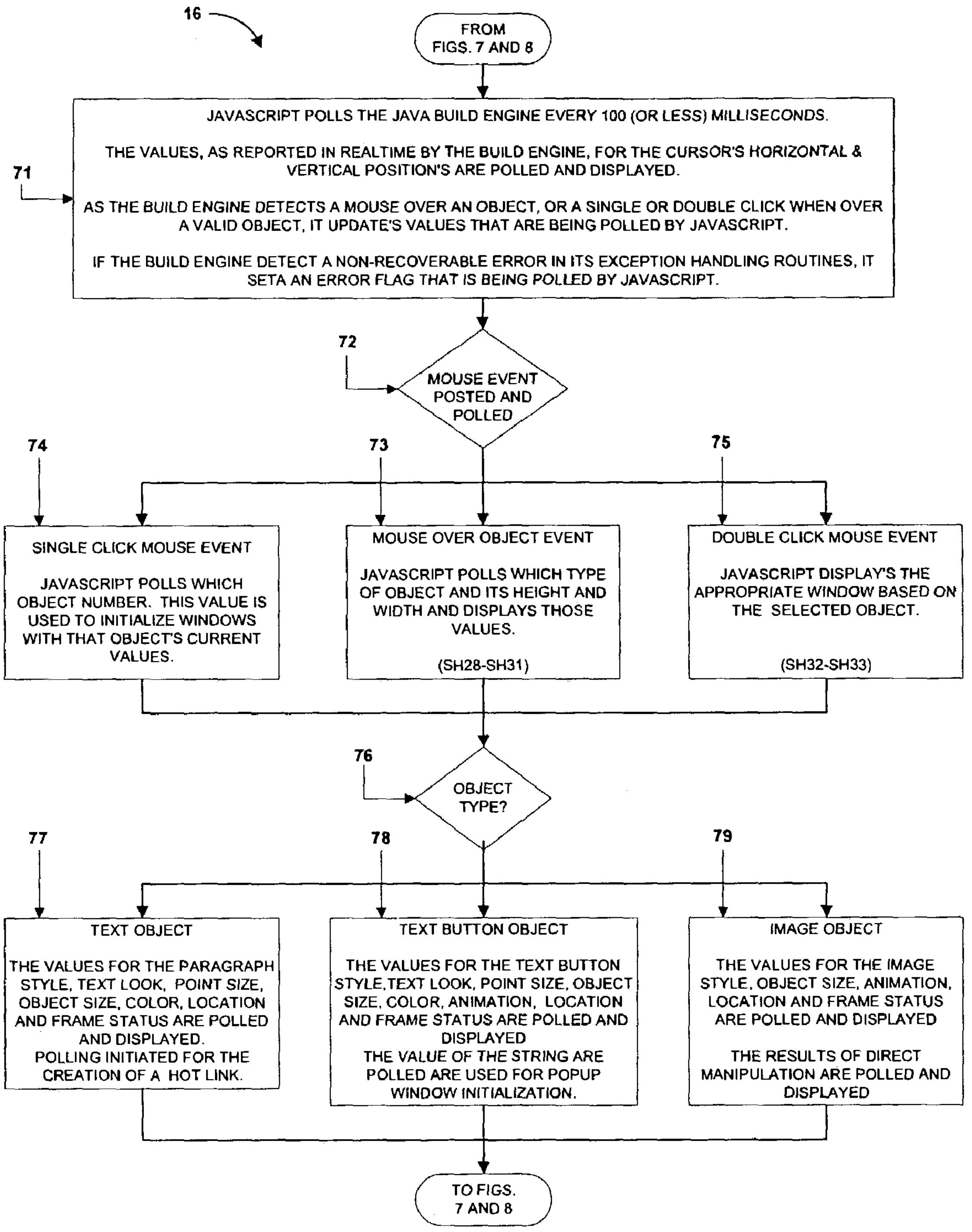

16
FROM FIGS. 7 AND 8
71
JAVASCRIPT POLLS THE JAVA BUILD ENGINE EVERY 100 (OR LESS) MILLISECONDS.
THE VALUES, AS REPORTED IN REALTIME BY THE BUILD ENGINE, FOR THE CURSOR'S HORIZONTAL & VERTICAL POSITION'S ARE POLLED AND DISPLAYED.
AS THE BUILD ENGINE DETECTS A MOUSE OVER AN OBJECT, OR A SINGLE OR DOUBLE CLICK WHEN OVER A VALID OBJECT, IT UPDATE'S VALUES THAT ARE BEING POLLED BY JAVASCRIPT.
IF THE BUILD ENGINE DETECT A NON-RECOVERABLE ERROR IN ITS EXCEPTION HANDLING ROUTINES, IT SETA AN ERROR FLAG THAT IS BEING POLLED BY JAVASCRIPT.
72
MOUSE EVENT POSTED AND POLLED
74
73
75
SINGLE CLICK MOUSE EVENT
JAVASCRIPT POLLS WHICH OBJECT NUMBER. THIS VALUE IS USED TO INITIALIZE WINDOWS WITH THAT OBJECT'S CURRENT VALUES.
MOUSE OVER OBJECT EVENT
JAVASCRIPT POLLS WHICH TYPE OF OBJECT AND ITS HEIGHT AND WIDTH AND DISPLAYS THOSE VALUES.
(SH28-SH31)
DOUBLE CLICK MOUSE EVENT
JAVASCRIPT DISPLAY'S THE APPROPRIATE WINDOW BASED ON THE SELECTED OBJECT.
(SH32-SH33)
76
OBJECT TYPE?
77
78
79
TEXT OBJECT
THE VALUES FOR THE PARAGRAPH STYLE, TEXT LOOK, POINT SIZE, OBJECT SIZE, COLOR, LOCATION AND FRAME STATUS ARE POLLED AND DISPLAYED.
POLLING INITIATED FOR THE CREATION OF A HOT LINK.
TEXT BUTTON OBJECT
THE VALUES FOR THE TEXT BUTTON STYLE, TEXT LOOK, POINT SIZE, OBJECT SIZE, COLOR, ANIMATION, LOCATION AND FRAME STATUS ARE POLLED AND DISPLAYED
THE VALUE OF THE STRING ARE POLLED ARE USED FOR POPUP WINDOW INITIALIZATION.
IMAGE OBJECT
THE VALUES FOR THE IMAGE STYLE, OBJECT SIZE, ANIMATION, LOCATION AND FRAME STATUS ARE POLLED AND DISPLAYED
THE RESULTS OF DIRECT MANIPULATION ARE POLLED AND DISPLAYED
TO FIGS. 7 AND 8

Fig. 9 **POLLING METHODS**

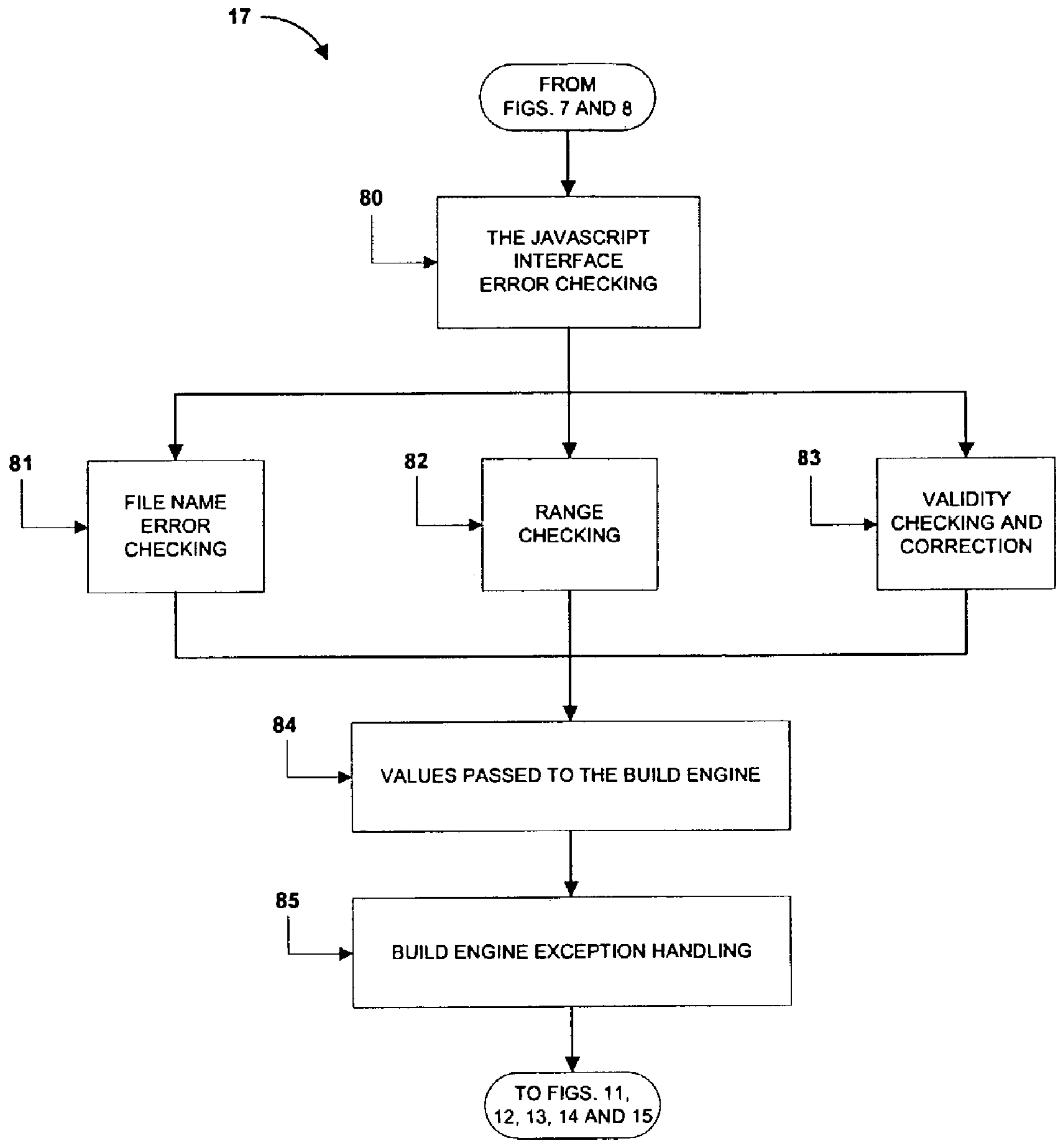

17
FROM FIGS. 7 AND 8
80
THE JAVASCRIPT INTERFACE ERROR CHECKING
81
FILE NAME ERROR CHECKING
82
RANGE CHECKING
83
VALIDITY CHECKING AND CORRECTION
84
VALUES PASSED TO THE BUILD ENGINE
85
BUILD ENGINE EXCEPTION HANDLING
TO FIGS. 11, 12, 13, 14 AND 15

Fig. 10

ANALYZE INPUT: ERROR CHECKING

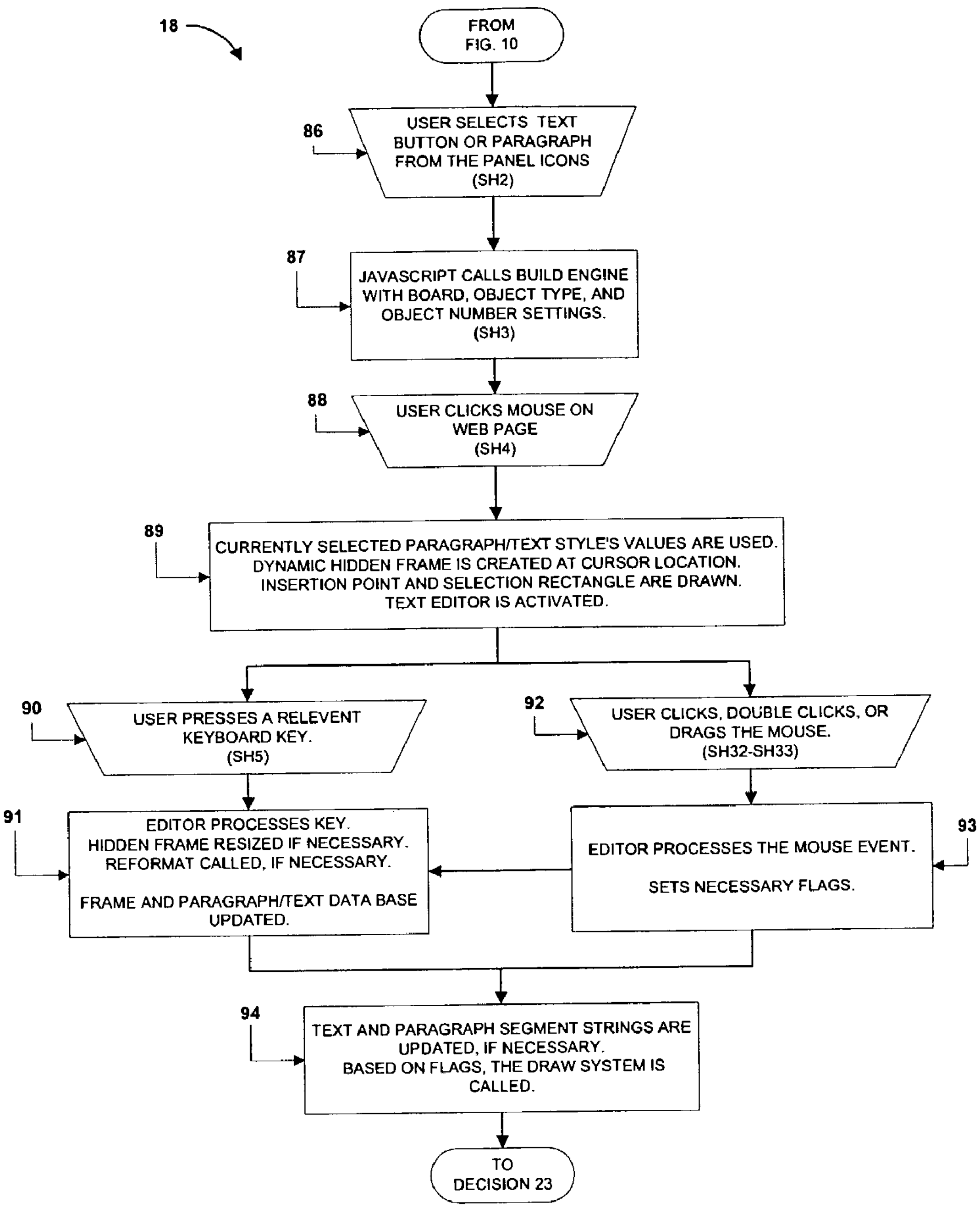

18
FROM FIG. 10
86
USER SELECTS TEXT BUTTON OR PARAGRAPH FROM THE PANEL ICONS (SH2)
87
JAVASCRIPT CALLS BUILD ENGINE WITH BOARD, OBJECT TYPE, AND OBJECT NUMBER SETTINGS. (SH3)
88
USER CLICKS MOUSE ON WEB PAGE (SH4)
89
CURRENTLY SELECTED PARAGRAPH/TEXT STYLE'S VALUES ARE USED. DYNAMIC HIDDEN FRAME IS CREATED AT CURSOR LOCATION. INSERTION POINT AND SELECTION RECTANGLE ARE DRAWN. TEXT EDITOR IS ACTIVATED.
90
USER PRESSES A RELEVENT KEYBOARD KEY. (SH5)
92
USER CLICKS, DOUBLE CLICKS, OR DRAGS THE MOUSE. (SH32-SH33)
91
EDITOR PROCESSES KEY. HIDDEN FRAME RESIZED IF NECESSARY. REFORMAT CALLED, IF NECESSARY. FRAME AND PARAGRAPH/TEXT DATA BASE UPDATED.
93
EDITOR PROCESSES THE MOUSE EVENT. SETS NECESSARY FLAGS.
94
TEXT AND PARAGRAPH SEGMENT STRINGS ARE UPDATED, IF NECESSARY. BASED ON FLAGS, THE DRAW SYSTEM IS CALLED.
TO DECISION 23

Fig. 11

DIRECT WEB PAGE DATA ENTRY AND TEXT PROCESSING

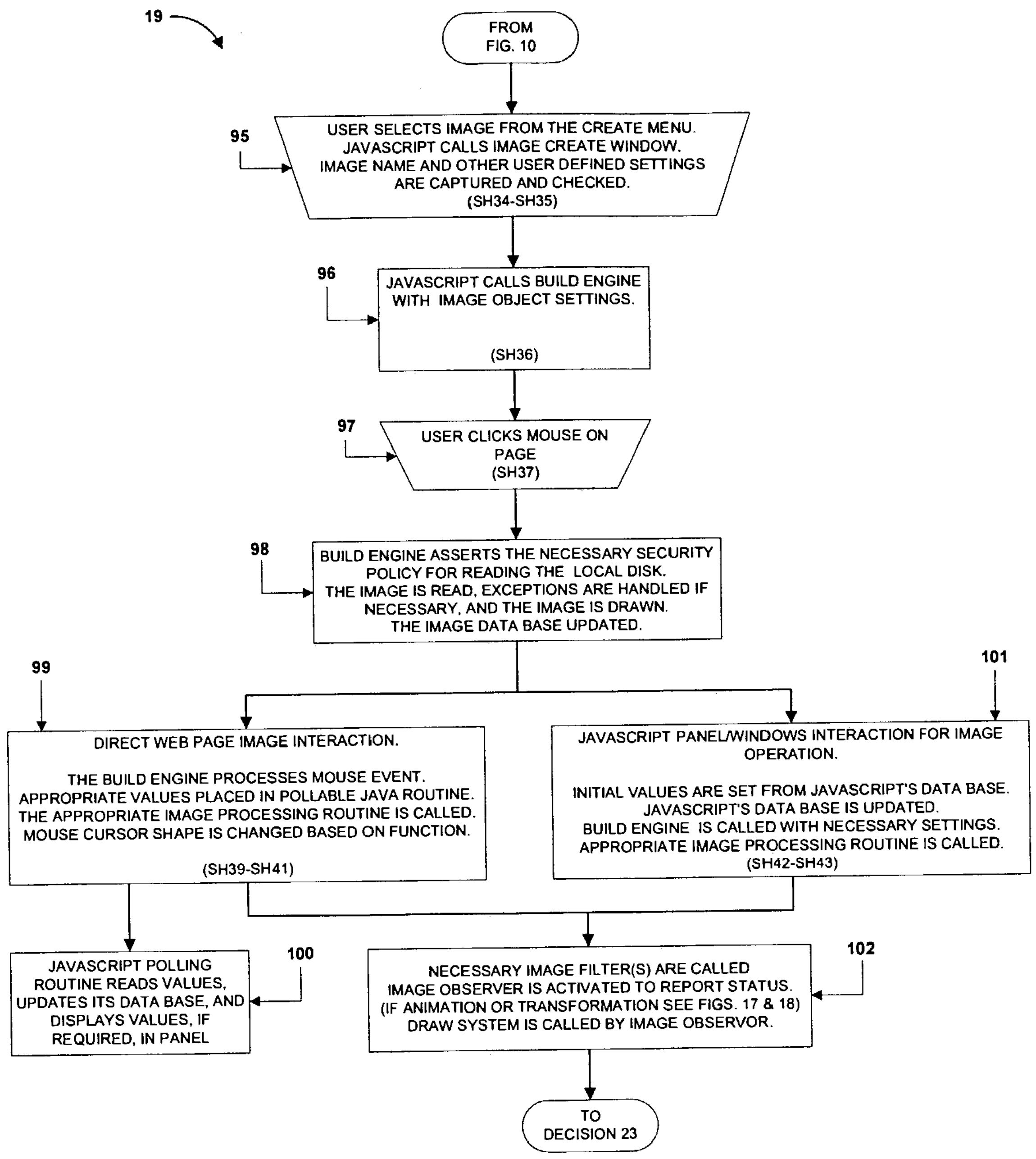

19
FROM FIG. 10
95
USER SELECTS IMAGE FROM THE CREATE MENU. JAVASCRIPT CALLS IMAGE CREATE WINDOW. IMAGE NAME AND OTHER USER DEFINED SETTINGS ARE CAPTURED AND CHECKED. (SH34-SH35)
96
JAVASCRIPT CALLS BUILD ENGINE WITH IMAGE OBJECT SETTINGS. (SH36)
97
USER CLICKS MOUSE ON PAGE (SH37)
98
BUILD ENGINE ASSERTS THE NECESSARY SECURITY POLICY FOR READING THE LOCAL DISK. THE IMAGE IS READ, EXCEPTIONS ARE HANDLED IF NECESSARY, AND THE IMAGE IS DRAWN. THE IMAGE DATA BASE UPDATED.
99
DIRECT WEB PAGE IMAGE INTERACTION. THE BUILD ENGINE PROCESSES MOUSE EVENT. APPROPRIATE VALUES PLACED IN POLLABLE JAVA ROUTINE. THE APPROPRIATE IMAGE PROCESSING ROUTINE IS CALLED. MOUSE CURSOR SHAPE IS CHANGED BASED ON FUNCTION. (SH39-SH41)
101
JAVASCRIPT PANEL/WINDOWS INTERACTION FOR IMAGE OPERATION. INITIAL VALUES ARE SET FROM JAVASCRIPT'S DATA BASE. JAVASCRIPT'S DATA BASE IS UPDATED. BUILD ENGINE IS CALLED WITH NECESSARY SETTINGS. APPROPRIATE IMAGE PROCESSING ROUTINE IS CALLED. (SH42-SH43)
JAVASCRIPT POLLING ROUTINE READS VALUES, UPDATES ITS DATA BASE, AND DISPLAYS VALUES, IF REQUIRED, IN PANEL
100
NECESSARY IMAGE FILTER(S) ARE CALLED IMAGE OBSERVER IS ACTIVATED TO REPORT STATUS. (IF ANIMATION OR TRANSFORMATION SEE FIGS. 17 & 18) DRAW SYSTEM IS CALLED BY IMAGE OBSERVOR.
102
TO DECISION 23

Fig. 12 **IMAGE PROCESSING**

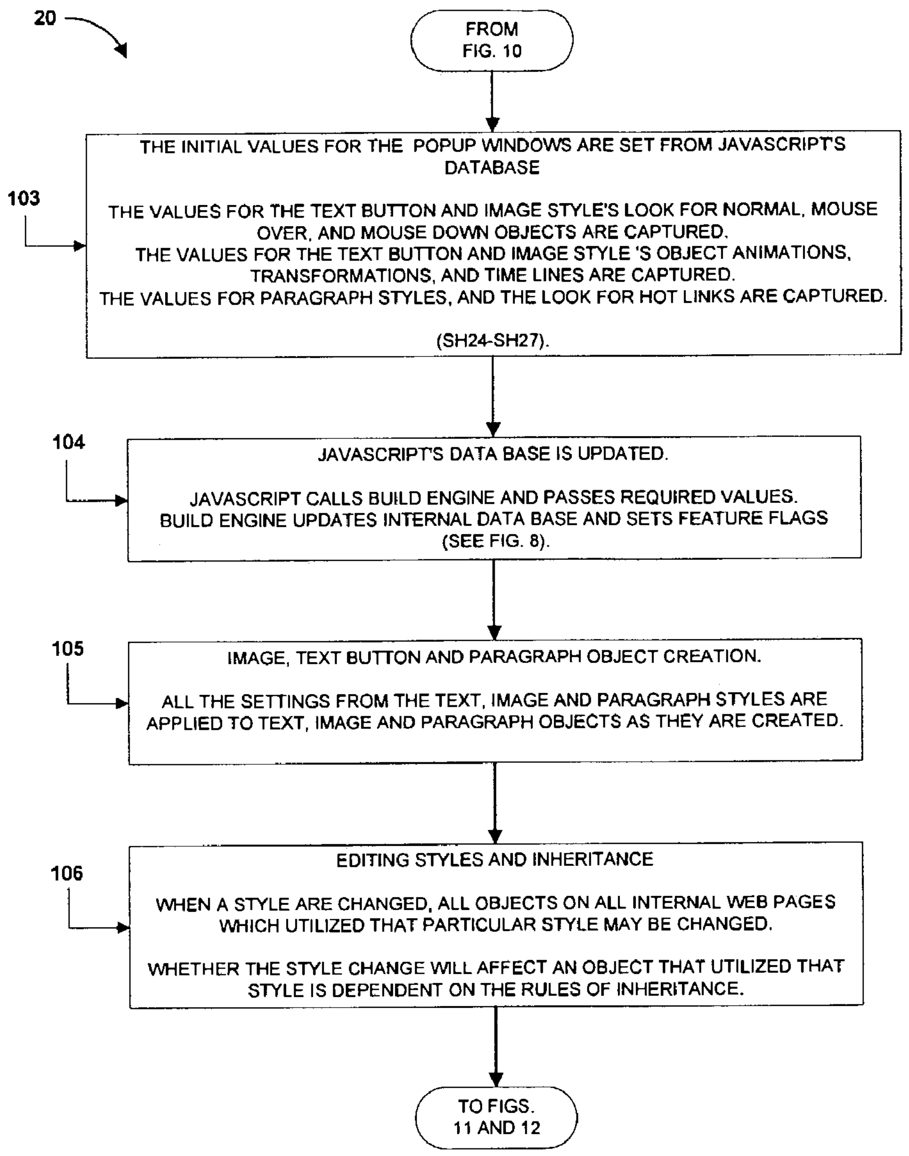

20
FROM FIG. 10
103
THE INITIAL VALUES FOR THE POPUP WINDOWS ARE SET FROM JAVASCRIPT'S DATABASE
THE VALUES FOR THE TEXT BUTTON AND IMAGE STYLE'S LOOK FOR NORMAL, MOUSE OVER, AND MOUSE DOWN OBJECTS ARE CAPTURED.
THE VALUES FOR THE TEXT BUTTON AND IMAGE STYLE 'S OBJECT ANIMATIONS, TRANSFORMATIONS, AND TIME LINES ARE CAPTURED.
THE VALUES FOR PARAGRAPH STYLES, AND THE LOOK FOR HOT LINKS ARE CAPTURED.
(SH24-SH27).
104
JAVASCRIPT'S DATA BASE IS UPDATED.
JAVASCRIPT CALLS BUILD ENGINE AND PASSES REQUIRED VALUES.
BUILD ENGINE UPDATES INTERNAL DATA BASE AND SETS FEATURE FLAGS (SEE FIG. 8).
105
IMAGE, TEXT BUTTON AND PARAGRAPH OBJECT CREATION.
ALL THE SETTINGS FROM THE TEXT, IMAGE AND PARAGRAPH STYLES ARE APPLIED TO TEXT, IMAGE AND PARAGRAPH OBJECTS AS THEY ARE CREATED.
106
EDITING STYLES AND INHERITANCE
WHEN A STYLE ARE CHANGED, ALL OBJECTS ON ALL INTERNAL WEB PAGES WHICH UTILIZED THAT PARTICULAR STYLE MAY BE CHANGED.
WHETHER THE STYLE CHANGE WILL AFFECT AN OBJECT THAT UTILIZED THAT STYLE IS DEPENDENT ON THE RULES OF INHERITANCE.
TO FIGS. 11 AND 12

Fig. 13

BUTTON, IMAGE AND PARAGRAPH STYLE SETTINGS AND TECHNOLOGY

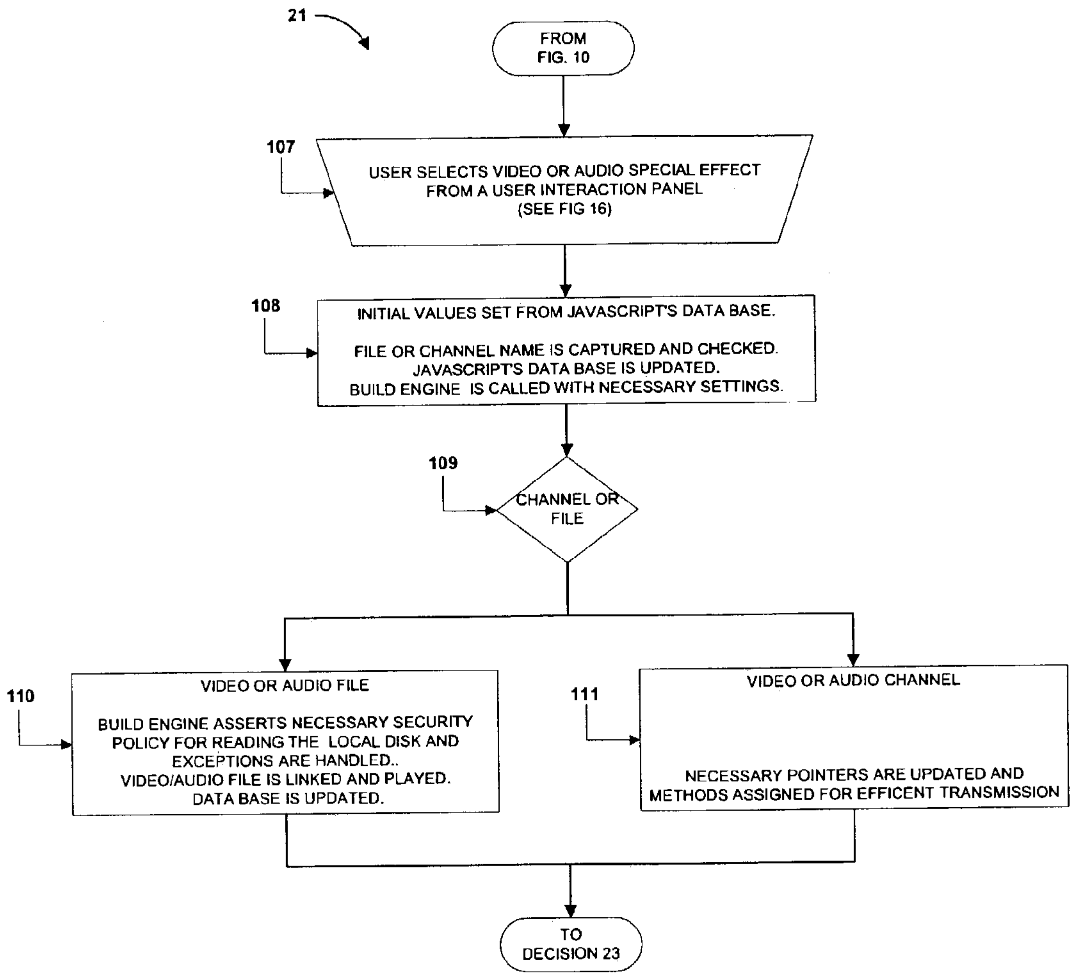

21
FROM FIG. 10
107
USER SELECTS VIDEO OR AUDIO SPECIAL EFFECT FROM A USER INTERACTION PANEL (SEE FIG 16)
108
INITIAL VALUES SET FROM JAVASCRIPT'S DATA BASE.
FILE OR CHANNEL NAME IS CAPTURED AND CHECKED.
JAVASCRIPT'S DATA BASE IS UPDATED.
BUILD ENGINE IS CALLED WITH NECESSARY SETTINGS.
109
CHANNEL OR FILE
110
VIDEO OR AUDIO FILE
BUILD ENGINE ASSERTS NECESSARY SECURITY POLICY FOR READING THE LOCAL DISK AND EXCEPTIONS ARE HANDLED..
VIDEO/AUDIO FILE IS LINKED AND PLAYED.
DATA BASE IS UPDATED.
111
VIDEO OR AUDIO CHANNEL
NECESSARY POINTERS ARE UPDATED AND METHODS ASSIGNED FOR EFFICENT TRANSMISSION
TO DECISION 23

Fig. 14

VIDEO AND AUDIO FILE/CHANNEL PROCESSING

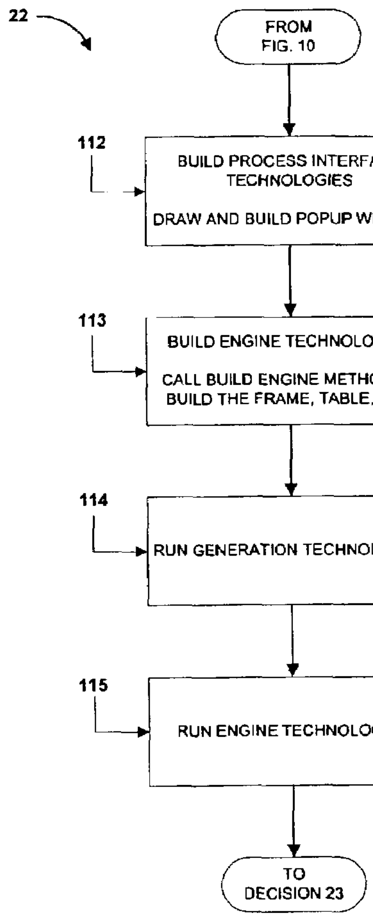
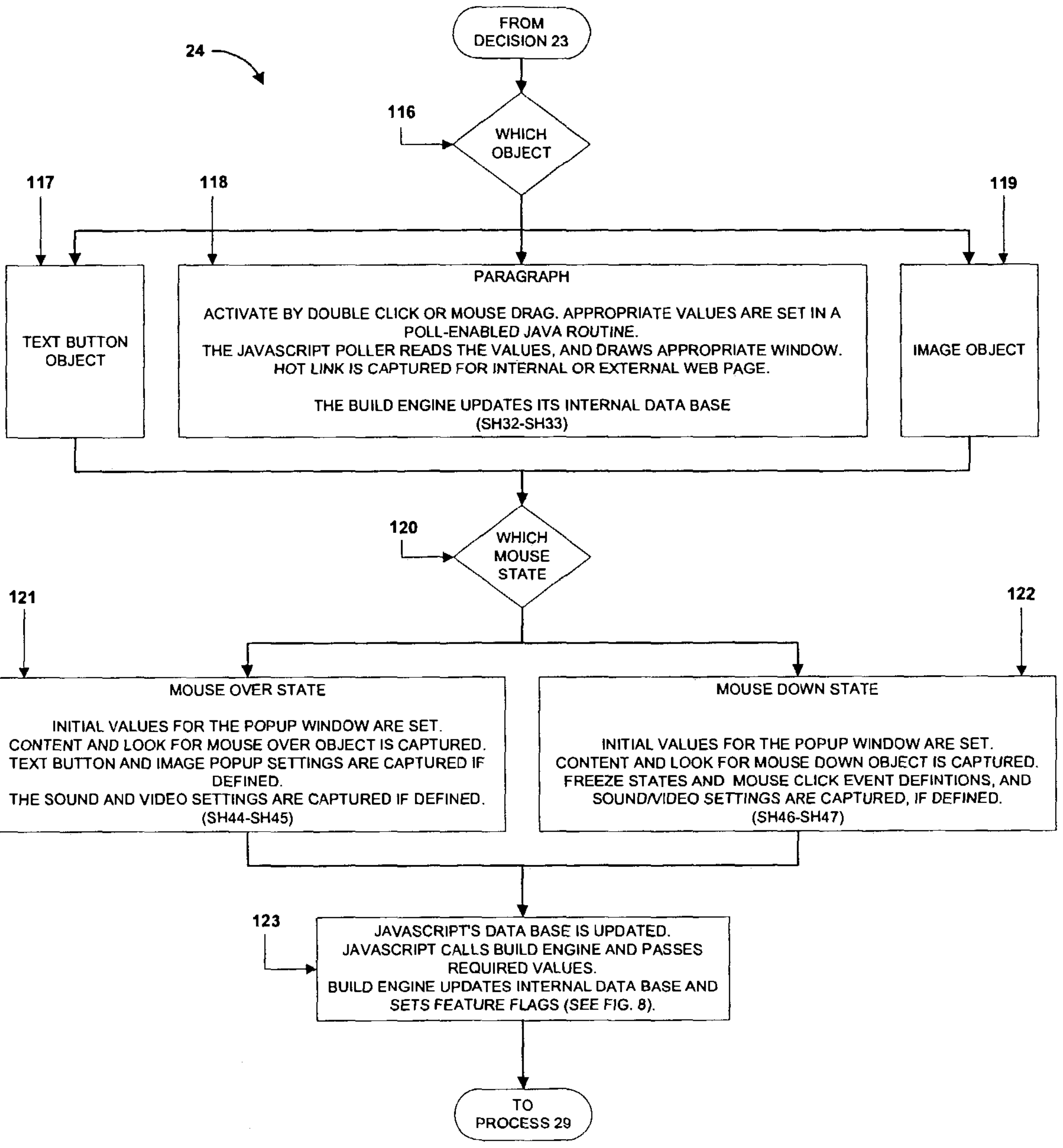

22
FROM FIG. 10
112
BUILD PROCESS INTERFACE TECHNOLOGIES
DRAW AND BUILD POPUP WINDOW
113
BUILD ENGINE TECHNOLOGIES
CALL BUILD ENGINE METHOD TO BUILD THE FRAME, TABLE, ETC.
114
RUN GENERATION TECHNOLOGIES
115
RUN ENGINE TECHNOLOGIES
TO DECISION 23

Fig. 15

FRAMES, TABLES, FORMS AND DRAW OBJECTS



24
FROM DECISION 23
116
WHICH OBJECT
117
118
119
TEXT BUTTON OBJECT
PARAGRAPH
ACTIVATE BY DOUBLE CLICK OR MOUSE DRAG. APPROPRIATE VALUES ARE SET IN A POLL-ENABLED JAVA ROUTINE.
THE JAVASCRIPT POLLER READS THE VALUES, AND DRAWS APPROPRIATE WINDOW.
HOT LINK IS CAPTURED FOR INTERNAL OR EXTERNAL WEB PAGE.
THE BUILD ENGINE UPDATES ITS INTERNAL DATA BASE (SH32-SH33)
IMAGE OBJECT
120
WHICH MOUSE STATE
121
122
MOUSE OVER STATE
INITIAL VALUES FOR THE POPUP WINDOW ARE SET.
CONTENT AND LOOK FOR MOUSE OVER OBJECT IS CAPTURED.
TEXT BUTTON AND IMAGE POPUP SETTINGS ARE CAPTURED IF DEFINED.
THE SOUND AND VIDEO SETTINGS ARE CAPTURED IF DEFINED.
(SH44-SH45)
MOUSE DOWN STATE
INITIAL VALUES FOR THE POPUP WINDOW ARE SET.
CONTENT AND LOOK FOR MOUSE DOWN OBJECT IS CAPTURED.
FREEZE STATES AND MOUSE CLICK EVENT DEFINTIONS, AND SOUND/VIDEO SETTINGS ARE CAPTURED, IF DEFINED.
(SH46-SH47)
123
JAVASCRIPT'S DATA BASE IS UPDATED.
JAVASCRIPT CALLS BUILD ENGINE AND PASSES REQUIRED VALUES.
BUILD ENGINE UPDATES INTERNAL DATA BASE AND SETS FEATURE FLAGS (SEE FIG. 8).
TO PROCESS 29

Fig. 16

USER INTERACTION SETTINGS AND TECHNOLOGY

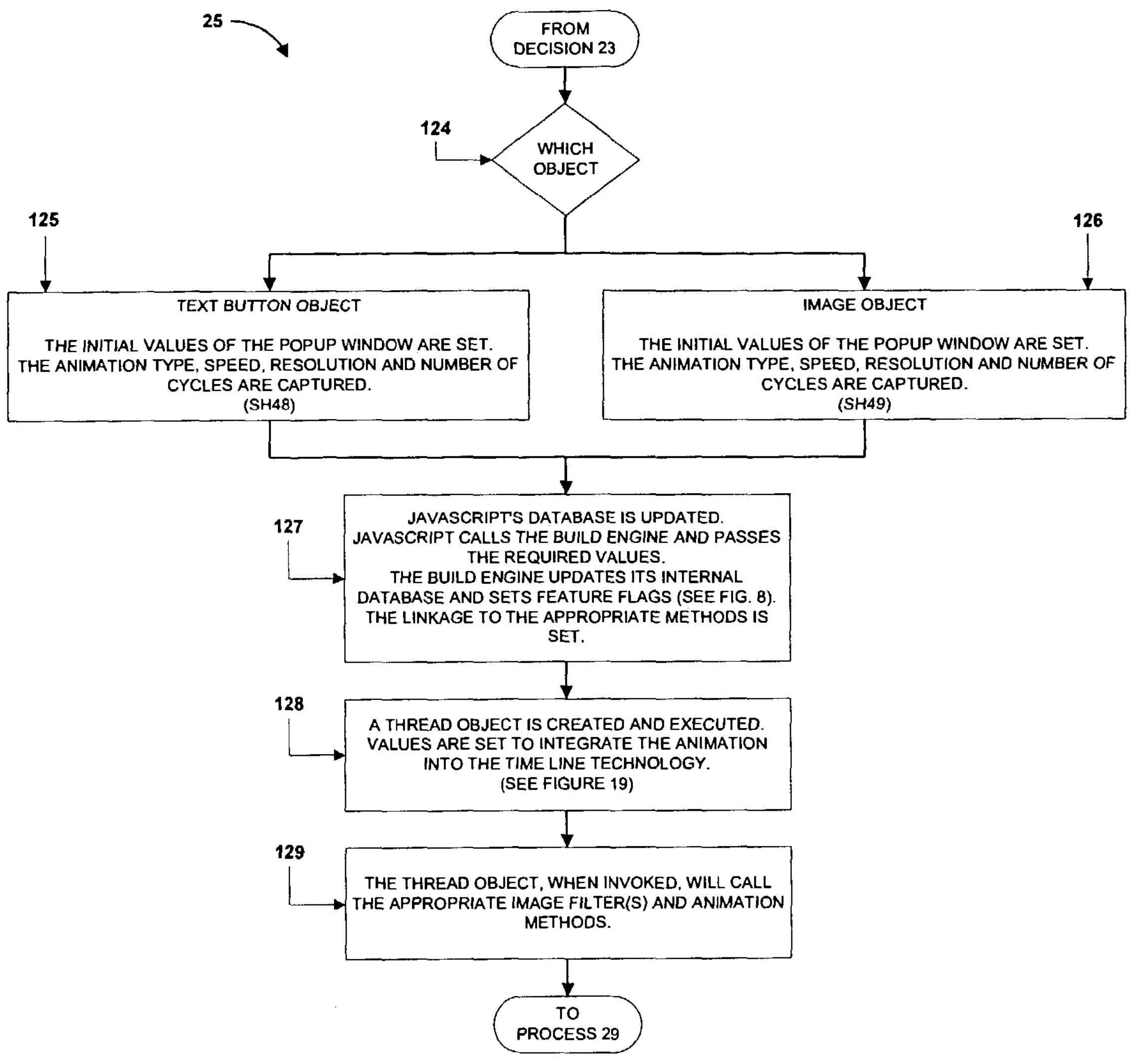

25
FROM DECISION 23
124
WHICH OBJECT
125
126
TEXT BUTTON OBJECT
THE INITIAL VALUES OF THE POPUP WINDOW ARE SET. THE ANIMATION TYPE, SPEED, RESOLUTION AND NUMBER OF CYCLES ARE CAPTURED. (SH48)
IMAGE OBJECT
THE INITIAL VALUES OF THE POPUP WINDOW ARE SET. THE ANIMATION TYPE, SPEED, RESOLUTION AND NUMBER OF CYCLES ARE CAPTURED. (SH49)
127
JAVASCRIPT'S DATABASE IS UPDATED. JAVASCRIPT CALLS THE BUILD ENGINE AND PASSES THE REQUIRED VALUES. THE BUILD ENGINE UPDATES ITS INTERNAL DATABASE AND SETS FEATURE FLAGS (SEE FIG. 8). THE LINKAGE TO THE APPROPRIATE METHODS IS SET.
128
A THREAD OBJECT IS CREATED AND EXECUTED. VALUES ARE SET TO INTEGRATE THE ANIMATION INTO THE TIME LINE TECHNOLOGY. (SEE FIGURE 19)
129
THE THREAD OBJECT, WHEN INVOKED, WILL CALL THE APPROPRIATE IMAGE FILTER(S) AND ANIMATION METHODS.
TO PROCESS 29

Fig. 17

ANIMATION SETTINGS AND TECHNOLOGY

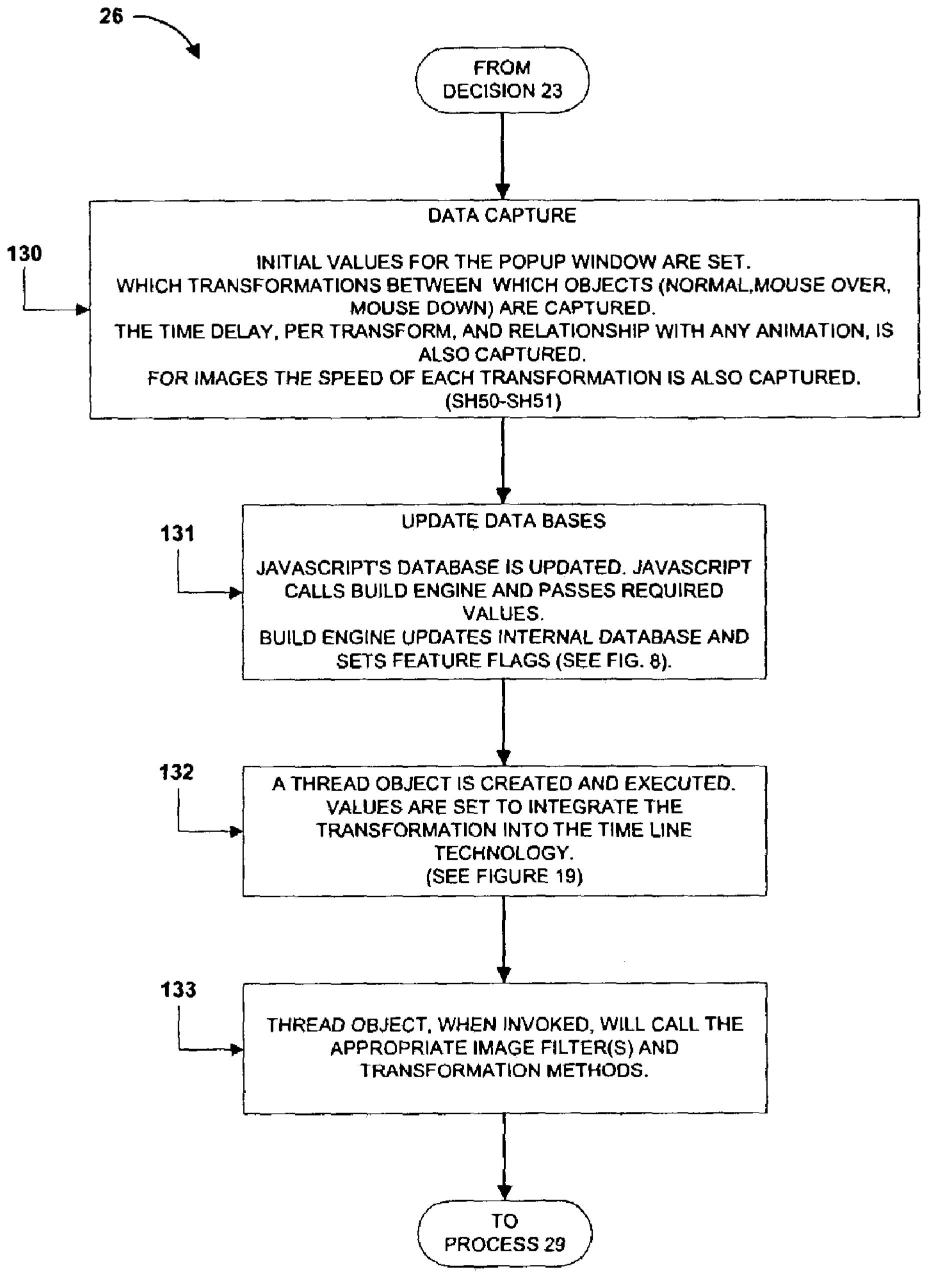

26
FROM DECISION 23
130
DATA CAPTURE
INITIAL VALUES FOR THE POPUP WINDOW ARE SET.
WHICH TRANSFORMATIONS BETWEEN WHICH OBJECTS (NORMAL,MOUSE OVER, MOUSE DOWN) ARE CAPTURED.
THE TIME DELAY, PER TRANSFORM, AND RELATIONSHIP WITH ANY ANIMATION, IS ALSO CAPTURED.
FOR IMAGES THE SPEED OF EACH TRANSFORMATION IS ALSO CAPTURED.
(SH50-SH51)
131
UPDATE DATA BASES
JAVASCRIPT'S DATABASE IS UPDATED. JAVASCRIPT CALLS BUILD ENGINE AND PASSES REQUIRED VALUES.
BUILD ENGINE UPDATES INTERNAL DATABASE AND SETS FEATURE FLAGS (SEE FIG. 8).
132
A THREAD OBJECT IS CREATED AND EXECUTED. VALUES ARE SET TO INTEGRATE THE TRANSFORMATION INTO THE TIME LINE TECHNOLOGY.
(SEE FIGURE 19)
133
THREAD OBJECT, WHEN INVOKED, WILL CALL THE APPROPRIATE IMAGE FILTER(S) AND TRANSFORMATION METHODS.
TO PROCESS 29

Fig. 18

TRANSFORMATION SETTINGS AND TECHNOLOGY

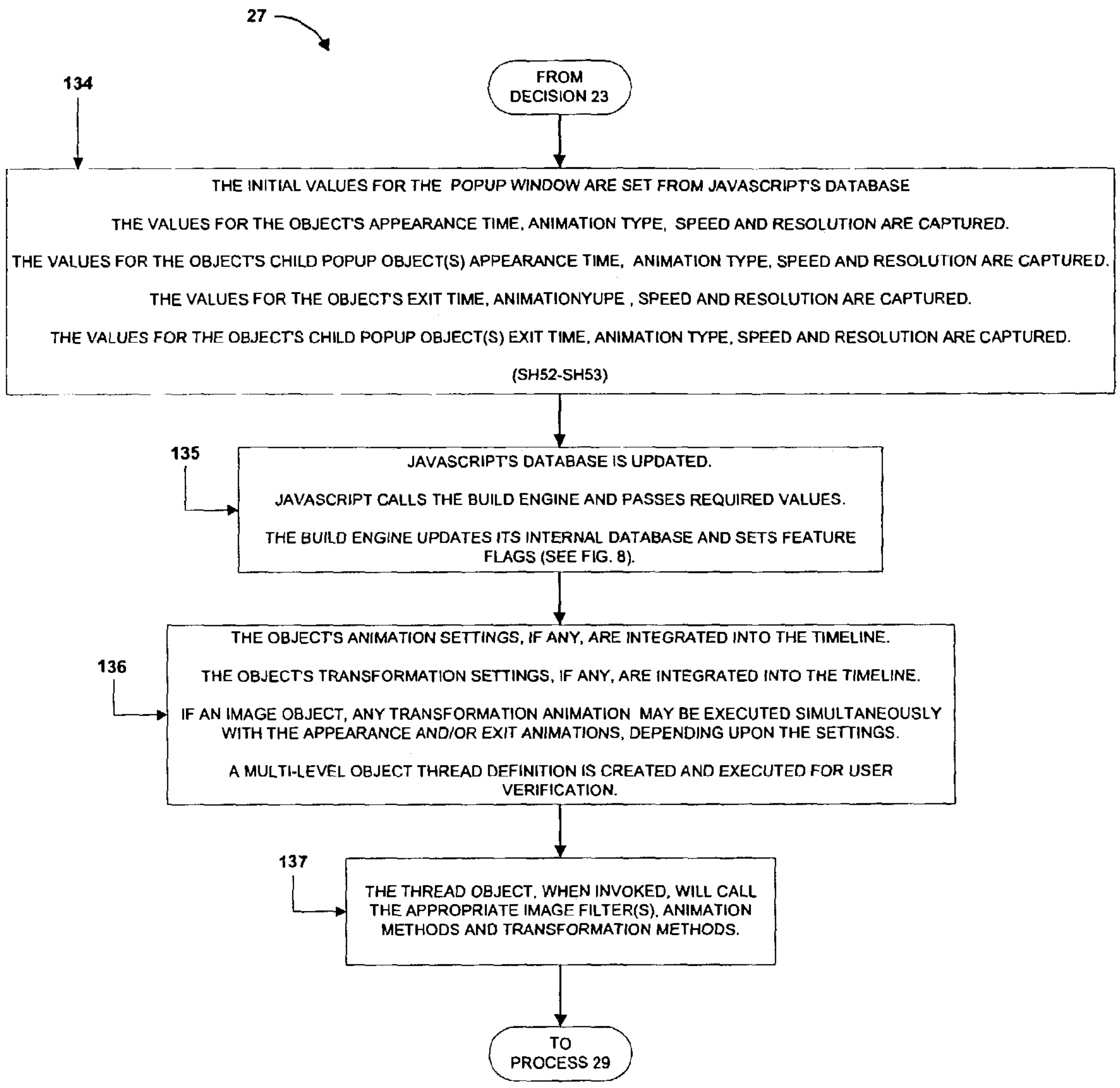

27
134
FROM DECISION 23
THE INITIAL VALUES FOR THE POPUP WINDOW ARE SET FROM JAVASCRIPT'S DATABASE
THE VALUES FOR THE OBJECT'S APPEARANCE TIME, ANIMATION TYPE, SPEED AND RESOLUTION ARE CAPTURED.
THE VALUES FOR THE OBJECT'S CHILD POPUP OBJECT(S) APPEARANCE TIME, ANIMATION TYPE, SPEED AND RESOLUTION ARE CAPTURED.
THE VALUES FOR THE OBJECT'S EXIT TIME, ANIMATIONYUPE , SPEED AND RESOLUTION ARE CAPTURED.
THE VALUES FOR THE OBJECT'S CHILD POPUP OBJECT(S) EXIT TIME, ANIMATION TYPE, SPEED AND RESOLUTION ARE CAPTURED.
(SH52-SH53)
135
JAVASCRIPT'S DATABASE IS UPDATED.
JAVASCRIPT CALLS THE BUILD ENGINE AND PASSES REQUIRED VALUES.
THE BUILD ENGINE UPDATES ITS INTERNAL DATABASE AND SETS FEATURE FLAGS (SEE FIG. 8).
136
THE OBJECT'S ANIMATION SETTINGS, IF ANY, ARE INTEGRATED INTO THE TIMELINE.
THE OBJECT'S TRANSFORMATION SETTINGS, IF ANY, ARE INTEGRATED INTO THE TIMELINE.
IF AN IMAGE OBJECT, ANY TRANSFORMATION ANIMATION MAY BE EXECUTED SIMULTANEOUSLY WITH THE APPEARANCE AND/OR EXIT ANIMATIONS, DEPENDING UPON THE SETTINGS.
A MULTI-LEVEL OBJECT THREAD DEFINITION IS CREATED AND EXECUTED FOR USER VERIFICATION.
137
THE THREAD OBJECT, WHEN INVOKED, WILL CALL THE APPROPRIATE IMAGE FILTER(S), ANIMATION METHODS AND TRANSFORMATION METHODS.
TO PROCESS 29

Fig. 19

OBJECT TIME LINES AND TECHNOLOGY

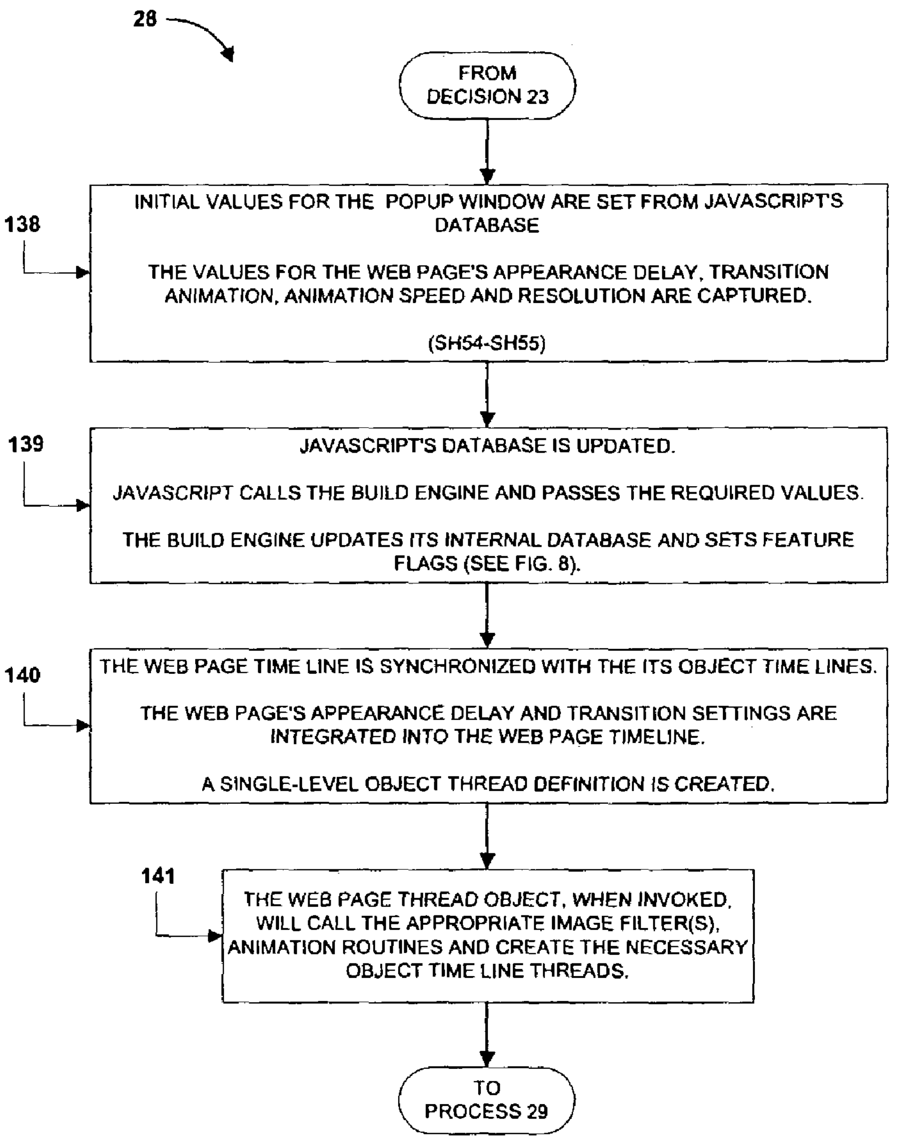

28
FROM DECISION 23
138
INITIAL VALUES FOR THE POPUP WINDOW ARE SET FROM JAVASCRIPT'S DATABASE
THE VALUES FOR THE WEB PAGE'S APPEARANCE DELAY, TRANSITION ANIMATION, ANIMATION SPEED AND RESOLUTION ARE CAPTURED.
(SH54-SH55)
139
JAVASCRIPT'S DATABASE IS UPDATED.
JAVASCRIPT CALLS THE BUILD ENGINE AND PASSES THE REQUIRED VALUES.
THE BUILD ENGINE UPDATES ITS INTERNAL DATABASE AND SETS FEATURE FLAGS (SEE FIG. 8).
140
THE WEB PAGE TIME LINE IS SYNCHRONIZED WITH THE ITS OBJECT TIME LINES.
THE WEB PAGE'S APPEARANCE DELAY AND TRANSITION SETTINGS ARE INTEGRATED INTO THE WEB PAGE TIMELINE.
A SINGLE-LEVEL OBJECT THREAD DEFINITION IS CREATED.
141
THE WEB PAGE THREAD OBJECT, WHEN INVOKED, WILL CALL THE APPROPRIATE IMAGE FILTER(S), ANIMATION ROUTINES AND CREATE THE NECESSARY OBJECT TIME LINE THREADS.
TO PROCESS 29

Fig. 20

WEB PAGE TRANSITION ANIMATIONS, TIME LINE SETTINGS AND TECHNOLOGY

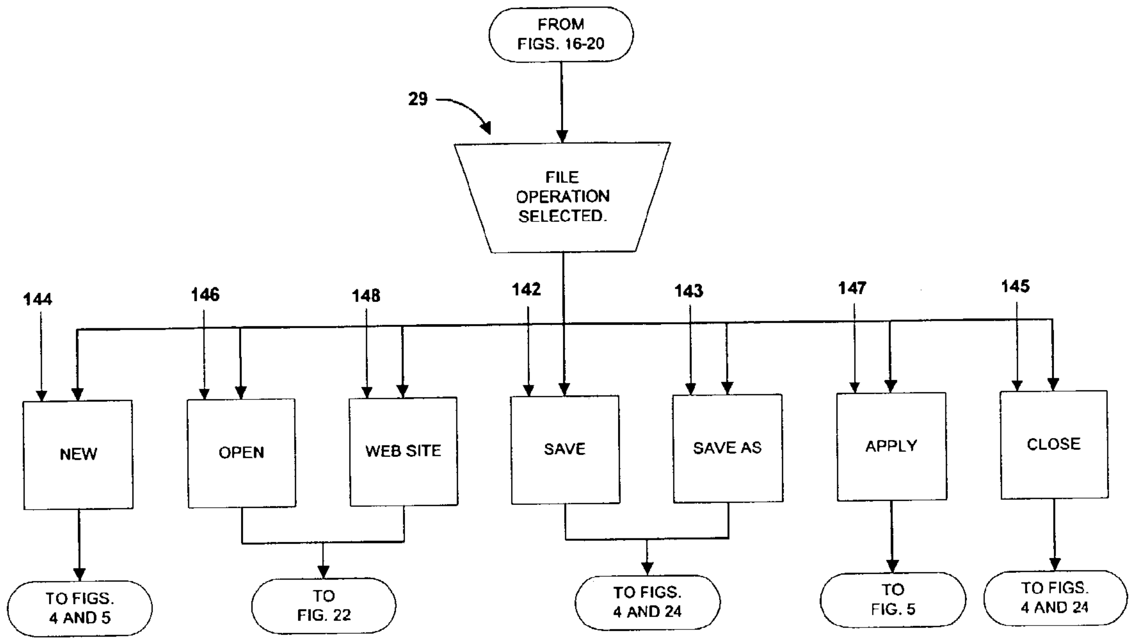

FROM FIGS. 16-20
29
FILE OPERATION SELECTED.
144
146
148
142
143
147
145
NEW
OPEN
WEB SITE
SAVE
SAVE AS
APPLY
CLOSE
TO FIGS. 4 AND 5
TO FIG. 22
TO FIGS. 4 AND 24
TO FIG. 5
TO FIGS. 4 AND 24

fig. 21a

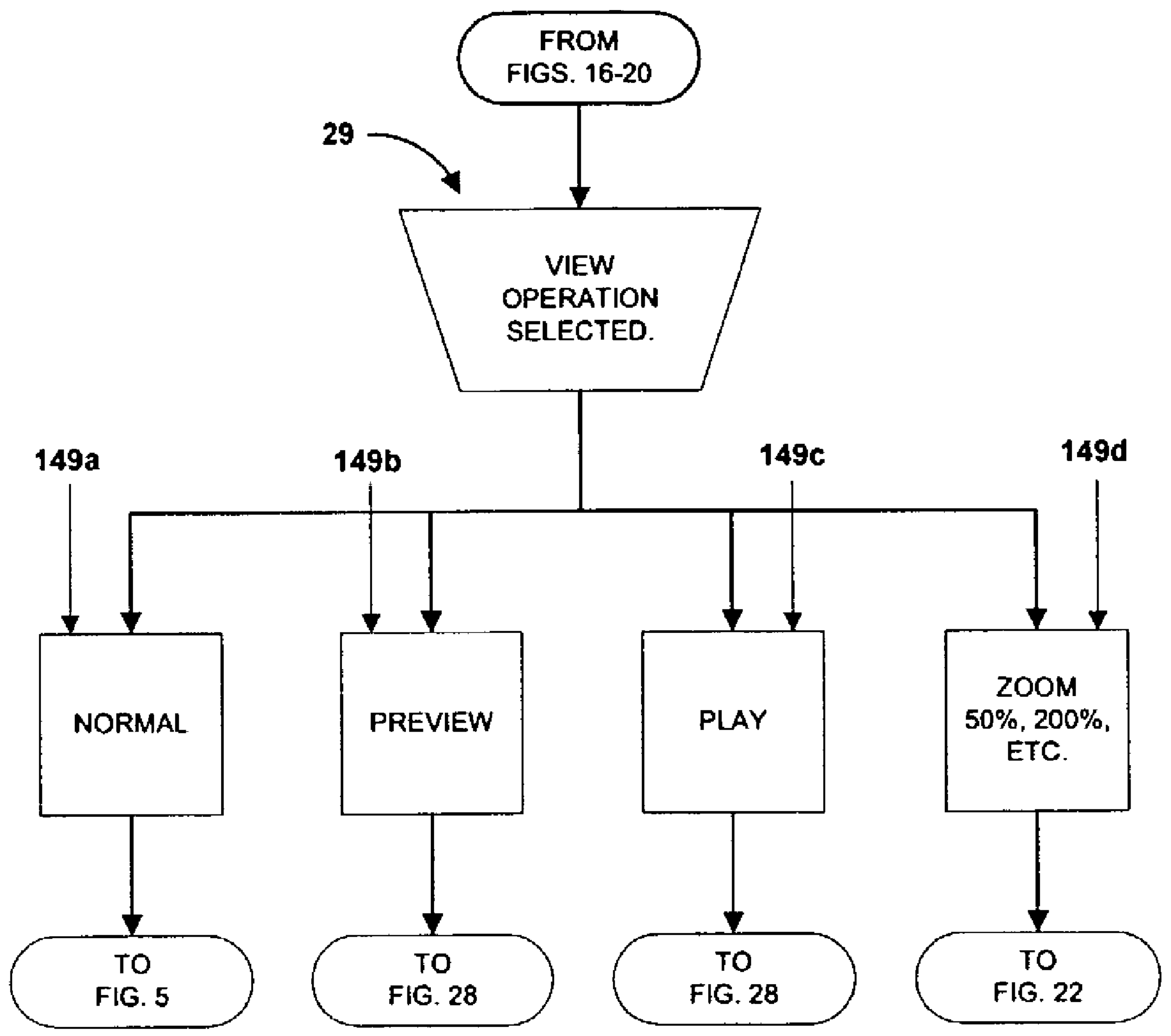

FROM FIGS. 16-20
29
VIEW OPERATION SELECTED.
149a
149b
149c
149d
NORMAL
PREVIEW
PLAY
ZOOM 50%, 200%, ETC.
TO FIG. 5
TO FIG. 28
TO FIG. 28
TO FIG. 22

fig. 21b

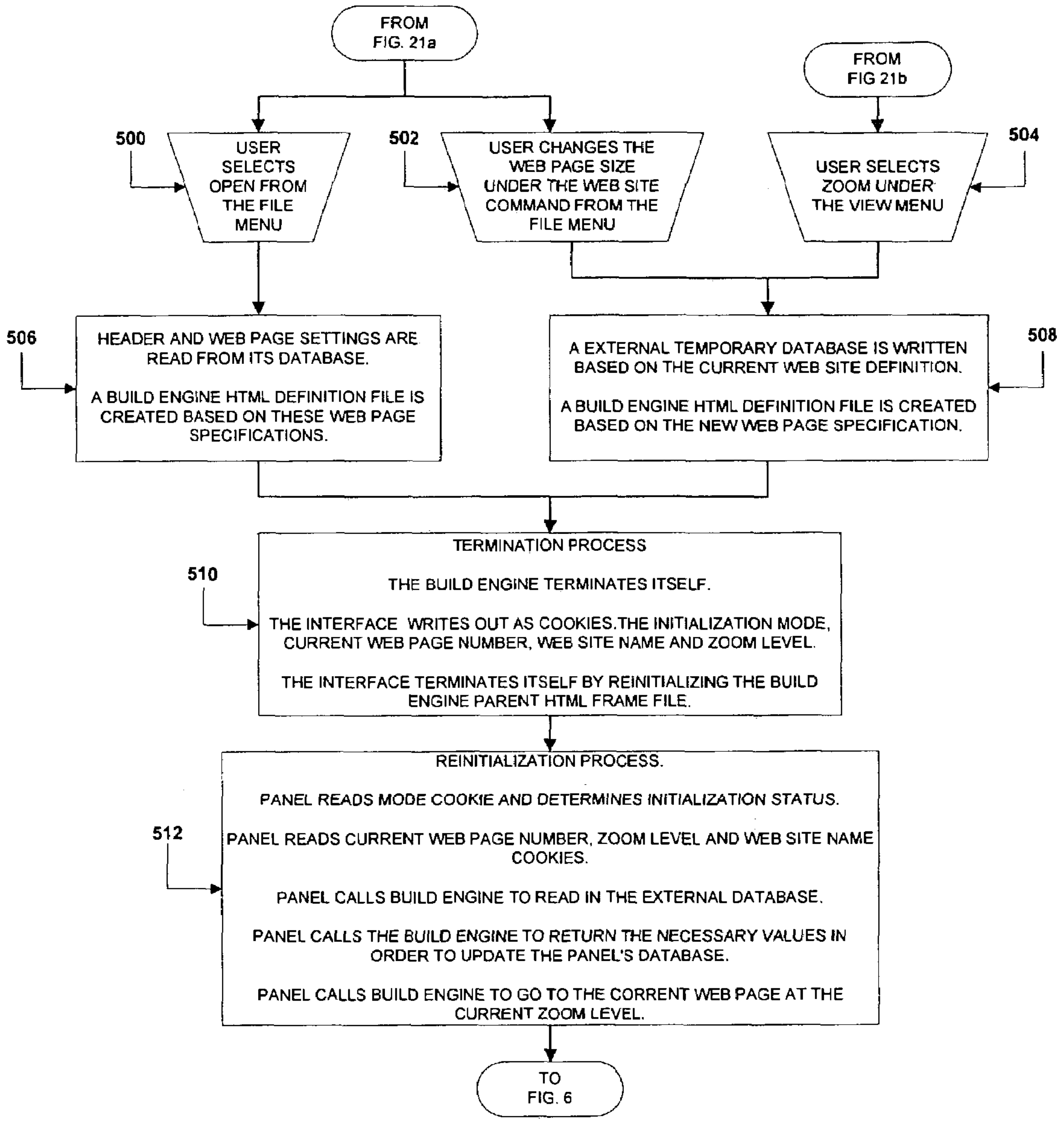
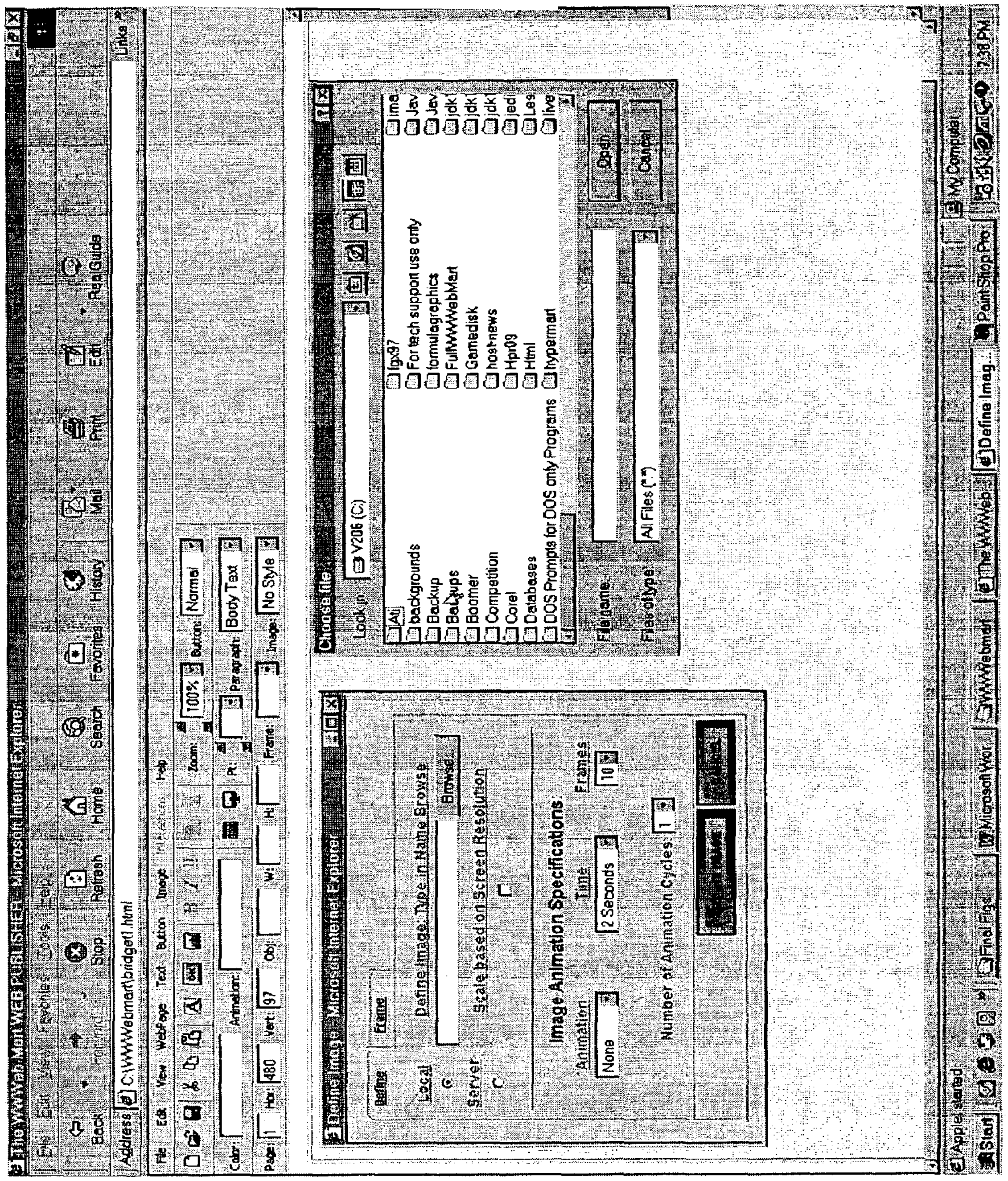

FROM FIG. 21a
FROM FIG 21b
500
USER SELECTS OPEN FROM THE FILE MENU
502
USER CHANGES THE WEB PAGE SIZE UNDER THE WEB SITE COMMAND FROM THE FILE MENU
504
USER SELECTS ZOOM UNDER THE VIEW MENU
506
HEADER AND WEB PAGE SETTINGS ARE READ FROM ITS DATABASE.
A BUILD ENGINE HTML DEFINITION FILE IS CREATED BASED ON THESE WEB PAGE SPECIFICATIONS.
508
A EXTERNAL TEMPORARY DATABASE IS WRITTEN BASED ON THE CURRENT WEB SITE DEFINITION.
A BUILD ENGINE HTML DEFINITION FILE IS CREATED BASED ON THE NEW WEB PAGE SPECIFICATION.
510
TERMINATION PROCESS
THE BUILD ENGINE TERMINATES ITSELF.
THE INTERFACE WRITES OUT AS COOKIES.THE INITIALIZATION MODE, CURRENT WEB PAGE NUMBER, WEB SITE NAME AND ZOOM LEVEL.
THE INTERFACE TERMINATES ITSELF BY REINITIALIZING THE BUILD ENGINE PARENT HTML FRAME FILE.
512
REINITIALIZATION PROCESS.
PANEL READS MODE COOKIE AND DETERMINES INITIALIZATION STATUS.
PANEL READS CURRENT WEB PAGE NUMBER, ZOOM LEVEL AND WEB SITE NAME COOKIES.
PANEL CALLS BUILD ENGINE TO READ IN THE EXTERNAL DATABASE.
PANEL CALLS THE BUILD ENGINE TO RETURN THE NECESSARY VALUES IN ORDER TO UPDATE THE PANEL'S DATABASE.
PANEL CALLS BUILD ENGINE TO GO TO THE CORRENT WEB PAGE AT THE CURRENT ZOOM LEVEL.
TO FIG. 6

Fig. 22

DYNAMIC WEB PAGE RESIZING PROCESS


The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Links
File Edit View WebPage Text Button Image Help
Zoom: 100% Button: Normal
Paragraph: Body Text
Image: No Style
Frame:
Define Image - Microsoft Internet Explorer
Define Frame
Define Image, Type in Name Browse
Browse
Local
Server
Scale based on Screen Resolution
Image Animation Specifications
Animation Time Frames
Number of Animation Cycles
Choose file
Look in:
File name:
Files of type:
Open
Cancel
Start
My Computer
PaintShop Pro
Define Imag...
The WWWeb...
Microsoft Wor...
Find Files
7:38 PM

Address C:\WWWebmart\bridget1.html

Color: Animation:

Page: 1 Hor: 480 Vert: 97 Obj: W: H: Frame:

Animation: None Time: 2 Seconds Frames: 10

Number of Animation Cycles: 1

Look in: Y206 (C:)

Ati
backgrounds
Backup
Backups
Boomer
Competition
Corel
Databases
DOS Prompts for DOS only Programs
fgx97
For tech support use only
formulagraphics
FullWWWebMart
Gamedisk
host-news
Hpr09
Html
typemart
ime
Jav
Jav
jdk
jdk
jdk
jed
Les
live

Files of type: All Files (*.*)

Apple started

Fig. 23

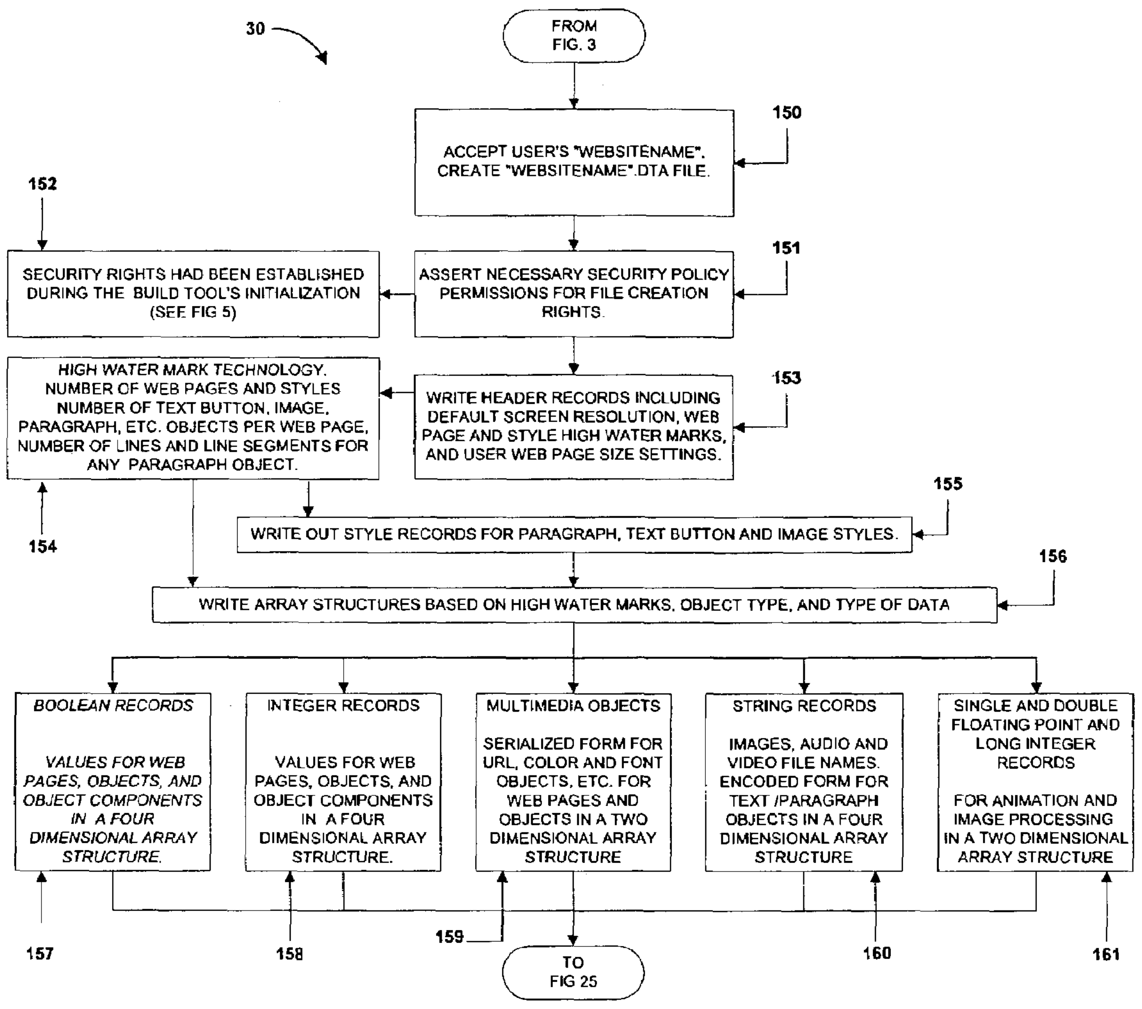

30
FROM FIG. 3
150
ACCEPT USER'S "WEBSITENAME". CREATE "WEBSITENAME".DTA FILE.
151
ASSERT NECESSARY SECURITY POLICY PERMISSIONS FOR FILE CREATION RIGHTS.
152
SECURITY RIGHTS HAD BEEN ESTABLISHED DURING THE BUILD TOOL'S INITIALIZATION (SEE FIG 5)
153
WRITE HEADER RECORDS INCLUDING DEFAULT SCREEN RESOLUTION, WEB PAGE AND STYLE HIGH WATER MARKS, AND USER WEB PAGE SIZE SETTINGS.
HIGH WATER MARK TECHNOLOGY. NUMBER OF WEB PAGES AND STYLES NUMBER OF TEXT BUTTON, IMAGE, PARAGRAPH, ETC. OBJECTS PER WEB PAGE, NUMBER OF LINES AND LINE SEGMENTS FOR ANY PARAGRAPH OBJECT.
154
155
WRITE OUT STYLE RECORDS FOR PARAGRAPH, TEXT BUTTON AND IMAGE STYLES.
156
WRITE ARRAY STRUCTURES BASED ON HIGH WATER MARKS, OBJECT TYPE, AND TYPE OF DATA
BOOLEAN RECORDS VALUES FOR WEB PAGES, OBJECTS, AND OBJECT COMPONENTS IN A FOUR DIMENSIONAL ARRAY STRUCTURE.
INTEGER RECORDS VALUES FOR WEB PAGES, OBJECTS, AND OBJECT COMPONENTS IN A FOUR DIMENSIONAL ARRAY STRUCTURE.
MULTIMEDIA OBJECTS SERIALIZED FORM FOR URL, COLOR AND FONT OBJECTS, ETC. FOR WEB PAGES AND OBJECTS IN A TWO DIMENSIONAL ARRAY STRUCTURE
STRING RECORDS IMAGES, AUDIO AND VIDEO FILE NAMES. ENCODED FORM FOR TEXT /PARAGRAPH OBJECTS IN A FOUR DIMENSIONAL ARRAY STRUCTURE
SINGLE AND DOUBLE FLOATING POINT AND LONG INTEGER RECORDS FOR ANIMATION AND IMAGE PROCESSING IN A TWO DIMENSIONAL ARRAY STRUCTURE
157
158
159
160
161
TO FIG 25

Fig. 24

EXTERNAL DATA BASE CREATION: SECURITY AND OPTIMIZATION TECHNIQUES

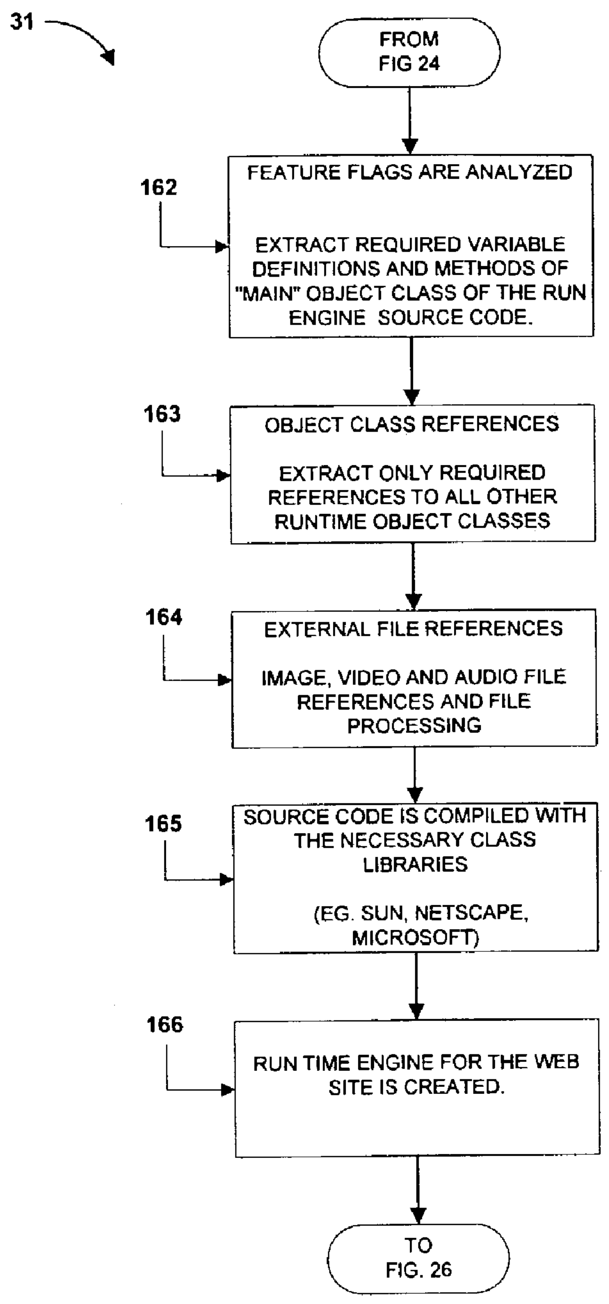

31
FROM FIG 24
162
FEATURE FLAGS ARE ANALYZED
EXTRACT REQUIRED VARIABLE DEFINITIONS AND METHODS OF "MAIN" OBJECT CLASS OF THE RUN ENGINE SOURCE CODE.
163
OBJECT CLASS REFERENCES
EXTRACT ONLY REQUIRED REFERENCES TO ALL OTHER RUNTIME OBJECT CLASSES
164
EXTERNAL FILE REFERENCES
IMAGE, VIDEO AND AUDIO FILE REFERENCES AND FILE PROCESSING
165
SOURCE CODE IS COMPILED WITH THE NECESSARY CLASS LIBRARIES
(EG. SUN, NETSCAPE, MICROSOFT)
166
RUN TIME ENGINE FOR THE WEB SITE IS CREATED.
TO FIG. 26

Fig. 25

CREATE CUSTOMIZED AND OPTIMIZED RUNTIME ENGINE

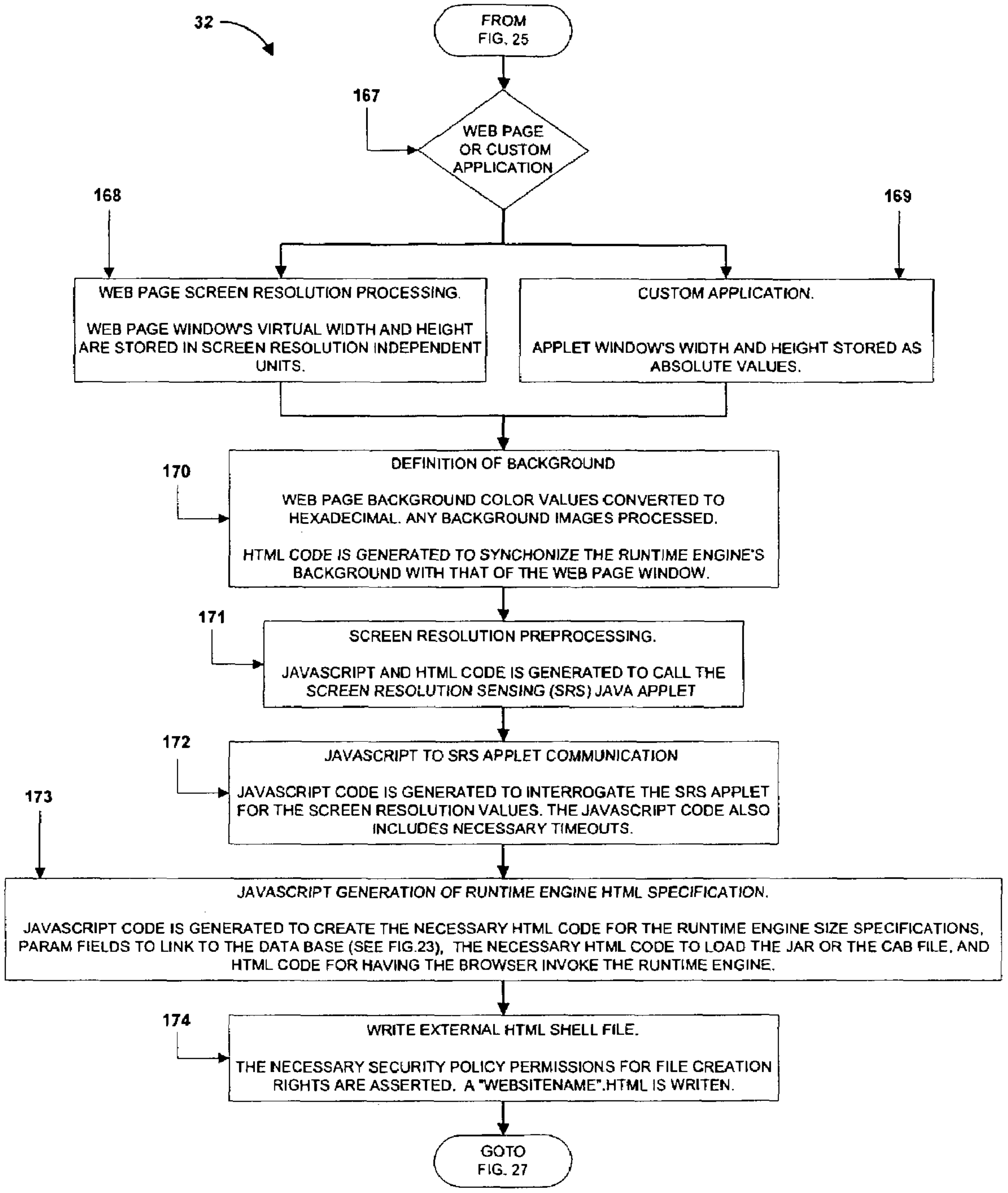

32
FROM FIG. 25
167
WEB PAGE OR CUSTOM APPLICATION
168
169
WEB PAGE SCREEN RESOLUTION PROCESSING.
WEB PAGE WINDOW'S VIRTUAL WIDTH AND HEIGHT ARE STORED IN SCREEN RESOLUTION INDEPENDENT UNITS.
CUSTOM APPLICATION.
APPLET WINDOW'S WIDTH AND HEIGHT STORED AS ABSOLUTE VALUES.
170
DEFINITION OF BACKGROUND
WEB PAGE BACKGROUND COLOR VALUES CONVERTED TO HEXADECIMAL. ANY BACKGROUND IMAGES PROCESSED.
HTML CODE IS GENERATED TO SYNCHONIZE THE RUNTIME ENGINE'S BACKGROUND WITH THAT OF THE WEB PAGE WINDOW.
171
SCREEN RESOLUTION PREPROCESSING.
JAVASCRIPT AND HTML CODE IS GENERATED TO CALL THE SCREEN RESOLUTION SENSING (SRS) JAVA APPLET
172
JAVASCRIPT TO SRS APPLET COMMUNICATION
JAVASCRIPT CODE IS GENERATED TO INTERROGATE THE SRS APPLET FOR THE SCREEN RESOLUTION VALUES. THE JAVASCRIPT CODE ALSO INCLUDES NECESSARY TIMEOUTS.
173
JAVASCRIPT GENERATION OF RUNTIME ENGINE HTML SPECIFICATION.
JAVASCRIPT CODE IS GENERATED TO CREATE THE NECESSARY HTML CODE FOR THE RUNTIME ENGINE SIZE SPECIFICATIONS, PARAM FIELDS TO LINK TO THE DATA BASE (SEE FIG.23), THE NECESSARY HTML CODE TO LOAD THE JAR OR THE CAB FILE, AND HTML CODE FOR HAVING THE BROWSER INVOKE THE RUNTIME ENGINE.
174
WRITE EXTERNAL HTML SHELL FILE.
THE NECESSARY SECURITY POLICY PERMISSIONS FOR FILE CREATION RIGHTS ARE ASSERTED. A "WEBSITENAME".HTML IS WRITEN.
GOTO FIG. 27

Fig. 26

CREATE THE HTML SHELL FILE

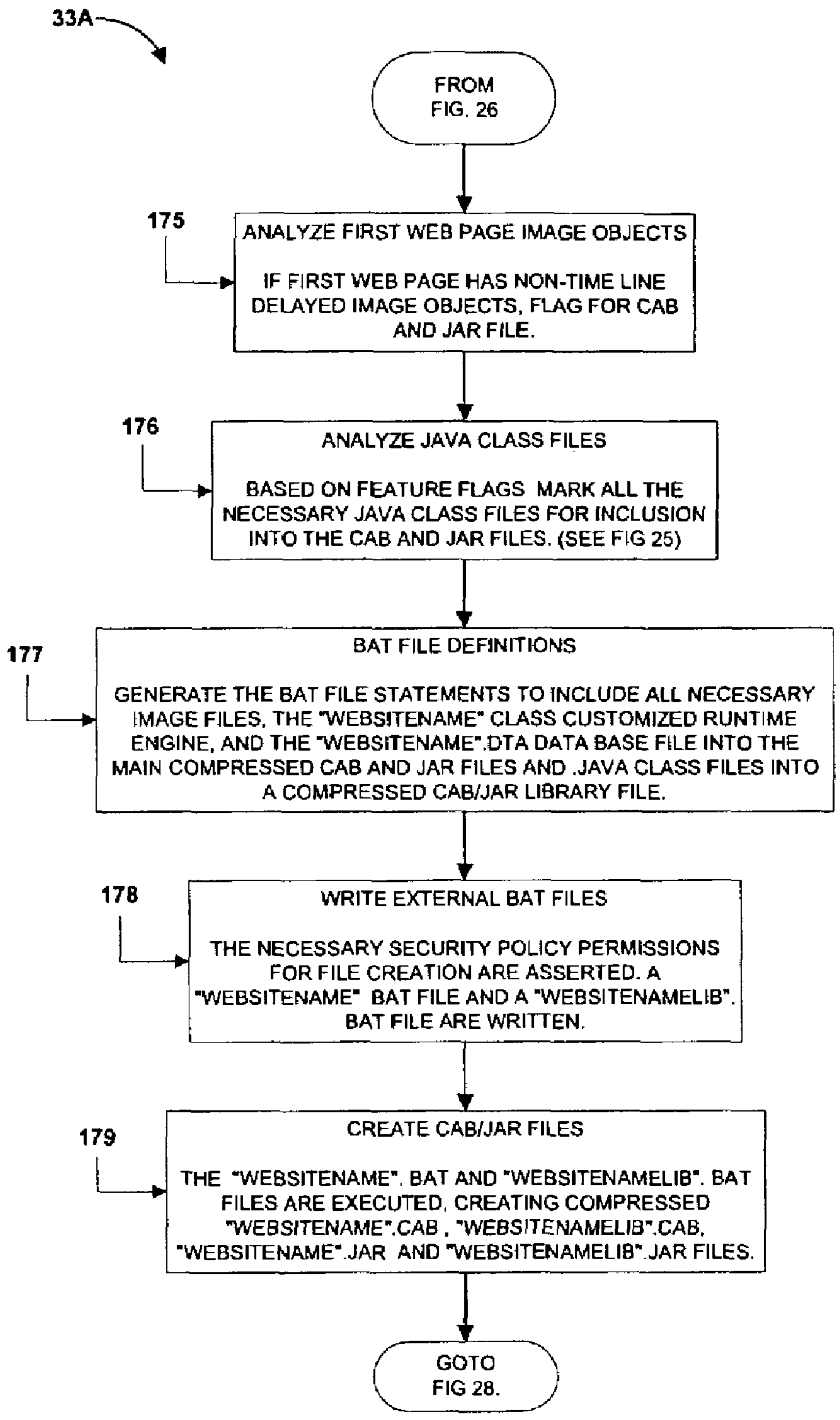

33A
FROM FIG. 26
175
ANALYZE FIRST WEB PAGE IMAGE OBJECTS
IF FIRST WEB PAGE HAS NON-TIME LINE DELAYED IMAGE OBJECTS, FLAG FOR CAB AND JAR FILE.
176
ANALYZE JAVA CLASS FILES
BASED ON FEATURE FLAGS MARK ALL THE NECESSARY JAVA CLASS FILES FOR INCLUSION INTO THE CAB AND JAR FILES. (SEE FIG 25)
177
BAT FILE DEFINITIONS
GENERATE THE BAT FILE STATEMENTS TO INCLUDE ALL NECESSARY IMAGE FILES, THE "WEBSITENAME" CLASS CUSTOMIZED RUNTIME ENGINE, AND THE "WEBSITENAME".DTA DATA BASE FILE INTO THE MAIN COMPRESSED CAB AND JAR FILES AND .JAVA CLASS FILES INTO A COMPRESSED CAB/JAR LIBRARY FILE.
178
WRITE EXTERNAL BAT FILES
THE NECESSARY SECURITY POLICY PERMISSIONS FOR FILE CREATION ARE ASSERTED. A "WEBSITENAME" BAT FILE AND A "WEBSITENAMELIB". BAT FILE ARE WRITTEN.
179
CREATE CAB/JAR FILES
THE "WEBSITENAME". BAT AND "WEBSITENAMELIB". BAT FILES ARE EXECUTED, CREATING COMPRESSED "WEBSITENAME".CAB , "WEBSITENAMELIB".CAB, "WEBSITENAME".JAR AND "WEBSITENAMELIB".JAR FILES.
GOTO FIG 28.

Fig. 27

CREATE THE CAB/JAR FILES

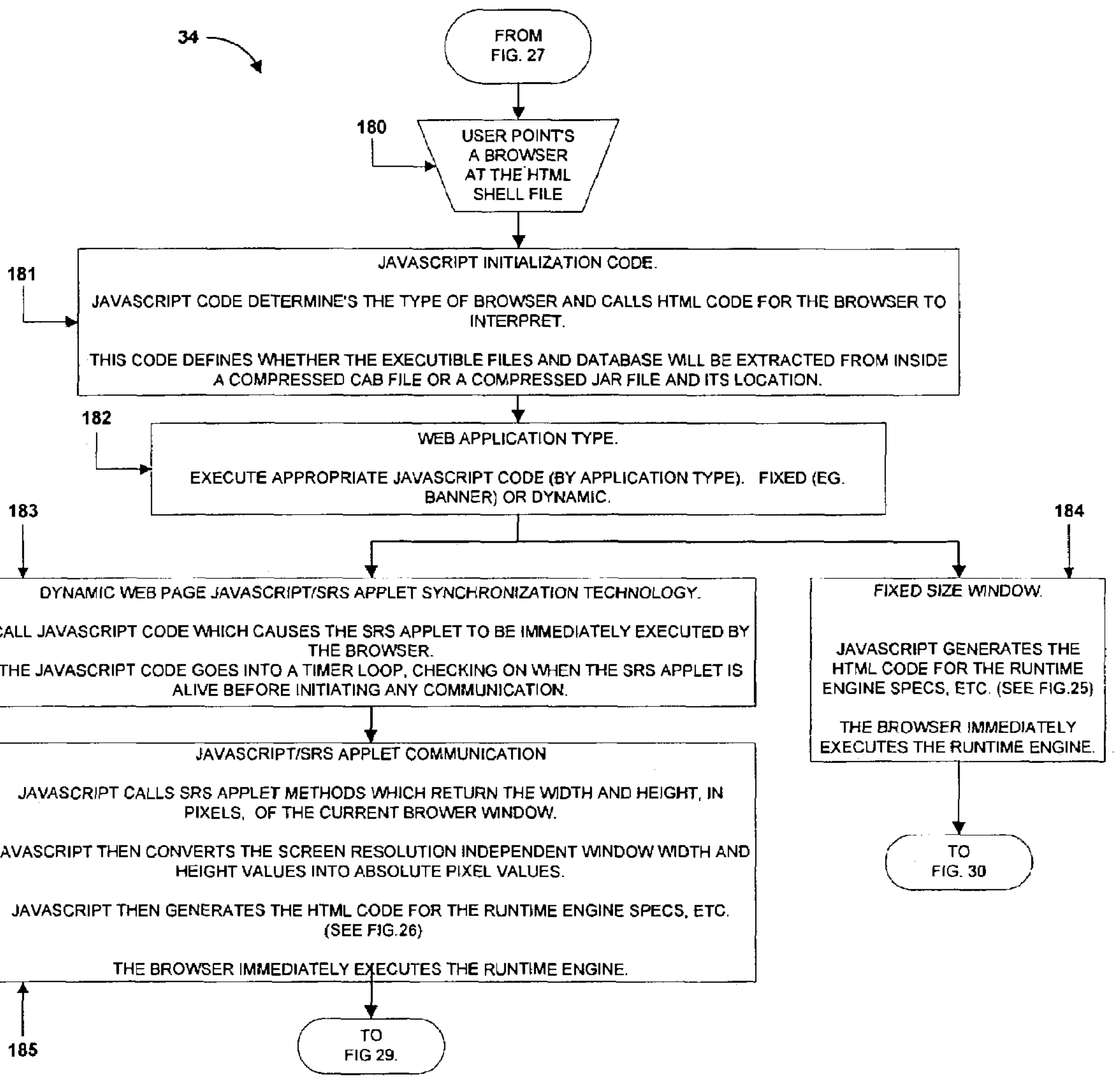

34
FROM FIG. 27
180
USER POINTS A BROWSER AT THE HTML SHELL FILE
181
JAVASCRIPT INITIALIZATION CODE.
JAVASCRIPT CODE DETERMINE'S THE TYPE OF BROWSER AND CALLS HTML CODE FOR THE BROWSER TO INTERPRET.
THIS CODE DEFINES WHETHER THE EXECUTIBLE FILES AND DATABASE WILL BE EXTRACTED FROM INSIDE A COMPRESSED CAB FILE OR A COMPRESSED JAR FILE AND ITS LOCATION.
182
WEB APPLICATION TYPE.
EXECUTE APPROPRIATE JAVASCRIPT CODE (BY APPLICATION TYPE). FIXED (EG. BANNER) OR DYNAMIC.
183
184
DYNAMIC WEB PAGE JAVASCRIPT/SRS APPLET SYNCHRONIZATION TECHNOLOGY.
CALL JAVASCRIPT CODE WHICH CAUSES THE SRS APPLET TO BE IMMEDIATELY EXECUTED BY THE BROWSER.
THE JAVASCRIPT CODE GOES INTO A TIMER LOOP, CHECKING ON WHEN THE SRS APPLET IS ALIVE BEFORE INITIATING ANY COMMUNICATION.
FIXED SIZE WINDOW.
JAVASCRIPT GENERATES THE HTML CODE FOR THE RUNTIME ENGINE SPECS, ETC. (SEE FIG.25)
THE BROWSER IMMEDIATELY EXECUTES THE RUNTIME ENGINE.
JAVASCRIPT/SRS APPLET COMMUNICATION
JAVASCRIPT CALLS SRS APPLET METHODS WHICH RETURN THE WIDTH AND HEIGHT, IN PIXELS, OF THE CURRENT BROWER WINDOW.
JAVASCRIPT THEN CONVERTS THE SCREEN RESOLUTION INDEPENDENT WINDOW WIDTH AND HEIGHT VALUES INTO ABSOLUTE PIXEL VALUES.
JAVASCRIPT THEN GENERATES THE HTML CODE FOR THE RUNTIME ENGINE SPECS, ETC. (SEE FIG.26)
THE BROWSER IMMEDIATELY EXECUTES THE RUNTIME ENGINE.
TO FIG. 30
185
TO FIG 29.

Fig. 28

WEB PAGE SIZE GENERATION TECHNOLOGY

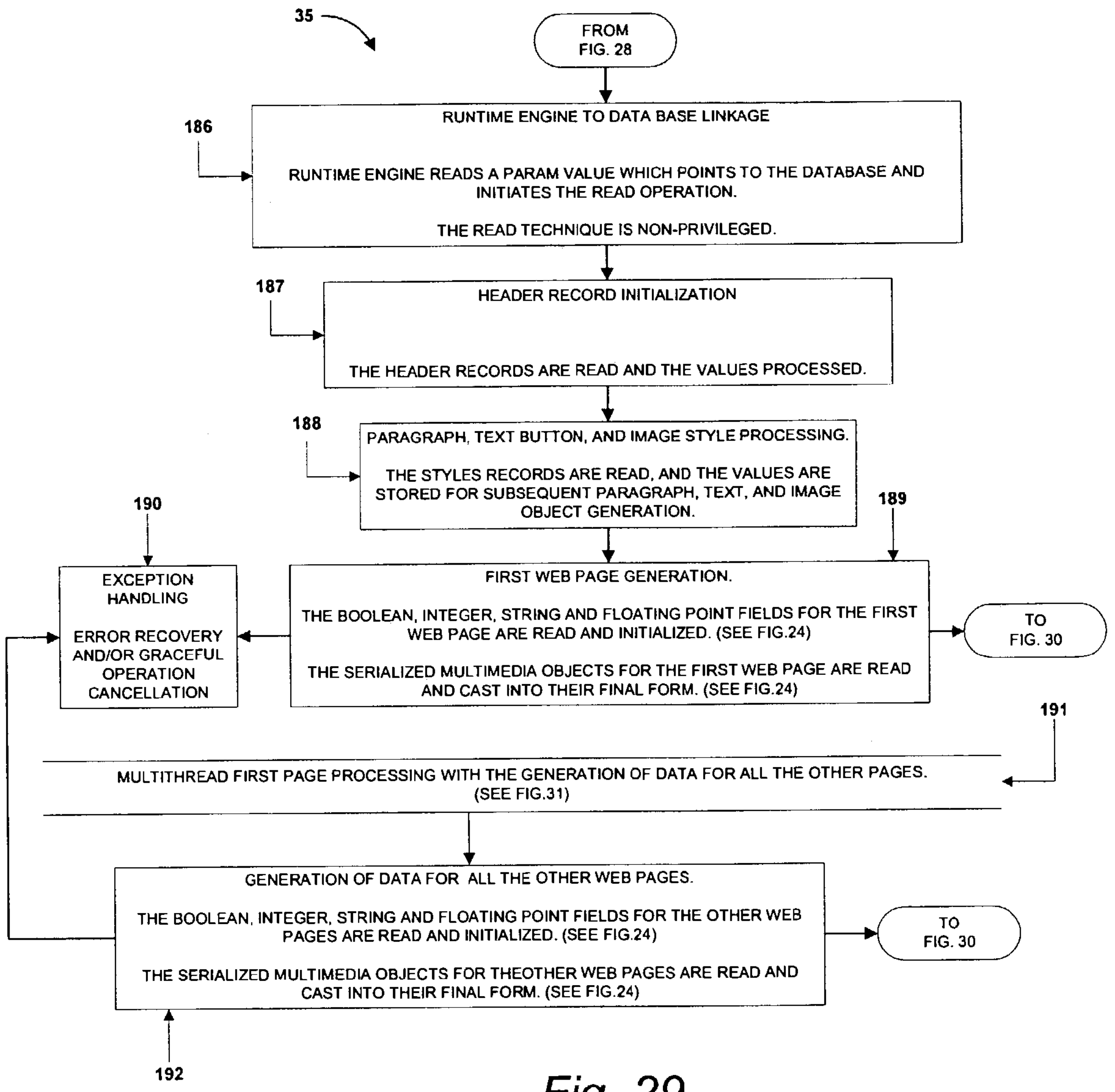

35
FROM FIG. 28
186
RUNTIME ENGINE TO DATA BASE LINKAGE
RUNTIME ENGINE READS A PARAM VALUE WHICH POINTS TO THE DATABASE AND INITIATES THE READ OPERATION.
THE READ TECHNIQUE IS NON-PRIVILEGED.
187
HEADER RECORD INITIALIZATION
THE HEADER RECORDS ARE READ AND THE VALUES PROCESSED.
188
PARAGRAPH, TEXT BUTTON, AND IMAGE STYLE PROCESSING.
THE STYLES RECORDS ARE READ, AND THE VALUES ARE STORED FOR SUBSEQUENT PARAGRAPH, TEXT, AND IMAGE OBJECT GENERATION.
190
189
EXCEPTION HANDLING
ERROR RECOVERY AND/OR GRACEFUL OPERATION CANCELLATION
FIRST WEB PAGE GENERATION.
THE BOOLEAN, INTEGER, STRING AND FLOATING POINT FIELDS FOR THE FIRST WEB PAGE ARE READ AND INITIALIZED. (SEE FIG.24)
THE SERIALIZED MULTIMEDIA OBJECTS FOR THE FIRST WEB PAGE ARE READ AND CAST INTO THEIR FINAL FORM. (SEE FIG.24)
TO FIG. 30
191
MULTITHREAD FIRST PAGE PROCESSING WITH THE GENERATION OF DATA FOR ALL THE OTHER PAGES. (SEE FIG.31)
GENERATION OF DATA FOR ALL THE OTHER WEB PAGES.
THE BOOLEAN, INTEGER, STRING AND FLOATING POINT FIELDS FOR THE OTHER WEB PAGES ARE READ AND INITIALIZED. (SEE FIG.24)
THE SERIALIZED MULTIMEDIA OBJECTS FOR THEOTHER WEB PAGES ARE READ AND CAST INTO THEIR FINAL FORM. (SEE FIG.24)
TO FIG. 30
192

Fig. 29

READ DATA BASE AND GENERATE NECESSARY OBJECTS.

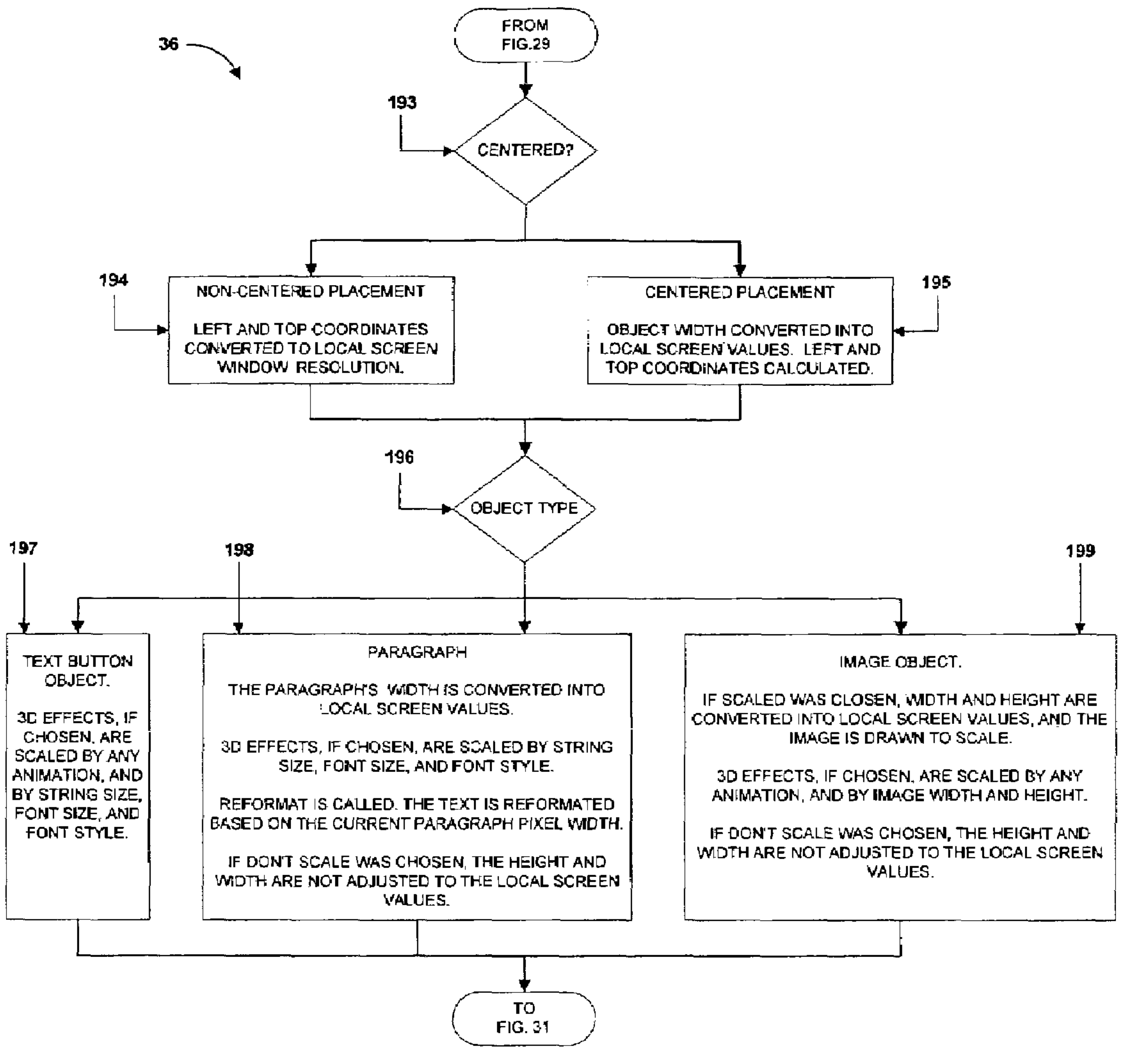

36
FROM FIG.29
193
CENTERED?
194
NON-CENTERED PLACEMENT
LEFT AND TOP COORDINATES CONVERTED TO LOCAL SCREEN WINDOW RESOLUTION.
195
CENTERED PLACEMENT
OBJECT WIDTH CONVERTED INTO LOCAL SCREEN VALUES. LEFT AND TOP COORDINATES CALCULATED.
196
OBJECT TYPE
197
198
199
TEXT BUTTON OBJECT.
3D EFFECTS, IF CHOSEN, ARE SCALED BY ANY ANIMATION, AND BY STRING SIZE, FONT SIZE, AND FONT STYLE.
PARAGRAPH
THE PARAGRAPH'S WIDTH IS CONVERTED INTO LOCAL SCREEN VALUES.
3D EFFECTS, IF CHOSEN, ARE SCALED BY STRING SIZE, FONT SIZE, AND FONT STYLE.
REFORMAT IS CALLED. THE TEXT IS REFORMATED BASED ON THE CURRENT PARAGRAPH PIXEL WIDTH.
IF DONT SCALE WAS CHOSEN, THE HEIGHT AND WIDTH ARE NOT ADJUSTED TO THE LOCAL SCREEN VALUES.
IMAGE OBJECT.
IF SCALED WAS CLOSEN, WIDTH AND HEIGHT ARE CONVERTED INTO LOCAL SCREEN VALUES, AND THE IMAGE IS DRAWN TO SCALE.
3D EFFECTS, IF CHOSEN, ARE SCALED BY ANY ANIMATION, AND BY IMAGE WIDTH AND HEIGHT.
IF DON'T SCALE WAS CHOSEN, THE HEIGHT AND WIDTH ARE NOT ADJUSTED TO THE LOCAL SCREEN VALUES.
TO FIG. 31

Fig. 30

WEB PAGE GENERATION WITH SCALING TECHNOLOGY

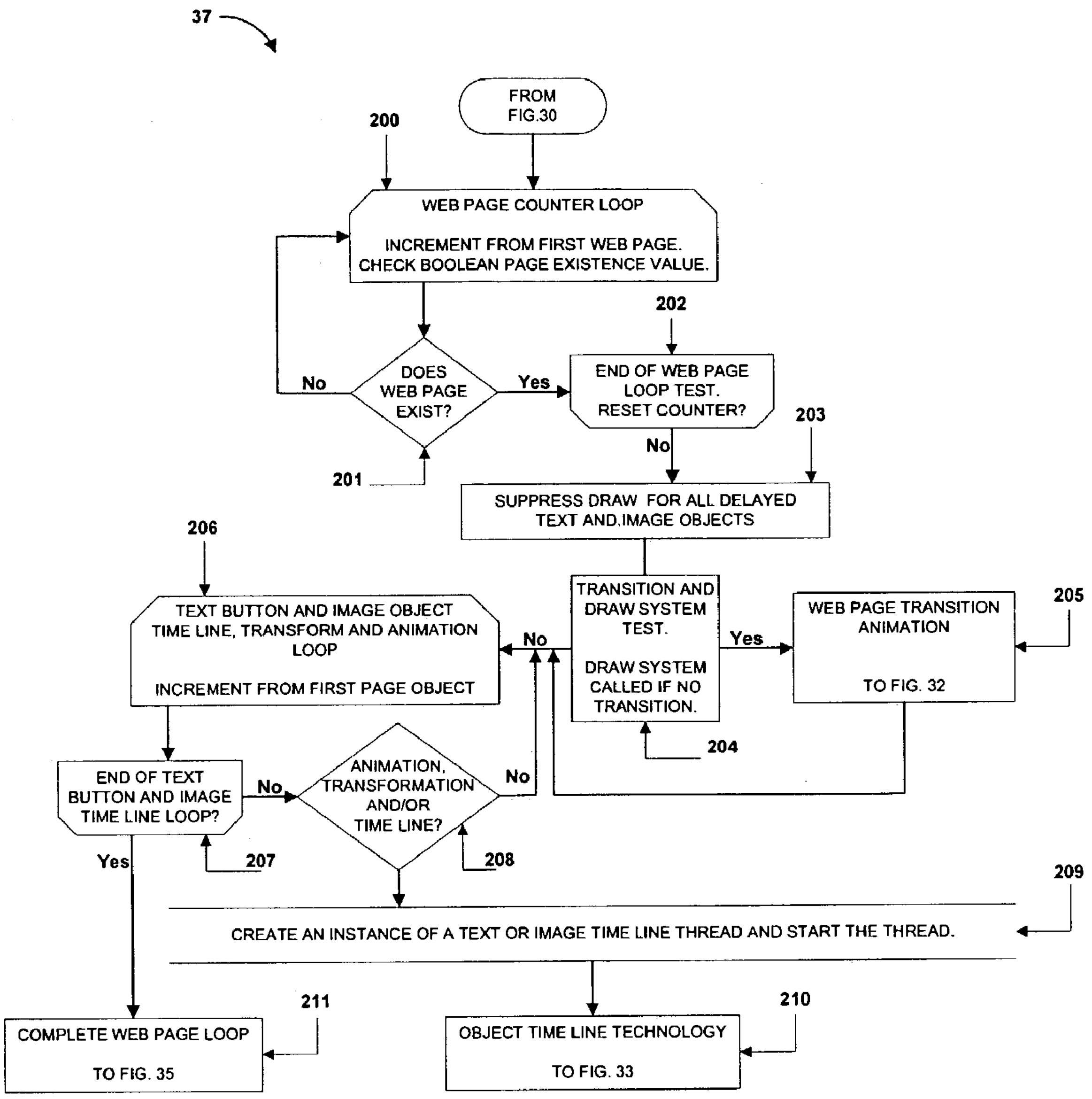

37
FROM FIG.30
200
WEB PAGE COUNTER LOOP
INCREMENT FROM FIRST WEB PAGE.
CHECK BOOLEAN PAGE EXISTENCE VALUE.
202
No
DOES WEB PAGE EXIST?
Yes
END OF WEB PAGE LOOP TEST. RESET COUNTER?
203
No
201
SUPPRESS DRAW FOR ALL DELAYED TEXT AND,IMAGE OBJECTS
206
TEXT BUTTON AND IMAGE OBJECT TIME LINE, TRANSFORM AND ANIMATION LOOP
INCREMENT FROM FIRST PAGE OBJECT
No
TRANSITION AND DRAW SYSTEM TEST.
DRAW SYSTEM CALLED IF NO TRANSITION.
Yes
WEB PAGE TRANSITION ANIMATION
TO FIG. 32
205
204
END OF TEXT BUTTON AND IMAGE TIME LINE LOOP?
No
ANIMATION, TRANSFORMATION AND/OR TIME LINE?
No
Yes
207
208
209
CREATE AN INSTANCE OF A TEXT OR IMAGE TIME LINE THREAD AND START THE THREAD.
211
210
COMPLETE WEB PAGE LOOP
TO FIG. 35
OBJECT TIME LINE TECHNOLOGY
TO FIG. 33

Fig. 31

THE MULTILEVEL WEB PAGE AND OBJECT THREAD TECHNOLOGY

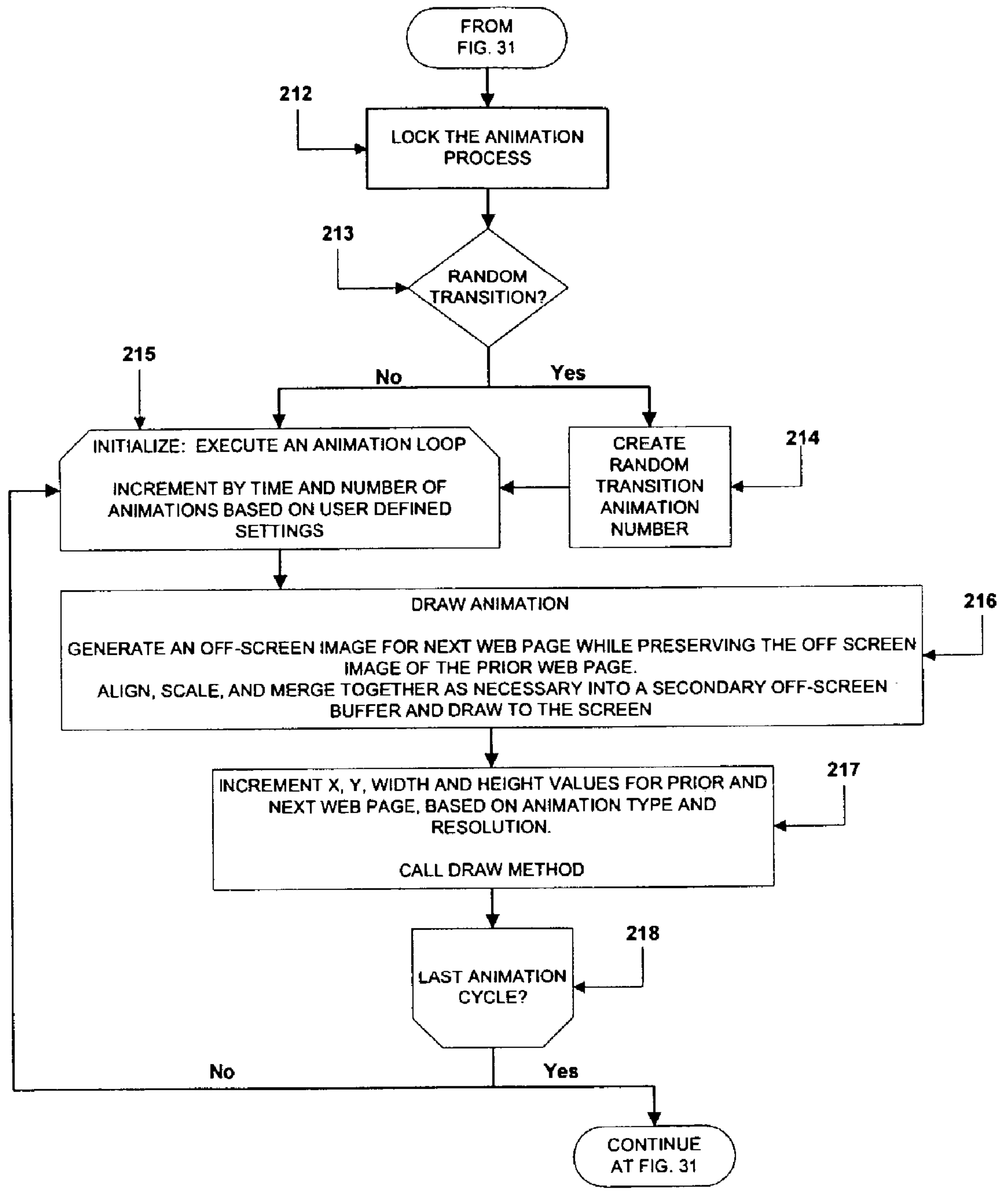

FROM FIG. 31
212
LOCK THE ANIMATION PROCESS
213
RANDOM TRANSITION?
215
No
Yes
INITIALIZE: EXECUTE AN ANIMATION LOOP
INCREMENT BY TIME AND NUMBER OF ANIMATIONS BASED ON USER DEFINED SETTINGS
CREATE RANDOM TRANSITION ANIMATION NUMBER
214
DRAW ANIMATION
GENERATE AN OFF-SCREEN IMAGE FOR NEXT WEB PAGE WHILE PRESERVING THE OFF SCREEN IMAGE OF THE PRIOR WEB PAGE.
ALIGN, SCALE, AND MERGE TOGETHER AS NECESSARY INTO A SECONDARY OFF-SCREEN BUFFER AND DRAW TO THE SCREEN
216
INCREMENT X, Y, WIDTH AND HEIGHT VALUES FOR PRIOR AND NEXT WEB PAGE, BASED ON ANIMATION TYPE AND RESOLUTION.
CALL DRAW METHOD
217
LAST ANIMATION CYCLE?
218
No
Yes
CONTINUE AT FIG. 31

Fig. 32

WEB PAGE TRANSITION ANIMATION

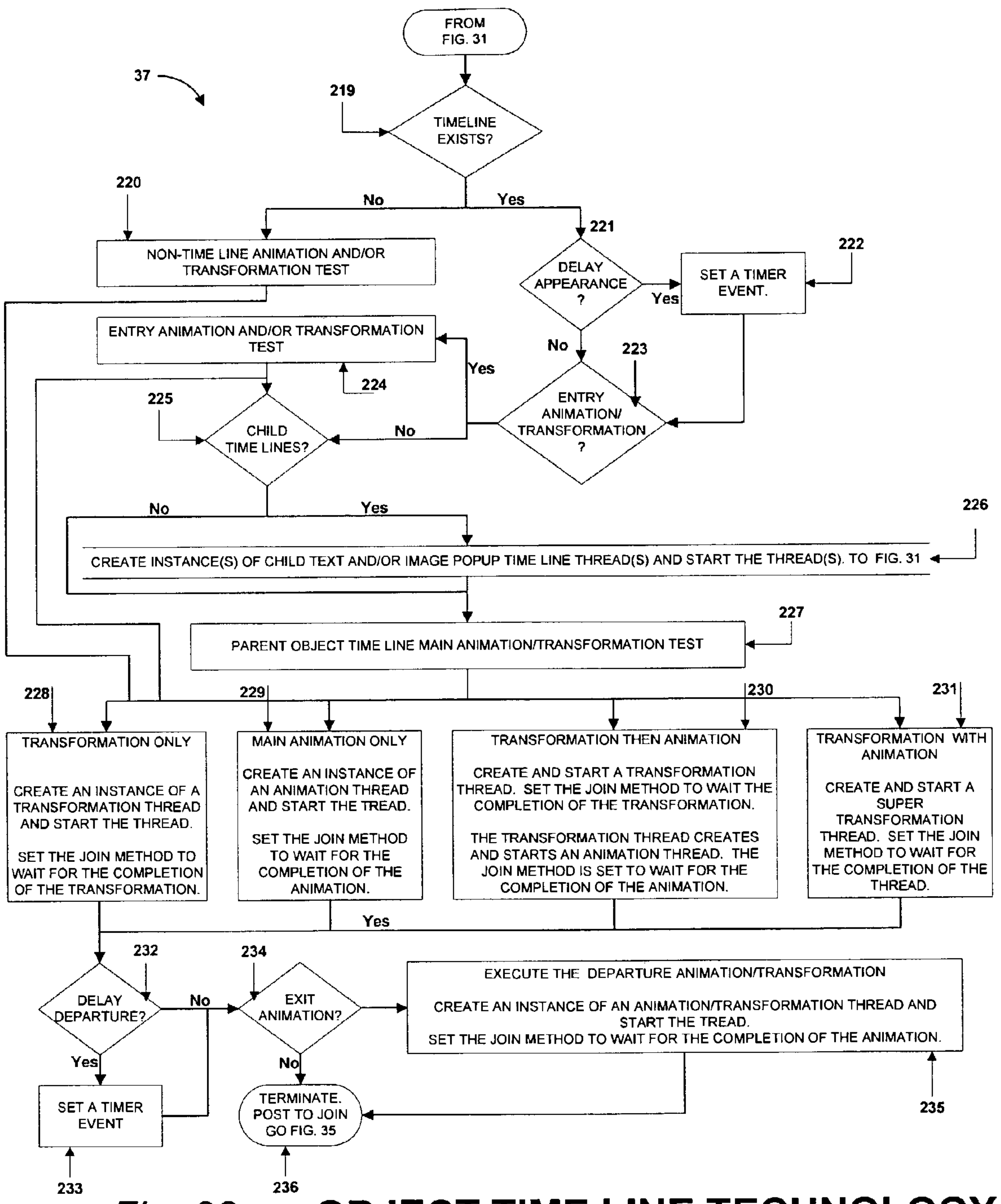

FROM FIG. 31
37
219
TIMELINE EXISTS?
220
No
Yes
221
NON-TIME LINE ANIMATION AND/OR TRANSFORMATION TEST
DELAY APPEARANCE ?
Yes
SET A TIMER EVENT.
222
ENTRY ANIMATION AND/OR TRANSFORMATION TEST
Yes
No
223
224
225
ENTRY ANIMATION/ TRANSFORMATION ?
CHILD TIME LINES?
No
No
Yes
226
CREATE INSTANCE(S) OF CHILD TEXT AND/OR IMAGE POPUP TIME LINE THREAD(S) AND START THE THREAD(S). TO FIG. 31
227
PARENT OBJECT TIME LINE MAIN ANIMATION/TRANSFORMATION TEST
228
229
230
231
TRANSFORMATION ONLY
CREATE AN INSTANCE OF A TRANSFORMATION THREAD AND START THE THREAD.
SET THE JOIN METHOD TO WAIT FOR THE COMPLETION OF THE TRANSFORMATION.
MAIN ANIMATION ONLY
CREATE AN INSTANCE OF AN ANIMATION THREAD AND START THE TREAD.
SET THE JOIN METHOD TO WAIT FOR THE COMPLETION OF THE ANIMATION.
TRANSFORMATION THEN ANIMATION
CREATE AND START A TRANSFORMATION THREAD. SET THE JOIN METHOD TO WAIT THE COMPLETION OF THE TRANSFORMATION.
THE TRANSFORMATION THREAD CREATES AND STARTS AN ANIMATION THREAD. THE JOIN METHOD IS SET TO WAIT FOR THE COMPLETION OF THE ANIMATION.
TRANSFORMATION WITH ANIMATION
CREATE AND START A SUPER TRANSFORMATION THREAD. SET THE JOIN METHOD TO WAIT FOR THE COMPLETION OF THE THREAD.
Yes
232
234
DELAY DEPARTURE?
No
EXIT ANIMATION?
EXECUTE THE DEPARTURE ANIMATION/TRANSFORMATION
CREATE AN INSTANCE OF AN ANIMATION/TRANSFORMATION THREAD AND START THE TREAD.
SET THE JOIN METHOD TO WAIT FOR THE COMPLETION OF THE ANIMATION.
Yes
No
SET A TIMER EVENT
TERMINATE. POST TO JOIN GO FIG. 35
235
233
236

Fig. 33 **OBJECT TIME LINE TECHNOLOGY**

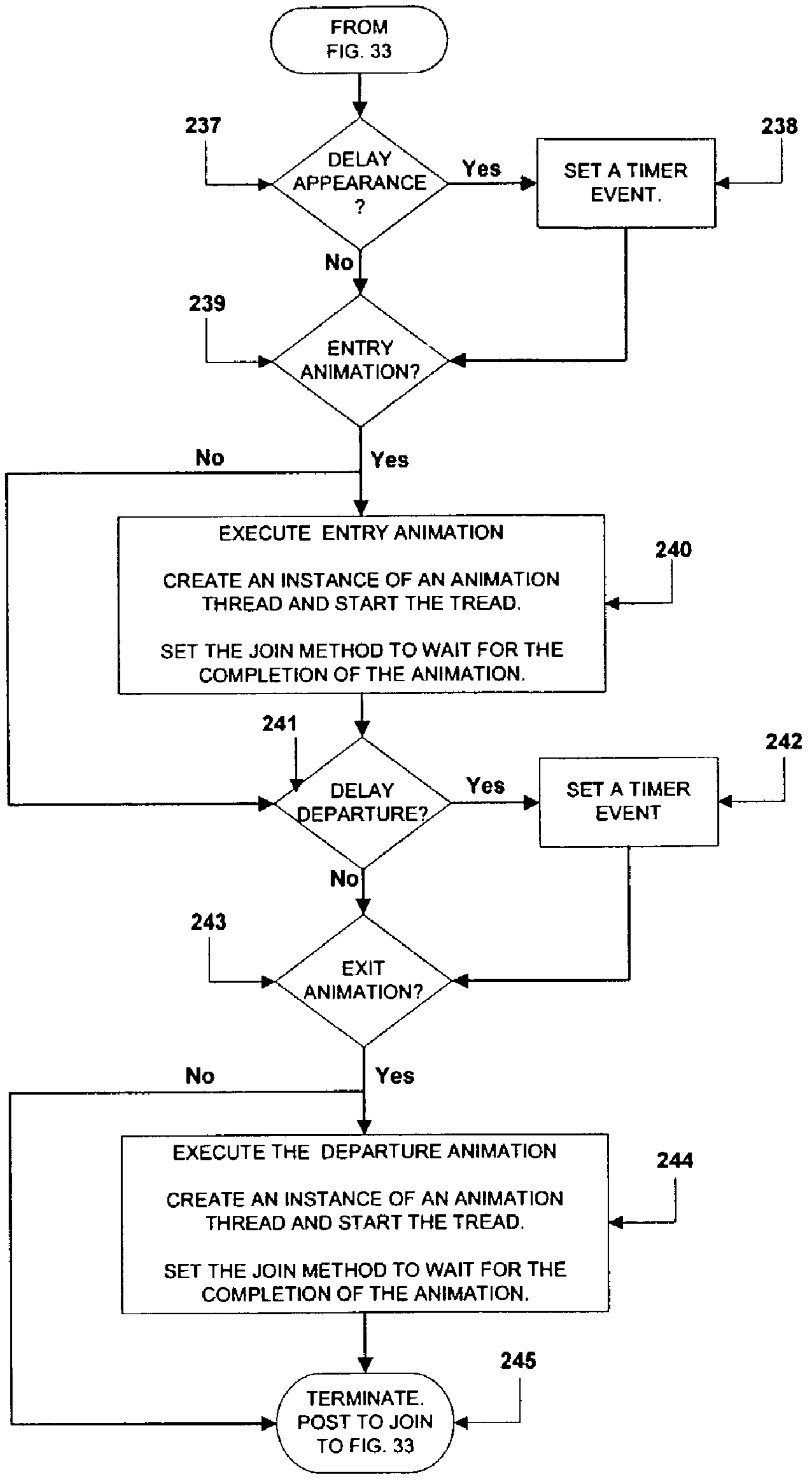

FROM FIG. 33
237
DELAY APPEARANCE ?
Yes
SET A TIMER EVENT.
238
No
239
ENTRY ANIMATION?
No
Yes
EXECUTE ENTRY ANIMATION
CREATE AN INSTANCE OF AN ANIMATION THREAD AND START THE TREAD.
SET THE JOIN METHOD TO WAIT FOR THE COMPLETION OF THE ANIMATION.
240
241
DELAY DEPARTURE?
Yes
SET A TIMER EVENT
242
No
243
EXIT ANIMATION?
No
Yes
EXECUTE THE DEPARTURE ANIMATION
CREATE AN INSTANCE OF AN ANIMATION THREAD AND START THE TREAD.
SET THE JOIN METHOD TO WAIT FOR THE COMPLETION OF THE ANIMATION.
244
245
TERMINATE. POST TO JOIN TO FIG. 33

Fig. 34

CHILD TIME LINES FOR TEXT BUTTON AND IMAGE OBJECTS

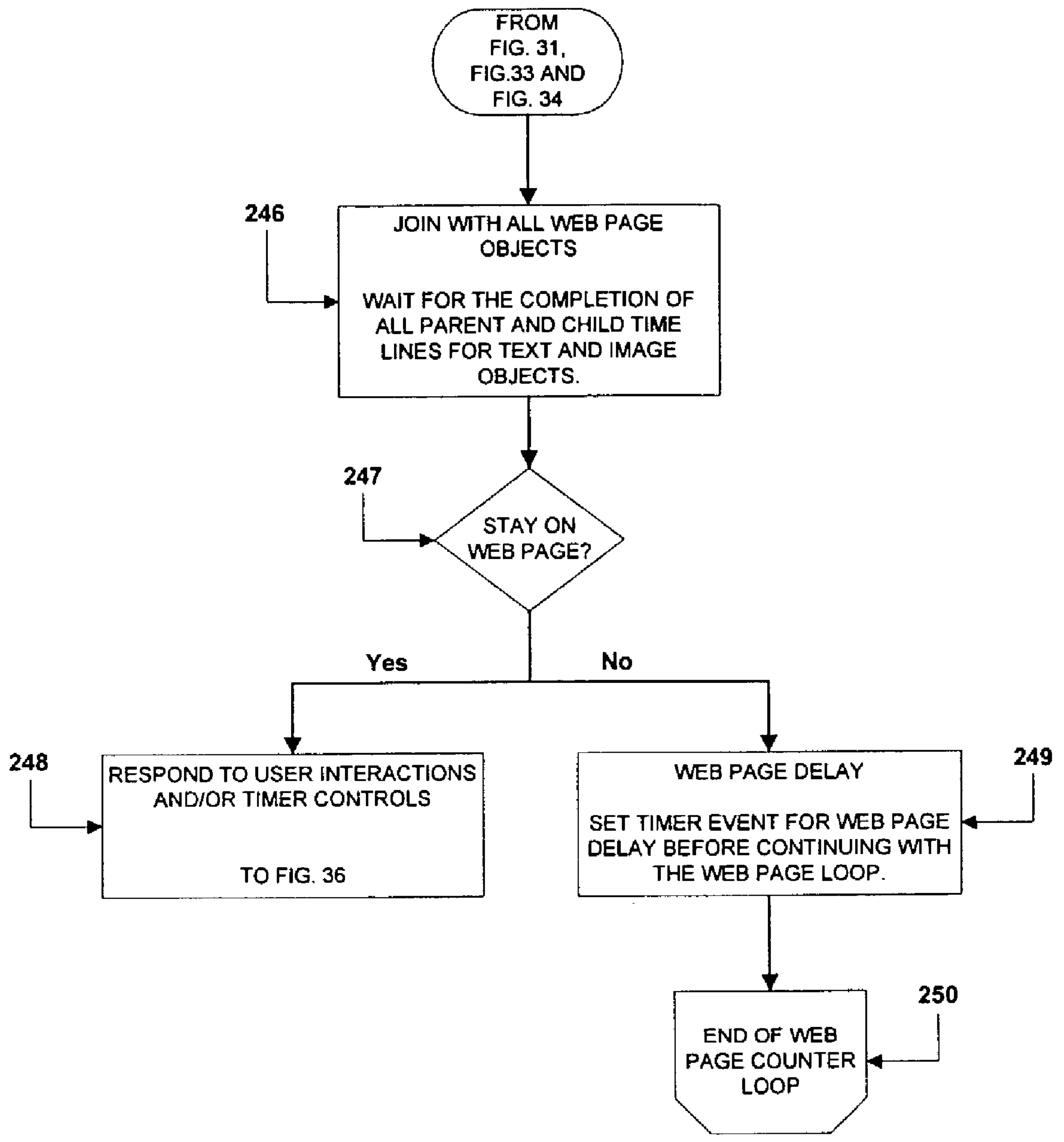

FROM FIG. 31, FIG.33 AND FIG. 34
246
JOIN WITH ALL WEB PAGE OBJECTS
WAIT FOR THE COMPLETION OF ALL PARENT AND CHILD TIME LINES FOR TEXT AND IMAGE OBJECTS.
247
STAY ON WEB PAGE?
Yes
No
248
RESPOND TO USER INTERACTIONS AND/OR TIMER CONTROLS
TO FIG. 36
WEB PAGE DELAY
SET TIMER EVENT FOR WEB PAGE DELAY BEFORE CONTINUING WITH THE WEB PAGE LOOP.
249
END OF WEB PAGE COUNTER LOOP
250

Fig. 35

COMPLETE WEB PAGE AND OBJECT THREAD LOOP

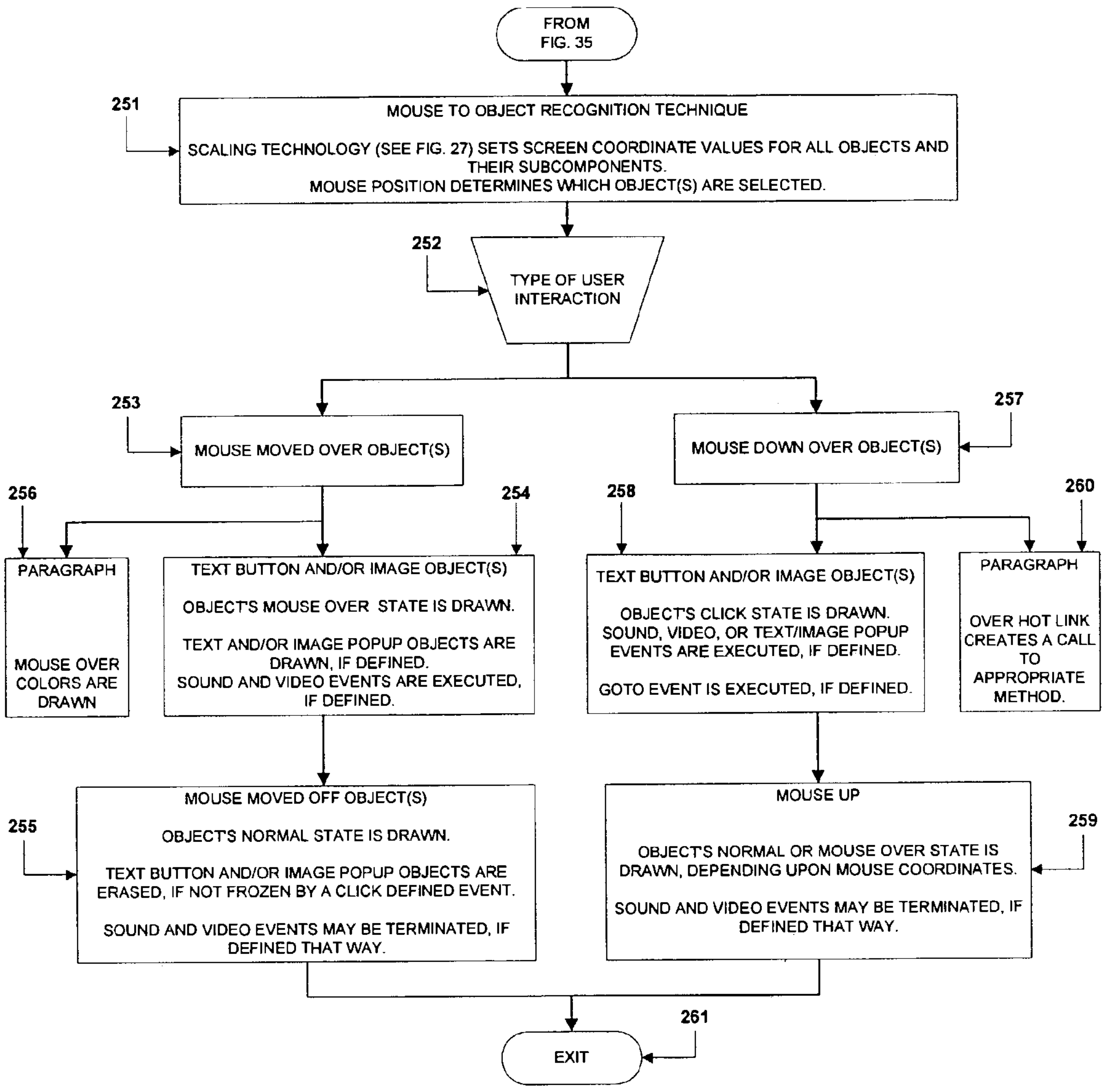
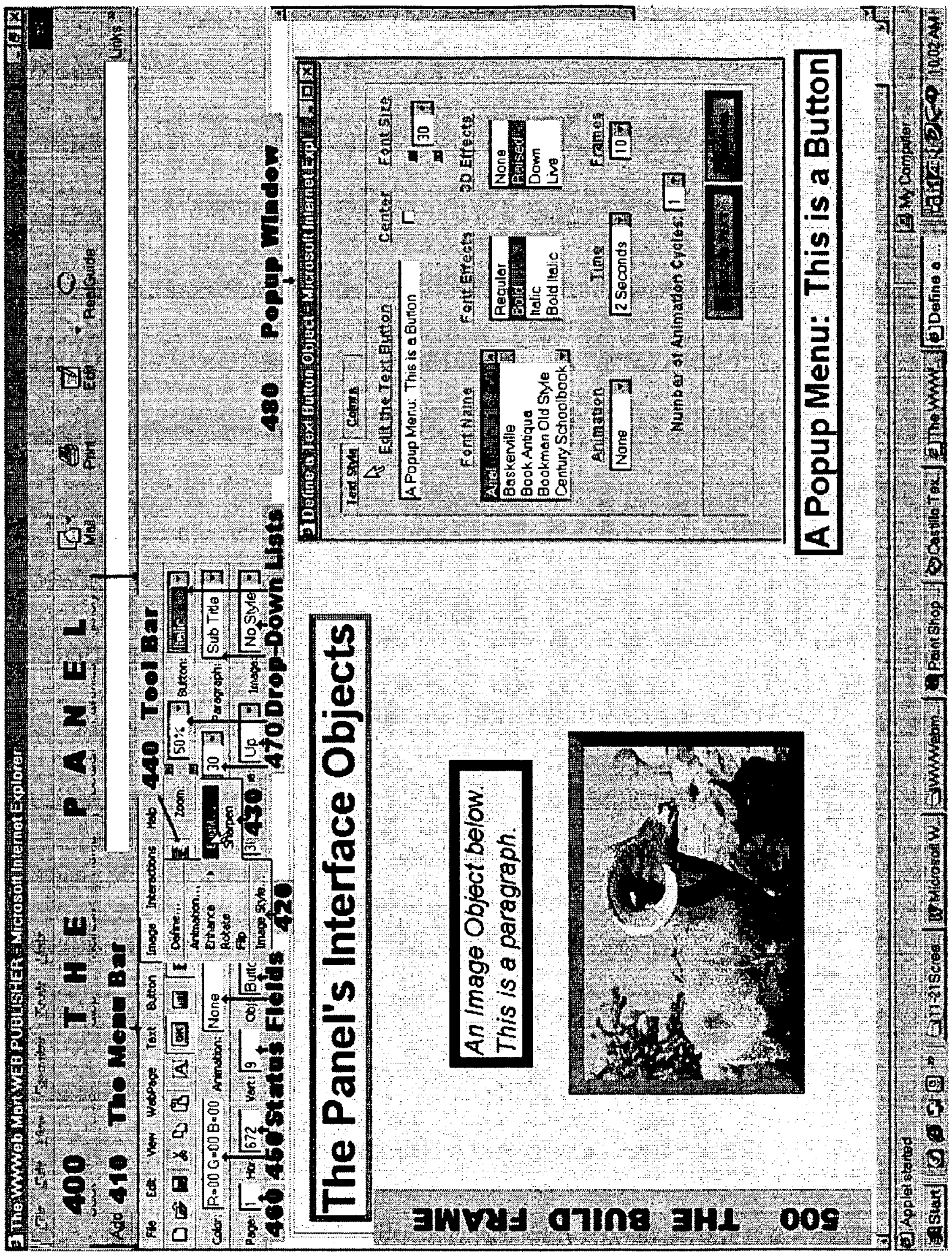

FROM FIG. 35
251
MOUSE TO OBJECT RECOGNITION TECHNIQUE
SCALING TECHNOLOGY (SEE FIG. 27) SETS SCREEN COORDINATE VALUES FOR ALL OBJECTS AND THEIR SUBCOMPONENTS.
MOUSE POSITION DETERMINES WHICH OBJECT(S) ARE SELECTED.
252
TYPE OF USER INTERACTION
253
MOUSE MOVED OVER OBJECT(S)
257
MOUSE DOWN OVER OBJECT(S)
256
PARAGRAPH
MOUSE OVER COLORS ARE DRAWN
254
TEXT BUTTON AND/OR IMAGE OBJECT(S)
OBJECT'S MOUSE OVER STATE IS DRAWN.
TEXT AND/OR IMAGE POPUP OBJECTS ARE DRAWN, IF DEFINED.
SOUND AND VIDEO EVENTS ARE EXECUTED, IF DEFINED.
258
TEXT BUTTON AND/OR IMAGE OBJECT(S)
OBJECT'S CLICK STATE IS DRAWN.
SOUND, VIDEO, OR TEXT/IMAGE POPUP EVENTS ARE EXECUTED, IF DEFINED.
GOTO EVENT IS EXECUTED, IF DEFINED.
260
PARAGRAPH
OVER HOT LINK CREATES A CALL TO APPROPRIATE METHOD.
255
MOUSE MOVED OFF OBJECT(S)
OBJECT'S NORMAL STATE IS DRAWN.
TEXT BUTTON AND/OR IMAGE POPUP OBJECTS ARE ERASED, IF NOT FROZEN BY A CLICK DEFINED EVENT.
SOUND AND VIDEO EVENTS MAY BE TERMINATED, IF DEFINED THAT WAY.
259
MOUSE UP
OBJECT'S NORMAL OR MOUSE OVER STATE IS DRAWN, DEPENDING UPON MOUSE COORDINATES.
SOUND AND VIDEO EVENTS MAY BE TERMINATED, IF DEFINED THAT WAY.
261
EXIT

Fig. 36

RESPOND TO USER INTERACTIONS



The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
400 THE PANEL
410 The Menu Bar
440 Tool Bar
450 Status Fields
460
430
420
470 Drop-Down Lists
480 Popup Window
500 THE BUILD FRAME
The Panel's Interface Objects
An Image Object below.
This is a paragraph.
Define Object Text Button Object - Microsoft Internet Explorer
Text Style
Colors
Edit the Text Button
A Popup Menu: This is a Button
Center
Font Size
30
Font Name
Arial
Baskerville
Book Antiqua
Bookman Old Style
Century Schoolbook
Font Effects
Regular
Bold
Italic
Bold Italic
3D Effects
None
Raised
Down
Live
Animation
None
Time
2 Seconds
Frames
10
Number of Animation Cycles: 1
A Popup Menu: This is a Button

Fig. 37

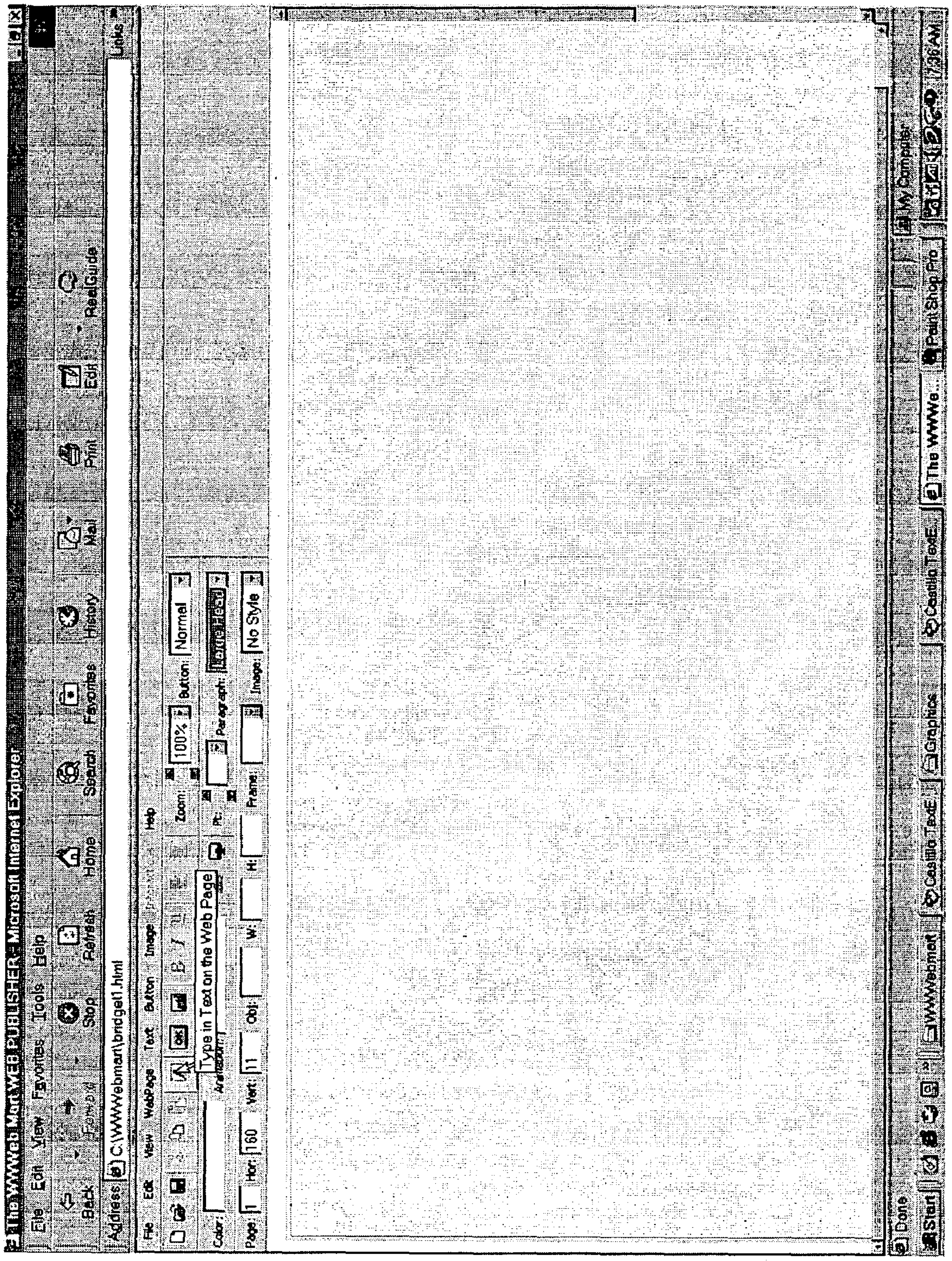

The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Address C:\WWWebmart\bridget1.html
Links
File Edit View WebPage Text Button Image Help
Zoom: 100% Button: Normal
Type in Text on the Web Page
Color: Arial Pt: Paragraph: Large Head
Page: 1 Hor: 160 Vert: 11 Obj: W: H: Frame: Image: No Style
Done
My Computer
Start WWWebmart Castillo TextE Graphics Castillo TextE The WWWe... Paint Shop Pro
7:36 AM

Fig. 38

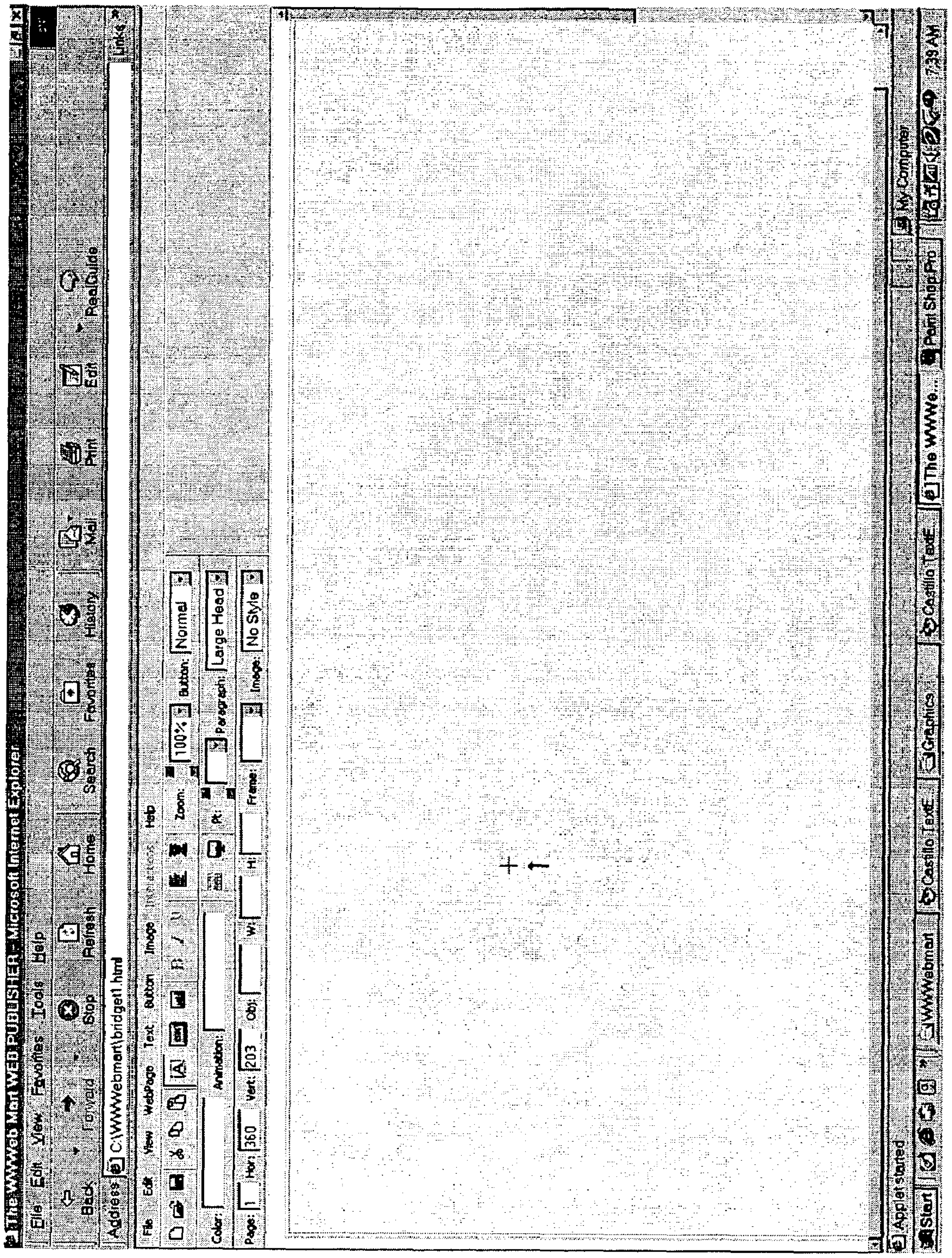
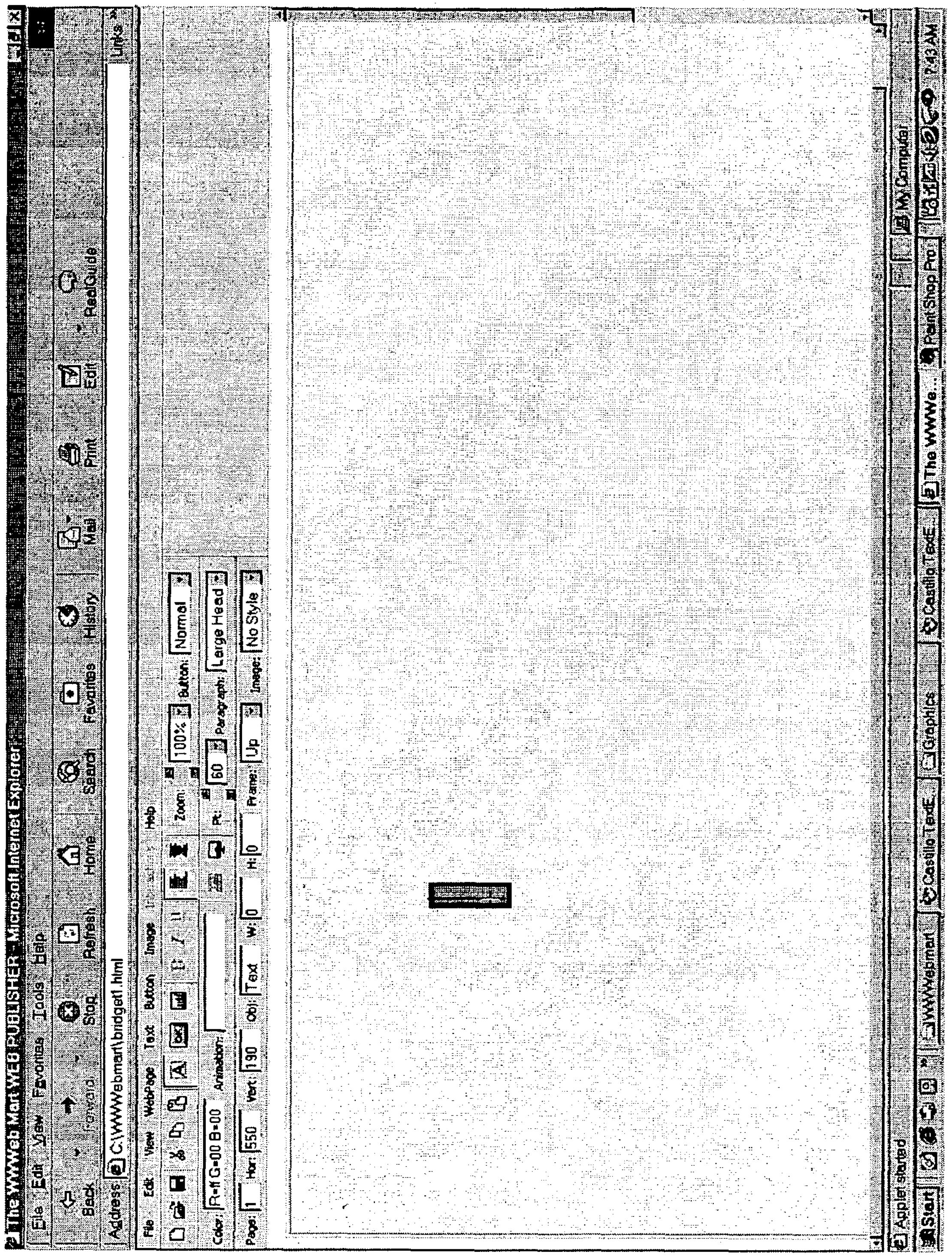

The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Address C:\WWWebmart\bridget1.html
Links
File Edit View WebPage Text Button Image Help
Zoom: 100% Button: Normal
Color: Animation: Pt: Paragraph: Large Head
Page: 1 Hor: 360 Vert: 203 Obj: W: H: Frame: Image: No Style
Applet started
My Computer
Start WWWebmart Castillo TextE Graphics Castillo TextE The WWWe... Paint Shop Pro
7:39 AM

Fig. 39



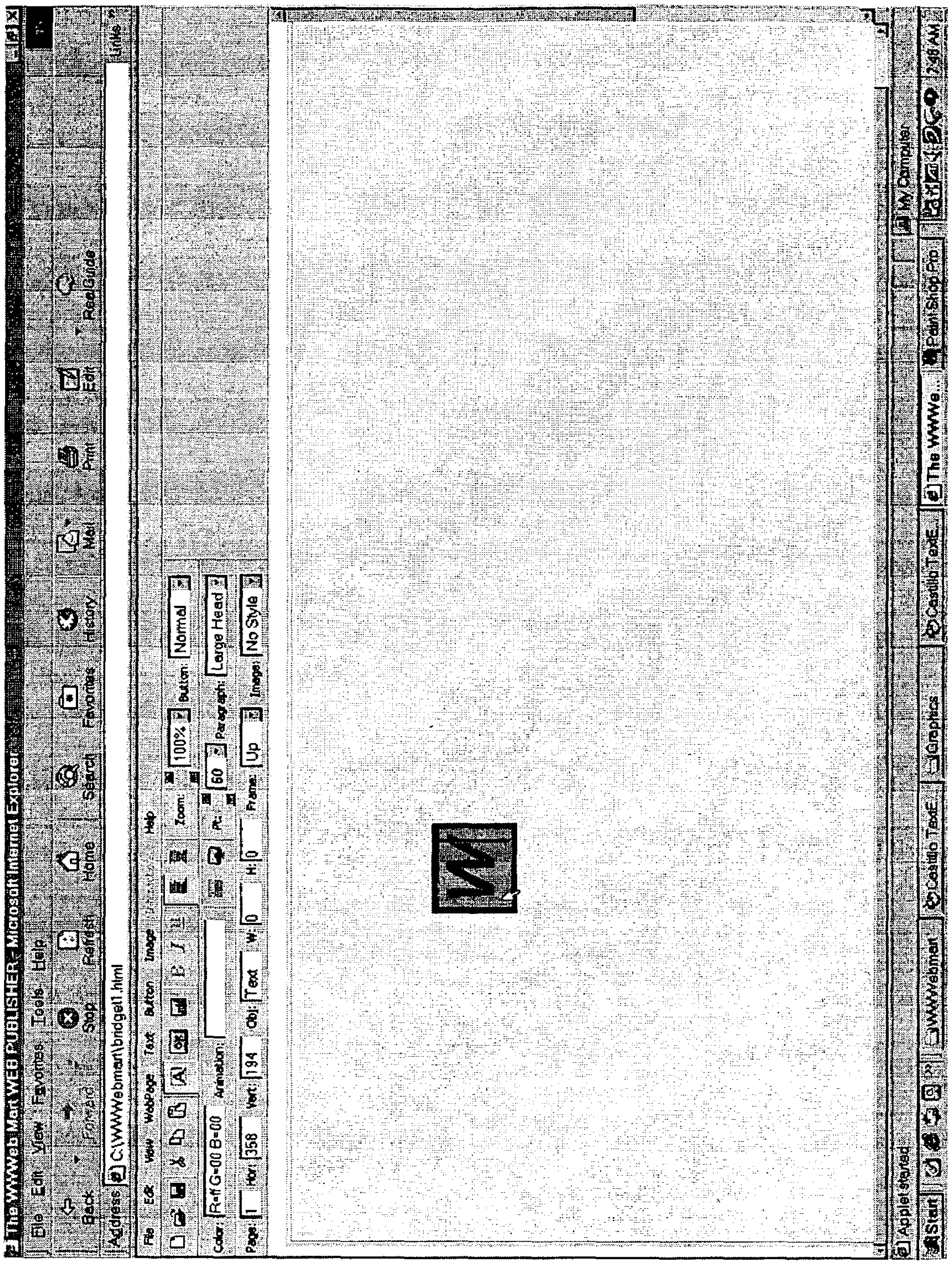
Address: C:\WWWebmart\bridget1.html

Zoom: 100% Button: Normal

Color: R=ff G=00 B=00 Animation: Pt: 60 Paragraph: Large Head

Page: 1 Hor: 550 Vert: 190 Obj: Text W: 0 H: 0 Frame: Up Image: No Style

Fig. 40


The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Links
File Edit View WebPage Text Button Image Help
Start
Applet started
My Computer
2:46 AM

Address C:\WWWebmart\bridget1.html

Zoom: 100% Button: Normal

Color: R=ff G=00 B=00 Animation: Pt: 60 Paragraph: Large Head

Page: 1 Hor: 358 Vert: 194 Obj: Text W: 0 H: 0 Frame: Up Image: No Style

Fig. 41

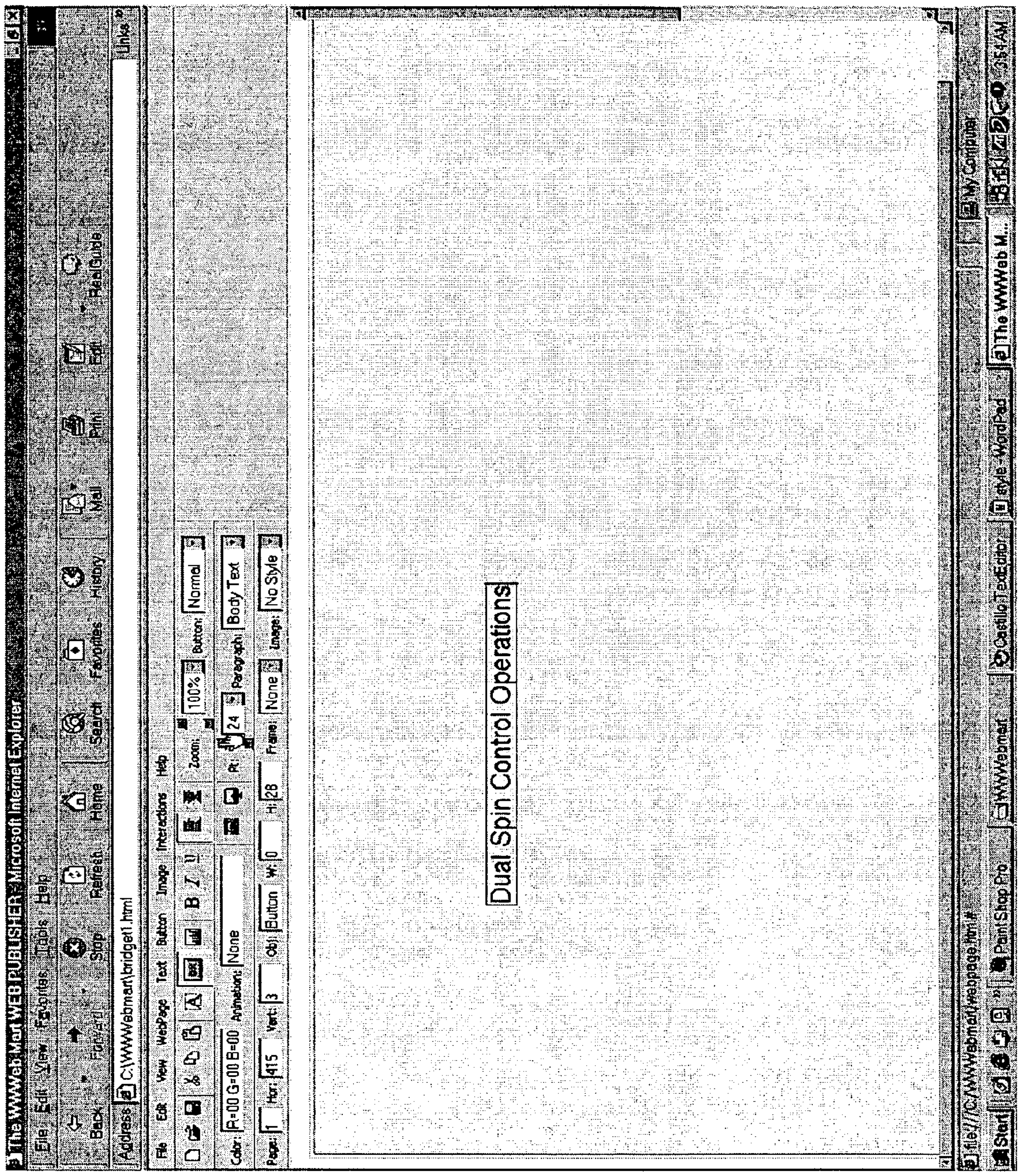

The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Address C:\WWWebmart\bridget1.html
Links
File Edit View WebPage Text Button Image Interactions Help
Zoom: 100%
Button: Normal
Color: R=00 G=00 B=00
Animation: None
Rt: 24
Paragraph: Body Text
Page: 1 Hor: 415 Vert: 3 Obj: Button W: 0 H: 28
Frame: None
Image: No Style
Dual Spin Control Operations
file://C:\WWWebmart\webpage.htm#
My Computer
Start
PaintShop Pro
WWWebmart
CastilloTextEditor
style - WordPad
The WWWeb M...
3:54 AM

Fig. 42

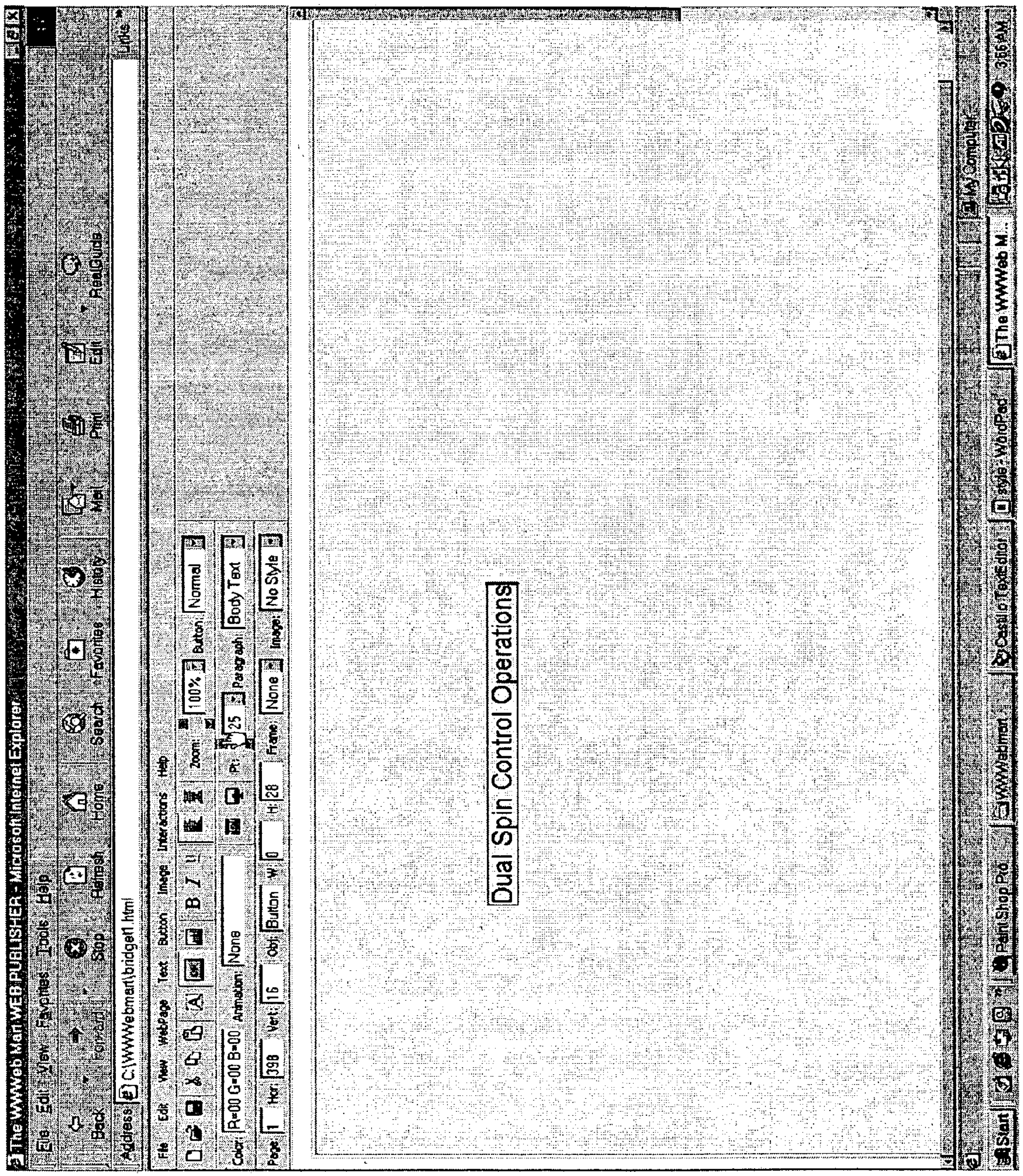
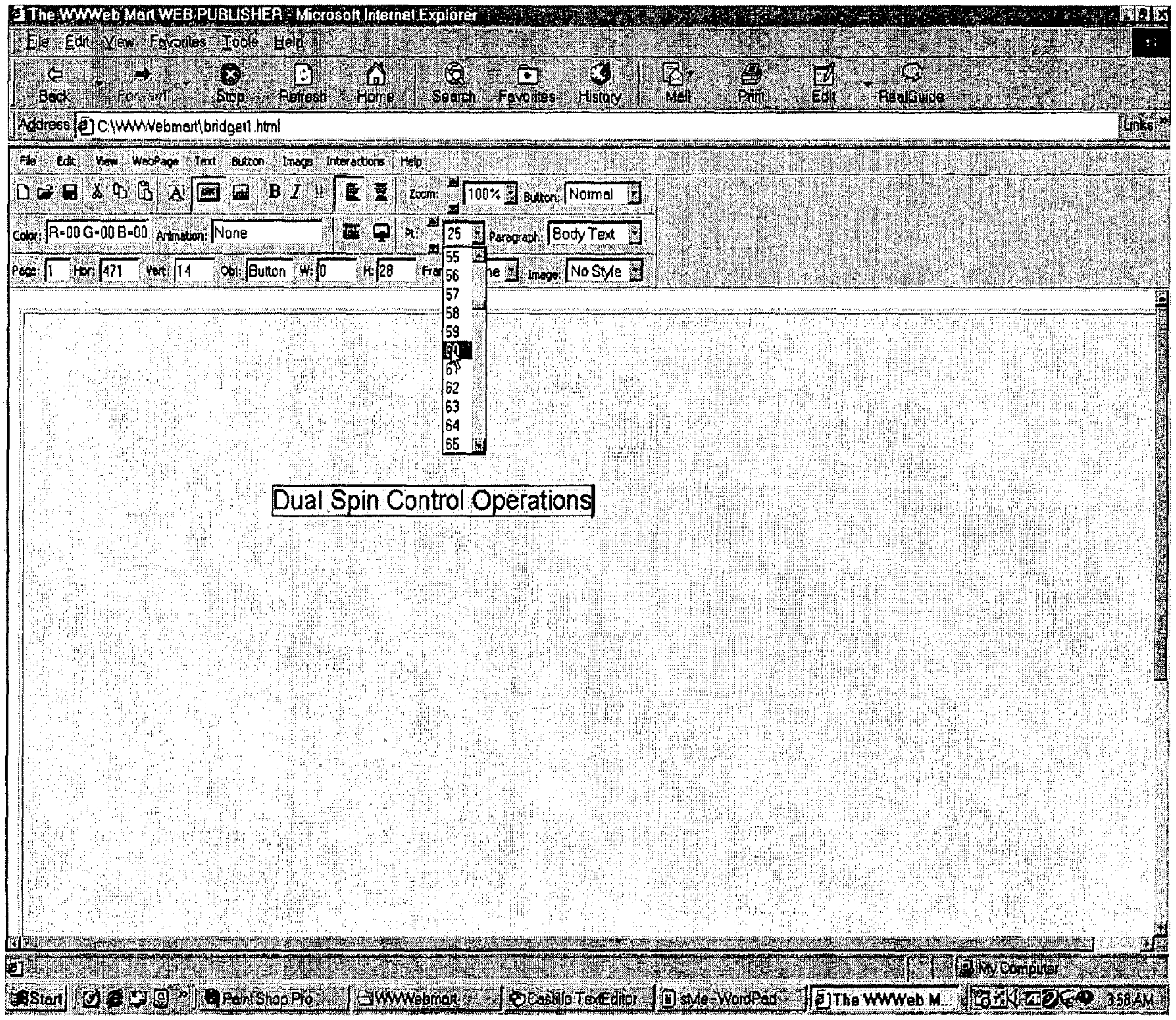

The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Address C:\WWWebmart\bridget1.html
Links
File Edit View WebPage Text Button Image Interactions Help
Zoom: 100% Button: Normal
Color: R=00 G=00 B=00 Animation: None Pt: 25 Paragraph: Body Text
Page: 1 Hor: 398 Vert: 16 Obj: Button W: 0 H: 28 Frame: None Image: No Style
Dual Spin Control Operations
Start Paint Shop Pro WWWebmart Casio TextEditor style - WordPad The WWWeb M... WJ Compuart 3:06 AM

Fig. 43


The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer

Address: C:\WWWebmart\bridget1.html

Zoom: 100% Button: Normal

Color: R=00 G=00 B=00 Animation: None Pt: 25 Paragraph: Body Text

Page: 1 Hor: 471 Vert: 14 Obj: Button W: 0 H: 28 Image: No Style

Dual Spin Control Operations

Fig. 44

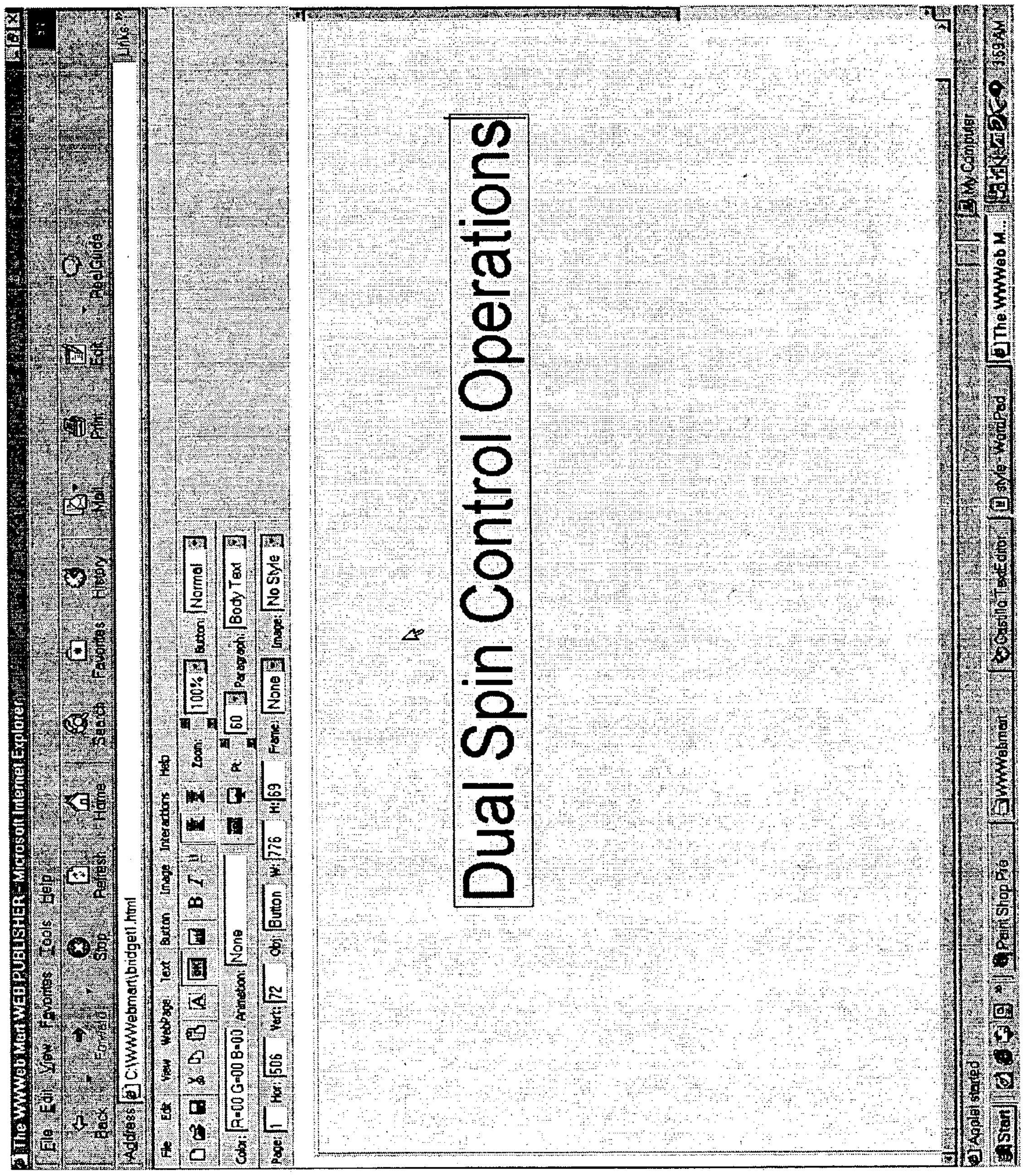

The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Address C:\WWWebmart\bridget1.html
Links
File Edit View WebPage Text Button Image Interactions Help
Zoom: 100%
Button: Normal
Color: R=00 G=00 B=00
Animation: None
Pt: 60
Paragraph: Body Text
Page: 1 Hor: 506 Vert: 72 Obj: Button W: 776 H: 69 Frame: None Image: No Style
Dual Spin Control Operations
Applet started
My Computer
Start
Paint Shop Pro
WWWebmart
Gabriel TextEditor
style - WordPad
The WWWeb M...
1:53 AM

Fig. 45

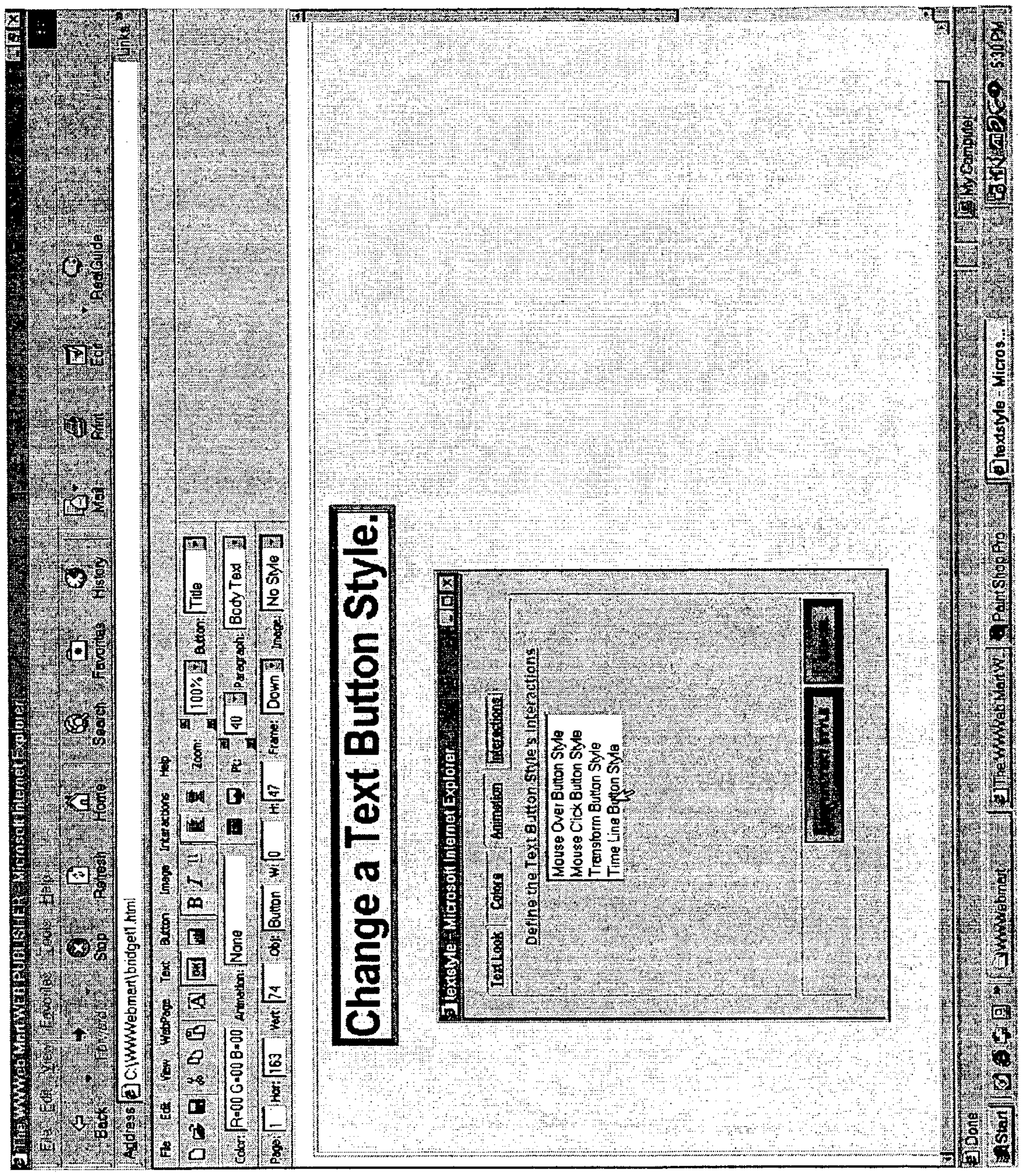

The WWWebMart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Address C:\WWWebmart\bridget1.html
Links
File Edit View WebPage Text Button Image Interactions Help
Zoom: 100% Button: Title
Color: R=00 G=00 B=00 Animation: None Pt: 40 Paragraph: Body Text
Page: 1 Hor: 163 Vert: 74 Obj: Button W: 0 H: 47 Frame: Down Image: No Style
Change a Text Button Style.
textstyle - Microsoft Internet Explorer
Text Look
Colors
Animation
Interactions
Define the Text Button Style's interactions
Mouse Over Button Style
Mouse Click Button Style
Transform Button Style
Time Line Button Style
Done
My Computer
Start
WWWebmart
The WWWebMart W...
Paint Shop Pro
textstyle - Micros...
5:30 PM

Fig. 46

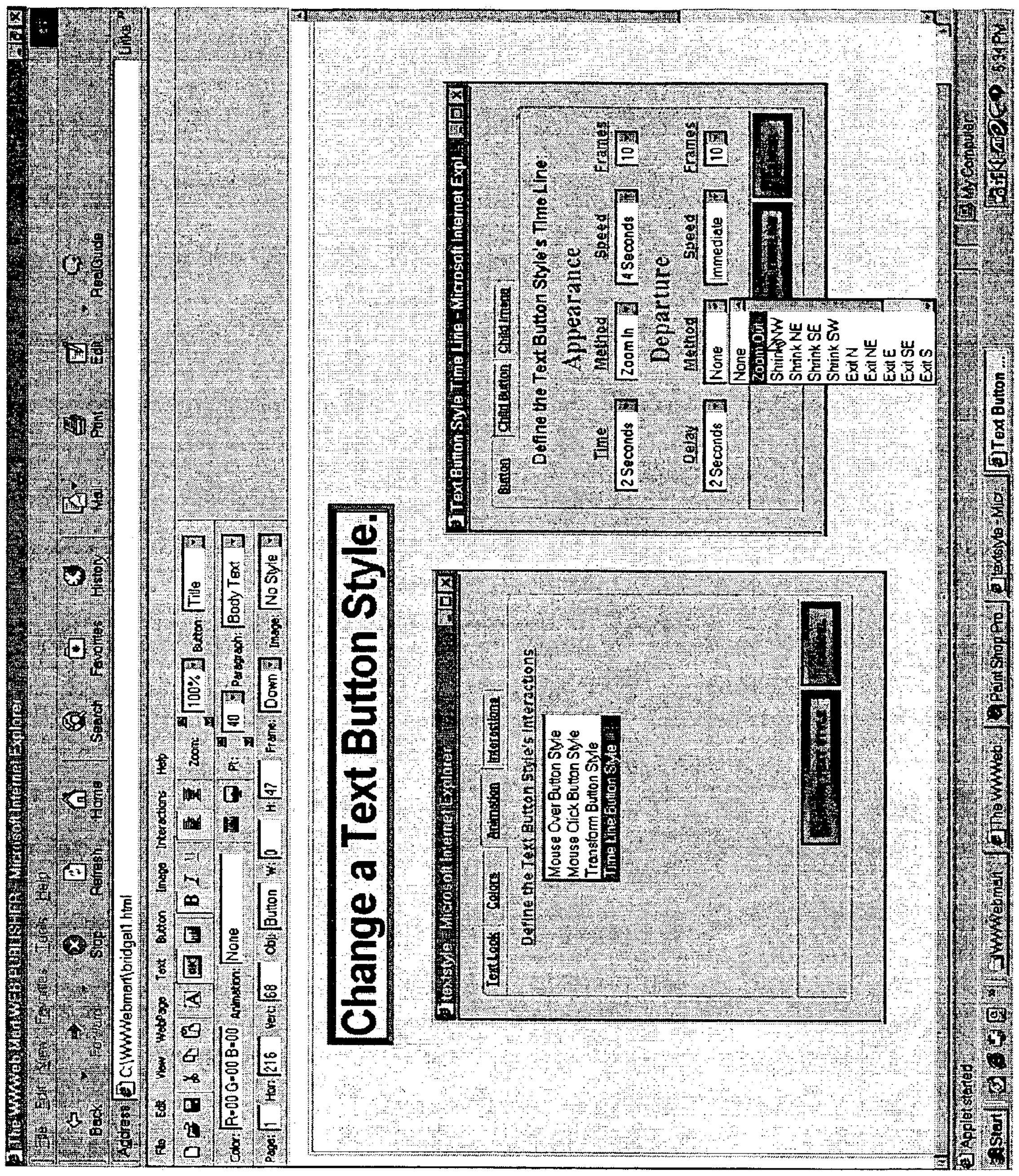
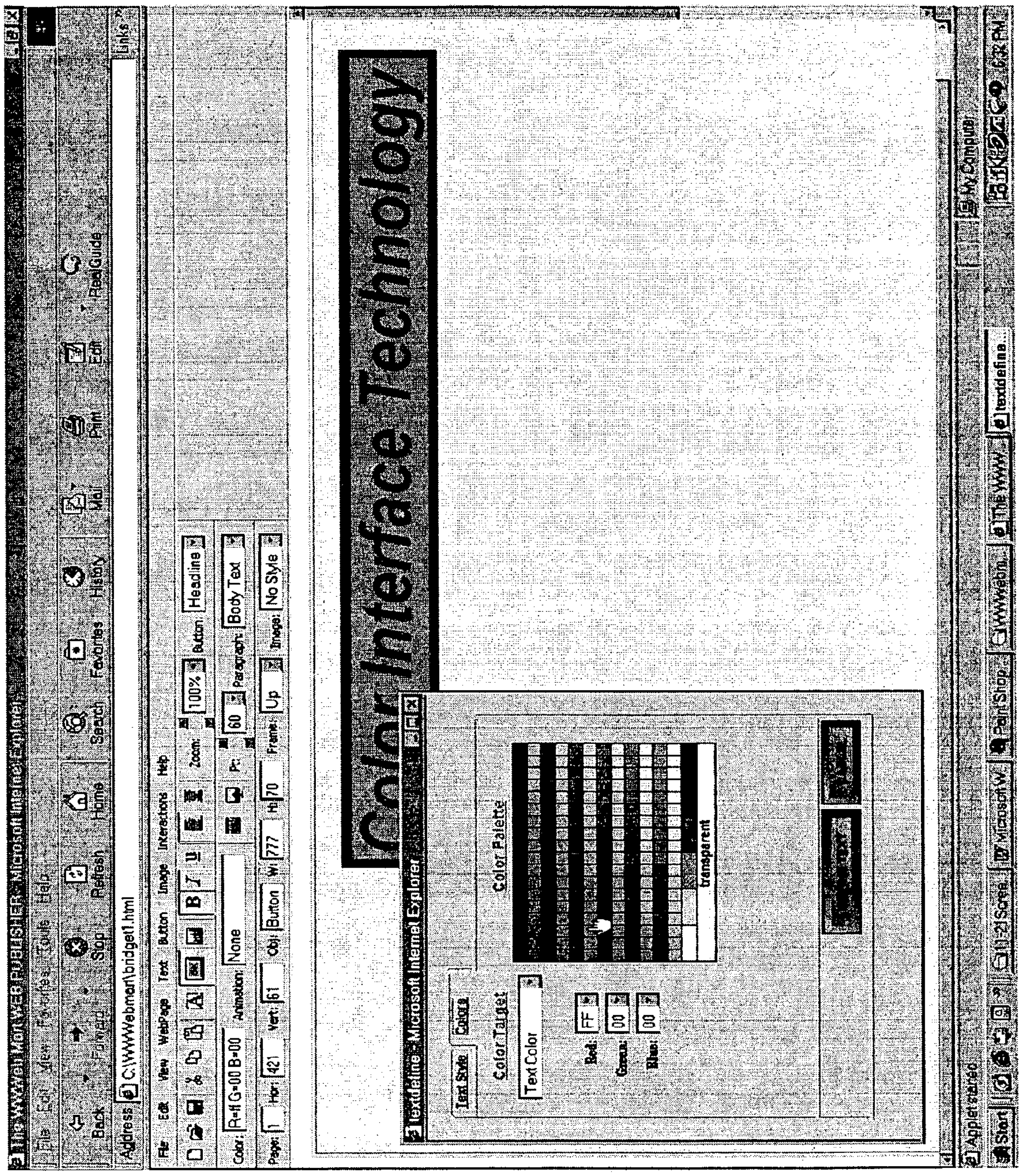

The WWWebMart WEB PUBLISHER - Microsoft Internet Explorer
Back
Forward
Stop
Refresh
Home
Search
Favorites
History
Mail
Print
Edit
RealGuide
Address C:\WWWebmart\bridge1.html
Links
File Edit View WebPage Text Button Image Interactions Help
Zoom: 100%
Button: Title
Color: R=00 G=00 B=00
Animation: None
Pt: 40
Paragraph: Body Text
Page: 1
Horz: 216
Vert: 68
Obj: Button
W: 0
H: 47
Frame: Down
Image: No Style
Change a Text Button Style.
textstyle - Microsoft Internet Explorer
Text Look
Colors
Animation
Interactions
Define the Text Button Style's Interactions
Mouse Over Button Style
Mouse Click Button Style
Transform Button Style
Time Line Button Style
Text Button Style Time Line - Microsoft Internet Expl...
Button
Child Button
Child Image
Define the Text Button Style's Time Line:
Appearance
Time
Method
Speed
Frames
2 Seconds
Zoom In
4 Seconds
10
Departure
Delay
Method
Speed
Frames
2 Seconds
None
Immediate
10
None
Zoom Out
Shrink NW
Shrink NE
Shrink SE
Shrink SW
Exit N
Exit NE
Exit E
Exit SE
Exit S
Applet started
My Computer
Start
WWWebmart
The WWWeb...
PaintShop Pro
textstyle - Micr...
Text Button ...
5:31 PM

Fig. 47


The WWWebMart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Links
File Edit View WebPage Text Button Image Interactions Help
Color Interface Technology
textdefine - Microsoft Internet Explorer
Text Style
Colors
Color Palette
transparent
Applet started
Start
The WWW...
textdefine...
My Computer
6:32 PM

Address C:\WWWebmart\bridget1.html

Zoom: 100% Button: Headline

Color: R=ff G=00 B=00 Animation: None Pt: 60 Paragraph: Body Text

Page: 1 Hor: 421 Vert: 61 Obj: Button W: 777 H: 70 Frame: Up Image: No Style

Color Target: Text Color

Red: FF

Green: 00

Blue: 00

Fig. 48

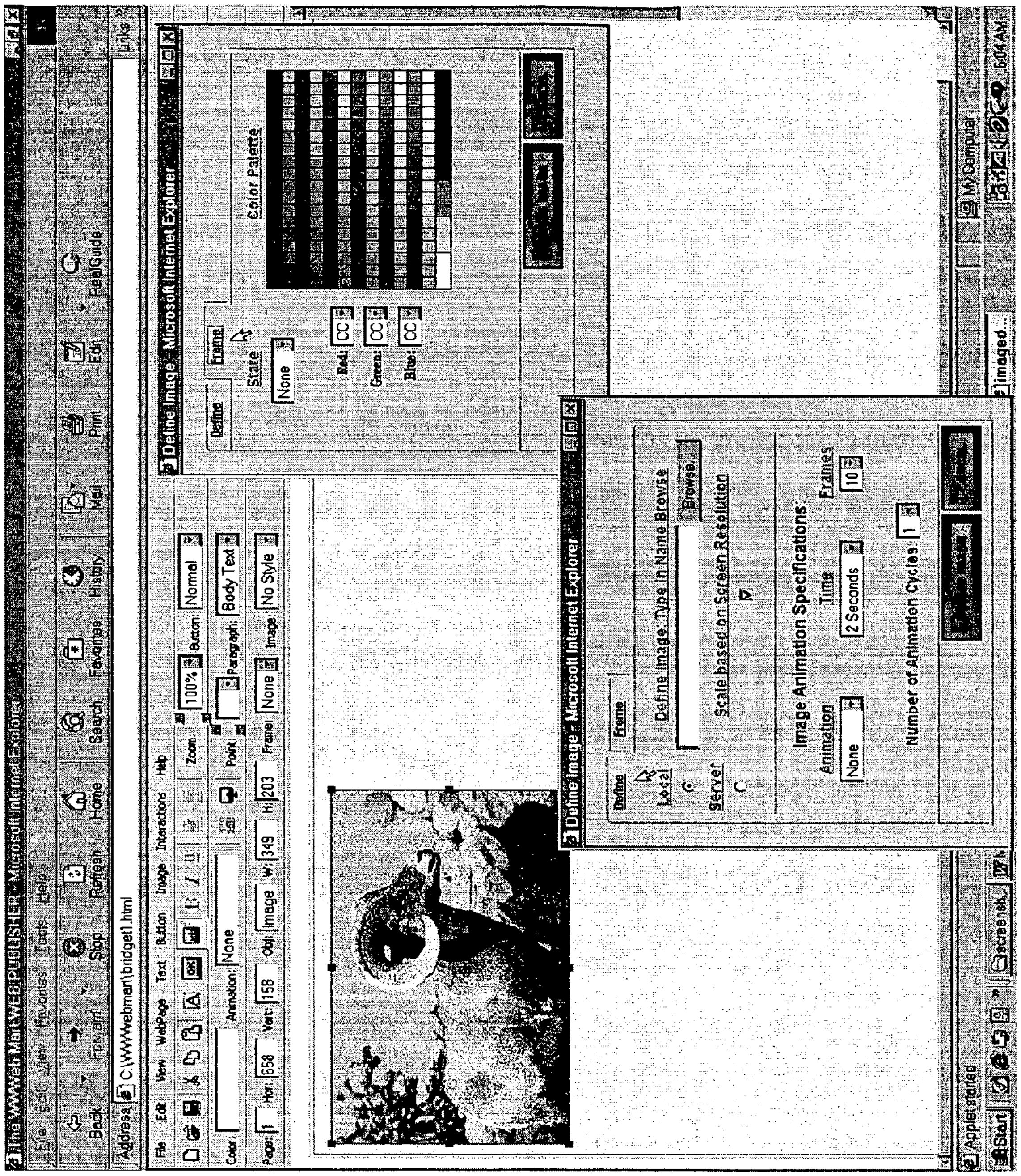
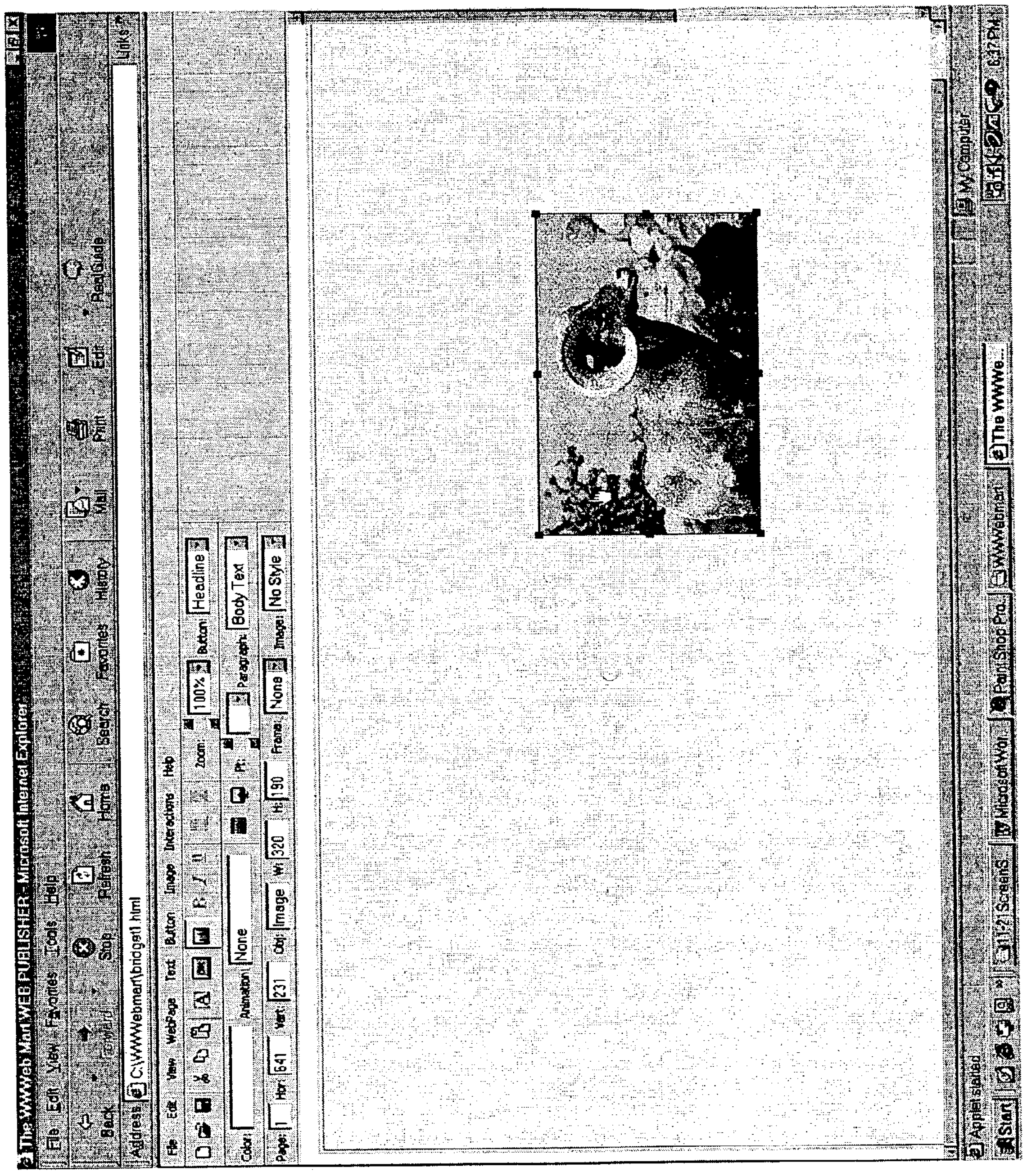

The WWWebMart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Address C:\WWWebmart\bridget1.html
Links
File Edit View WebPage Text Button Image Interaction Help
Zoom: 100% Button: Normal
Color: Animation: None Point: Paragraph: Body Text
Page: 1 Hor: 658 Vert: 158 Obj: Image W: 349 H: 203 Frame: None Image: No Style
Define Image - Microsoft Internet Explorer
Define Frame
State None
Color Palette
Red: CC
Green: CC
Blue: CC
Define Image - Microsoft Internet Explorer
Define Frame
Local
Server
Define Image, Type in Name Browse
Browse...
Scale based on Screen Resolution
Image Animation Specifications:
Animation None
Time 2 Seconds
Frames 10
Number of Animation Cycles: 1
Applet started
imaged...
My Computer
Start
screensh...
6:04 AM

Fig. 49


The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Links
File Edit View WebPage Text Button Image Interactions Help
Zoom: 100%
Button: Headline
Color:
Animation: None
Paragraph: Body Text
Page: 1
Frame: None
Image: No Style
Start
The WWWe...
Paint Shop Pro...
WWWebmart
Microsoft Wor...
My Computer
6:37 PM

Address: C:\WWWebmart\bridget1.html

Hor: 641 Vert: 231 Obj: Image W: 320 Ht: 190

Fig. 50

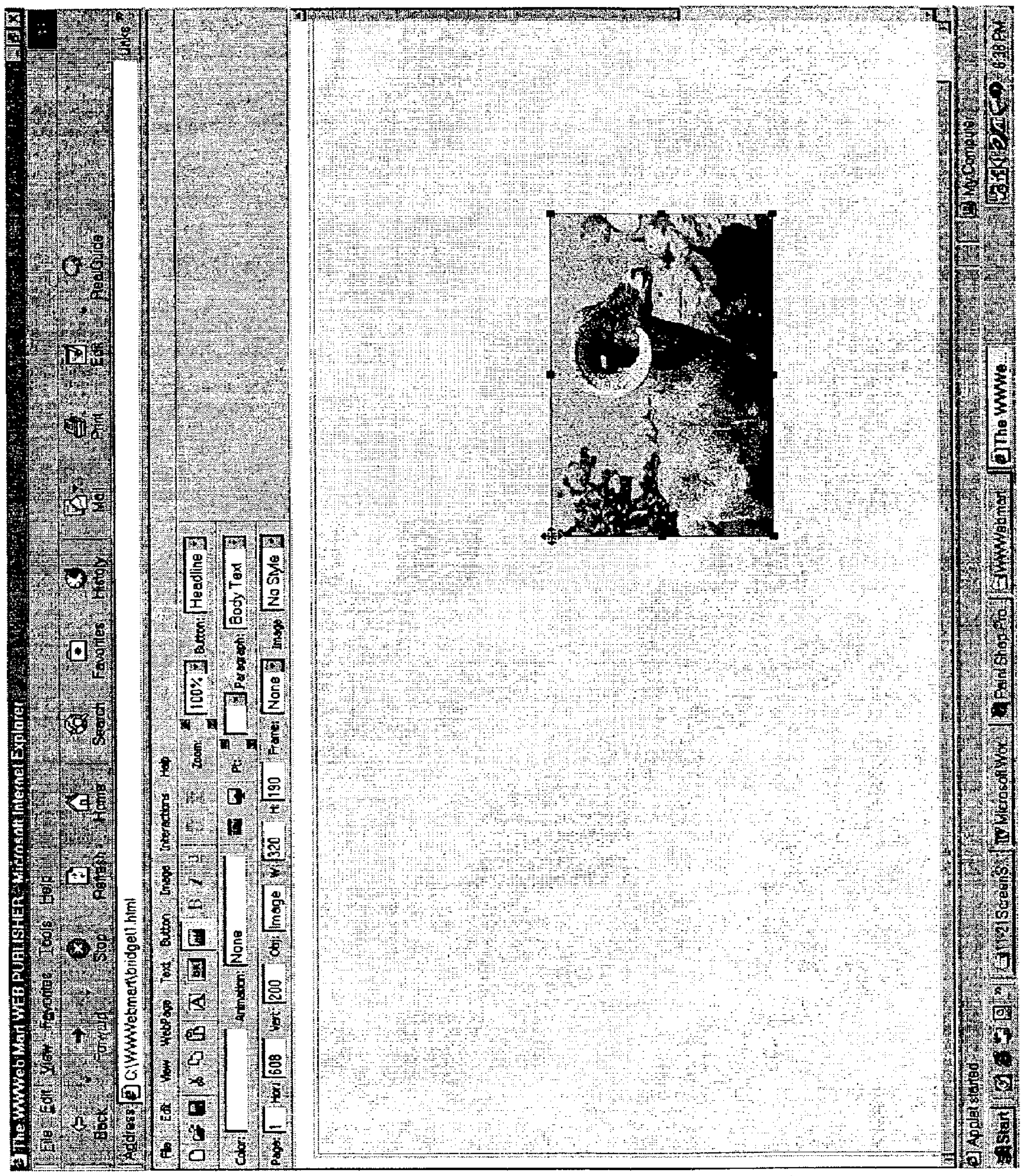

The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Address C:\WWWebmart\bridge1.html
File Edit View WebPage Text Button Image Interactions Help
Zoom: 100% Button: Headline
Color: Animation: None Pt: Paragraph: Body Text
Page: 1 Horz: 608 Vert: 200 Obj: Image W: 320 H: 190 Frame: None Image: No Style
Applet started
My Computer
Start 11121ScreenS... Microsoft Wor... Paint Shop Pro... WWWebmart The WWWe...
6:38 PM

Fig. 51

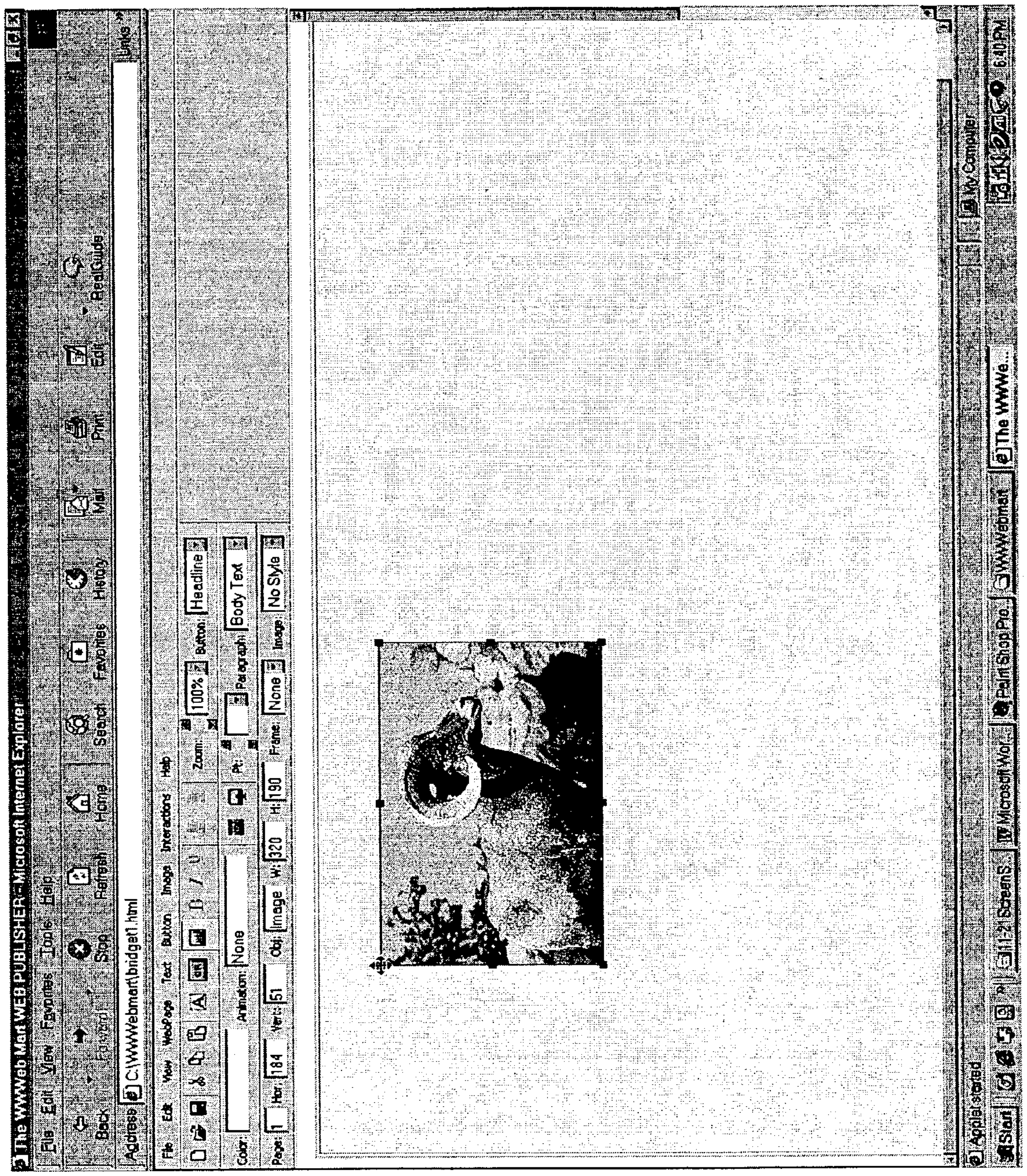
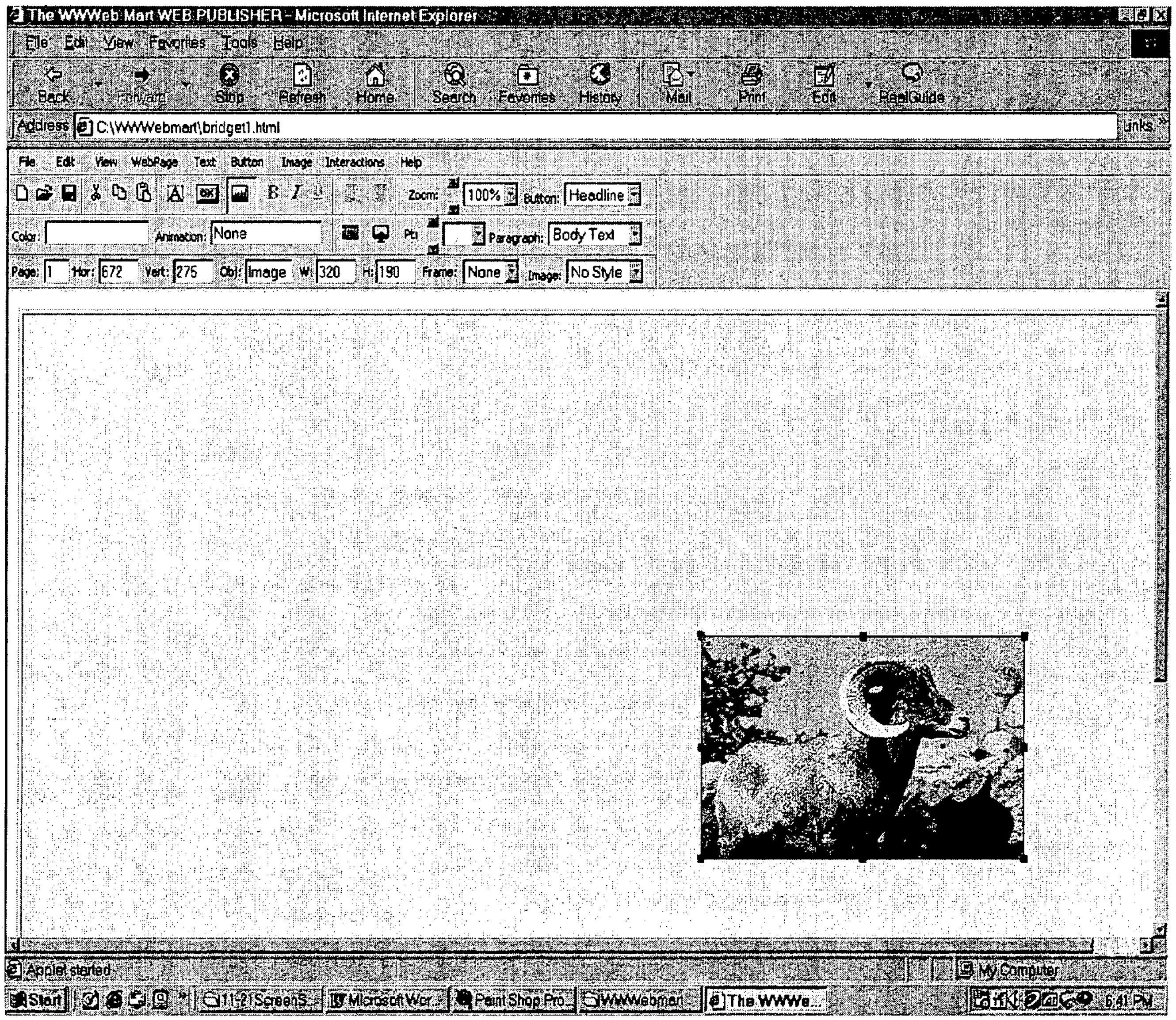

The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Address C:\WWWebmart\bridget1.html
Links
File Edit View WebPage Text Button Image Interactions Help
Zoom: 100%
Button: Headline
Color:
Animation: None
Pt:
Paragraph: Body Text
Page: 1 Hor: 184 Vert: 51 Obj: Image W: 320 H: 190 Frame: None Image: No Style
Applet started
My Computer
Start
11:2 ScreenS...
Microsoft Wor...
Paint Shop Pro
WWWebmart
The WWWe...
6:40 PM

Fig. 52


The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Links
File Edit View WebPage Text Button Image Interactions Help
Applet started
My Computer
Start 11-21ScreenS... Microsoft Wor... Paint Shop Pro... WWWebmart The WWWe...
6:41 PM

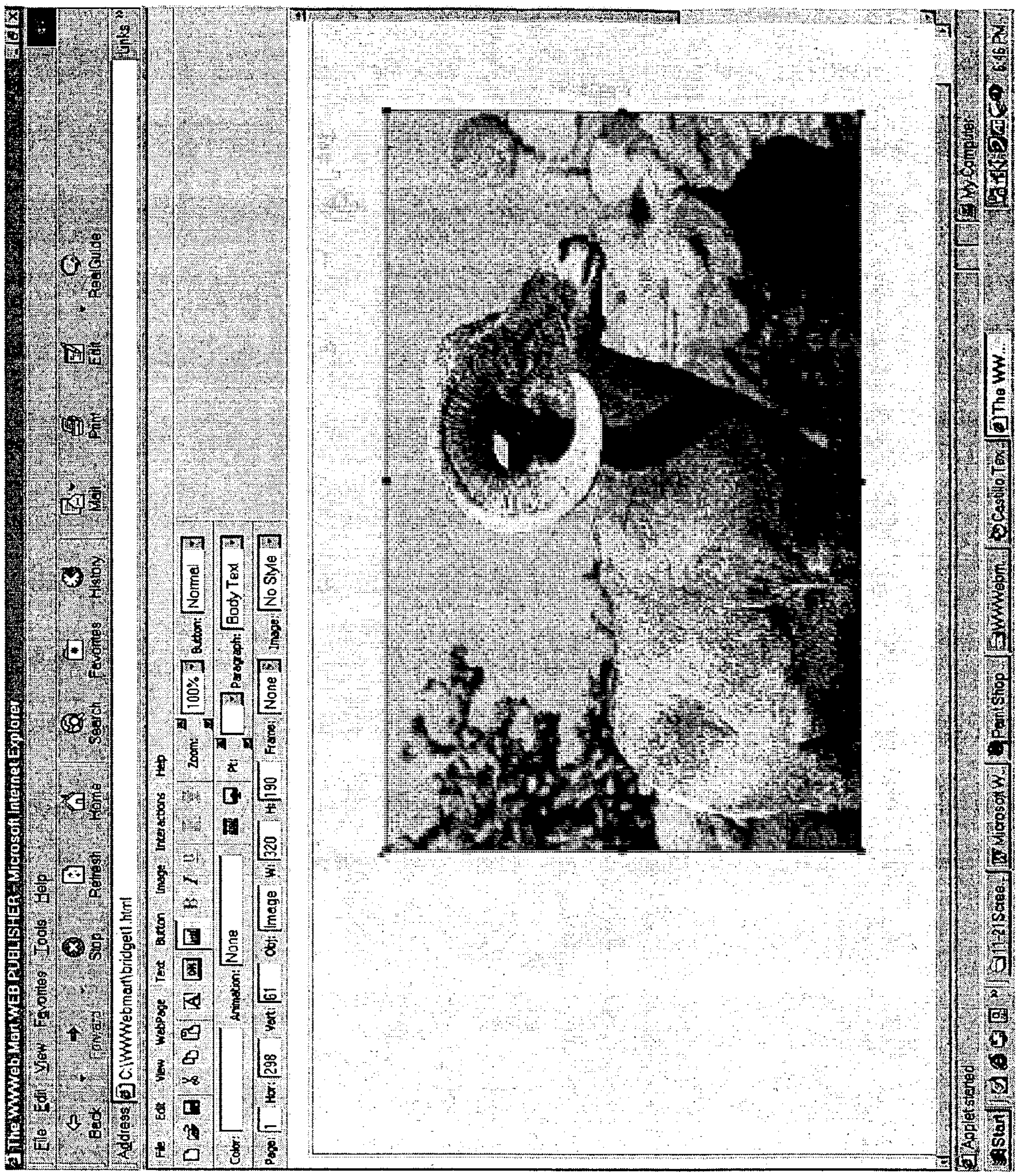
Address: C:\WWWebmart\bridget1.html

Zoom: 100% Button: Headline

Color: Animation: None Pt: Paragraph: Body Text

Page: 1 Hor: 672 Vert: 275 Obj: Image W: 320 H: 190 Frame: None Image: No Style

Fig. 53


The WWWebMart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Links
File Edit View WebPage Text Button Image Interactions Help
Start Applet started My Computer 6:45 PM
Microsoft W... Paint Shop WWWebm... Castillo Tex... The WW...

Address C:\WWWebmart\bridget1.html

Zoom: 100% Button: Normal
Color: Animation: None Pt: Paragraph: Body Text
Page: 1 Hor: 298 Vert: 61 Obj: Image W: 320 H: 190 Frame: None Image: No Style

Fig. 54

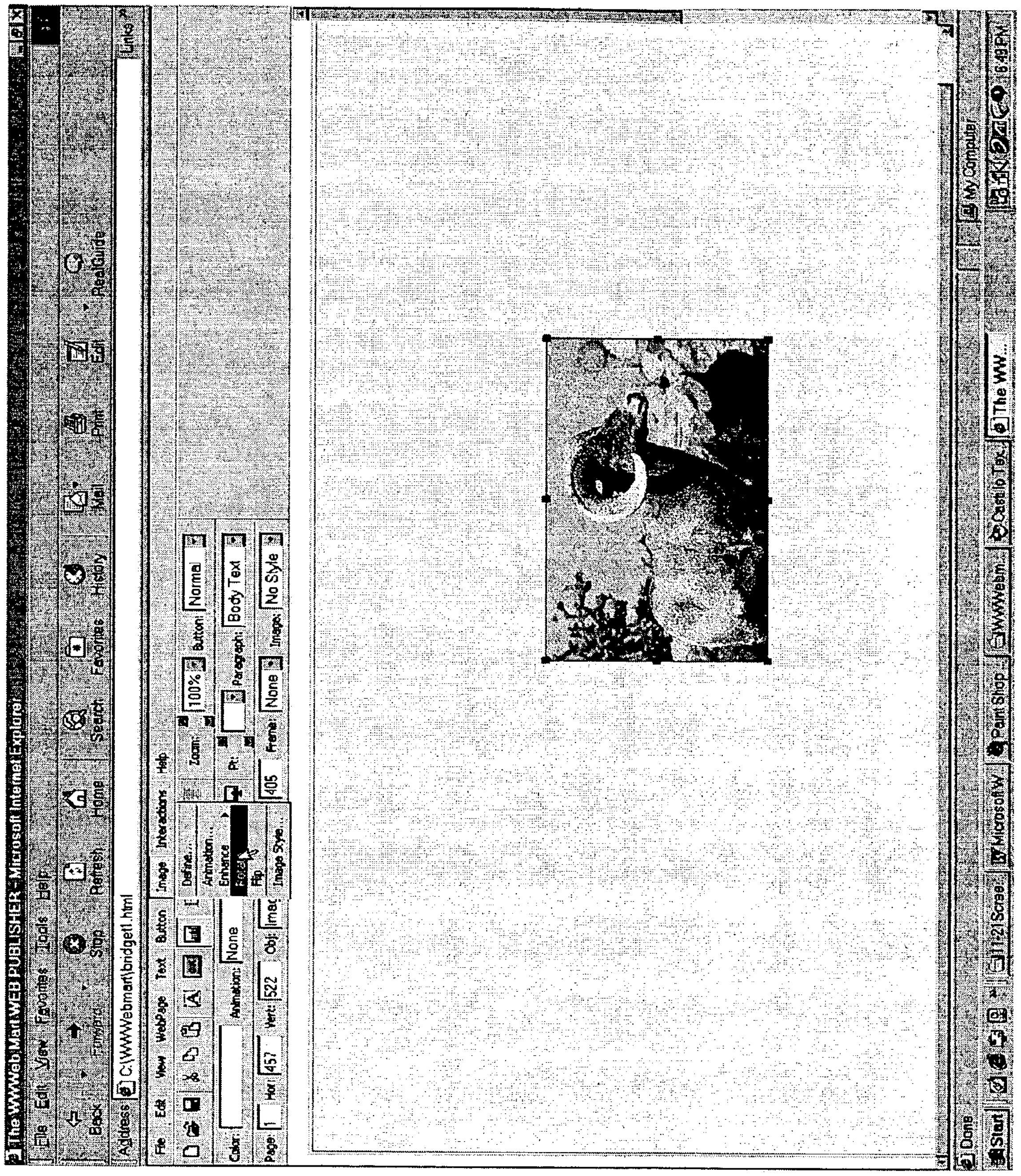

The WWWeb.Mart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Address C:\WWWebmart\bridget1.html
Links
File Edit View WebPage Text Button Image Interactions Help
Define...
Animation...
Enhance
Rotate
Flip
Image Style...
Zoom: 100%
Button: Normal
Color:
Animation: None
Pt:
Paragraph: Body Text
Page: 1
Hor: 457
Vert: 522
Obj: Imag
405
Frame: None
Image: No Style
Done
My Computer
Start
1121Scree...
Microsoft W...
Paint Shop ...
WWWebm...
Castillo Tex...
The WW...
6:49 PM

Fig. 55

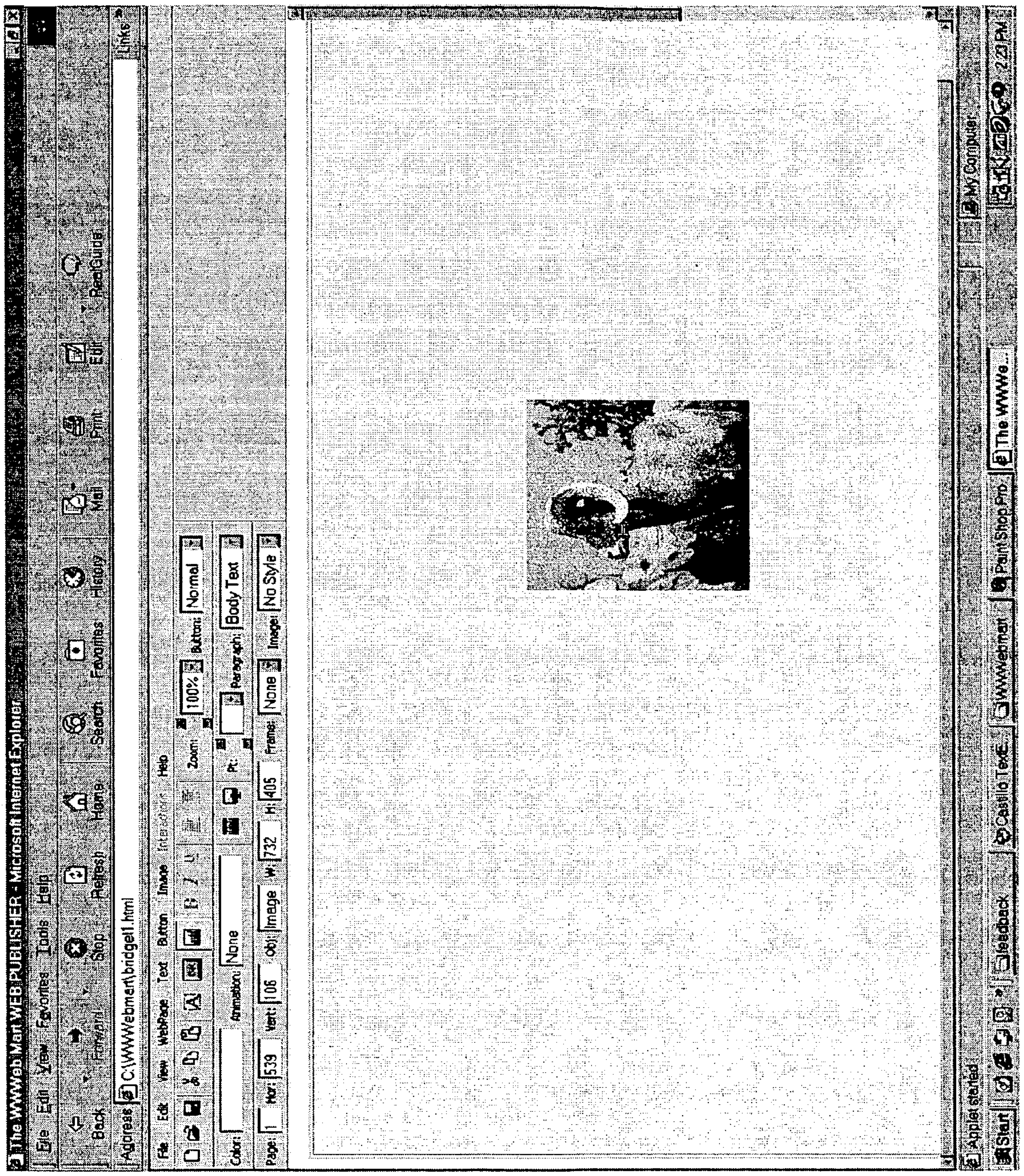

The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Address C:\WWWebmart\bridge1.html
Links
File Edit View WebPage Text Button Image Interactive Help
Zoom: 100% Button: Normal
Color: Animation: None Pt: Paragraph: Body Text
Page: 1 Hor: 539 Vert: 106 Obj: Image W: 732 H: 405 Frame: None Image: No Style
Applet started
My Computer
Start Feedback Castillo TextE... WWWebmart Paint Shop Pro The WWWe...
2:23 PM

Fig. 56

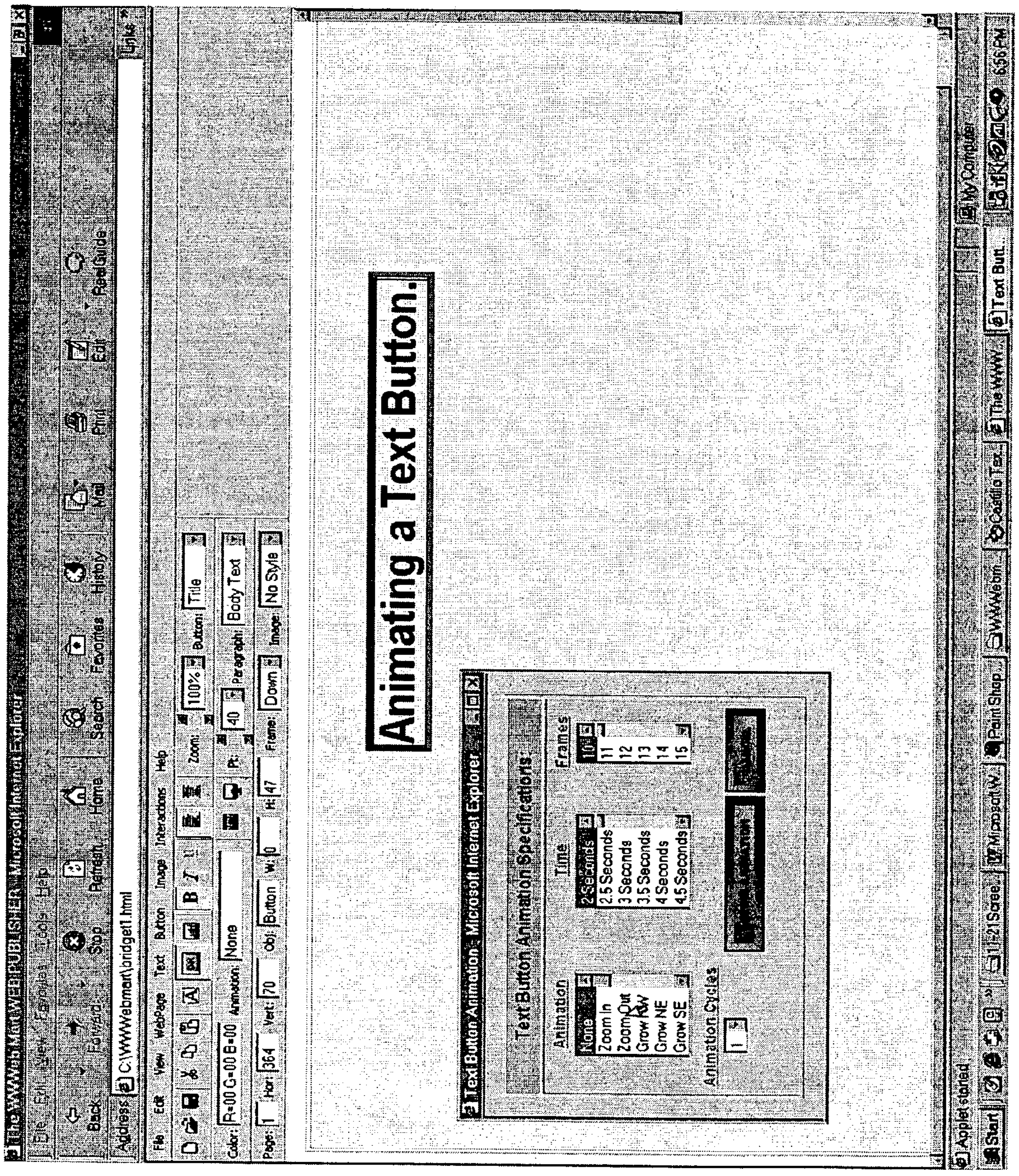
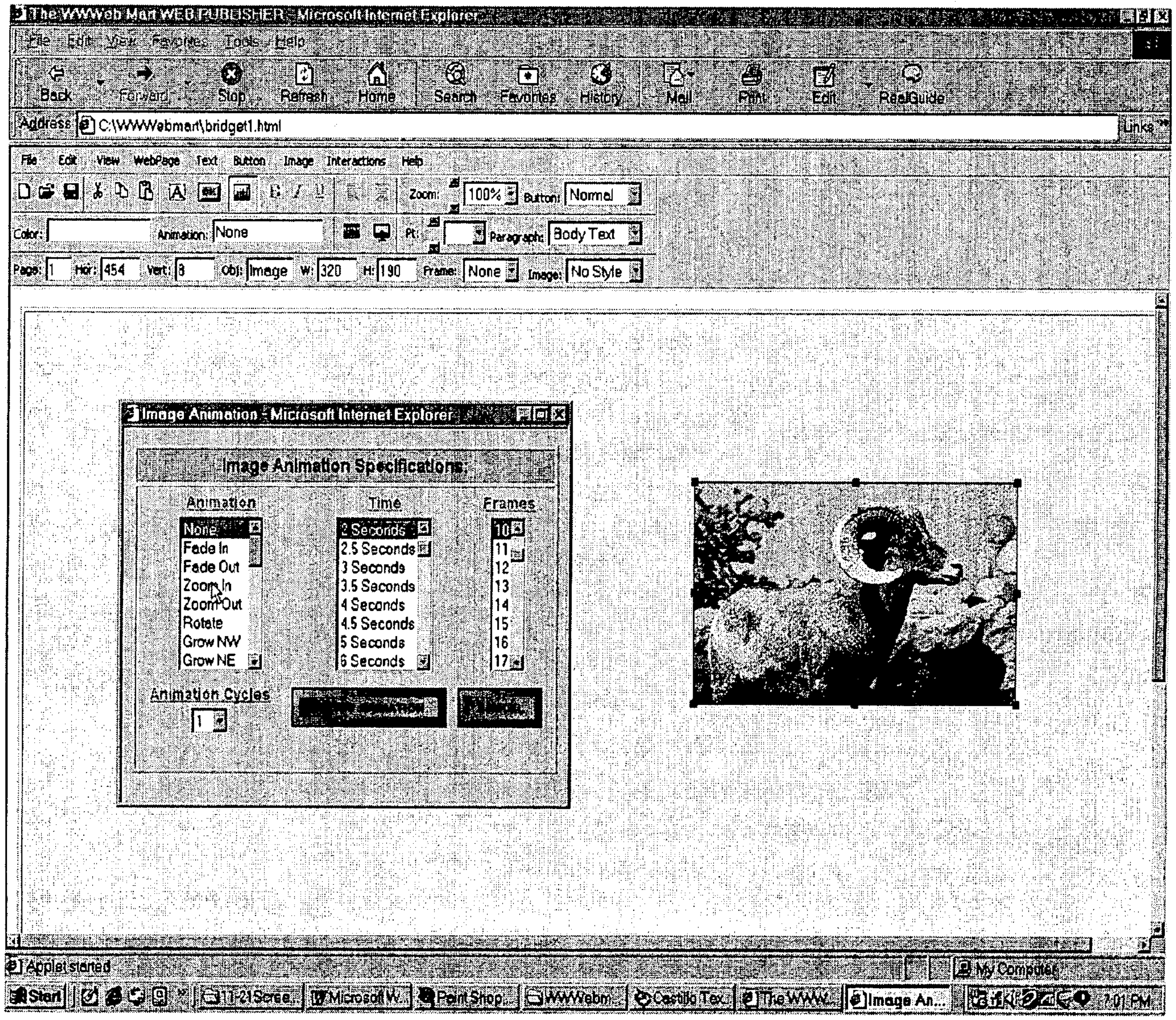

The WWWeb Mart WEBPUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Address C:\WWWebmart\bridget1.html
Links
File Edit View WebPage Text Button Image Interactions Help
Zoom: 100% Button: Title
Color: R=00 G=00 B=00 Animation: None Pt: 40 Paragraph: Body Text
Page: 1 Hor: 364 Vert: 70 Obj: Button W: 0 H: 47 Frame: Down Image: No Style
Animating a Text Button.
Text Button Animation - Microsoft Internet Explorer
Text Button Animation Specifications:
Animation
None
Zoom In
Zoom Out
Grow NW
Grow NE
Grow SE
Time
2 Seconds
2.5 Seconds
3 Seconds
3.5 Seconds
4 Seconds
4.5 Seconds
Frames
10
11
12
13
14
15
Animation Cycles
1
Applet started
Start
11-21 Scre...
Microsoft W...
Paint Shop...
WWWebm...
Cardio Tex...
The WWW...
Text Butt...
My Computer
6:56 PM

Fig. 57


The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Address
Links
File Edit View WebPage Text Button Image Interactions Help
Start
11-21Scree... Microsoft W... Paint Shop... WWWebm... Castillo Tex... The WWW... Image An...
My Computer
7:01 PM

Address: C:\WWWebmart\bridget1.html

Zoom: 100% Button: Normal

Color: Animation: None Pt: Paragraph: Body Text

Page: 1 Hor: 454 Vert: 8 Obj: Image W: 320 H: 190 Frame: None Image: No Style

Image Animation - Microsoft Internet Explorer

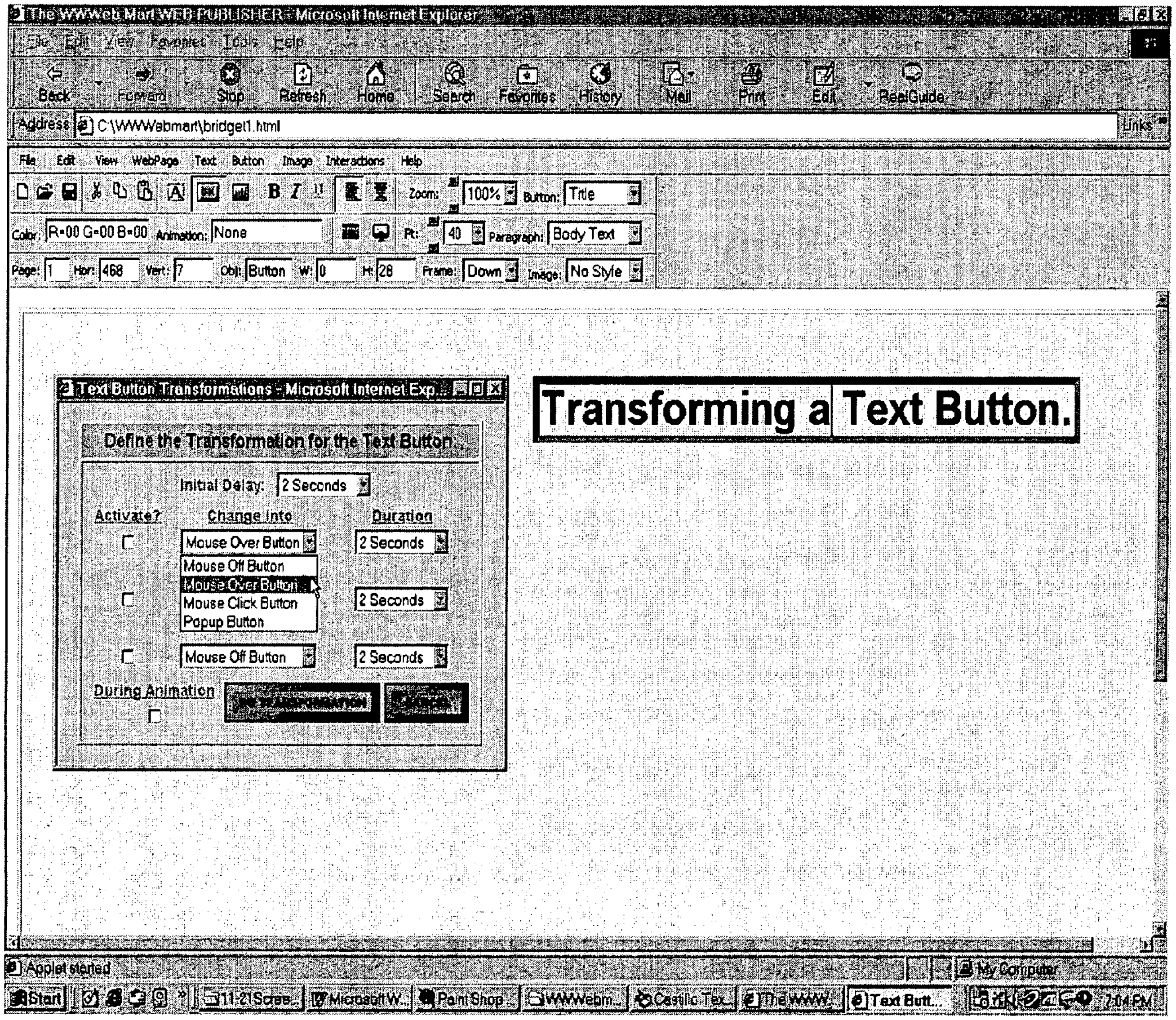
Image Animation Specifications

Animation	Time	Frames
None	2 Seconds	10
Fade In	2.5 Seconds	11
Fade Out	3 Seconds	12
Zoom In	3.5 Seconds	13
Zoom Out	4 Seconds	14
Rotate	4.5 Seconds	15
Grow NW	5 Seconds	16
Grow NE	6 Seconds	17

Animation Cycles: 1

Applet started

Fig. 58


The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Links
File Edit View WebPage Text Button Image Interactions Help
Text Button Transformations - Microsoft Internet Exp...
Applet started
Start 11-21Scree... Microsoft W... Paint Shop... WWWebm... Castilo Tex... The WWW... Text Butt...
My Computer
7:04 PM

Address C:\WWWebmart\bridget1.html

Zoom: 100% Button: Title

Color: R=00 G=00 B=00 Animation: None Pt: 40 Paragraph: Body Text

Page: 1 Hor: 468 Vert: 7 Obj: Button W: 0 H: 28 Frame: Down Image: No Style

Transforming a Text Button.

Define the Transformation for the Text Button...

Initial Delay: 2 Seconds

Activate?	Change Into	Duration
☐	Mouse Over Button	2 Seconds
☐	Mouse Off Button / Mouse Over Button / Mouse Click Button / Popup Button	2 Seconds
☐	Mouse Off Button	2 Seconds

During Animation ☐

Fig. 59

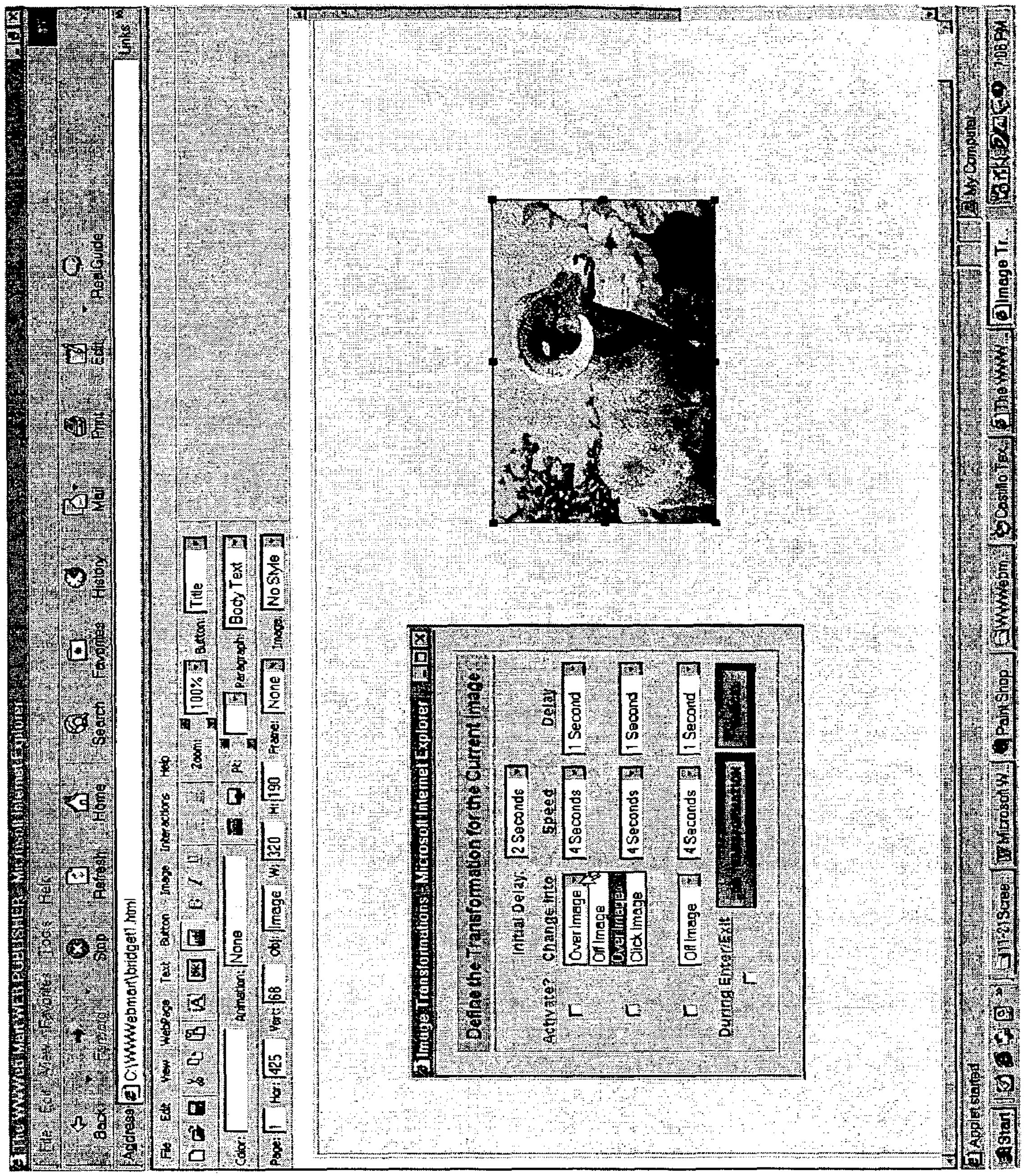

The WWWeb Mart WEB PUBLISHER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit RealGuide
Address C:\WWWebmart\bridget1.html
Links
File Edit View WebPage Text Button Image Interactions Help
Zoom: 100% Button: Title
Color: Animation: None Pt: Paragraph: Body Text
Page: 1 Hor: 425 Vert: 68 Obj: Image W: 320 H: 190 Frame: None Image: No Style
Applet started
My Computer
Start WWWebm... CosmoTex... The WWW... Image Tr... Microsoft W... Paint Shop... 7:05 PM

Image Transformations - Microsoft Internet Explorer

Define the Transformation for the Current Image

Initial Delay: 2 Seconds

Activate?	Change Into	Speed	Delay
☐	Over Image (Off Image, Over Image, Click Image)	4 Seconds	1 Second
☐		4 Seconds	1 Second
☐	Off Image	4 Seconds	1 Second

During Enter/Exit ☐

Fig. 60

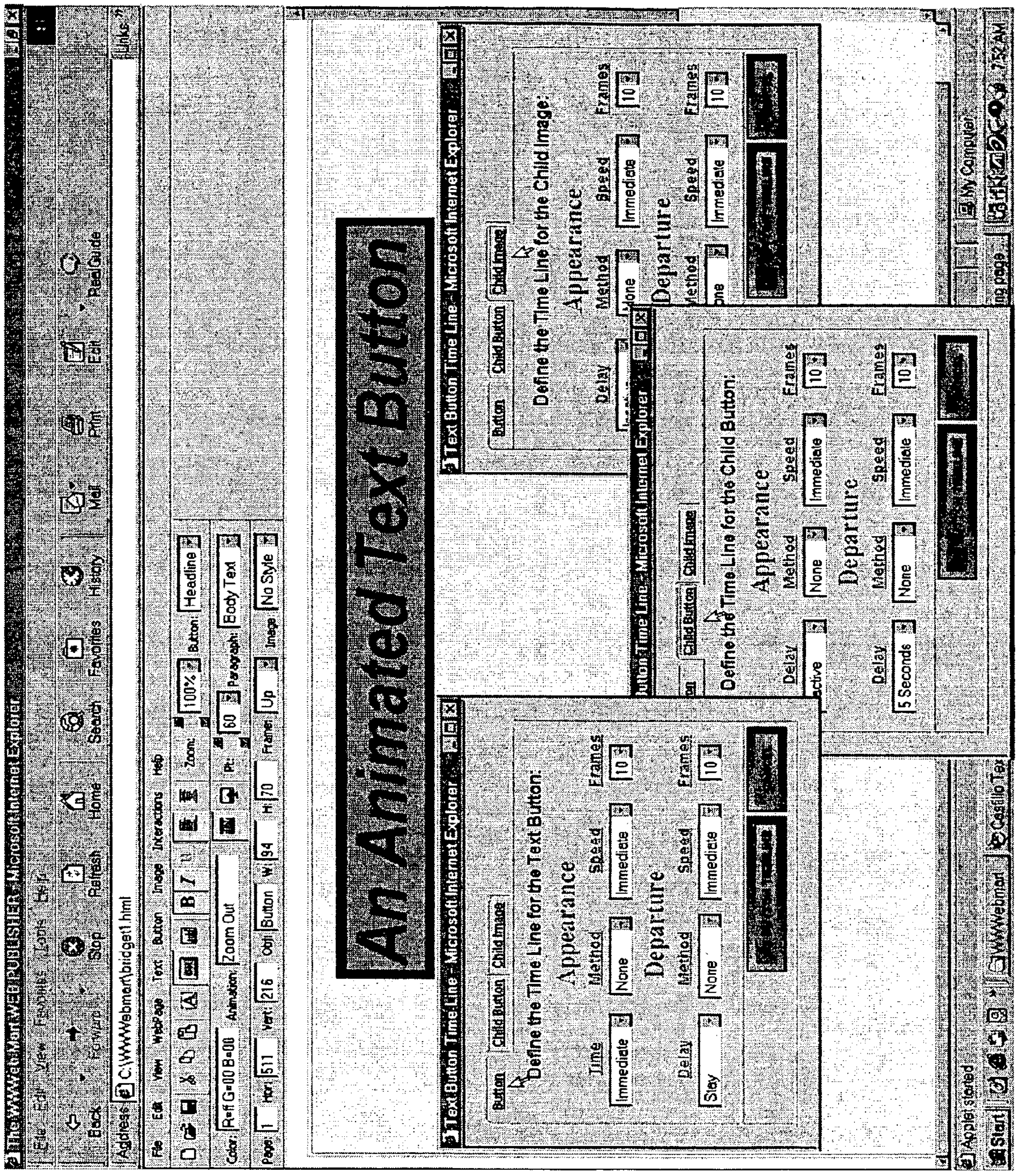

The WWWebMart WEBPUBLISHER - Microsoft Internet Explorer
Address C:\WWWebmart\bridget1.html
An Animated Text Button
Text Button Time Line - Microsoft Internet Explorer
Button
Child Button
Child Image
Define the Time Line for the Text Button:
Appearance
Time
Method
Speed
Frames
Immediate
None
10
Departure
Delay
Stay
Define the Time Line for the Child Button:
5 Seconds
Define the Time Line for the Child Image:

Fig. 61

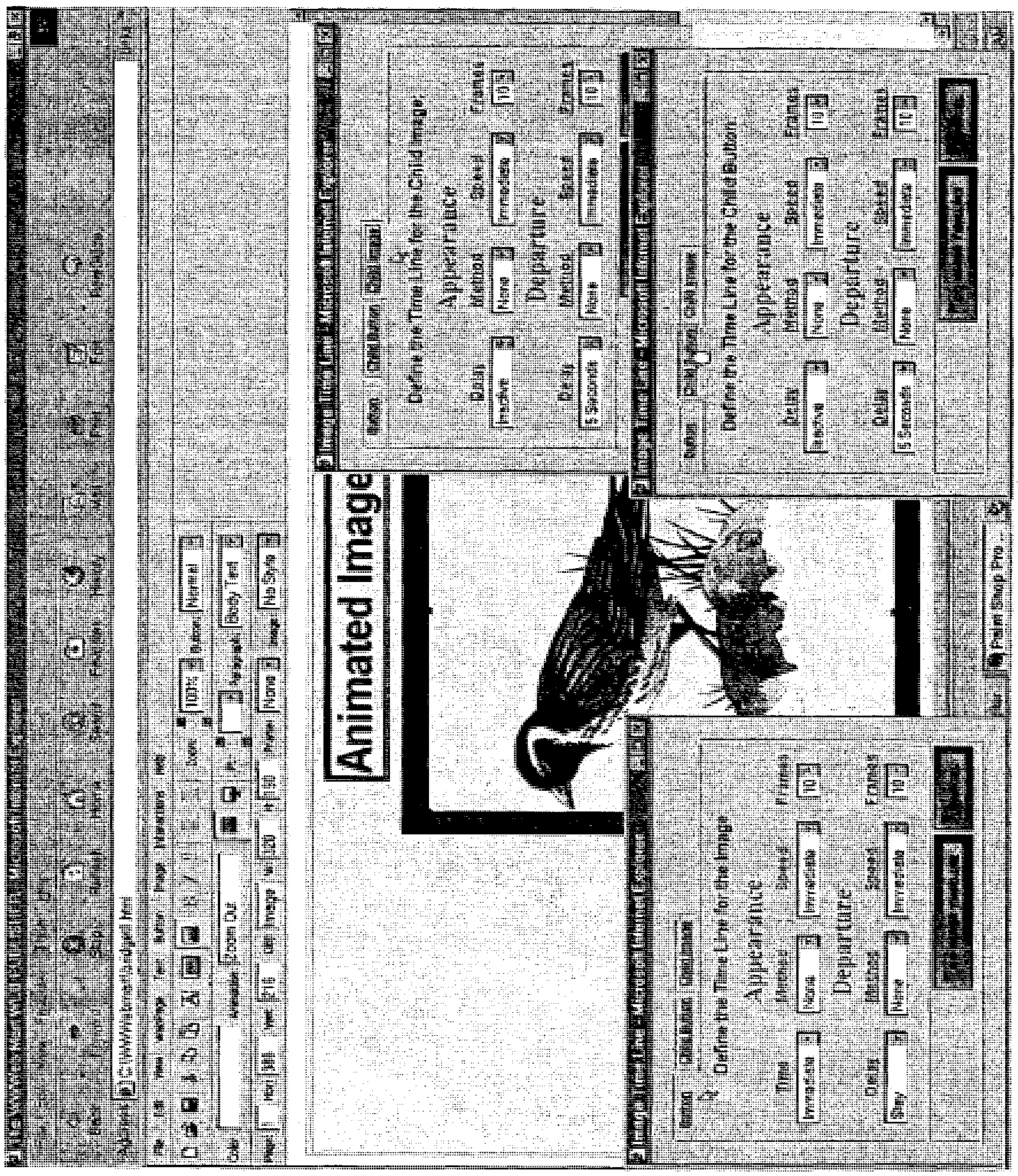

Animated Image
Define the Time Line for the Child Image
Define the Time Line for the Child Button
Define the Time Line for the Image
Button
Child Button
Child Image
Appearance
Departure
Delay
Method
Speed
Frames
Time
Inactive
Immediate
None
5 Seconds
Stay
10
Paint Shop Pro

Fig. 62

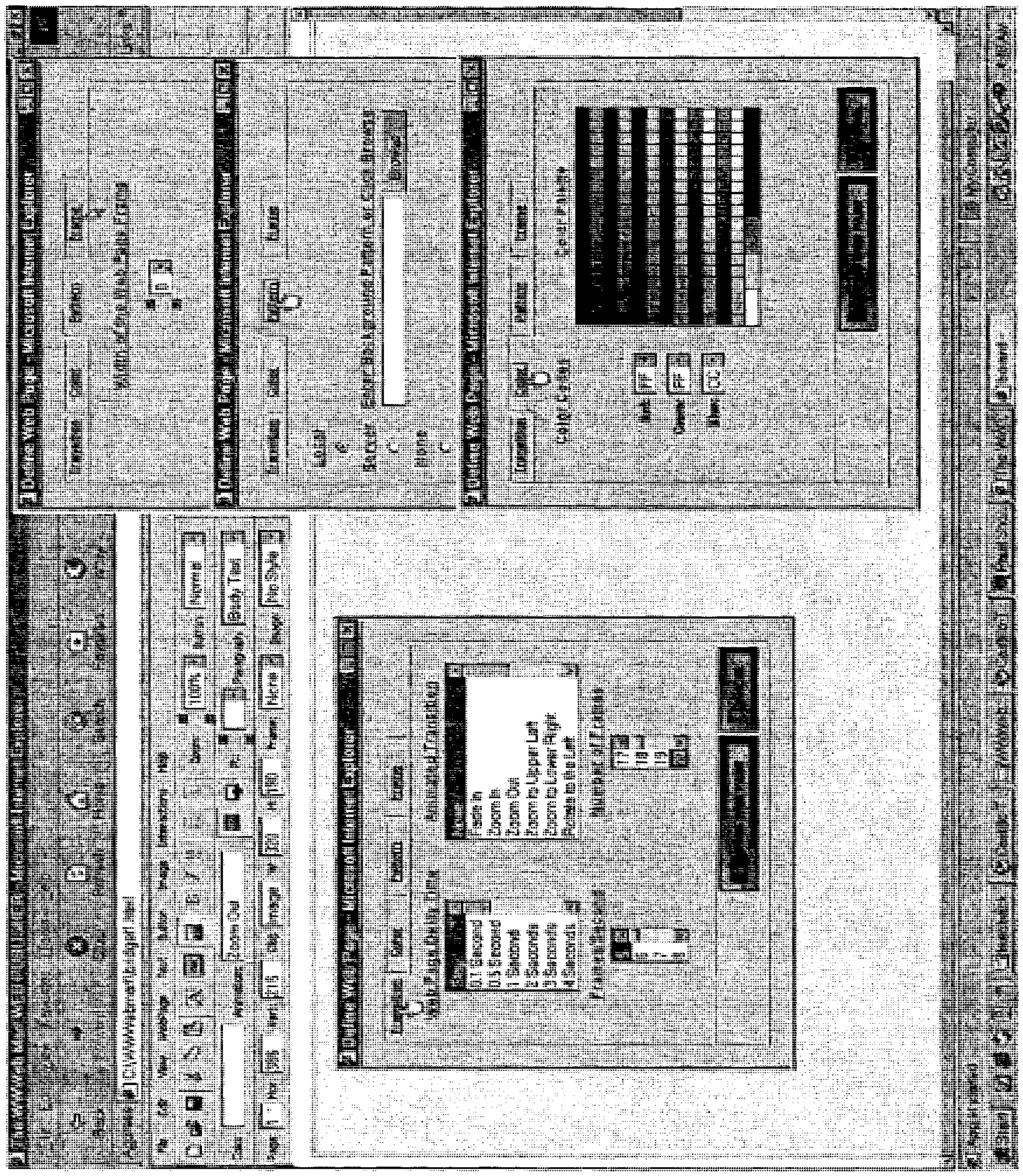


Fig. 63

BROWSER BASED WEB SITE GENERATION TOOL AND RUN TIME ENGINE

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a continuation of U.S. patent application Ser. No. 09/454,061, filed Dec. 2, 1999 now U.S. Pat. No. 6,546,397.

FIELD OF THE INVENTION

The present application is directed to computing systems, and more particularly to methods and apparatus for building a web site using a browser-based build engine.

BACKGROUND

Conventional web site construction tools operate on traditional operating system platforms and generate as output HTML (hyper text mark-up language) and Script Code (e.g., JavaScript). A browser draws a web page associated with the web site by interpreting the HTML and JavaScript Code. However, conventional mark-up and scripting languages include numerous inherent limitations. Conventional markup and scripting languages have not been designed for serious multimedia applications. They have almost no file handling ability and very little computational power. In addition, they are remarkably slow and inefficient.

As such it is virtually impossible to write a web publishing application in HTML and JavaScript. All conventional implementations must, and do, utilize a full-featured programming language, such as C++ or Visual Basic. Since the current popular browsers do not support these languages, by necessity, conventional web publishing applications run on platforms other than the World Wide Web (WWW) and its browsers. Therefore, at best, a conventional web publishing application can offer only a crude preview capability of what a real web page will look like.

Although C++ and Visual Basic are very capable languages, the conventional web publishing applications written in these languages are still necessarily limited by the limitations inherent in their form of output, which as described above is typically HTML and scripting code. As such, a conventional web publishing application written in one of these languages suffers from the severe performance problems inherent in these languages.

For example, HTML and JavaScript are incapable of reformatting text and scaling buttons or images dynamically. In addition, most conventional web publishing applications design a web page layout to fit into a 640 pixel wide screen. This means that the ability for higher resolution screens to display more data horizontally is lost. Since capability is wasted on the horizontal plane, unnecessary vertical scrolling may be required. Further, on higher output resolution devices (screens), unsightly extra white space or background may be prevalent.

SUMMARY

In one aspect the invention includes a Browser Based build engine that is written entirely in a web based full featured programming language (e.g., JAVA). A Browser Based Interface (the Interface") between the web designer and the build engine is included. The browser-based interface can be written in the World Wide Web's (WWW) Hypertext Markup Language (HTML) and its Extensions (Dynamic HTML,

JavaScript and Cascading Style Sheets). The Interface includes a unique set of communication techniques. One technique allows for effective two-way communications between a JAVA engine and JavaScript. Another technique allows for communications between a JAVA applet object inside a JavaScript window, with the JAVA engine, which permits the implementation of advanced intelligent interface objects, such as a "slider" or a "dial".

In one aspect the invention includes a screen resolution sensing mechanism that causes a build engine (i.e. build tools) to adopt its interface to the web designer's screen resolution.

In one aspect, the invention includes a multi-dimensional array structured database, that, in addition to storing the numeric and string data found in conventional databases, also stores multi-dimensional arrays of various multimedia objects. They include colors, fonts, images, audio clips, video clips, text areas, URLs and thread objects. The invention includes a run time generation procedure that creates a compressed web site specific customized run time engine program file, with its associated database and a build engine generated HTML shell file.

The invention can include web page scaling technology, so that when the web site/web page is accessed on the WWW, the web pages and all the objects within them can be scaled to the user's screen resolution and to the then current browser window size.

In one aspect, the invention includes a proprietary multilevel program animation model (threads) that responds to multiple user interactions and time sensitive operations simultaneously.

In one aspect, the invention includes a mechanism for the dynamic resizing of the build engine's web page size during various editing operations.

In one aspect, the invention includes techniques for creating browser based interface objects that visually and behaviorally are identical to those of the MS Window's standard.

Aspects of the invention can include one or more of the following features. A browser based build engine is provided that includes a browser based interface. The entire web site build process is WYSIWYG (what you see is what you get), with the web designer working directly on and with the final web page. The data produced by the build engine is processed and ultimately placed into a multi-dimensional array structured database, and stored in an external file. A run time generation procedure creates a compressed program customized run time engine file, with an associated database and a build engine generated HTML Shell File.

When the web site/web page is accessed on the WWW, web page scaling technology can be accessed to generate web pages that are scaled to the user's screen resolution. A technique is provided so that an applet's size (height and width) can be set in real time under the control of either the interface or the build engine. At the same time a multi-level program animation model (threads) is activated for user interactions and time sensitive operations.

The browser based interface technologies create a set of interface objects with a look and feel that is identical to that of MS Windows, yet includes technologies that equal or occasionally surpass those of high end word processors, desk top publishers, and image processing software programs, particularly in the areas of interaction, animation, and timeline technologies. The run time engine includes multimedia capabilities often rivaling the digital processing capabilities seen on television and in the movies.

Because of the implementation of a variety of performance and file reduction techniques, the entire run time environment

can range from as low as 12K, and no larger than 50K. This depends upon the features selected by the web designer. Although the compressed image, audio, and/or video files must also be downloaded, with a reasonable web site design, web pages should load quickly. The initial run time environment is no larger than 25K, thus the initial web page should generally load in less than 2 seconds, and subsequent web pages in less than 1 second with a 56K modem, even with numerous image files.

The present invention provides a real time, dynamic linkage between JAVA and HTML including two-way communications, in real time, between JAVA and JavaScript.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing aspects and many of the attendant advantages of the invention will become more readily appreciated through the following drawings and their associated screen shots, referred to throughout the detailed description, wherein:

FIG. **1** is a flow chart depicting a prior art conceptual overview of how a user and a web browser interface.

FIG. **2** is flow chart depicting a conceptual overview of how a user interfaces with a web browser when implementing the present invention to construct a web site.

FIG. **3***a* is a schematic diagram showing the main components of a build tool in accordance with one implementation of the present invention.

FIG. **3***b* is a process flow diagram showing a build process in accordance with one implementation of the present invention.

FIG. **4***a* is schematic diagram showing the main components of a run generation tool in accordance with one implementation of the present invention.

FIG. **4***b* is process flow diagram showing a run time process in accordance with one implementation of the present invention.

FIG. **5** is a flow chart, with its attendant screen shot shown in FIG. **37**, that depicts a detailed view of a build time initialization procedure in accordance with one implementation of the present invention.

FIG. **6** is a flow chart, with its attendant screenshots shown in FIGS. **38-48**, that depicts a detailed view of the build time supported user input techniques and techniques for communication of data and status between the build engine and the interface in accordance with one implementation of the present invention.

FIG. **7***a* is a flow chart that shows an overview of the build time techniques for implementation of pop-up windows (usually called "dialog boxes" in MS Windows), the panel interface, and interface for color selection.

FIG. **7***b* is a flow chart, with its attendant screenshots shown in FIGS. **37-38**, that shows a detailed view of the build time techniques for implementation of panel interface objects, including the menu bar, menus and sub-menus, the tool bars, status fields, interactive fields, and interactive pull down lists, in accordance with one implementation of the present invention.

FIG. **7***c* is a flow chart, with its attendant screenshots shown in FIG. **37** and FIG. **63**, that shows a detailed view of the build time techniques for implementation of tabbed pop-up windows (also called "dialog boxes" in MS Windows).

FIG. **8** is a flow chart that shows a detailed view of the build time techniques for updating the internal databases and the setting of feature flags for run time optimization purposes.

FIG. **9** is a flow chart, with its attendant screenshot shown in FIG. **37**, that shows a detailed view of the build time polling methods used to facilitate communication from the JAVA build engine to the interface.

FIG. **10** is a flow chart that shows a detailed view of the build time techniques for analyzing user input for error checking and data integrity.

FIG. **11** is a flow chart, with its attendant screenshot shown in FIGS. **38-41**, that shows a detailed view of the build time methods for direct text entry at an arbitrary cursor position and text editor implementation methods.

FIG. **12** is a flow chart, with its attendant screenshot shown in FIGS. **49-56**, that shows a detailed view of the build time techniques for reading external image files, creating them on a web page, and then manipulating them through either direct mouse interaction or through the interface's panel/windows.

FIG. **13** is a flow chart that shows a detailed view of the build time implementation of text, button and image styles in accordance with one implementation of the present invention.

FIG. **14** is a flow chart that shows a detailed view of the video and audio processing in accordance with one implementation of the present invention.

FIG. **15** is a flow chart that shows a detailed view of the frame, table, forms, and draw objects processing and technology in accordance with one implementation of the present invention.

FIG. **16** is a flow chart that shows a detailed view of the build time methods for supporting various user interactions at run time.

FIG. **17** is a flow chart, with its attendant screenshots shown in FIGS. **57-58**, that shows a detailed view of the build time methods for text button and image object animation.

FIG. **18** is a flow chart, with its attendant screenshots shown in FIGS. **59-60**, that shows a detailed view of the build time methods for text button and image transformations.

FIG. **19** is a flow chart, with its attendant screenshots shown in FIGS. **61-62**, that shows a detailed view of the build time methods for text button and image time lines.

FIG. **20** is a flow chart with its attendant screenshot shown in FIG. **63**, that shows a detailed view of the build time web page transition animations and time lines.

FIG. **21***a* is a flow chart that shows a detailed view of file operations performed in accordance with one implementation of the present invention.

FIG. **21***b* is a flow chart that shows a detailed view of the view operations performed in accordance with one implementation of the present invention.

FIG. **22** is a flow chart that shows a detailed view of a dynamic web resizing process that is activated by the "Open" and "Web Site" commands under the "File" menu and the "Zoom" command under the "View" menu.

FIG. **23** is a screen shot showing a file selection window operation in accordance with one implementation of the present invention.

FIG. **24** is a flow chart showing a detailed view of an external database in accordance with one implementation of the present invention and also shows the security and optimization techniques that can be employed.

FIG. **25** is a flow chart showing a detailed view of a method for creating a customized and optimized run time engine in accordance with one implementation of the present invention.

FIG. **26** is a flow chart showing a detailed view of the methods for creating an HTML shell file in accordance with one implementation of the present invention.

FIG. **27** is a flow chart showing a detailed view of the methods for creating compressed CAB and JAR files in accordance with one implementation of the present invention.

FIG. **28** is a flow chart showing a detailed view of the technology for dynamic web page size creation in accordance with one implementation of the present invention.

FIG. **29** is a flow chart showing a detailed view of the methods for reading the multimedia database and generating the necessary objects in accordance with one implementation of the present invention.

FIG. **30** is a flow chart showing a detailed view of the methods for dynamically scaling the web page object(s) to different screen resolutions and window sizes in accordance with one implementation of the present invention.

FIG. **31** is a flow chart showing a detailed view of the methods for executing a multi-level web page and object thread architecture in accordance with one implementation of the present invention.

FIG. **32** is a schematic diagram that shows a detailed view of the web page transition animation architecture in accordance with one implementation of the present invention.

FIG. **33** is a schematic diagram that shows a detailed view of the parent object time line architecture in accordance with one implementation of the present invention.

FIG. **34** is a schematic diagram that shows a detailed view of the child object time line architecture in accordance with one implementation of the present invention.

FIG. **35** completes the flow chart begun at FIG. **31**.

FIG. **36** is a flow chart showing a detailed view of the methods for responding to user interactions in accordance with one implementation of the present invention.

FIGS. **37**-**63** are screen shots of the user interface presented by the build process in accordance with one implementation of the present invention.

DETAILED DESCRIPTION

Referring to FIG. **1**, in a prior art process for creating and displaying a web site, the user either directly writes HTML and Script Code providing user input at **1** or operates a related prior art product at **2**, which generates the HTML and Script Code at **3**. A separate file, with its attendant HTML and Script Code is uploaded for each separate web page in the web site at **4**, which is then interpreted by a browser when accessed at **5**.

FIG. **2** shows a process for creating and displaying a web site in accordance with one aspect of the invention in which, a user operates a build tool at **6**, working directly with one or more of the final web pages in a full WYSIWYG mode. The build tool accepts the user input and creates a multi-dimensional embedded multimedia object database at **7**. A run time generation process is then invoked to create the necessary run time files at **8** (including HTML shell, CAB/JAR files and a customized runtime engine) which are then loaded to a user's web site at **9**. The web page(s), when viewed by a web surfer, are activated by the browser calling the customized run time engine at **10**. The run time engine then begins to read the database and down load image, audio and video files, while simultaneously drawing the first web page for viewing or user interaction at **11**.

Build Tool and Process

FIG. **3***a* shows a build tool **350** at the detailed component level. The build tool includes a build engine **352**, interface **354**, screen sensing mechanism **356**, multi-dimensional array structured database **358**, interface's database **360**, web page scaling engine **364**, time line engine **366** and installation Program **368**. The operation and use of each of these components is described in greater detail below.

FIG. **3***b* is a flow of the build process executed by the build tool to create a web page/web site. Referring to FIGS. **3***a* and **3***b*, the process begins with an initialization (**12**) and continues through to a point where a web site has been defined and stored in the build engine's internal database (**29**).

The build tool **350** includes plural individual tools that are created and initialized at (**12**). The processes for creating and initializing build tools are described in greater detail below in association with FIG. **5**. After the build tools are created and initialized at **12**, the build tool **350** interacts with the user, receiving user commands (actions), for example, to build a web site. The build tool **350** processes user responses and communicates the same and status information to both the build engine **352** and interface **354** at **13**. The processes for interacting with the user are described in greater detail below in association with FIG. **6**.

In one implementation, the interface includes a panel (and its objects, including a menu bar, menus and sub-menus, tool bars, status fields, interactive fields and interactive pull down lists), pop-up windows (called "dialog boxes" in MS Windows), color and alert message interface technologies, built with HTML, Dynamic HTML (DHTML), JavaScript, and Cascading Style Sheets (CSS). Interface **354** responds to the user input and may display a pop-up window, update the interface objects, or display alert messages, as shown at **15**. The operation of the interface **354** is described in greater detail below in association with FIG. **7***a*, FIG. **7***b* and FIG. **7***c*.

As the build engine **352** receives data and status information, it updates an internal database (part of multi-dimensional array structured database **358**) and sets feature flags at **14**. The processes for updating the internal database and setting flags are described in greater detail below in association with FIG. **8**. To enable effective two-way communication between the interface and the build engine, polling technology is included as shown at **16**. The details of the polling process are described in greater detail below in association with FIG. **9**.

Whenever user input is received, the build tool **350** analyzes the input including error checking at **17**. In one implementation, the input is analyzed and then processed by object type (class). The process for analyzing input to determine type is described in greater detail below in association with FIG. **10**. In one implementation, the number of different object processing technology classes are four, and include direct text entry (**18**), image processing (**19**), video or audio files and links (**21**) and frames, tables, forms and draw objects (**22**). The build tool **350** processes the user input based on class. The processes invoked for direct text entry are described in greater detail below in association with FIG. **11**. The processes invoked for image processing is described in greater detail below in association with FIG. **12**. The processes invoked by the text button, paragraph, and image style technologies are described in greater detail below in association with FIG. **13**. The processes invoked for processing audio and video files and channels are described in greater detail below in association with FIG. **14**. The processes invoked for processing frames, tables, forms and draw objects are described in greater detail below in association with FIG. **15**. When an image, text button or paragraph object is to be inserted in the web page, the current style that is selected in the panel defines the initial settings used when creating the object in the web page. As such, button, image and paragraph style setting and technology will be invoked at **20** depending on the user input. The processes invoked by the paragraph style setting and technology is described in greater detail below in association with FIG. **13**.

After the input is processed as described above, a check is made to determine if one or more animation or transformation (interaction) techniques are to be invoked at **23**. The run time engine provided in accordance with the teachings of the present invention support various user interactions, including support for numerous animation and transformation techniques, and both web page and object time lines. Depending on the user selections, one or more technologies may be invoked. In the implementation shown, the build tool **350** is configured to check to determine if the input data is related to plural technologies including: user interaction technology (**24**), animation technology (**25**), transformation technology (**26**), object time line technology (**27**) and web page transition animation technology (**28**). The processes invoked for user interaction technology are described in greater detail below in association with FIG. **16**. The processes invoked for animation technologies are described in greater detail below in association with FIG. **17**. The processes invoked for transformation technologies are described in greater detail below in association with FIG. **18**. The processes invoked for object timeline technologies are described in greater detail below in association with FIG. **19**. The processes invoked for web page transition animation technologies are described in greater detail below in association with FIG. **20**.

After the build tool **350** has processed the user input, one or more file operations can be invoked at **29***a*. In one implementation, the file operations are “save”, “save as”, “new”, “close”, “open”, “apply” and “web site”. If “open” or “web site” are selected, the build tool **350** initiates the dynamic web page resizing process at **29***c* (See FIG. **22**). If “save” or “save as” are selected, the build tool **350** initiates a run generation process (See FIG. **4** and FIG. **24**). File operations “close”, “open”, and “new” can also initiate the run generation process, based on the state of the build process and user action.

At any time during the processing of user input, one or more view operations can be invoked at **29***b*. In one implementation, the view operations supported are “normal”, “preview”, “play”, and “zoom” (at various zoom percentages). If any of the “zoom” levels are selected, the build tool initiates the dynamic web page resizing process at **29***c* (See FIG. **22**). If the “preview” or “play” view operations are selected they will initiate the run time process (See FIGS. **28** through **36**).

FIG. **4***a* shows a run generation and runtime tool **370** at the detailed component level. The run generation and runtime tool **370** includes a run generation procedure **371**, web scaling engine **372**, a database **374** and a (web) page size generation engine **376** and run time engine **377** including a runtime user interaction engine **378**, a runtime timeline engine **380** and a runtime drawing, animation, audio, and video engine **382**. In one implementation, run time engine **377** includes plural engines, each of which may in themselves include plural engines.

FIG. **4***b* shows the run processes including methods for creating the run time files, including the external database, the web site specific customized run time engine, the HTML shell file, and the compressed CAB/JAR file. The run processes also include methods for scaling each web page to the web surfer’s then current screen resolution and web browser window size. After a web page has been scaled, a run time engine executes a multi-level thread technology, which presents to the viewer web pages that can operate under time lines that may include animated transitions. Associated with the web page time lines can be object time lines that may define entrance, main and exit animations, transformations, and synchronized time lines for child objects. Each object can have multiple object states, responsive to various user interactions, which can result in numerous types of visual and audio responses and actions.

Referring now to FIGS. **4***a* and **4***b* , a run generation process **360** begins by invoking the run generation procedure **377**. The run generation procedure **371** begins by creating the external database (part of database **374**) at **30**. The external database may include references to image, video and audio files, and video and audio channels. The process for creating the external database is described in greater detail below in association with FIG. **24**. A customized and optimized run time engine (run time engine **377**) is created at **31**. The customized and optimized run time engine (run time engine **377**) generates the web pages for the web site and is activated from the user’s server. The process for creating the run time engine **377** is described in greater detail below in association with FIG. **25**. The HTML shell file is created at **32**, and then the CAB and JAR files are created at **33***a* . The HTML shell file includes JavaScript Code to activate and interrogate the page size generation engine **376**, and to activate the entire runtime engine. The CAB and JAR files both include the runtime engine and database in compressed executable form. The CAB file(s) will be activated by the HTML shell file if it senses the browser as being Microsoft Explorer, otherwise it will activate the JAR file(s). The processes for creating the HTML shell file and the CAB and JAR files are described in greater detail below in association with FIG. **26** and FIG. **27**, respectively. The run generation process portion of the run processes is completed as the HTML shell file and the CAB and JAR files are uploaded to the user’s web site at **33***b*.

After the upload, the run time process **365** portion begins with the run time engine **377** invoking a web page size generation technology (engine) **376** at **34**. The web page size generation technology can be used to determine the screen resolution and the current browser window size. The process for invoking and initializing the web page size generation technology is described in greater detail below in association with FIG. **28**. The external database is read and the necessary objects generated at **35** from their stored external references. These objects include image, audio, and video objects. The processes for generating the necessary objects are described in greater detail below in association with FIG. **29**. A web page generation and scaling technology (web page scaling engine **372**) is then invoked at **36**. The web page scaling engine **372** can be used to reformat and scale objects that had been placed in a web page during the build process. The processes employed by the web page generation and scaling technology are described in greater detail below in association with FIG. **30**. The run time engine then, as necessary, executes a multilevel web page and object thread technology at **37** while the runtime user interaction portion **378** of run time engine **371** responds to user interactions at **38**. The processes invoked by the multilevel web page and object thread technology are described in greater detail below in association with FIGS. **31-35**. The processes invoked by the run time engine to respond to user interactions are described in greater detail below in association with FIG. **36**.

Detailed build processes

Referring now to FIGS. **3***a* and **5** through FIG. **22** the build tool **350** and its associated build process are described. Referring first to FIGS. **3***a* and **5**, initialization methods are shown. At **39** the build tools are created as part of the execution of the installation program **368**. They can include:

1: Initial build tool HTML/JavaScript file (IBTF)

2: An initialization engine (IE).

3: A build engine.

4: The build engine parent HTML frame file. (PFF).
5: A "Control Panel and Status Line" HTML/JavaScript File ("panel") for;
  Controlling the JavaScript database.
  Calling and initializing all pop-up windows.
  Reading all pop-up window values, and updating a JavaScript database
  Calling the build engine and passing all necessary data and status information.
  Polling the build engine for two-way JAVA/JavaScript communication.
  Displaying and updating the status of its interface objects.
  Issuing alert messages.
  Processing direct user interactions with the panel's interface objects.
6: Numerous HTML/JavaScript files, one for each pop-up window.
7: JAVA applets, embedded in HTML/JavaScript pop-up window files.
8: A build engine HTML definition file that is created and modified dynamically.

d The initialization and build engines can be placed in a JAVA wrapper so that JavaScript code may receive and process return values from JAVA methods. The initialization and build engines are also created in a "Signed" CAB file, and assigned the necessary security rights, so that the engines can assert the necessary permissions, if permitted by a given browser's security manager, when read or write operations are required. In one implementation, an installation program is run prior to the first use of the build tools. After installing all of the files, the installation program can install the necessary class libraries required by the run generation process in which the customized and optimized run time engine is created (See FIG. **25**). The installation program can also set the necessary environmental variables and installation options.

At **40** the web surfer points a browser at (i.e. calls) an initial build tool HTML/JavaScript file (IBTF). At **41** the IBTF identifies the current browser type and version number. Presently, each browser has different security manager implementations. In one implementation, the invention supports the following three categories:

1: With appropriate signing and time stamping, and with appropriate assertions of permissions, the browser will permit local read/write operations.
2: With appropriate signing and time stamping, and with appropriate assertions of permissions, the browser will permit local read operations, but write is only legal if sent to a server.
3: Local read/write operations are illegal, but are permitted on the server.

The IBTF can include a flag that can be set to indicate which security implementation is supported, so that all subsequent read/write operations will comply with the current browser's security manager.

At **42**, the IBTF causes the browser to execute the IE so as to sense the screen resolution and for adapting the interface to the user's screen resolution. In one implementation, after entering a delay loop and waiting for the IE to report it is fully loaded and initialized, the IBTF calls two IE methods, which return the width and height of the current screen and browser window. The IBTF then checks for the presence and value of a "mode cookie", to determine whether this is an initialization process, a web site open command process, or a dynamic web page resizing process. If the mode cookie is set to initialize, or it doesn't exist, the IBTF calls the IE to generate the build engine's HTML definition file. At **43** the IE then asserts the required security permission and at **44** creates a build engine HTML definition file and writes this file to the local disk (as appropriate). At **45** the IBTF then turns control over to the PFF for activating the "panel" and build engine and displaying the build engine user interface screen.

The build engine user interface screen includes a "panel" portion and a build engine portion, each of which are loaded into their respective frames, after which the web site page(s) build process can begin. Screen shot FIG. **37** shows a representation of the user interface presented by the build tool. The user interface includes a panel **400** and build frame **500**. Panel **400** includes a menu bar **410**, menus **420** and sub-menus **430**, tool bars **440**, status fields **450**, interactive fields **460**, interactive pull down lists **470** and operational pop-up windows **480**. The menu bar **410** can be used for selecting a menu command that will cause a menu to be drawn. The menu (one or menus **420**) can be used to select a feature command that could cause an operational pop-up window to be drawn, a direct user input technique or object manipulation technique to be activated, or a sub-menu **430** to be drawn. A sub-menu (one of sub-menu **430**) can cause the same type of events as that of a menu. The tool bars **440** include various icons that are shortcuts to feature commands that are also available through the menu bar and its menus. In addition, the tool bar **440** can be used to show the current state of a feature. Status fields **450** show the current value of a certain setting. Interactive fields **460** also show the current value of a setting, but can also be directly changed by the user by typing into the field, with the result immediately processed by the build engine **352** and displayed in the build frame **500**. Interactive pull-down lists **470** also show the current value of a setting, but, if selected with a mouse click, will drop down a selection list, which may have an elevator attached. The user can click on an item in the selection list, which will become the current setting with the result immediately processed by the build engine **352** and displayed in the build frame. Operational pop-up windows **480** can have tabs assigned if the number of choices within the pop-up window is large. One or more settings can be changed through a pop-up window, with the results immediately processed by the build engine **352** and displayed in the build frame **500**. These interface techniques are described in greater detail below in the build process.

The build frame **500** is used to present the actual web page as constructed by a user. The user can directly enter text, import images, video and audio for display/playback and create animations and transformations that can be viewed in the build frame. FIG. **6**, with its attendant screen shots FIGS. **38** through **48**, shows the user input techniques supported in one implementation of the invention. In one implementation, the user inputs supported include: selection from a JAVA window object (**48**); selection from a JavaScript window (**49**) including selection with dual spin control (**50***a*) or selection from a JavaScript child window object (**50***b*); direct text entry (**51**); page resizing (**52**); direct object manipulation (**53**); and, selection from a JavaScript panel (**54**).

In the implementation shown, of the six user input techniques sensed at **13**, the code for supporting selections from a JavaScript pop-up window at **49** and selections from the "panel" at **54** were implemented entirely in HTML/JavaScript Code, while support for direct text entry at **51** and direct web page object manipulation at **53** were implemented entirely in JAVA (or any other browser-based full featured programming language). In one implementation, code for supporting selections from a JAVA Window object at **48** and dynamic web page resizing at **52** are implemented using both HTML/JavaScript and JAVA. Those of ordinary skill will recognize that, JAVA could have been used more extensively to implement the methods described at **48**, **49** and **54**. However, in order to achieve the most intuitive and MS Windows like interface, and because effective two-way communication between JavaScript and JAVA had been achieved (See FIG. **9**), the languages proposed appear to best support the particular user input technique.

For example, FIG. **23** shows an actual file selection window **2300**, implemented by the invention. This type of file selection window is available in JavaScript/HTML, but not supported by JAVA for applets. File selection window **2300** greatly enhances the interface for the user, as the image, sound clip, or video clip names need not be memorized. File selection window **2300** further eliminates possible operator error when typing in a pathname or filename. The present invention utilized the strengths of JavaScript/HTML with the power of JAVA to create a unique browser based interface solution. In one implementation, the HTML form element "INPUT type=file" was embedded in a JavaScript pop-up window to create the file selection window. The file selection window returns a string value of the image (or other file type) pathname to the pop-up window. The pop-up window's JavaScript then could be used to call a JavaScript function in the panel (panel **400**) which:

1: Reads the pathname value in the pop-up window.
2: Creates a string version of a valid URL by adding the correct URL protocol to the string.
3: Updates the panel's database (interface's database **360**).
4: Calls a JAVA method in the build engine, which casts the string value of the URL into a URL object, creates an image object which is then drawn on the screen, and updates its internal database.

User inputs that are a selection from a JAVA window object (**48**) permit the implementation of a vast array of intelligent user input interface objects, from sliders to dials, which are extremely intuitive and significantly enhance the user's ergonomic experience. In one implementation, user input interface objects are supported as follows. When a selection from a JAVA window is detected, a pop-up window (applet) is presented (associated with the feature being manipulated, e.g., color, volume) and an engine method is called to begin two-way communication (for passing as arguments any necessary status information). The engine begins polling a JAVA abstract object waiting for a static variable's value to change. The pop-up applet processes the value as defined by a user interaction event, and updates the static variable in that same JAVA abstract object with the new value. Upon detecting a change in the polled static variable, the engine calls the necessary methods to process that new value. These methods include can include a brightness filter that is applied to the image bitmap utilizing techniques very similar to that of that employed by the "fade in" and "fade out" animations, described in association with FIG. **33**

User inputs for a selection from a JavaScript pop-up window (**49**) can be made in a manner identical to that of making a selection from a dialog box under MS Windows, including the use of tabbed JavaScript pop-up windows. In one implementation when a selection from a JavaScript pop-up window is detected, the panel's (panel **400**) JavaScript opens a pop-up window. The pop-up window's initial values are set from a JavaScript database defined in the panel or by the panel calling the engine for the current values and then setting the initial values. In a tabbed JavaScript window, clicking on a tab will call the pop-up window's JavaScript in order to change the state and appearance of the tabbed JavaScript window in the expected way. The pop-up window's JavaScript calls the panel's JavaScript when a completion event occurs. The panel's JavaScript reads or the pop-up window's JavaScript writes the pop-up window's field values, causing the panel's database to be updated, and the panel then calls the appropriate build engine **352** method, passing as arguments the necessary data and status conditions. Initializing the pop-up window's values and updating the panel's database upon completion can alternatively be implemented by JavaScript functions executed within the pop-up window's HTML file.

In addition, there are interface extensions that can extend beyond the usual MS Windows implementations. One is support for a selection from a dual spin control at **50**A. Screen shots FIGS. **42-45** show a visualization of an implementation of this interface technique. Screen shot FIG. **42** shows the mouse placed over an upper spin control. Screen shot FIG. **43** shows the result after the user clicked once on the upper spin control. Notice that the value has been incremented by 1, and the text button object is now at a larger point size. Screen shot FIG. **44** shows a combo box list selected by the mouse with the user about to select a significantly larger point size. Screen shot FIG. **45** shows the result of that selection, including the effect on the text button object.

In one implementation, dual spin controls are supported as follows. Each spin control has three visual states, so that when the user places the mouse over the control it appears to light up, and when the mouse button is depressed (pressed down), the spin control is modified to give the appearance of being pressed. JavaScript methods are called in the panel (panel **400**) to:

1: process each mouse click event over either spin control,
2: range check as necessary,
3: update the value in the HTML frame object residing in the pop-up window,
4: update the JavaScript (panel **400**) database,
5: call the build engine **352**, if necessary, passing the necessary value and status.

If the mouse is clicked on a combo box, the selection window opens in the usual way. If a mouse click in that window is detected, another JavaScript method in the panel **400** is called to update the JavaScript database, and call the build engine **352**, if necessary, passing the necessary value and status as function call arguments.

Another interface extension is selection from a JavaScript child window at **50**B. This technique helps simplify the number of choices given to the user in a complex pop-up window operation. A selection from a JavaScript child window can be supported as follows. The panel's (panel **400**) JavaScript opens the pop-up window. The pop-up window and its child pop-up windows' initial values are set from the JavaScript database defined in the panel **400**. The pop-up window's JavaScript opens the child pop-up window and sets its initial values. The child pop-up window's JavaScript calls the pop-up window's JavaScript when a completion event occurs. The pop-up window's JavaScript reads the child pop-up window's values, sets those values to its own internally defined variables, and calls the panel's JavaScript. The panel's JavaScript reads the pop-up window's values (which include the settings for its own fields as well as those of its child windows), updates its database, and calls the appropriate build engine **352** method, passing as arguments the necessary data and status conditions. Screen shots FIGS. **46-47** show a visualization of an implementation of a JavaScript child window. Screen shot FIG. **46** show a change text button style pop-up window. Screen shot FIG. **47** shows the result after the user selected the "Define the Mouse Down Text Button Style" child pop-up window.

Direct text entry is supported at any arbitrary cursor location. In one implementation, "text areas" are utilized in an unconventional way, in order to support full text entry, text editing, text button and paragraph styles, and reformat. Direct text entry can be defined at any arbitrary cursor location, and then text can be dragged to any other arbitrary location.

Text areas are objects that are utilized by JAVA primarily as an interface object for the implementation of a form and are generally "added" to the screen at the initialization time of a JAVA applet. Text areas are decidedly not WYSIWYG. The present invention creates text areas dynamically. Screen shots FIG. **38** through FIG. **41** show a visualization for an implementation of this technique. Screen shot FIG. **38** shows the user selecting a text object from the create text icon object from a tool bar of the panel (panel **400**). When the text icon object is selected, the cursor shape is changed to indicate the selection while the text icon object is in the select state. Screen Shot FIG. **39** shows that the cursor has changed shape and that the user has placed the cursor at an arbitrary location on the web page. Screen shot FIG. **40** shows the result after the user has clicked the mouse. A text insertion point and a selection rectangle are drawn at the arbitrary web page location. Screen shot FIG. **41** shows the result after the user has pressed the letter "W" on the keyboard. As can be seen in screen shot FIG. **41**, a draw method associated with the build process immediately hides the text area. However, text editor methods associated with the build process continue to utilize the text area as a hidden, dynamically resizing frame, whose size is subject to text button or paragraph style settings, by the amount of text, by the text's orientation (vertical or horizontal) and by the text's font style(s) and font size(s). As the build engine **352** detects a relevant mouse event or keyboard event, the build engine **352** updates the necessary variables that are defined as return values in specified build engine methods. Polling technology (see FIG. **9**) retrieves the relevant values and calls the necessary JavaScript method for processing. In one implementation, these same techniques (text area techniques) are used in the scaling technology (See FIG. **30**). Since the direct text entry and editing processes bypass completely the interface and the JavaScript code, the polling technology (See FIG. **9**) is used to pass the text string values back to the JavaScript database, in order for the interface's pop-up windows to be correctly initialized for subsequent text operations.

Direct text processing at **51** begins with the build engine **352** detecting a "Mouse Drag" or a "Mouse Double Click" event. In one implementation of the present invention, if a mouse drag event is detected, the entire initial anchor word (assuming the "mouse down" event placed the text insertion point within a word) is selected as well as the entire closing anchor word. If a double click event occurs over a word, the entire word is selected. If a double click event is detected over a special hot zone (for example, just to the left of a paragraph line), then an integral number of words are selected. Appropriate four-dimensional variables are set, and a draw system is called. The draw system paints the selected line segment in the marked text background and text color. The build engine **352** then sets a return flag to be read by the polling technology (FIG. **9**). A panel JavaScript poller (FIG. **9**) detects this flag and redraws the panel's "Text" menu object showing the choices available when text is selected. In one implementation, the "Text" menu includes choices of "Text Style", "Hot Link", "Preferences", and "Format". The states for the tool bar icon objects of "Bold", "Italic" and "Underline" are set appropriately as is the setting for the point size interactive drop-down list. The panel's JavaScript then calls an appropriate build engine method that resets the flag. If the panel's

JavaScript detects the user selecting the "Text Style", "Hot Link", "Preferences" or "Format" choices, it creates the appropriate pop-up window. Upon detecting a user completion event, the panel's JavaScript reads the data settings in the pop-up window, closes that pop-up window, and sends this data to an appropriate build engine method for processing (See FIG. **11**).

Dynamic web page resizing at **52** is invoked when the build engine **352** detects a user initiated web page resize event. This could be caused by the "Open" or "Web Site" commands from the "File" menu, or from a "Zoom" command from the "View" menu. This technology is explained in detail below in association with FIG. **22**.

Direct object manipulation at **53** includes dragging of any object, resizing of non-text objects, rotation and other image manipulation functions, as required. The processing for direct object manipulation begins by analyzing the type of object selected and the state of the object, as set by the interface based on a user's panel selection. The build engine **352** then changes the mouse cursor's appearance, and the type of selection rectangle, including which attachment points, if any, should be drawn and activated. (See FIG. **10** for the mouse event processing technology and FIG. **12** for image processing technology). In one implementation, the same direct object manipulation polling technology is used as described above with regard to direct text entry.

If a selection of an interactive field, interactive drop-down list object, or a toll bar icon object from the JavaScript panel is detected at **54**, then the following steps can be invoked, depending on the selection. The point size of a paragraph, a marked text range inside a paragraph or text button object can be changed. The state of an object's 3D frame can be changed. In one implementation, three states for an object frame are supported. The 3D frame can be drawn as a "raised" 3D object, as a "depressed" 3D object, or as a "raised" 3D object that turns into a "depressed" 3D object if a mouse down event is detected over the object to which the 3D frame is assigned. An object's style can be changed. The current web page can be changed. Finally, any other operation that has been defined by a tool bar icon object in the panel can be invoked. This includes the "file" menu choices of new, open and save, the "edit" menu choices of cut, copy and paste, inserting a text, button or image object onto the web page, applying or removing the bold, italic, and underline text attributes for a text or button object, centering or uncentering any web page object, setting the animation for a button or image object, changing the zoom level of the web page, or previewing the web site.

As each new user input is received and processed in accordance with the steps shown in FIG. **6**, at all times the internal databases of the JavaScript panel and the build engine **352** are maintained completely in synchronization. Synchronization is maintained so that: all status information displayed by the panel is current and correct; all data and status information passed to the build engine **352** from the interface are consistent with the build engine's state at any given time; the values in all pop-up windows are correctly initialized. In order to meet these requirements, all of the variables in the JavaScript panel database are explicitly "typed", to be compliant with the strict variable typing methodology generally imposed in all full featured programming languages such as Java. As JavaScript does not explicitly type anything, where using JavaScript herein, all string, Boolean, and integer variables are typed. Full two-way real time communication support between the JavaScript/HTML interface and the JAVA build engine **352** is provided as described below in association with FIG. **9**.

FIG. **7a** shows four tools utilized for an implementation of the pop-up window and panel interface technology (**15** of FIG. **3**). The panel and pop-up windows make extensive use of JavaScript mouse events, including onMouseDown, onMouseUp, onMouseOver, onMouseOut, onClick and onchange methods (**56**). The pop-up windows make extensive use of the JavaScript onLoad and onUnLoad methods. In one implementation, when a pop-up window is loaded by the panel, the panel goes into a wait loop, set for 5 times a second using the JavaScript setTimeout method, interrogating in each loop whether the pop-up window's status flag has been set. Meanwhile the pop-up window, when loaded by the browser, executes the onLoad method in order to set a flag in the panel informing the panel that the pop-up window is now loaded. Upon detecting the load event completion, the panel then proceeds to initialize the fields in the pop-up window. The panel will always close a pop-up window after detecting its completion event. However, if the user has closed the pop-up window in a non-standard way, the pop-up window executes the onUnLoad JavaScript method, which sets a flag in the panel notifying it that the pop-up window has been closed.

The JavaScript code in the panel and in all pop-up windows make extensive use of JavaScript method onKeyDown for the following operations:

1: When the focus is on the icon representing completion ("OK" is used in many MS Windows applications) causing the enter key to initiate a pop-up window/panel completion event.
2: When the focus is on the icon representing cancellation ("cancel" is used in many MS Windows applications) causing the Esc key to initiate a pop-up window/panel cancellation event.
3: When the focus is on any pop-up window or panel object, such as a data entry field, a check box, a radio button, a drop-down and scrollable list, a scrollable list, an icon, or a DHTML tab object (discussed below), the navigation keys are captured by the onKeyDown method, a JavaScript function is called, and the appropriate change is made.

For all pop-up window and panel objects, when the Tab key or the combination of the Tab key with the Shift key are detected, the onFocus JavaScript method is employed and the focus moves to the appropriate pop-up window object. If the pop-up window or panel object is a data entry field, drop-down list or a scrollable list, all cursor key operations are detected and the insertion point is adjusted accordingly. If the pop-up window or panel object is a check box, radio button a icon, or a DHTML tab object, and a cursor key (up, down, left, right, home and end keys, with or without the Ctrl or Shift keys) is detected, the onFocus JavaScript method is employed and the focus moves to the appropriate pop-up window object.

One methodology for this feature requires that all keyboard events be monitored, at all times. When the scan code for the enter key is detected, the appropriate JavaScript function is called to close a pop-up window and to call the appropriate JavaScript function for processing of the relevant data (updated in the window) and communicating, as necessary, with the build engine **352**. In another implementation, rather than the panel going into a wait loop awaiting notification from the pop-up window for data initialization purposes, the pop-up window, when loaded, executes the onLoad JavaScript method, and reads the required data values directly from the panel's database, utilizing the JavaScript "opener.fieldname-.value" technique. Similarly, the pop-up window, when detecting its completion event, updates the panel's database with the revised values from its own fields and then calls the appropriate JavaScript function in the panel for further processing. Both implementations, and any combinations, assure that the pop-up windows are correctly initialized, the panel's database is correctly updated, and the data is successfully sent to the build engine **352** for processing.

Extensive use of JavaScript technology is employed to enhance the user interface and for communication between the various HTML frames and/or files, within a given HTML frame or file, between an HTML frame and the JAVA engine, and as a bridge between two different JAVA applets (**57**). Extensive use is made of JavaScript arrays to store the values of all page and object attributes, to initialize the correct values in all pop-up windows, and to pass data and status to the engine. Various JavaScript techniques are employed to "type" all variables (JavaScript does not explicitly type anything as described above) as a prerequisite for passing values to the build engine **352**. Variables that should be typed as strings, integers and Booleans are typed through the use of "Eval" and "New" JavaScript functions. The choice of color, found in most pop-up windows to define one or more color elements, can be implemented utilizing several innovative JavaScript techniques. They include:

1: Defining a complex image map through a JavaScript function utilizing arrays. Screen shot FIG. **48** shows a visualization of an image map. A JavaScript computational loop utilizing arrays can be used to define each individual rectangle in this color palette with its appropriate RGB value and a function call to the appropriate JavaScript method.
2: Limiting the color choices from the image map to only those colors that are designated as safe colors. Safe Colors are the subset of all colors that are browser independent, assuring a consistent color look across all browsers.
3: Supporting a dual color selection technology. The user can be presented with a color palette and can click on a particular color in the color palette. Image map technology can call a JavaScript function, which converts that choice into a RGB numeric definition. This definition updates the RGB values shown in screen shot FIG. **48**, as well as passing those values, though an appropriate function call, to a build engine JAVA method. The build engine **352** will then draw the actual color immediately on the web page. Alternatively, the user can select a value from Red, Green or Blue selection lists, which can be implemented using an HTML drop-down list form object. The value selected is then processed by an appropriate JavaScript function call to a build engine method, which converts the RGB to a JAVA compliant value, and then draws the actual color on the web page.
4: Supporting True Transparency. For appropriate color elements, such as the background for a text button object, the user can choose, either from the color palette by clicking on a "transparency" rectangle, as described above, or by selecting "TR" from a Red, Green or Blue selection list. This choice is then processed by an appropriate JavaScript function call to a build engine method, that in turn sets a particular flag for the draw system (of the Build Tool) to not draw a background color for that object.

Innovative techniques are used to enable JavaScript to dynamically create HTML code based on real time conditions. Cookies can be used for data communication between HTML frames and HTML files, some of which were created in real time. Many unique combinations of HTML elements, including frames, forms, and tables, enhanced by JavaScript code, are utilized to create a extensions beyond that of the MS Windows interface (**58**). For example, a dual combo box/spin control for both small and large numeric incremental jumps can be implemented by a combination of form and table elements, mouse events, and JavaScript methods.

Extensive use of Cascading Style Sheets (CSS) was employed to create a consistent look for all pop-up windows, and for precision placement of various HTML elements (**59**).

FIG. **7***b* shows a detailed view of the build time techniques for implementation of panel interface objects, including the menu bar, menus and sub-menus, the tool bars, status fields, interactive fields, and interactive pull down lists, in accordance with one implementation of the present invention (**15** of FIG. **3**). These techniques create panel interface objects that have the same look and feel of those which are implemented under the various MicroSoft Windows Operating Systems. In one implementation of the present invention, the status fields, interactive fields, and interactive drop-down lists are defined as HTML form objects (text boxes and lists) embedded within DHTML objects. The menu bar, menus and sub-menus, and the tool bars can be defined as pure DHTML objects. However, Cascading Style Sheets can be used for all panel interface objects; although more extensively with DHTML objects as will be described below. In an alternative implementation of the present invention, the status fields and interactive drop-down lists are defined as pure DHTML objects.

In one implementation of the present invention the menu bar at **270** is defined as sets of DHTML objects, each set corresponding to a menu command. Each set consists of four DHTML objects with absolute screen positioning, one defining the DHTML object in the Mouse Over state at **278**, the second for the Mouse Down state at **279**, the third for the Active state, and the fourth for the Inactive state. Each state has a different CSS style assigned, which defines the visual appearance of that state. When the build tool is initialized at FIG. **5**, the appropriate menu commands are initialized as active or inactive at **277**. If the menu command is defined to be inactive, that DHTML inactive object is assigned by a JavaScript function to the “visible” style attribute, while the other three DHTML objects are assigned the “hidden” style attribute. Screen shot FIG. **38** shows a visualization of the “Interactions” menu command in the inactive state. In the inactive state all user interactions are ignored. If the menu command is defined to be active, that DHTML active object is assigned by a JavaScript function to the “visible” style attribute, while the other three DHTML objects are assigned the “hidden” style attribute. While in the active state, the JavaScript functions for “onMouseDown”, “onMouseUp”, “onMouseOver” and “onMouseOut” are implemented. If a Mouse Down user interaction event is detected over an active menu DHTML object at **279**, a menu command specific JavaScript function is called. This function sets the DHTML object for the Mouse Down state to the “visible” style attribute, calls a generalized JavaScript function to reset the visibility states all the other appropriate DHTML objects, set certain status variables, and set the DHTML object which defines the menu associated with that menu command to the “visible” style attribute. Screen shot FIG. **37** shows a visualization of the “Image” menu command after having received a mouse down event, with its associated menu **420** having been set to the “visible” style attribute. If a mouse up user interaction event is detected over an active menu DHTML object at **281**, a generalized JavaScript function is called in which the DHTML object defining the mouse over state is passed as a function call argument. This function sets the DHTML object defining the mouse over state to the “hidden” style attribute thus resulting in the appearance as shown for the image menu command in screen shot FIG. **37**, even when the mouse has been moved off the menu object. If a mouse over user interaction event is detected over an active menu DHTML object at **278**, a generalized JavaScript function is called in which three DHTML objects are passed as function call arguments as well as a menu command name. These DHTML objects are the ones defining the mouse over state, the mouse down state, and the associated menu. This JavaScript function first tests to see if a menu has been activated by a previous mouse down event and is still visible. If so, a generalized “reset visiblity states” function is called, then both the mouse down and associated menu objects are set to visible. Finally the same menu specific function is called as with the mouse down event. If no menu is visible, then the object associated with mouse over state is set to visible. If a mouse off user interaction event is detected over an active menu DHTML object at **281**, a generalized JavaScript function is called in which the mouse over DHTML object and the menu command name are sent as arguments. Logic tests are made to determine which menu command object has been left, as well as whether any menus are currently visible. Depending upon the results, the mouse over DHTML object may be set to hidden.

In one implementation of the present invention the menus and sub-menus at **271** are defined as a set of DHTML objects, one for each menu choice, nested inside an DHTML object that defines the entire menu. Each menu object is given absolute positioning, while the menu items are given absolute positioning relative the menu objects origin. Both the entire menu and each choice are assigned CSS styles to define their visual appearances. For each menu choice the JavaScript functions for “onClick”, “onMouseOver” and “onMouseOut” are implemented. If a mouse click event is detected at **280** and no sub-menu is defined, a feature specific JavaScript function is called. First the menu bar and the menus are set to their appropriate visibility states. Then setting their visibility attribute style to “visible” activates the appropriate tool bar icon DHTML objects. Finally the feature specific JavaScript code is executed as discussed herewithin, which may cause a pop-up window to be displayed, the Panel’s database to be updated, and/or the build engine **352** to be called. If a mouse over event is detected at **278** and no sub-menu is defined, a generalized JavaScript function is called in which the menu choice object is passed as an argument. This function first calls a generalized JavaScript function to close any pop-up windows, then set a status variable and finally executes DHTML commands to set the correct text and background colors for the object. If a mouse off event is detected at **282** and no sub-menu is defined for a menu choice either immediately above or below, a generalized JavaScript function is called in which the menu choice object is passed as an argument. A status variable is set and DHTML commands are executed to set the correct text and background colors for the object. If a sub-menu is defined for a menu choice object, then the same sub-menu specific JavaScript function are called for both mouse click or mouse over events. This function performs the same steps as that of the generalized function that was called for a mouse over event, as well as setting the sub-menu object and its menu choice objects to the visible state. Screen shot FIG. **37** shows a visualization of the menu bar’s “Image” command having been activated, the drawing of its associated menu **420**, the selection of the “Enhance” menu choice, and the drawing of the “Enhance” sub-menu **430**. In the event that the cursor is moved to an adjacent menu choice under the “Image” menu, such as “Animation.” or “Rotate”, then a specific JavaScript function is called which, in addition to the functions executed by the generalized JavaScript mouse over function, also hides the "Enhance" submenu.

In one implementation of the present invention, the tool bars at **272** are defined as a DHTML objects, and a set of DHTML objects are defined for a tool icon. The tool is given absolute positioning and is assigned a CSS style in order to define is visual appearance. Each tool icon is assigned a set of three DHTML objects all with absolute screen positioning. The first DHTML object defines the mouse over state at **278**, the second for the mouse down state at **279**, and the third for the active state. Each state has a different CSS style assigned, which defines the visual appearance of that state. For each tool icon active state object the JavaScript functions for "onClick", "onMouseDown", "onMouseUp", "onMouseOver" and "onMouseOut" are implemented. GIF images are defined for the tool bar DHTML objects, and may be always visible. The inactive "grayed out" representations for each toll icon can be drawn on this image. When the build tool is initialized at FIG. **5**, the appropriate tool icon objects are defined as active or inactive at **277**. The inactive state for an tool icon is represented when all three of its associated objects are assigned the visibility style of "hidden". Screen shot FIG. **38** shows a visualization for several inactive tool icons including the icon commands for bold, italic, underline, left and centered. All user interaction events are ignored in the inactive state. If the tool icon, based on the state of the build engine and based on the polling technology described below, is set to an active state, then the tool icon's active state object is set to the visibility style of "visible". If a mouse click event is then detected at **280**, a feature specific JavaScript function is called in a manner identical to that for a mouse click event over a menu choice object as described above. If mouse down or mouse up events are detected at **279** or **281**, then respective generalized JavaScript functions are called, in which the DHTML object defining the mouse down state is passed as a function call argument. If a mouse down event was detected, then the generalized function sets the tool icon's mouse down object to the "visible" state. If a mouse up event was detected, then the generalized function sets the tool icon's mouse down object to the "hidden" state. If mouse over or mouse out events are detected at **278** or **282**, then respective generalized JavaScript functions are called, in which the DHTML object defining the mouse over state is passed as a function call argument. If a mouse over event was detected, then the generalized function sets the tool icon's mouse over object to the "visible" state. If a mouse off event was detected, then the generalized function sets the tool icon's mouse over object to the "hidden" state. Screen shot FIG. **37** shows a visualization of the button tool icon with both its associated the mouse down and active objects set to "visible". Screen shot FIG. **38** shows a visualization of the text tool icon with both its associated the mouse over and active objects set to "visible".

In one implementation of the present invention, the status fields at **273** and the interactive fields at **274** are defined as HTML text boxes. In an alternative implementation status fields are defined as DHTML objects. For both of these panel interface object types, under certain conditions, the panel draws status information into these panel interface objects. This information can result from user input as discussed in FIG. **6**, or through the polling and two-way communication technology between the interface and the build engine **352** as discussed below. In one implementation of the present invention the status fields are:

1: The color of the selected web page object, in which the red, green and blue settings are shown.

2: The animation state of the selected button or image object.

3: The zoom level for the current web page.

4: The point size for the selected text or button object.

5: The horizontal position, in pixels, of the mouse cursor.

6: The vertical position, in pixels, of the mouse cursor.

7: The type of web page object (text, button, image, table, form object, draw object, etc.,) if selected. The type of object that the mouse is over, if no object is selected.

8: The width, in pixels, of web page object (text, button, image, table, form object, draw object, etc.,) if selected. The width of the object that the mouse is over, if no object is selected.

9: The height, in pixels, of web page object (text, button, image, table, form object, draw object, etc.,) if selected. The height of the object that the mouse is over, if no object is selected.

Screen shot FIG. **38** shows a visualization of the status fields in one implementation of the invention **450**. In an alternate implementation using DHTML objects, the status fields will appear two-dimensional rather than the three-dimensional look currently shown.

There is one interactive field defined in one implementation of the present invention. Screen shot FIG. **37** at **460** shows a visualization of the page number interactive field. In addition to the current web page being displayed, either as a number in one implementation or as a user defined name in an alternative implementation, the user can place the cursor into this field and enter the desired page to go to. A click at **280** or Enter Key event will execute this function.

In one implementation of the present invention, the interactive drop-down lists at **275** are defined as HTML form lists. In an alternative implementation, status fields are defined as DHTML objects. For both of these panel interface object types, under certain conditions, the panel draws status information into these panel interface objects. The interactive drop-down lists behave in a manner very similar to that of interactive fields, except that when selected, a selection list drops down for selection. Depending upon the number of entries in the list, an elevator object may be drawn. The operations of selecting the interactive pull down list, the selecting of a list item, or the operation of the elevator is identical to that of comparable MS Windows objects. In one implementation of the present invention the interactive pull down list are:

1: Zoom. This interface object has dual spin controls as described above and is always selectable except when in a preview mode. It shows the current zoom level.

2: Button Style. This interface object is always selectable except when in preview. It shows the button style of the currently selected button, if any. Changing the button style will change the style of the currently selected button, and/or define the style of the next button to be created.

3: Point Size. This interface object has dual spin controls as described above and is selectable when a text or button object is selected. It shows the point size of the currently selected text or button object, if any. Changing the point size will change the point size of the currently selected text or button object.

4: Paragraph Style. This interface object is always selectable except when in preview. It shows the paragraph style of the currently selected paragraph, if any. Changing the paragraph style will change the style of the currently selected paragraph, and/or define the style of the next paragraph to be created.

5: Frame State: The state of the 3D frame (none, raised, pressed or live) of the currently selected text, button, or image object.

6: Image Style. This interface object is always selectable except when in preview. It shows the image style of the currently selected image, if any. Changing the image style will change the style of the currently selected image, and/or define the style of the next image to be created.

Screen shot FIG. **37** shows a visualization of interactive drop-down lists **470**. In an alternate implementation using DHTML objects, the interactive drop-down lists will appear two-dimensional rather than the three dimensional look currently shown.

Tool bar icon objects, status fields, interactive fields, and interactive pull down lists all show feedback of the current build engine state. The technology utilized by one implementation of the invention is described below.

FIG. **7***c* shows a detailed view of the of the build time techniques for implementation of tabbed pop-up windows (**15** of FIG. **3**). These techniques create a pop-up window interface that visually and behaviorally is identical to that which is implemented as dialog boxes under the various MicroSoft Windows Operating Systems. Pop-up windows can be non-tabbed as described in FIG. **7***a*, or can have from two to as many as 10 or more tabs, depending upon the complexity of the choices available to the user for a given feature. In one implementation of the present invention each tab at **283** is defined as a DHTML object at **284**. The tab is given absolute positioning and is assigned a CSS style at **286** in order to define is visual appearance. When a click is detected through the JavaScript "onClick" function, a tab specific JavaScript function at **285** is called within the pop-up window's HTML file. This function sets the display style attribute for the DHTML objects that define the settings for all the non-selected tabs to the display style attribute of "none". The DHTML objects that define the GIF image of the non-selected tab file representations are also set to the display style attribute of "none". The display style attribute for the DHTML objects that define the settings of the currently selected tab and the GIF image that depicts the selected tab file representation is set to the display style attribute of " ". If there is to a change of the focus of the selected field within the now to be visible tab specific choices, the focus attribute for that object is executed. Screen shot FIG. **37** shows a visualization of a tabbed pop-up window, and screen shot FIG. **63** shows a collage of four views of a tabbed pop-up window with four tabs. Notice that each state of the tabbed pop-up window has a different tab file representation, showing the selected tab as being in the forefront.

The interface technology of the invention, in addition to its utilization as part of a web-based web site generation tool, can be used to provide a general purpose interface for all web-based applications that want a MS Windows compliant interface.

A process for updating the internal database of the build engine **352** is shown schematically in FIG. **8**. The database is compact and efficient in order to meet the performance requirements for the run time process. The database handles a wide selection of data objects, including multi media objects such as colors, fonts, images, sound clips, URLs, threads, and video. The database supports a multi level animation, transformation, and time line model (discussed in greater detail below). The database complies with the differing rules imposed by the various popular browser security managers.

The process begins by determining the type of data to be updated at **60**. Data that defines generic web site settings (See FIG. **21***a*), screen resolution values (See FIG. **21***a* and FIG. **24**), and the web page high watermark setting (See FIG. **24**) can be stored in a header record as boolean and integer variables and URL and color objects at **62** and **63**. Data that defines web page, paragraph, text button, and image style and text button, image and paragraph high watermark settings can be stored in one-dimensional arrays as boolean, integer and string variables and URL, font, image or thread objects at **61** and **64**. The URL, color, font, image and thread objects can also be created as required at **64**.

Data that defines text button, image, paragraph, or other parent objects and paragraph line high watermark settings can be stored in two-dimensional arrays (by web page and by object number) as boolean, integer, string, floating point variables and URLs at **65** and **66**. Again, the URL, color, font, image, audio clip, video clip, text area and thread objects can also be created as required at **66**. Data that defines a paragraph line and paragraph line segment high watermarks can be stored in three-dimensional arrays (by web page, by paragraph number, and by line number) as Boolean, integer or string variables at **67** and **68**. Again, the URL, color or font objects can be created as required at **68**. Data that defines a paragraph line segment can be stored into four-dimensional arrays (by web page, by paragraph number, by line number and by line number segment) as Boolean, integer or string variables or URL, color and font objects at **67** and **68**.

As a data field is added, changed or deleted, a determination is made at **69** on whether a value for a given high watermark needs to be changed. If so, it is updated. As a specific method in the build engine is called, a determination is made at **70** on whether a feature flag needs to be set. For example, if a particular JAVA method is called, which requires an instance of a certain JAVA Class to be executed by the run time engine, then that JAVA Class is flagged, as well as any supporting methods, variables and/or object definitions. The use of these flags is described in greater detail below in association with FIG. **25** and FIG. **27** to create a compact and efficient customized run time environment.

In one implementation, the header record, the style record, the web page record, and the object records, are carefully defined in a specific order, written in that order, and explicitly cast by object type when read by the run time engine. Exception handling can be implemented to recover from any errors. This helps assure that data integrity is maintained throughout the build and run time processes.

FIG. **9** details the polling process (**16** of FIG. **3**). The polling technology is essential for creating the necessary two-way real time communication between the JavaScript/HTML interface and the JAVA build engine. Since there is no particular difficulty for JavaScript to be able to call and pass values directly to JAVA methods, the technological challenge is to find a reasonable technique to enable JAVA to communicate back to JavaScript. The polling technology is generic, and workable across all the current browsers. The polling technology is flexible, as there are no real constraints as to what data could be communicated from the build engine to the interface, and this communication can occur at any time. The polling technology is reasonably efficient, so that the performance of the build process is not significantly affected.

In one implementation, two different techniques were utilized to implement this capability. The first was to place the build engine inside a JAVA wrapper. The JAVA wrapper accepts direct communication from JavaScript function calls, interrogates a particular JAVA build engine method, and returns that method's return value back to the calling JavaScript function. The second technique was more unconventional. A polling loop is defined in the panel's (panel **400**)

JavaScript that creates a near continuous, at least from a human perception point of view, dynamic real time link, in order to monitor events occurring inside the build engine. The result is a real time retrieval (from an ergonomic perception point of view) of necessary data and status settings from the build engine back to the interface.

Upon the loading of the panel HTML file, a JavaScript function at **71** (the poller) is immediately called which initiates a polling loop. In one implementation, the polling loop is set at a poll rate of once every 100 milliseconds or less. The polling routine, operating through the JAVA wrapper, calls the build engine in order to read the current horizontal and vertical coordinates of the mouse cursor, and displays them in the panel's status fields (FIG. **37** at **450**). The polling routine also polls the build engine in order to detect whether the mouse has moved over a valid object or, by inference, whether a mouse single click, or double click event has occurred. The poller is also constantly requesting the JAVA wrapper to return the status of an error flag in order to inform the user, if necessary, of an unrecoverable error condition, and the reason for it. (See FIG. **10**). The poller then calls a panel JavaScript function that, in turn, calls the build engine to reset the flag. The poller constantly requests that the JAVA Wrapper return the status of whether the mouse cursor is over a valid object, and, if so, that object's number, type, height and width. The poller also constantly requests the JAVA wrapper to return the status of whether an object is selected, and, if so, the type and number of that selected object, as well as the objects height, width, and 3D frame state (and the point size of the object's current font if the object is a text button or paragraph object). In addition, if the object is a paragraph, the poller constantly requests the JAVA wrapper to return a flag if a double click or drag mouse event has occurred.

At **72** the polling routine detects a mouse event based on analyzing the return values received.

The poller can infer that the mouse has either moved off or moved on to a valid object at **73** if the mouse over object state has changed or the mouse over object number has changed. If so, the poller updates the relevant interface objects of the panel as appropriate and displays them as necessary, and, depending upon whether the object is a text button object, a paragraph, image object, etc., at **75**, begins polling their unique values.

The poller can infer that a single click mouse event has occurred at **74** if the selection state has changed, or the selected object changed. The poller updates the menu bar (FIG. **37** at **410**) as appropriate, making the appropriate menu commands either active or inactive. The poller also sets the necessary status variables, and, depending upon whether the newly selected object is a text button object, a text object, image object, etc., at **74**, begins polling their unique values. The poller also activates the appropriate menu choice objects inside the "Edit" menu, the "Text" menu, the "Button" menu, the "Image" menu, and the "Interactions" menu objects (FIG. **37** at **420** and **430**), depending upon whether an web page object is selected or not, which type of web page object is selected, or, if the selected web page object is a text object, whether text is marked through a drag or double click event. In a similar manner, the poller also sets the values for the interactive field objects (FIG. **37** at **460**) and the interactive drop-down list objects (FIG. **37** at **470**). More specifically, JavaScript can poll the web page object number. The value of the web page object number can be used to initialize pop-up windows with that object's web page current values, either from the panel's database or, if necessary, by interrogating the build engine's database.

The poller can infer that a double click or mouse drag operation has occurred if the flag indicating a double click or mouse drag operation is detected at **75**. The poller calls a panel JavaScript function that, in turn, calls the build engine to reset the flag. The poller then calls a panel JavaScript function to display the appropriate panel menu choices. For example, if the double click or mouse drag event occurs within a text object, then the "Text Style and "Hot Link" menu choice objects become active under the panel's "Text" menu object.

Depending on the object type (**76**), the polling technology performs various functions. If the object is a text object at **77**, the values for the paragraph style, point size, object height and width, text color, and the 3D frame status are polled and displayed. The panel's menu objects and the menu choice objects within that are active for a text object are set to the active state, and the non-text menu choice objects are set to the inactive state, which visually means they are unavailable and are "grayed out". In addition, polling can be initiated for the creation of a hot link. If the object is a text button object at **78**, the values for the text button style, point size, object height, width, text color, animation state, and 3D frame status are polled and displayed. The menu choice objects inside the panel's menu objects that are active for a text button object are set to the active state, and the non-text button menu choice objects are set to the inactive state, which visually means they are unavailable and are "grayed out". The value of the text button object string is also polled and saved in the panel's database for use when initializing relevant pop-up windows. If the object is an image object at **79**, the values for the image style, object height, width, frame color, animation state, and 3D frame status are polled and displayed. Again, the menu choice objects inside the panel's menu objects that are active for a image object are set to the active state and the non-text button menu choice objects are set to the inactive state. In addition, the results of any relevant direct object manipulation are polled and displayed.

FIG. **10** describes a two level error correction technology (**17** of FIG. **3**) employed by the build process. Initial error checking occurs during the interactions between the user and the interface with the JavaScript error checking code at **80**. Any file name, selected by the user through the file selection window or typed in a file pathname (See FIG. **6** at **49**) is checked by the panel's JavaScript to assure that it has the correct file type suffix (gif, jpg, au, etc.) at **81**.

The panel's JavaScript Code performs range checking at **82** to prevent user error or to prevent the breaking of any internal limits imposed by the build engine. These can include: going to a non-existent web page; exceeding any limit with the dual spin control (i.e. attempting to increment or decrement a point size outside of the legal range, or trying to illegally decrement a value to zero or a minus number; typing in a numeric value that is outside a legal range; and, implicitly creating an object that exceeds a limit imposed by the build engine).

The panel's JavaScript code also checks the file pathname to make sure it contains a valid address, and makes necessary additions or conversions, if necessary, at **83**. For example, if the user selected a file from the local disk, the correct URL protocol is appended to the file name in order to make it a valid string representation of a URL address. Any illegal characters for a pathname or a null file pathname entry are also caught at **83**. In addition to file pathname validity checking there are other validity checking functions that can be employed by the JavaScript at **83**. They include the attempt by the user to enter a non-numeric character into a numeric field, or leaving an essential fill-in field empty.

The panel's JavaScript then passes these values to the build engine though the arguments of a JAVA method function call at **84**. The build engine can utilize the extensive exception handling capability of JAVA at **85** (or that of any other full featured programming language used) to attempt to recover from any processing error. If recovery is not possible, the build engine sets an error flag, utilizing the polling technology (See FIG. **9** at **71**). The poller, upon detecting this flag, informs the user, for example, through an alert JavaScript pop-up message, what non-recoverable error has occurred, from which operation, and what actions, if any, the user should take. For example, if the user had selected a corrupted image file, the exception handling technology can inform the user of this fact so that user corrective action can resolve this very common problem. In one implementation, error handling and exception recovery support is provided for a malformed URL, an input or output error, a security manager violation, and a null pointer error.

FIG. **11** shows a process for text entry and text processing (**18** of FIG. **3**). The process begins when the panel's JavaScript detects the user selecting either "Button" or "Text" icon objects from the panel's tool bar or from their equivalent menu choices under the "Button" or "Text" menus, and calls the appropriate JavaScript function at **86**. The JavaScript function, after performing a range check to assure that no internal limits of the build engine are being broken, updates its database, and sets the necessary status variables. The panel's JavaScript then calls the appropriate build engine method, passing the necessary arguments, including the current board number, the internal number to be assigned to the object, the object type, and the current text button or paragraph style at **87**. The build engine then updates its internal database and sets the necessary status variables. The build engine also changes the mouse cursor shape to that of a text entry symbol. In one implementation, the mouse cursor is shaped like a crosshair, and can be moved onto the web page (the build frame **402**) at an arbitrary location.

The build engine detects a mouse click event through its "mouseDown" method at **88**. This method reports to the build engine the exact horizontal and vertical coordinates of the crosshair mouse cursor at the moment the mouse button is pressed. The build engine places these values into its internal database. The polling process is also supported, as discussed in FIG. **9**, by placing the necessary return values in the appropriate poll enabled methods.

The build engine creates a dynamically resizable frame utilizing JAVA's "Text Area" object class, whose coordinates and size coincide with that of the draw system for the object as defined below. Other full-featured programming languages, if used by the invention, also possess similar object types. The text area is immediately overdrawn by the draw system's background paint routine. The build engine, utilizing the font metrics as defined by the selected text button or paragraph style, and utilizing the crosshair cursor's coordinates, calls the draw system. The draw system paints the background and then paints an insertion point and a selection rectangle, in the appropriate colors, and with the appropriate height and width, into the appropriate web page location at **89**. If the text button or paragraph style has a 3D frame selected, this intelligent ornamental object would also be drawn, in the appropriate color, dimensions, and thickness. Screen shot FIG. **41** shows a visualization of this process. The text insert point is in black, surrounded by a red selection rectangle, and surrounded by a blue 3D frame, as defined by the selected style. The text editor is then initialized by setting the necessary status variables.

The build engine waits until a keyboard keystroke is detected. The scan code is interpreted, and if it is a text entry key, the text editor's methods are called at **90**. The text editor processes the key event at **91**. The build engine employs frame (Text Area) processing methods and draw methods to implement the text entry and text processing functions. As a keyboard key for a text character is pressed, the build engine passes this value to the editor's text entry method, which updates both the text area's frame definition, and the draw system's database. The width of the text area is dynamically resized as necessary. If the object was a paragraph, a check is made on whether a reformat event should occur, based on the paragraph style's definition and the width of the current line's text string. If so, the appropriate text editor reformat method is called, which may cause the text area's vertical dimension to also be resized. A high watermark variable may also be set, for optimization purposes. After the final state of the text area is determined for the text entry keyboard event, the internal database for the text area, and for the paragraph or text button object, are updated. The draw system is called, and the results of the text entry event are drawn on the web page at **94**.

In one implementation, the build engine also supports the usual text processing functions found in MS Windows or Macintosh based Word Processors or Desktop Publishers at **92** and **93**. For example, if the user single clicks the mouse when over an unselected paragraph or text button object, that object is selected, a selection rectangle is drawn, the mouse cursor shape is changed to a crosshair, and the poller reports the necessary information to the panel's JavaScript. If a mouse click occurs over a selected paragraph or text button object, the editor's "Set Text Insertion Point" method is called. Based on the coordinates of the mouse cursor, and based on a calculation by the build engine as to the nearest line, and the nearest character on that line, the text insertion point can be drawn appropriately, and the necessary status variables are updated. Text entry is then processed as discussed at **91**.

If a double click or mouse drag mouse event is detected over a paragraph, an appropriate "text string selection" method is called (See FIG. **6**). Based on the coordinates of the mouse cursor, and based on a calculation by the build engine as to what text string should be selected, the internal database in updated, appropriate status variables are set, and the draw system is called for marking the text string at **94**. The polling technology is activated as discussed in FIG. **9**. The build engine's reformat methods for paragraphs can utilize a "Clean Text Stream" model for calculating line breaks and for updating four-dimensional variables utilized by the draw system in order to draw each paragraph, each paragraph line, and each paragraph line segment in the correct location, with the correct font type, font style, font size, font effect, and background and text string color. Font style refers to a font format such as Normal, Bold, Italic, or Bold Italic. Font effect refers to style overrides such as Underline, Double Underline, Small Caps, Cross Out, Superscript, Subscript, etc. The "Clean Text Stream Model" implemented by the build engine maintains multi-dimensional array pointers and records for every paragraph line and line segment external to the text string defined within the text area. Three-dimensional and four-dimensional variables are updated after each text entry or text editing and processing event in order to assure that the pointers into the paragraph text stream, defined in the text area, are current. The three-dimensional variables that the build engine has implemented can include soft and hard line end pointers for each paragraph line. Their values can be the absolute character positions within the text area text string for that line end. Hard line breaks can be created by the user pressing the enter key. Soft line breaks can be created by a reformat method based on a calculation defined below.

The four-dimensional variables can be absolute pointers into the text area text string for the beginning and end of every style override, associated with each paragraph line segment. These style overrides can include hot links, font type, font style, font size, numerous font effects, and text and background colors. For each style override there is an associated style override record that maintains all the font and color settings for that paragraph line segment. Also positional and size data such as start and end pointers into the paragraph text stream, a left offset relative to the paragraph's left origin, a top offset relative to the paragraph's top origin, and the line height. The style override record is created when the build engine detects a mouse drag or mouse double click event within a selected paragraph. When the mouse button is initially pressed, the current paragraph line and current word on that line are calculated in a manner identical to that for calculating the location of the text insertion point on a mouse click operation. The entire word becomes one anchor for the paragraph line segment, while the word defined by the mouse coordinates when the mouse button is released becomes the other anchor. Up to two other paragraph line segments can be implicitly created by the word oriented selection method. If there is text to the left of the first anchor word, and that paragraph line had not previously had a style override defined in it, the text string from the beginning of the paragraph line to the first anchor point has a style override record created for it. The values are set to that of the underlying paragraph.

If style overrides had already been created on that paragraph line, and the anchor word is inside one of them, then that style override's end pointer is adjusted to the start of the anchor word. All other style overrides, if any, to the right of the anchor word are deleted, as overlapping style overrides are not permitted. In a similar manner, the text string, if any, to the right of the last anchor point, up to the line or paragraph end, can also be defined as a style override. If a mouse click occurs before a "text style" operation, then these pointers will be reset. If the panel's JavaScript detects a user selection of "text style" from the "Text" menu, the appropriate pop-up window is drawn and its values initialized from the JavaScript database. Upon detecting a user completion event (i.e., the depressing of the enter key), the panel's JavaScript database is updated and a call is made to an appropriate build engine method, with the necessary data and status information passed as function call arguments. The build engine updates its internal database and calls the reformat method if necessary. The draw system utilizes these four-dimensional variables in order to paint the paragraph line segment style override.

The calculation for the creation or updating of a soft line break begins with the maximum paragraph width, which is set at a percentage of the browser screen width. This percentage is converted to an absolute pixel number based on the web designer's screen resolution. When any text entry or text editing and processing event occurs, a build engine method is called which calculates the width, in pixels, for the current paragraph line, based on the character string in the text area that exists between the previous line end pointer and the current line end pointer. The font definition(s) that are related to this character string are applied, and a string width is calculated. If the string width exceeds that of the maximum paragraph width, an "OverFlow" reformat method is called. The overflow reformat method calls a method to determine the settings for the last word on that line, and that word overflows to the following paragraph line. All pointers for the current line, and subsequent lines are updated as necessary, as

are all pointers and records to paragraph line segments. If the string width is less than that of the maximum paragraph width, and the text processing operation was not text entry, then an "UnderFlow" Reformat method is called. The underflow reformat method calls a method to determine the width, in pixels, for the first word on the next line. If that word will fit on the current line it is placed there. As before, all pointers for the current line, and subsequent lines are updated as necessary, as are all pointers and records to paragraph line segments. The word oriented selection technique, and the reformat, database, and draw technologies that support it, greatly simplify the text editor and produce a run time engine that is smaller, faster and more reliable.

FIG. **12** shows the operation of the image processing technology utilized by the build engine (**19** at FIG. **3**). The process begins when the panel's JavaScript detects the user selecting the "Image" icon from the panel's tool bar or the comparable menu choice under the "Image" menu. The appropriate JavaScript function is called at **95**, which draws the define image pop-up window. The user then selects an image from the file selection window with the browser, types in the image pathname for the image file on the local disk, or types in the URL for the image that exists on a server. The user could also define a 3D frame for the selected image at this time. Screen shot FIG. **49** shows a visualization of a collage for the define image pop-up window and the user's selection choices under each tab setting. The user can complete the selection process by either pressing the Enter Key or clicking on the "Create Image" icon in the pop-up window. If the Enter Key is pressed, the pop-up window's JavaScript Code utilizes the onKeyDown function, or if a mouse click, the onClick function, as described in FIG. **7**, to recognize the completion event. An appropriate error checking JavaScript function is called, which performs a file name error check, a filename validity check, and a range check to assure that no internal limits of the build engine are being broken. If the error checking tests are successful another JavaScript function is called to update the panel's database, and set the necessary status variables.

The panel's JavaScript then calls the appropriate build engine method, passing the necessary arguments, including the current internal web page number, the internal number to be assigned to the image object, the object type, and the current image style at **96**. The build engine then updates its internal database and sets the necessary status variables. It also changes the mouse cursor shape to that of an "Image Create" symbol. In one implementation, the mouse cursor is shaped like an arrow. The build engine detects a mouse click event through its "mouseDown" Method at **97**. This method reports to the build engine the exact horizontal and vertical coordinates of the arrow mouse cursor at the time the mouse button was pressed, and places these values into its internal database. The polling process is also handled, as discussed in FIG. **9**. The build engine then asserts the necessary security permission for reading from the local disk, if necessary, and attempts to create the necessary image object at the current mouse coordinates at **98**, while checking for any exception conditions as described in FIG. **10**. If no unrecoverable exceptions are reported, the internal database is updated and the draw system is called.

The image processing technology supports direct web page image object interactions at **99**, utilizing the communication technology described in FIG. **6**. The build engine first processes the mouse event as described in FIG. **7**, and places the appropriate values into a poll enabled JAVA method as described in FIG. **9**. There are two types of direct web page image object interactions.

The first occurs by simply selecting the image object with a single mouse click. A red selection rectangle is drawn around the image, as are eight attachment points. When the user has pressed the mouse cursor, the mouse cursor's shape changes to that of an anchor, which is a symbol that can be used when dragging or moving an object. The mouse's location will jump to the origin for the image. In an alternative implementation, the anchor can be defined by the mouse location at the time of the mouse drag operation. In either case, while the mouse is being dragged, the build engine updates its internal database. The build engine also updates its poll-enabled methods for communication with the interface's polling technology at **100**. The JavaScript poller reads these values, updates the panel JavaScript database, and updates the panel's interface objects. In a similar way, placing the mouse cursor over an attachment point and dragging will result in an image resizing operation. Screen shots FIG. **50** through FIG. **52** show a visualization of an image dragging operation. Screen shot FIG. **50** shows the cursor over an unselected image. Screen shot FIG. **51** shows the screen state after the left mouse button has been pressed. Notice that the image is now selected and the cursor shape has changed to the drag state. Screen shot FIG. **52** shows the screen state after the mouse has been dragged to the northwest. Notice that the image stayed selected and moved with the mouse. Screen shots FIG. **53** and FIG. **54** show a visualization of an image resizing operation for a normal image. Notice that all eight-attachment points are drawn and active for the selected image. Screen shot FIG. **53** shows the cursor over the upper left attachment point. Notice that the cursor shape has changed to a northwest to southeast resize cursor shape. Screen shot FIG. **54** shows the result after the left mouse button has been pressed over the upper left attachment point and dragged to the northwest. Notice that the image's upper left corner is still under the cursor, the image has resized, and the cursor shape remained unchanged. For image resizing operations with the mouse over and mouse down objects, only the east, southeast, and south attachment points are drawn and active.

The second type of direct web page image object interaction occurs when the panel's JavaScript code detects that the user has selected an image object interaction feature from the panel's "Image" menu. The appropriate JavaScript function is called, which sets the necessary status variables, and then calls the appropriate JAVA method, passing the necessary values as arguments. The JAVA method then sets its necessary status variables, changes the mouse cursor shape as appropriate, depending upon the type of direct image operation, and awaits a direct mouse operation on the image object. Image rotation is an example of this type of direct image interaction. In one implementation, direct image object rotation is realized by utilizing the image rotation technology described in association with FIG. **33** below. Screen shots FIG. **55** and FIG. **56** show a visualization of an image rotation for a normal image. Screen shot FIG. **55** shows the user selecting the rotate command from the "Image" menu. Immediately the cursor's shape changes to the rotate (a dual left/right arrow) cursor, and the selected image's attachment points disappear. Placing the cursor on the image and dragging will cause the image to rotate on an east/west and/or north south axis. Screen shot FIG. **56** shows the result after the mouse was dragged on an east/west plane.

Image object interactions are invoked by selecting from the JavaScript panel, selecting from a JavaScript pop-up window, and by selecting from a JAVA window object at **101**, as described in FIG. **6**. The initial values in the pop-up window are set from JavaScript's database. After any user interaction, JavaScript's database is updated and the appropriate method

in the build engine is called with the necessary settings. The build engine, after updating its internal database, calls the appropriate image processing method. The image processing routine then calls the required image filter(s), which then perform the necessary processing on the image bitmap at **102**.

An image filter is a body of code, usually consisting of one or more digital image processing algorithms, which operate on an image bitmap, and create a transformed image bitmap. An image observer can be invoked by the image filter, which then reports when the image bitmap processing has been completed. An image observer is a independent process that monitor's a particular image processing event, such as the execution of an image filter or the reading in of an image file, and reports the status of that process when queried. When the image observer reports a successful completion, the image filter can call the build engine's draw system to display the transformed image bitmap. This interaction between the build engine's image processing method, the image filter(s), the image observer, and the draw system can occur many times, depending upon the image processing operation chosen. Inage animations and image transformations, which are technologies that rely heavily on image filters, and the image observer are discussed in greater detail below in association with FIG. **16** and FIG. **17**.

FIG. **13** shows a process for implementing text button, image and paragraph style settings (**20** of FIG. **3**). The initial values for all the settings inside a parent pop-up window and associated child pop-up windows, for a particular style, can be set from the JavaScript database at **103**. The settings can include: image object styles, text button object styles and paragraph object styles.

The following settings can be initialized and changed for image object styles.

a) The following settings are initialized for all image object states (Normal, mouse Over, mouse Down) and can be changed:
  (1) resize factor.
  (2) rotation factor.
  (3) main animation type, speed, number of animation steps (resolution) and number of cycles.
  (4) image processing factors. (brightness, contrast, etc.)
  (5) 3d effects and their color values.
  (6) web page centering attribute.
  (7) web page scaling attribute.

b) The following actions are initialized and can be changed.
  (1) sound effects and audio channels.
  (2) video files and video channels
  (3) text button and image pop ups and their attributes (See 1.a above and 2.a below.)
  (4) click events.

c) The following transformation settings are initialized and can be changed.
  (1) the initial delay
  (2) up to three transformations can be defined with the following settings:
    (a) which image states should the transformation be from and into.
    (b) the speed of the transformation.
    (c) any delay before the next transformation.
  (3) whether the transformation(s) should occur simultaneously with the enter and exit time line animation or after the enter and before the exit animations.

d) The following time line settings are initialized and can be changed.
  (1) the initial delay before the image object's appearance.

(2) the enter animation type, speed, and animation resolution.
(3) the delay after the enter animation and the main animation.
(4) the exit animation type, speed, and animation resolution.
(5) the initial delay, after the entrance of the parent object, before the child text button and image object's appearance(s).
(6) the child object(s) enter animation type, speed, and animation resolution.
(7) the delay after the child object(s) enter animation.
(8) the child object(s) exit animation type, speed, and animation resolution.

The following settings can be initialized and changed for text button object styles.

e) The following attributes are initialized for all text button object states (normal, mouse over, mouse down) and can be changed:
(1) all font specifications.
(2) vertical state.
(3) all color specifications.
(4) 3d effects and their color values.
(5) web page centering attribute.
(6) font processing attributes (available in java **2**)
(7) scale, shear, and rotate (available in java **2**)

f) The following actions are initialized and can be changed.
(1) sound effects and audio channels.
(2) video files and video channels
(3) text button and image pop ups
(4) click events.

g) The following transformation settings are initialized and can be changed.
(1) the initial delay
(2) up to three transformations can be defined with the following settings:
a) which image states should the transformation be from and into.
b) the delay before the next transformation.
(3) whether the transformation(s) should occur simultaneously with the enter and exit time line animation or after the enter and before the exit animations.

h) The time line settings are the same as those defined for image objects. They also are initialized and can be changed.

The following settings can be initialized and changed for paragraph styles. The following attributes are and can be changed:

i) all font specifications.
j) all color specifications.
k) 3d effects and their color values.
l) web page centering attribute.
m) the look of hot links, including the text and background colors when the link is active and when the mouse is over the link.

The reference to JAVA **2** under text button object styles refer to the most recent version of JAVA released by Sun Microsystems. This version supports a far more robust two-dimensional processing capability than JAVA **1.6**, including significant font processing capabilities and the scaling, shearing, and rotation of objects. Currently, most conventional browsers only support JAVA **1.6**. Provisions are made in the invention so that as the then popular browsers support more robust versions of programming languages, those new capabilities can be employed to further enhance the capability of the invention.

Referring again to FIG. **13**, upon detecting the completion of editing an image, text button or paragraph style, the panel's JavaScript calls a build engine method and passes the required values. The build engine updates its internal database and sets any necessary feature flags at **104**.

When an image, text button or paragraph object is created, all the style settings for the currently selected style are applied by the build engine as part of the definition for the newly created object at **105**.

If a style is changed, all objects on all internal web pages that are utilizing that style are candidates for being changed to those new values at **106**. Flags are kept for every possible style setting for each object. If a given object is edited through the text button, image, or interaction menus or other interface objects of the panel **400**, the flags are set for any setting that are changed. If that style is subsequently changed, only those settings that have not had their flags set will be changed for any given object.

FIG. **14** describes the video and audio file and video and audio channel processing techniques employed by the build engine (**21** of FIG. **3**). A user can select a video or audio special effect (i.e. user input is provided at **107** that indicates a video or audio special effect). The method for activating a video file or video channel is defined in the text button and image object "mouse over" interactive pop-up windows described later at FIG. **16**. Methods for defining a video object as a pop-up, or a frozen object, are described with reference to the text button and image object "mouse down" interactive pop-up window also described at FIG. **16**. Audio files and audio channels can be defined in both the "mouse over" and "mouse down" interactive pop-up windows also described at FIG. **16**. The pop-up or a frozen object settings for audio are also set in the object "mouse down" interactive pop-up windows discussed therein.

As before, the panel JavaScript code initializes any pop-up windows (where the initial values are set from the JavaScript database), captures a file or channel name (from the user input) and performs file and validity error checking upon detecting a user completion action at **108**. The build engine is then called, receiving the necessary data and status as function call arguments.

The build engine determines if the audio and video definition is a file pathname or the URL of a live channel at **109**, and thereafter initiates its exception handling. If the video or audio definition is a file, the build engine performs the relevant file exception handling checks, and asserts the necessary security permissions. If there were no errors, or the exception handling error was recoverable, the build engine reads and links the video/audio file to the database, and plays the file for user verification at **110**. If the video or audio definition was a channel, the necessary pointers are updated in the database, and methods are assigned for efficient transmission, at run time by the run time engine, at **111**. The ability of the run time engine to play multiple synchronized audio and video files and channels simultaneously will be described at FIGS. **31-35**.

FIG. **15** describes the frames, tables, forms and draw objects technologies employed by the build engine (**22** of FIG. **3**) in one implementation of the invention. When the panel JavaScript code detects a user action to create a "frame", "table", "form" or "draw object" from an appropriate panel interface object, it draws and initializes the appropriate pop-up window at **112**. Upon detecting a user completion action by the pop-up window's JavaScript code, a panel JavaScript function is called to perform the necessary error checking and updating of the panel's database. Panel JavaScript thereafter calls the appropriate build engine method(s) passing the necessary data and status values as function call arguments at **113**.

The build engine updates its internal database, sets the necessary status values, and initializes, as necessary, appropriate methods for run time processing. In one implementation, the build engine includes definitions to map a given object into a relational database. Also available are a full array of database operations. Support for popular databases (such as Oracle, Informix, Sybase and DB2) are available on a real time interactive basis.

The run generation technologies, as described later in FIGS. **24-27**, are also implemented for a given frames, table, form and draw object at **114**. The run time technologies, as described later in FIGS. **28-36**, are also implemented for a given frame, table, form and draw object at **115**.

FIG. **16** describes the user interaction settings and technology employed by the build engine (**24** of FIG. **3**). Depending upon the type of object currently selected at **116** (if no object is selected no user interaction choices will be available) the panel JavaScript Code draws an appropriate pop-up window. If the selected object was a text button object at **117**, or an image object at **119**, both "mouse over" and "mouse down" choices will be available from the panel's "Interactions" menu. If the selected object is a paragraph, user interaction definitions can be activated by a double click or a mouse drag event being detected by the build engine at **118**.

More specifically, appropriate values are set in a poll-enabled JAVA routine. The JavaScript poller reads the values, and draws the appropriate panel menu choices. The "Text Style", "Hot Link", "Preferences" and "Format" pop-up windows can be chosen. If the hot link choice under the panel's "Text" menu is selected and executed, the hot link definition for internal or external web pages is captured by an appropriate JavaScript function and file pathname error and validity checking is performed. If either the "Text Style", "Hot Link", Preferences" or "ForTnat" choices under the panel's "Text" menu are selected, the panel's JavaScript draws the appropriate pop-up window. Upon detecting a user completion event, the panel's JavaScript reads the values in the pop-up window and passes the font specification parameters to an appropriate build engine method as function call parameters. The build engine then processes this data, calls a reformat method, updates its internal database, and sets the necessary four-dimensional variables for communication with the draw system.

The normal and "mouse over" foreground and background colors for the hot link, which were defined in a link look pop-up window (available under the "Text" menu of the panel), are utilized by the build engine to draw the hot link. The build engine performs the necessary exception handling, and then updates its internal database.

Based on the panel's JavaScript Code detecting whether the user chose the "mouse over" or "mouse down" choice under the "Interactions" menu) at **120**, as well as based on whether an image or text button object is currently selected, the panel's JavaScript code draws the appropriate pop-up window. Initial values for the pop-up windows are set from the panel's database at **121** and **122**.

In one implementation, the following user interaction's for the "mouse over" and "mouse down" states for text button and image objects are supported:

1: 3D Frame, in a specified color, and selected for a specified 3D appearance, can be defined for text button and image object's "mouse over" and "mouse down" states, as well as for their text, image and video pop-ups.

2: The font typeface, font style, font size, font effect(s), text color, and text background color can be defined for a text button object's "mouse over" and "mouse down" states, as well as for the text pop-ups associated from both text button and image objects.

3: Text, image, and video pop-ups can be defined for the text button and image object's "mouse over" state.

4: A sound track (file) can be defined for the text button and image object's "mouse over" state with the following choices:
   a. play once when a "mouse over" event occurs.
   b. play until a click event while on the object.
   c. play until the mouse moves off the object.

5: A sound track (file) can be defined for the text button and image object's "mouse down" state with the following choices:
   a. play once when a mouse click event occurs when over the object.
   b. play until a second click event while on the object.
   c. play until the mouse moves off the object.

6: Both video and sounds can be defined as channels as well as files.

7: The text, image, and video pop-ups can be frozen (i.e. not disappear when the mouse moves off the object after a mouse click event, for both text button and image objects).

8: Text button and image objects can have one of the following click events defined:
   a. go to a specific internal web page.
   b. go to the next internal web page.
   c. return to the parent (calling) web page.
   d. go to an external web age. That web page will replace the current web page.
   e. go to an external web page. That web page will be launched into a new window so that both web pages will be visible and accessible.

After a user completion action is detected, the panel JavaScript code performs the necessary file error and validity checking, updates its database and sets necessary status values, and then calls the appropriate build engine method, passing the necessary data values and status as function call arguments at **123**. The build engine updates its internal database, sets the necessary status variables, then draws the appropriate "mouse over" or mouse down" text button or image object states. The build engine also plays the sound or video file for user verification. The run time technology behind the user interactions will be described in greater detail in association with FIG. **36**.

FIG. **17** describes the image and text button object animation settings and technology employed by the build engine (**25** of FIG. **3**). The panel's JavaScript code determines which type of object, and which object number, from the currently selected object, as reported by the poller at **124**. When the panel's JavaScript detects a user selection of "Define Image" or "Animate" from the panel's "Image" menu, or a user selection of "Define Button" or "Animate" from the panel's "Button" menu, it draws the appropriate pop-up window and initializes the pop-up window's values from its database at **125** and **126**. Screen shot FIG. **57** shows a visualization of one implementation of the "Text Button Animation Specifications" pop-up window and the animation settings available to the user. Screen shot FIG. **58** shows a visualization of one implementation of the "image animation specifications" pop-up window and the animation settings available to the user.

When a user completion event is detected, the panel's JavaScript code captures the values from the pop-up window for the animation type, speed, resolution, and number of animation cycles at **125** and **126**, respectively, and updates its database at **127**. The panel's JavaScript then calls the appropriate build engine method, passing the necessary data and status values as function call arguments. The build engine updates its internal database and sets the necessary feature flags (See FIG. **8**.) Linkage to the appropriate animation method(s) is also set.

A thread object (a thread is an independent asynchronous program that is multiprogrammed with other threads, are defined and executed by the invention, by a JAVA Virtual Machine and by the browser) is created and executed for user verification at **128**. Values are set to integrate the given animation thread with the object time line technology (See FIG. **19**). Values are set at **129** so that when the thread object is invoked by the run time engine, the appropriate image filter(s) and animation methods are called. The run time technology behind image and text button object animations is described in greater detail in association with FIG. **31** through FIG. **35**.

FIG. **18** describes the transformation settings and technology utilized by the build engine (**26** of FIG. **3**). A transformation is defined as the changing of an object from one state to another based on a timer control, subject to user settings. In one implementation, the available states for text button and image objects are their "normal", "mouse over", "mouse down" and "pop-up" definitions. For text button objects, a transformation is implemented as the instantaneous drawing of one object state while erasing the previous object state. For images, a transformation is the gradual fading out of the previous object state, while, simultaneously, fading into the next object state.

Prior to any user menu selection, the panel's JavaScript code already knows the status of any selected object through the poller mechanism (**124** of FIG. **17**). This includes what type of object and the object's internal identifying number. When the panel's JavaScript detects a user selection of "Transform" from the panel's "Interactions" menu, it draws an appropriate pop-up window and initializes the pop-up window's values from its database at **130**. Screen shot FIG. **59** shows a visualization of one implementation of a "define the transformation for the text button object" pop-up window and the transformation settings available to the user. Screen shot FIG. **60** shows a visualization of one implementation of a "define the transformation for the image object" pop-up window and the transformation settings available to the user. When a user completion event is detected, the panel's JavaScript Code captures the values from the pop-up window based on the object type.

In one implementation, the following settings for text button objects can be specified:

1. The initial delay.
2. Up to three transformations can be defined with the following settings:
   a. Which image states should the transformation be from and into.
   b. The delay before the next transformation.
3. Whether the transformation(s) should occur simultaneously with the enter and exit time line animation or after the enter and before the exit animations.

In one implementation, the following settings for image objects can be specified:

1. The initial delay.
2. Up to three transformations can be defined with the following settings:
   a. Which image states should the transformation be from and into.
   b. The speed of the transformation.
   c. The resolution of the transformation.
   d. Any delay before the next transformation.
3. Whether the transformation(s) should occur simultaneously with the enter and exit time line animation or after the enter and before the exit animations.

The panel's JavaScript updates its database at **131**. The panel's JavaScript then calls the appropriate build engine method, passing the necessary data and status values as function call arguments. The build engine updates its internal database and sets the necessary feature flags (See FIG. **8**.) Linkage to the appropriate transformation method(s) is also set.

A thread object is created and executed for user verification at **132**. Values are set to integrate this transformation thread with the object time line technology (See FIG. **19**). Values are set at **133** so that when the run time engine invokes the thread object, the appropriate image filter(s) and transformation methods are called. The run time technology behind image and text button object transformations is described in greater detail below in association with FIG. **31** through FIG. **35**.

FIG. **19** describes the text button and image time lines and technology utilized by the build engine (**27** of FIG. **3**). A time line is an independent asynchronous process that defines the existence of a given text button or image object. An object's time line begins at the time a given web page makes its appearance, either through an immediate draw or through a transition animation. In one implementation, an object time line can be created as an instance of a class, which has a threadable interface. This instance has its own data structures, which define the animations, and transitions associated with the time line definition. An image or text button object time line can spawn child time lines, at a designated moment. A complete description of time line technology, and how they integrate the animation and transformation technologies, will be described below in association with FIG. **31** through FIG. **35**.

The build process begins the time line definition process by having the panel's JavaScript determine what is the currently selected object, utilizing the polling technology at **134**.

That is, values for the object' appearance time, animation type, speed and resolution are captured. When the panel's JavaScript detects a user selection of "time line" from the panel's "Interactions" menu, it draws the appropriate pop-up window and initializes the pop-up window's values from its database. Screen shot FIG. **61** shows a visualization of a collage of one implementation of a "define the time line for the text button object" tabbed pop-up window and the time line settings available to the user under each tab. Screen shot FIG. **62** shows a visualization of a collage of one implementation of a "define the time line for the image object" pop-up window and the time line settings available to the user under each tab'. When a user completion event is detected, the panel's JavaScript captures the values from the pop-up window based on the object type. The currently available settings, for both text button and image objects, are:

1: The initial delay before the image object's appearance.
2: The enter animation type, speed, and animation resolution.
3: The delay after the enter animation and the main animation.
4: The exit animation type, speed, and animation resolution.
5: The initial delay, after the entrance of the parent object, before the child text button and image object's appearance(s).
6: The child object(s) enter animation type, speed, and animation resolution.
7: The delay after the child object(s) enter animation.

8: The child object(s) exit animation type, speed, and animation resolution.

The panel's JavaScript updates its database at **135**. The panel's JavaScript then calls the appropriate build engine method, passing the necessary data and status values as function call arguments. The build engine updates its internal database and sets the necessary feature flags (See FIG. **8**.) A build engine method then processes all the data related to this object. The object's animation settings, if any, are integrated into the timeline at **136**. The object's transformation settings, if any, are also integrated into the timeline. If an image object, any transformation animation may be executed simultaneously with the appearance and/or exit animations, depending upon the settings. Finally, a multi-level object thread definition is created and executed for user verification. Values are set at **137** so that when the run time engine invokes the thread object, the appropriate image filter(s), animation methods, and transformation methods are called.

FIG. **20** describes the web page transition animations, time line settings and technology utilized by the build engine (**28** of FIG. **3**). When the panel's JavaScript detects a user selection of "Define" from the panel's "Webpage" menu, it draws the appropriate pop-up window and initializes the pop-up window's values for the current web page from its database at **138**. Screen shot FIG. **63** shows a visualization of one implementation of the "define the current web page settings" pop-up window and the web page settings available to the user. In the implementation shown, the choices supported include:

1: The web page delay time (which is the delay, after the completion of the last object time line, to the loading of the next web Page).

2: The transition animation, which can include a random animation choice. This is the animation applied to the web page when it is loaded and to the previous web page as it departs.

3: The number of animation frames per second, which effectively is the resolution of the transition animation.

4: The number of animation frames, which effectively defines the time expected for the transition animation to complete.

5: The web page's background color. This setting will override the generic setting for the web site, defined in FIG. **21***a*.

6: A web page boarder. This boarder, if selected, will also override the setting for the web site, defined in FIG. **21***a*. The boarder can be drawn with a 3D effect, taking the background color, and applying a transformation so that, to the human eye, a lighter and darker shade of that color will be drawn appropriately to create a 3D effect.

7: The web page's background pattern. This setting will override the generic setting for the web site, defined in FIG. **21***a*.

The panel's JavaScript updates its database at **139**. The panel's JavaScript code then calls the appropriate build engine method, passing the necessary data and status values as function call arguments. The build engine updates its internal database and sets the necessary-feature flags (See FIG. **8**.). The web page time line is synchronized with its object time lines by an appropriate build engine method at **140**. The web page's appearance delay and transition settings are integrated into the web page time line. Thereafter, a single-level object thread definition is created. Values are set at **141** so that when the thread object is invoked by the run time engine, the appropriate animation methods and object time line threads are called. Again, the run time technology behind web page transition animations and web page time lines is described in greater detail below in association with FIG. **31** through FIG. **35**.

FIG. **21***a* describes the file operations supported by the build engine (**29***a* of FIG. **3**). In one implementation, the file operations supported include:

1: "Save" at **142** or "Save As" at **143**. If the selection from the panel's "File" menu is to "Save" as a web page, the current browser screen height percentage value is sent to the build engine. The build engine updates its internal database and the build process is completed. Thereafter, the run generation process is executed. (See FIG. **24** through FIG. **27**.) If the selection is to "Save As" a template for the run generation process is also executed but the generated files are placed in the template directory. If the selection is to save as a banner or custom application, those absolute screen dimensions are sent to the build engine and its internal database is updated and the run generation process is executed.

2: "New" at **144**. A test is made by the panel's JavaScript code to see if any user input has been processed and not saved. If so, the user is asked whether this data should be saved. If so, and if the selection is to "Save" as a web page, the build process is completed and the run generation process is executed as described above. If the selection is to "Save" as a template the run generation process is executed but the generated files are placed in the template directory as described above. The panel's JavaScript code then reinitializes its database and calls a build engine method that reinitializes the build engine database.

3: "Close" at **145**. A test is made by the panel's JavaScript to see if any user input has been processed and not saved. If so, the user is asked whether this data should be saved. If so, and if the selection is to "Save As" a web page, the build process is completed and the run generation process is executed. If the selection is to "Save As" a template the run generation process is executed but the generated files are placed in the template directory. The panel's JavaScript then terminates the build process.

4: "Open" at **146**. A test is made by the panel's JavaScript to see if any user input has been processed and not saved. If so, the user is asked whether this data should be saved. If so, and if the selection is to "Save As" a web Page, the build process is completed and the run generation process is executed. If the selection is to "Save As" a template the run generation process is executed but the generated files are placed in the template directory. The panel then initiates the dynamic web page resizing technology as described in FIG. **22** below for the open re-initialization mode.

5: "Apply" at **147**. A template is applied to the existing web site that is being processed by the build engine. The web page and style record definitions of the template replace those of the existing web site. The web page objects of the template are added to the web page objects of the existing web site.

6: "Web Site at **148**. The web designer can define settings that will be applied to all web pages in the web site. In one implementation, the web site applications supported include:

a: web page. The web page height can be set, as a percentage, larger than the browser window for long vertically scrolled web pages.

b: Standard banner sizes.

c: Custom. (The user can define any arbitrary web page size and resolution)

Screen shot FIG. **63** shows the generic web site setting choices presented to the user in one implementation of the invention.

FIG. **21***b* describes the view operations supported by the build engine (**29***a* of FIG. **3**). In one implementation, the file operations supported include:

1: "Normal" at **149***a*. This is the default file mode in which the interface and the build engine are processing user input as described in FIG. **5** through FIG. **23** above.

2: "Preview" at **149***b*. The build engine runs the web site off its internal database. The web site will perform in an identical manner as if it had gone through the entire run generation and run time process, but it is being executed on the web designer's computer.

3: "Play" at **149***c*. The build engine runs the web site off an external database in a separate browser window. The web site will perform in an identical manner as if it had gone through the entire run generation and run time process, but it is being executed on the web designer's computer.

4: "Zoom" at **149***d*. The dynamic web page resizing technology (see FIG. **22** below) is first executed. When the engine is fully reinitialized, and the engine has gone to the current web page, the page and all its objects are drawn to the scale as defined by the zoom level. All object coordinates and sizes are automatically scaled appropriately because they are always defined with virtual screen values, even when the web page is being draw in the "normal" view.

FIG. **22** describes the dynamic web page resizing technology supported by the build engine. If a user selection of the "Open" command from the "File" menu is detected by the panel at **500**, the panel calls an engine method to read selected contents from that web site's external database file.

In one implementation of the invention at **506**, the engine reads the web page width and length fields, as well as the background color or background image definition for the first web page of the Web Site. The engine then creates a build engine HTML definition file based on these specifications, and writes this file either to the local disk or the server, depending upon the origination of the build tool.

At **502**, if a user completion event occurs inside the web site JavaScript pop-up window, which had been activated when the user selected the "Web Site" command from the "File" menu, the panel determines if the web site page size has been changed. If so, the panel calls an engine method for processing.

Similarly, at **504**, if a user selection of a "Zoom" command from the "View" menu is detected by the panel at **504**, the panel also calls an engine for processing.

In both the cases at **502** or **504**, in one implementation of the invention, the engine writes out a checkpoint record at **508** that is similar to that of a "Websitename".dta "database file (See FIG. **24**). But is given the temporary checkpoint Websitename. The engine then creates a build engine HTML definition file based on these specifications, and writes this file either to the local disk or the server, depending upon the origination of the build tool.

In one implementation of the invention at **510** the engine terminates itself, by stopping all of its threads. Meanwhile the interface writes out four cookies onto the local disk which define the following:

1: The re-initialization mode. (Either Open or Checkpoint).

2: The current web page number when the resizing event occurred.

3: The Web Site Name. (The checkpoint name if in checkpoint mode)

4: The zoom level.

The interface then terminates itself by executing the JavaScript "parent.location.href" command, which causes the build engine parent HTML frame file (PFF) to be reloaded (See FIG. **5**).

In one implementation of the invention at **512** the re-initialization process begins. The PFF cause both the panel and the build engine to be reloaded and activated. The panel then reads the mode cookie. If the mode is either open or checkpoint, the interface reads the web site name, page number and zoom level cookies, then resets the mode cookie to the initialize state for subsequent operations. The interface then calls an engine method to read the external database, and then to return the necessary values from that database in order to update the interface's database. Finally the engine calls two engine methods in order for the engine to go to the correct current web page and draw that page at the now current zoom level. Normal processing can then resume.

Run Generation Process

FIG. **24** through FIG. **27** describe the run generation process. This process starts when the build process detects that the user is finished defining the web site (user has saved the web site and invokes the run generation process), and concludes with the actual uploading of all the necessary web site run time files to the user's server.

FIG. **24** describes the techniques employed by the build engine for the creation of the external database, and the security and optimization techniques that support this process (**30** of FIG. **4**).

When the panel's JavaScript Code detects a user selection of "Save" or "Save As" from the panel's "File" menu, it draws the appropriate pop-up window and initializes the pop-up window's values for the current web page size as had been defined at FIG. **5** and passed to the build engine. The panel's JavaScript in the "save the web page/template" pop-up window detects a user completion event at **150** (i.e., the designation of a user's web site name followed by the enter key), and calls the appropriate panel JavaScript function. More specifically, after completing the appropriate validity checks, the function calls the appropriate JAVA build engine method, passing as a function call argument the user defined "Websitename". The build engine method checks for the existence of a "Websitename".dta file, and, if so, posts that result into a poll-enabled method return value. The poller checks that value, and if set to true, calls a JavaScript function which draws a pop-up window asking the user to confirm whether the existing web site definition should be overwritten or not. This JavaScript function also calls an appropriate build engine method to reset the return value to false in order to be initialized for the next possible "Save" operation.

Once this verification process is completed the build engine begins the external database creation process at **151**, which will vary depending upon the security manager of a given browser at **152**. See FIG. **5** for a detailed description of the browser security manager alternatives. If the browser's security manager allows for local disk file creation, the build engine calls a method, which asserts the necessary security policy permissions to create and write a file. If not, the build engine calls the necessary method to create and write a file on the user's server.

The external database contains, as its first record, a "Header" record, which contains can include the following information:

1: A file format version number, used for upgrading database in future releases.

2: The default screen resolution, in virtual pixels, for both the screen width and height. This is usually set to the web designer's screen resolution, unless overwritten by the user at FIG. **5**.

3: Whether the application is a web site.

4: Virtual web page size settings. A calculation is performed by the build engine method, in order to calculate what the maximum web page length is, after reformatting all paragraphs on all internal web pages, based on the default screen resolution.

5: Web page and styles high watermarks.

6: The Websitename.

The header records are written at step **153**.

During the build process, as new web pages or new objects are created by the user, or as text is added to or deleted from a paragraph, or as new styles are created or deleted, appropriate high watermarks are set, in order to show the current number of each of these entities. Thus, at **154**, the values for the number of active web pages and the number of text button, image, paragraph or other styles are written as high watermarks in the header. The high watermarks for the number of text button, image, paragraph or other objects that exist for each web page, the number of lines for each paragraph object, and the number of line segments for each paragraph line are written within the body of the external database as described at **156**, and used as settings for each of the loops in the four-dimensional data structure. Because no structural limits are set on the number of web pages, objects per web page, styles, or paragraph size, these high watermarks greatly reduce the external database file size, and the time it takes for the run time engine to process the data stored in its database.

The settings for all paragraph, text button and image styles are then written as a style record at **155** based on their high watermark. This data includes Boolean and integer variables, and font and color objects, written as a one-dimensional array, based on the high watermark values for the number of styles that exist. The font and color objects are serialized as is discussed in greater detail below (See **159** below).

The body of the external database is then written at **156**. All Boolean values are written inside a four-dimensional loop at **157**. The outside loop contains the Boolean values used to define web pages (i.e. a one-dimensional array definition) as well as the high watermarks for the number of text button, image, paragraph or other objects per web page, with the loop set at the high watermark which defines the number of existing web pages for this web site structure. The second level consists of three or more two dimensional loops with the loops set to the high watermarks defining the actual number of text button, image, and paragraph or other objects that appear on any given web page and contains the values used to define web page objects ((i.e. a two-dimensional array definition; web page number by object number). Included within the loop for paragraph objects are the high watermarks for the number of lines for each paragraph object. The third loop is set by the high watermark defining the actual number of paragraph lines that for all paragraphs on any web page and contains the values used to define paragraph lines (i.e. a three-dimensional array definition; web page number by object number by paragraph line.) Included within the loop for paragraph lines are the high watermarks for the number of line segments for each paragraph line. The inner most loop is set by the high watermarks defining the number of line segments per paragraph line and contains the values used to define paragraph line segments (i.e. a four-dimensional array definition; web page number by object number by paragraph line by paragraph line segment.).

All integer values are written inside a four-dimensional loop at **158**. Their four loops are controlled by the same high watermark settings as used for the Boolean records, and they describe the same logical entities.

Multimedia objects are written inside a two-dimensional loop at **159**. They include URL, color, and font objects, and can include other types of objects. A URL object is the encoded form of a URL Address, used by a browser or a JAVA method to access files and web addresses. All multimedia objects must be serialized before they can be written. This means that the objects are converted into a common external definition format that can be understood by the appropriate deserialization technique when they are read back in and cast into their original object structure. The outside loop contains web page related objects, and the inner loop contains image, text button, paragraph, etc. related URL, color, and font objects. The outer loop is defined by the web page high watermark and the inner loops by the high watermarks for the actual number of text button, image, paragraph or other objects on a web page.

String records are written inside a four-dimensional loop at **160**. The outer loop may be empty. The second loop can include the string values for text button objects, audio and video filenames, and audio and video channel names. The third loop contains values for paragraph line related data, and the innermost loop contains the values for paragraph line segment definitions. The string records are controlled by the same high watermarks as those used for Boolean and integer records. String records are stored utilizing an appropriate field delimiter technology. In one implementation, a UTF encoding technology that is supported by JAVA is utilized.

Single and double floating-point, and long integer records are written inside a two-dimensional loop at **161**. The outer loop may be empty. The inner loop contains mathematical values required for certain animations and image processing algorithms. The single and double floating-point, and long integer records are controlled by the same high watermarks as those used for Boolean and integer records.

FIG. **25** describes the techniques used to create a customized and optimized run time engine by the build engine (**31** of FIG. **4**). A versionizing program analyzes the feature flags, and only those variable definitions, defined in the "Main" object class, relating to the object classes and methods that will be executed at run time, are extracted at **162**. All references to object classes that will be called at run time are extracted, creating the source code for the run engine "Main" object class that is ready for compilation at **163**.

All external image, video and audio files are resolved at **164**. The external references can be copied to designated directories at **164**, either on the user's local disk or file server. The file Pathnames can be changed to reflect these new locations. During the installation of the build tools, the necessary class libraries from Sun, Microsoft and Netscape are either installed on the local system (See FIG. **5**) or made available on the server where the build tools can be optionally located. The necessary environmental variables are set to permit normal access to the required class libraries.

The customized run engine and a library of the referenced run time classes are compiled and converted into byte code at **165**. Finally, the run time engine for the web site is created at **166**. The required set of class objects required at run time is flagged for inclusion into the CAB/JAR file (See FIG. **27**).

FIG. **26** shows the techniques used to create the HTML Shell File (HSF) (**32** of FIG. **4**).

The first step of the process at **167** is to determine whether the dynamic web page and object resizing is desired by testing the application setting, set by the user at FIG. **21***a*, or possibly reset at FIG. **24**. If the application was a web page, and thus requiring dynamic web page and object resizing, virtual screen resolution settings, calculated at FIG. **24** at **153**, are placed in an appropriate HTML compliant string at **168**. If the application is a banner or other customized application, the absolute values for the run time object (applet size) height and width are placed in an appropriate HTML compliant string as absolute width and height values at **169**.

An analysis is made for the background definition for the first internal web page at **170**. If a background pattern is defined, an appropriate HTML compliant string for setting the HTML "background" to the same background image is generated. If the first web page definition is a color instead, then the RGB values from those colors are converted to hexadecimal and an appropriate HTML compliant String is generated setting the "bgcolor" to the required hexadecimal value. This process synchronizes the web page background with the background that will be drawn by the browser when it first interprets the HSF.

Thereafter, a JAVA method generates HTML and JavaScript compliant strings, that when executed by a browser, generate additional sets of HTML and JavaScript compliant strings that are again executed by the browser. More specifically, if the application required dynamic web page and object resizing (See **167**) then JavaScript and HTML compliant strings are generated at **171** so that, when interpreted by the browser at the time the HTML Shell File is initialized, the screen resolution sensing JAVA applet (SRS) will be executed. JavaScript code is generated at **172** in order to enable JavaScript to SRS applet communication. In one implementation, the code is generated by performing the following functions:

1: Determine the current browser type.
2: Load the SRS from either a JAR or CAB File, based on browser type.
3: Enter a timing loop, interrogating when the SRS is loaded.
4: When the SRS returns an "available" status, interrogate the SRS, which will return the current screen and window's actual height and width.
5: Convert the virtual screen resolution settings into appropriate absolute screen width and height values.

Strings defining additional JavaScript code are generated at **173** that perform the following steps at the time the HSF is initialized by the browser:

1: Generate HTML compliant strings that set the run time engine's applet size to the appropriate values.
2: Generate an HTML complaint string that contains a "param" definition for linking the run time engine to the External Database created at FIG. **24**.
3: Generate an HTML complaint string, dependent upon the type of browser, which causes the current browser to load either the JAR or the CAB File(s).
4: Generate JavaScript Code compliant strings that create and dynamically write the applet size defining HTML strings utilizing the JavaScript "document.write" function. This dynamically created code causes the browser to execute the run time engine, in the correctly sized window, from the correct JAR or CAB file, and linked to the external database.

At **174**, writing out the above-generated HTML and JavaScript compliant strings creates the HSF. The necessary security policy permissions are asserted, and a "Websitename".html file is created.

FIG. **27** describes the processes for creating the CAB and JAR Files (**33***a* of FIG. **4**). The image objects, if any, which were defined on the first internal web page are analyzed at

175. If they are set to draw immediately upon the loading of the first web page, then they are flagged for compression and inclusion in the CAB and JAR Files. The feature flags are analyzed at **176** to determine which JAVA classes have been compiled (See FIG. **25**). These class files are flagged for compression and inclusion in the library CAB and JAR Files. Strings that are BAT compliant definitions are created at **177** that will, when executed in DOS, create compressed CAB and JAR Files. These CAB and JAR Files contain the compressed versions of all necessary JAVA class files, image files, the "Websitename".class, customized run time engine file, and the "Websitename".dta database file. In one implementation of the invention, two BAT files are created. The first, when executed, will create a CAB/JAR file with the "Websitename".dta database file and the customized "main" run time engine, excluding all the image and button object animation, transformation, and image processing code. The second BAT file, when executed, will create a CAB/JAR file with all the library of all the referenced image and button object animation, transformation, and image processing code.

The necessary security policy permissions for file creation are then asserted, and "Websitename".bat and "Websitenamelib".bat files are written at **178**. The "Websitename".bat and "Websitename".bat files are then executed under DOS, creating compressed "Websitename".cab and "Websitenamelib".cab files and compressed "Websitename" jar and "Websitenamelib" jar files at **179**. The HTML Shell File and the JAR and CAB files are then, either as an automatic process, or manually, uploaded to the user's web site. This completes the run generation processes.

Run Time Process

The run time process is shown in FIG. **28** through FIG. **36**.

FIG. **28** shows the web page size generation technology utilized by the run time engine. A web surfer points a browser at the HTML shell file (HSF) at **180**. The browser begins to interpret the HTML and JavaScript code in the HSF that was created (See FIG. **26**). The browser draws either the background color or background image pattern, as defined by the HTML complaint code (See FIG. **26**) at **170**. The browser then executes the HSF's JavaScript initialization code, which "sniffs" the current browser at **181** to determine its type, and then generates the appropriate HTML code for that particular browser to interpret. This code defines whether the executable files and database will be extracted from inside a compressed CAB file or a compressed JAR file and its location.

Based on the user application (defined at FIG. **21***a*, or possibly reset at FIG. **24**), the HSF at **182** will then execute an appropriate JavaScript function (as created in FIG. **26** at **167**). If the application required dynamic resizing of the web page's dimensions, JavaScript code is called which generates HTML code using the JavaScript "document.write" function, which causes the SRS applet to be immediately executed by the browser at **183**. The JavaScript code then goes into a timer loop, checking on when the SRS applet is alive before initiating any communication.

After detecting that the SRS has been initialized, a JavaScript function calls the appropriate SRS applet methods at **185**, which return the width and height, in pixels, of the current browser window. JavaScript Code is then called which converts the screen resolution independent window width and height values into absolute pixel values. A JavaScript function is then called which use the JavaScript "document.write" function to generate HTML code that define the run time engine specifications, etc. (see FIG. **26**) and cause the browser to immediately execute the run time engine. If the application had not required dynamic resizing of the web page's dimensions, then a JavaScript function is called which generates HTML code using the JavaScript "document-.write" function that defines the fixed dimensions for the web page size and cause the browser to immediately execute the run time engine at **184**.

FIG. **29** shows the techniques employed by the run time engine to read the external database and to generate the necessary web page objects (**35** of FIG. **4**). The run time engine reads a "PARAM" value at **186**, from HTML Code that was generated above (see FIG. **26**), which points to the "Websitename".dta external database that is compressed into the JAR or CAB File (that was loaded and accessed in FIG. **28**). The run time engine then initiates the read operation. In one implementation, the read technique is always non-privileged. If permitted by the current browser as a non-privileged operation, the "Websitename".dta file will be extracted and read from the CAB/JAR file residing in temporary local storage. If not, the run time engine A will read the "Websitename".dta file directly from the server. The header record is read at **187**. Any objects, such as fonts and colors, are cast into their original form. The high watermark values, as they are encountered in the header and in the body of the database, are immediately used for setting the limits for the subsequent multi-level read loops for reading in the style record and the web page(s) and object(s) definitions. The virtual screen resolution values are read for the subsequent dynamic resizing of the web page objects.

The style record is read based on its high watermarks, and processed at **188**. The definitions for all paragraph, text button, image or other styles are read and stored for subsequent initialization and processing of all paragraph, text button, image or other objects. The data representing the values for the first web page and all its objects is read at **189**. The Boolean, integer, string and floating point fields for the first web page are initialized. The serialized multimedia objects for the first web page are read and cast into their final form. (See FIG. **24**)

If external files, such as image, audio and video files, must be read as part of the first web page's generation, exception handling routines are executed at **190**, as necessary, in the event of any processing errors. In one implementation, error recovery at this stage places the highest priority on a gracefuil operation cancellation, rather than a web page crash. In the worst case, a particular image, sound or video file may not be available to the web surfer. All other aspects of the web page will likely be available even in this error scenario.

Process step **191** is executed simultaneously with the generation of all the other web pages at **192** by means of multi-programming utilizing thread technology. Thus the first web page will be drawn and active for user viewing and user interaction long before the data for all the other web pages have been read, processed, and initialized. The data representing the values for the subsequent web pages and all their objects are read at **192**. The Boolean, integer, string and floating point fields for these web pages are initialized. The serialized multimedia objects for these web pages are read and cast into their final form. (See FIG. **24**)

FIG. **30** shows the scaling techniques employed by the run time engine for web page generation (**36** of FIG. **4**). The first step in the scaling process is to calculate the coordinates that define the origin for the placement of each object for a given web page. (This is usually the upper left corner of the object, defined in actual screen pixels.) A test is made at **193** to determine if the centering attribute is set for the object. If not, the left and top coordinates are converted from the virtual screen values to the local screen values, based on the local screen window resolution at **194**. In one implementation, multiplying the virtual coordinate by the local screen window resolution and dividing by the virtual screen resolution determine the conversion.

If the centering attribute is on, then a calculation for the object's width is performed. See processes **197**, **198**, and **199** below for a description of this calculation. Based on this calculated width, and based on the local screen window resolution, the left coordinate is calculated at **195**. One algorithm that can be used is to subtract the screen width, as calculated in **197-199** below, from the local screen window resolution, and divide that result by 2. The top coordinate is calculated the same as in process **194** above.

Based on the object type, determined at **196**, a different scaling technology is employed.

If the object is a text button object at **197**, the text button object itself, including its background, is not scaled. The virtual width and the local screen width remain the same. However, if a 3D Frame effect is defined, it is scaled based on the following algorithm: if the text string's orientation is Left to Right, the inner width of the 3D Frame, and its placement relative to the text string, is calculated as the length of the text string, plus ⅛ of an "n" space on each side, plus an additional offset appended to the right of the inner width to compensate for the italic font style, if defined for the font of that text string. The italic offset can be defined as the font size for the text string, divided by 10, plus 1. The inner height of the 3D Frame can be defined as the font height plus 2 pixels. The font height equals the font's leading plus its ascent plus its descent specifications. The inner height origin can equal the text string origin. The style of the 3D effect (i.e., either a 3D raised look or a 3D depressed look), plus the inner width and height, is sent to a 3D frame build method for the construction of the 3D frame. The width of the 3D frame in pixels can be calculated as the inner width divided by 10 plus 3.

If the text string's orientation is vertical, the inner width of the 3D Frame is an "m" space. The inner height of the 3D Frame can be calculated as the font height times the number of characters in the text string. Both the left and top placement of the 3D frame can be set to the left and top origin of the text string. The width of the 3D Frame can then be calculated as the inner height divided by 10, plus 3.

If an animation is assigned to the text string, the font size used for the initial calculation of the 3D frame is the same as that used to define the animation's initialization value. If the object is a paragraph at **198**, and the scaling attribute is on, the maximum width for the paragraph can be defined by the attached paragraph style (or paragraph override) as a percentage of the screen width. This screen width percentage can be converted into an actual width in pixels, based on the local screen's window resolution. If the current screen resolution is the same as that used by the web designer, then the paragraph line end values (just read from the external database) are used without adjustment, bypassing the entire paragraph reformat process. If the current screen resolution is different than that of the virtual screen resolution, then a very compact method of reformat is called (relative to the build engine reformat methods at FIG. **6** and at FIG. **18**), and the text for the paragraph is reformatted based on this width.

The run time engine's reformat technology begins by creating one paragraph line for the entire text string assigned to the paragraph text area. All the style overrides are renumbered sequentially with the style records or the non-marked text strings ignored. A simplified "Overflow" reformat method can be called, which chops up the single paragraph line first into paragraph line segments, where each word is defined as a line segment. Because of the word oriented style override architecture, the style overrides have a one-for-one correspondence with the line segments. Each paragraph line break can be calculated by relying on the simplified word oriented style override technology described above. The paragraph line can be built inside a tight word-by-word loop, with a simple logic check for a style override or hard line break. The paragraph width is then derived as the width of the longest line of the reformatted paragraph, while the paragraph height is defined as the font height times the number of lines. If a 3D frame was defined for the paragraph, it can be scaled based on the following algorithm:

The inner width is defined as the same as that of a text string, but the width of the text string for the longest line is used. The same "n" space and italic offset calculations are used. The inner height is calculated as the font height times the number of lines plus 2 pixels.

If the object is a paragraph, and the scaling attribute is off, then the paragraph is treated the same as a text button object, with the only exceptions that there is no vertical orientation, and the height and width of the 3D frame, if defined, is calculated using the same algorithm as was used for the scaled paragraph above.

If the object is an image at **199**, and the scaling attribute is on, the image width can be calculated as the virtual width times the local screen window width divided by the virtual screen width. The image height can be calculated as the virtual height times the local screen widow height divided by the virtual screen height. If the image had been resized or rotated, then the virtual width and height of the image would differ from that of that of the original image. If a 3D frame is defined for the image, it can be scaled based on the following algorithm:

The inner width and the inner height of the 3D frame will coincide exactly with the outer edges of the image, after the image had been scaled. Adding the scaled image height to the scaled image width and dividing the result by 40 and adding 3 can calculate the width in pixels of the 3D frame.

If an animation is assigned to the image, then the animation's initialization values for the image's width and height can be used to calculate and draw the initial 3D frame. The coordinates and sizes for the backgrounds for text button, image and paragraph objects can be calculated using the same algorithms as was employed for the calculation and placement of the inner width and inner height for the 3d frame for each object.

FIG. **31** through FIG. **35** shows the multilevel web page and object thread technology employed by the run time engine. The description includes all the animation technologies, transformation technologies, time line technologies and drawing technologies that support this multilevel architecture.

FIG. **31** describes the initial processes for the invention's multilevel web page and object thread technology employed by the run time engine (**37** of FIG. **4**). Upon the completion of the processing of all the data definitions for the first internal web page (FIG. **30**), the main web page thread is created and executed. This causes the run method for the main run time engine class to be executed simultaneously with the reading, processing, and scaling of the data for the subsequent web pages (See FIG. **29**). In addition, the reading of any image files defined for the first web page is also performed simultaneously, under the control of an image observer (See FIG. **12**). The main run method enters a web page counter loop at **200**, the loop being defined from the first internal web page to the high watermark that was set to the number of existing internal web pages for the web site.

A check is made at **201** to see if the current web page exists. If the web page does not exist, and the current web page number is less than that of the high watermark, then the web page counter is incremented by one and the web page counter loop is reentered. If the current web page number equals the high watermark at **202**, then the web page counter is reinitialized to the first web page, so that the web page loop may repeat itself, from the first internal web page, depending upon the delay setting for the last web page.

A test is then made on all objects defined for this web page at **203**, utilizing a loop whose range is defined by the number of objects per web page high watermarks. More specifically, within this universe of possible objects, if the object exists, and it is defined by a time line in which there is a delayed entrance, then a boolean flag is set for those objects that causes the draw system to suppress drawing these objects during the web page transition as defined below.

A test is then made to determine if the web page has a transition animation defined at **204**. If not, the draw system is called for the first time. The draw system for a given web page utilizes a loop whose range is defined by the number of objects per web page high watermarks. The draw system can also employ technology so that the draw process generates a screen image in one or more off-screen buffers, only drawing to the screen when the screen image, or the clipping area for the screen, has been fully generated. This greatly reduces, if not totally eliminates, any screen flicker, and creates visually smooth animation effects.

The first draw function is to draw the web page background into the primary off-screen buffer. The web page background color is drawn, as defined initially at FIG. **21***a*, or modified for that particular web page at FIG. **20**. A test is then made to determine if the web page has a background image pattern, as defined initially at FIG. **21***a*, or modified for that particular web page at FIG. **20**. If it does, and the image observer reports that the image is ready to be drawn, a background image draw loop is executed, defined by the height and width of the background image, and the screen resolution of the current browser window. In the unlikely event that the background image pattern is not yet available, there is a delay until the image observer reports the completion of the image processing operation. The tiled background image pattern is also drawn into the primary off-screen buffer, completely overdrawing the background color. The backgrounds for all non-suppressed (See **203**) parent web page text button and paragraph objects are then drawn into the primary off-screen buffer, unless a background transparency flag has been set (See FIG. **7**).

The text strings for non-suppressed parent web page text button and paragraph objects are then drawn into the primary off-screen buffer. These text strings are drawn based on their font name, style, size, effect(s), and color. If a paragraph line string, the string may have multiple string segments, each with their own font name, style, etc. If the text button object has its vertical attribute set to true, then the draw system executes a loop defined by the number of characters defined in the text button object. The top and left origin coordinates were set in the usual way (See FIG. **30**), but the top coordinate is adjusted by the font height for each iteration of this draw loop. The intelligent 3D Frame, if defined, is then drawn into the primary off-screen buffer for the paragraph and text button objects (See FIG. **30**). The primary image objects for the web page are then processed by the draw system. If the image observer reports that the image is ready to be drawn, it is drawn into the primary off-screen buffer, based on the coordinates and size as defined in FIG. **30**. If not ready, there is a delay until the image observer reports the completion of the image processing operation. The Intelligent 3D frame, if defined, is then drawn into the primary off-screen buffer for the image objects (See FIG. **30**).

The draw system is responsive to two other technologies at this stage. The first is user interaction based on the location of the mouse cursor and any user initiated mouse event. This subject will be described in greater detail below in association with FIG. **36**. The second is object animation for non-delayed web page objects. This subject will be described in greater detail below in association with FIG. **33**.

If the web page transition test at **204** was true, then the run time engine's main run method executes the web page transition animation technology at **205**.

FIG. **32** describes the web page transition animation technology. First a lock is placed on this method at **212**, as a safety precaution to prevent any interference from other threads during the animation. A test is then made on whether the transition animation setting (See FIG. **20**) for the web page is random at **213**. If so, a random transition number is generated at **214**. The web page thread then begins a particular animation loop at **215**, depending upon the random number that was generated at **214** or by the transition animation that was set previously (at FIG. **20**). In one implementation, 13 different transition animations plus random are supported including. They are: Fade In, Zoom In, Zoom Out, Zoom to Upper Left, Zoom to Lower Right, Rotate to the Left, Rotate to the Right, Rotate Bottom to Top, Rotate Top to Bottom, Slide to the Left, Slide to the Right, Slide Bottom to Top, and Slide Top to Bottom.

For all web page transition animations, the X and Y animation increment values are calculated by dividing the current browser's screen width and height by the user defined animation resolution at **215**. In all animation and draw loops, the number of loops can equal the number of animation frames as set at FIG. **20**. The timer delay for all animations, in milliseconds, can be calculated by dividing the number of frames per second (See FIG. **20**) into 1,000.

For a description of "Fade In" Technology see FIG. **33**. A "Zoom In" algorithm sets the initial scaled width and height for the current web page image to zero and the prior web page image to its full size. In each animation and draw loop the previous web page's final image state is drawn into a secondary off-screen buffer at **216**. (If this is the first occurrence of the first web page, then the secondary off-screen buffer is set to the background of the first web page.) The upper left hand corner (origin) of the current web page can be calculated based on the following formula: browser screen width minus the scaled width divided by two.

The scaled image of the current web page is then drawn into the secondary off-screen buffer at the calculated origin, using the current scaled width and height for the web page image. This merged image of the prior and scaled version of the current web page is then drawn to the screen.

A timer delay then occurs as defined at **215**, after which the X and Y animation increment values are added to the scaled width and height for the current web page image. The animation loop is then repeated to its conclusion at **218**.

The other eleven web page transition animations follow a similar methodology, but have quite different calculations, which are based on the following variables:

1: Order of drawing of the prior and current web pages.

2: Initialization values for the X and Y origin coordinates for the current and prior web pages.

3: The initial values for the scaled width and height for the current and prior web pages.

4: Whether X and Y origin coordinates for the current and prior web pages increment, decrement, or remain the same.

5: Whether the values for the scaled width and height for the current and prior web pages increment, decrement, or remain the same.

For the "Zoom Out" animation, the current page is drawn first and always drawn at 100%. The prior web page is initialized also at 100%, but its X and Y origin coordinates are incremented and its scaled width and height values are decremented, by the appropriate values, for each animation iteration.

For the "Zoom to Upper Left", "Zoom to Lower Right", "Rotate to the Left", "Rotate to the Right", "Rotate Bottom to Top" and "Rotate Top to Bottom" animations, a common data initialization and data increment strategy is implemented.

1: The X and Y variables for page image one is set to zero.

2: The X and Y variables for page image two is set to the right and bottom edges of the browser window.

3: The scaled width and height variables for page image one is set to 100% of the browser window's resolution.

4: The scaled width and height variables for page image two is set to zero.

5: For each loop iteration, the scaled width and height variables for page image one are decremented by the X and Y animation increment values defined at **215**.

6: For each loop iteration, the scaled width and height variables for page image two are incremented by the X and Y Animation increment values defined at **215**.

For the "Zoom to Upper Left" animation, the prior web page is drawn first; with its X and Y origin coordinates always set to zero. (upper left corner of the browser window) Its scaled width and height values are always set to the current values for scaled width and height variables for page image one. The X and Y origin coordinates for the current web page can be calculated by subtracting the current values of image two's scaled width and height variables from the initial values of the X and Y variables for page image two. The scaled width and height values for the current web page can be set to the current values for the scaled width and height variables for page image two.

For the "Zoom to Lower Right" animation, the current web page is drawn first; with its X and Y origin coordinates always set to zero. Its scaled width and height values are always set to the current values for scaled width and height variables for page image two. The X and Y origin coordinates for the prior web page are set to current values of image two's scaled width and height variables. The scaled width and height values for the prior web page are set to the current values for the scaled width and height variables for page image one.

For the "Rotate to the Left" animation, the prior web page is drawn first; with its X and Y origin coordinates always set to zero. Its scaled width value is set to current value of image one's scaled width variable. Its scaled height value is always set to the bottom of the browser's window. The X origin coordinate for the current web page can be calculated by subtracting the current value for image two's scaled width variable from the initial value for image two's X origin coordinate. The Y origin coordinate for the current web page is always set to zero. Its scaled width value is set to current value of image two's scaled width variable. Its scaled height value is always set to the bottom of the browser's window.

For the "Rotate Bottom to Top" animation, the prior web page is drawn first; with its X and Y origin coordinates always set to zero. Its scaled width value is set to the width of the browser window. Its scaled height value is set to current value of image one's scaled height variable.

The current web page's X origin coordinate is always set to zero. The Y origin coordinate is calculated by subtracting the current value of image two's scaled height variable from the initial value for image two's Y origin coordinate. Its scaled width value is always set to the right edge of the browser's window. Its scaled height value is set to current value of image two's scaled height variable.

For the "Rotate Top to Bottom" animation, the current web page is drawn first; with its X and Y origin coordinates always set to zero. Its scaled width value is set to the width of the browser window. Its scaled height value is set to current value of image two's scaled height variable.

The prior web page's X origin coordinate is always set to zero. The Y origin coordinate is set to the current value of image two's scaled height. Its scaled width value is always set to the right edge of the browser's window. Its scaled height value is set to current value of image one's scaled height variable.

For the "Slide to the Left", "Slide to the Right", "Slide Bottom to Top" and "Slide Top to Bottom" transition animations, a common data initialization and data increment strategy is implemented. The strategy includes:

1: The X and Y variables for page image one is set to zero.

2: The X and Y variables for page image two is set to the right and bottom edges of the browser window.

3: For each loop iteration, the X and Y variables for page image one are incremented by the X and Y animation increment values defined at **215**.

4: For each loop iteration, the X and Y variables for page image two are decremented by the X and Y animation increment values defined at **215**.

5: The scaled width and height values always remain at 100% of the browser windows width and height.

For the "Slide to the Left" animation, the prior web page is drawn first; with its X and Y origin coordinates always set to zero. The current web page's X origin coordinate is set to the current value of page image two's X variable. Its Y origin coordinate is always set to zero.

For the "Slide to the Right" Animation, the current web page is drawn first, with its X and Y origin coordinates always set to zero. The prior web page's X origin coordinate is set to the current value of page image one's X variable. Its Y origin coordinate is always set to zero.

For the "Slide Bottom to Top" animation, the prior web page is drawn first; with its X and Y origin coordinates always set to zero. The current web page's Y origin coordinate is set to the current value of page image two's Y variable. Its X origin coordinate is always set to zero.

For the "Slide Top to Bottom" animation, the current web page is drawn first; with its X and Y origin coordinates always set to zero. The prior web page's Y origin coordinate is set to the current value of page image one's Y variable. Its X origin coordinate is always set to zero.

After the last animation cycle is completed for any of the transition animations at **218**, the animation process is unlocked, and process step **206** shown in FIG. **31** is then executed.

Returning to FIG. **31**, the main web page thread's run method then executes a text button and image object time line, transformation and animation loop at **206**. This range loop is defined from the first object on the given web page to the high watermark for the number of those objects on a web page for this web site. A test is made on each object on whether an animation, transformation and/or time line has been assigned at **208**.

If so, an "instance" of the time line class for that particular object type is created at **209**. An "instance" of a class is a fundamental aspect of object oriented programming (OOP). Each time, the line class is implemented with a "runnable" interface, so that they can be executed as independent threads.

Communication of data, between the "instance" of a class and the main run engine class can be accomplished in OOP using several different techniques. In one implementation, this construction, passed as an argument, is used to permit different objects to address each other's variables and databases. A thread is then created, utilizing a two-dimensional object internal database architecture (web page number by internal object number). This methodology is convenient for permitting all object time lines for a given web page to be managed and synchronized. The object's thread is then "started".

The result of this process at **209** is that an independent thread has been created for each appropriate object on a given web page, all executing simultaneously with each other and with the main run time engine web page thread, subject to the definitions of their independent time lines at **210**. See FIG. **33** for a description of the time line technology. When the main web page thread has finished the text and image loop at **207**, the draw system is activated; the run time engine can now respond to user interactions, and the main web page thread transitions into a "Join" loop at **211**. See FIG. **35** for a description of this process.

FIG. **33** shows the time line technology used by the run time engine. The techniques and algorithms employed to create this technology permit each web page object to have an independent yet synchronized existence with each other, with the main web page thread, and with child objects that each main or parent object may spawn. Furthermore, each object and each of their child objects are capable of performing multiple animations and transformations, either serially or simultaneously. Database initialization is first accomplished for each object thread. This assures that the object thread's database is set to the correct initial values as required for that particular object, and that the references to the main web page thread's database are established.

A test is then made to determine if the object has a time line definition assigned to it at **219**.

If not, a test is made at **220** on certain two-dimensional object definition variables in order to determine which of the following four states have been defined for the object: animation without a transformation; transformation without animation; animation, with the transformation occurring simultaneously with the animation; and animation and transformations occurring in a serial manner.

If the test shows that the object has an animation defined, but no transformation, then certain two-dimensional status variables are set, and an "instance" of the "animation class" for that particular object type is created at **229**. Each "animation class" is also implemented with a "runnable" interface. An object animation thread is then created, utilizing the two-dimensional object internal database architecture (See FIG. **8**). This object animation thread is then "started". Communication between the object animation thread, the parent time line thread, and its parent, and the main web page thread, are accomplished as discussed in process **209**. The object time line thread then executes a "Join" method. This puts the object time line thread in a "wait state". When the thread it is waiting for is completed, this child thread "joins" the parent object time line thread, and the object time line thread then continues its process. Other forms of synchronization between two independent threads could have been implemented as is known in the art.

The techniques employed at **229** to implement object animation vary by object type. In one implementation, for text button object animations, 26 different animations are supported including: Zoom In, Zoom Out, Grow NW, Grow NE, Grow SE, Grow SW, Shrink SE, Shrink SW, Shrink NW, Shrink NE, Enter N, Enter NE, Enter E, Enter SE, Enter S, Enter SW, Enter W, Enter NW, Exit N, Exit NE, Exit E, Exit SE, Exit S, Exit SW, Exit W and Exit NW. In one implementation, for image object animations, 29 different animations are supported including: Fade In, Fade out, Rotate, Zoom In, Zoom Out, Grow NW, Grow NE, Grow SE, Grow SW, Shrink SE, Shrink SW, Shrink NW, Shrink NE, Enter N, Enter NE, Enter E, Enter SE, Enter S, Enter SW, Enter W, Enter NW, Exit N, Exit NE, Exit E, Exit SE, Exit S, Exit SW, Exit W and Exit NW.

As discussed above with regard to FIG. **17**, each animation type has a defined speed, resolution, and number of animation cycles. These settings are stored in the main web page class, and are passed to the particular animation thread through a two-dimensional object internal database architecture as discussed in process step **209** above during the animation thread's initialization process. The animation thread then executes, in its run method, a main animation loop that has the number of iterations set to the end number of animation cycles, as assigned to that particular text button object.

Text button animations are currently implemented in three logical groups. Group One includes "Zoom In", "Grow NW", "Grow NE", "Grow SE", and "Grow SW". Group Two includes "Zoom Out", "Shrink SE", "Shrink SW", "Shrink NW", and "Shrink NE". Group Three includes "Enter N", "Enter NE", "Enter E", "Enter SE", "Enter S", "Enter SW", "Enter W", "Enter NW", "Exit N", "Exit NE", "Exit E", "Exit SE", "Exit S", "Exit SW", "Exit W" and "Exit NW".

For Group One text button animations, the animation font size is initialized at a very small value, and in one implementation is set at 4 Points. The animation point size increment can be derived by dividing the resolution (number of animation frames) into the font size for that text button object. The run method then executes a secondary animation loop, which will terminate when the animation font size equals the text button object point size. For each secondary animation loop, the length of the current animated text string is calculated, a new font object is created for the current animation point size, and the font metrics for that new font are created. If the text button object has a vertical orientation, the animated text button object's width is calculated to be the width of an "m" space, in the current animated font. The animated text button object's height is calculated to be current animated font height times the number of characters in the text string. If the text had a horizontal orientation, the animated text button object's width is calculated to be the width of the text string in the current animated font. The animated text button object's height can be calculated to be the font height of the current animated font. The calculations for X and Y coordinates for the animated text button object depend upon which animation was defined within the Group One-text button animations. The X and Y animation increments can be calculated utilizing the height and width, in pixels, of the text button object scaled to the current browser's window, utilizing the text button animation resolution, and considering whether the animating text button object is being centered during the animation ("Zoom Out") or not. These calculations are similar to those for the web page transition animations discussed with regard to FIG. **32**.

The draw system is then called. Based on the values of the two-dimensional status variables that had been set initially, the draw system executes the appropriate animation draw routine utilizing, through the data communication techniques already discussed, the current animation font point size, and the current animation X and Y coordinates. If a text background is to be drawn, the same algorithm as defined in FIG. **31** is used. If a 3D Frame is assigned, the current animated string width and height are passed to the appropriate 3D frame generation method, and the frame is drawn with the same algorithm as defined in FIG. **31**, but utilizing the current animation X and Y coordinates. The text button object's orientation is also handled by the draw system with the same algorithms as defined in FIG. **31**.

The text button animation thread then executes a timer delay, whose value had been defined in FIG. **17**. When the timer reactivates the text button animation thread after the appropriate delay, an animation cycle completion test is made to see if the text button object's point size minus the animation point size is less than the animation point size increment. This type of testing methodology permits the invention to utilize integer values, as opposed to floating point values, for the text button animation. This improves the execution of the animation considerably.

If the above test is true, the animation point size is set equal to the object point size and a final call is made to the draw system for that animation cycle. A test is then made to see if the current animation cycle equals the total number of animation cycles as defined in FIG. **17**. If not, a new animation cycle is initiated, with the animation values reinitialized. If this was the last animation cycle the text button animation thread calls its "stop" method, which sets the required status variables as appropriate, then terminates itself This causes the parent text button time line thread to be reactivated through the "join" mechanism.

If the results of animation cycle completion test are false, the current animation point size is increased by the animation point size increment. A new font object is created for the now current animation point size, and new font metrics for that new font are created. If the text button object has a vertical orientation, the animated text button object's width is calculated to be the width of an "m" space, in the now current animated font. The animated text button object's height is calculated to be the now current animated font height times the number of characters in the text string. If the text has a horizontal orientation, the animated text button object's width is calculated to be the width of the text string in the now current animated font. The animated text button object's height is calculated to be the font height of the now current animated font. The calculations for the new X and Y coordinates for the animated text button object are then completed, as appropriate, and the draw system is called again.

The algorithms for Group Two text button animations are very similar to those of Group One. The differences are just in what are the initial animation values, and whether the animation point size increments and the animation X and Y coordinate increments are added or subtracted from the then current animation point size and the then current X and Y coordinates for the animating text button object.

For Group Three text button animations, the distance that the text button animation will move is calculated, in pixels, from its initial location to its final location in the current browser window. The X and Y animation increments are calculated by dividing that distance by the resolution of the text button animation. All the other algorithms for Group Three text button animations are generally a subset of those for Group One, and similar to the web page slide transition animations defined with reference to FIG. **31**.

Referring again to FIG. **33**, image animations at process step **229** can currently be grouped into five logical classes. As with text button animations, Group One includes "Zoom In", "Grow NW", "Grow NE", "Grow SE", and "Grow SW". Group Two includes "Zoom Out", "Shrink SE", "Shrink SW", "Shrink NW", and "Shrink NE". Group Three includes "Enter N", "Enter NE", "Enter E", "Enter SE", "Enter S", "Enter SW", "Enter W", "Enter NW", "Exit N", "Exit NE", "Exit E", "Exit SE", "Exit S", "Exit SW", "Exit W" and "Exit NW". In addition, image animations have a Group Four, which includes "Fade In" and "Fade Out". Group **5** image animations include the "Rotate" Animation.

For Group One mage animations, the animation width and height increments are calculated by dividing the image object's width and height by the resolution (number of animation frames) as set in FIG. **17**. The initial animation width and animation height values are set to a very small number, currently equal to the animation width and height increment values just calculated. The calculations for X and Y coordinates for the animated image object depends upon which animation was defined within the Group One text button animations. The X and Y animation increments are calculated utilizing the height and width, in pixels, of the image object scaled to the current browser's window, utilizing the image animation resolution, and considering whether the animating image object is being centered during the animation ("Zoom Out") or not. These calculations are similar to those for the web page Transition Animations discussed above with regard to FIG. **32**.

The run method then executes a secondary animation loop, which will terminate when the animation width equals the image object's width. The algorithms employed by the invention to change the animating object's height, width, X coordinate, and Y coordinate are very similar to those employed for Group One text button animations, and will not be repeated here. The techniques to utilize the draw system for drawing the image animation, the time delay technique, and the post draw logic tests and actions are also very similar.

The algorithms for Group Two image animations are very similar to those of Group One. The differences are just in what are the initial animation values, and whether the animation width and height increments and the animation X and Y coordinate increments are added or subtracted from the then current animation width and height and the then current X and Y coordinates for the animating image object.

For Group Three image animations, the algorithms are identical to those of Group Three text button animations. For Group Four image animations, the "alpha" value of a given image object is utilized in order to implement "Fade In" and "fade Out" animations. The alpha value can range from 0 to 255, depending upon the image strength desired. The value for an alpha animation increment variable can be calculated by dividing the resolution of the animation into 255, after making the necessary adjustments to keep the data in integer form, without losing resolution due to integer rounding errors. For a "Fade In" animation the value of an alpha animation variable is set to zero. The run method then executes a secondary animation loop, which will not terminate until **255** minus the then current value of the alpha animation variable is less than the value alpha animation increment variable. A "Fade In" image filter can be created for each iteration of the animation loop, using the current setting of the alpha animation variable. An image producer can also be created with pointers to the last image bitmap produced for the image object in the last animation loop and to the image filter that has just been created. The image producer, under the control of a media tracker then creates a new image bitmap. The animation thread then "waits" for the completion of this image-processing event using the media tracker. Upon completion, the draw system is called which draws the then current state of the image object. The image animation thread goes into a timer delay of some preset value (in one implementation 500 milliseconds), to permit a smooth visual animation effect. The value for the alpha animation increment is added to alpha animation variable and the loop is then repeated until the loop condition is met. Then the "stop" method is called, certain status variables are set, and the image animation thread terminates itself. This causes the parent image time line thread to be reactivated through the "Join" mechanism.

The "Fade Out" animation employs very similar technology, except that:

1: the alpha animation variable is set to zero,
2: the value for the alpha animation increment is subtracted, and
3: the loop termination test is when the value for the alpha animation variable is less than the value for the alpha animation increment.

For the Group Five image rotate animation, a different bitmap for the image object is created for each animation frame through the use of a progression of standard geometrical transformations on the original image bitmap. A secondary animation loop is then executed as defined by the number of animation frames. In each loop iteration, an image object is created from an appropriate image bitmap selected from among the set just created, the necessary two-dimensional variables are set to communicate with the draw system, and the draw system is then called. The image animation thread then executes a timer delay method based on the delay setting as defined above with reference to FIG. **17**. When the timer reactivates the image animation thread after the appropriate delay, the next iteration of the secondary animation loop is repeated until the loop condition is met. Then the "stop" method is called, certain status variables are set, and the image animation thread terminates itself. This causes the parent image time line thread to be reactivated through the "join" mechanism.

Returning to process step **220** shown in FIG. **33**, if the object had a transformation, but not an animation, then certain two-dimensional status variables are set, and an "instance" of the "transformation class" for that particular object type is created at **228**. Each "transformation class" is also implemented with a "runnable" interface. An object transformation thread is then created, utilizing the invention's two-dimensional object internal database architecture. This object transformation thread is then "started". The inter-thread communication technology and the "join" technology employed for object transformations is the same as for object animations.

If the transformation is being applied to a text button object at **228**, then a timer delay method is executed based on the delay setting as described in association with FIG. **18**. When the timer reactivates the text button transformation thread after the appropriate delay, the appropriate two-dimensional status variables are set to inform the draw system which state of the current text button object to draw. The draw system is called and, based on the settings for the above mentioned two-dimensional status variables, either the "normal", mouse over", mouse down" or "pop-up" states of the text button object's background, if any, the text button object's string, and the 3D frame, if any, are drawn. If additional transformations are defined (FIG. **18**), the above process is repeated, based on the timer delay and object states defined for the subsequent transformations. When the last transformation is completed, the "stop" method is called, which sets the required status variables as appropriate. This causes the parent text button time line thread to be reactivated through the "join" mechanism.

If the transformation is being applied to an image object at **228**, then a timer delay method is executed based on the delay setting (as defined in FIG. **18**). When the timer reactivates the image transformation thread after the appropriate delay, image transformation technology is executed. In one implementation, the image transformation technology utilizes the "alpha" value of a given image object state in order to fade in and fade out images. The alpha value can range from 0 to 255, depending upon the image strength desired. The value for an alpha transformation increment variable is calculated by dividing the resolution of the transformation into 255, after making the necessary adjustments to keep the data in integer form, without losing resolution due to integer rounding errors. The value of an alpha transformation variable is set to zero. Depending upon the settings as defined in FIG. **18**, the bitmap for one image object state is initialized to an alpha value of zero, while another is initialized to an alpha value of 255. The appropriate two-dimensional status variables are set for communication with the draw system.

A transformation loop is then executed, until 255 minus the then current value of the alpha transformation variable is less than the value alpha transformation increment variable. This methodology again keeps all calculations in the form of integers, as opposed to floating point, thus speeding up the transformation process.

Two "Fade In" image filters are created for each iteration of the transformation loop. The first uses an alpha value calculated at the current setting of the alpha transformation variable. The second uses an alpha value calculated at 255 minus the current setting of the alpha transformation variable. Two image producers are also created with pointers to the last image bitmap produced for each image object state in the last transformation loop and to the two image filters that had just been created. The two image producers under the control of two media trackers then create two new image bitmaps. The transformation thread then "waits" for the completion of these two image processing events using the media trackers. Upon completion, the draw system is called which draws the then current state of the two image object states, in the correct order, and in the correct location. The image transformation thread goes into a timer delay of some preset value (500 milliseconds in one implementation), to permit a smooth visual transformation effect. The loop is then repeated until the loop condition is met. Then the "stop" method is called, certain status variables are set, and the image transformation thread terminates itself. This causes the parent image time line thread to be reactivated through the "join" mechanism.

Returning to process step **220** in FIG. **33**, if the object was defined with an animation and transformation that would execute in a serial manner, then certain two-dimensional status variables are set, and an "instance" of the "transformation" class for that particular object type is created at **230**. An object transformation thread is then created, utilizing the two-dimensional object internal database architecture. This object transformation thread is then "started" and the parent object time line thread "waits" to be "joined".

If a text button object, then a primary loop is executed, with the number of iterations set to the number of transformations. After the execution and return from a timer delay event, if any, an "instance" of the text button animation class is created, and then a text button animation thread is created and "started". The parent text button transformation thread then waits to be "joined". This causes the text button animation thread to be executed, in the manner described at **229**. When the text button animation thread completes its execution, it calls its "stop" method, which sets the necessary status variables and then terminates itself. This causes the text button animation thread to "join" the parent text button animation thread, causing that thread to resume processing. The first text button transformation is then executed, in the manner described at **228**. After the execution and return from another timer delay event, if any, another "instance" of the text button animation class is created, and then another text button animation thread is created and "started". The parent text button transformation thread again waits to be "joined". This causes the text button animation thread to be executed again with the animation being executed, based on the definition set at FIG. **18**, on a different text button object state. The loop is then repeated until the last text button transformation is completed. Then the text button transformation thread calls its "stop" method, certain status variables are set, and the text button transformation thread terminates itself. This causes the parent text button time line thread to be reactivated through the "join" mechanism.

If an image object, the mechanism of the image transformation thread spawns image animation threads, before each transformation, and is the same as that of a text button object. The actual image transformation process is identical to that described at **228**. When completed the "stop" method is called, certain status variables are set, and the image transformation thread terminates itself. This causes the parent image time line thread to be reactivated through the "join" mechanism.

Returning to process step **220** in FIG. **33**, if the object was defined with a simultaneous animation and transformation, then certain two-dimensional status variables are set, and an "instance" of the "super transformation class" for that particular object type is created at **231**. In one implementation, the animation, transformation, and super transformation classes are integrated into one structure in order to reduce code size and increase execution speed. Each "super transformation class" is also implemented with a "runnable" interface. An object super transformation thread is then created, utilizing the two-dimensional object internal database architecture. This object super transformation thread is then "started". The inter-thread communication technology and the "join" technology employed for object super transformations is the same as for object transformations.

If a text button object, a calculation is made in order to prorate the text button animation process across the defined text button transformation process. The calculation is driven by the number of text button animation frames, and prorates from that total the number of frames that should be assigned to each transformation state. This can be done by dividing the sum of all the transformation times by each individual transformation time, and multiplying that result by the number of frames, making necessary adjustments to prevent integer rounding error. After these calculations are completed, the text button animation is executed in a similar manner as was defined at **229**. However, when the appropriate number of animation frames had been drawn, certain two-dimensional status variables are set prior to calling the draw system for the next animation frame, so that the correct text button object state is drawn, in the correct size and with the correct coordinates, by the draw system. When the super transformation process is completed the "stop" method is called, certain status variables are set, and the text button super transformation thread terminates itself. This causes the parent text button time line thread to be reactivated through the "join" mechanism.

If an image object, a calculation is made in order to prorate the image animation process across the defined image transformation process. The calculation is driven by the number of image transformation events that would occur (where each one can be set at approximately 500 milliseconds) over the entire animation event. A calculation is performed in order to calculate how many image transformation events should be assigned to each transformation state. This is done by dividing the sum of all the transformation times by each individual transformation time, and multiplying that result by the total number of transformation events, making necessary adjustments to prevent integer rounding error. A calculation is then made to allocate the number of animation frames to each image transformation event. After these calculations are completed, the image animation is executed in a similar manner as was defined at **229**. However, when the appropriate number of animation frames had been drawn, the image transformation technology is called to perform the next transformation event. The alpha transformation increment can be defined by dividing 255 by the number of transformation events assigned to that transformation. The draw system is then called. When the number of image transformation events assigned to a given image transformation is reached, then certain two-dimensional status variables are set prior to calling the draw system for the next animation frame, so that the correct image states, in the correct size and with the correct coordinates, are utilized by the draw system. This entire animation/transformation process will be repeated by the number of image animation cycles. When the super transformation process is completed the "stop" method is called, certain status variables are set, and the text button super transformation thread terminates itself. This causes the parent text button time line thread to be reactivated through the "join" mechanism.

Returning to process step **219** in FIG. **33**, if the object had a time line, then a test is made at **221** on whether an appearance delay had been defined in FIG. **19**. If so, a timer event is set at **222**.

When the timer reactivates the object time line thread after the appropriate delay, a test is made on whether an entry animation/transformation has been defined for this object time line at **224**, as described FIG. **19**. If so, based which animation/transformation process was defined, it is created and executed at **228**, **229**, **230**, or **231**. In one implementation, 13 entry animations are supported for both text button and image objects, and an additional "Fade In" entry animation is supported for image objects. The 13 common entry animations supported include Zoom In, Grow NW, Grow NE, Grow SE, Grow SW, Enter N, Enter NE, Enter E, Enter SE, Enter S, Enter SW, Enter W and Enter NW

If no entry animation/transformation is defined, or when the entry animation/transformation has "joined" the object time line thread, a test is made to determine if any child time lines have been defined at **225**, as described in FIG. **19**, for this parent object time line. If so, an "instance" of the "child time line class" for that particular object type is created at **226**. Each "child time line class" is also implemented with a "runnable" interface. An object child time line thread is then created, utilizing the two-dimensional object internal database architecture. This object child time line thread is then "started". The inter-thread communication technology and the "join" technology employed for object child time lines is the same as for object time lines. Either a text button child time line thread or an image child time line thread, or both, can be spawned at this time.

Simultaneous with the execution of any spawned text button child time line threads, the parent object thread then executes the defined main animation and or transformation. As with non-time line object threads, a test is made on certain two-dimensional object definition variables in order to determine which of the following four states have been defined for the object at **227**: animation without a transformation; transformation without animation; animation, with the transformation occurring simultaneously with the animation; and, animation and transformations occurring in a serial manner.

Based on the results of this test, an appropriate "instance" of an appropriate animation, transformation, or super transformation class is created, and an appropriate animation, transformation, or super transformation thread is created and "started". This results in the execution of process steps **228**, **229**, **230**, or **231**, as defined above.

The parent object time line thread then executes a "join" method. This again puts the object time line thread in a "wait state". When the thread it is waiting for is completed, the child thread "joins" the parent object time line thread, and the object time line thread then continues its process.

The object time line thread then checks to see if there is a departure delay defined at **232**. If so, it sets a timer event at **233**. When the timer reactivates the object time line thread after the appropriate delay, a test is made at **234** on whether an exit animation/transformation has been defined for this object time line, as described in FIG. **19**. If so, it is created at **235**, and performed as discussed with reference to processes **228**, **229**, **230**, or **231**. In one implementation, 13 exit animations supported for both text button and image objects, and an additional "Fade Out" exit animation is supported for image objects. The 13 common exit animations include: Zoom Out, Shrink NW, Shrink NE, Shrink SE, Shrink SW, Exit N, Exit NE, Exit E, Exit SE, Exit S, Exit SW, Exit W and Exit NW.

If no exit animation/transformation is defined, or when the exit animation/transformation has "joined" the object time line thread, the parent object time line thread then executes a "join" method if it had spawned any child time lines. This again puts the object time line thread in a "wait state". Finally, when then the child time line threads, if any, "join" the parent object time line, the "stop" method for the parent time line is called. Certain status variables are set, and the parent object time line thread terminates itself. This causes the main web page time line that had been in a "join" loop at **211** of FIG. **31**, since the invocation of the object time lines, to be "joined" by this particular object time line thread.

FIG. **34** shows the technology employed by the run time engine for implementing child time lines for text button and image objects. Child text button object time lines and child image object time lines are subsets of their parent object time lines. First a test is made at **237** on whether an appearance delay had been defined (See FIG. **19**). If so, a timer event is set at **238**. When the timer reactivates the child object time line thread after the appropriate delay, a test is made on whether an entry animation has been defined for this child object time line at **239** (as described FIG. **19**). If so, it is created and executed at **240** in a manner identical to that described at process step **229** in FIG. **33**. The same 13 entry animations supported for parent object time lines are also supported for both child text button and image objects, and the additional "Fade In" entry animation is supported for child image objects. The "join" mechanism described in FIG. **33** is employed in an identical manner at **240** to synchronize the child time line thread with its entry animation thread.

After being "joined" and reactivated, the child object time line performs a test at **241** on whether an exit delay had been defined (See FIG. **19**). If so, a timer event is set at **242**. When the timer reactivates the child object time line thread after the appropriate delay, a test is made on whether an exit animation has been defined for this child object time line at **243**, as described in association with FIG. **19**. If so, it is created and executed at **244** in a manner identical to that described above at process step **229** in FIG. **33**. The same 13 exit animations supported for parent object time lines are also supported for both child text button and image objects, and the additional "Fade Out" exit animation is supported for child image objects. The "join" mechanism described above in association with FIG. **33** is employed in an identical manner at **245** to synchronize the child time line thread with its parent object time line thread. As discussed at process step **236** in FIG. **35**, the parent object time lines "wait" until all their child time lines have terminated, before they in turn terminate and "join" the main web page time line at FIG. **35**.

FIG. **35** describes technology employed by the run time engine for the web page and object thread loop. As noted in FIG. **31** at process step **211**, after all the text button and image time line threads for the current web page had been launched, the main web page thread executed a "join" loop, waiting for the completion of all the parent object time line threads. Because each parent object time line thread waited for their child object time line threads to be "joined", as well as any other spawned animation threads, transformation threads, and/or super transformation threads, the effect of this "join" loop at **246** is that the web page thread will not resume processing until all parent time line threads have completed and that of all of their spawned threads.

Upon resuming its processing after the "join" process at **246** has been completed, the main web page thread checks at **247** to see whether the current web page has an automatic termination, based on a timer delay, or whether the web page will wait for a user interaction before terminating.

If the web page has a time delay based termination setting, then a timer method is called at **249**, and the web page goes to "sleep" awaiting the completion of the timer event.

When the timer event occurs, the web page thread resumes processing by incrementing the web page counter by one, and the entire web page process, which began at process step **200** in FIG. **31**, is repeated. If the current web page termination setting was to set to wait until user interaction, then Web page thread is placed in a "pause" state, and the run time engine waits to respond to any mouse, keyboard or other user initiated event.

FIG. **36** describes the technology employed by the run time engine for responding to user interactions. As mentioned in association with process step **204** of FIG. **31**, as soon as the draw system has been activated, the run time engine will respond to any user interactions that have been defined (See FIG. **16**). This is also true during any object time line events, as with respect to process step **207** of FIG. **31**. The run time engine currently responds to "mouse over" and "mouse down" events for text button, image, and paragraph objects. For form objects, the run time engine will also respond to keyboard events. As the full-featured programming languages supported by browsers evolve, the run time engine may be configured to respond to other user interactions, including but not limited to single and double clicks from both the left and right mouse button, voice commands, eye focusing technologies, touch screen technologies, and push technologies originating from a server.

The run time engine invokes a "dynamic mouse to object recognition" technology at **251** in order to be responsive to the following elements:

1: The location of objects will vary based on the viewer's screen resolution and browser window size as discussed above with regard to FIG. **27**.

2: Objects may move or resize themselves based on time lines and animations.

3: Objects may have different sizes based on the state they are being displayed in based on time lines and transformations.

4: More than one object can occupy the same screen location, and which objects occupy that location may change over time based on time lines, animations, and transformations.

The run time engine maintains, in its internal database, the object's current X and Y origin coordinates, and the object's current width and height, in pixels, based on the current viewer's screen and browser window size. This can be accomplished by first converting all coordinates and sizes to the current viewer environment with the scaling technology as discussed above with regard to FIG. **27**. Every time line, animation, and transformation thread updates, in real time, the run time engine's internal database positional and size variables of the objects they define, utilizing the data communication techniques described above with reference to FIG. **33**.

The run time engine employs mouseEnter, mouseMove, mouseDown, mouseDrag, mouseUp, and mouseExit methods to constantly monitor the state of the mouse at all times. The onClick method (to detect a single click) and a specialized method to detect a double click event are also employed. The onKeyDown method, with processing the returned scan code, permits the run time engine to process all keyboard events. The mouseEnter, mouseMove, mouseDown, mouseDrag, mouseUp, and mouseExit methods return to the run time engine the exact X and Y coordinates of the mouse cursor at the instant that particular mouse event occurred. Thus for each supported mouse based user interaction technique supported by the run time engine, a two-dimensional loop exists (web page number by object number) in which the current bounding rectangle for every object on a given web page is being compared to the current mouse cursor location at all times. The bounding rectangle is simply the current X and Y origin coordinates of an object, extended by its current width and height. In this way, the run time engine is informed if the current mouse cursor location falls within one or more bounding rectangles. Parenthetically, the run time engine also knows when the current mouse cursor falls outside the bounding rectangle of a given object.

Based on the type of mouse user interaction at **252**, the run time engine employs different techniques and executes different methods. If the viewer moves the mouse at **253**, the mouseMove method informs the run time engine immediately of this event and the current mouse cursor coordinates.

If this user mouse move action caused the mouse to move into one or more bounding rectangles of any text button or image object(s) at **254**, or out of one or more bounding rectangles of any text button or image object(s) at **255**, then appropriate two-dimensional status variables are set and the draw system is called. The draw system interprets the relevant two-dimensional variables for the existing text button and image object(s) on the current web page in its draw loop as described above with reference to FIG. **16**, and draws the correct backgrounds, text strings, images, and 3D frames based on the state of each object, as just set by the "mouseMove" computational two-dimensional loop. If the mouse has entered into or out of the bounding rectangles of any image and/or text button object that has a defined text button, image and/or video pop-up object (See FIG. **16**), then the draw system paints or effectively erases the appropriate background, text string, images and/or 3D frame for these pop-up objects. If the location of any of these text button and image objects or child pop-up objects is changing over time because of time line, animation, or transformation threads, the draw system, as described in association with FIG. **33** and FIG. **34**, is aware of these dynamics, and redraws the screen as these real time events occur. If any sound or video events were defined (See FIG. **16**) for any text button or image objects, then the run time engine plays those sound and/or video files or channels as defined. As multiple objects can be defined that each have associated sound (and even video) files, and these objects can be overlaid on each other, either completely or partially, very interesting synchronized multiple sound tracks can be constructed, in which certain designated sounds are played, or stopped, based purely on user mouse movement.

If the viewer moves the mouse into or out of the bounding rectangle for a paragraph hot link at **256**, then appropriate four-dimensional status variables (web page by paragraph number, by paragraph line number by paragraph line segment number) are set or reset and the draw system is called. The draw system paints the background color behind the hot link text string, and the color for the hot link text string in the hot link "mouse over" or the hot link normal colors. The text string may be underlined or in a bold font, depending on the settings.

If the viewer presses a mouse button at **257**, the mouse-down method informs the run time engine immediately of this event and the current mouse cursor coordinates. If the viewer releases the mouse at **259**, the mouse-up method informs the run time engine immediately of this event. Either way, if the original "mouse down" event had occurred inside one or more bounding rectangles of any text button or image object(s) at **258**, then appropriate two-dimensional status variables are set and the draw system is called. The "mouse up" event will cause the run time engine to reset those appropriate two-dimensional status variables, and then call the draw system. If the mouse was inside of the bounding rectangles of any image and/or text button object that had a defined text button, image and/or video pop-up object (See FIG. **16**) with the freeze attribute, the appropriate two-dimensional status variables are set so that the draw system will not erase these pop-up objects when the mouse moves outside the bounding rectangle(s) of the parent text button or image objects. The draw system interprets the relevant two-dimensional variables for the existing text button and image object(s) on the current web page in its draw loop described above with reference to FIG. **16**, and draws the correct backgrounds, text strings, images, and 3D frames based on the state of each object, as just set by the "mouseDown" computational two-dimensional loop.

If the 3D frame had been previously defined (See FIG. **16**) to have the "live" setting, then the 3D effect is changed from a "raised" effect to a "depressed" effect. If the location of any of these text button and image objects or child pop-up objects is changing over time because of time line, animation, or transformation threads, the draw system, as described in association with FIG. **33** and FIG. **34**, is aware of these dynamics, and redraws the screen as these real time events occur, but does not recognize any new "mouse down" or "mouse up" Events. In one implementation, only "mouse over" events are recognized dynamically by the draw system.

If any sound or video events were defined (See FIG. **16**) for any text button or image objects, then the run time engine plays those sound and/or video files or channels as defined. As multiple objects can be defined that each have associated sound (and even video) files, and these objects can be overlaid on each other, either completely or partially, very interesting synchronized multiple sound tracks can be constructed, in which certain designated sounds are played, or stopped, based purely on the user pressing a mouse button.

If a mouse down event as described previously in association with FIG. **16** was defined for one or more effected text button or image objects, only the object that was drawn last will have its event processed. If the event was to go to a different internal web page, then the run time engine sets the web page counter described in association with FIG. **31** for the "current" page to be one less than that of the desired web page. The current web page's object time lines, if any, complete normally, and the desired web page is then executed. If the "mouse down" event was to go to an external web page in a different window, then the run time engine creates a new browser window. In JAVA this can be accomplished with the "getAppletContext( ).showDocument(theURL, "_blank")" method, where "theURL" is the URL address for the external web page. The run time engine, however, continues executing.

If the mouse down event was to go to an external web page in the same browser window, then the run time engine terminates by turning control over to the designated external web page. In JAVA this can be accomplished with the "getAppletContext( ).showDocument(theURL)" method, where "theURL" is the URL address for the external web page.

If the viewer presses a mouse button while inside the bounding rectangle for a paragraph hot link at **260**, then appropriate four-dimensional status variables (web page by paragraph number, by paragraph line number, by paragraph line segment number) are set. If the hot link setting at FIG. **16** was to go to a different internal web page, then the run time engine sets the web page counter described previously in association with FIG. **31** for the "current" page to be one less than that of the desired web page. The current web page's object time lines, if any, complete normally, and the desired web page is then executed. If the hot link setting was to go to an external web page in a different window, then the run time engine creates a new browser window and loads the external web page. The run time engine, however, continues executing. If the hot link setting was to go to an external web page in the same browser window, then the run time engine terminates by turning control over to the designated external web page. This completes the detailed description of the run time process at **261**.

As is obvious from the descriptions of the various features and processes described above, it will be apparent to one skilled in the art that variations in form and detail may be made in the preferred implementation and methods without varying from the spirit and scope of the invention as defined in the claims or as interpreted under the doctrine of equivalents. It should also be clear that terms such as "browser", "mouse", "server", "web", etc., while adequate to describe the current state of interactive systems such as the World Wide Web, may evolve into new and more powerful entities. This evolution of terminology and technology is independent of the preferred implementation and methods of the invention as defined in the claims or as interpreted under the doctrine of equivalents. The preferred implementation and methods are thus provided for purposes of explanation and illustration, but not limitation.

I claim:

1. A system for assembling a web site comprising:

a server comprising a build engine configured to:

accept user input to create a web site, the web site comprising a plurality of web pages, each web page comprising a plurality of objects,

accept user input to associate a style with objects of the plurality of web pages, wherein each web page comprises at least one button object or at least one image object, and wherein the at least one button object or at least one image object is associated with a style that includes values defining transformations and time lines for the at least one button object or at least one image object; and wherein each web page is defined entirely by each of the plurality of objects comprising that web page and the style associated with the object,

produce a database with a multidimensional array comprising the objects that comprise the web site including data defining, for each object, the object style, an object number, and an indication of the web page that each object is part of, and provide the database to a server accessible to web browser;

wherein the database is produced such that a web browser with access to a runtime engine is configured to generate the web-site from the objects and style data extracted from the provided database.

2. The system of claim **1**,

wherein one of said plurality of objects is a child, and

wherein the build engine is configured to accept user input to associate a style with child button and child image objects.

3. The system of claim **2**, wherein at least one of said styles includes values defining time lines for child button and child image objects.

4. The system of claim **1**, wherein at least one of said styles includes settings for multiple object states.

5. The system of claim **1**, further including file size reduction means for reducing the total size of files generated by said build engine to a size of between 12K and 50K.

6. The system of claim **1**, where said data is stored as one or more of a Boolean an integer, a string, a floating point variables, or a URL.

* * * * *

EXHIBIT C

US009063755B2

(12) **United States Patent**
Rempell et al.

(10) **Patent No.: US 9,063,755 B2**
(45) **Date of Patent: Jun. 23, 2015**

(54) **SYSTEMS AND METHODS FOR PRESENTING INFORMATION ON MOBILE DEVICES**

(75) Inventors: **Steven H. Rempell**, Novato, CA (US); **David Chrobak**, Clayton, CA (US); **Ken Brown**, San Martin, CA (US)

(73) Assignee: **Express Mobile, inc.**, Novato, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 154 days.

(21) Appl. No.: **12/936,395**

(22) PCT Filed: **Apr. 6, 2009**

(86) PCT No.: **PCT/US2009/039695**

§ 371 (c)(1),
(2), (4) Date: **Nov. 3, 2010**

(87) PCT Pub. No.: **WO2009/126591**

PCT Pub. Date: **Oct. 15, 2009**

(65) **Prior Publication Data**

US 2011/0107227 A1 May 5, 2011

Related U.S. Application Data

(60) Provisional application No. 61/166,651, filed on Apr. 3, 2009, provisional application No. 61/113,471, filed on Nov. 11, 2008, provisional application No. 61/123,438, filed on Apr. 7, 2008.

(51) **Int. Cl.**

G06F 3/048	(2013.01)
G06F 9/44	(2006.01)
G06F 9/45	(2006.01)

(52) **U.S. Cl.**
CPC ***G06F 9/4443*** (2013.01)

(58) **Field of Classification Search**
CPC G06F 3/048
USPC 715/738
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2004/0055017	A1 *	3/2004	Delpuch et al.	725/110
2004/0163020	A1 *	8/2004	Sidman	714/100
2005/0149935	A1 *	7/2005	Benedetti	718/102
2005/0273705	A1 *	12/2005	McCain	715/513
2006/0063518	A1 *	3/2006	Paddon et al.	455/418

OTHER PUBLICATIONS

Stina Nylander et al. "The Ubiquitous Interactor-Device Independent Access to Mobile Services" (Computer-Aided Design for User Interfaces IV, Proceedings of the Fifth International Conference on Computer-Aided Design of User Interfaces CADUI'2004, Jan. 2004, pp. 271-282).*

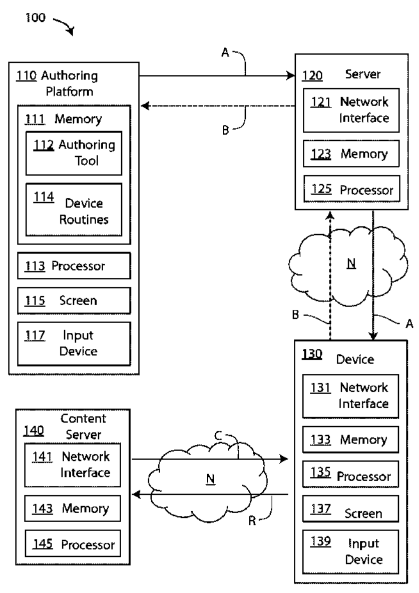
* cited by examiner

Primary Examiner — Jennifer To
Assistant Examiner — Xuyang Xia
(74) *Attorney, Agent, or Firm* — Steven R. Vosen

(57) **ABSTRACT**

Embodiments of a system and method are described for generating and distributing programming to mobile devices over a network. Devices are provided with Players specific to each device and Applications that are device independent. Embodiments include a full-featured WYSIWYG authoring environment, including the ability to bind web components to objects.

28 Claims, 18 Drawing Sheets



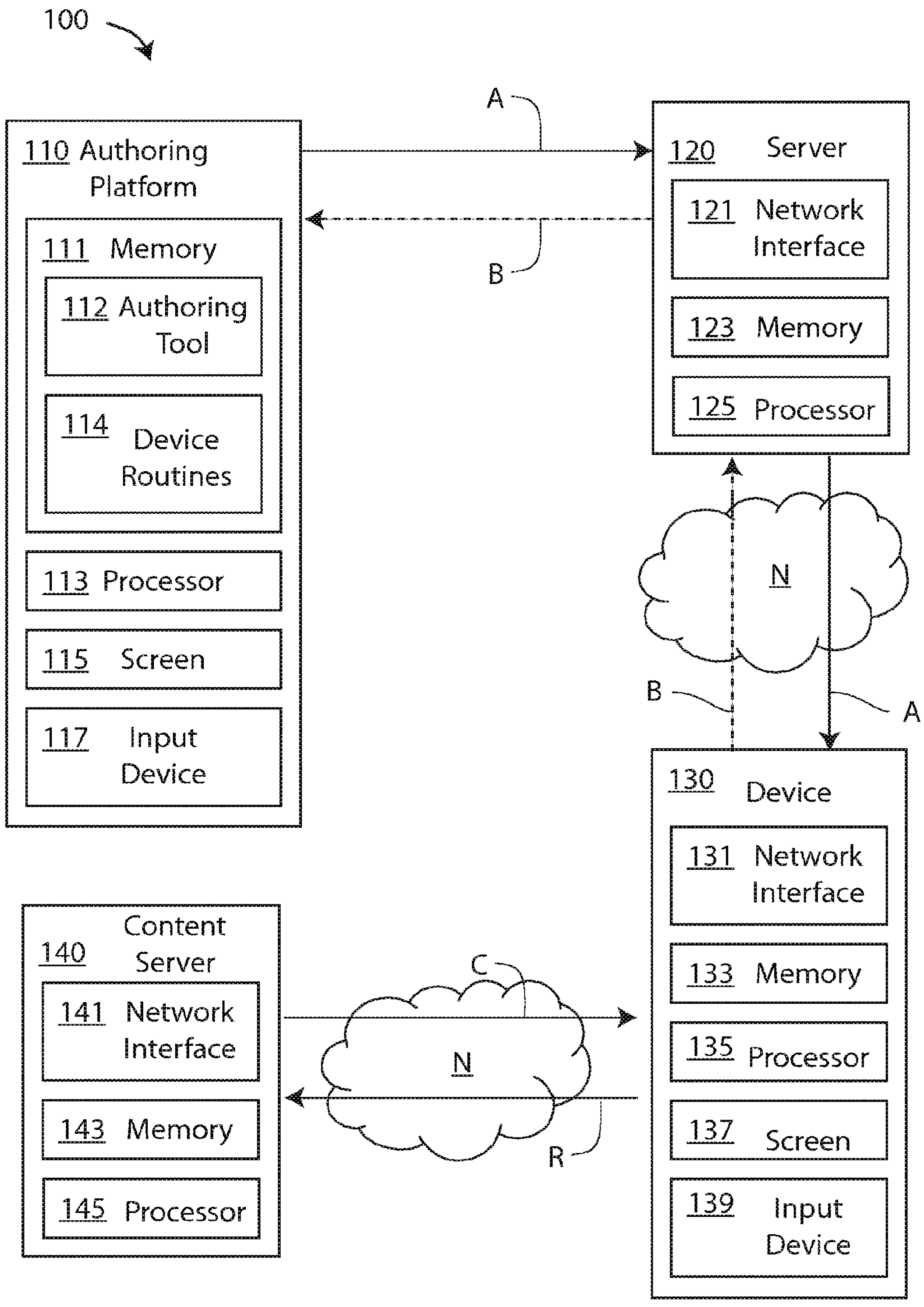

100
110 Authoring Platform
111 Memory
112 Authoring Tool
114 Device Routines
113 Processor
115 Screen
117 Input Device
A
B
120 Server
121 Network Interface
123 Memory
125 Processor
N
B
A
130 Device
131 Network Interface
133 Memory
135 Processor
137 Screen
139 Input Device
140 Content Server
141 Network Interface
143 Memory
145 Processor
C
N
R

FIG. 1A

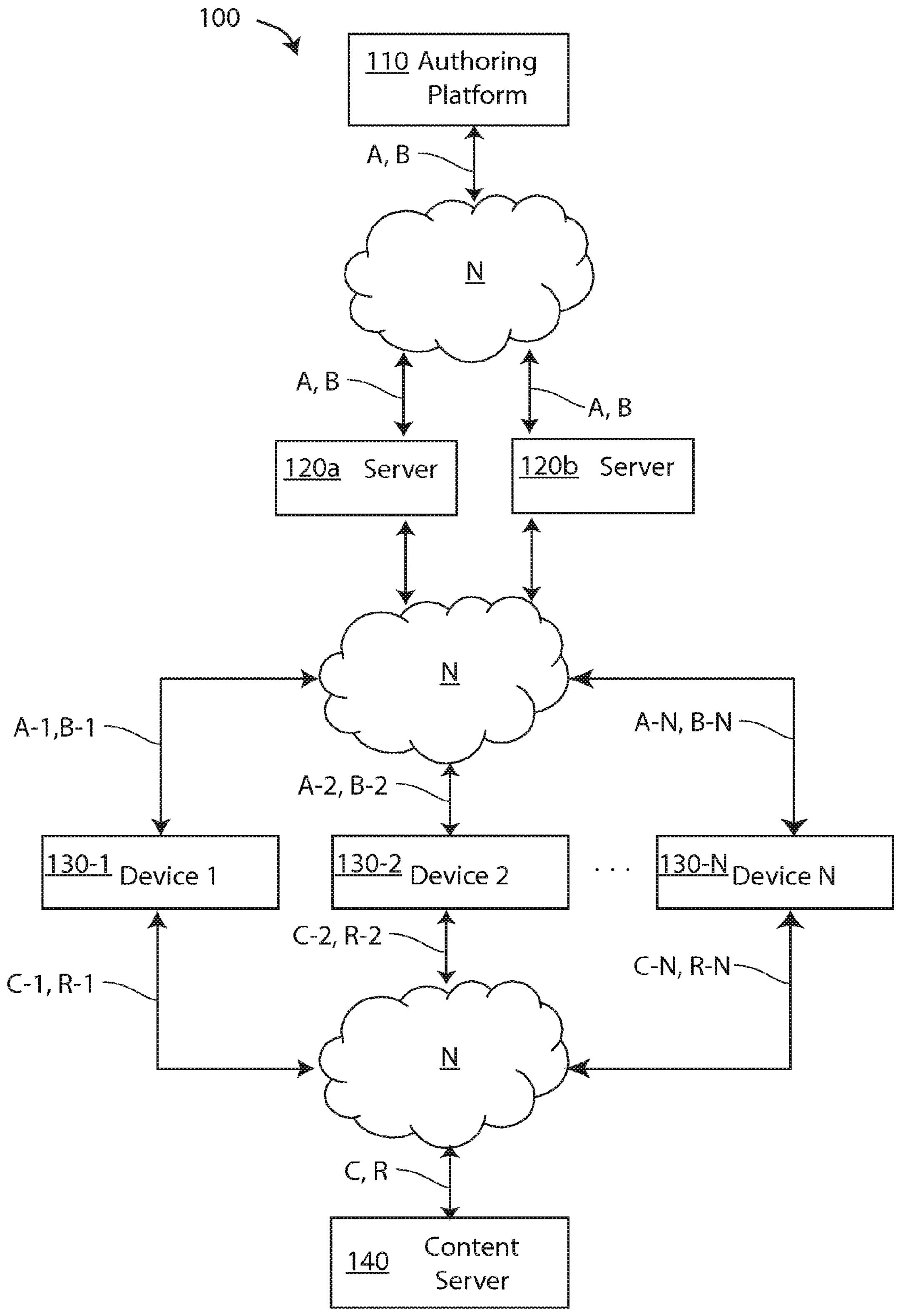

100
110 Authoring Platform
A, B
N
A, B
A, B
120a Server
120b Server
N
A-1,B-1
A-N, B-N
A-2, B-2
130-1 Device 1
130-2 Device 2
130-N Device N
C-2, R-2
C-1, R-1
C-N, R-N
N
C, R
140 Content Server

FIG. 1B

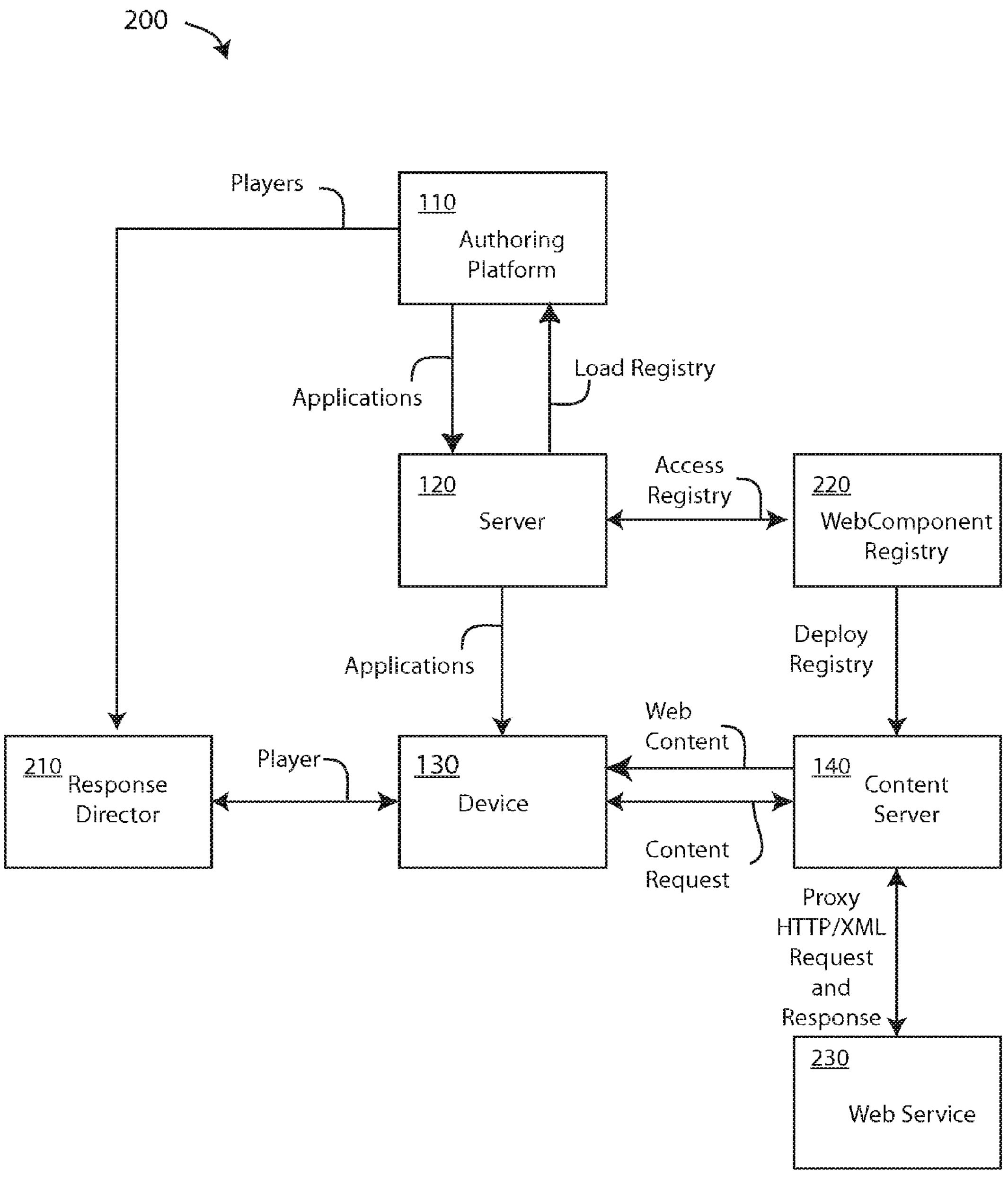

200
Players
110 Authoring Platform
Applications
Load Registry
120 Server
Access Registry
220 WebComponent Registry
Applications
Deploy Registry
Web Content
210 Response Director
Player
130 Device
140 Content Server
Content Request
Proxy HTTP/XML Request and Response
230 Web Service

FIG. 2A

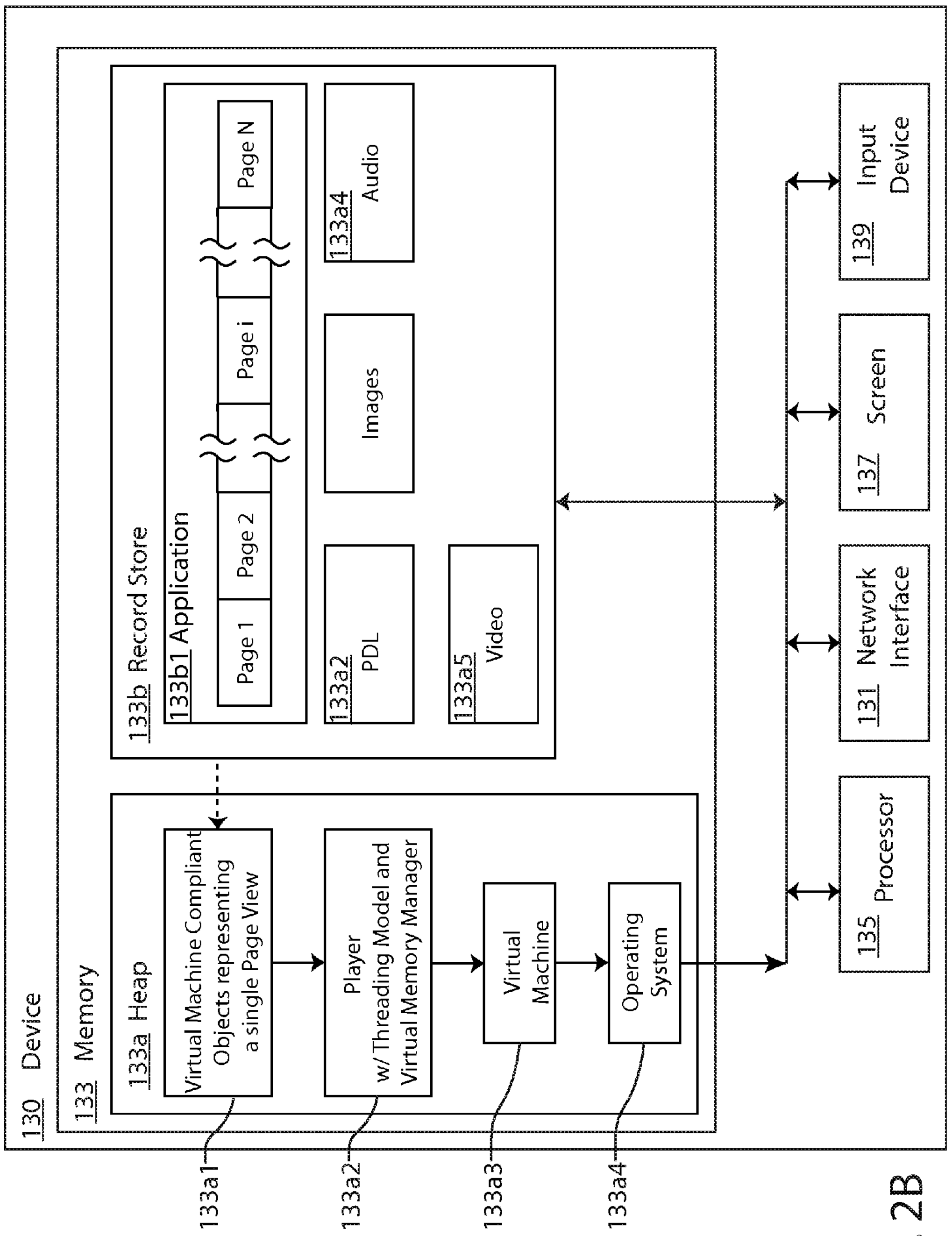

130 Device
133 Memory
133a Heap
Virtual Machine Compliant Objects representing a single Page View
Player w/ Threading Model and Virtual Memory Manager
Virtual Machine
Operating System
133b Record Store
133b1 Application
Page 1
Page 2
Page i
Page N
133a2 PDL
133a5 Video
Images
133a4 Audio
131 Network Interface
135 Processor
137 Screen
139 Input Device
133a1
133a2
133a3
133a4

FIG. 2B

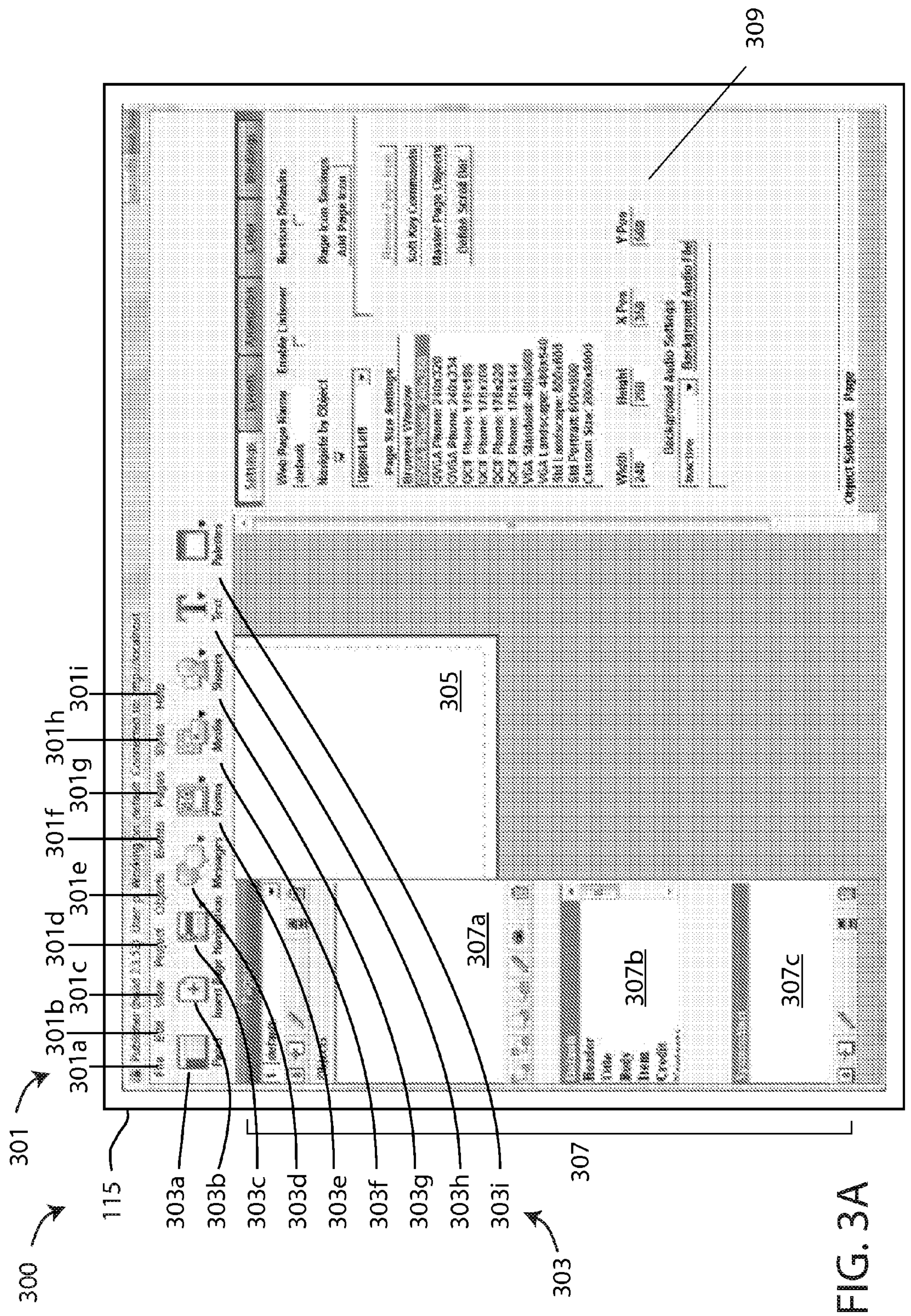

300
301
115
301a
301b
301c
301d
301e
301f
301g
301h
301i
303a
303b
303c
303d
303e
303f
303g
303h
303i
303
305
307
307a
307b
307c
309

FIG. 3A

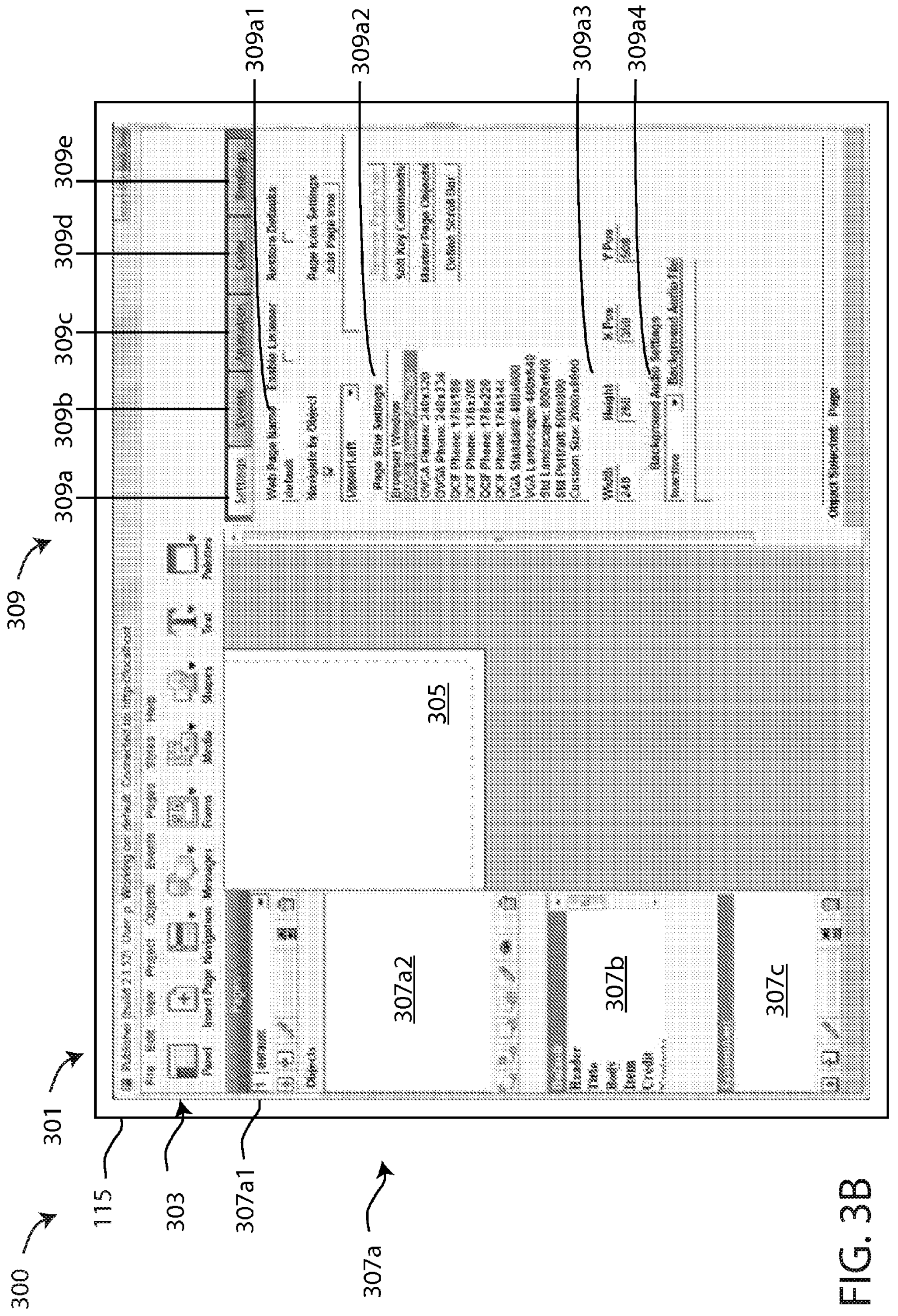
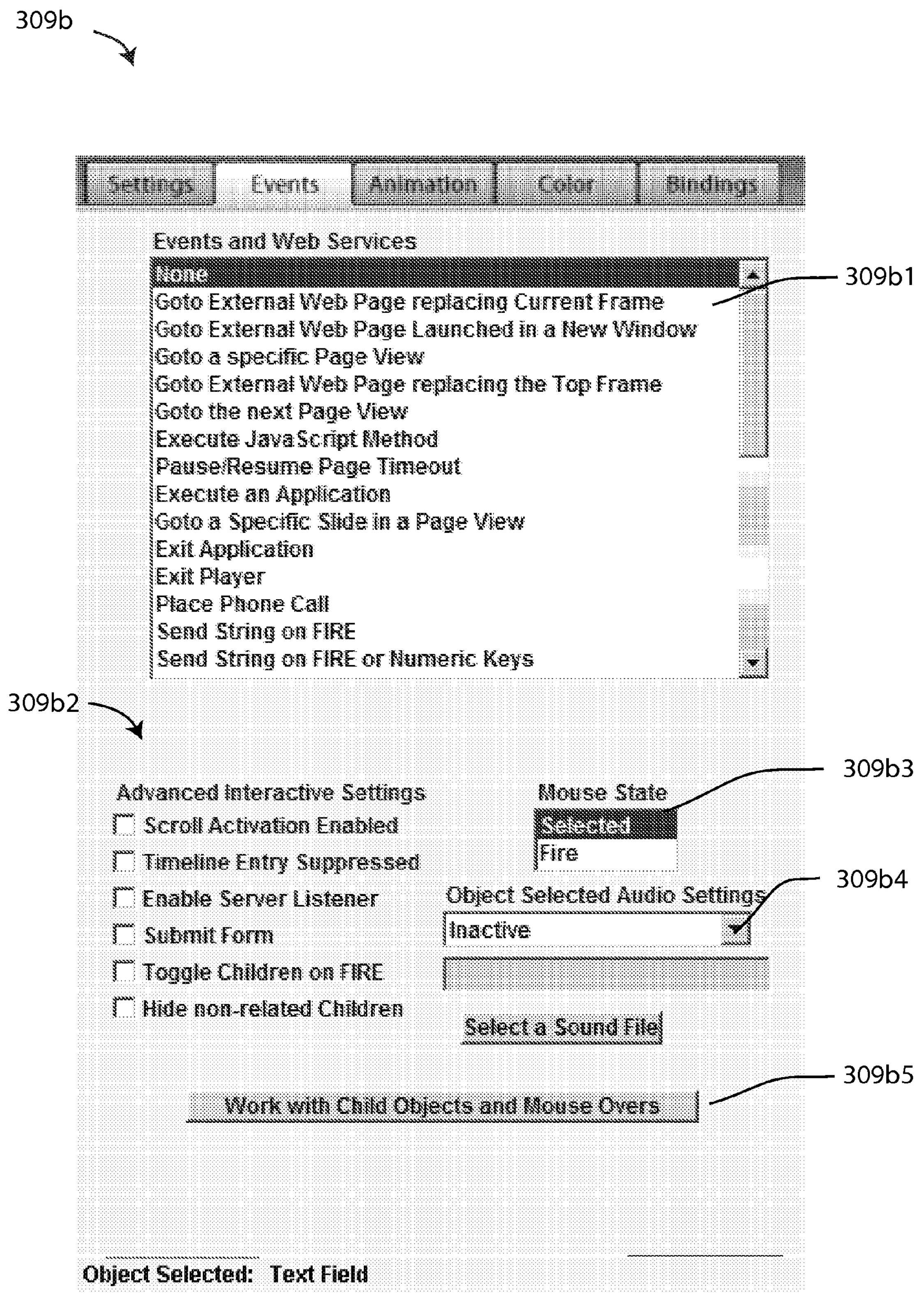

300
115
301
303
307a1
307a
307a2
307b
307c
305
309
309a
309b
309c
309d
309e
309a1
309a2
309a3
309a4

FIG. 3B



Events and Web Services

- None
- Goto External Web Page replacing Current Frame
- Goto External Web Page Launched in a New Window
- Goto a specific Page View
- Goto External Web Page replacing the Top Frame
- Goto the next Page View
- Execute JavaScript Method
- Pause/Resume Page Timeout
- Execute an Application
- Goto a Specific Slide in a Page View
- Exit Application
- Exit Player
- Place Phone Call
- Send String on FIRE
- Send String on FIRE or Numeric Keys

Advanced Interactive Settings

- [ ] Scroll Activation Enabled
- [ ] Timeline Entry Suppressed
- [ ] Enable Server Listener
- [ ] Submit Form
- [ ] Toggle Children on FIRE
- [ ] Hide non-related Children

Mouse State

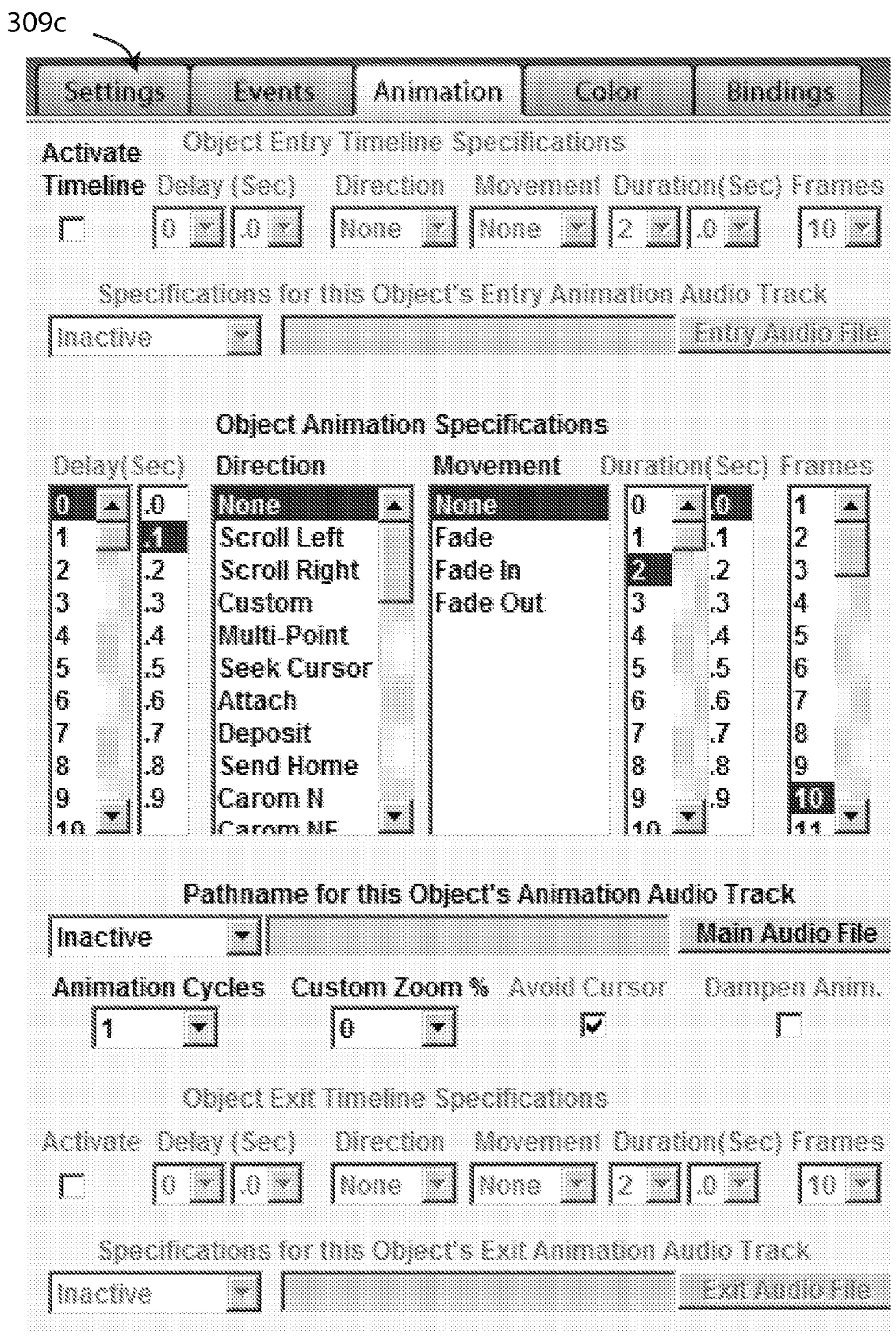
- Selected
- Fire

Object Selected Audio Settings

Inactive

Object Selected: Text Field

FIG. 3C



Object Entry Timeline Specifications

Activate Timeline	Delay (Sec)		Direction	Movement	Duration(Sec)		Frames
☐	0	.0	None	None	2	.0	10

Specifications for this Object's Entry Animation Audio Track

Inactive

Object Animation Specifications

Delay(Sec)		Direction	Movement	Duration(Sec)		Frames
0	.0	None	None	0	0	1
1	.1	Scroll Left	Fade	1	.1	2
2	.2	Scroll Right	Fade In	2	.2	3
3	.3	Custom	Fade Out	3	.3	4
4	.4	Multi-Point		4	.4	5
5	.5	Seek Cursor		5	.5	6
6	.6	Attach		6	.6	7
7	.7	Deposit		7	.7	8
8	.8	Send Home		8	.8	9
9	.9	Carom N		9	.9	10

Pathname for this Object's Animation Audio Track

Inactive

Animation Cycles	Custom Zoom %	Avoid Cursor	Dampen Anim.
1	0	☑	☐

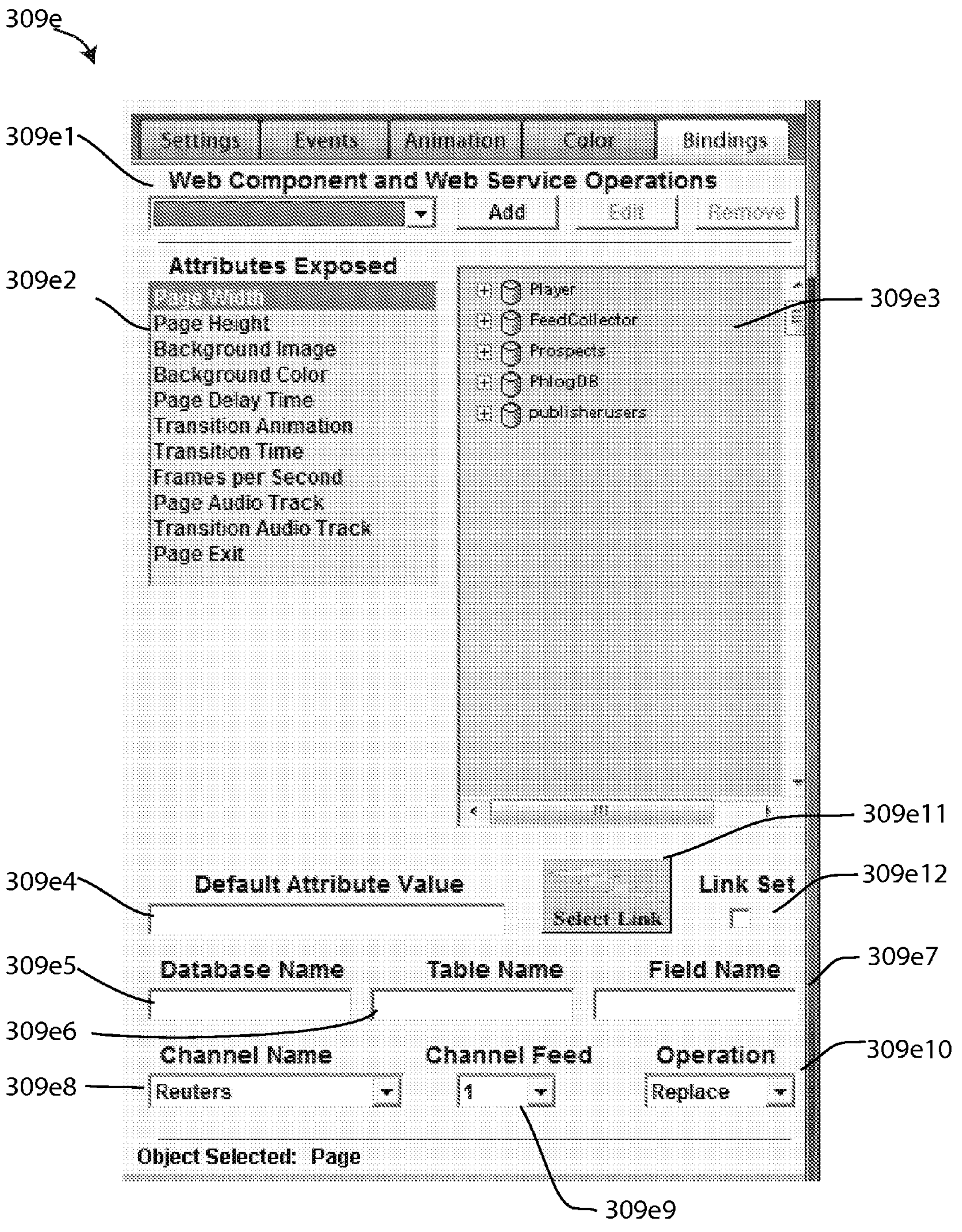
Object Exit Timeline Specifications

Activate	Delay (Sec)		Direction	Movement	Duration(Sec)		Frames
☐	0	.0	None	None	2	.0	10

Specifications for this Object's Exit Animation Audio Track

Inactive

FIG. 3D



Web Component and Web Service Operations

Attributes Exposed

Page Width
Page Height
Background Image
Background Color
Page Delay Time
Transition Animation
Transition Time
Frames per Second
Page Audio Track
Transition Audio Track
Page Exit

Player
FeedCollector
Prospects
PhlogDB
publisherusers

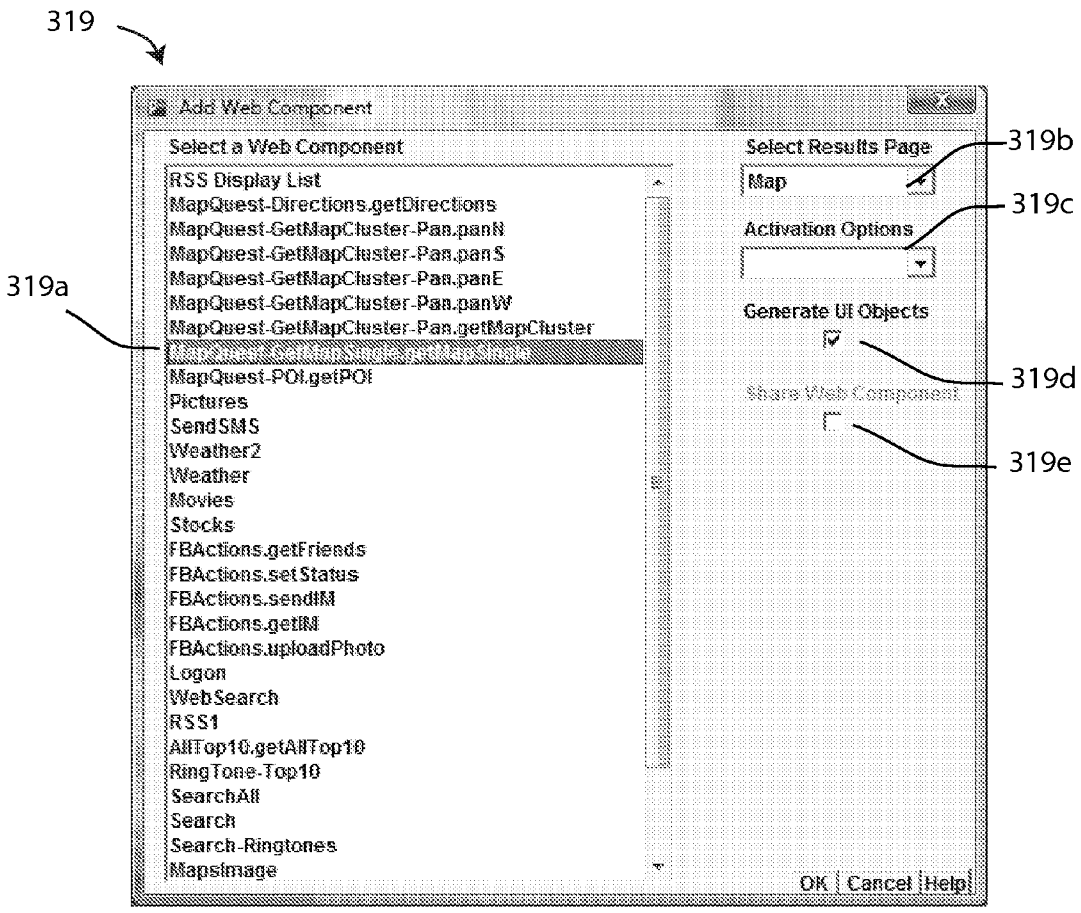
Default Attribute Value:

Database Name:
Table Name:
Field Name:

Channel Name: Reuters
Channel Feed: 1
Operation: Replace

Object Selected: Page

FIG. 3E



Add Web Component

Select a Web Component

RSS Display List
MapQuest-Directions.getDirections
MapQuest-GetMapCluster-Pan.panN
MapQuest-GetMapCluster-Pan.panS
MapQuest-GetMapCluster-Pan.panE
MapQuest-GetMapCluster-Pan.panW
MapQuest-GetMapCluster-Pan.getMapCluster
MapQuest-GetMapSingle.getMapSingle
MapQuest-POI.getPOI
Pictures
SendSMS
Weather2
Weather
Movies
Stocks
FBActions.getFriends
FBActions.setStatus
FBActions.sendIM
FBActions.getIM
FBActions.uploadPhoto
Logon
WebSearch
RSS1
AllTop10.getAllTop10
RingTone-Top10
SearchAll
Search
Search-Ringtones
MapsImage

Select Results Page: Map

Activation Options:

Generate UI Objects: ✓

Share Web Component: ☐

OK | Cancel | Help

FIG. 3F

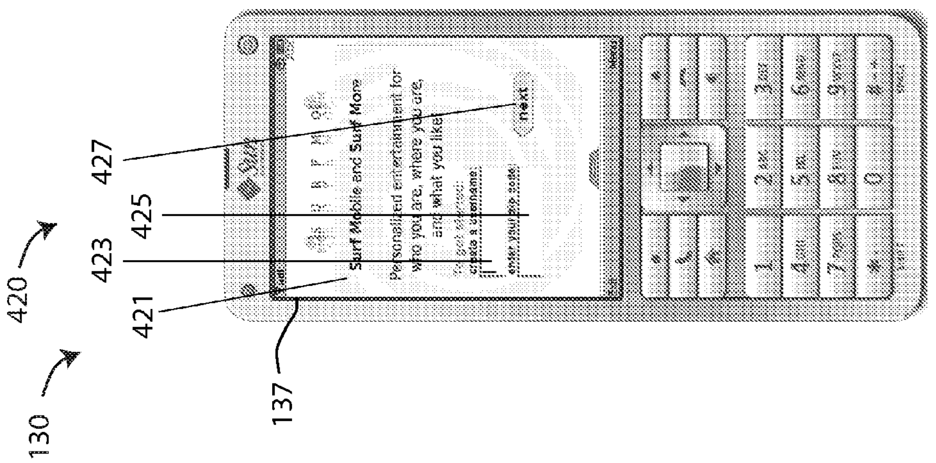

420
427
425
423
421
130
137
Surf Mobile and Surf More
Personalized entertainment for who you are, where you are, and what you like!
To get started:
create a username:
enter your zip code:
next

FIG. 4B

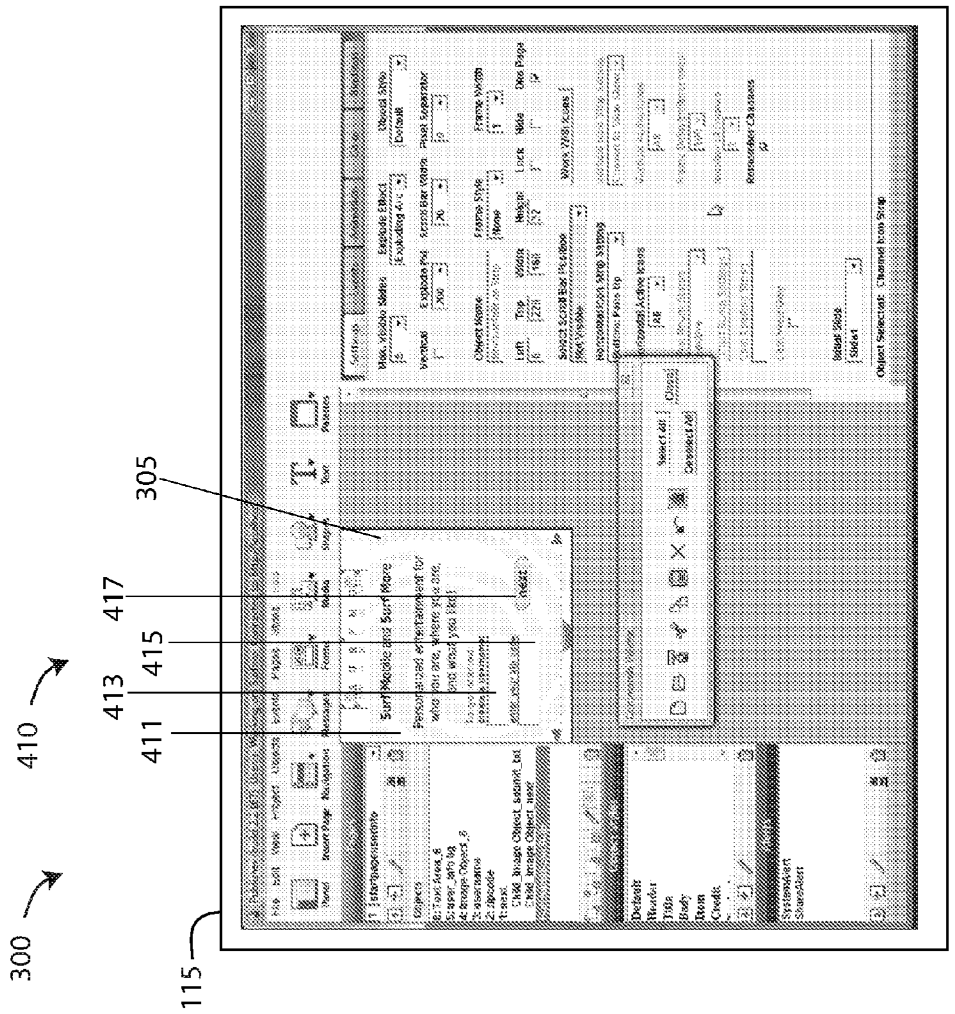

300
410
411
413
415
417
305
115

FIG. 4A

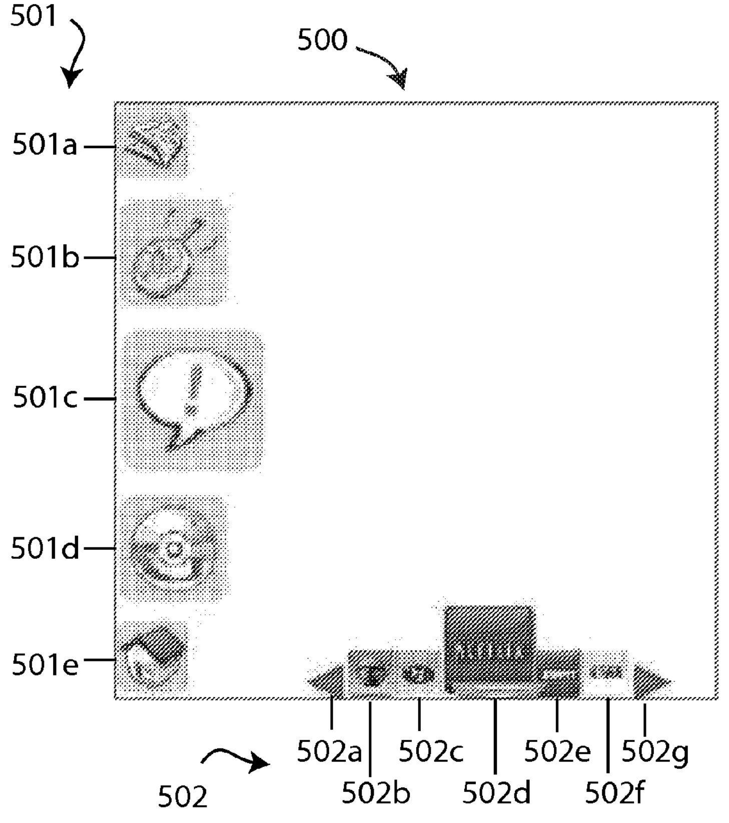
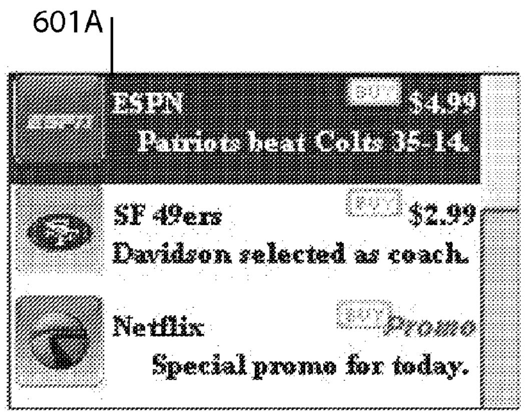
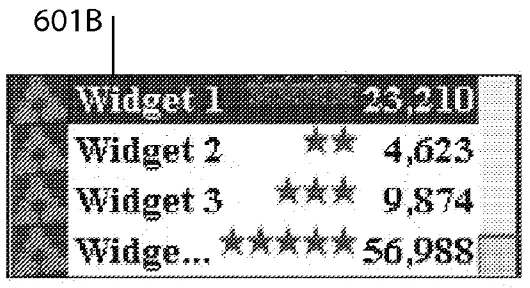


FIG. 5



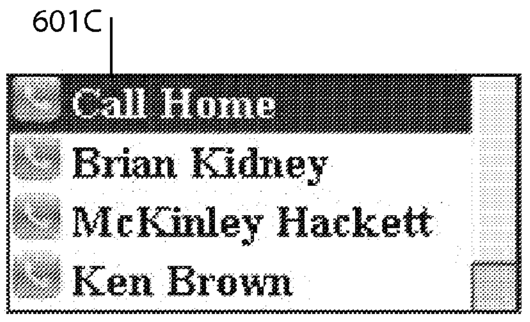
ESPN $4.99
Patriots beat Colts 35-14.
SF 49ers $2.99
Davidson selected as coach.
Netflix Promo
Special promo for today.

FIG. 6A



Widget 1 23,210
Widget 2 ★★ 4,623
Widget 3 ★★★ 9,874
Widge... ★★★★★ 56,988

FIG. 6B



Call Home
Brian Kidney
McKinley Hackett
Ken Brown

FIG. 6C

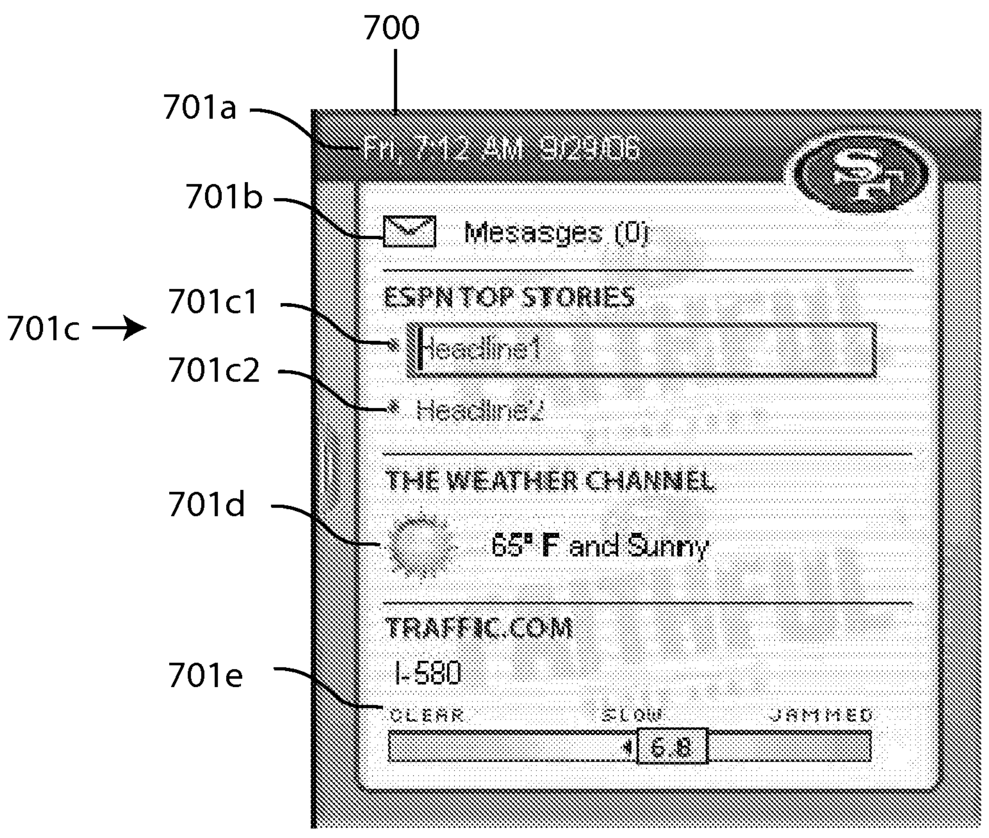

700
701a
701b
701c
701c1
701c2
701d
701e
Fri 7:12 AM 9/29/06
SF
Mesasges (0)
ESPN TOP STORIES
Headline1
Headline2
THE WEATHER CHANNEL
65° F and Sunny
TRAFFIC.COM
I-580
CLEAR
SLOW
JAMMED
6.8

FIG. 7

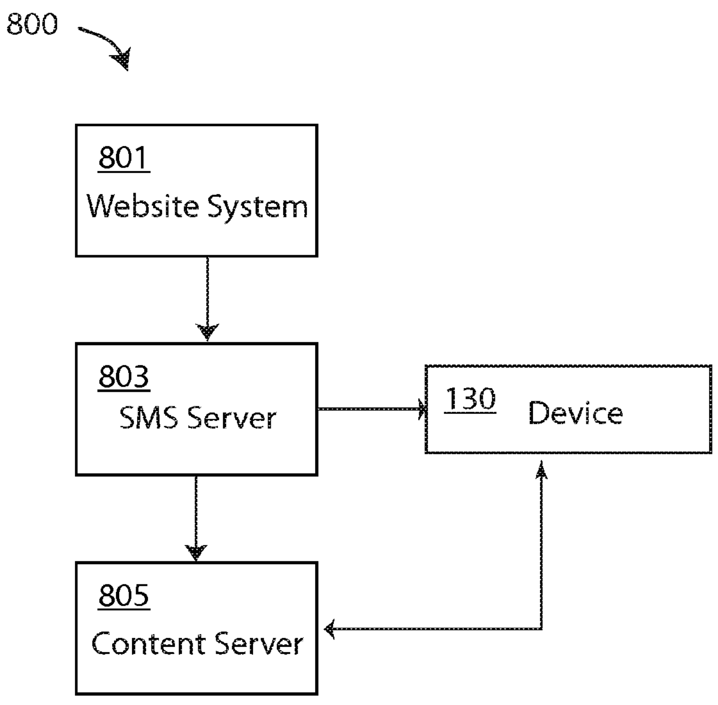

800
801 Website System
803 SMS Server
130 Device
805 Content Server

FIG. 8

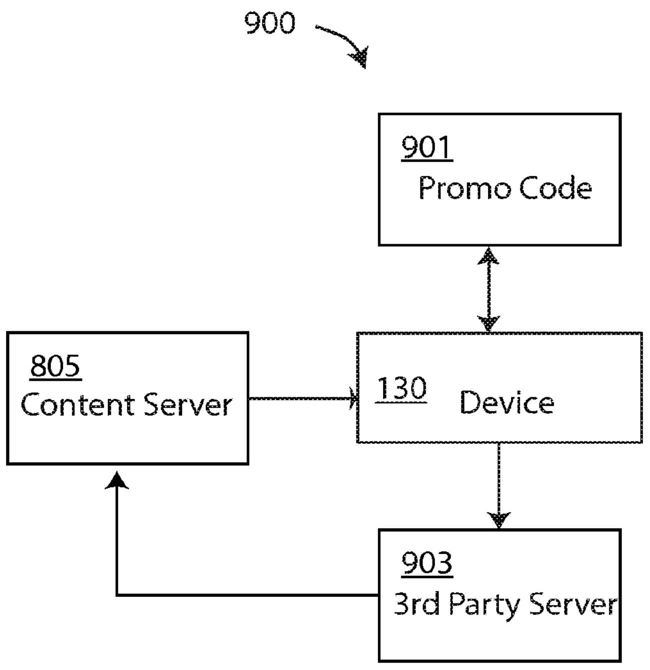

900
901 Promo Code
805 Content Server
130 Device
903 3rd Party Server

FIG. 9

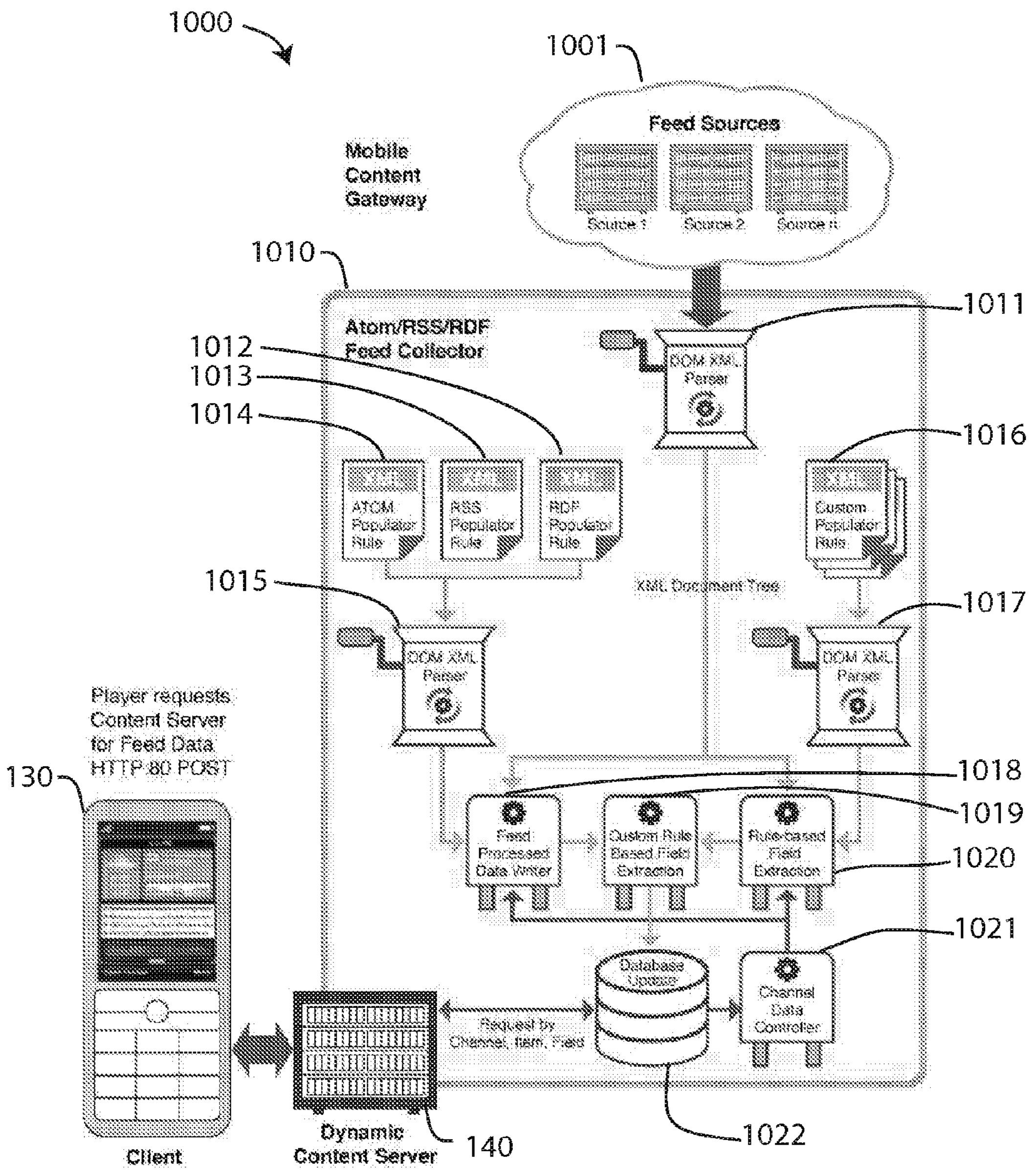
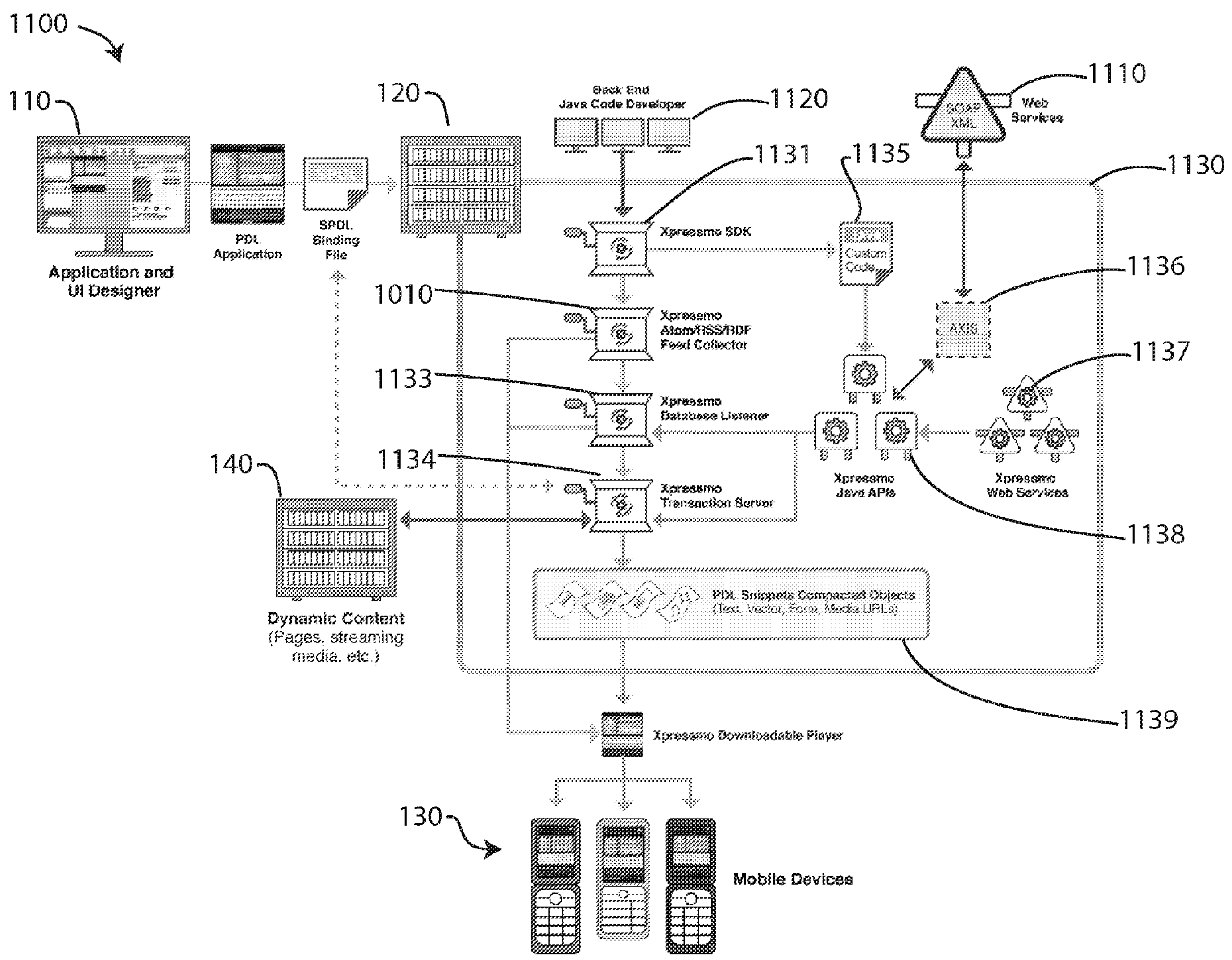

1000
1001
Feed Sources
Source 1
Source 2
Source n
Mobile Content Gateway
1010
Atom/RSS/RDF Feed Collector
1011
DOM XML Parser
1012
1013
1014
ATOM Populator Rule
RSS Populator Rule
RDF Populator Rule
1016
Custom Populator Rule
XML Document Tree
1015
DOM XML Parser
1017
DOM XML Parser
Player requests Content Server for Feed Data HTTP 80 POST
130
1018
1019
Feed Processed Data Writer
Custom Rule Based Field Extraction
Rule-based Field Extraction
1020
1021
Database Update
Channel Data Controller
Request by Channel, Item, Field
Client
Dynamic Content Server
140
1022

FIG. 10



1100
110
Application and UI Designer
PDL Application
SPDL Binding File
120
Back End Java Code Developer
1120
1110
SOAP XML
Web Services
1131
1135
1130
Xpressmo SDK
Custom Code
1136
AXIS
1010
Xpressmo Atom/RSS/RDF Feed Collector
1137
1133
Xpressmo Database Listener
Xpressmo Java APIs
Xpressmo Web Services
140
1134
Xpressmo Transaction Server
1138
Dynamic Content (Pages, streaming media, etc.)
PDL Snippets Compacted Objects (Text, Vector, Form, Media URLs)
1139
Xpressmo Downloadable Player
130
Mobile Devices

FIG. 11

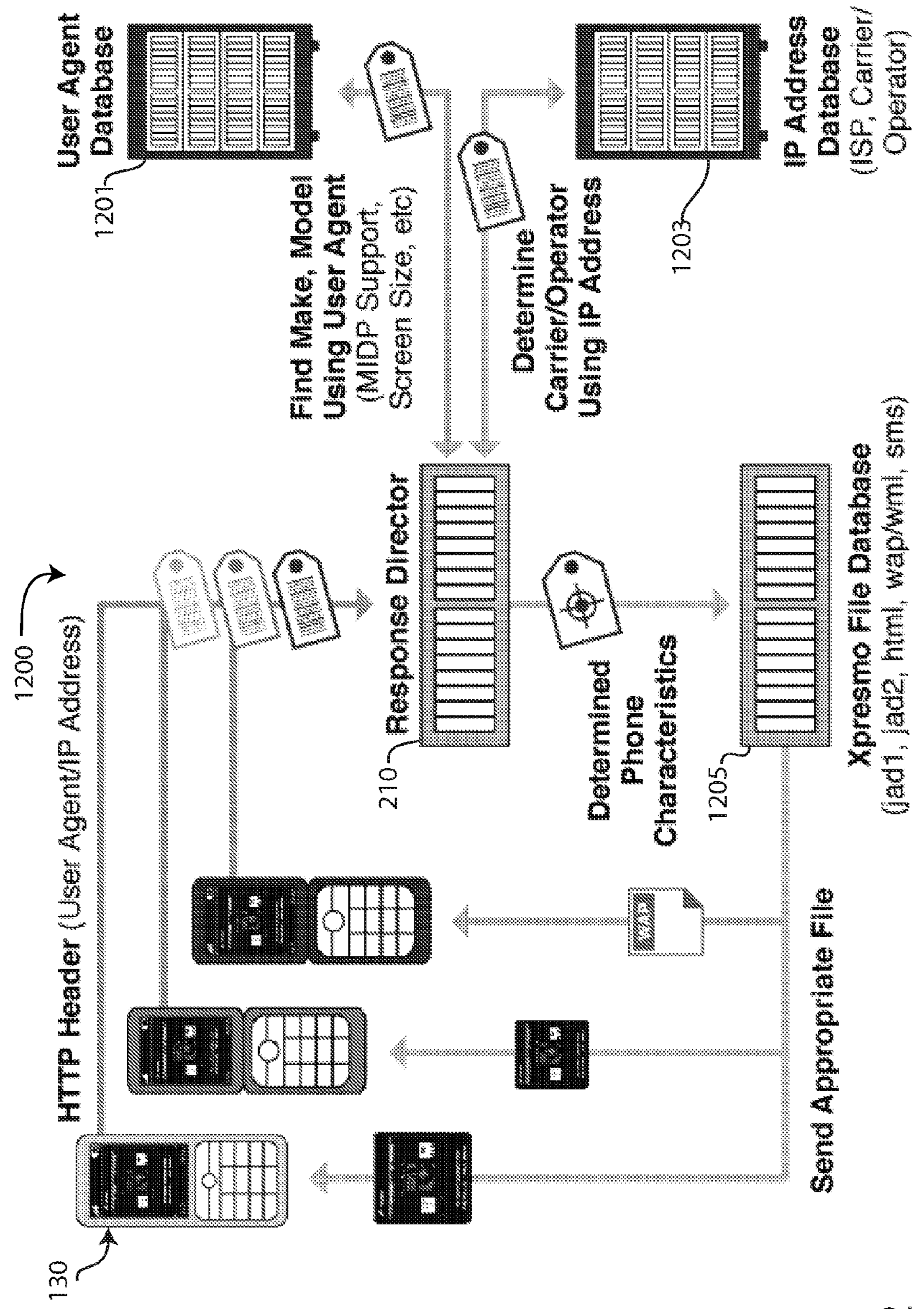

1200
User Agent Database
1201
Find Make, Model Using User Agent (MIDP Support, Screen Size, etc)
Determine Carrier/Operator Using IP Address
IP Address Database (ISP, Carrier/ Operator)
1203
HTTP Header (User Agent/IP Address)
Response Director
210
Determined Phone Characteristics
Xpresmo File Database (jad1, jad2, html, wap/wml, sms)
1205
Send Appropriate File
130

FIG. 12

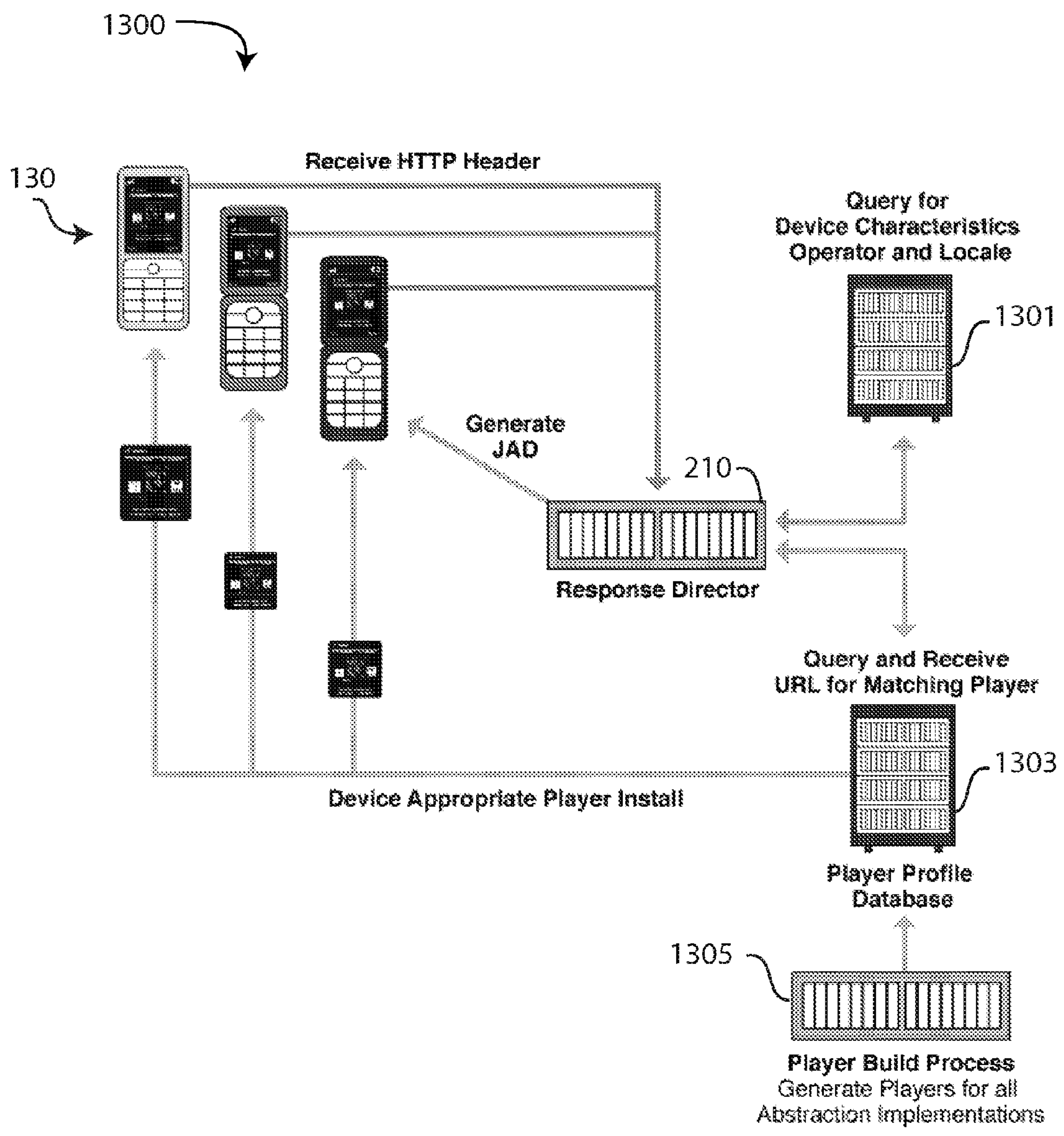

1300
130
Receive HTTP Header
Query for
Device Characteristics
Operator and Locale
1301
Generate
JAD
210
Response Director
Query and Receive
URL for Matching Player
1303
Device Appropriate Player Install
Player Profile
Database
1305
Player Build Process
Generate Players for all
Abstraction Implementations

FIG. 13

SYSTEMS AND METHODS FOR PRESENTING INFORMATION ON MOBILE DEVICES

TECHNICAL FIELD

The present invention generally relates to providing software for mobile devices, and more particularly to a method and system for authoring Applications for devices.

BACKGROUND ART

Internet-connected mobile devices are becoming ever more popular. While these devices provide portability to the Internet, they generally do not have the capabilities of non-mobile devices including computing, input and output capabilities.

In addition, the mobility of the user while using such devices provides challenges and opportunities for the use of the Internet. Further, unlike non-mobile devices, there are a large number of types of devices and they tend to have a shorter lifetime in the marketplace. The programming of the myriad of mobile devices is a time-consuming and expensive proposition, thus limiting the ability of service providers to update the capabilities of mobile devices.

Thus there is a need in the art for a method and apparatus that permits for the efficient programming of mobile devices. Such a method and apparatus should be easy to use and provide output for a variety of devices.

DISCLOSURE OF INVENTION

In certain embodiments, a system is provided to generate code to provide content on a display of a platform. The system includes a database of web services obtainable over a network and an authoring tool. The authoring tool is configured to define an object for presentation on the display, select a component of a web service included in said database, associate said object with said selected component, and produce code that, when executed on the platform, provides said selected component on the display of the platform.

In certain other embodiments, a method is provided for providing information to platforms on a network. The method includes accepting a first code over the network, where said first code is platform-dependent; providing a second code over the network, where said second code is platform-independent; and executing said first code and said second code on the platform to provide web components obtained over the network.

In certain embodiments, a method for displaying content on a platform utilizing a database of web services obtainable over a network is provided. The method includes: defining an object for presentation on the display; selecting a component of a web service included in said database; associating said object with said selected component; and producing code that, when executed on the platform, provides said selected component on the display of the platform.

In one embodiment, one of the codes is a Player, which is a thin client architecture that operates in a language that manages resources efficiently, is extensible, supports a robust application model, and has no device specific dependencies. In another embodiment, Player P is light weight and extends the operating system and/or virtual machine of the device to: Manage all applications and application upgrades, and resolve device, operating system, VM and language fragmentation.

In another embodiment, one of the codes is an Application that is a device independent code that interpreted by the Player.

These features together with the various ancillary provisions and features which will become apparent to those skilled in the art from the following detailed description, are attained by the system and method of the present invention, preferred embodiments thereof being shown with reference to the accompanying drawings, by way of example only, wherein:

BRIEF DESCRIPTION OF DRAWINGS

FIG. **1**A is an illustrative schematic of one embodiment of a system including an authoring platform and a server for providing programming instructions to a device over a network;

FIG. **1**B is schematic of an alternative embodiment system for providing programming instructions to device over a network;

FIG. **2**A is a schematic of an embodiment of system illustrating the communications between different system components;

FIG. **2**B is a schematic of one embodiment of a device illustrating an embodiment of the programming generated by authoring platform;

FIGS. **3**A and **3**B illustrate one embodiment of a publisher interface as it appears, for example and without limitation, on a screen while executing an authoring tool;

FIG. **3**C illustrates an embodiment of the Events Tab'

FIG. **3**D illustrates one embodiment of an Animation Tab;

FIG. **3**E illustrates one embodiment of Bindings Tab;

FIG. **3**F illustrates one embodiment of a pop-up menu for adding web components;

FIG. **4**A shows a publisher interface having a layout on a canvas; and FIG. **4**B shows a device having the resulting layout on a device screen;

FIG. **5** shows a display of launch strips;

FIG. **6**A is a display of a Channel Selection List;

FIG. **6**B is a display of a Widget Selection List;

FIG. **6**C is a display of a Phone List;

FIG. **7** shows a display of a mash-up;

FIG. **8** is a schematic of an embodiment of a push capable system;

FIG. **9** is a schematic of an alternative embodiment of a push capable system;

FIG. **10** is a schematic of one embodiment of a feed collector;

FIG. **11** is a schematic of an embodiment of a Mobile Content Gateway;

FIG. **12** is a schematic of one embodiment of a system that includes a response director, a user agent database, an IP address database, and a file database; and

FIG. **13** is a schematic of another embodiment of a system that includes a response director, a user agent database, an IP address database, and a file database.

Reference symbols are used in the Figures to indicate certain components, aspects or features shown therein, with reference symbols common to more than one Figure indicating like components, aspects or features shown therein.

MODE(S) FOR CARRYING OUT THE INVENTION

FIG. **1**A is an illustrative schematic of one embodiment of a system **100** including an authoring platform **110** and a server **120** for providing programming instructions to a

device **130** over a network N. In one embodiment, device **130** is a wireless device, and network N includes wireless communication to the device. Alternatively, system **100** may provide access over network N to other information, data, or content, such as obtainable as a web service over the Internet. In general, a user of authoring platform **110** may produce programming instructions or files that may be transmitted over network N to operate device **130**, including instructions or files that are sent to device **130** and/or server **120**. The result of the authoring process is also referred to herein, and without limitation, as publishing an Application.

Embodiments include one or more databases that store information related to one or more devices **130** and/or the content provided to the devices. It is understood that such databases may reside on any computer or computer system on network N, and that, in particular, the location is not limited to any particular server, for example.

Device **130** may be, for example and without limitation, a cellular telephone or a portable digital assistant, includes a network interface **131**, a memory **133**, a processor **135**, a screen **137**, and an input device **139**. Network interface **131** is used by device **130** to communication over a wireless network, such as a cellular telephone network, a WiFi network or a WiMax network, and then to other telephones through a public switched telephone network (PSTN) or to a satellite, or over the Internet. Memory **133** includes programming required to operate device **130** (such as an operating system or virtual machine instructions), and may include portions that store information or programming instructions obtained over network interface **131**, or that are input by the user (such as telephone numbers or images from a device camera (not shown). In one embodiment screen **137** is a touch screen, providing the functions of the screen and input device **139**.

Authoring platform **110** includes a computer or computer system having a memory **111**, a processor **113**, a screen **115**, and an input device **117**. It is to be understood that memory **111**, processor **113**, screen **115**, and input device **117** are configured such a program stored in the memory may be executed by the processor to accept input from the input device and display information on the screen. Further, the program stored in memory **111** may also instruct authoring platform **110** to provide programming or information, as indicated by the line labeled "A" and to receive information, as indicated by the line labeled "B."

Memory **111** is shown schematically as including a stored program referred to herein, and without limitation, as an authoring tool **112**. In one embodiment, authoring tool **112** is a graphical system for designing the layout of features as a display that is to appear on screen **137**. One example of authoring tool **112** is the CDER™ publishing platform (Express Mobile, Inc., Novato, Calif.).

In another embodiment, which is not meant to limit the scope of the present invention, device **130** may include an operating system having a platform that can interpret certain routines. Memory **111** may optionally include programming referred to herein, and without limitation, as routines **114** that are executable on device **130**.

Routines **114** may include device-specific routines—that is, codes that are specific to the operating system, programming language, or platform of specific devices **130**, and may include, but are not limited to, Java, Windows Mobile, Brew, Symbian OS, or Open Handset Alliance (OHA). Several examples and embodiments herein are described with reference to the use of Java. It is to be understood that the invention is not so limited, except as provided in the claims, and that one skilled in the art could provide Players for devices using routines provided on a platform. Thus as an example, routines **114** may include Java API's and an authoring tool System Development Kit (SDK) for specific devices **130**.

Server **120** is a computer or computer system that includes a network interface **121**, a memory **123**. and a processor **125**. Is to be understood that network interface **121**, memory **123**, and processor **125** are configured such that a program stored in the memory may be executed by the processor to: accept input and/or provide output to authoring platform **110**; accept input and/or provide output through network interface **121** over network N to network interface **131**; or store information from authoring platform **110** or from device **130** for transmission to another device or system at a later time.

In one embodiment, authoring platform **110** permits a user to design desired displays for screen **137** and actions of device **130**. In other words, authoring platform **110** is used to program the operation of device **130**. In another embodiment, authoring platform **110** allows a user to provide input for the design of one or more device displays and may further allow the user to save the designs as device specific Applications. The Applications may be stored in memory **123** and may then be sent, when requested by device **130** or when the device is otherwise accessible, over network N, through network interface **130** for storage in memory **133**.

In an alternative embodiment, analytics information from devices **130** may be returned from device **130**, through network N and server **120**, back to authoring platform **110**, as indicated by line B, for later analysis. Analytics information includes, but is not limited to, user demographics, time of day, and location. The type of analytic content is only limited by which listeners have been activated for which objects and for which pages. Analytic content may include, but is not limited to, player-side page view, player-side forms-based content, player-side user interactions, and player-side object status.

Content server **140** is a computer or computer system that includes a network interface **141**, a memory **143**. and a processor **145**. It is to be understood that network interface **141**, memory **143**, and processor **145** are configured such that a stored program in the memory may be executed by the processor to accepts requests R from device **130** and provide content C over a network, such as web server content the Internet, to device **130**.

FIG. **1**B is schematic of an alternative embodiment system **100** for providing programming instructions to device **130** over a network N that is generally similar to the system of FIG. **1**A. The embodiment of FIG. **1**B illustrates that system **100** may include multiple servers **120** and/or multiple devices **130**.

In the embodiment of FIG. **1**B, system **100** is shown as including two or more servers **120**, shown illustratively and without limitation as servers **120***a* and **120***b*. Thus some of the programming or information between authoring platform **110** and one or more devices **130** may be stored, routed, updated, or controlled by more than one server **120**. In particular, the systems and methods described herein may be executed on one or more server **120**.

Also shown in FIG. **1**B are a plurality of devices **130**, shown illustratively and without limitation as device **130-1**, **130-1**, . . . **130**-N. System **100** may thus direct communication between individual server(s) **120** and specific device(s) **130**.

As described subsequently, individual devices **130** may be provided with program instructions which may be stored in each device's memory **133** and where the instructions are executed by each device's processor **135**. Thus, for example, server(s) **120** may provide device(s) **130** with programming in response to the input of the uses of the individual devices. Further, different devices **130** may be operable using different sets of instructions, that is having one of a variety of different "device platforms." Differing device platforms may result, for example and without limitation, to different operating systems, different versions of an operating system, or different versions of virtual machines on the same operating system. In some embodiments, devices **130** are provided with some programming from authoring system **100** that is particular to the device.

In one embodiment, system **100** provides permits a user of authoring platform **110** to provide instructions to each of the plurality of devices **130** in the form of a device- or device-platform specific instructions for processor **135** of the device, referred to herein and without limitation as a "Player," and a device-independent program, referred to herein and without limitation as an "Application" Thus, for example, authoring platform **110** may be used to generate programming for a plurality of devices **130** having one of several different device platforms. The programming is parsed into instructions used by different device platforms and instructions that are independent of device platform. Thus in one embodiment, device **130** utilizes a Player and an Application to execute programming from authoring platform **110**. A device having the correct Player is then able to interpret and be programmed according to the Application.

In one alternative embodiment, the Player is executed the first time by device **130** ("activated") through an Application directory. In another alternative embodiment, the Player is activated by a web browser or other software on device **130**. In yet another alternative embodiment, Player is activated through a signal to device **130** by a special telephone numbers, such as a short code.

When the Application and the Player are provided to memory **133**, the functioning of device **130** may occur in accordance with the desired programming. Thus in one embodiment, the Application and Player includes programming instructions which may be stored in memory **133** and which, when executed by processor **135**, generate the designed displays on screen **137**. The Application and Player may also include programming instructions which may be stored in memory **133** and which provide instructions to processor **135** to accept input from input device **139**.

Authoring tool **112** may, for example, produce and store within memory **111***a* plurality of Players (for different devices **130**) and a plurality of Applications for displaying pages on all devices. The Players and Applications are then stored on one or more servers **120** and then provided to individual devices **130**. In general, Applications are provided to device **130** for each page of display or a some number of pages. A Player need be provided once or updated as necessary, and thus may be used to display a large number of Applications. This is advantageous for the authoring process, since all of the device-dependent programming is provided to a device only once (or possibly for some small number of upgrades), permitting a smaller Application, which is the same for each device **130**.

Thus, for example and without limitation, in one embodiment, the Player transforms device-independent instructions of the Application into device-specific instructions that are executable by device **130**. Thus, by way of example and without limitation, the Application may include Java programming for generating a display on screen **137**, and the Player may interpret the Java and instruct processor **135** to produce the display according to the Application for execution on a specific device **130** according to the device platform. The Application may in general include, without limitation, instructions for generating a display on screen **137**, instructions for accepting input from input device **139**, instructions for interacting with a user of device **130**, and/or instructions for otherwise operating the device, such as to place a telephone call.

The Application is preferably code in a device-independent format, referred to herein and without limitation as a Portable Description Language (PDL). The device's Player interprets or executes the Application to generate one or more "pages" ("Applications Pages") on a display as defined by the PDL. The Player may include code that is device-specific—that it, each device is provided with a Player that is used in the interpretation and execution of Applications. Authoring tool **112** may thus be used to design one or more device-independent Applications and may also include information on one or more different devices **130** that can be used to generate a Player that specific devices may use to generate displays from the Application.

In one embodiment, system **100** provides Players and Applications to one server **120**, as in FIG. **1**A. In another embodiment, system **100** provides Players to a first server **120***a* and Applications to a second server **120***b*, as in FIG. **1**B.

In one embodiment, authoring tool **112** may be used to program a plurality of different devices **130**, and routines **114** may include device-specific routines. In another embodiment, the Player is of the type that is commonly referred to as a "thin client"—that is, software for running on the device as a client in client-server architecture with a device network which depends primarily on a central server for processing activities, and mainly focuses on conveying input and output between the user and the server.

In one embodiment, authoring platform **110** allows user to arrange objects for display on screen. A graphical user interface ("GUI," or "UI") is particularly well suited to arranging objects, but is not necessary. The objects may correspond to one or more of an input object, an output object, an action object, or may be a decorative display, such as a logo, or background color or pattern, such as a solid or gradient fill. In another embodiment, authoring platform **110** also permits a user to assign actions to one or more of an input object, an output object, or an action object. In yet another embodiment, authoring platform **110** also permits a user to bind one or more of an input object, an output object, or an action object with web services or web components, or permits a user to provide instructions to processor **135** to store or modify information in memory **133**, to navigate to another display or service, or to perform other actions, such as dialing a telephone number.

In certain embodiments, the applicant model used in developing and providing Applications is a PDL. The PDL can be conceptually viewed as a device, operating system and virtual machine agnostic representation of Java serialized objects. In certain embodiments, the PDL is the common language for authoring tool **112**, the Application, and Player. Thus while either designing the Application with the authoring tool **112**, or programming with the SDK, the internal representation of the programming logic is in Java. In one embodiment the SDK is used within a multi-language software development platform comprising an IDE and a plug-in system to extend it, such as the Eclipse Integrated Development Environment (see, for example, http://www.eclipse.org/). At publish time the Java code is translated into a PDL. This translation may also occur in real-time during the execution of any Web Services or backend business logic that interacts with the user.

One embodiment for compacting data that may be used is described in U.S. Pat. No. 6,546,397 to Rempell ("Rempell"), the contents of which are incorporated herein by reference. In that patent the compressed data is described as being a database. The terminology used here is a PDL, that is the "internal database" of Rempell is equivalent to the PDL of the present Application.

The use of a PDL, as described in Rempell, permits for efficient code and data compaction. Code, as well as vector, integer and Boolean data may be compacted and then compressed resulting in a size reduction of 40 to 80 times that of the original Java serialized objects. This is important not only for performance over the network but for utilizing the virtual memory manager of the Player more efficiently. As an example, the reassembled primitives of the Java objects may first undergo logical compression, followed by LZ encoding.

The use of a PDL also provides virtual machine and operating system independence. Since the reassembled primitives of the Application no longer have any dependencies from the original programming language (Java) that they were defined in. The PDL architecture takes full advantage of this by abstracting all the virtual machine and/or operating system interfaces from the code that processes the PDL.

In one embodiment, the PDL is defined by the means of nested arrays of primitives. Accordingly, the use of a PDL provides extensibility and compatibility, with a minimal amount of constraints in extending the Player seamlessly as market demands and device capabilities continue to grow. Compatibility with other languages is inherent based on the various Player abstraction implementations, which may be, for example and without limitation, Java CDC, J2SE or MIDP2 implementations.

In one embodiment, the architecture of Player P includes an abstraction interface that separates all device, operating system and virtual machine dependencies from the Player's Application model business logic (that is, the logic of the server-side facilities) that extend the Application on the Player so that it is efficiently integrated into a comprehensive client/server Application. The use of an abstraction interface permits the more efficient porting to other operating systems and virtual machines and adding of extensions to the Application model so that a PDL can be implemented once and then seamlessly propagated across all platform implementations. The Application model includes all the currently supported UI objects and their attributes and well as all of the various events that are supported in the default Player. Further, less robust platforms can be augmented by extending higher end capabilities inside that platform's abstraction interface implementation.

In one embodiment, authoring platform **110** provides one or more pages, which may be provided in one Application, or a plurality of Applications, which are stored in memory **123** and subsequently provided to memory **133**. In certain embodiments, the Application includes instructions R to request content or web services C from content server **140**. Thus, for example and without limitation, the request is for information over the network via a web service, and the request R is responded to with the appropriate information for display on device **130**. Thus, for example, a user may request a news report. The Application may include the layout of the display, including a space for the news, which is downloaded form content server **140** for inclusion on the display. Other information that may be provided by content server **140** may include, but is not limited to, pages, Applications, multimedia, and audio.

FIG. **2**A is a schematic of a system **200** of an embodiment of system **100** illustrating the communications between different system components. System includes a response director **210**, a web component registry **220**, and a web service **230**. System **200** further includes authoring platform **110**, server

120, device **130** and content server **140** are which are generally similar to those of the embodiments of FIGS. **1**A and **1**B, except as explicitly noted.

Response director **210** is a computer or computer system that may be generally similar to server **120** including the ability to communicate with authoring platform **110** and one or more devices **130**. In particular, authoring platform **110** generates one or more Players (each usable by certain devices **130**) which are provided to response director **210**. Devices **130** may be operated to provide response director **210** with a request for a Player and to receive and install the Player. In one embodiment, device **130** provides response director **210** with device-specific information including but not limited to make, model, and/or software version of the device. Response director **210** then determines the appropriate Player for the device, and provides the device with the Player over the network.

Web service **230** is a plurality of services obtainable over the Internet. Each web service is identified and/or defined as an entry in web component registry **230**, which is a database, XML file, or PDL that exists on a computer that may be a server previously described or another server **120**. Web component registry **230** is provided through server **120** to authoring platform **110** so that a user of the authoring platform may bind web services **230** to elements to be displayed on device **130**, as described subsequently.

In one embodiment, authoring platform **110** is used in conjunction with a display that provides a WYSIWYG environment in which a user of the authoring platform can produce an Application and Player that produces the same display and the desired programming on device **130**. Thus, for example, authoring tool **112** provides a display on screen **115** that corresponds to the finished page that will be displayed on screen **137** when an Application is intercepted, via a Player, on processor **135** of device **130**.

Authoring platform **110** further permits a user of the authoring platform to associate objects, such as objects for presenting on screen **137**, with components of one or more web services **230** that are registered in web component registry **220**. In one embodiment, information is provided in an XML file to web component registry **220** for each registered components of each web service **230**. Web component registry **220** may contain consumer inputs related to each web service **230**, environmental data such as PIM, time or location values, persistent variable data, outputs related to the web service, and/or optional hinting for improving the user's productivity.

A user of authoring platform **110** of system **200** may define associations with web services as WebComponent Bindings. In one embodiment, authoring platform **110** allows a user to associate certain objects for display that provide input or output to components of web service **230**. The associated bindings are saved as a PDL in server **120**.

In one embodiment, an XML web component registry **220** for each registered web service **230** is loaded into authoring platform **110**. The user of system **200** can then assign components of any web service **230** to an Application without any need to write code. In one embodiment, a component of web service **230** is selected from authoring platform **110** which presents the user with WYSIWYG dialog boxes that enable the binding of all the inputs and outputs of component of web service **230** to a GUI component of the Application as will be displayed on screen **137**. In addition, multiple components of one or more web service **230** can be assigned to any Object or Event in order to facilitate mashups. These Object and/or Event bindings, for each instance of a component of any web service **230**, are stored in the PDL. The content server **140** handles all communication between device **130** and the web service **230** and can be automatically deployed as a web application archive to any content server.

Device **130**, upon detecting an event in which a component of a web service **230** has been defined, assembles and sends all related inputs to content server **240**, which proxies the request to web service **230** and returns the requested information to device **130**. The Player on device **130** then takes the outputs of web service **230** and binds the data to the UI components in the Application, as displayed on screen **137**.

In one embodiment, the mechanism for binding the outputs of the web service to the UI components is through symbolic references that matches each output to the symbolic name of the UI component. The outputs, in one embodiment, may include meta-data which could become part of the inputs for subsequent interactions with the web service.

For example, if a user of authoring platform **110** wants to present an ATOM feed on device **130**, they would search through a list of UI Components available in the authoring platform, select the feed they want to use, and bind the output of the feed summary to a textbox. The bindings would be saved into the PDL on server **120** and processed by device **130** at runtime. If the ATOM feed does not exist a new one can be added to the web component registry that contains all the configuration data required, such as the actual feed URL, the web component manager URL, and what output fields are available for binding.

In another embodiment, components of web services **230** are available either to the user of authoring platform **110** or otherwise accessible through the SDK and Java APIs of routines **114**. System **200** permits an expanding set of components of web services **230** including, but not limited to: server pages from content server **120**; third-party web services including, but not limited to: searching (such through Google or Yahoo), maps (such as through MapQuest and Yahoo), storefronts (such as through ThumbPlay), SMS share (such as through clickatel), stock quotes, social networking (such as through FaceBook), stock quotes, weather (such as through Accuweather) and/or movie trailers. Other components include web services for communication and sharing through chats and forums and rich messaging alerts, where message alerts are set-up that in turn could have components of Web Services **230** defined within them, including the capture of consumer generated and Web Service supplied rich media and textual content.

System **200** also permits dynamic binding of real-time content, where the inputs and outputs of XML web services are bound to GUI components provided on screen **137**. Thus, for example, a user of authoring platform **110** may bind attributes of UI Objects to a particular data base field on a Server. When running the Application, the current value in the referenced data base will be immediately applied. During the Application session, any other real time changes to these values in the referenced data base will again be immediately displayed.

As an example of dynamic binding of real-time content, an RSS feeds and other forms of dynamic content may be inserted into mobile Applications, such as device **130**, using system **200**. Authoring platform **110** may include a "RSS display" list which permits a user to select RSS channels and feeds from an extensible list of available dynamic content. Meta data, such as titles, abstracts and Images can be revealed immediately by the user as they traverse this RSS display list, bringing the PC experience completely and conveniently to mobile devices **130**. In addition, Authoring platform **110** may include a dialog box that dynamically links objects to data and feeds determined by RSS and chat databases. Any relevant attribute for a page view and/or object can be dynamically bound to a value in a server-side database. This includes elements within complex objects such as: any icon or text element within a graphical list; any icon within a launch strip; any feature within any geographical view of a GIS service object; and/or any virtual room within a virtual tour.

As an example of third-party web services **230** that may be provided using system **200**, a user of authoring platform **110** can place, for example, Yahoo maps into device **130** by binding the required component of the Yahoo Maps Web Service, such as Yahoo Map's Inputs and/or Outputs to appropriate Objects of authoring platform **110**. System **200** also provides binding to web services for text, image and video searching by binding to components of those web services.

In one embodiment, an Application for displaying on device **130** includes one or more Applications Pages, each referred to herein as an "XSP," that provides functionality that extends beyond traditional web browsers. The XSP is defined as a PDL, in a similar manner as any Application, although it defines a single page view, and is downloaded to the Player dynamically as required by the PDL definition of the Application. Thus, for example, while JSPs and ASPs, are restricted to the functionality supported by the web browser, the functionality of XSPs can be extended through authoring platform **110** having access to platform dependent routines **114**, such as Java APIs. Combined with dynamic binding functionality, an XSP, a page can be saved as a page object in an author's "pages" library, and then can be dynamically populated with real-time content simultaneously as the page is downloaded to a given handset Player based on a newly expanded API. XSP Server Pages can also be produced programmatically, but in most cases authoring platform **110** will be a much more efficient way to generate and maintain libraries of dynamically changing XSPs.

With XSPs, Applications Pages that have dynamic content associated with them can be sent directly to device **130**, much like how a web browser downloads an HTML page through a external reference. Without XSPs, content authors would have to define each page in the Application. With XSPs, no pages need to be defined. Thus, for example, in a World Cup Application, one page could represent real-time scores that change continuously on demand. With polling (for example, a prompt to the users asking who they predict will win a game), a back-end database would tabulate the information and then send the results dynamically to the handsets. With a bar chart, the Application would use dynamic PDL with scaling on the fly. For example, the server would recalibrate the bar chart for every ten numbers.

Other combinations of components of web services **230** include, but are not limited to, simultaneous video chat sessions, inside an integrated page view, with a video or television station; multiple simultaneous chat sessions, each with a designated individual and/or group, with each of the chat threads visible inside an integrated page view.

Another extension of an XSP is a widget object. Widgets can be developed from numerous sources including, but not limited to, authoring platform **110**, a Consumer Publishing Tool, and an XML to Widget Conversion Tool where the SDK Widget Libraries are automatically populated and managed, or Widget Selection Lists that are available and can be populated with author defined Icons.

Applications, Players, and Processing in a Device

FIG. **2**B is a schematic of one embodiment of a device **130** illustrating an embodiment of the programming generated by authoring platform **110**. Memory **133** may include several different logical portions, such as a heap **133***a*, a record store **133***b* and a filesystem (not shown).

As shown in FIG. **2**B, heap **133***a* and record store **133***b* include programming and/or content. In general, heap **133***a* is readily accessible by processor **135** and includes, but is not limited to portions that include the following programming: a portion **133***a***1** for virtual machine compliant objects representing a single Page View for screen **137**; a portion **133***a***2** for a Player; a portion **133***a***3** for a virtual machine; and a portion **133***a***4** for an operating system.

Record store **133***b* (or alternatively the filesystem) includes, but is not limited to, portions **133***b***1** for Applications and non-streaming content, which may include portions **133***a***2** for images, portions **133***a***4** for audio, and/or portions **133***a***5** for video. and portions **133***b***2** for non-Application PDLs, such as a Master Page PDL for presenting repeating objects, and Alerts, which are overlayed on the current page view. Other content, such as streaming content may be provided from network interface **131** directly to the Media Codec of device **130** with instructions from Player on how to present the audio or video.

In one embodiment, the Player includes a Threading Model and a Virtual Memory Manager. The Threading Model first manages a queue of actions that can be populated based on Input/Output events, Server-side events, time-based events, or events initiated by user interactions. The Threading Model further manages the simultaneous execution of actions occurring at the same time. The Virtual Memory Manager includes a Logical Virtual Page controller that provides instructions from the record store to the heap, one page at time. Specifically, the Virtual Memory Manager controls the transfer of one of the Application Pages and its virtual machine compliant objects into portion **133***a***1** as instructions readable by the Player or Virtual Machine. When the Player determines that a new set of instructions is required, the information (such as one Application Page is retrieve from the Record store, converted into virtual machine compliant objects (by processor **135** and according to operation by the Player, Virtual Machine, etc). and stored in heap **133***a*. Alternatively, the Player may augment virtual machine compliant objects with its own libraries for managing user interactions, events, memory, etc.

The connection of portions **133***a***1**, **133***a***2**, **133***a***3**, **133***a***4**, record store **133***b* and processor **135** are illustrative of the logical connection between the different types of programming stored in Heap **133***a* and record store **133***b*, that is, how data is processed by processor **135**.

The Player determines which of the plurality of Application Pages in portion **133***b***1** is required next. This may be determined by input actions from the Input Device **139**, or from instructions from the current Application Page. The Player instructs processor **135** to extract the PDF from that Applications Page and store it in portion **133***a***1**. The Player then interprets the Application Page extracted from PDL which in turn defines all of the virtual machine compliant Objects, some of which could have attributes that refer to images, audio, and/or video stored in portions **133***a***3**, **133***a***4**, **133***a***5**, respectively.

The Virtual Machine in portion **133***a***3** processes the Player output, the Operating System in portion **133***a***3** processes the Virtual Machine output which results in machine code that is processed by the Operating System in portion **133***a***4**.

In another embodiment, the Player is a native program that interacts directly with the operating system.

Embodiments of a Publishing Environment

In one embodiment, authoring platform **110** includes a full-featured authoring tool **112** that provides a what-you-see-is-what-you-get (WYSIWYG) full featured editor. Thus, for example, authoring tool **112** permits a user to design an Application by placing objects on canvas **305** and optionally assigning actions to the objects and save the Application. System **100** then provides the Application and Player to a device **130**. The Application as it runs on device **130** has the same look and operation as designed on authoring platform **110**. In certain embodiments, authoring platform **110** is, for example and without limitation, a PC-compatible or a Macintosh computer.

Authoring platform **110** produces an Application having one or more Applications Pages, which are similar to web pages. That is, each Applications Page, when executed on device **130** may, according to its contents, modify what is displayed on screen **137** or cause programming on the device to change in a manner similar to how web pages are displayed and navigated through on a website.

In one embodiment, authoring tool **112** allows a user to place one or more objects on canvas **305** and associate the objects with an Applications Pages. Authoring platform **110** maintains a database of object data in memory **111**, including but not limited to type of object, location on which page, and object attributes. The user may add settings, events, animations or binding to the object, from authoring tool **112**, which are also maintained in memory **111**. Authoring tool **112** also allows a user to define more than one Applications Page.

In another embodiment, authoring tool **112**, provides Java programming functions of the Java API for specific devices **130** as pull-down menus, dialog boxes, or buttons. This permits a user of authoring platform **110** to position objects that, after being provided as an Application to device **130**, activate such Java functions on the device.

In certain embodiments, authoring platform **110**, as part of system **100**, permits designers to include features of advanced web and web services Applications for access by users of device **130**. Some of the features of advanced web and web services include, but are not limited to: slide shows, images, video, audio, animated transitions, multiple chats, and mouse interaction; full 2-D vector graphics; GIS (advanced LBS), including multiple raster and vector layers, feature sensitive interactions, location awareness, streaming and embedded audio/video, virtual tours, image processing and enhancement, and widgets. In other embodiments the features are provided for selection in authoring platform **110** through interactive object libraries.

In certain embodiments, authoring platform **110**, as part of system **100**, allows the inclusion of child objects which may eventually be activated on device **130** by the user of the device or by time. The uses of the child objects on device **130** include, but are not limited to: mouse over (object selection), hover and fire events and launching of object-specific, rich-media experiences.

In certain other embodiments, authoring platform **110**, as part of system **100**, provides advanced interactive event models on device **130**, including but not limited to: user-, time- and/or location-initiated events, which allow content developers to base interactivity on specific user interactions and/or instances in time and space; timelines, which are critical for timing of multiple events and for animations when entering, on, or exiting pages of the Application; waypoints, which act similar to key frames, to allow smooth movement of objects within pages of the Application. Waypoints define positions on a page object's animation trajectory. When an object reaches a specific waypoint other object timelines can be initiated, creating location-sensitive multiple object interaction, and/or audio can be defined to play until the object reaches the next waypoint.

Authoring platform **110** may also define a Master Page, which acts as a template for an Applications Page, and may also define Alert Pages, which provide user alerts to a user of device **130**.

In certain embodiments, authoring platform **110**, as part of system **100**, provides full style inheritance on device **130**. Thus, for example and without limitation, both master page inheritance (for structural layout inheritance and repeating objects) and object styles (for both look and feel attribute inheritance) are supported. After a style has been defined for an object, the object will inherit the style. Style attributes include both the look and the feel of an object, including mouse interaction, animations, and timelines. Each page may include objects that may be a parent object or a child object. A child object is one that was created by first selecting a parent object, and then creating a child object. Child objects are always part of the same drawing layer as its parent object, but are drawn first, and are not directly selectable when running the Application. A parent object is any object that is not a child object, and can be selected when running the Application.

As an example, the user of authoring tool **112** may create various text objects on canvas **305** using a style that sets the font to red, the fonts of these objects will be red. Suppose user of authoring tool **112** changes the font color of a specific button to green. If later, the user of authoring tool **112** changes the style to blue; all other text objects that were created with that style will become blue except for the button that had been specifically set to green.

In certain other embodiments, authoring platform **110** provides page view, style, object, widget and Application template libraries. Authoring platform **110** may provide templates in private libraries (available to certain users of the authoring platform) and public libraries (available to all users of the authoring platform). Templates may be used to within authoring platform **110** to define the look and feel of the entire Application, specific pages, or specific slide shows and virtual tours a seen on device **130**.

FIGS. **3**A and **3**B illustrate one embodiment of a publisher interface **300** as it appears, for example and without limitation, on screen **115** while executing authoring tool **112**. In one embodiment, publisher interface **300** includes a Menu bar **301**, a Tool bar **303**, a Canvas **305**, a Layer Inspector **307** having subcomponents of a page/object panel **307***a*, an object style panel **307***b*, and a page alert panel **307***c*, and a Resource Inspector **309**.

In general, publisher interface **300** permits a user of authoring platform **110** to place objects on canvas **305** and then associate properties and/or actions to the object, which are stored in the Application. As described subsequently, publisher interface **300** permits a user to program a graphical interface for the screen **137** of device **130** on screen **115** of authoring platform **110**, save an Application having the programming instructions, and save a Player for the device. The intended programming is carried out on device **130** when the device, having the appropriate device platform Player, receives and executes the device-independent Application.

Thus, for example, authoring tool **112** maintains, in memory **111**, a list of every type of object and any properties, actions, events, or bindings that may be assigned to that object. As objects are selected for an Application, authoring tool **112** further maintains, in memory **111**, a listing of the objects. As the user selects objects, publisher interface **300**

provides the user with a choice of further defining properties, actions, events, or bindings that may be assigned to each particular object, and continues to store the information in memory **111**.

In one embodiment, publisher interface **300** is a graphical interface that permits the placement and association of objects in a manner typical of, for example, vector graphics editing programs (such as Adobe Illustrator). Objects located on canvas **305** placed and manipulated by the various commands within publisher interface **300** or inputs such as an input device **117** which may be a keyboard or mouse. As described herein, the contents of canvas **305** may be saved as an Application that, through system **100**, provide the same or a similar placement of objects on screen **137** and have actions defined within publisher interface **300**. Objects placed on canvas **305** are intended for interaction with user of device **130** and are referred to herein, without limitation, as objects or UI (user interface) objects. In addition, the user of interface **300** may assign or associate actions or web bindings to UI objects placed on canvas **305** with result in the programming device **130** that cause it to respond accordingly.

Objects include, but are not limited to input UI objects, response UI objects. Input UI objects include but are not limited to: text fields (including but not limited to alpha, numeric, phone number, or SMS number); text areas; choice objects (including but not limited to returning the selected visible string or returning a numeric hidden attribute); single item selection lists (including but not limited to returning the selected visible string or returning a numeric hidden attribute); multi item selection lists (including but not limited to returning all selected items (visible text string or hidden attribute) or cluster item selection lists (returning the hidden attributes for all items).

Other input UI objects include but are not limited to: check boxes; slide show (including but not limited to returning a numeric hidden attribute, returning a string hidden attribute, or returning the hidden attributes for all slides); and submit function (which can be assigned to any object including submit buttons, vectors, etc.).

Response UI Objects may include, but are not limited to: single line text objects, which include: a text Field (including but not limited to a URL, audio URL, or purchase URL), a text button, a submit button, or a clear button. Another response UI objects include: a multiple line text object, which may include a text area or a paragraph; a check box; an image; a video; a slide show (with either video or image slides, or both); choice objects; list objects; or control lists, which control all the subordinate output UI objects for that web component. Control list objects include, but are not limited to: list type or a choice type, each of which may include a search response list or RSS display list.

As a further example of objects that may be used with authoring tool **112**, Table I lists Data Types, Preferred Input, Input Candidates, Preferred Output and Output Candidates for one embodiment of an authoring tool.

TABLE I

One embodiment of supported objects

Data Types	Preferred Input	Input Candidates	Preferred Output	Output Candidates
boolean	Check Box	Check Box	Check Box	Check Box
Int	Text Field (integer)	Text Field (integer) Text Field (Phone #) Text Field (SMS #) Choice List (single select)	Text Field (integer)	Text Field (integer) Text Field (Phone #) Text Field (SMS #) Choice List (single select) Text Button

TABLE I-continued

One embodiment of supported objects

Data Types	Preferred Input	Input Candidates	Preferred Output	Output Candidates
String	Text Field (Alpha)	Any	Text Field (Alpha)	Any
multilineString	Text Area	Text Area	Text Area	Text Area Paragraph
ImageURL	N/A	N/A	Image	Image Slide Show
VideoURL	N/A	N/A	Video	Video Slide Show
List	Single Item List	Single Item List Multi-Select List Complex List Choice Slide Show	Single Item List	Any List Type Any Choice Type (see Complex List Specification)
ComplexList	Complex List	Single Item List Multi-Select List Complex List	Single Item List	Any List Type (see Complex List Specification)
Slideshow	Slide Show	Slide Show	Slide Show	Slide Show
SearchResponseList	N/A	N/A	Search Response List	Search Response List Control List Complex List Choice
RSSList	N/A	N/A	RSS Display List	RSS Display List Control List Complex List Choice
SingleSelectionList	Choice	Choice Complex List	Choice	Choice Complex List
MultiSelectionList	Multi-Selection List	Multi-Selection List	Multi-Selection List	Multi-Selection List
ServiceActivation	Submit Button	Any	N/A	N/A
ChannelImageURL	N/A	N/A	Image	Image Video Slide Show
ChannelDescription	N/A	N/A	Text Area	Text Area Paragraph Text Field Text Button List Choice
ChannelTitle	N/A	N/A	Text Field	Text Field Text Button Paragraph Text Area List Choice
URL			Text Field (URL request)	Text Field (URL request)
Audio URL			Text Field (Audio URL request)	Text Field (Audio URL request)
Purchase URL			Text Field (Purchase URL request)	Text Field (Purchase URL request)
Image Data			Image	Image Slide Show
Image List Data			Slide Show	Slide Show Image
Persistent Variable	N/A	N/A	N/A	N/A
Pipeline Multiple Select	Multi-select List	Multi-select List Complex List Slide Show	N/A	N/A
Phone Number	Text Field (numeric type)	Text Field Text Button	Text Field (numeric type)	Text Field Text Button
Hidden Attribute	Complex List	Complex List Slide Show	Complex List	Complex List Slide Show
Collection List	N/A	N/A	Slide Show	Complex List Slide Show

In general, publisher interface **300** permits a user to define an Application as one or more Applications Pages, select UI objects from Menu bar **301** or Tool bar **303** and arrange them on an Applications Page by placing the objects canvas **305**. An Application Page is a page that is available to be visited through any navigation event. Application Pages inherit all the attributes of the Master Page, unless that attribute is specifically changed during an editing session.

Authoring platform **110** also stores information for each UI object on each Application Page of an Application. Layer Inspector **307** provides lists of Applications Pages, UI objects on each Applications Page, and Styles, including templates. Objects may be selected from canvas **305** or Layer Inspector **307** causing Resource Inspector **309** to provide lists of various UI objects attributes which may be selected from within the Resource Inspector. Publisher interface **300** also permits a user to save their work as an Application for layer transfer and operation of device **130**. Publisher interface **300** thus provides an integrated platform for designing the look and operation of device **130**.

The information stored for each UI object depends, in part, on actions which occur as the result of a user of device **130** selecting the UI object from the device. UI objects include, but are not limited to: navigational objects, such as widget or channel launch strips or selection lists; message objects for communicating, such as a multiple chat, video chat, phone and/or SMS lists or fields or a pop-up alert; text fields or areas; check boxes; pull down menus; selection lists and buttons; pictures; slide shows; video or LBS maps; shapes or text defined by a variety of tools; a search response; or an RSS display.

In certain embodiments, publisher interface **300** permits a user to assign action to UI objects, including but not limited to, programming of the device **130** or a request for information over network N. In one embodiment, for example and without limitation, publisher interface **300** has a selection to bind a UI object to a web service—that is, associate the UI object or a manipulation or selection of UI object with web services. Publisher interface **300** may also include many drawing and text input functions for generating displays that may be, in some ways, similar to drawing and/or word processing programs, as well as toolbars and for zooming and scrolling of a workspace.

Each UI object has some form, color, and display location associate with it. Further, for example and without limitation, UI objects may have navigational actions (such as return to home page), communications actions (such as to call the number in a phone number field), or web services (such as to provide and/or retrieve certain information from a web service). Each of the these actions requires authoring platform **110** to store the appropriate information for each action. In addition, UI objects may have associated patent or child objects, default settings, attributes (such as being a password or a phone number), whether a field is editable, animation of the object, all of which may be stored by authoring platform **110**, as appropriate.

Menu bar **301** provides access features of publisher interface **300** through a series of pull-down menus that may include, but are not limited to, the following pull-down menus: a File menu **301***a*, an Edit menu **301***b*, a View menu **301***c*, a Project menu **301***d*, an Objects menu **301***e*, an Events menu **301***f*, a Pages menu **301***g*, a Styles menu **301***h*, and a Help menu **301***i*.

File menu **301***a* provides access to files on authoring platform **110** and may include, for example and without limitation, selections to open a new Application or master page, open a saved Application, Application template, or style template, import a page, alert, or widget, open library objects including but not limited to an image, video, slide show, vector or list, and copying an Application to a user or to Server **120**.

Edit menu **301***b* may include, but is not limited to, selections for select, cut, copy, paste, and edit functions.

View menu **301***c* may include, but is not limited to, selections for zooming in and out, previewing, canvas **305** grid display, and various palette display selections.

Project menu **301***d* may include, but is not limited to, selections related to the Application and Player, such as selections that require a log in, generate a universal Player, generate server pages, activate server APIs and extend Player APIs. A Universal Player will include all the code libraries for the Player, including those that are not referenced by the current Application. Server APIs and Player APIs logically extend the Player with Server-side or device-side Application specific logic.

Objects menu **301***e* includes selections for placing various objects on canvas **305** including, but not limited to: navigation UI objects, including but not limited to widget or channel launch strips or selection lists; message-related UI objects, including but not limited to multiple chat, video chat, phone and/or SMS lists or fields, or a pop-up alert; shapes, which provides for drawing tools; forms-related objects, including but not limited to text fields; scrolling text box, check box, drop-down menu, list menu, submit button or clear button; media-related UI objects such as pictures, slide shows, video or LBS maps; text-related UI objects such as buttons or paragraphs; and variables, including but not limited to time, date and audio mute control.

Events menu **301***f* includes selections for defining child objects, mouse events, animations or timelines.

Pages menu **301**g includes selection for handling multi-page Applications, and may include selections to set a master page, delete, copy, add or go to Applications Pages.

Styles menu **301***h* includes selections to handle styles, which are the underlying set of default appearance attributes or behaviors that define any object that is attached to a style. Styles are a convenient way for quickly creating complex objects, and for changing a whole collection of objects by just modifying their common style. Selections of Styles menu **301***h* include, but not limited to, define, import, or modify a style, or apply a template. Help menu **301***i* includes access a variety of help topics.

Tool bar **303** provides more direct access to some of the features of publisher interface **300** through a series of pull-down menus. Selections under tool bar **303** may include selections to:

control the look of publisher interface **300**, such as a Panel selection to control the for hiding or viewing various panels on publisher interface **300**;

control the layout being designed, such as an Insert Page selection to permit a user to insert and name pages;

control the functionality of publisher interface **300**, such as a Palettes selection to choose from a variety of specialized palettes, such as a View Palette for zooming and controlling the display of canvas **305**, a Command Palette of common commands, and Color and Shape Palettes;

place objects on canvas **305**, which may include selections such as: a Navigation selection to place navigational objects, such as widget or channel launch strips or selection lists), a Messages selection to place objects for communicating, such as a multiple chat, video chat, phone and/or SMS lists or fields, or a pop-up alert, a Forms selection to place objects such as text fields or areas, check boxes, pull down menus, selection lists, and buttons, a Media selection to place pictures, slide shows, video or LBS maps, and a Shapes selection having a variety of drawing tools, a Text selection for placing text, a search response, or an RSS display, and Palettes.

In one embodiment, Tool bar **303** includes a series of pull-down menus that may include, but are not limited to, items from Menu bar **301** organized in the following pull-down menus: a Panel menu **303***a*, an Insert Page menu **303***b*, a Navigation menu **303***c*, a Messages menu **303***d*, a Forms menu **303***e*, a Media menu **303***f*, a Shapes menu **303***g*, a Text menu **303***h*, and a Palettes menu **301***i*.

Panel menu **303***a* permits a user of authoring platform **110** to change the appearance of interface **300** by, controlling which tools are on the interface or the size of canvas **305**. Insert Page menu **303***b* permits a user of authoring platform **110** to open a new Application Page. Navigation menu **303***c* displays a drop down menu of navigational-related objects such as a widget or channel launch strip or selection list. Messages menu **303***d* displays a drop down menu of messaging-related objects such as multiple chat, video chat, phone or SMS lists or fields, and pop-up alerts. Forms menu **303***e* displays a drop down menu of forms-related objects including, but not limited to, a text field, a text area, a check box, a drop down menu, a selection list, a submit button, and a clear button. Media menu **303***f* displays a drop down menu of media-related objects including, but not limited to, a picture, slide show, video or LBS map. Shapes menu **303***g* displays a drop down menu of draw tools, basic shapes, different types of lines and arrows and access to a shape library. Text menu **303***j* displays a drop down menu of text-related objects, including but not limited to a text button, paragraph, search response, RSS display and variables such as time and date.

Palettes menu **301***i* includes a selection of different palettes that can be moved about publisher interface **300**, where each palette has specialized commands for making adjustments or associations to objects easier. Palettes include, but are not limited to: a page view palette, to permit easy movement between Applications Pages; a view palette, to execute an Application or zoom or otherwise control the viewing of an Application; a commands palette having editing commands; a color palette for selection of object colors; and a shapes palette to facilitate drawing objects.

Layer inspector **307** permits a user of publisher interface **300** to navigate, select and manipulate UI objects on Applications Pages. Thus, for example, a Page/objects panel **307***a* of layer inspector **307** has a listing that may be selected to choose an Applications Pages within and Application, and UI objects and styles within an Applications Page. An Object styles panel **307***b* of layer inspector **307** displays all styles on the Applications Page and permits selection of UI objects for operations to be performed on the objects.

Thus, for example, when objects from Menu bar **301** or Tool bar **303** are placed on canvas **305**, the name of the object appears in Page/objects panel **307***a*. Page/objects panel **307***a* includes a page display **307***a***1** and an objects display **307***a***2**. Page display **307***a***1** includes a pull down menu listing all Applications Pages of the Application, and objects display **307***a***2** includes a list of all objects in the Applications Page (that is, objects on canvas **305**).

In general, page/objects panel **307***a* displays various associations with a UI object and permits various manipulations including, but not limited to, operations for parent and child objects that are assigned to a page, and operations for object styles, and permits navigating between page types and object styles, such as switching between the master page and Application pages and deselecting object styles and alerts, opening an Edit Styles Dialog Box and deselecting any master, Application or alert page, or selecting an alert page and deselecting any Master Page or Application Page. A parent or child object can also be selected directly from the Canvas. In either case, the Resource Inspector can then be used for modifying any attribute of the selected object.

Examples of operations provided by page/objects panel **307***a* on pages include, but are not limited to: importing from either a user's private page library or a public page library; deleting a page; inserting a new page, inheriting all the attributes of the Master Page, and placing the new page at any location in the Page List; editing the currently selected page, by working with an Edit Page Dialog Box. While editing all the functions of the Resource Inspector **309** are available, as described subsequently, but are not applied to the actual page until completing the editing process.

Examples of operations provided by of page/objects panel **307***a* on objects, which may be user interface (UI) objects, include but are not limited to: changing the drawing order layer to: bring to the front, send to the back, bring to the front one layer, or send to the back one layer; hiding (and then reshowing) selected objects to show UI objects obstructed by other UI Objects, delete a selected UI Page Object, and editing the currently selected page, by working with a Edit Page Dialog Box.

Object styles panel **307***b* of layer inspector **307** displays all styles on the Applications Page and permits operations to be performed on objects, and is similar to panel **307***a*. Examples of operations provided by object style panel **307***b* include, but are not limited to: importing from either a user's private object library or a public object library; inserting a new object style, which can be inherited from a currently selected object, or from a previously defined style object; and editing a currently selected object style by working with an Edit Style Dialog Box.

Style attributes can be assigned many attributes, including the look, and behavior of any object that inherits these objects. In addition, List Layout Styles can be created or changed as required. A layout style can define a unbounded set of Complex List Layouts, including but not limited to: the number of lines per item in a list, the number of text and image elements and their location for each line for each item in the last, the color and font for each text element, and the vertical and horizontal offset for each image and text element.

Alerts Panel **307***c* provides a way of providing alert pages, which can have many of the attributes of Application Pages, but they are only activated through an Event such as a user interaction, a network event, a timer event, or a system variable setting, and will be superimposed onto whatever is currently being displayed. Alert Pages all have transparent backgrounds, and they function as a template overlay, and can also have dynamic binding to real time content.

Resource inspector **309** is the primary panel for interactively working with UI objects that have been placed on the Canvas **305**. When a UI object is selected on Canvas **305**, a user of authoring platform **110** may associate properties of the selected object by entering or selecting from resource inspector **309**. In one embodiment, resource inspector **309** includes five tab selections: Setting Tab **309***a*, Events Tab **309***b*, Animation Tab **309***c*, Color Tab **309***d* which includes a color palette for selecting object colors, and Bindings Tab **309***e*.

Settings Tab **309***a* provides a dialog box for the basic configuration of the selected object including, but not limited to, name, size, location, navigation and visual settings. Depending upon the type of object, numerous other attributes could be settable. As an example, the Setting Tab for a Text Field may include dialog boxes to define the text field string, define the object style, set the font name, size and effects, set an object name, frame style, frame width, text attributes (text field, password field, numeric field, phone number, SMS number, URL request).

As an example of Setting Tab **309***a*, FIG. **3**B shows various selections including, but not limited to, setting **309***a***1** for the web page name, setting **309***a***2** for the page size, including selections for specific devices **130**, setting **309***a***3** indicating the width and height of the object, and setting **309***a***4** to select whether background audio is present and to select an audio file.

FIG. **3**C illustrates an embodiment of the Events Tab **309***b*, which includes all end user interactions and time based operations. The embodiment of Events Tab **309***b* in FIG. **3**C includes, for example and without limitation, an Events and

Services **309*b*1**, Advanced Interactive Settings **309*b*2**, Mouse State **309*b*3**, Object Selected Audio Setting **309*b*4**, and Work with Child Objects and Mouse Overs button **309*b*5**.

Events and Services **309*b*1** lists events and services that may be applied to the selected objects. These include, but are not limited to, going to external web pages or other Applications pages, either as a new page or by launching a new window, executing an Application or JavaScript method, pausing or exiting, placing a phone call or SMS message, with or without single or multiple Player download, show launch strip, or go back to previous page. Examples of events and services include, but are not limited to those listed in Table II

TABLE II

Events and Services	
Goto External Web Page replacing Current Frame	ChoiceObject: Remove Icon from Launch Strip
Goto External Web Page Launched in a New Window	Goto a specific Internal Web Page with Alert. "Backend Synchronization"
Goto a specific Internal Web Page	Goto Widget Object
Goto the next Internal Web Page	Generate Alert. "With a Fire Event"
Goto External Web Page replacing the Top Frame	Send SMS Message from Linked Text Field
Execute JavaScript Method	Toggle Alert. "Display OnFocus, Hide OffFocus"
Pause/Resume Page Timeout	Execute an Application with Alert. "With a Fire Event"
Execute an Application	Goto Logical First Page
Goto a specific Internal Web Page with setting starting slide	Generate Alert with Backend Synchronization
Exit Application	Send SMS Message with Share (Player Download)
Exit Player	Place PhoneCall from linked Text Field with Share (Player Download)
Place PhoneCall from linked Text Field	Send IM Alert from linked Text Field or Text Area
Text Field/Area: Send String on FIRE	Set and Goto Starting Page
ChoiceObject: Add Icon to Launch Strip	Populate Image
Text Field/Area: Send String on FIRE or Numeric Keys	Preferred Launch Strip

Advanced Interactive Settings **309*b*2** include Scroll Activation Enabled, Timeline Entry Suppressed, Enable Server Listener, Submit Form, Toggle Children on FIRE, and Hide Non-related Children, Mouse State **309*b*3** selections are Selected or Fire. When Mouse State Selected is chosen, Object Selected Audio Setting **309*b*4** of Inactive, Play Once, Loop, and other responses are presented. When Mouse State Fire is chosen, Object Selected Audio Setting **309*b*4** is replaced with FIRE Audi Setting, with appropriate choices presented.

When Work with Child Objects and Mouse Overs button **309*b*5** is selected, a Child Object Mode box pops up, allowing a user to create a child object with shortcut to Menu bar **301** actions that may be used define child objects.

FIG. **3**D illustrates one embodiment of an Animation Tab **309***c*, which includes all animations and timelines. The Color Tab includes all the possible color attributes, which may vary significantly by object type.

Animation Tab **309***c* includes settings involved in animation and timelines that may be associated with objects. One embodiment of Animation Tab **309***c* is shown, without limitation, in FIG. **3**D, and is described, in Rempell ("Rempell").

A Color Tab **309***d* includes a color palette for selecting object colors.

Bindings Tab **309***e* is where web component operations are defined and dynamic binding settings are assigned. Thus, for

example, a UI object is selected from canvas **305**, and a web component may be selected and configured from the bindings tab. When the user's work is saved, binding information is associated with the UI object that will appear on screen **137**.

FIG. **3**E illustrates one embodiment of Bindings Tab and includes, without limitation, the following portions: Web Component and Web Services Operations **309*e*1**, Attributes Exposed list **309*e*2**, panel **309*e*3** which includes dynamic binding of server-side data base values to attributes for the selected object, Default Attribute Value **309*e*4**, Database Name **309*e*5**, Table Name **309*e*6**, Field Name **309*e*7**, Channel Name **309*e*8**, Channel Feed **309*e*9**, Operation **309*e*10**, Select Link **309*e*11**, and Link Set checkbox **309*e*12**.

Web Component and Web Services Operations **309*e*1** includes web components that may be added, edited or removed from a selected object. Since multiple web components can be added to the same object, any combination of mash-ups of 3rd party web services is possible. When the "Add" button of Web Component and Web Services Operations **309*e*1** is selected, a pop-up menu **319**, as shown in FIG. **3**F, appears on publisher interface **300**. Pop-up menu **319** includes, but is not limited to, the options of: Select a Web Component **319***a*; Select Results Page **319***b*; Activation Options **319***c*; Generate UI Objects **319***d*; and Share Web Component **319***e*.

The Select a Web Component **319***a* portion presents a list of web components. As discussed herein, the web components are registered and are obtained from web component registry **220**.

Select Results Page **319***b* is used to have the input and output on different pages—that is, when the Results page is different from Input page. The default selected results page is either the current page, or, if there are both inputs and outputs, it will be set provisionally to the next page in the current page order, if one exists.

Activation Options **319***c* include, if there are no Input UI Objects, a choice to either "Preload" the web component, similar to how dynamic binding, or have the web component executed when the "Results" page is viewed by the consumer.

Generate UI Objects **319***c*, if selected, will automatically generate the UI objects. If not selected, then the author will bind the Web Component Inputs and Results to previously created UI Objects.

Share Web Component **319***e* is available and will become selected under the following conditions: 1) Web Component is Selected which already has been used by the current Application; or 2) the current Input page is also a "Result" page for that Web component. This permits the user of device **130**, after viewing the results, to extend the Web Component allowing the user to make additional queries against the same Web Component. Examples of this include, but are not limited to, interactive panning and zooming for a Mapping Application, or additional and or refined searches for a Search Application.

Dynamic Binding permits the binding of real time data, that could either reside in a 3rd party server-side data base, or in the database maintained by Feed Collector **1010** for aggregating live RSS feeds, as described subsequently with reference to FIG. **10**.

Referring again to FIG. **3**E, Attributes Exposed list **309*e*2** are the attributes available for the selected object that can be defined in real time through dynamic binding.

Panel **309*e*3** exposes all the fields and tables associated with registered server-side data bases. In one embodiment, the user would select an attribute from the "Attributes Exposed List" and then select a data base, table and field to define the real time binding process. The final step is to define the record. If the Feed Collector data base is selected, for example, then the RSS "Channel Name" and the "Channel Feed" drop down menus will be available for symbolically selected the record. For other data bases the RSS "Channel Name" and the "Channel Feed" drop down menus are replaced by a "Record ID" text field.

Default Attribute Value **309***e***4** indicates the currently defined value for the selected attribute. It will be overridden in real time based on the dynamic linkage setting.

Database Name **309***e***5** indicates which server side data base is currently selected. Table Name **309***e***6** indicates which table of the server side data base is currently selected.

Field Name **309***e***7**, indicates which field form the selected table of the server side data base is currently selected.

Channel Name **309***e***8** indicates a list of all the RSS feeds currently supported by the Feed Collector. This may be replaced by "Record ID" if a data base other than the Feed Collector **1010** is selected.

Channel Feed **309***e***9** indicates the particular RSS feed for the selected RSS Channel. Feed Collector **1010** may maintain multiple feeds for each RSS channel.

Operation **309***e***10**, as a default operation, replaces the default attribute value with the real time value. In other embodiments this operation could be append, add, subtract, multiply or divide.

Select Link **309***e***11** a button that, when pressed, creates the dynamic binding. Touching the "Select Link" will cause the current data base selections to begin the blink is some manner, and the "Select Link" will change to "Create Link". The user could still change the data base and attribute choices. Touching the "Create Link" will set the "Link Set" checkbox and the "Create Link" will be replaced by "Delete Link" if the user wishes to subsequently remove the link. When the application is saved, the current active links are used to create the SPDL.

Link Set checkbox **309***e***12** indicates that a link is currently active.

An example of the design of a display is shown in FIGS. **4**A and **4**B according the system **100**, where FIG. **4**A shows publisher interface **300** having a layout **410** on canvas **305**, and FIG. **4**B shows a device **130** having the resulting layout **420** on screen **137**. Thus, for example, authoring platform **110** is used to design layout **410**. Authoring platform **110** then generates an Application and a Player specific to device **130** of FIG. **4**B. The Application and Player are thus used by device **130** to produce layout **420** on screen **137**.

As illustrated in FIG. **4**A, a user has placed the following on canvas **305** to generate layout **410**: text and background designs **411**, a first text input box **413**, a second text input box **415**, and a button **417**. As an example which is not meant to limit the scope of the present invention, layout **410** is screen prompts a user to enter a user name in box **413** and a password in box **415**, and enter the information by clicking on button **417**.

In one embodiment, all UI objects are initially rendered as Java objects on canvas **305**. When the Application is saved, the UI objects are transformed into the PDL, as described subsequently.

Thus, for example, layout **410** may be produced by the user of authoring platform **110** selecting and placing a first Text Field as box **413** then using the Resource Inspector **309** portion of interface **300** to define its attributes.

Device User Experience

Systems **100** and **200** provide the ability for a very large number of different types of user experiences. Some of these are a direct result of the ability of authoring platform **110** to bind UI objects to components of web services. The following

description is illustrative of some of the many types of experiences of using a device **130** as part of system **100** or **200**.

Device **130** may have a one or more of a very powerful and broad set of extensible navigation objects, as well as object- and pointer-navigation options to make it easy to provide a small mobile device screen **137** with content and to navigate easily among page views, between Applications, or within objects in a single page view of an Application.

Navigation objects include various types of launch strips, various intelligent and user-friendly text fields and scrolling text boxes, powerful graphical complex lists, as well as Desktop-level business forms. In fact, every type of object can be used for navigation by assigning a navigation event to it. The authoring tool offers a list of navigation object templates, which then can be modified in numerous ways.

Launch Strips and Graphical List Templates Launch Strips

Launch strips may be designed by the user of authoring platform **110** with almost no restrictions. They can be stationary or appear on command from any edge of the device, their size, style, audio feedback, and animations can be freely defined to create highly compelling experiences.

FIG. **5** shows a display **500** of launch strips which may be on displayed canvas **305** or on screen **137** of device **130** having the proper Player and Application. Display **501** includes a portal-type Launch Strip **501** and a channel-type Launch Strip **502**, either one of which may be included for navigating the Application.

Launch Strip **501** includes UI objects **501***a*, **501***b*, **501***c*, **501***d*, and **501***e* that that becomes visible from the left edge of the display, when requested. UI objects **501***a*, **501***b*, **501***c*, **501***d*, and **501***e* are each associated, through resource inspector **309** with navigational instructions, including but not limited to navigating to a different Applications Page, or requesting web content. When the Applications Page, having been saved by authoring platform **110** and transferred to display **130**, is executed on device **130**, a user of the device may easily navigate the Application.

Launch Strip **502** includes UI objects **502***b*, **502***c*, **502***d*, and **503***e* that that becomes visible from the bottom of the display, when requested. UI objects **501***a*, **501***b*, **501***c*, **501***d*, and **501***e* are each associated, through resource inspector **309** with navigational instructions, including but not limited to navigating to a different Applications Page, or requesting web content. Launch Strip **502** also includes UI objects **502***a* and **503***g*, which include the graphic of arrows, and which provide access to additional navigation objects (not shown) when selected by a user of device **130**. Launch strip **502** may also include sound effects for each channel when being selected, as well as popup bubble help.

Additional navigational features are illustrated in FIG. **6**A as a display of a Channel Selection List **601***a*, in FIG. **6**B as a display of a Widget Selection List **601***b*, and in FIG. **6**C as display of a Phone List **601***c*. Lists **601***a*, **601***b*, and **601***c* may be displayed on canvas **305** or on screen **137** of device **130** having the proper Player and Application. As illustrated, graphical lists **601***a*, **601***b*, and **601***c* may contain items with many possible text and image elements. Each element can be defined at authoring time and/or populated dynamically through one or more Web Service **250** or API. Assignable Navigation Events. All objects, and/or all elements within an object, can be assigned navigation events that can be extended to registered web services or APIs. For example, a Rolodex-type of navigation event can dynamically set the starting slide of the targeted page view (or the starting view of a targeted Application).

In the embodiment of FIGS. **6**A, **6**B, and **6**C, each list **601***a*, **601***b*, and **601***c* has several individual entries that are each linked to specific actions. Thus Channel Selection List **601***a* shows three objects, each dynamically linked to a web service (ESPN, SF 49ers, and Netflix) each providing a link to purchase or obtain items from the Internet. Widget Selection List **601***b* includes several objects presenting different widgets for selecting. Phone List **601***c* includes a list phone number objects of names that, when selected by a user of device **130** cause the number to be dialed Entries in Phone List **601***c* may be generated automatically from either the user's contact list that is resident on the device, or though a dynamic link to any of user's chosen server-side facilities such as Microsoft Outlook, Google Mail, etc. In one embodiment, Phone List **601***c* may be generated automatically using a web component assigned to the Application, which would automatically perform those functions.

In another embodiment, authoring platform **110** allows a navigation selection of objects with a Joy Stick and/or Cursor Keys in all 4 directions. When within a complex object the navigation system automatically adopts to the navigation needs for that object. For coordinate sensitive objects such as geographical information services (GIS) and location-based services (LBS) or virtual tours a soft cursor appears. For Lists, scrolling text areas and chats, Launch strips, and slide shows the navigation process permits intuitive selection of elements within the object. Scroll bars and elevators are optionally available for feedback. If the device has a pointing mechanism then scroll bars are active and simulate the desktop experience.

Personalization and Temporal Adoption

System **100** and **200** permit for the personalization of device **130** by a variety of means. Specifically, what is displayed on screen **137** may depend on either adoption or customization. Adoption refers to the selection of choices, navigation options, etc. are based on user usage patterns. Temporal adoption permits the skins, choices, layouts, content. widgets, etc. to be further influenced by location (for example home, work or traveling) and time of day (including season and day of week). Customization refers to user selectable skins, choices, layouts, dynamic content, widgets, etc. that are available either through a customization on the phone or one that is on the desktop but dynamically linked to the user's other internet connected devices.

To support many personalization functions there must be a convenient method for maintaining, both within a user's session, and between sessions, memory about various user choices and events. Both utilizing a persistent storage mechanism on the device, or a database for user profiles on a server, may be employed.

FIG. **7** shows a display **700** of a mash-up which may be on displayed canvas **305** or on screen **137** of device **130** having the proper Player and Application. Display **700** includes several object **701** that have been dynamically bound, including an indication of time **701***a*, an indication of unread text messages **701***b*, an RSS news feed **701***c* (including 2 "ESPN Top Stories" **701***c***1** and **701***c***2**), components **701***d* from two Web Services—a weather report ("The Weather Channel"), and a traffic report **701***e* ("TRAFFIC.COM").

In assembling the information of display **700**, device **130** is aware of the time and location of the device—in this example the display is for a workday when a user wakes. Device **130** has been customized so that on a work day morning the user wishes to receive the displayed information. Thus in the morning, any messages received overnight would be flagged, the user's favorite RSS sports feeds would be visible, today's weather forecast would be available, and the current traffic conditions between the user's home and office would be graphically depicted. User personalization settings may be maintained as persistent storage on device **130** when appropriate, or in a user profile which is maintained and updated in real-time in a server-side data base.

Push Capable Systems

In another embodiment system **100** or **200** is a push-capable system. As an example, of such systems, short codes may be applied to cereal boxes and beverage containers, and SMS text fields can be applied to promotional websites. In either case, a user of device **130** can text the short code or text field to an SMS server, which then serves the appropriate Application link back to device **130**.

FIG. **8** is a schematic of an embodiment of a push enabled system **800**. System **800** is generally similar to system **100** or **200**. Device **130** is shown as part of a schematic of a push capable system **800** in FIG. **8**. System **800** includes a website system **801** hosting a website **801**, a server **803** and a content server **805**. System **801** is connected to servers **803** and/or **805** through the Internet. Server **803** is generally similar to server **120**, servers **805** is generally similar to server **140**.

In one embodiment, a user sets up a weekly SMS update from website system **801**. System **801** provides user information to server **803**, which is an SMS server, when an update is ready for delivery. Server **803** provides device **130** with an SMS indication that the subscribed information is available and queries the user to see if they wish to receive the update. Website **801** also provides content server **805** with the content of the update. When a user of device **130** responds to the SMS query, the response is provided to content server **805**, which provides device **130** with updates including the subscribed content.

In an alternative embodiment of system **800**, server **803** broadcasts alerts to one or more devices **130**, such as a logical group of devices. The user is notified in real-time of the pending alert, and can view and interact with the massage without interrupting the current Application.

FIG. **9** is a schematic of an alternative embodiment of a push enabled system **900**. System **900** is generally similar to system **100**, **200**, or **800**. In system **900** a user requests information using an SMS code, which is delivered to device **130**. System **900** includes a promotional code **901**, a third-party server **903**, and content server **805**. Server **803** is connected to servers **803** and/or **805** through the Internet, and is generally similar to server **120**.

A promotional code **901** is provided to a user of device **130**, for example and without limitation, on print media, such as on a cereal box. The use of device **130** sends the code server **903**. Server **903** then notifies server **805** to provide certain information to device **130**. Server **805** then provides device **130** with the requested information.

Device Routines

Device routines **114** may include, but are not limited to: an authoring tool SDK for custom code development including full set of Java APIs to make it easy to add extensions and functionality to mobile Applications and tie Applications to back-end databases through the content server **140**; an expanding set of web services **250** available through the authoring tool SDK; a web services interface to SOAP/XML enabled web services; and an RSS/Atom and RDF feed collector **1010** and content gateway **1130**.

Authoring Tool SDK for Custom Code Development Including Full Set of Java APIs

In one embodiment, authoring platform **110** SDK is compatible for working with various integrated development environments (IDE) and popular plug ins such as J2ME Polish. In one embodiment the SDK would be another plug in to these IDEs. A large and powerful set of APIs and interfaces are thus available through the SDK to permit the seamless extension of any Application to back end business logic, web services, etc. These interfaces and APIs may also support listeners and player-side object operations.

There is a large set of listeners that expose both player-side events and dynamically linked server side data base events. Some examples of player side events are: player-side time based event, a page entry event, player-side user interactions and player-side object status. Examples of server-side data base events are when a particular set of linked data base field values change, or some filed value exceeds a certain limit, etc.

A superset of all authoring tool functionality is available through APIs for layer-side object operations. These include, but are not limited to: page view level APIs for inserting, replacing, and or modifying any page object; Object Level APIs for modifying any attribute of existing objects, adding definitions to attributes, and adding, hiding or replacing any object.

Authoring Tool SDK Available Web Services

The APIs permit, without limit, respond, with or without relying on back-end business logic, that is, logic that what an enterprise has developed for their business, to any player-side event or server-side dynamically linked data-base, incorporating any open 3rd party web service(s) into the response.

RSS/ATOM and RDF Feed Conversion Web Service

FIG. **10** is a schematic of one embodiment a system **1000** having a feed collector **1010**. System **1000** is generally similar to system **100**, **200**, **800**, or **900**. Feed collector **1010** is a server side component of system **100** that collects RSS, ATOM and RDF format feeds from various sources **1001** and aggregates them into a database **1022** for use by the Applications built using authoring platform **110**.

Feed collector **1010** is a standard XML DOM data extraction process, and includes Atom Populator Rule **1012**, RSS Populator Rule **1013**, RDF Populator Rule **1014**, and Custom Populator Rule **1016**, DOM XML Parsers **1011**, **1015**, and **1017**, Feed Processed Data Writer **1018**, Custom Rule Based Field Extraction **1019**, Rule-based Field Extraction **1020**, Channel Data Controller **1021**, and Database **1022**.

The feed collector is primarily driven by two sets of parameters: one is the database schema (written as SQL DDL) which defines the tables in the database, as well as parameters for each of the feeds to be examined. The other is the feed collection rules, written in XML, which can be used to customize the information that is extracted from the feeds. Each of the feeds is collected at intervals specified by the feed parameter set in the SQL DDL.

Feed collector **1010** accepts information from ATOM, RDF or RSS feed sources **1001**. Using a rules-based populator, any of these feeds can be logically parsed, with any type of data extraction methodology, either by using supplied rules, or by the author defining their own custom extraction rule. The rules are used by the parser to parse from the feed sources, and the custom rule base field extraction replaces the default rules and assembles the parsed information into the database

In particular, Atom Populator Rule **1012**, RSS Populator Rule **1013**, RDF Populator Rule **1014**, Custom Populator Rule **1016**, and DOM XML Parsers **1011**, **1015**, and **1017** are parse information from the feeds **1001**, and Feed Processed Data Writer **1018**, Custom Rule Based Field Extraction **1019**, Rule-based Field Extraction **1020**, and Channel Data Controller **1021**, supply the content of the feeds in Database **1022**, which is accessible through content server **140**.

FIG. **11** is a schematic of an embodiment of a system **1100** having a Mobile Content Gateway **1130**. System **1100** is generally similar to system **100**, **200**, **800**, **900**, or **1000**. System **1100** includes an SDK **1131**, feed collector **1010**,

database listener **1133**, transaction server **1134**, custom code **1135** generated from the SDK, Java APIs, Web Services **1137**, and PDL snippets compacted objects **1139**. System **1100** accepts input from Back End Java Code Developer **1120** and SOAP XML from Web Services **1110**, and provides dynamic content to server **140** and Players to devices **130**.

In one embodiment authoring platform **110** produces a Server-side PDL (SPDL) at authoring time. The SPDL resides in server **120** and provides a logical link between the Application's UI attributes and dynamic content in database **1022**. When a user of device **130** requests dynamic information, server **120** uses the SPDL to determine the link required to access the requested content.

In another embodiment Web Services **1137** interface directly with 3rd party Web Services **1110**, using SOAP, REST, JAVA, JavaScript, or any other interface for dynamically updating the attributes of the Application's UI objects.

XSP Web Pages as a Web Service

In one embodiment, a PDL for a page is embedded within an HTML shell, forming one XSP page. The process of forming XSP includes compressing the description of the page and then embedding the page within an HTML shell.

In another embodiment, a PDL, which contains many individual page definitions, is split into separate library objects on the server, so that each page can to presented as a PDL as part of a Web Service.

Prior to compression the code has already been transformed so that there are no dependencies on the original programming language (Java), and The code and data have been reduced by 4 to 10 times.

Compression has two distinct phases. The first takes advantage of how the primitive representations had been assembled, while the second utilizes standard LZ encoding.

The final result is an overall reduction of 40 to 100 times the original size as represented by Java serialized objects.

One embodiment for compacting data that may be used is described in Rempell. In that patent the compressed data is described as being a database. The terminology used here is a PDL, that is the "internal database" of Rempell is equivalent to the PDL of the present Application.

In Rempell, a process for compacting a "database" (that is, generating a compact PDL) is described, wherein data objects, including but not limited to, multi media objects such as colors, fonts, images, sound clips, URLs, threads, and video, including multi level animation, transformation, and time line are compacted. As an extension to Rempell in all cases these objects are reduced and transformed to Boolean, integer and string arrays.

The compression technique involves storing data in the smallest arrays necessary to compactly store web page information. The technique also includes an advanced form of delta compression that reduces integers so that they can be stored in a single byte, a as high water marks.

Thus, for example, the high water mark for different types of data comprising specific web site settings are stored in a header record as Boolean and integer variables and URL and color objects. Data that defines web page, paragraph, text button, and image style and text button, image and paragraph high watermark settings can be stored in one-dimensional arrays as Boolean, integer and string variables and URL, font, image or thread objects at. The URL, color, font, image and thread objects can also be created as required

Data that defines text button, image, paragraph, or other parent objects and paragraph line high watermark settings can be stored in two-dimensional arrays (by web page and by object number) as Boolean, integer, string, floating point variables and URLs. Again, the URL, color, font, image, audio clip, video clip, text area and thread objects can also be created as required. Data that defines a paragraph line and paragraph line segment high watermarks can be stored in three-dimensional arrays (by web page, by paragraph number, and by line number) as Boolean, integer or string variables. Again, the URL, color or font objects can be created as required. Data that defines a paragraph line segment can be stored into four-dimensional arrays (by web page, by paragraph number, by line number and by line number segment) as Boolean, integer or string variables or URL, color and font objects.

As a data field is added, changed or deleted, a determination is made at on whether a value for a given high watermark needs to be changed. If so, it is updated. As a specific method in the build engine is called, a determination is made on whether a feature flag needs to be set. For example, if a particular JAVA method is called, which requires an instance of a certain JAVA Class to be executed by the run time engine, then that JAVA Class is flagged, as well as any supporting methods, variables and/or object definitions.

In one implementation, the header record, the style record, the web page record, and the object records, are carefully defined in a specific order, written in that order, and explicitly cast by object type when read by the run time engine. Exception handling can be implemented to recover from any errors. This helps assure that data integrity is maintained throughout the build and run time processes.

Also described in Rempell is the "run generation process." This is equivalent generating a Player in the present application. This process starts when the build process detects that the user is finished defining the web site (user has saved the web site and invokes the run generation process), and concludes with the actual uploading of all the necessary web site run time files to the user's server.

In one embodiment, the PDL includes a first record, a "Header" record, which contains can include the following information:

1: A file format version number, used for upgrading database in future releases.

2: The default screen resolution, in virtual pixels, for both the screen width and height. This is usually set to the web designer's screen resolution, unless overwritten by the user.

3: Whether the Application is a web site.

4: Virtual web page size settings. A calculation is performed by the build engine method, in order to calculate what the maximum web page length is, after reformatting all paragraphs on all internal web pages, based on the default screen resolution.

5: Web page and styles high watermarks.

6: The Websitename.

As new web pages or new objects are created by the user, or as text is added to or deleted from a paragraph, or as new styles are created or deleted, appropriate high watermarks are set, in order to show the current number of each of these entities. Thus, the values for the number of active web pages and the number of text button, image, paragraph or other styles are written as high watermarks in the header. The high watermarks for the number of text button, image, paragraph or other objects that exist for each web page, the number of lines for each paragraph object, and the number of line segments for each paragraph line are written within the body of the PDL, and used as settings for each of the loops in the four-dimensional data structure. Because no structural limits are set on the number of web pages, objects per web page, styles, or paragraph size, these high watermarks greatly reduce the external database file size, and the time it takes for the run time engine to process the data stored in its database.

The settings for all paragraph, text button and image styles are then written as a style record based on their high watermark. This data includes Boolean and integer variables, and font and color objects, written as a one-dimensional array, based on the high watermark values for the number of styles that exist.

The body of the PDL is then written. All Boolean values are written inside a four-dimensional loop. The outside loop contains the Boolean values used to define web pages (i.e. a one-dimensional array definition) as well as the high watermarks for the number of text button, image, paragraph or other objects per web page, with the loop set at the high watermark which defines the number of existing web pages for this web site structure. The second level consists of three or more two dimensional loops with the loops set to the high watermarks defining the actual number of text button, image, and paragraph or other objects that appear on any given web page and contains the values used to define web page objects ((i.e. a two-dimensional array definition; web page number by object number). Included within the loop for paragraph objects are the high watermarks for the number of lines for each paragraph object. The third loop is set by the high watermark defining the actual number of paragraph lines that for all paragraphs on any web page and contains the values used to define paragraph lines (i.e. a three-dimensional array definition; web page number by object number by paragraph line.) Included within the loop for paragraph lines are the high watermarks for the number of line segments for each paragraph line. The inner most loop is set by the high watermarks defining the number of line segments per paragraph line and contains the values used to define paragraph line segments (i.e. a four-dimensional array definition; web page number by object number by paragraph line by paragraph line segment).

All integer values are written inside a four-dimensional loop. Their four loops are controlled by the same high watermark settings as used for the Boolean records, and they describe the same logical entities.

Multimedia objects are written inside a two-dimensional loop. They include URL, color, and font objects, and can include other types of objects. A URL object is the encoded form of a URL Address, used by a web browser or a JAVA method to access files and web addresses. All multimedia objects must be serialized before they can be written. This means that the objects are converted into a common external definition format that can be understood by the appropriate deserialization technique when they are read back in and cast into their original object structure. The outside loop contains web page related objects, and the inner loop contains image, text button, paragraph, etc. related URL, color, and font objects. The outer loop is defined by the web page high watermark and the inner loops by the high watermarks for the actual number of text button, image, paragraph or other objects on a web page.

String records are written inside a four-dimensional loop. The outer loop may be empty. The second loop can include the string values for text button objects, audio and video filenames, and audio and video channel names. The third loop contains values for paragraph line related data, and the innermost loop contains the values for paragraph line segment definitions. The string records are controlled by the same high watermarks as those used for Boolean and integer records. String records are stored utilizing an appropriate field delimiter technology. In one implementation, a UTF encoding technology that is supported by JAVA is utilized.

Single and double floating-point, and long integer records are written inside a two-dimensional loop. The outer loop may be empty. The inner loop contains mathematical values required for certain animations and image processing algorithms. The single and double floating-point, and long integer records are controlled by the same high watermarks as those used for Boolean and integer records.

In one embodiment, a versionizing program analyzes the feature flags, and only those variable definitions, defined in the "Main" object class, relating to the object classes and methods that will be executed at run time, are extracted. All references to object classes that will be called at run time are extracted, creating the source code for the run engine "Main" object class that is ready for compilation.

All external image, video and audio files are resolved. The external references can be copied to designated directories, either on the user's local disk or file server. The file Pathnames can be changed to reflect these new locations. During the installation of the build tools, the necessary class libraries are either installed on the local system or made available on the server where the build tools can be optionally located. The necessary environmental variables are set to permit normal access to the required class libraries.

The customized run engine and a library of the referenced run time classes are compiled and converted into byte code. Finally, the run time engine for the web site is created. The required set of class objects required at run time is flagged for inclusion into the CAB/JAR file.

Next, an HTML Shell File (HSF) is constructed. The first step of this process is to determine whether the dynamic web page and object resizing is desired by testing the Application setting. If the Application was a web page, and thus requiring dynamic web page and object resizing, virtual screen resolution settings are placed in an appropriate HTML compliant string. If the Application is a banner or other customized Application, the absolute values for the run time object (applet size) height and width are placed in an appropriate HTML compliant string as absolute width and height values.

An analysis is made for the background definition for the first internal web page. If a background pattern is defined, an appropriate HTML compliant string for setting the HTML "background" to the same background image is generated. If the first web page definition is a color instead, then the RGB values from those colors are converted to hexadecimal and an appropriate HTML compliant String is generated setting the "bgcolor" to the required hexadecimal value. This process synchronizes the web page background with the background that will be drawn by the web browser when it first interprets the HSF.

Thereafter, a JAVA method generates HTML and JavaScript compliant strings, that when executed by a web browser, generate additional sets of HTML and JavaScript compliant strings that are again executed by the web browser. More specifically, if the Application required dynamic web page and object resizing then JavaScript and HTML compliant strings are generated so that, when interpreted by the web browser at the time the HTML Shell File is initialized, the screen resolution sensing JAVA applet (SRS) will be executed. JavaScript code is generated in order to enable JavaScript to SRS applet communication. In one implementation, the code is generated by performing the following functions:

1: Determine the current web browser type.

2: Load the SRS from either a JAR or CAB File, based on web browser type.

3: Enter a timing loop, interrogating when the SRS is loaded.

4: When the SRS returns an "available" status, interrogate the SRS, which will return the current screen and window's actual height and width.

5: Convert the virtual screen resolution settings into appropriate absolute screen width and height values.

Strings defining additional JavaScript code are generated that perform the following steps at the time the HSF is initialized by the web browser:

1: Generate HTML compliant strings that set the run time engine's applet size to the appropriate values.

2: Generate an HTML complaint string that contains a "param" definition for linking the run time engine to the PDL.

3: Generate an HTML complaint string, dependent upon the type of web browser, which causes the current web browser to load either the JAR or the CAB File(s).

4: Generate JavaScript Code compliant strings that create and dynamically write the applet size defining HTML strings utilizing the JavaScript "document.write" function. This dynamically created code causes the web browser to execute the run time engine, in the correctly sized window, from the correct JAR or CAB file, and linked to the external database.

The writing out the above-generated HTML and JavaScript compliant strings creates the HSF. The necessary security policy permissions are asserted, and a "Websitename".html file is created.

In one embodiment, the processes for creating the CAB and JAR Files is as follows. The image objects, if any, which were defined on the first internal web page are analyzed. If they are set to draw immediately upon the loading of the first web page, then they are flagged for compression and inclusion in the CAB and JAR Files. The feature flags are analyzed to determine which JAVA classes have been compiled. These class files are flagged for compression and inclusion in the library CAB and JAR Files. Strings that are BAT compliant definitions are created that will, when executed in DOS, create compressed CAB and JAR Files. These CAB and JAR Files contain the compressed versions of all necessary JAVA class files, image files, the "Websitename".class, customized run time engine file, and the "Websitename".dta database file. In one implementation of the invention, two BAT files are created. The first, when executed, will create a CAB/JAR file with the "Websitename".dta database file and the customized "main" run time engine, excluding all the image and button object animation, transformation, and image processing code. The second BAT file, when executed, will create a CAB/JAR file with all the library of all the referenced image and button object animation, transformation, and image processing code.

The necessary security policy permissions for file creation are then asserted, and "Websitename".bat and "Websitenamelib".bat files are written. The "Websitename".bat and "Websitename".bat files are then executed under DOS, creating compressed "Websitename".cab and "Websitenamelib".cab files and compressed "Websitename" jar and "Websitenamelib" jar files. The HTML Shell File and the JAR and CAB files are then, either as an automatic process, or manually, uploaded to the user's web site. This completes the production of an XSP page that may be accessed through a web browser.

Displaying Content on a Device

Decompression Management

Authoring platform **110** uses compaction to transform the code and data in an intelligent way while preserving all of the original classes, methods and attributes. This requires both an intelligent server engine and client (handset) Player, both of which fully understand what the data means and how it will be used.

The compaction technology described above includes transformation algorithms that deconstruct the logic and data into their most primitive representations, and then reassembles them in a way that can be optimally digested by further compression processing. This reassembled set of primitive representations defines the PDL of authoring platform **110**.

Prior to compression the code has already been transformed so that there are no dependencies on the original programming language (Java). The data is then compressed by first taking advantage of how the primitive representations had been assembled, and then by utilizing standard LZ encoding. The final result is an overall reduction of 40 to 100 times the original size as represented by Java serialized objects.

The Player, when preparing a page view for execution, decompresses and then regenerate the original objects, but this time in compliance with the programming APIs of device **130**. Specifically, device **130** operates on compacted image pages, one at a time. The cache manager retrieves, decompresses, and reassembles the compacted page images into device objects, which are then interpreted by device **130** for display on screen **137**.

Response Director

In one embodiment, system **100** includes a Response Director, which determines a user's handset, fetches the correct Application from different databases, and delivers a respective highly compressed Application in a PDL format over the air (OTA).

In one embodiment, the Response Director operates on a network connected computer to provide the correct Player to a given device based on the information the device sent to it. As an example, this may occur when a device user enters their phone number into some call-to-action web page. The response director is called and sends an SMS message to the device, which responds, beginning the recognition process.

FIG. **12** illustrates one embodiment of a system **1200** that includes a response director **210**, a user agent database **1201**, an IP address database **1203**, and a file database **1205**. System **1200** is generally similar to system **100**, **200**, **800**, **900**, **1000**, or **1100**.

Databases **1201**, **1203**, and **1205** may reside on server **120**, **210**, or any computer system in communication with response director **210**. System **1200**, any mobile device can be serviced, and the most appropriate Application for the device will be delivered to the device, based on the characteristics of the device.

User agent database **1201** includes user agent information regarding individual devices **130** that are used to identify the operating system on the device. IP address database **1203** identifies the carrier/operator of each device **130**. File database **1205** includes data files that may operate on each device **130**.

The following is an illustrative example of the operation of response director **210**. First, a device **1300** generates an SMS message, which automatically sends an http://stream that includes handset information and its phone number to response director **210**. Response director **210** then looks at a field in the http header (which includes the user agent and IP address) that identifies the web browser (i.e., the "User Agent"). The User Agent prompts a database lookup in user agent database **1201** which returns data including, but not limited to, make, model, attributes, MIDP 1.0 MIDP 2.0, WAP and distinguishes the same models from different countries. A lookup of the IP address in IP address **1203** identifies the carrier/operator.

File database **1205** contains data types, which may include as jad1, jad2, html, wml/wap2, or other data types, appropriate for each device **130**. A list of available Applications are returned to a decision tree, which then returns, to device **130**, the Application that is appropriate for the respective device.

For each file type, there is an attributes list (e.g., streaming video, embedded video, streaming audio, etc.) to provide enough information to determine what to send to the handset.

Response director **210** generates or updates an html or jad file populating this text file with the necessary device and network dependent parameters, including the Application dependent parameters, and then generate, for example, a CAB or JAD file which contains the necessary Player for that device. For example, the jad file could contain the operator or device type or extended device-specific functions that the player would then become aware of.

If there is an Application that has a data type that device **130** cannot support, for example, video, response director **210** sends an alternative Application to the handset, for example one that has a slide show instead. If the device cannot support a slide show, an Application might have text and images and display a message that indicates it does not support video.

Another powerful feature of response director **210** is its exposed API from the decision tree that permits the overriding of the default output of the decision tree by solution providers. These solution providers are often licensees who want to further refine the fulfillment of Applications and Players to specific devices beyond what the default algorithms provide. Solution providers may be given a choice of Applications and then can decide to use the defaults or force other Applications.

Authoring platform **110** automatically scales Applications at publishing time to various form factors to reduce the amount of fragmentation among devices, and the Response Director serves the appropriately scaled version to the device. For example, a QVGA Application will automatically scale to the QCIF form factor. This is important because one of the most visible forms of fragmentation resides in the various form factors of wireless, and particularly mobile, devices, which range from 128×128, 176×208, 240×260, 220×220, and many other customized sizes in between.

FIG. **13** is a schematic of an embodiment of a system **1300**. System **1300** is generally similar to system **1200**. System **1300** is an overview of the entire Player fulfillment process, starting with the generation of players during the player build process.

System **1300** includes response director **210**, a device characteristics operator and local database **1301**, a player profile database **1303** and a player build process **1305**, which may be authoring platform **110**.

As an example of system **1300**, when response director **210** receives an SMS message from device **130**, the response director identifies the device characteristics operator and locale from database **1301** and a Player URL from database **1303** and provides the appropriate Player to the device.

In another embodiment, Player P extend the power of response director **210** by adapting the Application to the resources and limitations of any particular device. Some of these areas of adaptation include the speed of the device's microprocessor, the presence of device resources such as cameras and touch screens. Another area of adaptation is directed to heap, record store and file system memory constraints. In one embodiment, the Player will automatically throttle down an animation to the frame rate that the device can handle so that the best possible user experience is preserved. Other extensions include device specific facilities such as location awareness, advanced touch screen interactions, push extensions, access to advanced phone facilities, and many others

Memory Management

In one embodiment, Player P includes a logical page virtual memory manager. This architecture requires no supporting hardware and works efficiently with constrained devices. All page view images, which could span multiple Applications, are placed in a table as highly compacted and compressed code. A typical page view will range from 500 bytes up to about 1,500 bytes. (See, for example, the Rempell patent) When rolled into the heap and instantiated this code increases to the more typical 50,000 up to 250,000 bytes. Additional alert pages may also be rolled into the heap and superimposed on the current page view. Any changes to any page currently downloaded are placed in a highly compact change vector for each page, and rolled out when the page is discarded. Note that whenever an Application is visited that had previously been placed in virtual memory the Server is interrogated to see if a more current version is available, and, if so, downloads it. This means that Application logic can be changed in real-time and the results immediately available to mobile devices.

To operate efficiently with the bandwidth constraints of mobile devices, authoring platform **110** may also utilize anticipatory streaming and multi-level caching. Anticipatory streaming includes multiple asynchronous threads and IO request queues. In this process, the current Application is scanned to determine if there is content that is likely to be required in as-yet untouched page views. Anticipatory streaming also looks for mapping Applications, where the user may zoom or pan next so that map content is retrieved prior to the user requesting it. For mapping applications, anticipatory streaming downloads a map whose size is greater than the map portal size on the device and centered within the portal. Any pan operation will anticipatory stream a section of the map to extend the view in the direction of the pan while, as a lower priority, bring down the next and prior zoom levels for this new geography. Zooming will always anticipatory stream the next zoom level up and down.

Multi-level caching determines the handset's heap through an API, and also looks at the record store to see how much memory is resident. This content is placed in record store and/or the file system, and may, if there is available heap, also place the content there as well. Multi-level caching permits the management of memory such that mobile systems best use limited memory resources. Multi-level caching is a memory management system with results similar to embedding, without the overhead of instantiating the content. In other words, with multi-level caching, handset users get an "embedded" performance without the embedded download. Note that when content is flagged as cacheable and is placed in persistent storage, a digital rights management (DRM) solution will be used.

One embodiment of each of the methods described herein is in the form of a computer program that executes on a processing system. Thus, as will be appreciated by those skilled in the art, embodiments of the present invention may be embodied as a method, an apparatus such as a special purpose apparatus, an apparatus such as a data processing system, or a carrier medium, e.g., a computer program product. The carrier medium carries one or more computer readable code segments for controlling a processing system to implement a method. Accordingly, aspects of the present invention may take the form of a method, an entirely hardware embodiment, an entirely software embodiment or an embodiment combining software and hardware aspects. Furthermore, the present invention may take the form of carrier medium (e.g., a computer program product on a computer-readable storage medium) carrying computer-readable program code segments embodied in the medium. Any suitable computer readable medium may be used including a magnetic storage device such as a diskette or a hard disk, or an optical storage device such as a CD-ROM.

It will be understood that the steps of methods discussed are performed in one embodiment by an appropriate processor (or processors) of a processing (i.e., computer) system executing instructions (code segments) stored in storage. It will also be understood that the invention is not limited to any particular implementation or programming technique and that the invention may be implemented using any appropriate techniques for implementing the functionality described herein. The invention is not limited to any particular programming language or operating system. It should thus be appreciated that although the coding for programming devices has not be discussed in detail, the invention is not limited to a specific coding method. Furthermore, the invention is not limited to any one type of network architecture and method of encapsulation, and thus may be utilized in conjunction with one or a combination of other network architectures/protocols.

Reference throughout this specification to "one embodiment," "an embodiment," or "certain embodiments" means that a particular feature, structure or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, appearances of the phrases "in one embodiment," "in an embodiment," or "in certain embodiments" in various places throughout this specification are not necessarily all referring to the same embodiment. Furthermore, the particular features, structures or characteristics may be combined in any suitable manner, as would be apparent to one of ordinary skill in the art from this disclosure, in one or more embodiments.

Throughout this specification, the term "comprising" shall be synonymous with "including," "containing," or "characterized by," is inclusive or open-ended and does not exclude additional, unrecited elements or method steps. "Comprising" is a term of art which means that the named elements are essential, but other elements may be added and still form a construct within the scope of the statement. "Comprising" leaves open for the inclusion of unspecified ingredients even in major amounts.

Similarly, it should be appreciated that in the above description of exemplary embodiments, various features of the invention are sometimes grouped together in a single embodiment, figure, or description thereof for the purpose of streamlining the disclosure and aiding in the understanding of one or more of the various inventive aspects. This method of disclosure, however, is not to be interpreted as reflecting an intention that the claimed invention requires more features than are expressly recited in each claim. Rather, as the following claims reflect, inventive aspects lie in less than all features of a single foregoing disclosed embodiment, and the invention may include any of the different combinations embodied herein. Thus, the following claims are hereby expressly incorporated into this Mode(s) for Carrying Out the Invention, with each claim standing on its own as a separate embodiment of this invention.

Thus, while there has been described what is believed to be the preferred embodiments of the invention, those skilled in the art will recognize that other and further modifications may be made thereto without departing from the spirit of the invention, and it is intended to claim all such changes and modifications as fall within the scope of the invention. For example, any formulas given above are merely representative of procedures that may be used. Functionality may be added or deleted from the block diagrams and operations may be interchanged among functional blocks. Steps may be added or deleted to methods described within the scope of the present invention.

We claim:

1. A system for generating code to provide content on a display of a device, said system comprising:

computer memory storing a registry of:

a) symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network, where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service, and

b) the address of the web service;

an authoring tool configured to:

define a user interface (UI) object for presentation on the display, where said UI object corresponds to the web component included in said registry selected from the group consisting of an input of the web service and an output of the web service,

access said computer memory to select the symbolic name corresponding to the web component of the defined UI object,

associate the selected symbolic name with the defined UI object,

produce an Application including the selected symbolic name of the defined UI object, where said Application is a device-independent code, and

produce a Player, where said Player is a device-dependent code;

such that, when the Application and Player are provided to the device and executed on the device, and when a user of the device provides one or more input values associated with an input symbolic name to an input of defined UI object,

1) the device provides the user provided one or more input values and corresponding input symbolic name to the web service,

2) the web service utilizes the input symbolic name and the user provided one or more input values for generating one or more output values having an associated output symbolic name,

3) said Player receives the output symbolic name and corresponding one or more output values and provides instructions for a display of the device to present an output value in the defined UI object.

2. The system of claim **1**, where said registry includes definitions of input and output related to said web service.

3. The system of claim **1**, where said web component is a text chat, a video chat, an image, a slideshow, a video, or an RSS feed.

4. The system of claim **1**, where said UI object is an input field for a chat.

5. The system of claim **1**, where said UI object is an input field for a web service.

6. The system of claim **1**, where said UI object is an input field usable to obtain said web component, where said input field includes a text field, a scrolling text box, a check box, a drop down-menu, a list menu, or a submit button.

7. The system of claim **1**, where said web component is an output of a web service, is the text provided by one or more simultaneous chat sessions, is the video of a video chat session, is a video, an image, a slideshow, an RSS display, or an advertisement.

8. The system of claim **1**, where said authoring tool is further configured to:

define a phone field or list; and

generate code that, when executed on the device, allows a user to supply a phone number to said phone field or list.

9. The system of claim **1**, where said authoring tool is further configured to:

define a SMS field or list; and

generate code that, when executed on the device, allows a user to supply an SMS address to said SMS field or list.

10. The system of claim **1**,

where said code includes three or more codes, where one of said three or more codes is device specific, and where two of said three or more codes is device independent.

11. The system of claim **1**, where said code is provided over said network.

12. A method of displaying content on a display of a device utilizing a registry of one or more web components related to inputs and outputs of a web service obtainable over a network, where each web component includes a plurality of symbolic names of inputs and outputs associated with each web service, and where the registry includes: a) symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network, where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service, and

b) the address of the web service, said method comprising:

defining a user interface (UI) object for presentation on the display, where said UI object corresponds to a web component included in said registry selected from the group consisting of an input of the web service and an output of the web service;

selecting a symbolic name from said web component corresponding to the defined UI object;

associating the selected symbolic name with the defined UI object;

producing an Application including the selected symbolic name of the defined UI object, where said Application is a device-dependent code; and

producing a Player, where said Player is a device-dependent code;

such that, when the Application and Player are provided to the device and executed on the device, and when a user of the device provides one or more input values associated with an input symbolic name to an input of defined UI object,

1) the device provides the user provided one or more input values and corresponding input symbolic name to the web service,

2) the web service utilizes the input symbolic name and the user provided one or more input values for generating one or more output values having an associated output symbolic name,

3) said Player receives the output symbolic name and corresponding one or more output values and provides instructions for a display of the device to present an output value in the defined UI object.

13. The method of claim **12**, where said registry includes definitions of input and output related to said web service.

14. The method of claim **12**, where said web component is a text chat, a video chat, an image, a slideshow, a video, or an RSS feed.

15. The method of claim **12**, where said UI object is an input field for a chat.

16. The method of claim **12**, where said UI object is an input field for a web service.

17. The method of claim **12**, where said UI object is an input field usable to obtain said web component, where said input field includes a text field, a scrolling text box, a check box, a drop down-menu, a list menu, or a submit button.

18. The method of claim **12**, where said web component is an output of a web service, is the text provided by one or more simultaneous chat sessions, is the video of a video chat session, is a video, an image, a slideshow, an RSS display, or an advertisement.

19. The method of claim **12**, further comprising:
defining a phone field or list; and
generating code that, when executed on the device, allows a user to supply a phone number to said phone field or list.

20. The method of claim **12**, further comprising:
defining a SMS field or list; and
generating code that, when executed on the device, allows a user to supply an SMS address to said SMS field or list.

21. Previously Presented The method of claim **12**, and such that said Player interprets dynamically received, device-independent values of the web component defined in the Application.

22. The method of claim **12**, further comprising:
providing said Application and Player over said network.

23. A method of providing information to a device having a display from a web component of a web service to a device on a network, said method comprising:
accepting, on the device, a first code over the network, where said first code is device-dependent;
accepting, on the device, a second code over the network, where said second code is device-independent and includes a plurality of symbolic names of inputs and outputs associated with the web service; and
executing said first code on the device,
where the symbolic names are provided from a registry of one or more web components related to inputs and outputs of a web service obtainable over a network,

where the web service requires both an input symbolic name and one or more associated input values and returns one or more output values having an associated output symbolic name, and
where the registry includes
a) symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network, where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service, and
b) the address of the web service;
where said executing includes:
processing said symbolic names of the second code on the device,
transmitting processed instructions from the device to the web service, and
accepting a third code on the device over the network, where said third code is a device-independent third code including the output of the web component provided by the web service over the network and in response to the second code.

24. The method of claim **23**, where said third code is a text chat, a video chat, an image, a slideshow, a video, or an RSS feed.

25. The method of claim **23**, where said third code is an output of a web service, is the text provided by one or more simultaneous chat sessions, is the video of a video chat session, is a video, an image, a slideshow, an RSS display, or an advertisement.

26. The method of claim **23**, where said first code and said second code are generated using an authoring tool.

27. The method of claim **23**, where said first code is a Player.

28. The method of claim **23**, where said second code is an Application which includes one or more web components.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO.	: 9,063,755 B2	Page 1 of 1
APPLICATION NO.	: 12/936395	
DATED	: June 23, 2015	
INVENTOR(S)	: Rempell et al.	

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Title page, item 73, Assignee: Express Mobile, inc.,Novato, CA (US) - the “inc.” should be -- Inc. --.

In the claims

Column 39, line 18, Claim 21, delete the text reading “Previously Presented”.

Signed and Sealed this
Twentieth Day of October, 2015

Michelle K. Lee

Michelle K. Lee
Director of the United States Patent and Trademark Office

EXHIBIT D

US009471287B2

(12) **United States Patent**
Rempell et al.

(10) **Patent No.: US 9,471,287 B2**
(45) **Date of Patent: *Oct. 18, 2016**

(54) SYSTEMS AND METHODS FOR INTEGRATING WIDGETS ON MOBILE DEVICES

(71) Applicant: **Express Mobile, Inc.**, Novato, CA (US)

(72) Inventors: **Steven H. Rempell**, Novato, CA (US); **David Chrobak**, Clayton, CA (US); **Ken Brown**, San Martin, CA (US)

(73) Assignee: **Express Mobile, Inc.**, Novato, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **14/708,074**

(22) Filed: **May 8, 2015**

(65) **Prior Publication Data**

US 2015/0317130 A1 Nov. 5, 2015

Related U.S. Application Data

(63) Continuation of application No. 12/936,395, filed as application No. PCT/US2009/039695 on Apr. 6, 2009, now Pat. No. 9,063,755.

(60) Provisional application No. 61/123,438, filed on Apr. 7, 2008, provisional application No. 61/113,471, filed on Nov. 11, 2008, provisional application No. 61/166,651, filed on Apr. 3, 2009.

(51) **Int. Cl.**

G06F 3/048	(2013.01)
G06F 9/44	(2006.01)
H04L 29/08	(2006.01)
G06F 3/0484	(2013.01)
G06F 3/0482	(2013.01)
H04L 29/06	(2006.01)
H04L 12/58	(2006.01)

(52) **U.S. Cl.**
CPC ***G06F 8/34*** (2013.01); ***G06F 3/0482*** (2013.01); ***G06F 3/04842*** (2013.01); ***G06F 9/4443*** (2013.01); ***H04L 51/046*** (2013.01); ***H04L 65/60*** (2013.01); ***H04L 67/02*** (2013.01)

(58) **Field of Classification Search**
CPC G06F 3/048
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2004/0055017	A1	3/2004	Delpuch et al.	
2004/0163020	A1	8/2004	Sidman	
2004/0199614	A1 *	10/2004	Shenfield	H04L 29/06 709/220

(Continued)

OTHER PUBLICATIONS

Stina Nylander et al. "The Ubiquitous Interactor—Device Independent Access to Mobile Services" (Computer-Aided Design for User Interfaces IV, Proceedings of the Fifth International Conference on Computer-Aided Design of User Interfaces CADUI'2004, Jan. 2004, pp. 271-282).*

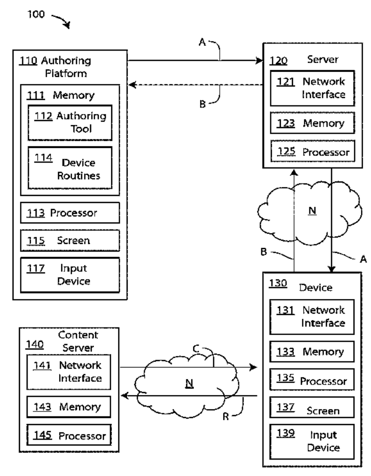
(Continued)

Primary Examiner — Jennifer To
Assistant Examiner — Xuyang Xia
(74) *Attorney, Agent, or Firm* — Steven R. Vosen

(57) **ABSTRACT**

Embodiments of a system and method are described for generating and distributing programming to mobile devices over a network. Devices are provided with Players specific to each device and Applications that are device independent. Embodiments include a full-featured WYSIWYG authoring environment, including the ability to bind web components to objects.

28 Claims, 18 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

2005/0149935	A1	7/2005	Benedetti	
2005/0273705	A1 *	12/2005	McCain	G06F 17/24 715/234
2006/0063518	A1	3/2006	Paddon et al.	

OTHER PUBLICATIONS

Stina Nylander et at. "The Ubiquitous Interactor—Device Independent Access to Mobile Services" (Computer-Aided Design for User Interfaces IV, Proceedings of the Fifth international Conference on Computer-Aided Design of User Interfaces CADUT2004, Jan. 2004, pp. 271-282).

International Search Report and Written Opinion —PCT/US2009/039695—Aug. 21, 2009.

International Preliminary Report on Patentability and Written Opinion—PCT/US2009/039695—Oct. 21, 2010.

Rempell et al, co-pending U.S. Appl. No. 14/708,087, filed May 8, 2015.

Rempell et al, co-pending U.S. Appl. No. 14/708,094, filed May 8, 2015.

Rempell et al, co-pending U.S. Appl. No. 14/708,097, filed May 8, 2015.

Rempell et al, co-pending U.S. Appl. No. 14/708,100, filed May 8, 2015.

Rempell et al, co-pending U.S. Appl. No. 14/708,108, filed May 8, 2015.

* cited by examiner

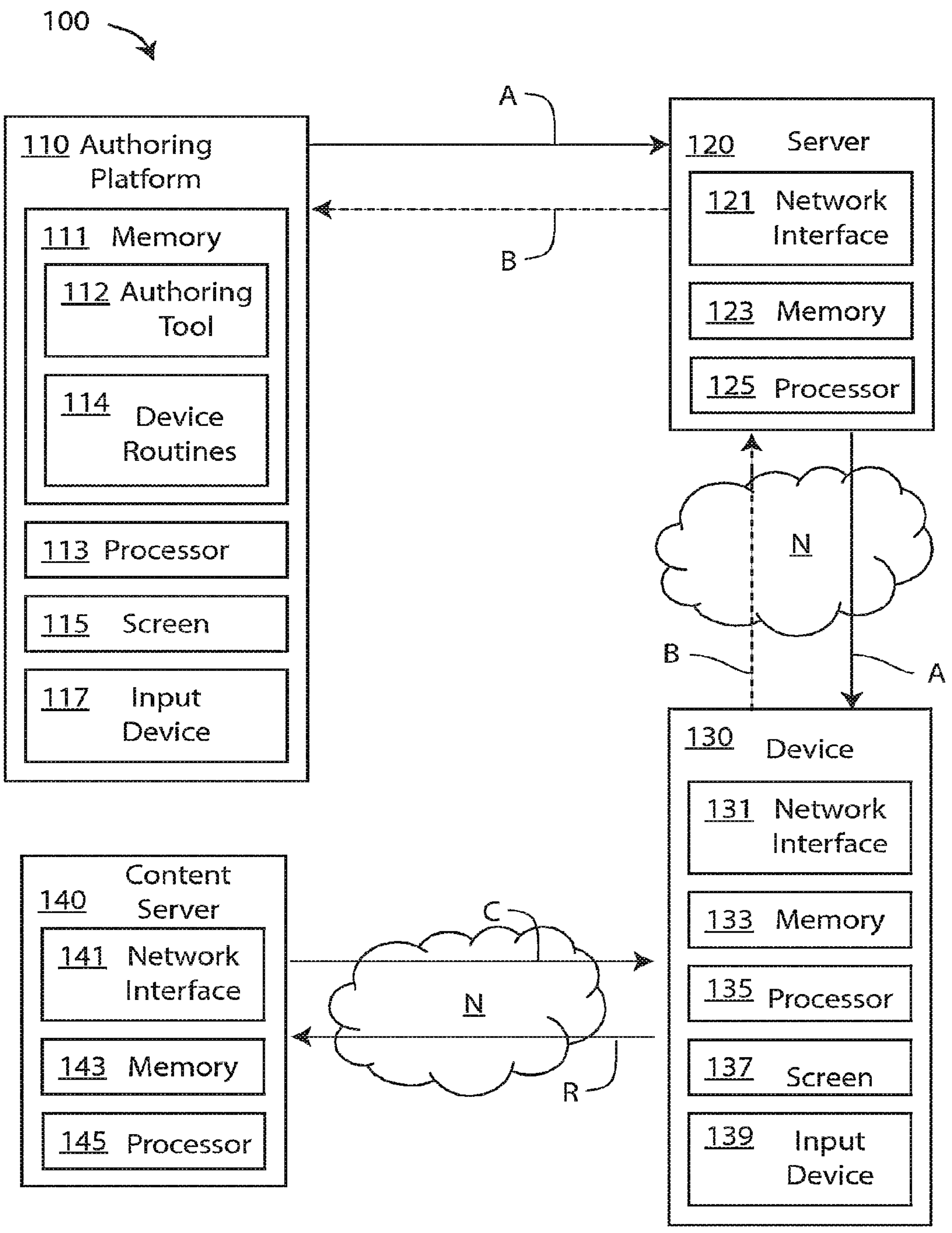

100
110 Authoring Platform
111 Memory
112 Authoring Tool
114 Device Routines
113 Processor
115 Screen
117 Input Device
A
B
120 Server
121 Network Interface
123 Memory
125 Processor
N
B
A
130 Device
131 Network Interface
133 Memory
135 Processor
137 Screen
139 Input Device
140 Content Server
141 Network Interface
143 Memory
145 Processor
C
N
R

FIG. 1A

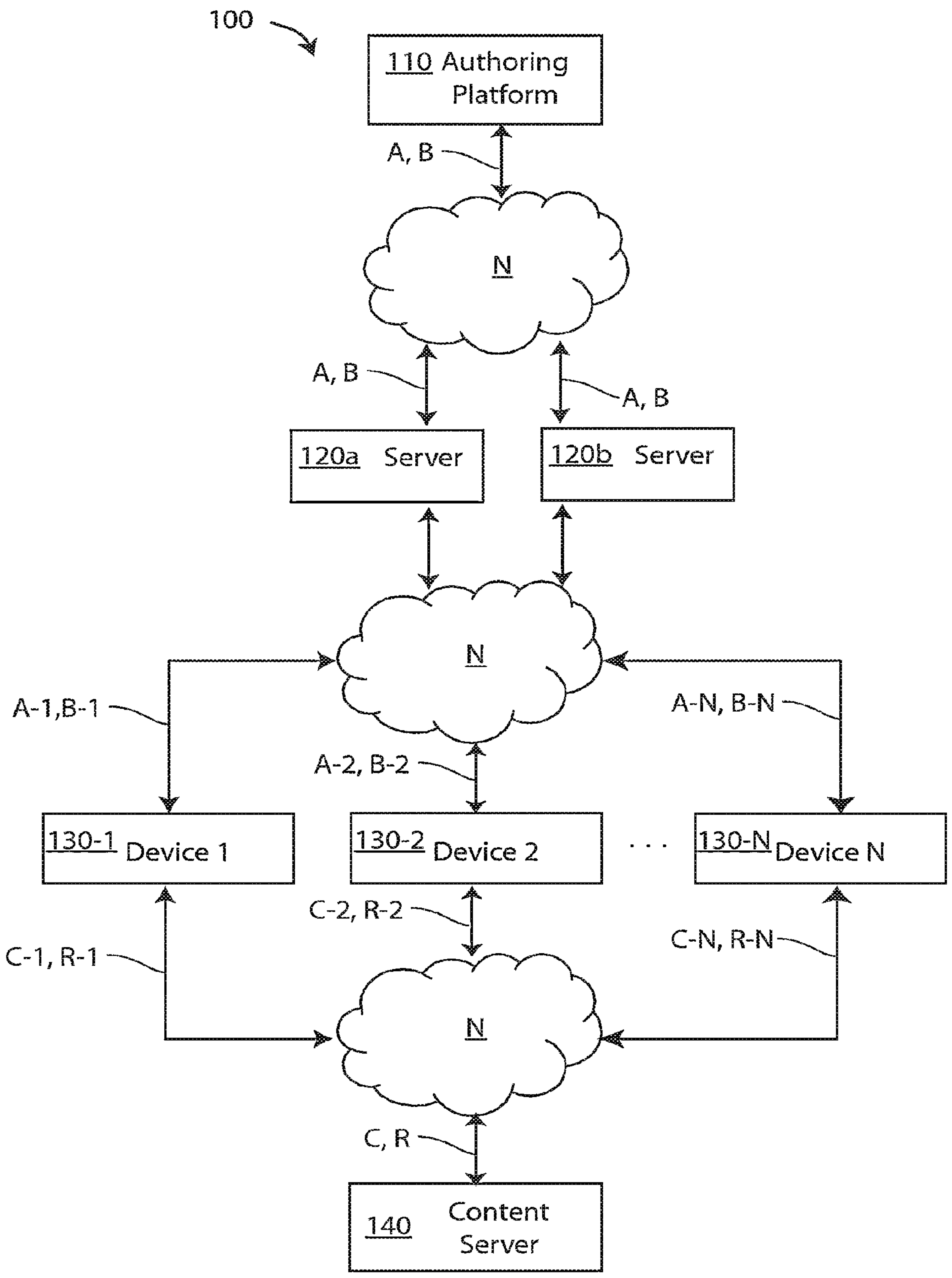

100
110 Authoring Platform
A, B
N
A, B
A, B
120a Server
120b Server
N
A-1,B-1
A-N, B-N
A-2, B-2
130-1 Device 1
130-2 Device 2
. . .
130-N Device N
C-2, R-2
C-1, R-1
C-N, R-N
N
C, R
140 Content Server

FIG. 1B

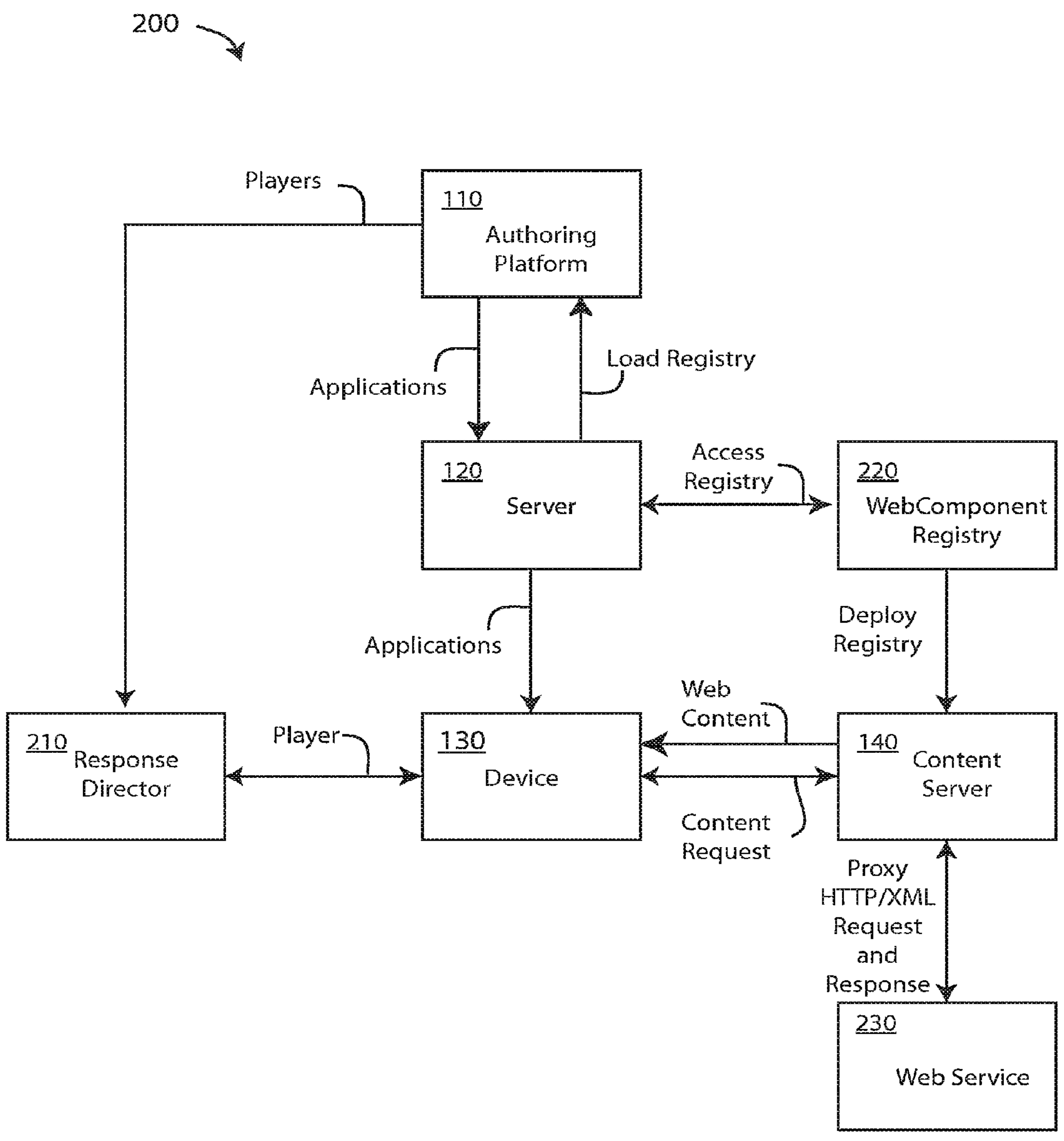

200
110 Authoring Platform
Players
Applications
Load Registry
120 Server
Access Registry
220 WebComponent Registry
Applications
Deploy Registry
Web Content
210 Response Director
Player
130 Device
140 Content Server
Content Request
Proxy HTTP/XML Request and Response
230 Web Service

FIG. 2A

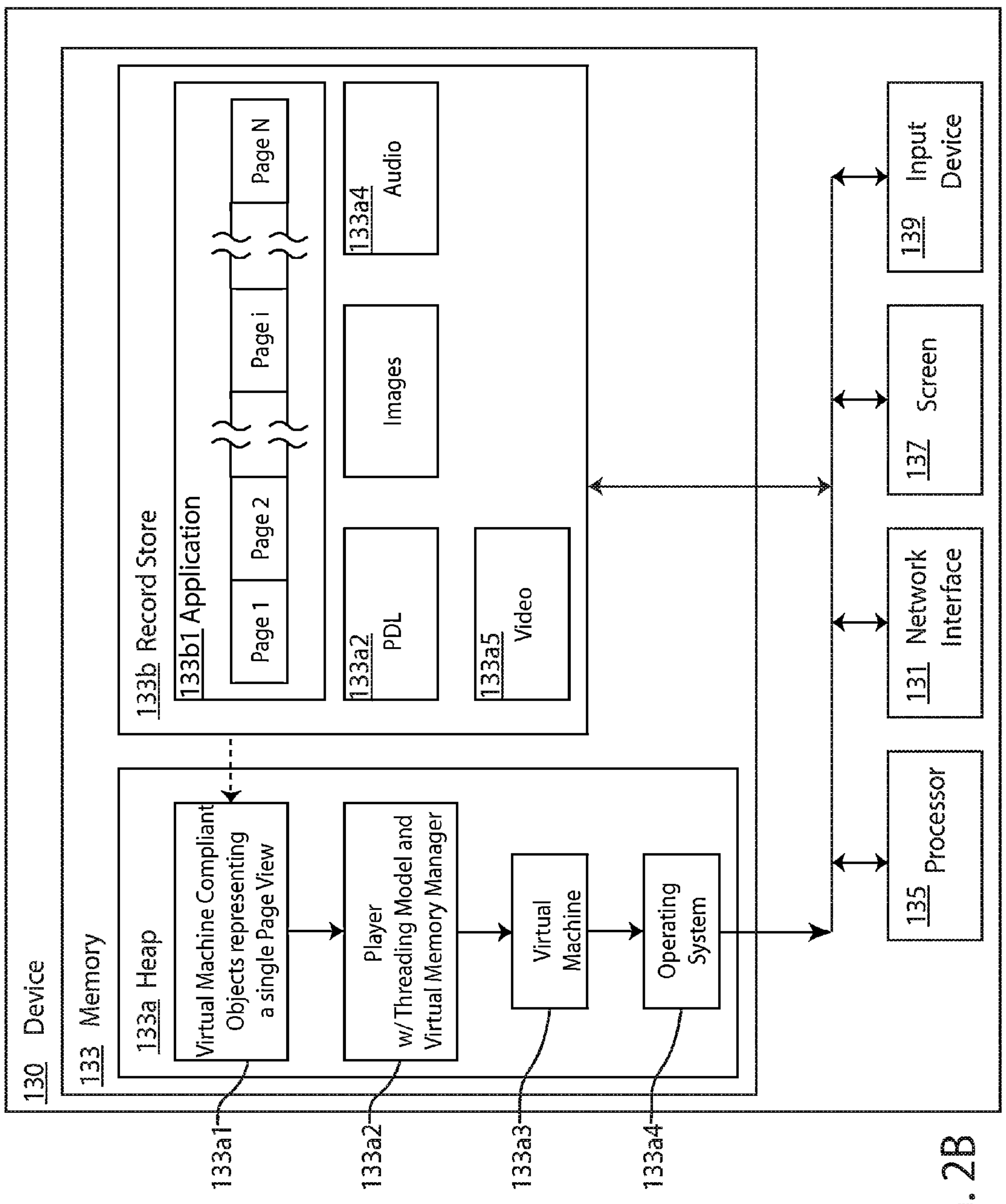

130 Device
133 Memory
133a Heap
Virtual Machine Compliant Objects representing a single Page View
Player w/ Threading Model and Virtual Memory Manager
Virtual Machine
Operating System
133b Record Store
133b1 Application
Page 1
Page 2
Page i
Page N
133a2 PDL
133a5 Video
Images
133a4 Audio
135 Processor
131 Network Interface
137 Screen
139 Input Device
133a1
133a2
133a3
133a4

FIG. 2B

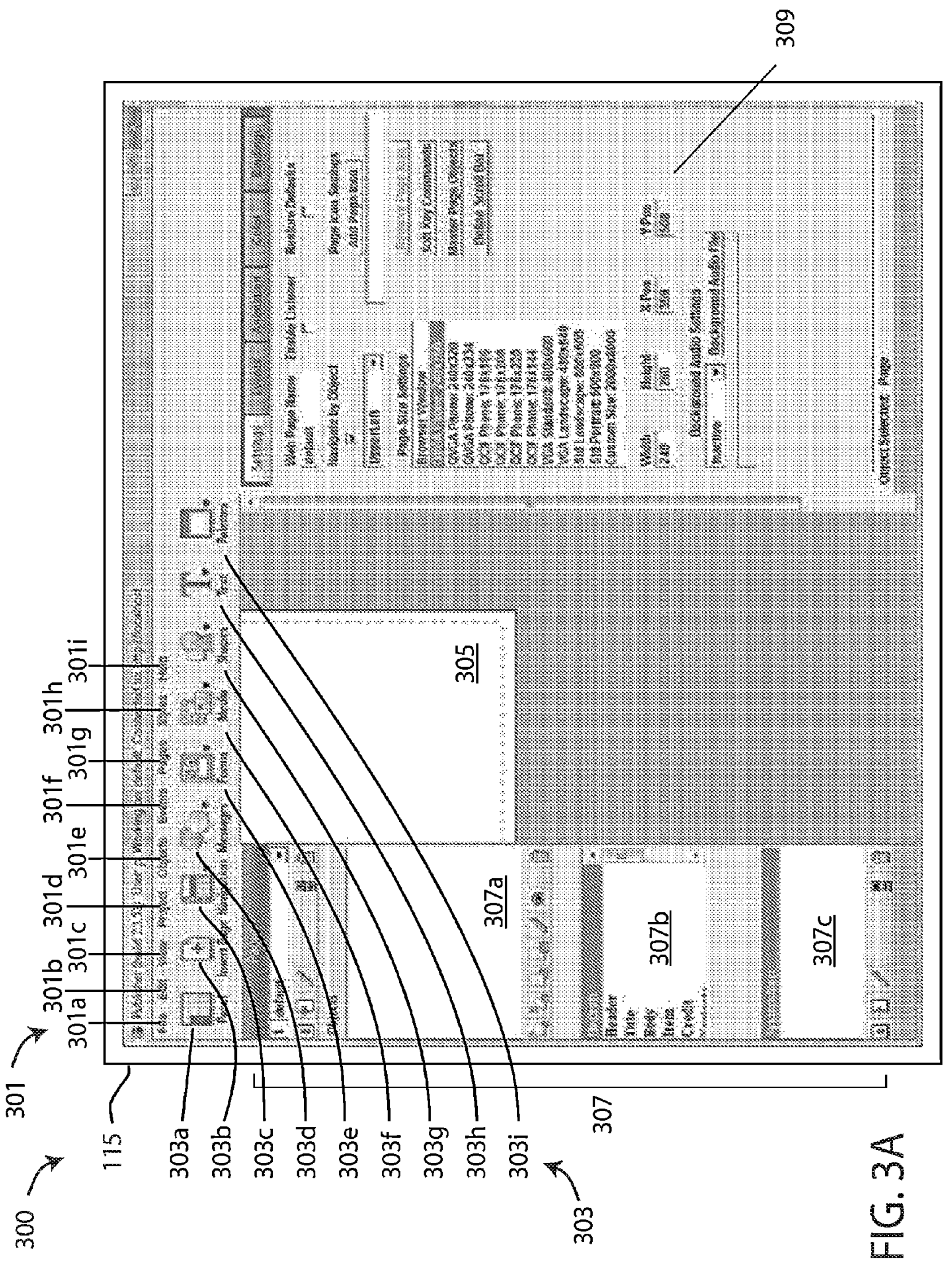

300
301
301a
301b
301c
301d
301e
301f
301g
301h
301i
115
303a
303b
303c
303d
303e
303f
303g
303h
303i
303
305
307
307a
307b
307c
309

FIG. 3A

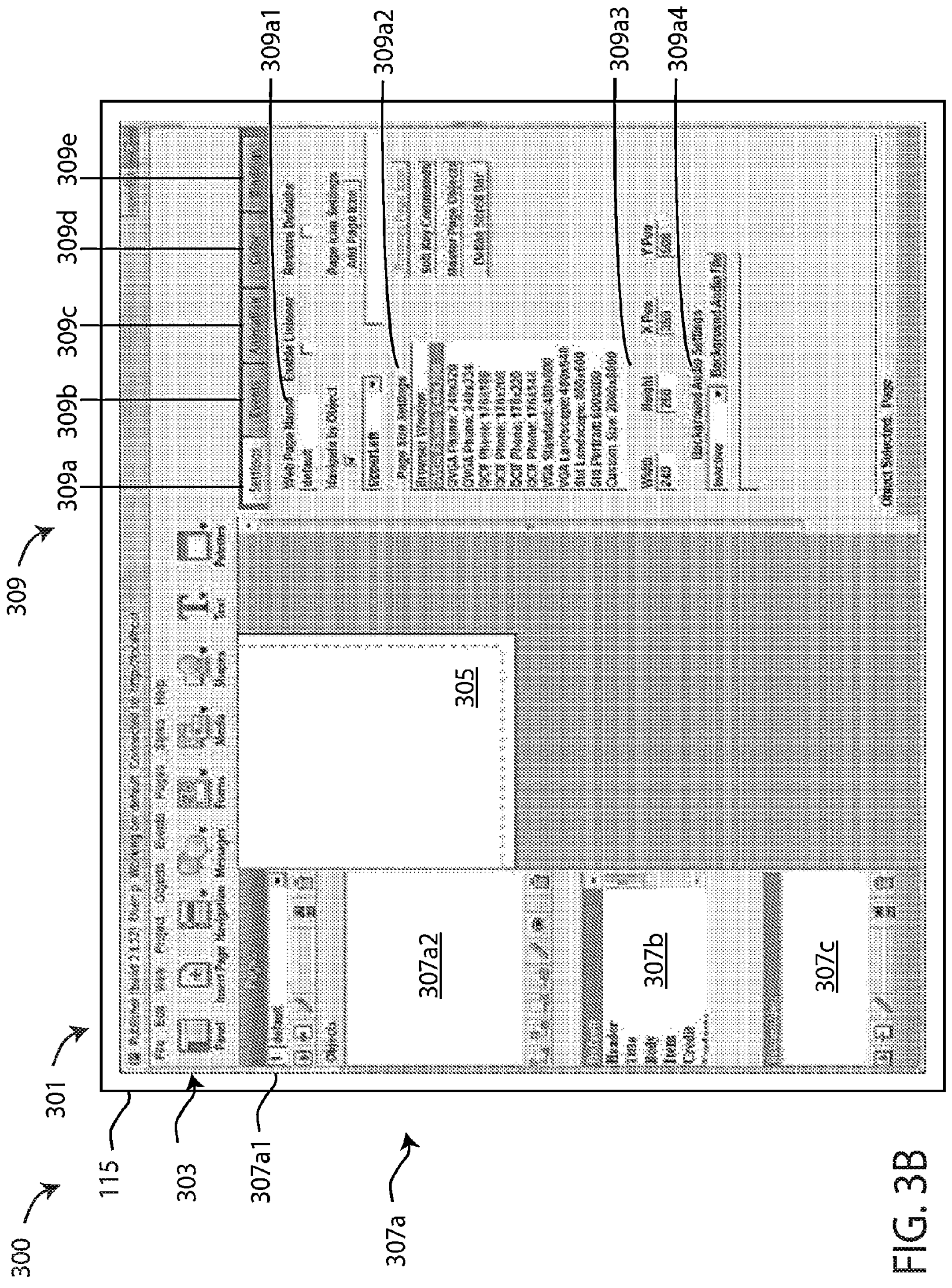
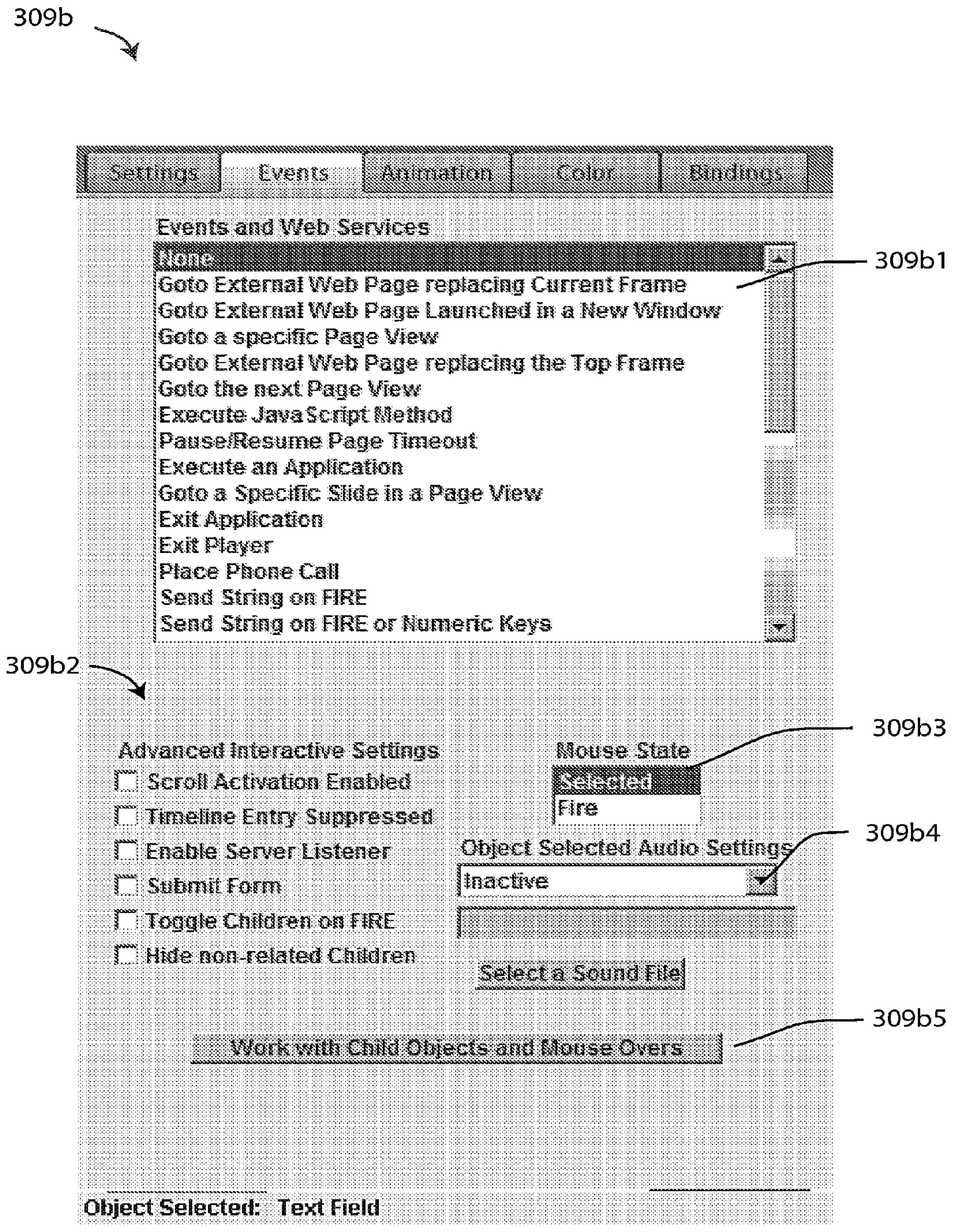

300
115
301
303
307a1
307a
309
309a
309b
309c
309d
309e
309a1
309a2
309a3
309a4
305
307a2
307b
307c

FIG. 3B



Events and Web Services

- None
- Goto External Web Page replacing Current Frame
- Goto External Web Page Launched in a New Window
- Goto a specific Page View
- Goto External Web Page replacing the Top Frame
- Goto the next Page View
- Execute JavaScript Method
- Pause/Resume Page Timeout
- Execute an Application
- Goto a Specific Slide in a Page View
- Exit Application
- Exit Player
- Place Phone Call
- Send String on FIRE
- Send String on FIRE or Numeric Keys

Advanced Interactive Settings

- ☐ Scroll Activation Enabled
- ☐ Timeline Entry Suppressed
- ☐ Enable Server Listener
- ☐ Submit Form
- ☐ Toggle Children on FIRE
- ☐ Hide non-related Children

Mouse State

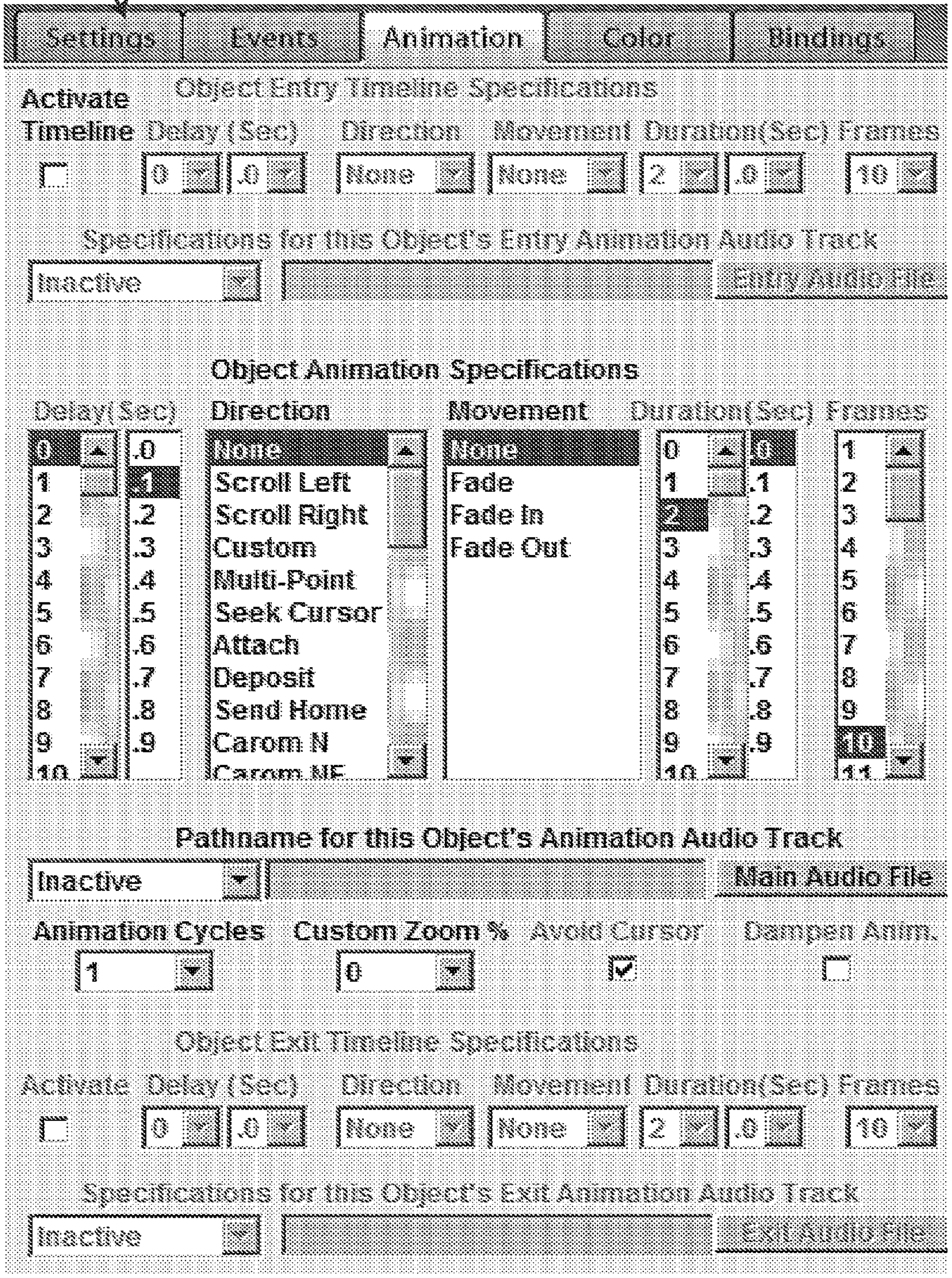
- Selected
- Fire

Object Selected Audio Settings: Inactive

Object Selected: Text Field

FIG. 3C

309c



Object Entry Timeline Specifications

Activate Timeline ☐ · Delay (Sec): 0 .0 · Direction: None · Movement: None · Duration(Sec): 2 .0 · Frames: 10

Specifications for this Object's Entry Animation Audio Track: Inactive · Entry Audio File

Object Animation Specifications

Delay(Sec)		Direction	Movement	Duration(Sec)		Frames
0	.0	None	None	0	0	1
1	.1	Scroll Left	Fade	1	.1	2
2	.2	Scroll Right	Fade In	2	.2	3
3	.3	Custom	Fade Out	3	.3	4
4	.4	Multi-Point		4	.4	5
5	.5	Seek Cursor		5	.5	6
6	.6	Attach		6	.6	7
7	.7	Deposit		7	.7	8
8	.8	Send Home		8	.8	9
9	.9	Carom N		9	.9	10
10		Carom NE		10		11

Pathname for this Object's Animation Audio Track: Inactive · Main Audio File

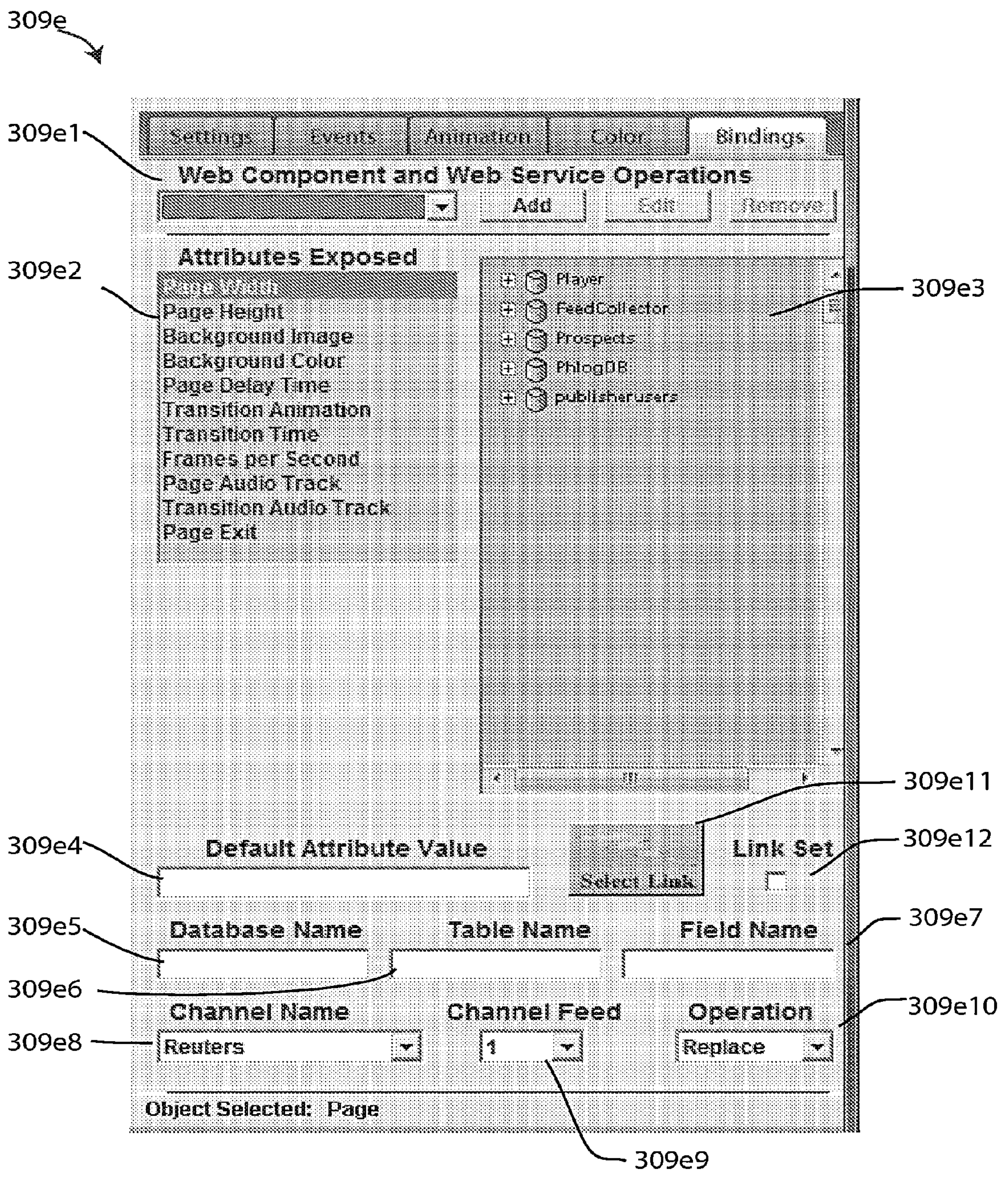
Animation Cycles: 1 · Custom Zoom %: 0 · Avoid Cursor: ☑ · Dampen Anim.: ☐

Object Exit Timeline Specifications

Activate ☐ · Delay (Sec): 0 .0 · Direction: None · Movement: None · Duration(Sec): 2 .0 · Frames: 10

Specifications for this Object's Exit Animation Audio Track: Inactive · Exit Audio File

FIG. 3D



Web Component and Web Service Operations

Attributes Exposed
Page Width
Page Height
Background Image
Background Color
Page Delay Time
Transition Animation
Transition Time
Frames per Second
Page Audio Track
Transition Audio Track
Page Exit

Default Attribute Value

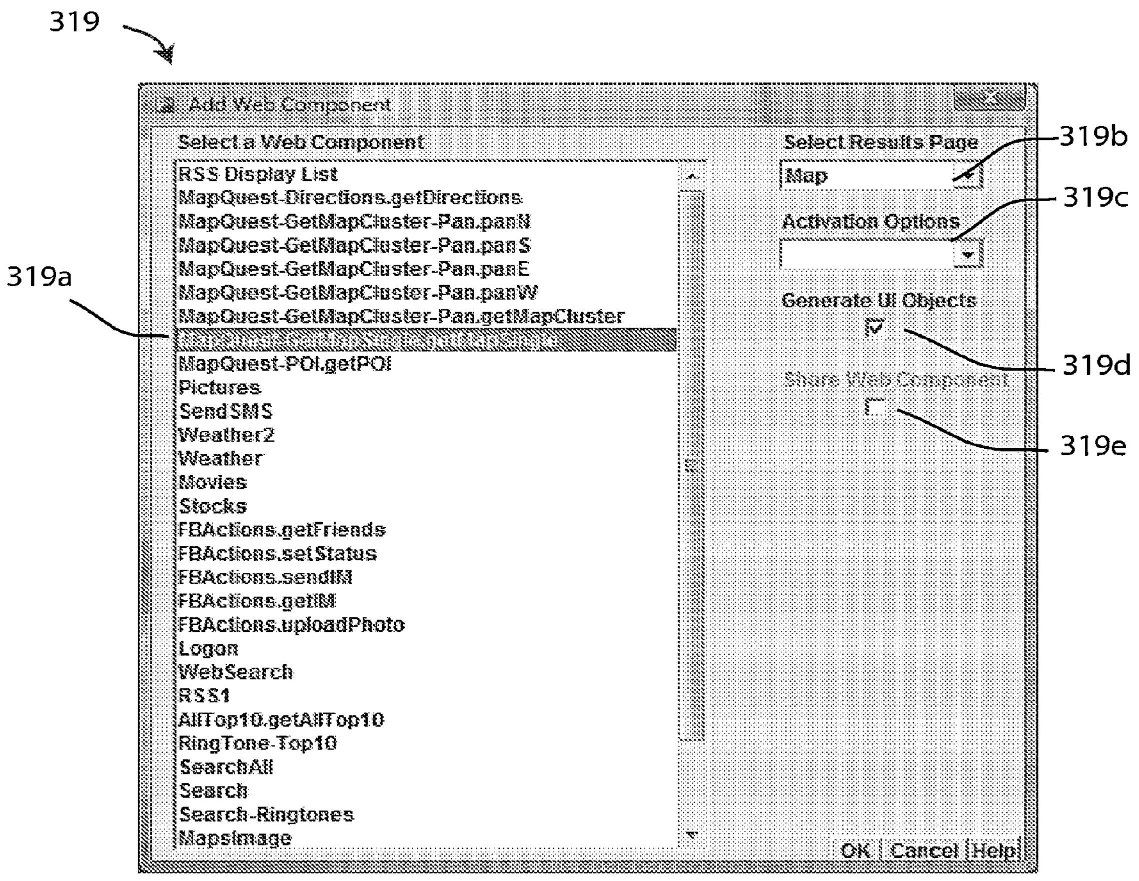
Link Set

Database Name	Table Name	Field Name

Channel Name	Channel Feed	Operation
Reuters	1	Replace

Object Selected: Page

FIG. 3E



Select a Web Component

- RSS Display List
- MapQuest-Directions.getDirections
- MapQuest-GetMapCluster-Pan.panN
- MapQuest-GetMapCluster-Pan.panS
- MapQuest-GetMapCluster-Pan.panE
- MapQuest-GetMapCluster-Pan.panW
- MapQuest-GetMapCluster-Pan.getMapCluster
- [illegible]
- MapQuest-POI.getPOI
- Pictures
- SendSMS
- Weather2
- Weather
- Movies
- Stocks
- FBActions.getFriends
- FBActions.setStatus
- FBActions.sendIM
- FBActions.getIM
- FBActions.uploadPhoto
- Logon
- WebSearch
- RSS1
- AllTop10.getAllTop10
- RingTone-Top10
- SearchAll
- Search
- Search-Ringtones
- MapsImage

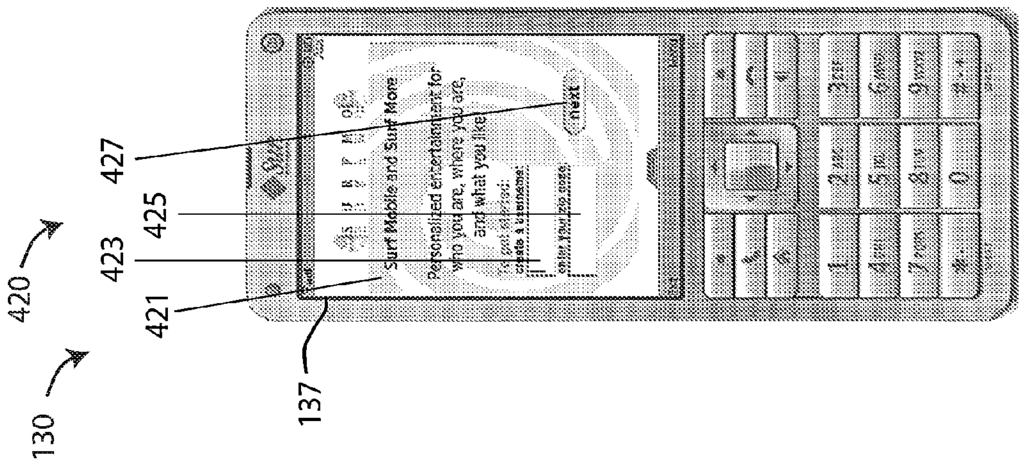
Select Results Page: Map

Activation Options:

Generate UI Objects: ✓

Share Web Component: ☐

FIG. 3F



Surf Mobile and Surf More

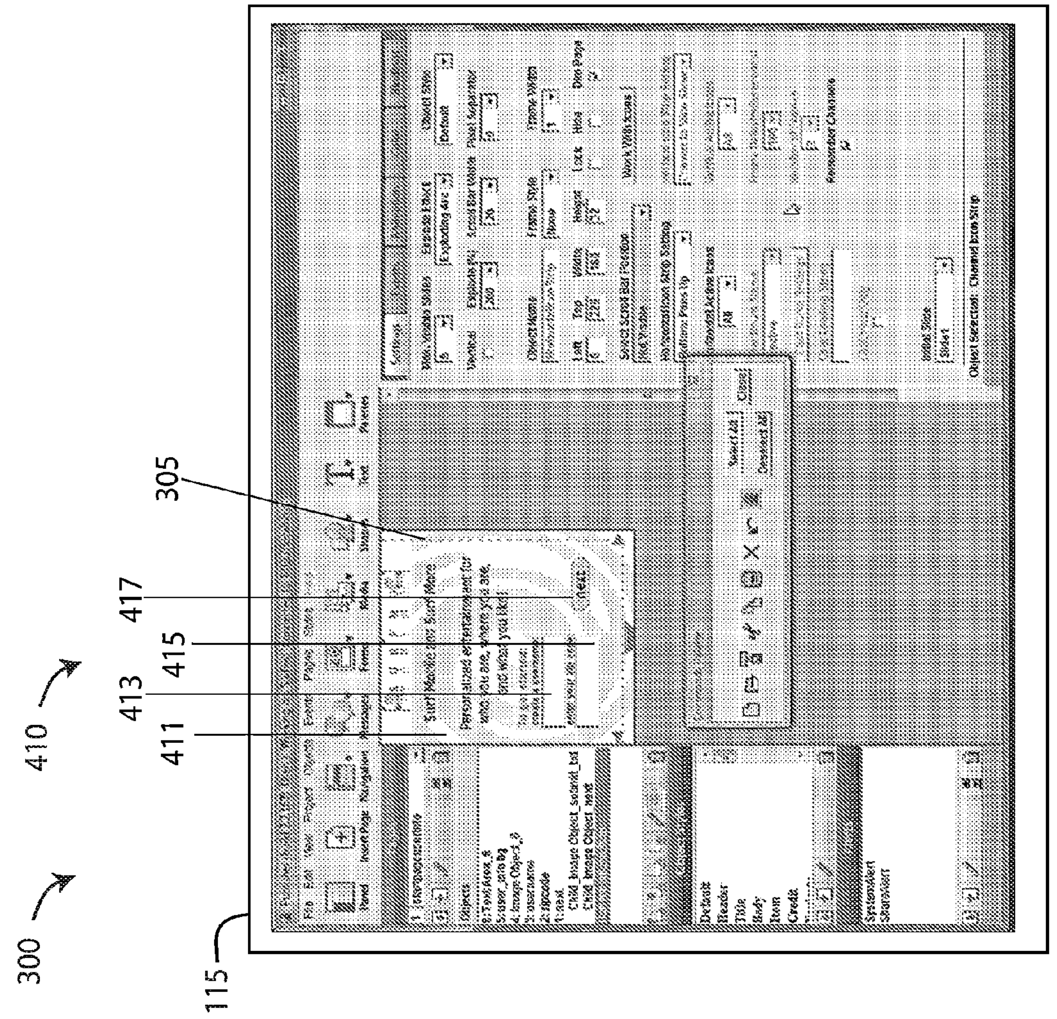
Personalized entertainment for who you are, where you are, and what you like!

To get started: create a username:

enter your nickname

next

FIG. 4B



Surf Mobile and Surf More

Personalized entertainment for who you are, where you are, and what you like!

To get started: create a username:

next

FIG. 4A

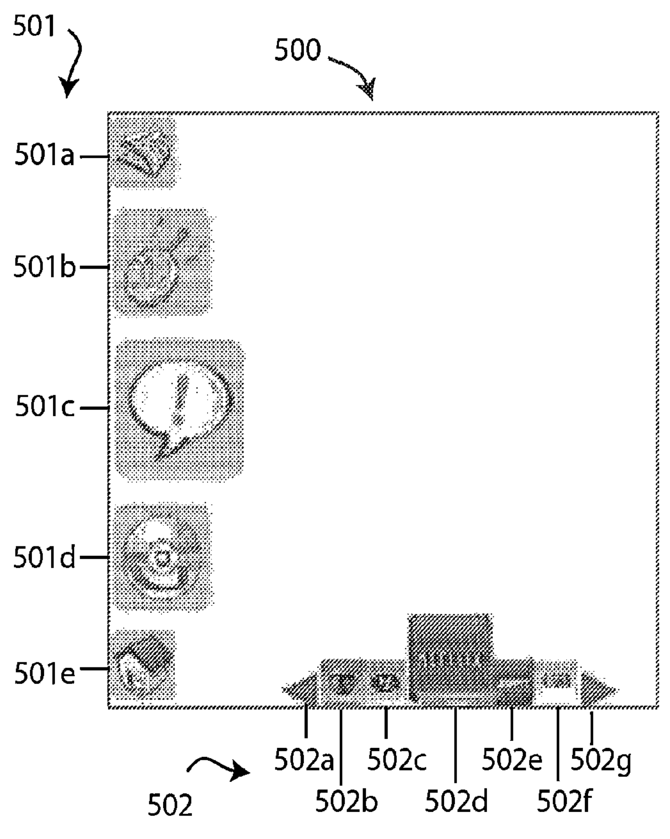
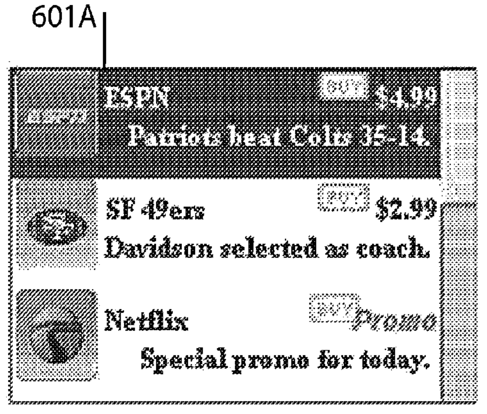
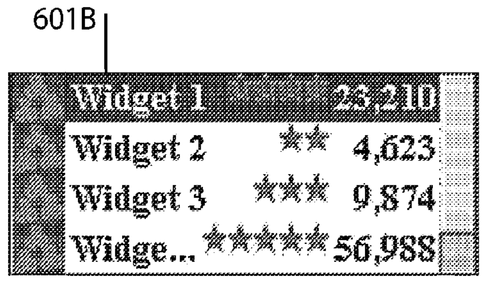


FIG. 5



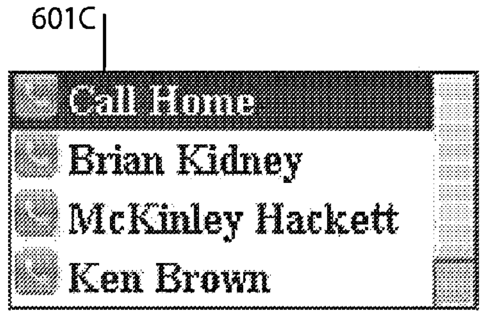
ESPN BUY $4.99
Patriots beat Colts 35-14.
SF 49ers BUY $2.99
Davidson selected as coach.
Netflix BUY Promo
Special promo for today.

FIG. 6A



Widget 1		23,210
Widget 2	★★	4,623
Widget 3	★★★	9,874
Widge...	★★★★★	56,988

FIG. 6B



Call Home
Brian Kidney
McKinley Hackett
Ken Brown

FIG. 6C

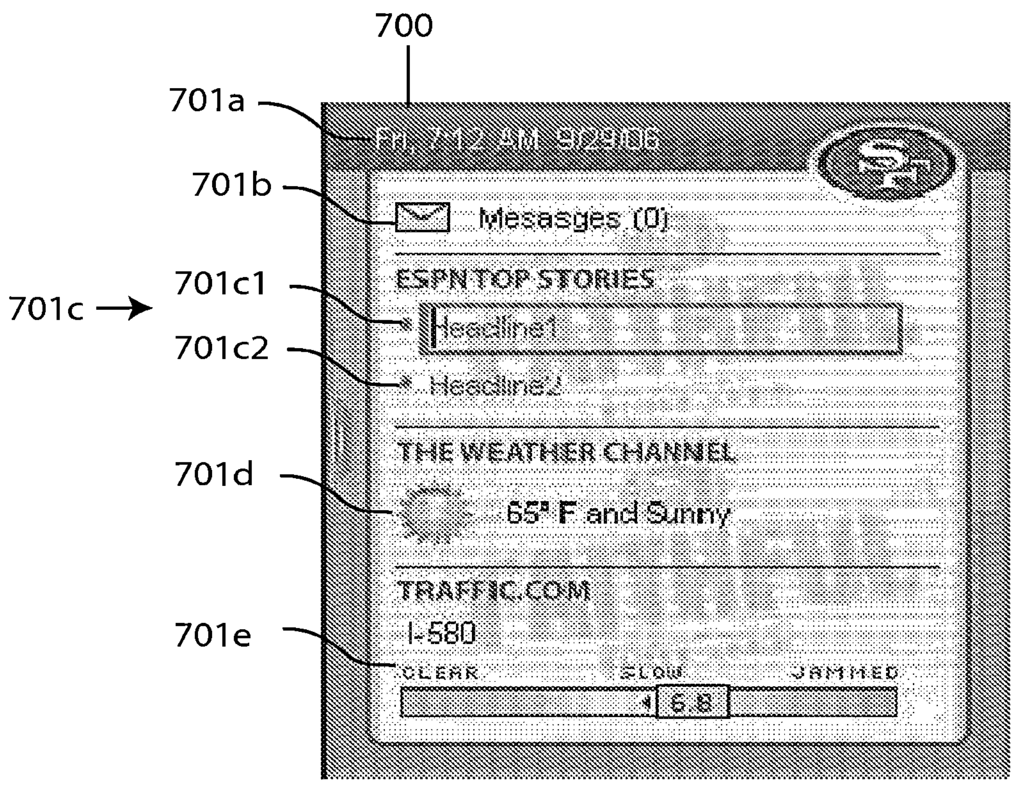

700
701a
701b
701c
701c1
701c2
701d
701e
Fri, 7:12 AM 9/29/06
Mesasges (0)
ESPN TOP STORIES
Headline1
Headline2
THE WEATHER CHANNEL
65° F and Sunny
TRAFFIC.COM
I-580
CLEAR
SLOW
JAMMED
6.8

FIG. 7

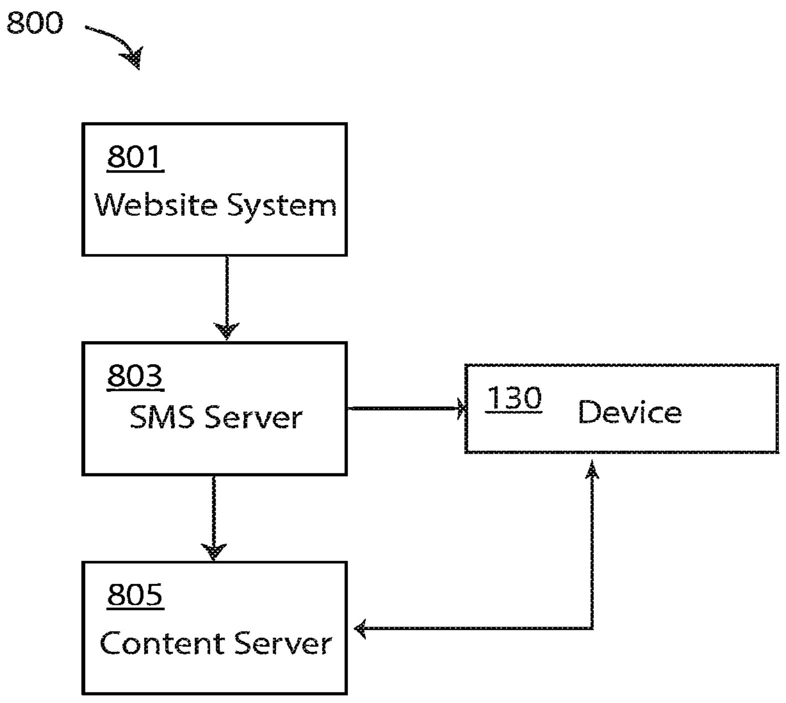

800
801
Website System
803
SMS Server
130
Device
805
Content Server

FIG. 8

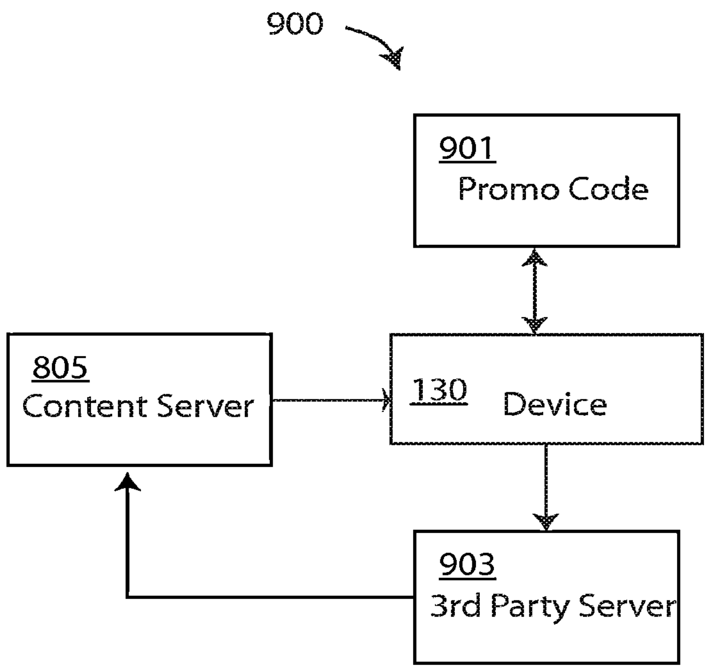

900
901
Promo Code
805
Content Server
130
Device
903
3rd Party Server

FIG. 9

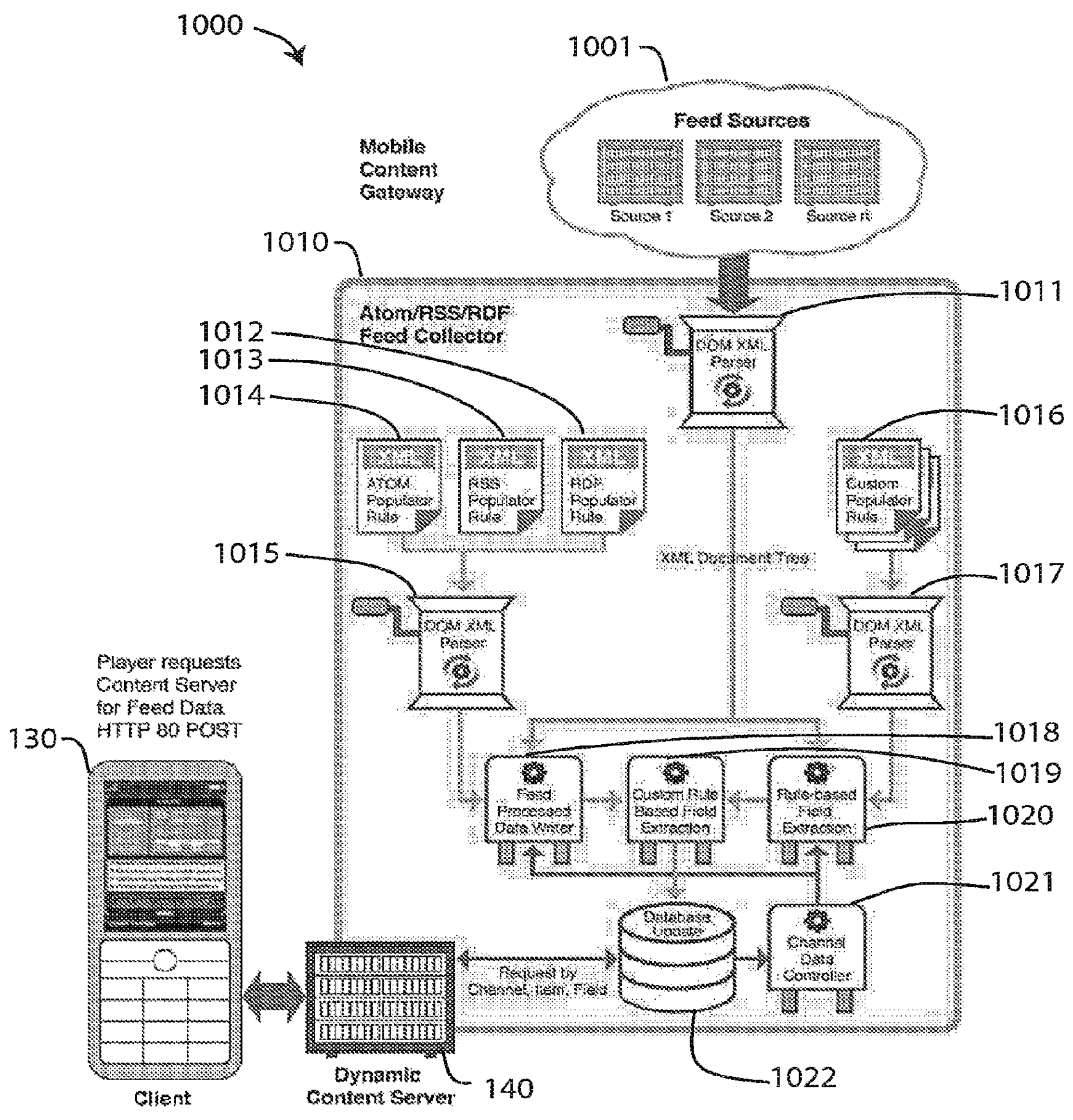

1000
1001
Feed Sources
Source 1
Source 2
Source n
Mobile Content Gateway
1010
Atom/RSS/RDF Feed Collector
1011
DOM XML Parser
1012
1013
1014
ATOM Populator Rule
RSS Populator Rule
RDF Populator Rule
1016
Custom Populator Rule
XML Document Tree
1015
DOM XML Parser
1017
DOM XML Parser
Player requests Content Server for Feed Data HTTP 80 POST
130
1018
1019
Feed Processed Data Writer
Custom Rule Based Field Extraction
Rule-based Field Extraction
1020
1021
Database Update
Channel Data Controller
Request by Channel, Item, Field
Client
Dynamic Content Server
140
1022

FIG. 10

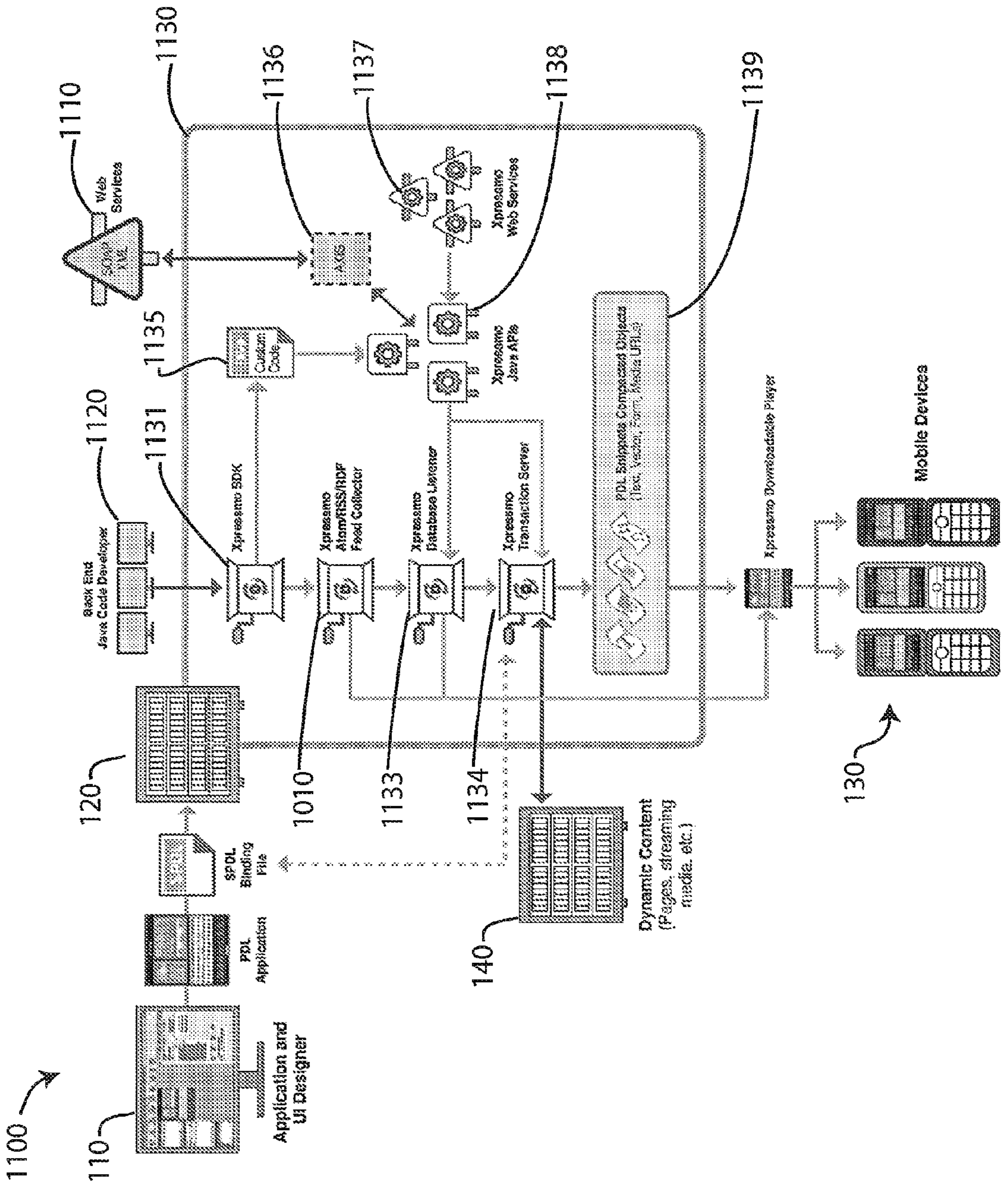

1100
110
Application and UI Designer
PDL Application
SPDL Binding File
120
Back End Java Code Developer
1120
1131
Xpressmo SDK
1135
Custom Code
1110
Web Services
SOAP XML
1130
1136
XAS
1137
Xpressmo Web Services
1138
Xpressmo Java APIs
1010
Xpressmo Atom/RSS/RDF Feed Collector
1133
Xpressmo Database Listener
1134
Xpressmo Transaction Server
140
Dynamic Content (Pages, streaming media, etc.)
1139
PDL Snippets Compacted Objects (Text, Vector, Form, Media URLs)
Xpressmo Downloadable Player
130
Mobile Devices

FIG. 11

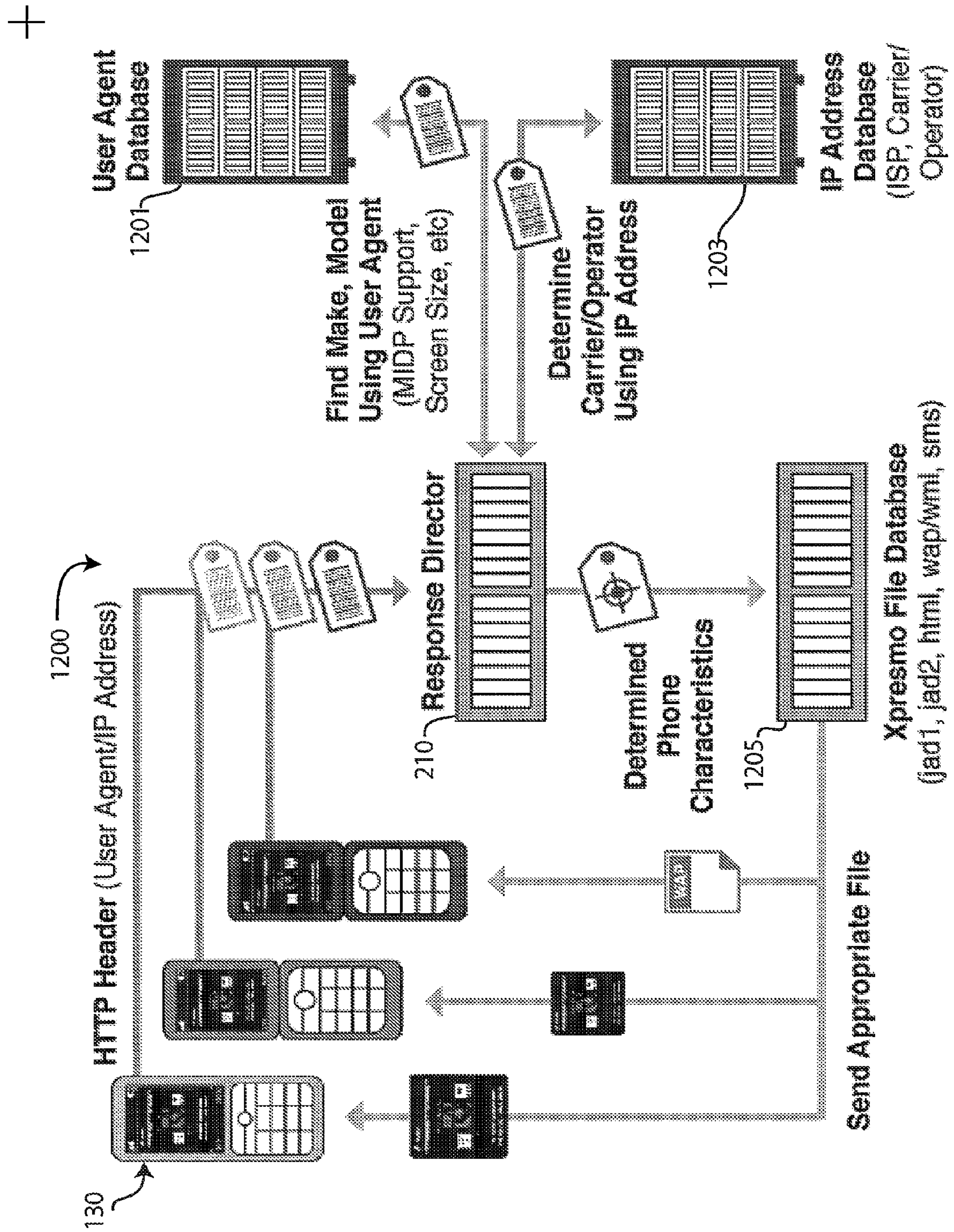

User Agent Database
1201
Find Make, Model Using User Agent (MIDP Support, Screen Size, etc)
Determine Carrier/Operator Using IP Address
IP Address Database (ISP, Carrier/ Operator)
1203
1200
HTTP Header (User Agent/IP Address)
Response Director
210
Determined Phone Characteristics
Xpresmo File Database (jad1, jad2, html, wap/wml, sms)
1205
Send Appropriate File
130

FIG. 12

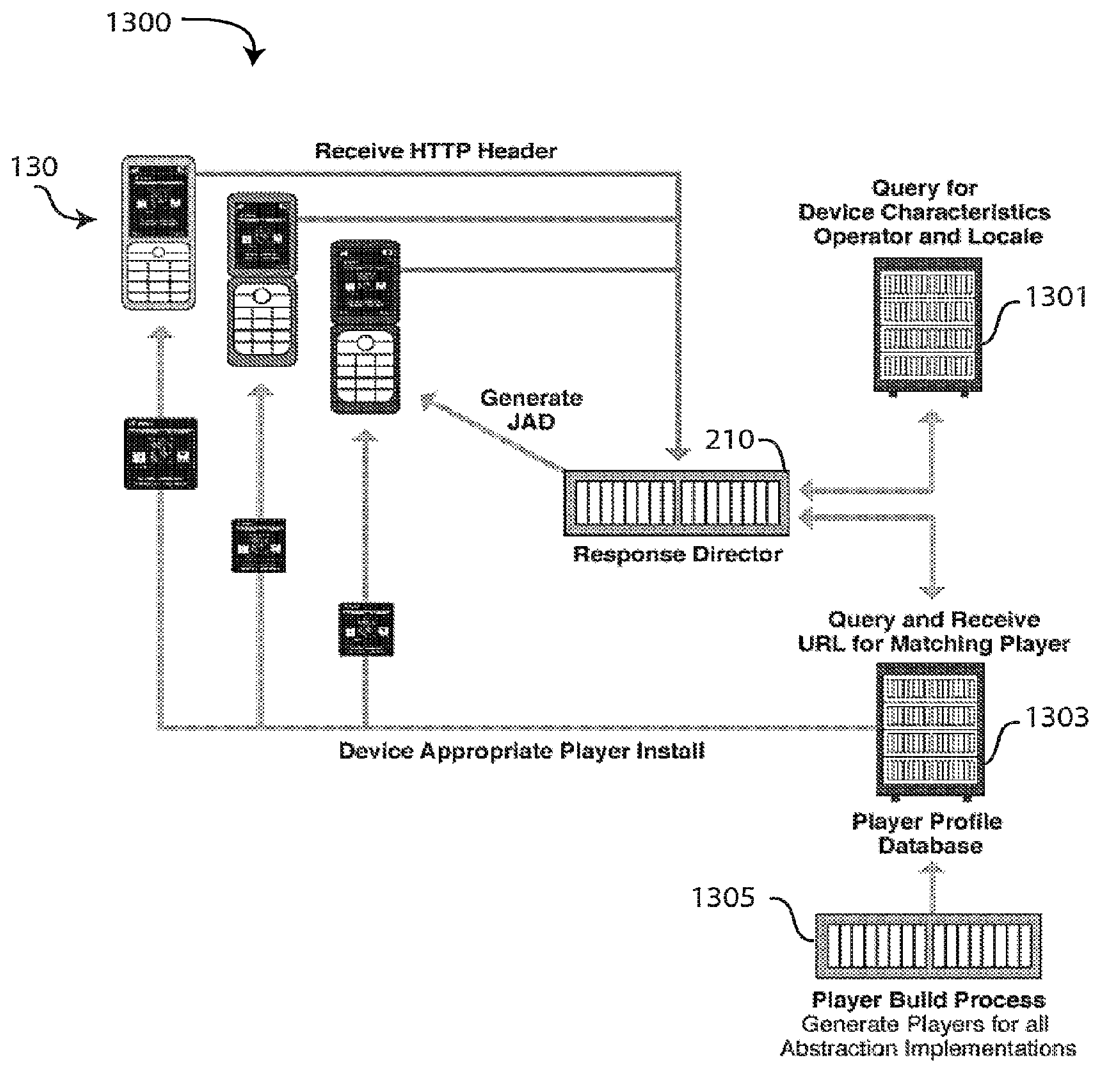

1300
130
Receive HTTP Header
Query for
Device Characteristics
Operator and Locale
1301
Generate
JAD
210
Response Director
Query and Receive
URL for Matching Player
1303
Device Appropriate Player Install
Player Profile
Database
1305
Player Build Process
Generate Players for all
Abstraction Implementations

FIG. 13

SYSTEMS AND METHODS FOR INTEGRATING WIDGETS ON MOBILE DEVICES

TECHNICAL FIELD

The present invention generally relates to providing software for mobile devices, and more particularly to a method and system for authoring Applications for devices.

BACKGROUND ART

Internet-connected mobile devices are becoming ever more popular. While these devices provide portability to the Internet, they generally do not have the capabilities of non-mobile devices including computing, input and output capabilities.

In addition, the mobility of the user while using such devices provides challenges and opportunities for the use of the Internet. Further, unlike non-mobile devices, there are a large number of types of devices and they tend to have a shorter lifetime in the marketplace. The programming of the myriad of mobile devices is a time-consuming and expensive proposition, thus limiting the ability of service providers to update the capabilities of mobile devices.

Thus there is a need in the art for a method and apparatus that permits for the efficient programming of mobile devices. Such a method and apparatus should be easy to use and provide output for a variety of devices.

DISCLOSURE OF INVENTION

In certain embodiments, a system is provided to generate code to provide content on a display of a platform. The system includes a database of web services obtainable over a network and an authoring tool. The authoring tool is configured to define an object for presentation on the display, select a component of a web service included in said database, associate said object with said selected component, and produce code that, when executed on the platform, provides said selected component on the display of the platform.

In certain other embodiments, a method is provided for providing information to platforms on a network. The method includes accepting a first code over the network, where said first code is platform-dependent; providing a second code over the network, where said second code is platform-independent; and executing said first code and said second code on the platform to provide web components obtained over the network.

In certain embodiments, a method for displaying content on a platform utilizing a database of web services obtainable over a network is provided. The method includes: defining an object for presentation on the display; selecting a component of a web service included in said database; associating said object with said selected component; and producing code that, when executed on the platform, provides said selected component on the display of the platform.

In one embodiment, one of the codes is a Player, which is a thin client architecture that operates in a language that manages resources efficiently, is extensible, supports a robust application model, and has no device specific dependencies. In another embodiment, Player P is light weight and extends the operating system and/or virtual machine of the device to: Manage all applications and application upgrades, and resolve device, operating system, VM and language fragmentation.

In another embodiment, one of the codes is an Application that is a device independent code that interpreted by the Player.

These features together with the various ancillary provisions and features which will become apparent to those skilled in the art from the following detailed description, are attained by the system and method of the present invention, preferred embodiments thereof being shown with reference to the accompanying drawings, by way of example only, wherein:

BRIEF DESCRIPTION OF DRAWINGS

FIG. **1**A is an illustrative schematic of one embodiment of a system including an authoring platform and a server for providing programming instructions to a device over a network;

FIG. **1**B is schematic of an alternative embodiment system for providing programming instructions to device over a network;

FIG. **2**A is a schematic of an embodiment of system illustrating the communications between different system components;

FIG. **2**B is a schematic of one embodiment of a device illustrating an embodiment of the programming generated by authoring platform;

FIGS. **3**A and **3**B illustrate one embodiment of a publisher interface as it appears, for example and without limitation, on a screen while executing an authoring tool;

FIG. **3**C illustrates an embodiment of the Events Tab'

FIG. **3**D illustrates one embodiment of an Animation Tab;

FIG. **3**E illustrates one embodiment of Bindings Tab;

FIG. **3**F illustrates one embodiment of a pop-up menu for adding web components;

FIG. **4**A shows a publisher interface having a layout on a canvas; and FIG. **4**B shows a device having the resulting layout on a device screen;

FIG. **5** shows a display of launch strips;

FIG. **6**A is a display of a Channel Selection List;

FIG. **6**B is a display of a Widget Selection List;

FIG. **6**C is a display of a Phone List;

FIG. **7** shows a display of a mash-up;

FIG. **8** is a schematic of an embodiment of a push capable system;

FIG. **9** is a schematic of an alternative embodiment of a push capable system;

FIG. **10** is a schematic of one embodiment of a feed collector;

FIG. **11** is a schematic of an embodiment of a Mobile Content Gateway;

FIG. **12** is a schematic of one embodiment of a system that includes a response director, a user agent database, an IP address database, and a file database; and

FIG. **13** is a schematic of another embodiment of a system that includes a response director, a user agent database, an IP address database, and a file database.

Reference symbols are used in the Figures to indicate certain components, aspects or features shown therein, with reference symbols common to more than one Figure indicating like components, aspects or features shown therein.

MODE(S) FOR CARRYING OUT THE INVENTION

FIG. **1**A is an illustrative schematic of one embodiment of a system **100** including an authoring platform **110** and a server **120** for providing programming instructions to a

device **130** over a network N. In one embodiment, device **130** is a wireless device, and network N includes wireless communication to the device. Alternatively, system **100** may provide access over network N to other information, data, or content, such as obtainable as a web service over the Internet. In general, a user of authoring platform **110** may produce programming instructions or files that may be transmitted over network N to operate device **130**, including instructions or files that are sent to device **130** and/or server **120**. The result of the authoring process is also referred to herein, and without limitation, as publishing an Application.

Embodiments include one or more databases that store information related to one or more devices **130** and/or the content provided to the devices. It is understood that such databases may reside on any computer or computer system on network N, and that, in particular, the location is not limited to any particular server, for example.

Device **130** may be, for example and without limitation, a cellular telephone or a portable digital assistant, includes a network interface **131**, a memory **133**, a processor **135**, a screen **137**, and an input device **139**. Network interface **131** is used by device **130** to communication over a wireless network, such as a cellular telephone network, a WiFi network or a WiMax network, and then to other telephones through a public switched telephone network (PSTN) or to a satellite, or over the Internet. Memory **133** includes programming required to operate device **130** (such as an operating system or virtual machine instructions), and may include portions that store information or programming instructions obtained over network interface **131**, or that are input by the user (such as telephone numbers or images from a device camera (not shown). In one embodiment screen **137** is a touch screen, providing the functions of the screen and input device **139**.

Authoring platform **110** includes a computer or computer system having a memory **111**, a processor **113**, a screen **115**, and an input device **117**. It is to be understood that memory **111**, processor **113**, screen **115**, and input device **117** are configured such a program stored in the memory may be executed by the processor to accept input from the input device and display information on the screen. Further, the program stored in memory **111** may also instruct authoring platform **110** to provide programming or information, as indicated by the line labeled "A" and to receive information, as indicated by the line labeled "B."

Memory **111** is shown schematically as including a stored program referred to herein, and without limitation, as an authoring tool **112**. In one embodiment, authoring tool **112** is a graphical system for designing the layout of features as a display that is to appear on screen **137**. One example of authoring tool **112** is the CDER™ publishing platform (Express Mobile, Inc., Novato, Calif.).

In another embodiment, which is not meant to limit the scope of the present invention, device **130** may include an operating system having a platform that can interpret certain routines. Memory **111** may optionally include programming referred to herein, and without limitation, as routines **114** that are executable on device **130**.

Routines **114** may include device-specific routines—that is, codes that are specific to the operating system, programming language, or platform of specific devices **130**, and may include, but are not limited to, Java, Windows Mobile, Brew, Symbian OS, or Open Handset Alliance (OHA). Several examples and embodiments herein are described with reference to the use of Java. It is to be understood that the invention is not so limited, except as provided in the claims, and that one skilled in the art could provide Players for devices using routines provided on a platform. Thus as an example, routines **114** may include Java API's and an authoring tool System Development Kit (SDK) for specific devices **130**.

Server **120** is a computer or computer system that includes a network interface **121**, a memory **123**, and a processor **125**. Is to be understood that network interface **121**, memory **123**, and processor **125** are configured such that a program stored in the memory may be executed by the processor to: accept input and/or provide output to authoring platform **110**; accept input and/or provide output through network interface **121** over network N to network interface **131**; or store information from authoring platform **110** or from device **130** for transmission to another device or system at a later time.

In one embodiment, authoring platform **110** permits a user to design desired displays for screen **137** and actions of device **130**. In other words, authoring platform **110** is used to program the operation of device **130**. In another embodiment, authoring platform **110** allows a user to provide input for the design of one or more device displays and may further allow the user to save the designs as device specific Applications. The Applications may be stored in memory **123** and may then be sent, when requested by device **130** or when the device is otherwise accessible, over network N, through network interface **130** for storage in memory **133**.

In an alternative embodiment, analytics information from devices **130** may be returned from device **130**, through network N and server **120**, back to authoring platform **110**, as indicated by line B, for later analysis. Analytics information includes, but is not limited to, user demographics, time of day, and location. The type of analytic content is only limited by which listeners have been activated for which objects and for which pages. Analytic content may include, but is not limited to, player-side page view, player-side forms-based content, player-side user interactions, and player-side object status.

Content server **140** is a computer or computer system that includes a network interface **141**, a memory **143**, and a processor **145**. It is to be understood that network interface **141**, memory **143**, and processor **145** are configured such that a stored program in the memory may be executed by the processor to accepts requests R from device **130** and provide content C over a network, such as web server content the Internet, to device **130**.

FIG. **1**B is schematic of an alternative embodiment system **100** for providing programming instructions to device **130** over a network N that is generally similar to the system of FIG. **1**A. The embodiment of FIG. **1**B illustrates that system **100** may include multiple servers **120** and/or multiple devices **130**.

In the embodiment of FIG. **1**B, system **100** is shown as including two or more servers **120**, shown illustratively and without limitation as servers **120***a* and **120***b*. Thus some of the programming or information between authoring platform **110** and one or more devices **130** may be stored, routed, updated, or controlled by more than one server **120**. In particular, the systems and methods described herein may be executed on one or more server **120**.

Also shown in FIG. **1**B are a plurality of devices **130**, shown illustratively and without limitation as device **130-1**, **130-1**, . . . **130**-N. System **100** may thus direct communication between individual server(s) **120** and specific device(s) **130**.

As described subsequently, individual devices **130** may be provided with program instructions which may be stored in each device's memory **133** and where the instructions are executed by each device's processor **135**. Thus, for example, server(s) **120** may provide device(s) **130** with programming in response to the input of the uses of the individual devices. Further, different devices **130** may be operable using different sets of instructions, that is having one of a variety of different "device platforms." Differing device platforms may result, for example and without limitation, to different operating systems, different versions of an operating system, or different versions of virtual machines on the same operating system. In some embodiments, devices **130** are provided with some programming from authoring system **100** that is particular to the device.

In one embodiment, system **100** provides permits a user of authoring platform **110** to provide instructions to each of the plurality of devices **130** in the form of a device- or device-platform specific instructions for processor **135** of the device, referred to herein and without limitation as a "Player," and a device-independent program, referred to herein and without limitation as an "Application" Thus, for example, authoring platform **110** may be used to generate programming for a plurality of devices **130** having one of several different device platforms. The programming is parsed into instructions used by different device platforms and instructions that are independent of device platform. Thus in one embodiment, device **130** utilizes a Player and an Application to execute programming from authoring platform **110**. A device having the correct Player is then able to interpret and be programmed according to the Application.

In one alternative embodiment, the Player is executed the first time by device **130** ("activated") through an Application directory. In another alternative embodiment, the Player is activated by a web browser or other software on device **130**. In yet another alternative embodiment, Player is activated through a signal to device **130** by a special telephone numbers, such as a short code.

When the Application and the Player are provided to memory **133**, the functioning of device **130** may occur in accordance with the desired programming. Thus in one embodiment, the Application and Player includes programming instructions which may be stored in memory **133** and which, when executed by processor **135**, generate the designed displays on screen **137**. The Application and Player may also include programming instructions which may be stored in memory **133** and which provide instructions to processor **135** to accept input from input device **139**.

Authoring tool **112** may, for example, produce and store within memory **111** a plurality of Players (for different devices **130**) and a plurality of Applications for displaying pages on all devices. The Players and Applications are then stored on one or more servers **120** and then provided to individual devices **130**. In general, Applications are provided to device **130** for each page of display or a some number of pages. A Player need be provided once or updated as necessary, and thus may be used to display a large number of Applications. This is advantageous for the authoring process, since all of the device-dependent programming is provided to a device only once (or possibly for some small number of upgrades), permitting a smaller Application, which is the same for each device **130**.

Thus, for example and without limitation, in one embodiment, the Player transforms device-independent instructions of the Application into device-specific instructions that are executable by device **130**. Thus, by way of example and without limitation, the Application may include Java programming for generating a display on screen **137**, and the Player may interpret the Java and instruct processor **135** to produce the display according to the Application for execution on a specific device **130** according to the device platform. The Application may in general include, without limitation, instructions for generating a display on screen **137**, instructions for accepting input from input device **139**, instructions for interacting with a user of device **130**, and/or instructions for otherwise operating the device, such as to place a telephone call.

The Application is preferably code in a device-independent format, referred to herein and without limitation as a Portable Description Language (PDL). The device's Player interprets or executes the Application to generate one or more "pages" ("Applications Pages") on a display as defined by the PDL. The Player may include code that is device-specific—that it, each device is provided with a Player that is used in the interpretation and execution of Applications. Authoring tool **112** may thus be used to design one or more device-independent Applications and may also include information on one or more different devices **130** that can be used to generate a Player that specific devices may use to generate displays from the Application.

In one embodiment, system **100** provides Players and Applications to one server **120**, as in FIG. **1**A. In another embodiment, system **100** provides Players to a first server **120***a* and Applications to a second server **120***b*, as in FIG. **1**B.

In one embodiment, authoring tool **112** may be used to program a plurality of different devices **130**, and routines **114** may include device-specific routines. In another embodiment, the Player is of the type that is commonly referred to as a "thin client"—that is, software for running on the device as a client in client-server architecture with a device network which depends primarily on a central server for processing activities, and mainly focuses on conveying input and output between the user and the server.

In one embodiment, authoring platform **110** allows user to arrange objects for display on screen. A graphical user interface ("GUI," or "UI") is particularly well suited to arranging objects, but is not necessary. The objects may correspond to one or more of an input object, an output object, an action object, or may be a decorative display, such as a logo, or background color or pattern, such as a solid or gradient fill. In another embodiment, authoring platform **110** also permits a user to assign actions to one or more of an input object, an output object, or an action object. In yet another embodiment, authoring platform **110** also permits a user to bind one or more of an input object, an output object, or an action object with web services or web components, or permits a user to provide instructions to processor **135** to store or modify information in memory **133**, to navigate to another display or service, or to perform other actions, such as dialing a telephone number.

In certain embodiments, the applicant model used in developing and providing Applications is a PDL. The PDL can be conceptually viewed as a device, operating system and virtual machine agnostic representation of Java serialized objects. In certain embodiments, the PDL is the common language for authoring tool **112**, the Application, and Player. Thus while either designing the Application with the authoring tool **112**, or programming with the SDK, the internal representation of the programming logic is in Java. In one embodiment the SDK is used within a multi-language software development platform comprising an IDE and a plug-in system to extend it, such as the Eclipse Integrated Development Environment (see, for example, http://www.eclipse.org/). At publish time the Java code is translated into a PDL. This translation may also occur in real-time during the execution of any Web Services or backend business logic that interacts with the user.

One embodiment for compacting data that may be used is described in co-pending U.S. Pat. No. 6,546,397 to Rempell ("Rempell"), the contents of which are incorporated herein by reference. In that patent the compressed data is described as being a database. The terminology used here is a PDL, that is the "internal database" of Rempell is equivalent to the PDL of the present Application.

The use of a PDL, as described in Rempell, permits for efficient code and data compaction. Code, as well as vector, integer and Boolean data may be compacted and then compressed resulting in a size reduction of 40 to 80 times that of the original Java serialized objects. This is important not only for performance over the network but for utilizing the virtual memory manager of the Player more efficiently. As an example, the reassembled primitives of the Java objects may first undergo logical compression, followed by LZ encoding.

The use of a PDL also provides virtual machine and operating system independence. Since the reassembled primitives of the Application no longer have any dependencies from the original programming language (Java) that they were defined in. The PDL architecture takes full advantage of this by abstracting all the virtual machine and/or operating system interfaces from the code that processes the PDL.

In one embodiment, the PDL is defined by the means of nested arrays of primitives. Accordingly, the use of a PDL provides extensibility and compatibility, with a minimal amount of constraints in extending the Player seamlessly as market demands and device capabilities continue to grow. Compatibility with other languages is inherent based on the various Player abstraction implementations, which may be, for example and without limitation, Java CDC, J2SE or MIDP2 implementations.

In one embodiment, the architecture of Player P includes an abstraction interface that separates all device, operating system and virtual machine dependencies from the Player's Application model business logic (that is, the logic of the server-side facilities) that extend the Application on the Player so that it is efficiently integrated into a comprehensive client/server Application. The use of an abstraction interface permits the more efficient porting to other operating systems and virtual machines and adding of extensions to the Application model so that a PDL can be implemented once and then seamlessly propagated across all platform implementations. The Application model includes all the currently supported UI objects and their attributes and well as all of the various events that are supported in the default Player. Further, less robust platforms can be augmented by extending higher end capabilities inside that platform's abstraction interface implementation.

In one embodiment, authoring platform **110** provides one or more pages, which may be provided in one Application, or a plurality of Applications, which are stored in memory **123** and subsequently provided to memory **133**. In certain embodiments, the Application includes instructions R to request content or web services C from content server **140**. Thus, for example and without limitation, the request is for information over the network via a web service, and the request R is responded to with the appropriate information for display on device **130**. Thus, for example, a user may request a news report. The Application may include the layout of the display, including a space for the news, which is downloaded form content server **140** for inclusion on the display. Other information that may be provided by content server **140** may include, but is not limited to, pages, Applications, multimedia, and audio.

FIG. **2**A is a schematic of a system **200** of an embodiment of system **100** illustrating the communications between different system components. System includes a response director **210**, a web component registry **220**, and a web service **230**. System **200** further includes authoring platform **110**, server **120**, device **130** and content server **140** are which are generally similar to those of the embodiments of FIGS. **1**A and **1**B, except as explicitly noted.

Response director **210** is a computer or computer system that may be generally similar to server **120** including the ability to communicate with authoring platform **110** and one or more devices **130**. In particular, authoring platform **110** generates one or more Players (each usable by certain devices **130**) which are provided to response director **210**. Devices **130** may be operated to provide response director **210** with a request for a Player and to receive and install the Player. In one embodiment, device **130** provides response director **210** with device-specific information including but not limited to make, model, and/or software version of the device. Response director **210** then determines the appropriate Player for the device, and provides the device with the Player over the network.

Web service **230** is a plurality of services obtainable over the Internet. Each web service is identified and/or defined as an entry in web component registry **230**, which is a database, XML file, or PDL that exists on a computer that may be a server previously described or another server **120**. Web component registry **230** is provided through server **120** to authoring platform **110** so that a user of the authoring platform may bind web services **230** to elements to be displayed on device **130**, as described subsequently.

In one embodiment, authoring platform **110** is used in conjunction with a display that provides a WYSIWYG environment in which a user of the authoring platform can produce an Application and Player that produces the same display and the desired programming on device **130**. Thus, for example, authoring tool **112** provides a display on screen **115** that corresponds to the finished page that will be displayed on screen **137** when an Application is intercepted, via a Player, on processor **135** of device **130**.

Authoring platform **110** further permits a user of the authoring platform to associate objects, such as objects for presenting on screen **137**, with components of one or more web services **230** that are registered in web component registry **220**. In one embodiment, information is provided in an XML file to web component registry **220** for each registered components of each web service **230**. Web component registry **220** may contain consumer inputs related to each web service **230**, environmental data such as PIM, time or location values, persistent variable data, outputs related to the web service, and/or optional hinting for improving the user's productivity.

A user of authoring platform **110** of system **200** may define associations with web services as WebComponent Bindings. In one embodiment, authoring platform **110** allows a user to associate certain objects for display that provide input or output to components of web service **230**. The associated bindings are saved as a PDL in server **120**.

In one embodiment, an XML web component registry **220** for each registered web service **230** is loaded into authoring platform **110**. The user of system **200** can then assign components of any web service **230** to an Application without any need to write code. In one embodiment, a component of web service **230** is selected from authoring platform **110** which presents the user with WYSIWYG dialog boxes that enable the binding of all the inputs and outputs of component of web service **230** to a GUI component of the Application as will be displayed on screen **137**. In addition, multiple components of one or more web service **230** can be assigned to any Object or Event in order to facilitate mashups. These Object and/or Event bindings, for each instance of a component of any web service **230**, are stored in the PDL. The content server **140** handles all communication between device **130** and the web service **230** and can be automatically deployed as a web application archive to any content server.

Device **130**, upon detecting an event in which a component of a web service **230** has been defined, assembles and sends all related inputs to content server **240**, which proxies the request to web service **230** and returns the requested information to device **130**. The Player on device **130** then takes the outputs of web service **230** and binds the data to the UI components in the Application, as displayed on screen **137**.

In one embodiment, the mechanism for binding the outputs of the web service to the UI components is through symbolic references that matches each output to the symbolic name of the UI component. The outputs, in one embodiment, may include meta-data which could become part of the inputs for subsequent interactions with the web service.

For example, if a user of authoring platform **110** wants to present an ATOM feed on device **130**, they would search through a list of UI Components available in the authoring platform, select the feed they want to use, and bind the output of the feed summary to a textbox. The bindings would be saved into the PDL on server **120** and processed by device **130** at runtime. If the ATOM feed does not exist a new one can be added to the web component registry that contains all the configuration data required, such as the actual feed URL, the web component manager URL, and what output fields are available for binding.

In another embodiment, components of web services **230** are available either to the user of authoring platform **110** or otherwise accessible through the SDK and Java APIs of routines **114**. System **200** permits an expanding set of components of web services **230** including, but not limited to: server pages from content server **120**; third-party web services including, but not limited to: searching (such through Google or Yahoo), maps (such as through MapQuest and Yahoo), storefronts (such as through ThumbPlay), SMS share (such as through clickatel), stock quotes, social networking (such as through FaceBook), stock quotes, weather (such as through Accuweather) and/or movie trailers. Other components include web services for communication and sharing through chats and forums and rich messaging alerts, where message alerts are set-up that in turn could have components of Web Services **230** defined within them, including the capture of consumer generated and Web Service supplied rich media and textual content.

System **200** also permits dynamic binding of real-time content, where the inputs and outputs of XML web services are bound to GUI components provided on screen **137**. Thus, for example, a user of authoring platform **110** may bind attributes of UI Objects to a particular data base field on a Server. When running the Application, the current value in the referenced data base will be immediately applied. During the Application session, any other real time changes to these values in the referenced data base will again be immediately displayed.

As an example of dynamic binding of real-time content, an RSS feeds and other forms of dynamic content may be inserted into mobile Applications, such as device **130**, using system **200**. Authoring platform **110** may include a "RSS display" list which permits a user to select RSS channels and feeds from an extensible list of available dynamic content. Meta data, such as titles, abstracts and Images can be revealed immediately by the user as they traverse this RSS display list, bringing the PC experience completely and conveniently to mobile devices **130**. In addition, Authoring platform **110** may include a dialog box that dynamically links objects to data and feeds determined by RSS and chat databases. Any relevant attribute for a page view and/or object can be dynamically bound to a value in a server-side database. This includes elements within complex objects such as: any icon or text element within a graphical list; any icon within a launch strip; any feature within any geographical view of a GIS service object; and/or any virtual room within a virtual tour.

As an example of third-party web services **230** that may be provided using system **200**, a user of authoring platform **110** can place, for example, Yahoo maps into device **130** by binding the required component of the Yahoo Maps Web Service, such as Yahoo Map's Inputs and/or Outputs to appropriate Objects of authoring platform **110**. System **200** also provides binding to web services for text, image and video searching by binding to components of those web services.

In one embodiment, an Application for displaying on device **130** includes one or more Applications Pages, each referred to herein as an "XSP," that provides functionality that extends beyond traditional web browsers. The XSP is defined as a PDL, in a similar manner as any Application, although it defines a single page view, and is downloaded to the Player dynamically as required by the PDL definition of the Application. Thus, for example, while JSPs and ASPs, are restricted to the functionality supported by the web browser, the functionality of XSPs can be extended through authoring platform **110** having access to platform dependent routines **114**, such as Java APIs. Combined with dynamic binding functionality, an XSP, a page can be saved as a page object in an author's "pages" library, and then can be dynamically populated with real-time content simultaneously as the page is downloaded to a given handset Player based on a newly expanded API. XSP Server Pages can also be produced programmatically, but in most cases authoring platform **110** will be a much more efficient way to generate and maintain libraries of dynamically changing XSPs.

With XSPs, Applications Pages that have dynamic content associated with them can be sent directly to device **130**, much like how a web browser downloads an HTML page through a external reference. Without XSPs, content authors would have to define each page in the Application. With XSPs, no pages need to be defined. Thus, for example, in a World Cup Application, one page could represent real-time scores that change continuously on demand. With polling (for example, a prompt to the users asking who they predict will win a game), a back-end database would tabulate the information and then send the results dynamically to the handsets. With a bar chart, the Application would use dynamic PDL with scaling on the fly. For example, the server would recalibrate the bar chart for every ten numbers.

Other combinations of components of web services **230** include, but are not limited to, simultaneous video chat sessions, inside an integrated page view, with a video or television station; multiple simultaneous chat sessions, each with a designated individual and/or group, with each of the chat threads visible inside an integrated page view.

Another extension of an XSP is a widget object. Widgets can be developed from numerous sources including, but not limited to, authoring platform **110**, a Consumer Publishing Tool, and an XML to Widget Conversion Tool where the SDK Widget Libraries are automatically populated and managed, or Widget Selection Lists that are available and can be populated with author defined Icons.

Applications, Players, and Processing in a Device

FIG. **2**B is a schematic of one embodiment of a device **130** illustrating an embodiment of the programming generated by authoring platform **110**. Memory **133** may include several different logical portions, such as a heap **133***a*, a record store **133***b* and a filesystem (not shown).

As shown in FIG. **2**B, heap **133***a* and record store **133***b* include programming and/or content. In general, heap **133***a* is readily accessible by processor **135** and includes, but is not limited to portions that include the following programming: a portion **133***a***1** for virtual machine compliant objects representing a single Page View for screen **137**; a portion **133***a***2** for a Player; a portion **133***a***3** for a virtual machine; and a portion **133***a***4** for an operating system.

Record store **133***b* (or alternatively the filesystem) includes, but is not limited to, portions **133***b***1** for Applications and non-streaming content, which may include portions **133***a***2** for images, portions **133***a***4** for audio, and/or portions **133***a***5** for video, and portions **133***b***2** for non-Application PDLs, such as a Master Page PDL for presenting repeating objects, and Alerts, which are overlayed on the current page view. Other content, such as streaming content may be provided from network interface **131** directly to the Media Codec of device **130** with instructions from Player on how to present the audio or video.

In one embodiment, the Player includes a Threading Model and a Virtual Memory Manager. The Threading Model first manages a queue of actions that can be populated based on Input/Output events, Server-side events, time-based events, or events initiated by user interactions. The Threading Model further manages the simultaneous execution of actions occurring at the same time. The Virtual Memory Manager includes a Logical Virtual Page controller that provides instructions from the record store to the heap, one page at time. Specifically, the Virtual Memory Manager controls the transfer of one of the Application Pages and its virtual machine compliant objects into portion **133***a***1** as instructions readable by the Player or Virtual Machine. When the Player determines that a new set of instructions is required, the information (such as one Application Page is retrieve from the Record store, converted into virtual machine compliant objects (by processor **135** and according to operation by the Player, Virtual Machine, etc), and stored in heap **133***a*. Alternatively, the Player may augment virtual machine compliant objects with its own libraries for managing user interactions, events, memory, etc.

The connection of portions **133***a***1**, **133***a***2**, **133***a***3**, **133***a***4**, record store **133***b* and processor **135** are illustrative of the logical connection between the different types of programming stored in Heap **133***a* and record store **133***b*, that is, how data is processed by processor **135**.

The Player determines which of the plurality of Application Pages in portion **133***b***1** is required next. This may be determined by input actions from the Input Device **139**, or from instructions from the current Application Page. The Player instructs processor **135** to extract the PDF from that Applications Page and store it in portion **133***a***1**. The Player then interprets the Application Page extracted from PDL

which in turn defines all of the virtual machine compliant Objects, some of which could have attributes that refer to images, audio, and/or video stored in portions **133***a***3**, **133***a***4**, **133***a***5**, respectively.

The Virtual Machine in portion **133***a***3** processes the Player output, the Operating System in portion **133***a***3** processes the Virtual Machine output which results in machine code that is processed by the Operating System in portion **133***a***4**.

In another embodiment, the Player is a native program that interacts directly with the operating system.

Embodiments of a Publishing Environment

In one embodiment, authoring platform **110** includes a full-featured authoring tool **112** that provides a what-you-see-is-what-you-get (WYSIWYG) full featured editor. Thus, for example, authoring tool **112** permits a user to design an Application by placing objects on canvas **305** and optionally assigning actions to the objects and save the Application. System **100** then provides the Application and Player to a device **130**. The Application as it runs on device **130** has the same look and operation as designed on authoring platform **110**. In certain embodiments, authoring platform **110** is, for example and without limitation, a PC-compatible or a Macintosh computer.

Authoring platform **110** produces an Application having one or more Applications Pages, which are similar to web pages. That is, each Applications Page, when executed on device **130** may, according to its contents, modify what is displayed on screen **137** or cause programming on the device to change in a manner similar to how web pages are displayed and navigated through on a website.

In one embodiment, authoring tool **112** allows a user to place one or more objects on canvas **305** and associate the objects with an Applications Pages. Authoring platform **110** maintains a database of object data in memory **111**, including but not limited to type of object, location on which page, and object attributes. The user may add settings, events, animations or binding to the object, from authoring tool **112**, which are also maintained in memory **111**. Authoring tool **112** also allows a user to define more than one Applications Page.

In another embodiment, authoring tool **112**, provides Java programming functions of the Java API for specific devices **130** as pull-down menus, dialog boxes, or buttons. This permits a user of authoring platform **110** to position objects that, after being provided as an Application to device **130**, activate such Java functions on the device.

In certain embodiments, authoring platform **110**, as part of system **100**, permits designers to include features of advanced web and web services Applications for access by users of device **130**. Some of the features of advanced web and web services include, but are not limited to: slide shows, images, video, audio, animated transitions, multiple chats, and mouse interaction; full 2-D vector graphics; GIS (advanced LBS), including multiple raster and vector layers, feature sensitive interactions, location awareness, streaming and embedded audio/video, virtual tours, image processing and enhancement, and widgets. In other embodiments the features are provided for selection in authoring platform **110** through interactive object libraries.

In certain embodiments, authoring platform **110**, as part of system **100**, allows the inclusion of child objects which may eventually be activated on device **130** by the user of the device or by time. The uses of the child objects on device **130** include, but are not limited to: mouse over (object selection), hover and fire events and launching of object-specific, rich-media experiences.

In certain other embodiments, authoring platform **110**, as part of system **100**, provides advanced interactive event models on device **130**, including but not limited to: user-, time- and/or location-initiated events, which allow content developers to base interactivity on specific user interactions and/or instances in time and space; timelines, which are critical for timing of multiple events and for animations when entering, on, or exiting pages of the Application; waypoints, which act similar to key frames, to allow smooth movement of objects within pages of the Application. Waypoints define positions on a page object's animation trajectory. When an object reaches a specific waypoint other object timelines can be initiated, creating location-sensitive multiple object interaction, and/or audio can be defined to play until the object reaches the next waypoint.

Authoring platform **110** may also define a Master Page, which acts as a template for an Applications Page, and may also define Alert Pages, which provide user alerts to a user of device **130**.

In certain embodiments, authoring platform **110**, as part of system **100**, provides full style inheritance on device **130**. Thus, for example and without limitation, both master page inheritance (for structural layout inheritance and repeating objects) and object styles (for both look and feel attribute inheritance) are supported. After a style has been defined for an object, the object will inherit the style. Style attributes include both the look and the feel of an object, including mouse interaction, animations, and timelines. Each page may include objects that may be a parent object or a child object. A child object is one that was created by first selecting a parent object, and then creating a child object. Child objects are always part of the same drawing layer as its parent object, but are drawn first, and are not directly selectable when running the Application. A parent object is any object that is not a child object, and can be selected when running the Application.

As an example, the user of authoring tool **112** may create various text objects on canvas **305** using a style that sets the font to red, the fonts of these objects will be red. Suppose user of authoring tool **112** changes the font color of a specific button to green. If later, the user of authoring tool **112** changes the style to blue; all other text objects that were created with that style will become blue except for the button that had been specifically set to green.

In certain other embodiments, authoring platform **110** provides page view, style, object, widget and Application template libraries. Authoring platform **110** may provide templates in private libraries (available to certain users of the authoring platform) and public libraries (available to all users of the authoring platform). Templates may be used to within authoring platform **110** to define the look and feel of the entire Application, specific pages, or specific slide shows and virtual tours a seen on device **130**.

FIGS. **3**A and **3**B illustrate one embodiment of a publisher interface **300** as it appears, for example and without limitation, on screen **115** while executing authoring tool **112**. In one embodiment, publisher interface **300** includes a Menu bar **301**, a Tool bar **303**, a Canvas **305**, a Layer Inspector **307** having subcomponents of a page/object panel **307***a*, an object style panel **307***b*, and a page alert panel **307***c*, and a Resource Inspector **309**.

In general, publisher interface **300** permits a user of authoring platform **110** to place objects on canvas **305** and then associate properties and/or actions to the object, which are stored in the Application. As described subsequently, publisher interface **300** permits a user to program a graphical interface for the screen **137** of device **130** on screen **115** of

authoring platform **110**, save an Application having the programming instructions, and save a Player for the device. The intended programming is carried out on device **130** when the device, having the appropriate device platform Player, receives and executes the device-independent Application.

Thus, for example, authoring tool **112** maintains, in memory **111**, a list of every type of object and any properties, actions, events, or bindings that may be assigned to that object. As objects are selected for an Application, authoring tool **112** further maintains, in memory **111**, a listing of the objects. As the user selects objects, publisher interface **300** provides the user with a choice of further defining properties, actions, events, or bindings that may be assigned to each particular object, and continues to store the information in memory **111**.

In one embodiment, publisher interface **300** is a graphical interface that permits the placement and association of objects in a manner typical of, for example, vector graphics editing programs (such as Adobe Illustrator). Objects located on canvas **305** placed and manipulated by the various commands within publisher interface **300** or inputs such as an input device **117** which may be a keyboard or mouse. As described herein, the contents of canvas **305** may be saved as an Application that, through system **100**, provide the same or a similar placement of objects on screen **137** and have actions defined within publisher interface **300**. Objects placed on canvas **305** are intended for interaction with user of device **130** and are referred to herein, without limitation, as objects or UI (user interface) objects. In addition, the user of interface **300** may assign or associate actions or web bindings to UI objects placed on canvas **305** with result in the programming device **130** that cause it to respond accordingly.

Objects include, but are not limited to input UI objects, response UI objects. Input UI objects include but are not limited to: text fields (including but not limited to alpha, numeric, phone number, or SMS number); text areas; choice objects (including but not limited to returning the selected visible string or returning a numeric hidden attribute); single item selection lists (including but not limited to returning the selected visible string or returning a numeric hidden attribute); multi item selection lists (including but not limited to returning all selected items (visible text string or hidden attribute) or cluster item selection lists (returning the hidden attributes for all items).

Other input UI objects include but are not limited to: check boxes; slide show (including but not limited to returning a numeric hidden attribute, returning a string hidden attribute, or returning the hidden attributes for all slides); and submit function (which can be assigned to any object including submit buttons, vectors, etc.).

Response UI Objects may include, but are not limited to: single line text objects, which include: a text Field (including but not limited to a URL, audio URL, or purchase URL), a text button, a submit button, or a clear button. Another response UI objects include: a multiple line text object, which may include a text area or a paragraph; a check box; an image; a video; a slide show (with either video or image slides, or both); choice objects; list objects; or control lists, which control all the subordinate output UI objects for that web component. Control list objects include, but are not limited to: list type or a choice type, each of which may include a search response list or RSS display list.

As a further example of objects that may be used with authoring tool **112**, Table I lists Data Types, Preferred Input, Input Candidates, Preferred Output and Output Candidates for one embodiment of an authoring tool.

TABLE I

One embodiment of supported objects

Data Types	Preferred Input	Input Candidates	Preferred Output	Output Candidates
boolean	Check Box	Check Box	Check Box	Check Box
Int	Text Field (integer)	Text Field (integer) Text Field (Phone #) Text Field (SMS #) Choice List (single select)	Text Field (integer)	Text Field (integer) Text Field (Phone #) Text Field (SMS #) Choice List (single select) Text Button
String	Text Field (Alpha)	Any	Text Field (Alpha)	Any
multilineString	Text Area	Text Area	Text Area	Text Area Paragraph
ImageURL	N/A	N/A	Image	Image Slide Show
VideoURL	N/A	N/A	Video	Video Slide Show
List	Single Item List	Single Item List Multi-Select List Complex List Choice Slide Show	Single Item List	Any List Type Any Choice Type (see Complex List Specification)
ComplexList	Complex List	Single Item List Multi-Select List Complex List	Single Item List	Any List Type (see Complex List Specification)
Slideshow	Slide Show	Slide Show	Slide Show	Slide Show
SearchResponseList	N/A	N/A	Search Response List	Search Response List Control List Complex List Choice
RSSList	N/A	N/A	RSS Display List	RSS Display List Control List Complex List Choice
SingleSelectionList	Choice	Choice Complex List	Choice	Choice Complex List
MultiSelectionList	Multi-Selection List	Multi-Selection List	Multi-Selection List	Multi-Selection List
ServiceActivation	Submit Button	Any	N/A	N/A
ChannelImageURL	N/A	N/A	Image	Image Video Slide Show
ChannelDescription	N/A	N/A	Text Area	Text Area Paragraph Text Field Text Button List Choice
ChannelTitle	N/A	N/A	Text Field	Text Field Text Button Paragraph Text Area List Choice
URL			Text Field (URL request)	Text Field (URL request)
Audio URL			Text Field (Audio URL request)	Text Field (Audio URL request)
Purchase URL			Text Field (Purchase URL request)	Text Field (Purchase URL request)
Image Data			Image	Image Slide Show
Image List Data			Slide Show	Slide Show Image
Persistent Variable	N/A	N/A	N/A	N/A
Pipeline Multiple Select	Multi-select List	Multi-select List Complex List Slide Show	N/A	N/A
Phone Number	Text Field (numeric type)	Text Field Text Button	Text Field (numeric type)	Text Field Text Button
Hidden Attribute	Complex List	Complex List Slide Show	Complex List	Complex List Slide Show
Collection List	N/A	N/A	Slide Show	Complex List Slide Show

In general, publisher interface **300** permits a user to define an Application as one or more Applications Pages, select UI objects from Menu bar **301** or Tool bar **303** and arrange them on an Applications Page by placing the objects canvas **305**. An Application Page is a page that is available to be visited through any navigation event. Application Pages inherit all the attributes of the Master Page, unless that attribute is specifically changed during an editing session.

Authoring platform **110** also stores information for each UI object on each Application Page of an Application. Layer Inspector **307** provides lists of Applications Pages, UI objects on each Applications Page, and Styles, including templates. Objects may be selected from canvas **305** or Layer Inspector **307** causing Resource Inspector **309** to provide lists of various UI objects attributes which may be selected from within the Resource Inspector. Publisher interface **300** also permits a user to save their work as an Application for layer transfer and operation of device **130**. Publisher interface **300** thus provides an integrated platform for designing the look and operation of device **130**.

The information stored for each UI object depends, in part, on actions which occur as the result of a user of device **130** selecting the UI object from the device. UI objects include, but are not limited to: navigational objects, such as widget or channel launch strips or selection lists; message objects for communicating, such as a multiple chat, video chat, phone and/or SMS lists or fields or a pop-up alert; text fields or areas; check boxes; pull down menus; selection lists and buttons; pictures; slide shows; video or LBS maps; shapes or text defined by a variety of tools; a search response; or an RSS display.

In certain embodiments, publisher interface **300** permits a user to assign action to UI objects, including but not limited to, programming of the device **130** or a request for information over network N. In one embodiment, for example and without limitation, publisher interface **300** has a selection to bind a UI object to a web service—that is, associate the UI object or a manipulation or selection of UI object with web services. Publisher interface **300** may also include many drawing and text input functions for generating displays that may be, in some ways, similar to drawing and/or word processing programs, as well as toolbars and for zooming and scrolling of a workspace.

Each UI object has some form, color, and display location associate with it. Further, for example and without limitation, UI objects may have navigational actions (such as return to home page), communications actions (such as to call the number in a phone number field), or web services (such as to provide and/or retrieve certain information from a web service). Each of the these actions requires authoring platform **110** to store the appropriate information for each action. In addition, UI objects may have associated patent or child objects, default settings, attributes (such as being a password or a phone number), whether a field is editable, animation of the object, all of which may be stored by authoring platform **110**, as appropriate.

Menu bar **301** provides access features of publisher interface **300** through a series of pull-down menus that may include, but are not limited to, the following pull-down menus: a File menu **301***a*, an Edit menu **301***b*, a View menu **301***c*, a Project menu **301***d*, an Objects menu **301***e*, an Events menu **301***f*, a Pages menu **301***g*, a Styles menu **301***h*, and a Help menu **301***i*.

File menu **301***a* provides access to files on authoring platform **110** and may include, for example and without limitation, selections to open a new Application or master page, open a saved Application, Application template, or style template, import a page, alert, or widget, open library objects including but not limited to an image, video, slide show, vector or list, and copying an Application to a user or to Server **120**.

Edit menu **301***b* may include, but is not limited to, selections for select, cut, copy, paste, and edit functions.

View menu **301***c* may include, but is not limited to, selections for zooming in and out, previewing, canvas **305** grid display, and various palette display selections.

Project menu **301***d* may include, but is not limited to, selections related to the Application and Player, such as selections that require a log in, generate a universal Player, generate server pages, activate server APIs and extend Player APIs. A Universal Player will include all the code libraries for the Player, including those that are not referenced by the current Application. Server APIs and Player APIs logically extend the Player with Server-side or device-side Application specific logic.

Objects menu **301***e* includes selections for placing various objects on canvas **305** including, but not limited to: navigation UI objects, including but not limited to widget or channel launch strips or selection lists; message-related UI objects, including but not limited to multiple chat, video chat, phone and/or SMS lists or fields, or a pop-up alert; shapes, which provides for drawing tools; forms-related objects, including but not limited to text fields; scrolling text box, check box, drop-down menu, list menu, submit button or clear button; media-related UI objects such as pictures, slide shows, video or LBS maps; text-related UI objects such as buttons or paragraphs; and variables, including but not limited to time, date and audio mute control.

Events menu **301***f* includes selections for defining child objects, mouse events, animations or timelines.

Pages menu **301***g* includes selection for handling multi-page Applications, and may include selections to set a master page, delete, copy, add or go to Applications Pages.

Styles menu **301***h* includes selections to handle styles, which are the underlying set of default appearance attributes or behaviors that define any object that is attached to a style. Styles are a convenient way for quickly creating complex objects, and for changing a whole collection of objects by just modifying their common style. Selections of Styles menu **301***h* include, but not limited to, define, import, or modify a style, or apply a template. Help menu **301***i* includes access a variety of help topics.

Tool bar **303** provides more direct access to some of the features of publisher interface **300** through a series of pull-down menus. Selections under tool bar **303** may include selections to:

control the look of publisher interface **300**, such as a Panel selection to control the for hiding or viewing various panels on publisher interface **300**;

control the layout being designed, such as an Insert Page selection to permit a user to insert and name pages;

control the functionality of publisher interface **300**, such as a Palettes selection to choose from a variety of specialized palettes, such as a View Palette for zooming and controlling the display of canvas **305**, a Command Palette of common commands, and Color and Shape Palettes;

place objects on canvas **305**, which may include selections such as: a Navigation selection to place navigational objects, such as widget or channel launch strips or selection lists), a Messages selection to place objects for communicating, such as a multiple chat, video chat, phone and/or SMS lists or fields, or a pop-up alert, a Forms selection to place objects such as text fields or areas, check boxes, pull down menus, selection lists, and buttons, a Media selection to place pictures, slide shows, video or LBS maps, and a Shapes selection having a variety of drawing tools, a Text selection for placing text, a search response, or an RSS display, and Palettes.

In one embodiment, Tool bar **303** includes a series of pull-down menus that may include, but are not limited to, items from Menu bar **301** organized in the following pull-down menus: a Panel menu **303***a*, an Insert Page menu **303***b*, a Navigation menu **303***c*, a Messages menu **303***d*, a Forms menu **303***e*, a Media menu **303***f*, a Shapes menu **303***g*, a Text menu **303***h*, and a Palettes menu **301***i*.

Panel menu **303***a* permits a user of authoring platform **110** to change the appearance of interface **300** by, controlling which tools are on the interface or the size of canvas **305**. Insert Page menu **303***b* permits a user of authoring platform **110** to open a new Application Page. Navigation menu **303***c* displays a drop down menu of navigational-related objects such as a widget or channel launch strip or selection list. Messages menu **303***d* displays a drop down menu of messaging-related objects such as multiple chat, video chat, phone or SMS lists or fields, and pop-up alerts. Forms menu **303***e* displays a drop down menu of forms-related objects including, but not limited to, a text field, a text area, a check box, a drop down menu, a selection list, a submit button, and a clear button. Media menu **303***f* displays a drop down menu of media-related objects including, but not limited to, a picture, slide show, video or LBS map. Shapes menu **303***g* displays a drop down menu of draw tools, basic shapes, different types of lines and arrows and access to a shape library. Text menu **303***j* displays a drop down menu of text-related objects, including but not limited to a text button, paragraph, search response, RSS display and variables such as time and date.

Palettes menu **301***i* includes a selection of different palettes that can be moved about publisher interface **300**, where each palette has specialized commands for making adjustments or associations to objects easier. Palettes include, but are not limited to: a page view palette, to permit easy movement between Applications Pages; a view palette, to execute an Application or zoom or otherwise control the viewing of an Application; a commands palette having editing commands; a color palette for selection of object colors; and a shapes palette to facilitate drawing objects.

Layer inspector **307** permits a user of publisher interface **300** to navigate, select and manipulate UI objects on Applications Pages. Thus, for example, a Page/objects panel **307***a* of layer inspector **307** has a listing that may be selected to choose an Applications Pages within and Application, and UI objects and styles within an Applications Page. An Object styles panel **307***b* of layer inspector **307** displays all styles on the Applications Page and permits selection of UI objects for operations to be performed on the objects.

Thus, for example, when objects from Menu bar **301** or Tool bar **303** are placed on canvas **305**, the name of the object appears in Page/objects panel **307***a*. Page/objects panel **307***a* includes a page display **307***a***1** and an objects display **307***a***2**. Page display **307***a***1** includes a pull down menu listing all Applications Pages of the Application, and objects display **307***a***2** includes a list of all objects in the Applications Page (that is, objects on canvas **305**).

In general, page/objects panel **307***a* displays various associations with a UI object and permits various manipulations including, but not limited to, operations for parent and child objects that are assigned to a page, and operations for object styles, and permits navigating between page types and object styles, such as switching between the master page and Application pages and deselecting object styles and alerts, opening an Edit Styles Dialog Box and deselecting any master, Application or alert page, or selecting an alert page and deselecting any Master Page or Application Page. A parent or child object can also be selected directly from the Canvas. In either case, the Resource Inspector can then be used for modifying any attribute of the selected object.

Examples of operations provided by page/objects panel **307***a* on pages include, but are not limited to: importing from either a user's private page library or a public page library; deleting a page; inserting a new page, inheriting all the attributes of the Master Page, and placing the new page at any location in the Page List; editing the currently selected page, by working with an Edit Page Dialog Box. While editing all the functions of the Resource Inspector **309** are available, as described subsequently, but are not applied to the actual page until completing the editing process.

Examples of operations provided by of page/objects panel **307***a* on objects, which may be user interface (UI) objects, include but are not limited to: changing the drawing order layer to: bring to the front, send to the back, bring to the front one layer, or send to the back one layer; hiding (and then reshowing) selected objects to show UI objects obstructed by other UI Objects, delete a selected UI Page Object, and editing the currently selected page, by working with a Edit Page Dialog Box.

Object styles panel **307***b* of layer inspector **307** displays all styles on the Applications Page and permits operations to be performed on objects, and is similar to panel **307***a*. Examples of operations provided by object style panel **307***b* include, but are not limited to: importing from either a user's private object library or a public object library; inserting a new object style, which can be inherited from a currently selected object, or from a previously defined style object; and editing a currently selected object style by working with an Edit Style Dialog Box.

Style attributes can be assigned many attributes, including the look, and behavior of any object that inherits these objects. In addition, List Layout Styles can be created or changed as required. A layout style can define a unbounded set of Complex List Layouts, including but not limited to: the number of lines per item in a list, the number of text and image elements and their location for each line for each item in the last, the color and font for each text element, and the vertical and horizontal offset for each image and text element.

Alerts Panel **307***c* provides a way of providing alert pages, which can have many of the attributes of Application Pages, but they are only activated through an Event such as a user interaction, a network event, a timer event, or a system variable setting, and will be superimposed onto whatever is currently being displayed. Alert Pages all have transparent backgrounds, and they function as a template overlay, and can also have dynamic binding to real time content.

Resource inspector **309** is the primary panel for interactively working with UI objects that have been placed on the Canvas **305**. When a UI object is selected on Canvas **305**, a user of authoring platform **110** may associate properties of the selected object by entering or selecting from resource inspector **309**. In one embodiment, resource inspector **309** includes five tab selections: Setting Tab **309***a*, Events Tab **309***b*, Animation Tab **309***c*, Color Tab **309***d* which includes a color palette for selecting object colors, and Bindings Tab **309***e*.

Settings Tab **309***a* provides a dialog box for the basic configuration of the selected object including, but not limited to, name, size, location, navigation and visual settings. Depending upon the type of object, numerous other attributes could be settable. As an example, the Setting Tab for a Text Field may include dialog boxes to define the text field string, define the object style, set the font name, size and effects, set an object name, frame style, frame width, text attributes (text field, password field, numeric field, phone number, SMS number, URL request).

As an example of Setting Tab **309***a*, FIG. **3**B shows various selections including, but not limited to, setting **309***a***1** for the web page name, setting **309***a***2** for the page size, including selections for specific devices **130**, setting **309***a***3** indicating the width and height of the object, and setting **309***a***4** to select whether background audio is present and to select an audio file.

FIG. **3**C illustrates an embodiment of the Events Tab **309***b*, which includes all end user interactions and time based operations. The embodiment of Events Tab **309***b* in FIG. **3**C includes, for example and without limitation, an Events and Services **309***b***1**, Advanced Interactive Settings **309***b***2**, Mouse State **309***b***3**, Object Selected Audio Setting **309***b***4**, and Work with Child Objects and Mouse Overs button **309***b***5**.

Events and Services **309***b***1** lists events and services that may be applied to the selected objects. These include, but are not limited to, going to external web pages or other Applications pages, either as a new page or by launching a new window, executing an Application or JavaScript method, pausing or exiting, placing a phone call or SMS message, with or without single or multiple Player download, show launch strip, or go back to previous page. Examples of events and services include, but are not limited to those listed in Table II

TABLE II

Events and Services	
Goto External Web Page replacing Current Frame	ChoiceObject: Remove Icon from Launch Strip
Goto External Web Page Launched in a New Window	Goto a specific Internal Web Page with Alert. "Backend Synchronization"
Goto a specific Internal Web Page	Goto Widget Object
Goto the next Internal Web Page	Generate Alert. "With a Fire Event"
Goto External Web Page replacing the Top Frame	Send SMS Message from Linked Text Field
Execute JavaScript Method	Toggle Alert. "Display OnFocus, Hide OffFocus"
Pause/Resume Page Timeout	Execute an Application with Alert. "With a Fire Event"
Execute an Application	Goto Logical First Page
Goto a specific Internal Web Page with setting starting slide	Generate Alert with Backend Synchronization
Exit Application	Send SMS Message with Share (Player Download)
Exit Player	Place PhoneCall from linked Text Field with Share (Player Download)
Place PhoneCall from linked Text Field	Send IM Alert from linked Text Field or Text Area
Text Field/Area: Send String on FIRE	Set and Goto Starting Page
ChoiceObject: Add Icon to Launch Strip	Populate Image
Text Field/Area: Send String on FIRE or Numeric Keys	Preferred Launch Strip

Advanced Interactive Settings **309***b***2** include Scroll Activation Enabled, Timeline Entry Suppressed, Enable Server Listener, Submit Form, Toggle Children on FIRE, and Hide Non-related Children, Mouse State **309***b***3** selections are Selected or Fire. When Mouse State Selected is chosen,

Object Selected Audio Setting **309***b***4** of Inactive, Play Once, Loop, and other responses are presented. When Mouse State Fire is chosen, Object Selected Audio Setting **309***b***4** is replaced with FIRE Audi Setting, with appropriate choices presented.

When Work with Child Objects and Mouse Overs button **309***b***5** is selected, a Child Object Mode box pops up, allowing a user to create a child object with shortcut to Menu bar **301** actions that may be used define child objects.

FIG. **3**D illustrates one embodiment of an Animation Tab **309***c*, which includes all animations and timelines. The Color Tab includes all the possible color attributes, which may vary significantly by object type.

Animation Tab **309***c* includes settings involved in animation and timelines that may be associated with objects. One embodiment of Animation Tab **309***c* is shown, without limitation, in FIG. **3**D, and is described, in Rempell ("Rempell").

A Color Tab **309***d* includes a color palette for selecting object colors.

Bindings Tab **309***e* is where web component operations are defined and dynamic binding settings are assigned. Thus, for example, a UI object is selected from canvas **305**, and a web component may be selected and configured from the bindings tab. When the user's work is saved, binding information is associated with the UI object that will appear on screen **137**.

FIG. **3**E illustrates one embodiment of Bindings Tab and includes, without limitation, the following portions: Web Component and Web Services Operations **309***e***1**, Attributes Exposed list **309***e***2**, panel **309***e***3** which includes dynamic binding of server-side data base values to attributes for the selected object, Default Attribute Value **309***e***4**, Database Name **309***e***5**, Table Name **309***e***6**, Field Name **309***e***7**, Channel Name **309***e***8**, Channel Feed **309***e***9**, Operation **309***e***10**, Select Link **309***e***11**, and Link Set checkbox **309***e***12**.

Web Component and Web Services Operations **309***e***1** includes web components that may be added, edited or removed from a selected object. Since multiple web components can be added to the same object, any combination of mash-ups of 3rd party web services is possible. When the "Add" button of Web Component and Web Services Operations **309***e***1** is selected, a pop-up menu **319**, as shown in FIG. **3**F, appears on publisher interface **300**. Pop-up menu **319** includes, but is not limited to, the options of: Select a Web Component **319***a*; Select Results Page **319***b*; Activation Options **319***c*; Generate UI Objects **319***d*; and Share Web Component **319***e*.

The Select a Web Component **319***a* portion presents a list of web components. As discussed herein, the web components are registered and are obtained from web component registry **220**.

Select Results Page **319***b* is used to have the input and output on different pages—that is, when the Results page is different from Input page. The default selected results page is either the current page, or, if there are both inputs and outputs, it will be set provisionally to the next page in the current page order, if one exists.

Activation Options **319***c* include, if there are no Input UI Objects, a choice to either "Preload" the web component, similar to how dynamic binding, or have the web component executed when the "Results" page is viewed by the consumer.

Generate UI Objects **319***c*, if selected, will automatically generate the UI objects. If not selected, then the author will bind the Web Component Inputs and Results to previously created UI Objects.

Share Web Component **319***e* is available and will become selected under the following conditions: 1) Web Component is Selected which already has been used by the current Application; or 2) the current Input page is also a "Result" page for that Web component. This permits the user of device **130**, after viewing the results, to extend the Web Component allowing the user to make additional queries against the same Web Component. Examples of this include, but are not limited to, interactive panning and zooming for a Mapping Application, or additional and or refined searches for a Search Application.

Dynamic Binding permits the binding of real time data, that could either reside in a 3rd party server-side data base, or in the database maintained by Feed Collector **1010** for aggregating live RSS feeds, as described subsequently with reference to FIG. **10**.

Referring again to FIG. **3**E, Attributes Exposed list **309***e***2** are the attributes available for the selected object that can be defined in real time through dynamic binding.

Panel **309***e***3** exposes all the fields and tables associated with registered server-side data bases. In one embodiment, the user would select an attribute from the "Attributes Exposed List" and then select a data base, table and field to define the real time binding process. The final step is to define the record. If the Feed Collector data base is selected, for example, then the RSS "Channel Name" and the "Channel Feed" drop down menus will be available for symbolically selected the record. For other data bases the RSS "Channel Name" and the "Channel Feed" drop down menus are replaced by a "Record ID" text field.

Default Attribute Value **309***e***4** indicates the currently defined value for the selected attribute. It will be overridden in real time based on the dynamic linkage setting.

Database Name **309***e***5** indicates which server side data base is currently selected.

Table Name **309***e***6** indicates which table of the server side data base is currently selected.

Field Name **309***e***7**, indicates which field form the selected table of the server side data base is currently selected.

Channel Name **309***e***8** indicates a list of all the RSS feeds currently supported by the Feed Collector. This may be replaced by "Record ID" if a data base other than the Feed Collector **1010** is selected.

Channel Feed **309***e***9** indicates the particular RSS feed for the selected RSS Channel. Feed Collector **1010** may maintain multiple feeds for each RSS channel.

Operation **309***e***10**, as a default operation, replaces the default attribute value with the real time value. In other embodiments this operation could be append, add, subtract, multiply or divide.

Select Link **309***e***11** a button that, when pressed, creates the dynamic binding. Touching the "Select Link" will cause the current data base selections to begin the blink is some manner, and the "Select Link" will change to "Create Link". The user could still change the data base and attribute choices. Touching the "Create Link" will set the "Link Set" checkbox and the "Create Link" will be replaced by "Delete Link" if the user wishes to subsequently remove the link. When the application is saved, the current active links are used to create the SPDL.

Link Set checkbox **309***e***12** indicates that a link is currently active.

An example of the design of a display is shown in FIGS. **4**A and **4**B according the system **100**, where FIG. **4**A shows publisher interface **300** having a layout **410** on canvas **305**, and FIG. **4**B shows a device **130** having the resulting layout **420** on screen **137**. Thus, for example, authoring platform **110** is used to design layout **410**. Authoring platform **110** then generates an Application and a Player specific to device **130** of FIG. **4**B. The Application and Player are thus used by device **130** to produce layout **420** on screen **137**.

As illustrated in FIG. **4**A, a user has placed the following on canvas **305** to generate layout **410**: text and background designs **411**, a first text input box **413**, a second text input box **415**, and a button **417**. As an example which is not meant to limit the scope of the present invention, layout **410** is screen prompts a user to enter a user name in box **413** and a password in box **415**, and enter the information by clicking on button **417**.

In one embodiment, all UI objects are initially rendered as Java objects on canvas **305**. When the Application is saved, the UI objects are transformed into the PDL, as described subsequently.

Thus, for example, layout **410** may be produced by the user of authoring platform **110** selecting and placing a first Text Field as box **413** then using the Resource Inspector **309** portion of interface **300** to define its attributes.

Device User Experience

Systems **100** and **200** provide the ability for a very large number of different types of user experiences. Some of these are a direct result of the ability of authoring platform **110** to bind UI objects to components of web services. The following description is illustrative of some of the many types of experiences of using a device **130** as part of system **100** or **200**.

Device **130** may have a one or more of a very powerful and broad set of extensible navigation objects, as well as object- and pointer-navigation options to make it easy to provide a small mobile device screen **137** with content and to navigate easily among page views, between Applications, or within objects in a single page view of an Application.

Navigation objects include various types of launch strips, various intelligent and user-friendly text fields and scrolling text boxes, powerful graphical complex lists, as well as Desktop-level business forms. In fact, every type of object can be used for navigation by assigning a navigation event to it. The authoring tool offers a list of navigation object templates, which then can be modified in numerous ways.

Launch Strips and Graphical List Templates Launch Strips

Launch strips may be designed by the user of authoring platform **110** with almost no restrictions. They can be stationary or appear on command from any edge of the device, their size, style, audio feedback, and animations can be freely defined to create highly compelling experiences.

FIG. **5** shows a display **500** of launch strips which may be on displayed canvas **305** or on screen **137** of device **130** having the proper Player and Application. Display **501** includes a portal-type Launch Strip **501** and a channel-type Launch Strip **502**, either one of which may be included for navigating the Application.

Launch Strip **501** includes UI objects **501***a*, **501***b*, **501***c*, **501***d*, and **501***e* that that becomes visible from the left edge of the display, when requested. UI objects **501***a*, **501***b*, **501***c*, **501***d*, and **501***e* are each associated, through resource inspector **309** with navigational instructions, including but not limited to navigating to a different Applications Page, or requesting web content. When the Applications Page, having been saved by authoring platform **110** and transferred to display **130**, is executed on device **130**, a user of the device may easily navigate the Application.

Launch Strip **502** includes UI objects **502***b*, **502***c*, **502***d*, and **503***e* that that becomes visible from the bottom of the display, when requested. UI objects **501***a*, **501***b*, **501***c*, **501***d*, and **501***e* are each associated, through resource inspector

309 with navigational instructions, including but not limited to navigating to a different Applications Page, or requesting web content. Launch Strip **502** also includes UI objects **502***a* and **503***g*, which include the graphic of arrows, and which provide access to additional navigation objects (not shown) when selected by a user of device **130**. Launch strip **502** may also include sound effects for each channel when being selected, as well as popup bubble help.

Additional navigational features are illustrated in FIG. **6**A as a display of a Channel Selection List **601***a*, in FIG. **6**B as a display of a Widget Selection List **601***b*, and in FIG. **6**C as display of a Phone List **601***c*. Lists **601***a*, **601***b*, and **601***c* may be displayed on canvas **305** or on screen **137** of device **130** having the proper Player and Application. As illustrated, graphical lists **601***a*, **601***b*, and **601***c* may contain items with many possible text and image elements. Each element can be defined at authoring time and/or populated dynamically through one or more Web Service **250** or API. Assignable Navigation Events. All objects, and/or all elements within an object, can be assigned navigation events that can be extended to registered web services or APIs. For example, a Rolodex-type of navigation event can dynamically set the starting slide of the targeted page view (or the starting view of a targeted Application).

In the embodiment of FIGS. **6**A, **6**B, and **6**C, each list **601***a*, **601***b*, and **601***c* has several individual entries that are each linked to specific actions. Thus Channel Selection List **601***a* shows three objects, each dynamically linked to a web service (ESPN, SF 49ers, and Netflix) each providing a link to purchase or obtain items from the Internet. Widget Selection List **601***b* includes several objects presenting different widgets for selecting. Phone List **601***c* includes a list phone number objects of names that, when selected by a user of device **130** cause the number to be dialed Entries in Phone List **601***c* may be generated automatically from either the user's contact list that is resident on the device, or though a dynamic link to any of user's chosen server-side facilities such as Microsoft Outlook, Google Mail, etc. In one embodiment, Phone List **601***c* may be generated automatically using a web component assigned to the Application, which would automatically perform those functions.

In another embodiment, authoring platform **110** allows a navigation selection of objects with a Joy Stick and/or Cursor Keys in all 4 directions. When within a complex object the navigation system automatically adopts to the navigation needs for that object. For coordinate sensitive objects such as geographical information services (GIS) and location-based services (LBS) or virtual tours a soft cursor appears. For Lists, scrolling text areas and chats, Launch strips, and slide shows the navigation process permits intuitive selection of elements within the object. Scroll bars and elevators are optionally available for feedback. If the device has a pointing mechanism then scroll bars are active and simulate the desktop experience.

Personalization and Temporal Adoption

System **100** and **200** permit for the personalization of device **130** by a variety of means. Specifically, what is displayed on screen **137** may depend on either adoption or customization. Adoption refers to the selection of choices, navigation options, etc. are based on user usage patterns. Temporal adoption permits the skins, choices, layouts, content, widgets, etc. to be further influenced by location (for example home, work or traveling) and time of day (including season and day of week). Customization refers to user selectable skins, choices, layouts, dynamic content, widgets, etc. that are available either through a customization on the phone or one that is on the desktop but dynamically linked to the user's other internet connected devices.

To support many personalization functions there must be a convenient method for maintaining, both within a user's session, and between sessions, memory about various user choices and events. Both utilizing a persistent storage mechanism on the device, or a database for user profiles on a server, may be employed.

FIG. **7** shows a display **700** of a mash-up which may be on displayed canvas **305** or on screen **137** of device **130** having the proper Player and Application. Display **700** includes several object **701** that have been dynamically bound, including an indication of time **701***a*, an indication of unread text messages **701***b*, an RSS news feed **701***c* (including 2 "ESPN Top Stories" **701***c***1** and **701***c***2**), components **701***d* from two Web Services—a weather report ("The Weather Channel"), and a traffic report **701***e* ("TRAFFIC.COM").

In assembling the information of display **700**, device **130** is aware of the time and location of the device—in this example the display is for a workday when a user wakes. Device **130** has been customized so that on a work day morning the user wishes to receive the displayed information. Thus in the morning, any messages received overnight would be flagged, the user's favorite RSS sports feeds would be visible, today's weather forecast would be available, and the current traffic conditions between the user's home and office would be graphically depicted. User personalization settings may be maintained as persistent storage on device **130** when appropriate, or in a user profile which is maintained and updated in real-time in a server-side data base.

Push Capable Systems

In another embodiment system **100** or **200** is a push-capable system. As an example, of such systems, short codes may be applied to cereal boxes and beverage containers, and SMS text fields can be applied to promotional websites. In either case, a user of device **130** can text the short code or text field to an SMS server, which then serves the appropriate Application link back to device **130**.

FIG. **8** is a schematic of an embodiment of a push enabled system **800**. System **800** is generally similar to system **100** or **200**. Device **130** is shown as part of a schematic of a push capable system **800** in FIG. **8**. System **800** includes a website system **801** hosting a website **801**, a server **803** and a content server **805**. System **801** is connected to servers **803** and/or **805** through the Internet. Server **803** is generally similar to server **120**, servers **805** is generally similar to server **140**.

In one embodiment, a user sets up a weekly SMS update from website system **801**. System **801** provides user information to server **803**, which is an SMS server, when an update is ready for delivery. Server **803** provides device **130** with an SMS indication that the subscribed information is available and queries the user to see if they wish to receive the update. Website **801** also provides content server **805** with the content of the update. When a user of device **130** responds to the SMS query, the response is provided to content server **805**, which provides device **130** with updates including the subscribed content.

In an alternative embodiment of system **800**, server **803** broadcasts alerts to one or more devices **130**, such as a logical group of devices. The user is notified in real-time of the pending alert, and can view and interact with the massage without interrupting the current Application.

FIG. **9** is a schematic of an alternative embodiment of a push enabled system **900**. System **900** is generally similar to system **100**, **200**, or **800**. In system **900** a user requests information using an SMS code, which is delivered to device

130. System **900** includes a promotional code **901**, a third-party server **903**, and content server **805**. Server **803** is connected to servers **803** and/or **805** through the Internet, and is generally similar to server **120**.

A promotional code **901** is provided to a user of device **130**, for example and without limitation, on print media, such as on a cereal box. The use of device **130** sends the code server **903**. Server **903** then notifies server **805** to provide certain information to device **130**. Server **805** then provides device **130** with the requested information.

Device Routines

Device routines **114** may include, but are not limited to: an authoring tool SDK for custom code development including full set of Java APIs to make it easy to add extensions and functionality to mobile Applications and tie Applications to back-end databases through the content server **140**; an expanding set of web services **250** available through the authoring tool SDK; a web services interface to SOAP/XML enabled web services; and an RSS/Atom and RDF feed collector **1010** and content gateway **1130**.

Authoring Tool SDK for Custom Code Development Including Full Set of Java APIs

In one embodiment, authoring platform **110** SDK is compatible for working with various integrated development environments (IDE) and popular plug ins such as J2ME Polish. In one embodiment the SDK would be another plug in to these IDEs. A large and powerful set of APIs and interfaces are thus available through the SDK to permit the seamless extension of any Application to back end business logic, web services, etc. These interfaces and APIs may also support listeners and player-side object operations.

There is a large set of listeners that expose both player-side events and dynamically linked server side data base events. Some examples of player side events are: player-side time based event, a page entry event, player-side user interactions and player-side object status. Examples of server-side data base events are when a particular set of linked data base field values change, or some filed value exceeds a certain limit, etc.

A superset of all authoring tool functionality is available through APIs for layer-side object operations. These include, but are not limited to: page view level APIs for inserting, replacing, and or modifying any page object; Object Level APIs for modifying any attribute of existing objects, adding definitions to attributes, and adding, hiding or replacing any object.

Authoring Tool SDK Available Web Services

The APIs permit, without limit, respond, with or without relying on back-end business logic, that is, logic that what an enterprise has developed for their business, to any player-side event or server-side dynamically linked data-base, incorporating any open 3rd party web service(s) into the response.

RSS/Atom and RDF Feed Conversion Web Service

FIG. **10** is a schematic of one embodiment a system **1000** having a feed collector **1010**. System **1000** is generally similar to system **100**, **200**, **800**, or **900**. Feed collector **1010** is a server side component of system **100** that collects RSS, ATOM and RDF format feeds from various sources **1001** and aggregates them into a database **1022** for use by the Applications built using authoring platform **110**.

Feed collector **1010** is a standard XML DOM data extraction process, and includes Atom Populator Rule **1012**, RSS Populator Rule **1013**, RDF Populator Rule **1014**, and Custom Populator Rule **1016**, DOM XML Parsers **1011**, **1015**, and **1017**, Feed Processed Data Writer **1018**, Custom Rule

Based Field Extraction **1019**, Rule-based Field Extraction **1020**, Channel Data Controller **1021**, and Database **1022**.

The feed collector is primarily driven by two sets of parameters: one is the database schema (written as SQL DDL) which defines the tables in the database, as well as parameters for each of the feeds to be examined. The other is the feed collection rules, written in XML, which can be used to customize the information that is extracted from the feeds. Each of the feeds is collected at intervals specified by the feed parameter set in the SQL DDL.

Feed collector **1010** accepts information from ATOM, RDF or RSS feed sources **1001**. Using a rules-based populator, any of these feeds can be logically parsed, with any type of data extraction methodology, either by using supplied rules, or by the author defining their own custom extraction rule. The rules are used by the parser to parse from the feed sources, and the custom rule base field extraction replaces the default rules and assembles the parsed information into the database

In particular, Atom Populator Rule **1012**, RSS Populator Rule **1013**, RDF Populator Rule **1014**, Custom Populator Rule **1016**, and DOM XML Parsers **1011**, **1015**, and **1017** are parse information from the feeds **1001**, and Feed Processed Data Writer **1018**, Custom Rule Based Field Extraction **1019**, Rule-based Field Extraction **1020**, and Channel Data Controller **1021**, supply the content of the feeds in Database **1022**, which is accessible through content server **140**.

FIG. **11** is a schematic of an embodiment of a system **1100** having a Mobile Content Gateway **1130**. System **1100** is generally similar to system **100**, **200**, **800**, **900**, or **1000**. System **1100** includes an SDK **1131**, feed collector **1010**, database listener **1133**, transaction server **1134**, custom code **1135** generated from the SDK, Java APIs, Web Services **1137**, and PDL snippets compacted objects **1139**. System **1100** accepts input from Back End Java Code Developer **1120** and SOAP XML from Web Services **1110**, and provides dynamic content to server **140** and Players to devices **130**.

In one embodiment authoring platform **110** produces a Server-side PDL (SPDL) at authoring time. The SPDL resides in server **120** and provides a logical link between the Application's UI attributes and dynamic content in database **1022**. When a user of device **130** requests dynamic information, server **120** uses the SPDL to determine the link required to access the requested content.

In another embodiment Web Services **1137** interface directly with 3rd party Web Services **1110**, using SOAP, REST, JAVA, JavaScript, or any other interface for dynamically updating the attributes of the Application's UI objects.

XSP Web Pages as a Web Service

In one embodiment, a PDL for a page is embedded within an HTML shell, forming one XSP page. The process of forming XSP includes compressing the description of the page and then embedding the page within an HTML shell.

In another embodiment, a PDL, which contains many individual page definitions, is split into separate library objects on the server, so that each page can to presented as a PDL as part of a Web Service.

Prior to compression the code has already been transformed so that there are no dependencies on the original programming language (Java), and The code and data have been reduced by 4 to 10 times.

Compression has two distinct phases. The first takes advantage of how the primitive representations had been assembled, while the second utilizes standard LZ encoding.

The final result is an overall reduction of 40 to 100 times the original size as represented by Java serialized objects.

One embodiment for compacting data that may be used is described in Rempell. In that patent the compressed data is described as being a database. The terminology used here is a PDL, that is the "internal database" of Rempell is equivalent to the PDL of the present Application.

In Rempell, a process for compacting a "database" (that is, generating a compact PDL) is described, wherein data objects, including but not limited to, multi media objects such as colors, fonts, images, sound clips, URLs, threads, and video, including multi level animation, transformation, and time line are compacted. As an extension to Rempell in all cases these objects are reduced and transformed to Boolean, integer and string arrays.

The compression technique involves storing data in the smallest arrays necessary to compactly store web page information. The technique also includes an advanced form of delta compression that reduces integers so that they can be stored in a single byte, a as high water marks.

Thus, for example, the high water mark for different types of data comprising specific web site settings are stored in a header record as Boolean and integer variables and URL and color objects. Data that defines web page, paragraph, text button, and image style and text button, image and paragraph high watermark settings can be stored in one-dimensional arrays as Boolean, integer and string variables and URL, font, image or thread objects at. The URL, color, font, image and thread objects can also be created as required

Data that defines text button, image, paragraph, or other parent objects and paragraph line high watermark settings can be stored in two-dimensional arrays (by web page and by object number) as Boolean, integer, string, floating point variables and URLs. Again, the URL, color, font, image, audio clip, video clip, text area and thread objects can also be created as required. Data that defines a paragraph line and paragraph line segment high watermarks can be stored in three-dimensional arrays (by web page, by paragraph number, and by line number) as Boolean, integer or string variables. Again, the URL, color or font objects can be created as required. Data that defines a paragraph line segment can be stored into four-dimensional arrays (by web page, by paragraph number, by line number and by line number segment) as Boolean, integer or string variables or URL, color and font objects.

As a data field is added, changed or deleted, a determination is made at on whether a value for a given high watermark needs to be changed. If so, it is updated. As a specific method in the build engine is called, a determination is made on whether a feature flag needs to be set. For example, if a particular JAVA method is called, which requires an instance of a certain JAVA Class to be executed by the run time engine, then that JAVA Class is flagged, as well as any supporting methods, variables and/or object definitions.

In one implementation, the header record, the style record, the web page record, and the object records, are carefully defined in a specific order, written in that order, and explicitly cast by object type when read by the run time engine. Exception handling can be implemented to recover from any errors. This helps assure that data integrity is maintained throughout the build and run time processes.

Also described in Rempell is the "run generation process." This is equivalent generating a Player in the present application. This process starts when the build process detects that the user is finished defining the web site (user has saved the web site and invokes the run generation

process), and concludes with the actual uploading of all the necessary web site run time files to the user's server.

In one embodiment, the PDL includes a first record, a "Header" record, which contains can include the following information:

1: A file format version number, used for upgrading database in future releases.
2: The default screen resolution, in virtual pixels, for both the screen width and height. This is usually set to the web designer's screen resolution, unless overwritten by the user.
3: Whether the Application is a web site.
4: Virtual web page size settings. A calculation is performed by the build engine method, in order to calculate what the maximum web page length is, after reformatting all paragraphs on all internal web pages, based on the default screen resolution.
5: Web page and styles high watermarks.
6: The Websitename.

As new web pages or new objects are created by the user, or as text is added to or deleted from a paragraph, or as new styles are created or deleted, appropriate high watermarks are set, in order to show the current number of each of these entities. Thus, the values for the number of active web pages and the number of text button, image, paragraph or other styles are written as high watermarks in the header. The high watermarks for the number of text button, image, paragraph or other objects that exist for each web page, the number of lines for each paragraph object, and the number of line segments for each paragraph line are written within the body of the PDL, and used as settings for each of the loops in the four-dimensional data structure. Because no structural limits are set on the number of web pages, objects per web page, styles, or paragraph size, these high watermarks greatly reduce the external database file size, and the time it takes for the run time engine to process the data stored in its database.

The settings for all paragraph, text button and image styles are then written as a style record based on their high watermark. This data includes Boolean and integer variables, and font and color objects, written as a one-dimensional array, based on the high watermark values for the number of styles that exist.

The body of the PDL is then written. All Boolean values are written inside a four-dimensional loop. The outside loop contains the Boolean values used to define web pages (i.e. a one-dimensional array definition) as well as the high watermarks for the number of text button, image, paragraph or other objects per web page, with the loop set at the high watermark which defines the number of existing web pages for this web site structure. The second level consists of three or more two dimensional loops with the loops set to the high watermarks defining the actual number of text button, image, and paragraph or other objects that appear on any given web page and contains the values used to define web page objects ((i.e. a two-dimensional array definition; web page number by object number). Included within the loop for paragraph objects are the high watermarks for the number of lines for each paragraph object. The third loop is set by the high watermark defining the actual number of paragraph lines that for all paragraphs on any web page and contains the values used to define paragraph lines (i.e. a three-dimensional array definition; web page number by object number by paragraph line.) Included within the loop for paragraph lines are the high watermarks for the number of line segments for each paragraph line. The inner most loop is set by the high watermarks defining the number of line segments per paragraph line and contains the values used to define paragraph line segments (i.e. a four-dimensional array definition; web page number by object number by paragraph line by paragraph line segment).

All integer values are written inside a four-dimensional loop. Their four loops are controlled by the same high watermark settings as used for the Boolean records, and they describe the same logical entities.

Multimedia objects are written inside a two-dimensional loop. They include URL, color, and font objects, and can include other types of objects. A URL object is the encoded form of a URL Address, used by a web browser or a JAVA method to access files and web addresses. All multimedia objects must be serialized before they can be written. This means that the objects are converted into a common external definition format that can be understood by the appropriate deserialization technique when they are read back in and cast into their original object structure. The outside loop contains web page related objects, and the inner loop contains image, text button, paragraph, etc. related URL, color, and font objects. The outer loop is defined by the web page high watermark and the inner loops by the high watermarks for the actual number of text button, image, paragraph or other objects on a web page.

String records are written inside a four-dimensional loop. The outer loop may be empty. The second loop can include the string values for text button objects, audio and video filenames, and audio and video channel names. The third loop contains values for paragraph line related data, and the innermost loop contains the values for paragraph line segment definitions. The string records are controlled by the same high watermarks as those used for Boolean and integer records. String records are stored utilizing an appropriate field delimiter technology. In one implementation, a UTF encoding technology that is supported by JAVA is utilized.

Single and double floating-point, and long integer records are written inside a two-dimensional loop. The outer loop may be empty. The inner loop contains mathematical values required for certain animations and image processing algorithms. The single and double floating-point, and long integer records are controlled by the same high watermarks as those used for Boolean and integer records.

In one embodiment, a versionizing program analyzes the feature flags, and only those variable definitions, defined in the "Main" object class, relating to the object classes and methods that will be executed at run time, are extracted. All references to object classes that will be called at run time are extracted, creating the source code for the run engine "Main" object class that is ready for compilation.

All external image, video and audio files are resolved. The external references can be copied to designated directories, either on the user's local disk or file server. The file Pathnames can be changed to reflect these new locations. During the installation of the build tools, the necessary class libraries are either installed on the local system or made available on the server where the build tools can be optionally located. The necessary environmental variables are set to permit normal access to the required class libraries.

The customized run engine and a library of the referenced run time classes are compiled and converted into byte code. Finally, the run time engine for the web site is created. The required set of class objects required at run time is flagged for inclusion into the CAB/JAR file.

Next, an HTML Shell File (HSF) is constructed. The first step of this process is to determine whether the dynamic web page and object resizing is desired by testing the Application setting. If the Application was a web page, and thus requiring dynamic web page and object resizing, virtual screen resolution settings are placed in an appropriate HTML compliant string. If the Application is a banner or other customized Application, the absolute values for the run time object (applet size) height and width are placed in an appropriate HTML compliant string as absolute width and height values.

An analysis is made for the background definition for the first internal web page. If a background pattern is defined, an appropriate HTML compliant string for setting the HTML "background" to the same background image is generated. If the first web page definition is a color instead, then the RGB values from those colors are converted to hexadecimal and an appropriate HTML compliant String is generated setting the "bgcolor" to the required hexadecimal value. This process synchronizes the web page background with the background that will be drawn by the web browser when it first interprets the HSF.

Thereafter, a JAVA method generates HTML and JavaScript compliant strings, that when executed by a web browser, generate additional sets of HTML and JavaScript compliant strings that are again executed by the web browser. More specifically, if the Application required dynamic web page and object resizing then JavaScript and HTML compliant strings are generated so that, when interpreted by the web browser at the time the HTML Shell File is initialized, the screen resolution sensing JAVA applet (SRS) will be executed. JavaScript code is generated in order to enable JavaScript to SRS applet communication. In one implementation, the code is generated by performing the following functions:

1: Determine the current web browser type.
2: Load the SRS from either a JAR or CAB File, based on web browser type.
3: Enter a timing loop, interrogating when the SRS is loaded.
4: When the SRS returns an "available" status, interrogate the SRS, which will return the current screen and window's actual height and width.
5: Convert the virtual screen resolution settings into appropriate absolute screen width and height values.

Strings defining additional JavaScript code are generated that perform the following steps at the time the HSF is initialized by the web browser:

1: Generate HTML compliant strings that set the run time engine's applet size to the appropriate values.
2: Generate an HTML complaint string that contains a "param" definition for linking the run time engine to the PDL.
3: Generate an HTML complaint string, dependent upon the type of web browser, which causes the current web browser to load either the JAR or the CAB File(s).
4: Generate JavaScript Code compliant strings that create and dynamically write the applet size defining HTML strings utilizing the JavaScript "document.write" function. This dynamically created code causes the web browser to execute the run time engine, in the correctly sized window, from the correct JAR or CAB file, and linked to the external database.

The writing out the above-generated HTML and JavaScript compliant strings creates the HSF. The necessary security policy permissions are asserted, and a "Websitename".html file is created.

In one embodiment, the processes for creating the CAB and JAR Files is as follows. The image objects, if any, which were defined on the first internal web page are analyzed. If they are set to draw immediately upon the loading of the first web page, then they are flagged for compression and inclusion in the CAB and JAR Files. The feature flags are analyzed to determine which JAVA classes have been compiled. These class files are flagged for compression and inclusion in the library CAB and JAR Files. Strings that are BAT compliant definitions are created that will, when executed in DOS, create compressed CAB and JAR Files. These CAB and JAR Files contain the compressed versions of all necessary JAVA class files, image files, the "Websitename".class, customized run time engine file, and the "Websitename".dta database file. In one implementation of the invention, two BAT files are created. The first, when executed, will create a CAB/JAR file with the "Websitename".dta database file and the customized "main" run time engine, excluding all the image and button object animation, transformation, and image processing code. The second BAT file, when executed, will create a CAB/JAR file with all the library of all the referenced image and button object animation, transformation, and image processing code.

The necessary security policy permissions for file creation are then asserted, and "Websitename".bat and "Websitenamelib".bat files are written. The "Websitename".bat and "Websitename".bat files are then executed under DOS, creating compressed "Websitename".cab and "Websitenamelib".cab files and compressed "Websitename" jar and "Websitenamelib" jar files. The HTML Shell File and the JAR and CAB files are then, either as an automatic process, or manually, uploaded to the user's web site. This completes the production of an XSP page that may be accessed through a web browser.

Displaying Content on a Device

Decompression Management

Authoring platform **110** uses compaction to transform the code and data in an intelligent way while preserving all of the original classes, methods and attributes. This requires both an intelligent server engine and client (handset) Player, both of which fully understand what the data means and how it will be used.

The compaction technology described above includes transformation algorithms that deconstruct the logic and data into their most primitive representations, and then reassembles them in a way that can be optimally digested by further compression processing. This reassembled set of primitive representations defines the PDL of authoring platform **110**.

Prior to compression the code has already been transformed so that there are no dependencies on the original programming language (Java). The data is then compressed by first taking advantage of how the primitive representations had been assembled, and then by utilizing standard LZ encoding. The final result is an overall reduction of 40 to 100 times the original size as represented by Java serialized objects.

The Player, when preparing a page view for execution, decompresses and then regenerate the original objects, but this time in compliance with the programming APIs of device **130**. Specifically, device **130** operates on compacted image pages, one at a time. The cache manager retrieves, decompresses, and reassembles the compacted page images into device objects, which are then interpreted by device **130** for display on screen **137**.

Response Director

In one embodiment, system **100** includes a Response Director, which determines a user's handset, fetches the correct Application from different databases, and delivers a respective highly compressed Application in a PDL format over the air (OTA).

In one embodiment, the Response Director operates on a network connected computer to provide the correct Player to a given device based on the information the device sent to it. As an example, this may occur when a device user enters their phone number into some call-to-action web page. The response director is called and sends an SMS message to the device, which responds, beginning the recognition process.

FIG. **12** illustrates one embodiment of a system **1200** that includes a response director **210**, a user agent database **1201**, an IP address database **1203**, and a file database **1205**. System **1200** is generally similar to system **100**, **200**, **800**, **900**, **1000**, or **1100**.

Databases **1201**, **1203**, and **1205** may reside on server **120**, **210**, or any computer system in communication with response director **210**. System **1200**, any mobile device can be serviced, and the most appropriate Application for the device will be delivered to the device, based on the characteristics of the device.

User agent database **1201** includes user agent information regarding individual devices **130** that are used to identify the operating system on the device. IP address database **1203** identifies the carrier/operator of each device **130**. File database **1205** includes data files that may operate on each device **130**.

The following is an illustrative example of the operation of response director **210**. First, a device **1300** generates an SMS message, which automatically sends an http:// stream that includes handset information and its phone number to response director **210**. Response director **210** then looks at a field in the http header (which includes the user agent and IP address) that identifies the web browser (i.e., the "User Agent"). The User Agent prompts a database lookup in user agent database **1201** which returns data including, but not limited to, make, model, attributes, MIDP 1.0 MIDP 2.0, WAP and distinguishes the same models from different countries. A lookup of the IP address in IP address **1203** identifies the carrier/operator.

File database **1205** contains data types, which may include as jad1, jad2, html, wml/wap2, or other data types, appropriate for each device **130**. A list of available Applications are returned to a decision tree, which then returns, to device **130**, the Application that is appropriate for the respective device. For each file type, there is an attributes list (e.g., streaming video, embedded video, streaming audio, etc.) to provide enough information to determine what to send to the handset.

Response director **210** generates or updates an html or jad file populating this text file with the necessary device and network dependent parameters, including the Application dependent parameters, and then generate, for example, a CAB or JAD file which contains the necessary Player for that device. For example, the jad file could contain the operator or device type or extended device-specific functions that the player would then become aware of.

If there is an Application that has a data type that device **130** cannot support, for example, video, response director **210** sends an alternative Application to the handset, for example one that has a slide show instead. If the device cannot support a slide show, an Application might have text and images and display a message that indicates it does not support video.

Another powerful feature of response director **210** is its exposed API from the decision tree that permits the overriding of the default output of the decision tree by solution providers. These solution providers are often licensees who want to further refine the fulfillment of Applications and Players to specific devices beyond what the default algorithms provide. Solution providers may be given a choice of Applications and then can decide to use the defaults or force other Applications.

Authoring platform **110** automatically scales Applications at publishing time to various form factors to reduce the amount of fragmentation among devices, and the Response Director serves the appropriately scaled version to the device. For example, a QVGA Application will automatically scale to the QCIF form factor. This is important because one of the most visible forms of fragmentation resides in the various form factors of wireless, and particularly mobile, devices, which range from 128×128, 176×208, 240×260, 220×220, and many other customized sizes in between.

FIG. **13** is a schematic of an embodiment of a system **1300**. System **1300** is generally similar to system **1200**. System **1300** is an overview of the entire Player fulfillment process, starting with the generation of players during the player build process.

System **1300** includes response director **210**, a device characteristics operator and local database **1301**, a player profile database **1303** and a player build process **1305**, which may be authoring platform **110**.

As an example of system **1300**, when response director **210** receives an SMS message from device **130**, the response director identifies the device characteristics operator and locale from database **1301** and a Player URL from database **1303** and provides the appropriate Player to the device.

In another embodiment, Player P extend the power of response director **210** by adapting the Application to the resources and limitations of any particular device. Some of these areas of adaptation include the speed of the device's microprocessor, the presence of device resources such as cameras and touch screens. Another area of adaptation is directed to heap, record store and file system memory constraints. In one embodiment, the Player will automatically throttle down an animation to the frame rate that the device can handle so that the best possible user experience is preserved. Other extensions include device specific facilities such as location awareness, advanced touch screen interactions, push extensions, access to advanced phone facilities, and many others

Memory Management

In one embodiment, Player P includes a logical page virtual memory manager. This architecture requires no supporting hardware and works efficiently with constrained devices. All page view images, which could span multiple Applications, are placed in a table as highly compacted and compressed code. A typical page view will range from 500 bytes up to about 1,500 bytes. (See, for example, the Rempell patent) When rolled into the heap and instantiated this code increases to the more typical 50,000 up to 250,000 bytes. Additional alert pages may also be rolled into the heap and superimposed on the current page view. Any changes to any page currently downloaded are placed in a highly compact change vector for each page, and rolled out when the page is discarded. Note that whenever an Application is visited that had previously been placed in virtual memory the Server is interrogated to see if a more current version is available, and, if so, downloads it. This means that Application logic can be changed in real-time and the results immediately available to mobile devices.

To operate efficiently with the bandwidth constraints of mobile devices, authoring platform **110** may also utilize anticipatory streaming and multi-level caching. Anticipatory streaming includes multiple asynchronous threads and IO request queues. In this process, the current Application is

scanned to determine if there is content that is likely to be required in as-yet untouched page views. Anticipatory streaming also looks for mapping Applications, where the user may zoom or pan next so that map content is retrieved prior to the user requesting it. For mapping applications, anticipatory streaming downloads a map whose size is greater than the map portal size on the device and centered within the portal. Any pan operation will anticipatory stream a section of the map to extend the view in the direction of the pan while, as a lower priority, bring down the next and prior zoom levels for this new geography. Zooming will always anticipatory stream the next zoom level up and down.

Multi-level caching determines the handset's heap through an API, and also looks at the record store to see how much memory is resident. This content is placed in record store and/or the file system, and may, if there is available heap, also place the content there as well. Multi-level caching permits the management of memory such that mobile systems best use limited memory resources. Multi-level caching is a memory management system with results similar to embedding, without the overhead of instantiating the content. In other words, with multi-level caching, handset users get an "embedded" performance without the embedded download. Note that when content is flagged as cacheable and is placed in persistent storage, a digital rights management (DRM) solution will be used.

One embodiment of each of the methods described herein is in the form of a computer program that executes on a processing system. Thus, as will be appreciated by those skilled in the art, embodiments of the present invention may be embodied as a method, an apparatus such as a special purpose apparatus, an apparatus such as a data processing system, or a carrier medium, e.g., a computer program product. The carrier medium carries one or more computer readable code segments for controlling a processing system to implement a method. Accordingly, aspects of the present invention may take the form of a method, an entirely hardware embodiment, an entirely software embodiment or an embodiment combining software and hardware aspects. Furthermore, the present invention may take the form of carrier medium (e.g., a computer program product on a computer-readable storage medium) carrying computer-readable program code segments embodied in the medium. Any suitable computer readable medium may be used including a magnetic storage device such as a diskette or a hard disk, or an optical storage device such as a CD-ROM.

It will be understood that the steps of methods discussed are performed in one embodiment by an appropriate processor (or processors) of a processing (i.e., computer) system executing instructions (code segments) stored in storage. It will also be understood that the invention is not limited to any particular implementation or programming technique and that the invention may be implemented using any appropriate techniques for implementing the functionality described herein. The invention is not limited to any particular programming language or operating system. It should thus be appreciated that although the coding for programming devices has not be discussed in detail, the invention is not limited to a specific coding method. Furthermore, the invention is not limited to any one type of network architecture and method of encapsulation, and thus may be utilized in conjunction with one or a combination of other network architectures/protocols.

Reference throughout this specification to "one embodiment," "an embodiment," or "certain embodiments" means that a particular feature, structure or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, appearances of the phrases "in one embodiment," "in an embodiment," or "in certain embodiments" in various places throughout this specification are not necessarily all referring to the same embodiment. Furthermore, the particular features, structures or characteristics may be combined in any suitable manner, as would be apparent to one of ordinary skill in the art from this disclosure, in one or more embodiments.

Throughout this specification, the term "comprising" shall be synonymous with "including," "containing," or "characterized by," is inclusive or open-ended and does not exclude additional, unrecited elements or method steps. "Comprising" is a term of art which means that the named elements are essential, but other elements may be added and still form a construct within the scope of the statement. "Comprising" leaves open for the inclusion of unspecified ingredients even in major amounts.

Similarly, it should be appreciated that in the above description of exemplary embodiments, various features of the invention are sometimes grouped together in a single embodiment, figure, or description thereof for the purpose of streamlining the disclosure and aiding in the understanding of one or more of the various inventive aspects. This method of disclosure, however, is not to be interpreted as reflecting an intention that the claimed invention requires more features than are expressly recited in each claim. Rather, as the following claims reflect, inventive aspects lie in less than all features of a single foregoing disclosed embodiment, and the invention may include any of the different combinations embodied herein. Thus, the following claims are hereby expressly incorporated into this Mode(s) for Carrying Out the Invention, with each claim standing on its own as a separate embodiment of this invention.

Thus, while there has been described what is believed to be the preferred embodiments of the invention, those skilled in the art will recognize that other and further modifications may be made thereto without departing from the spirit of the invention, and it is intended to claim all such changes and modifications as fall within the scope of the invention. For example, any formulas given above are merely representative of procedures that may be used. Functionality may be added or deleted from the block diagrams and operations may be interchanged among functional blocks. Steps may be added or deleted to methods described within the scope of the present invention.

We claim:

1. A system for generating code to provide content on a display of a device, said system comprising:

computer memory storing a registry of:

a) symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network, where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service, where each symbolic name has an associated data format class type corresponding to a subclass of User Interface (UI) objects that support the data format type of the symbolic name, and has a preferred UI object, and

b) an address of the web service;

an authoring tool configured to:

define a (UI) object for presentation on the display, where said defined UI object corresponds to a web component included in said registry selected from a group consisting of an input of the web service and an output of the web service, where each defined UI object is either: 1) selected by a user of the authoring tool; or 2) automatically selected by the system as the preferred UI object corresponding to the symbolic name of the web component selected by the user of the authoring tool,

access said computer memory to select the symbolic name corresponding to the web component of the defined UI object,

associate the selected symbolic name with the defined UI object, where the selected symbolic name is only available to UI objects that support the defined data format associated with that symbolic name, and

produce an Application including the selected symbolic name of the defined UI object, where said Application is a device-independent code; and

a Player, where said Player is a device-dependent code, wherein, when the Application and Player are provided to the device and executed on the device, and when the user of the device provides one or more input values associated with an input symbolic name to an input of the defined UI object,

1) the device provides the user provided one or more input values and corresponding input symbolic name to the web service,

2) the web service utilizes the input symbolic name and the user provided one or more input values for generating one or more output values having an associated output symbolic name,

3) said Player receives the output symbolic name and corresponding one or more output values and provides instructions for the display of the device to present an output value in the defined UI object.

2. The system of claim **1**, where said registry includes definitions of input and output related to said web service.

3. The system of claim **1**, where said web component is a text chat, a video chat, an image, a slideshow, a video, or an RSS feed.

4. The system of claim **1**, where said UI object is an input field for a chat.

5. The system of claim **1**, where said UI object is an input field for a web service.

6. The system of claim **1**, where said UI object is an input field usable to obtain said web component, where said input field includes a text field, a scrolling text box, a check box, a drop down-menu, a list menu, or a submit button.

7. The system of claim **1**, where said web component is an output of a web service, is the text provided by one or more simultaneous chat sessions, is the video of a video chat session, is a video, an image, a slideshow, an RSS display, or an advertisement.

8. The system of claim **1**, where said authoring tool is further configured to:

define a phone field or list; and

generate code that, when executed on the device, allows a user to supply a phone number to said phone field or list.

9. The system of claim **1**, where said authoring tool is further configured to:

define a SMS field or list; and

generate code that, when executed on the device, allows a user to supply an SMS address to said SMS field or list.

10. The system of claim **1**,

where said code includes three or more codes, where one of said three or more codes is device specific, and where two of said three or more codes is device independent.

11. The system of claim **1**, where said code is provided over said network.

12. The system of claim **1**, wherein said defined UI object corresponds to a widget.

13. The system of claim **1**, where said Player is activated and runs in a web browser.

14. The system of claim **1**, where said Player is a native program.

15. A method of displaying content on a display of a device having a Player, where said Player is a device-dependent code, said method comprising:

defining a user interface (UI) object for presentation on the display, where said UI object corresponds to a web component included in a registry of one or more web components selected from a group consisting of an input of a web service and an output of the web service, where each web component includes a plurality of symbolic names of inputs and outputs associated with each web service, and where the registry includes: a) symbolic names required for evoking one or more web components each related to a set of inputs and outputs of the web service obtainable over a network, where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service, and b) an address of the web service, and where each defined UI object is either: 1) selected by a user of an authoring tool; or 2) automatically selected by a system as a preferred UI object corresponding to a symbolic name of the web component selected by the user of the authoring tool;

selecting the symbolic name from said web component corresponding to the defined UI object, where the selected symbolic name has an associated data format class type corresponding to a subclass of UI objects that support the data format type of the symbolic name, and has the preferred UI object;

associating the selected symbolic name with the defined UI object; and

producing an Application including the selected symbolic name of the defined UI object, where said Application is a device-independent code, wherein, when the Application and Player are provided to the device and executed on the device, and when the user of the device provides one or more input values associated with an input symbolic name to an input of the defined UI object,

1) the device provides the user provided one or more input values and corresponding input symbolic name to the web service,

2) the web service utilizes the input symbolic name and the user provided one or more input values for generating one or more output values having an associated output symbolic name,

3) said Player receives the output symbolic name and corresponding one or more output values and provides instructions for the display of the device to present an output value in the defined UI object.

16. The method of claim **15**, where said registry includes definitions of input and output related to said web service.

17. The method of claim **15**, where said web component is a text chat, a video chat, an image, a slideshow, a video, or an RSS feed.

18. The method of claim **15**, where said UI object is an input field for a chat.

19. The method of claim **15**, where said UI object is an input field for a web service.

20. The method of claim **15**, where said UI object is an input field usable to obtain said web component, where said input field includes a text field, a scrolling text box, a check box, a drop down-menu, a list menu, or a submit button.

21. The method of claim **15**, where said web component is an output of a web service, is the text provided by one or more simultaneous chat sessions, is the video of a video chat session, is a video, an image, a slideshow, an RSS display, or an advertisement.

22. The method of claim **15**, further comprising:

defining a phone field or list; and

generating code that, when executed on the device, allows a user to supply a phone number to said phone field or list.

23. The method of claim **15**, further comprising:

defining a SMS field or list; and

generating code that, when executed on the device, allows a user to supply an SMS address to said SMS field or list.

24. The method of claim **15**, and such that said Player interprets dynamically received, device independent values of the web component defined in the Application.

25. The method of claim **15**, further comprising:

providing said Application and Player over said network.

26. The method of claim **15**, wherein said UI object corresponds to a widget.

27. The method of claim **15**, where said Player is activated and runs in a web browser.

28. The method of claim **15**, where said Player is a native program.

* * * * *

EXHIBIT E

US009928044B2

(12) **United States Patent**
Rempell et al.

(10) **Patent No.:** **US 9,928,044 B2**
(45) **Date of Patent:** ***Mar. 27, 2018**

(54) **SYSTEMS AND METHODS FOR PROGRAMMING MOBILE DEVICES**

(71) Applicant: **Express Mobile, Inc.**, Novato, CA (US)

(72) Inventors: **Steven H. Rempell**, Novato, CA (US); **David Chrobak**, Clayton, CA (US); **Ken Brown**, San Martin, CA (US)

(73) Assignee: **Express Mobile, Inc.**, Novato, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **15/706,746**

(22) Filed: **Sep. 17, 2017**

(65) **Prior Publication Data**

US 2018/0004493 A1 Jan. 4, 2018

Related U.S. Application Data

(63) Continuation of application No. 15/370,990, filed on Dec. 6, 2016, now Pat. No. 9,766,864, which is a

(Continued)

(51) **Int. Cl.**

G06F 3/048	(2013.01)
G06F 9/44	(2018.01)
H04L 29/08	(2006.01)
G06F 3/0484	(2013.01)
G06F 3/0482	(2013.01)
H04L 12/58	(2006.01)
H04L 29/06	(2006.01)

(52) **U.S. Cl.**
CPC ***G06F 8/38*** (2013.01); ***G06F 3/0482*** (2013.01); ***G06F 3/04842*** (2013.01); ***G06F 8/34*** (2013.01); ***G06F 9/4443*** (2013.01); ***H04L 51/046*** (2013.01); ***H04L 65/60*** (2013.01); ***H04L 67/02*** (2013.01)

(58) **Field of Classification Search**
CPC G06F 3/048
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2004/0055017	A1	3/2004	Delpuch et al.
2004/0163020	A1	8/2004	Sidman

(Continued)

OTHER PUBLICATIONS

Stina Nylander et al. "The Ubiquitous Interactor-Device Independent Access to Mobile Services" (Computer-Aided Design for User Interfaces IV, Proceedings of the Fifth International Conference on Computer-Aided Design of User Interfaces CADUI'2004, Jan. 2004, pp. 271-282).*

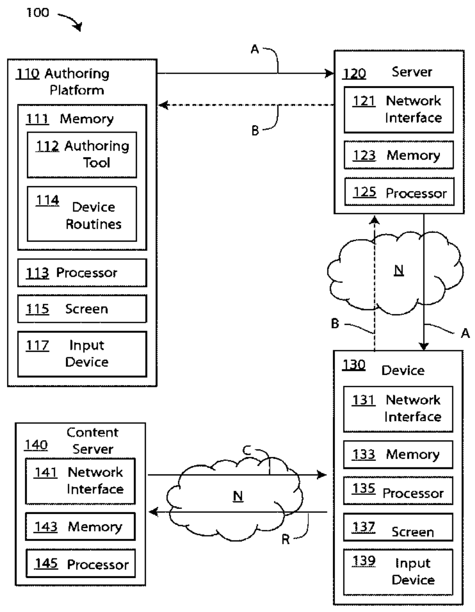
(Continued)

Primary Examiner — Xuyang Xia
(74) *Attorney, Agent, or Firm* — Steven R. Vosen

(57) **ABSTRACT**

Embodiments of a system and method are described for generating and distributing programming to mobile devices over a network. Devices are provided with Players specific to each device and Applications that are device independent. Embodiments include a full-featured WYSIWYG authoring environment, including the ability to bind web components to objects.

28 Claims, 18 Drawing Sheets



Related U.S. Application Data

continuation of application No. 14/708,094, filed on May 8, 2015, now Pat. No. 9,542,163, which is a continuation of application No. 12/936,395, filed as application No. PCT/US2009/039695 on Apr. 6, 2009, now Pat. No. 9,063,755.

(60) Provisional application No. 61/123,438, filed on Apr. 7, 2008, provisional application No. 61/113,471, filed on Nov. 11, 2008, provisional application No. 61/166,651, filed on Apr. 3, 2009.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2004/0199614	A1 *	10/2004	Shenfield	H04L 29/06 709/220
2005/0149935	A1	7/2005	Benedetti	
2005/0273705	A1 *	12/2005	McCain	G06F 17/24 715/234
2006/0063518	A1	3/2006	Paddon et al.	

OTHER PUBLICATIONS

International Search Report and Written Opinion—PCT/US2009/039695—Aug. 21, 2009.
International Preliminary Report on Patentability and Written Opinion—PCT/US2009/039695—Oct. 21, 2010.
Rempell et al, co-owned U.S. Pat. No. 9,471,287, Issue date of Oct. 18, 2016.
Rempell et al, co-owned U.S. Pat. No. 9,507,571, Issue date of Nov. 29, 2016.
Rempell et al, co-owned U.S. Pat. No. 9,542,163, Issue date of Nov. 29, 2016.
Rempell et al, co-owned U.S. Pat. No. 9,766,864, Issue date of Sep. 19, 2017.

* cited by examiner

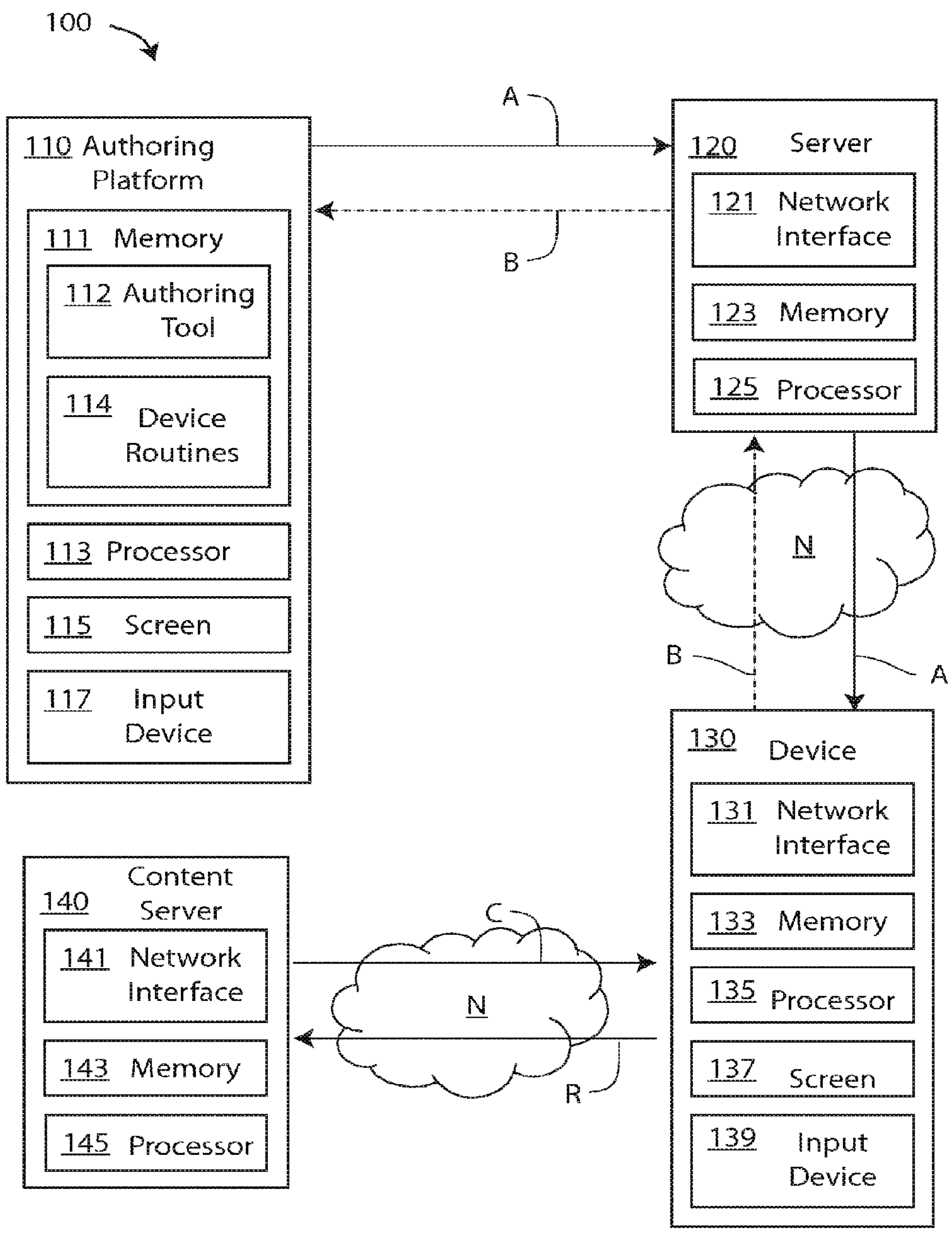

100
110 Authoring Platform
111 Memory
112 Authoring Tool
114 Device Routines
113 Processor
115 Screen
117 Input Device
A
B
120 Server
121 Network Interface
123 Memory
125 Processor
N
B
A
130 Device
131 Network Interface
133 Memory
135 Processor
137 Screen
139 Input Device
140 Content Server
141 Network Interface
143 Memory
145 Processor
C
N
R

FIG. 1A

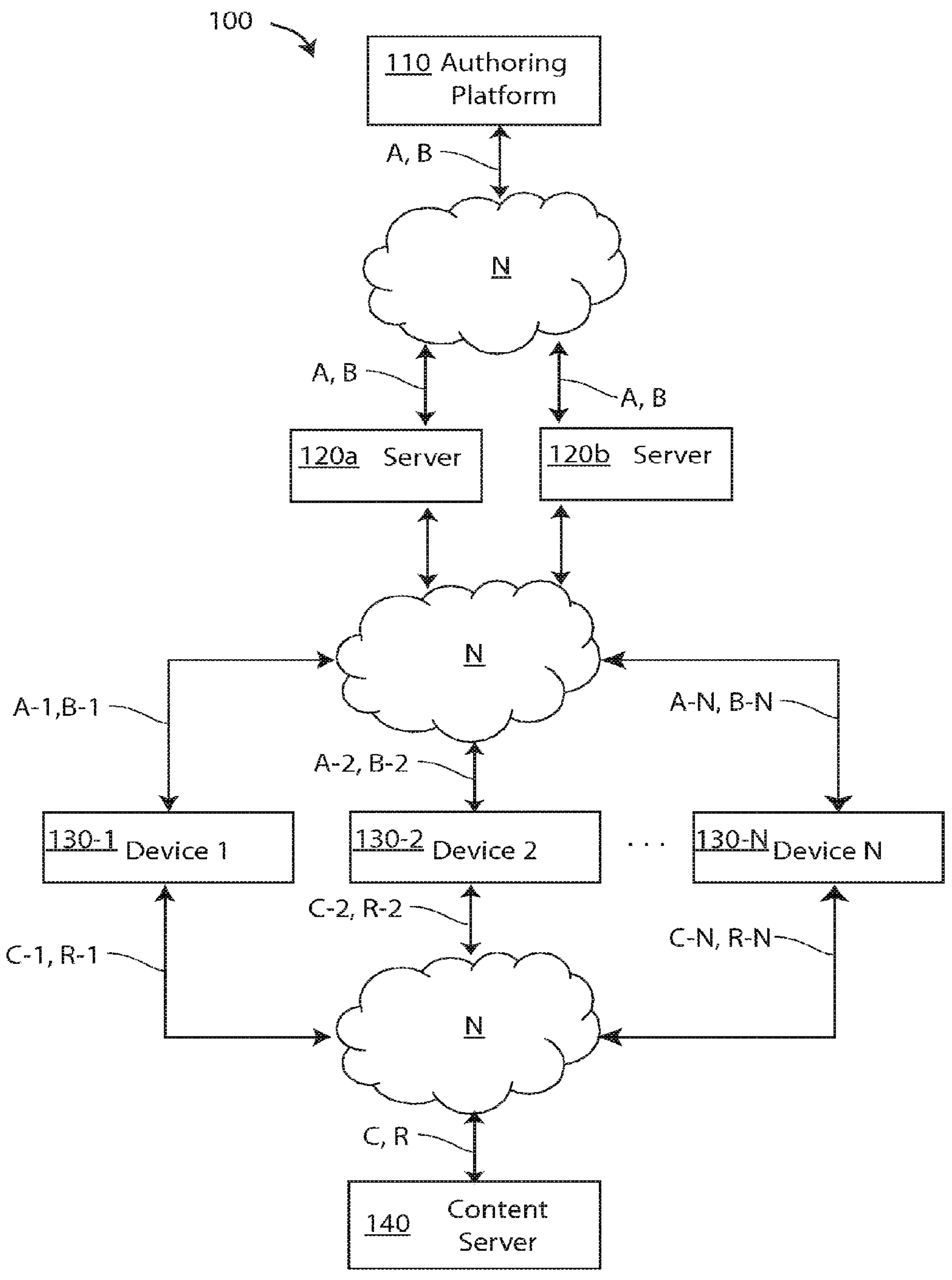

100
110 Authoring Platform
A, B
N
A, B
A, B
120a Server
120b Server
N
A-1,B-1
A-N, B-N
A-2, B-2
130-1 Device 1
130-2 Device 2
. . .
130-N Device N
C-2, R-2
C-1, R-1
C-N, R-N
N
C, R
140 Content Server

FIG. 1B

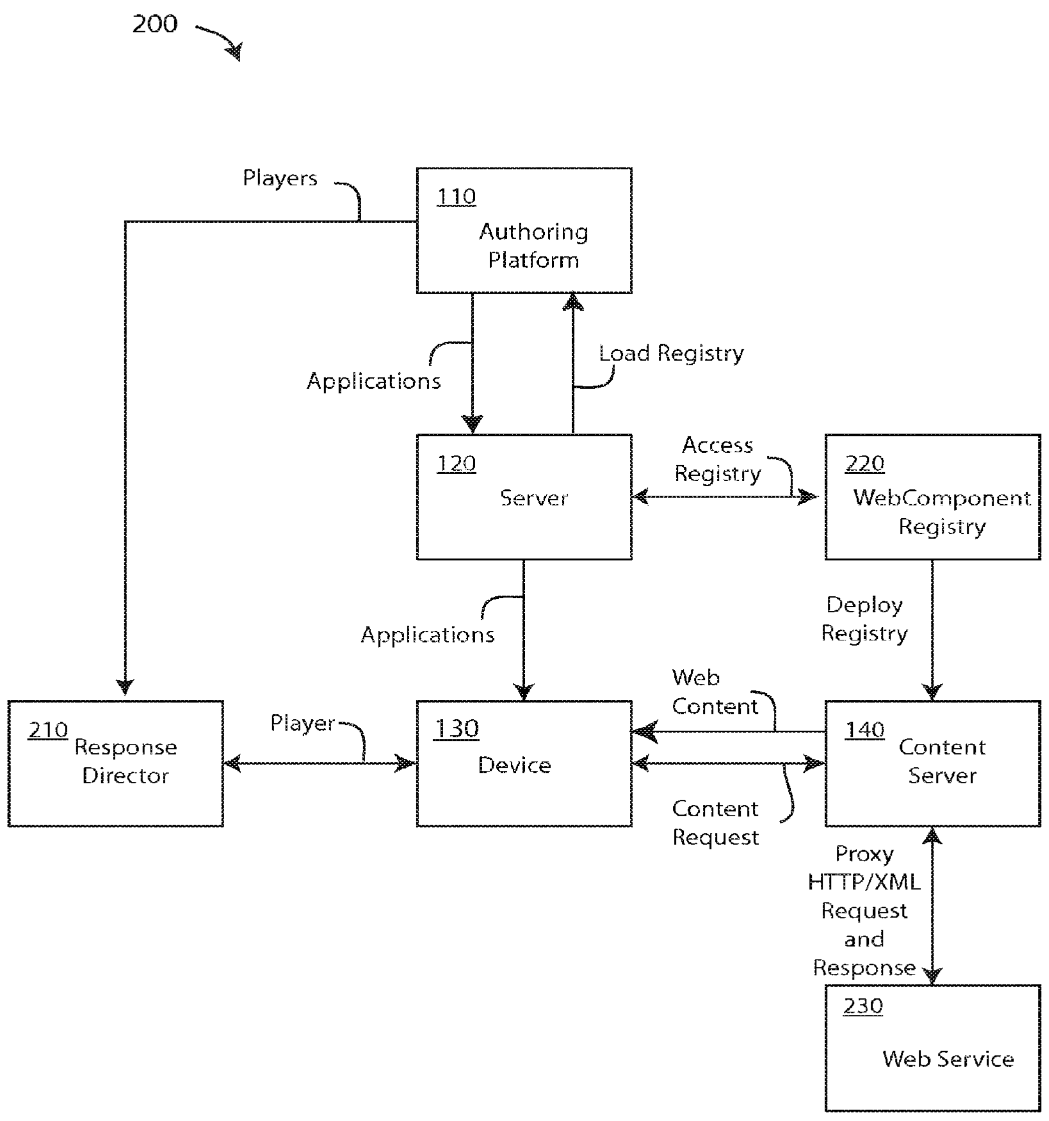

200
Players
110 Authoring Platform
Applications
Load Registry
120 Server
Access Registry
220 WebComponent Registry
Applications
Deploy Registry
Web Content
210 Response Director
Player
130 Device
140 Content Server
Content Request
Proxy HTTP/XML Request and Response
230 Web Service

FIG. 2A

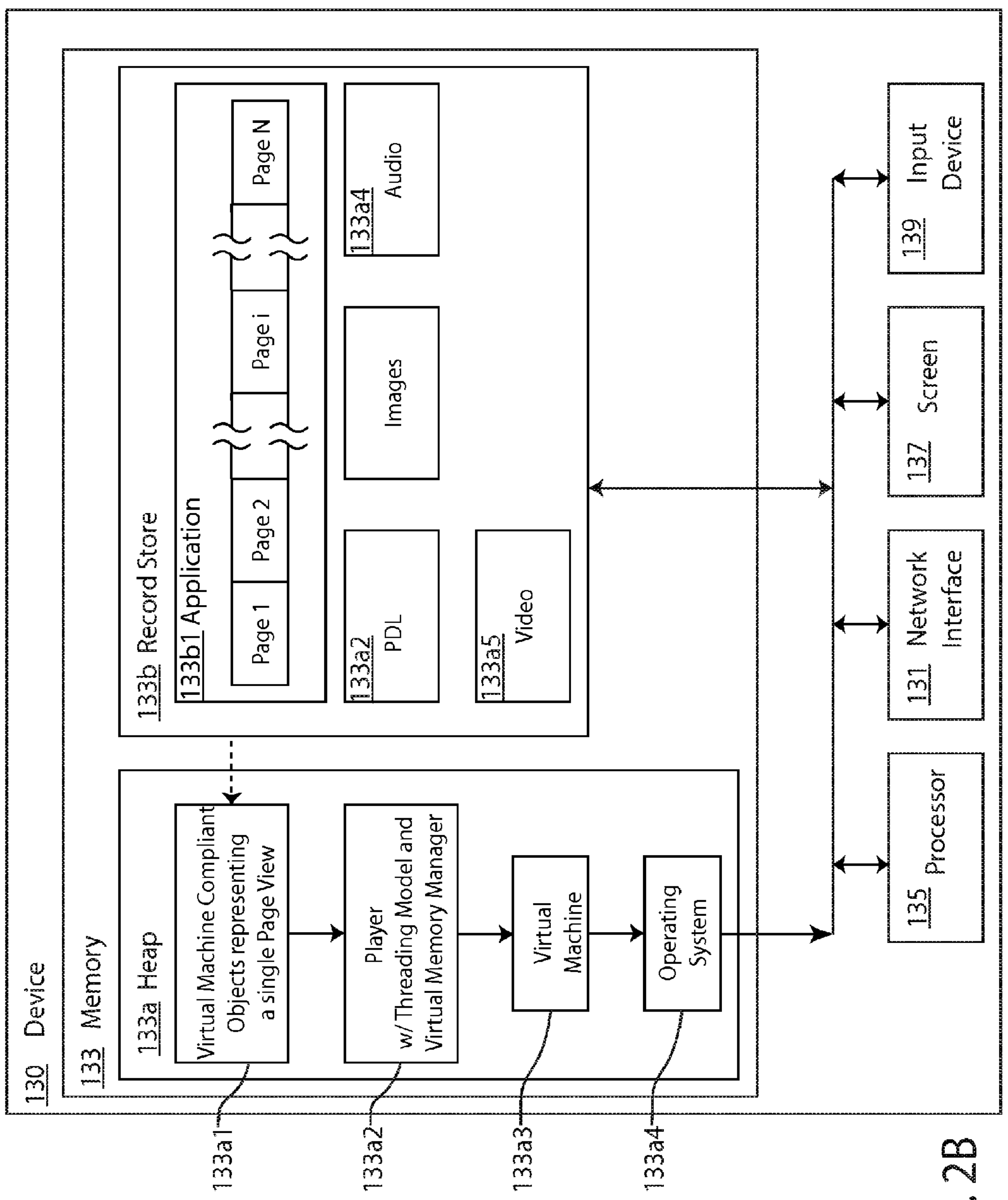

130 Device
133 Memory
133a Heap
Virtual Machine Compliant Objects representing a single Page View
Player w/ Threading Model and Virtual Memory Manager
Virtual Machine
Operating System
133b Record Store
133b1 Application
Page 1
Page 2
Page i
Page N
133a2 PDL
Images
133a4 Audio
133a5 Video
135 Processor
131 Network Interface
137 Screen
139 Input Device
133a1
133a2
133a3
133a4

FIG. 2B

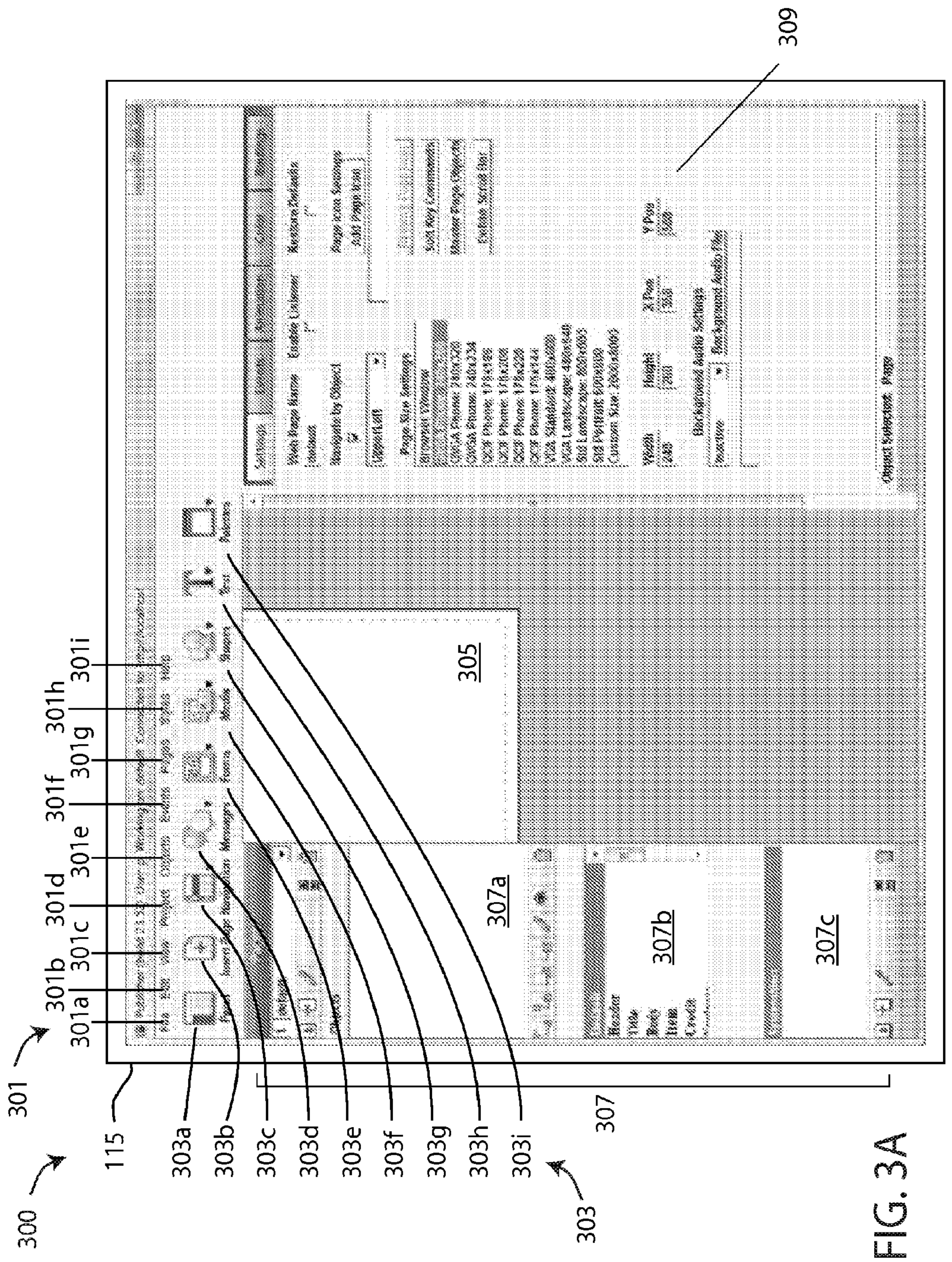

300
301
301a
301b
301c
301d
301e
301f
301g
301h
301i
115
303a
303b
303c
303d
303e
303f
303g
303h
303i
303
305
307
307a
307b
307c
309

FIG. 3A

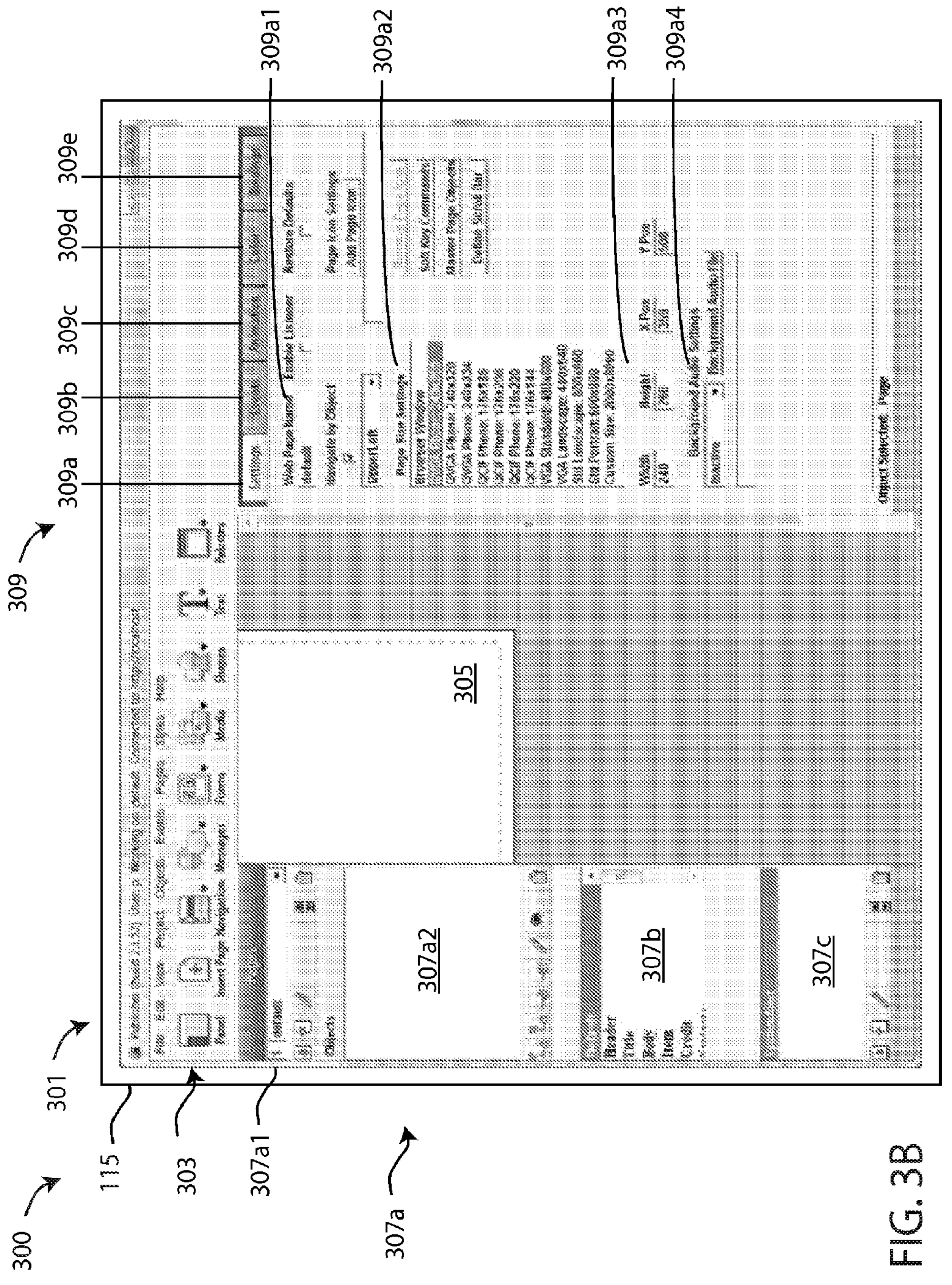
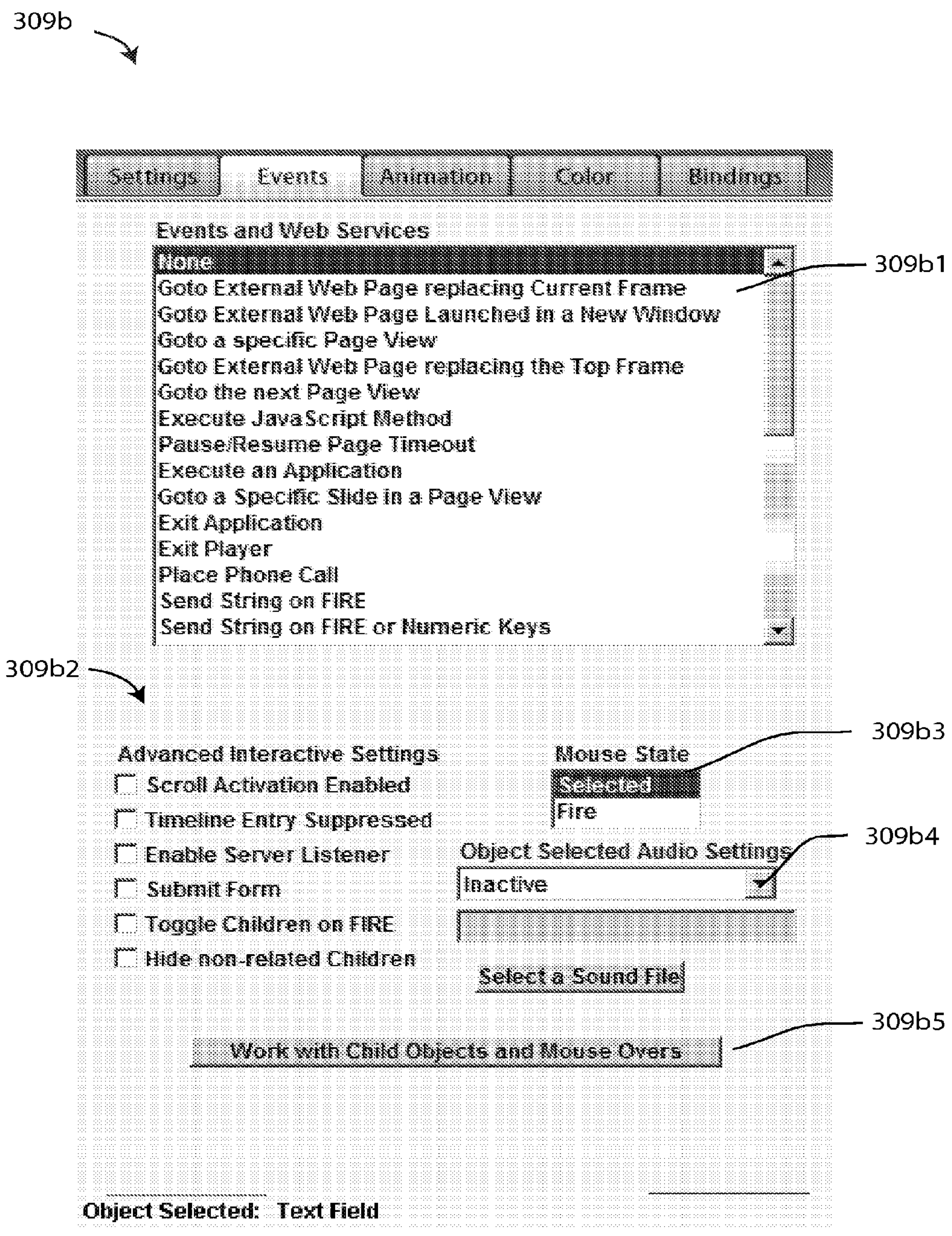

300
301
115
303
307a1
307a
307a2
307b
307c
305
309
309a
309b
309c
309d
309e
309a1
309a2
309a3
309a4
Settings
Page Size Settings
Background Audio Settings
Object Selected: Page

FIG. 3B



Events and Web Services

- None
- Goto External Web Page replacing Current Frame
- Goto External Web Page Launched in a New Window
- Goto a specific Page View
- Goto External Web Page replacing the Top Frame
- Goto the next Page View
- Execute JavaScript Method
- Pause/Resume Page Timeout
- Execute an Application
- Goto a Specific Slide in a Page View
- Exit Application
- Exit Player
- Place Phone Call
- Send String on FIRE
- Send String on FIRE or Numeric Keys

Advanced Interactive Settings

- ☐ Scroll Activation Enabled
- ☐ Timeline Entry Suppressed
- ☐ Enable Server Listener
- ☐ Submit Form
- ☐ Toggle Children on FIRE
- ☐ Hide non-related Children

Mouse State

- Selected
- Fire

Object Selected Audio Settings

Inactive

Object Selected: Text Field

FIG. 3C

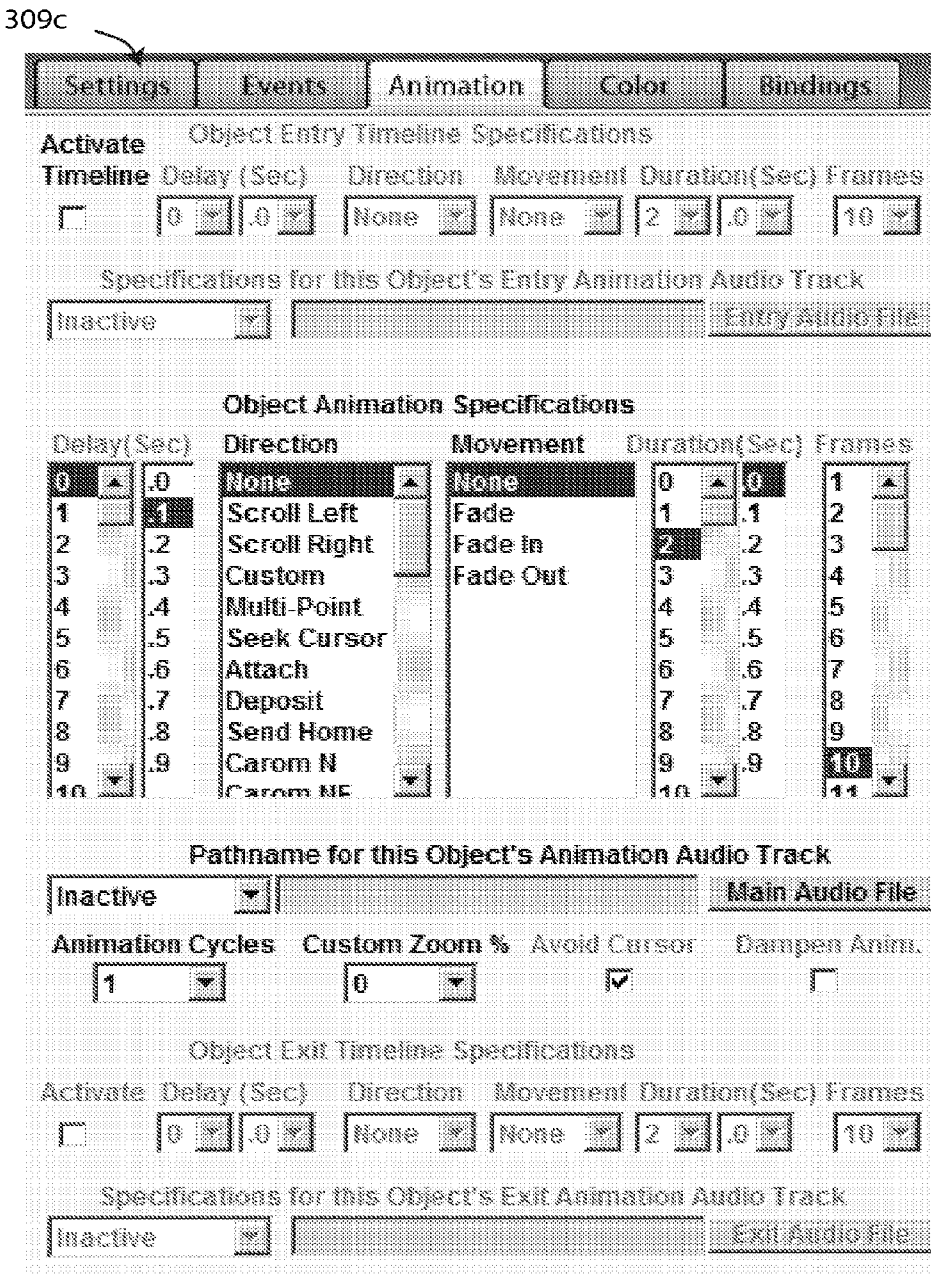

309c
Settings
Events
Animation
Color
Bindings
Object Entry Timeline Specifications
Activate Timeline
Delay (Sec)
Direction
Movement
Duration(Sec)
Frames
Specifications for this Object's Entry Animation Audio Track
Inactive
Entry Audio File
Object Animation Specifications
Delay(Sec)
Direction
Movement
Duration(Sec)
Frames
None
Scroll Left
Scroll Right
Custom
Multi-Point
Seek Cursor
Attach
Deposit
Send Home
Carom N
Carom NE
Fade
Fade In
Fade Out
Pathname for this Object's Animation Audio Track
Inactive
Main Audio File
Animation Cycles
Custom Zoom %
Avoid Cursor
Dampen Anim.
Object Exit Timeline Specifications
Activate
Delay (Sec)
Direction
Movement
Duration(Sec)
Frames
Specifications for this Object's Exit Animation Audio Track
Inactive
Exit Audio File

FIG. 3D

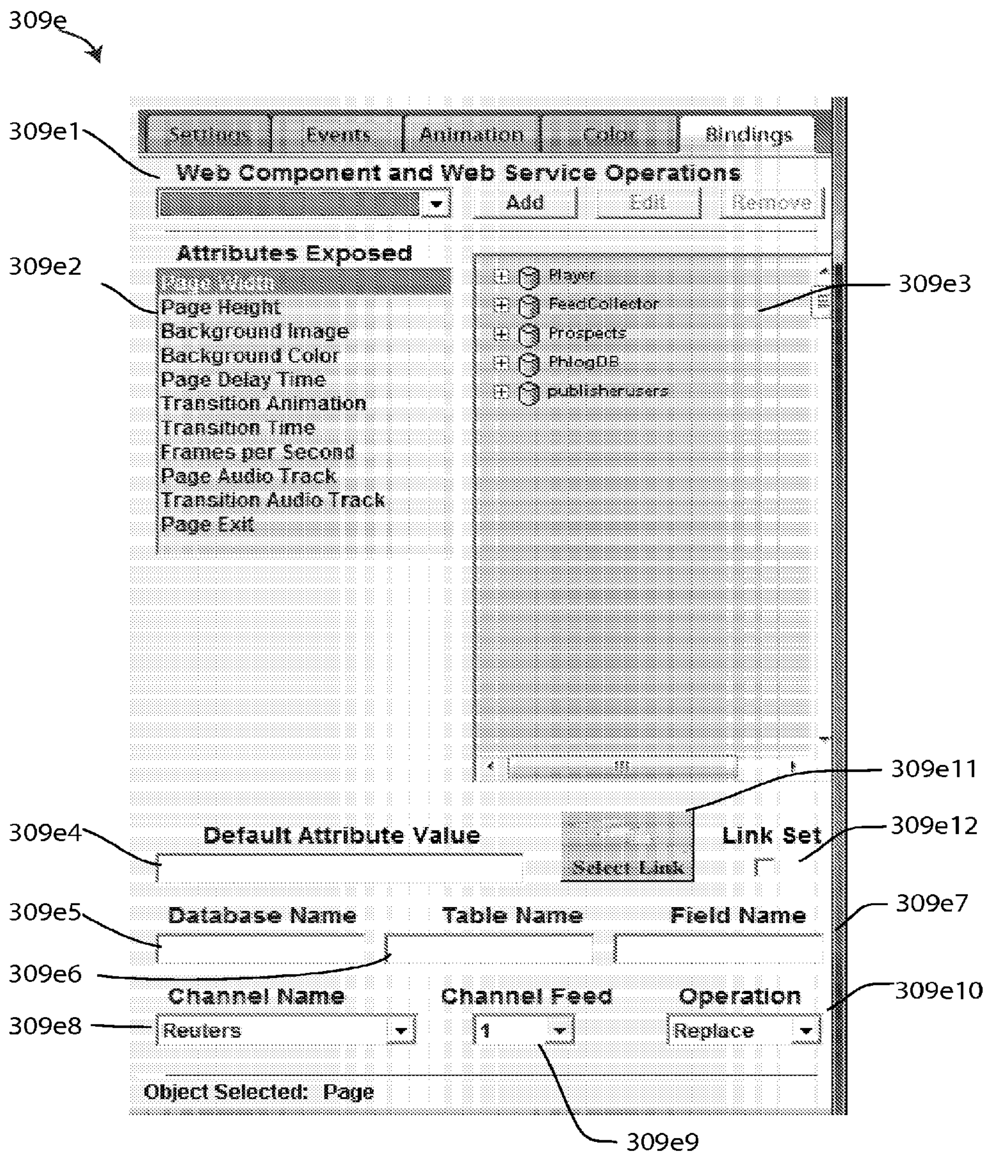
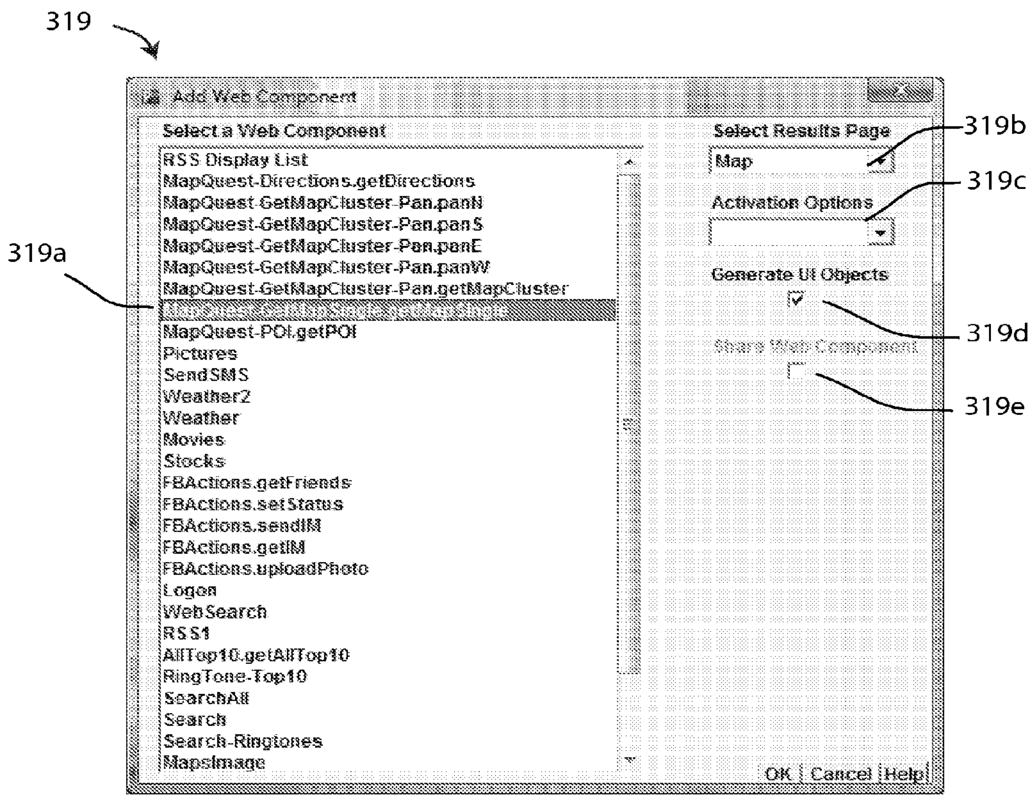

309e
309e1
309e2
309e3
309e4
309e5
309e6
309e7
309e8
309e9
309e10
309e11
309e12
Settings
Events
Animation
Color
Bindings
Web Component and Web Service Operations
Add
Edit
Remove
Attributes Exposed
Page Width
Page Height
Background Image
Background Color
Page Delay Time
Transition Animation
Transition Time
Frames per Second
Page Audio Track
Transition Audio Track
Page Exit
Player
FeedCollector
Prospects
PhlogDB
publisherusers
Default Attribute Value
Select Link
Link Set
Database Name
Table Name
Field Name
Channel Name
Channel Feed
Operation
Reuters
1
Replace
Object Selected: Page

FIG. 3E



Select a Web Component

RSS Display List
MapQuest-Directions.getDirections
MapQuest-GetMapCluster-Pan.panN
MapQuest-GetMapCluster-Pan.panS
MapQuest-GetMapCluster-Pan.panE
MapQuest-GetMapCluster-Pan.panW
MapQuest-GetMapCluster-Pan.getMapCluster
MapQuest-GetMapSingle.getMapSingle
MapQuest-POI.getPOI
Pictures
SendSMS
Weather2
Weather
Movies
Stocks
FBActions.getFriends
FBActions.setStatus
FBActions.sendIM
FBActions.getIM
FBActions.uploadPhoto
Logon
WebSearch
RSS1
AllTop10.getAllTop10
RingTone-Top10
SearchAll
Search
Search-Ringtones
MapsImage

Select Results Page: Map

Activation Options:

Generate UI Objects: ✓

Share Web Component: ☐

FIG. 3F

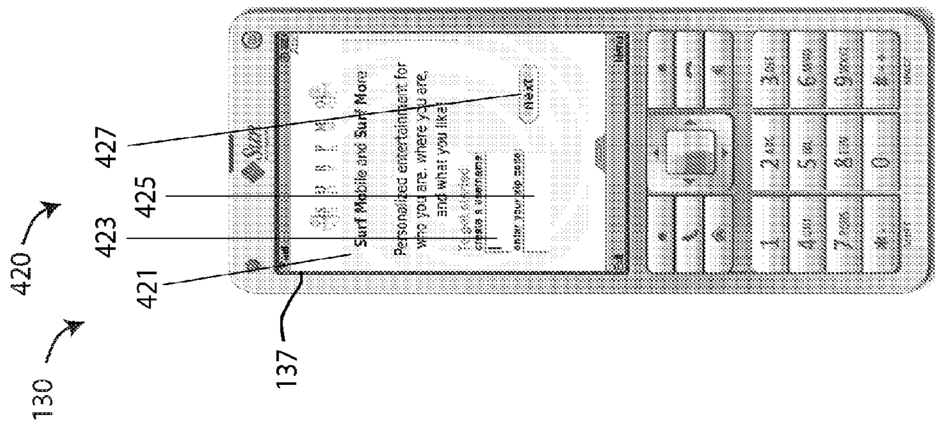

130
420
421
423
425
427
137
Surf Mobile and Surf More
Personalized entertainment for who you are, where you are, and what you like!
create a username:
enter your zip code
Next

FIG. 4B

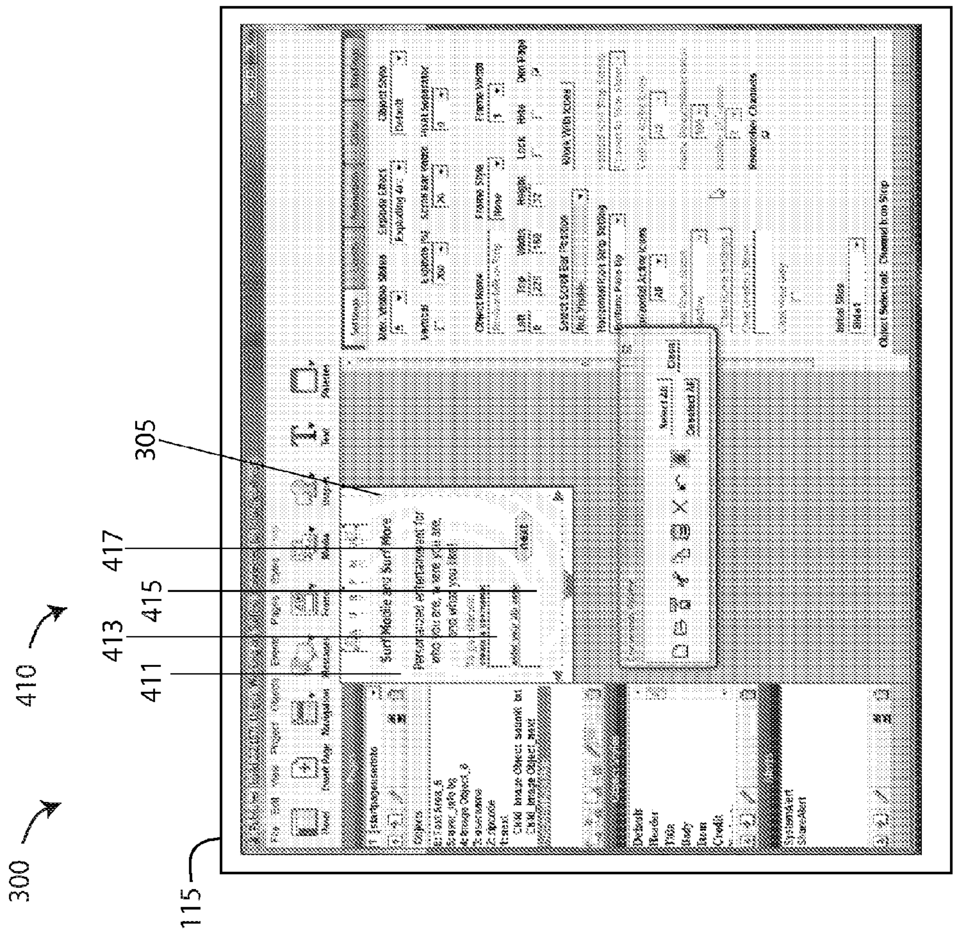

300
410
411
413
415
417
305
115
Surf Mobile and Surf More
Personalized entertainment for who you are, where you are, and what you like!
create a username:
enter your zip code
Next

FIG. 4A

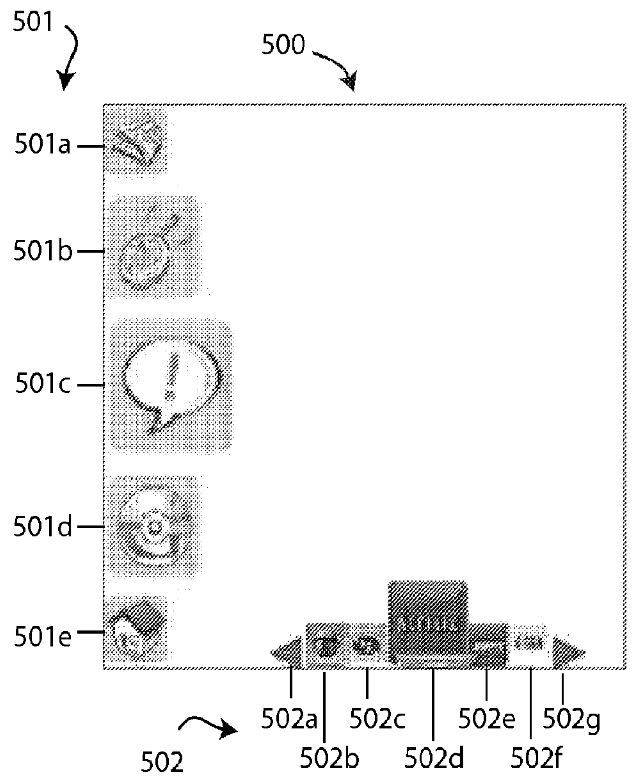
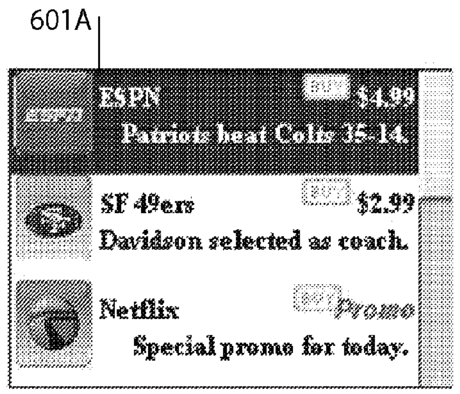
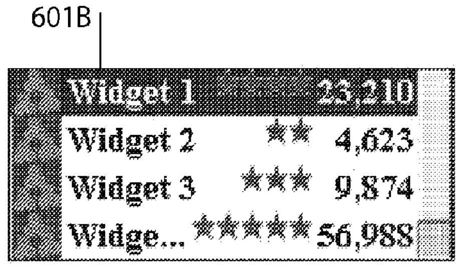


FIG. 5



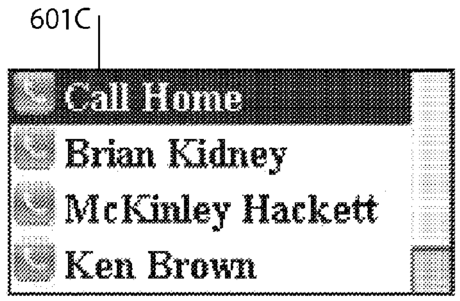
ESPN $4.99
Patriots beat Colts 35-14.
SF 49ers $2.99
Davidson selected as coach.
Netflix Promo
Special promo for today.

FIG. 6A



Widget 1		23,210
Widget 2	★★	4,623
Widget 3	★★★	9,874
Widge...	★★★★★	56,988

FIG. 6B



Call Home
Brian Kidney
McKinley Hackett
Ken Brown

FIG. 6C

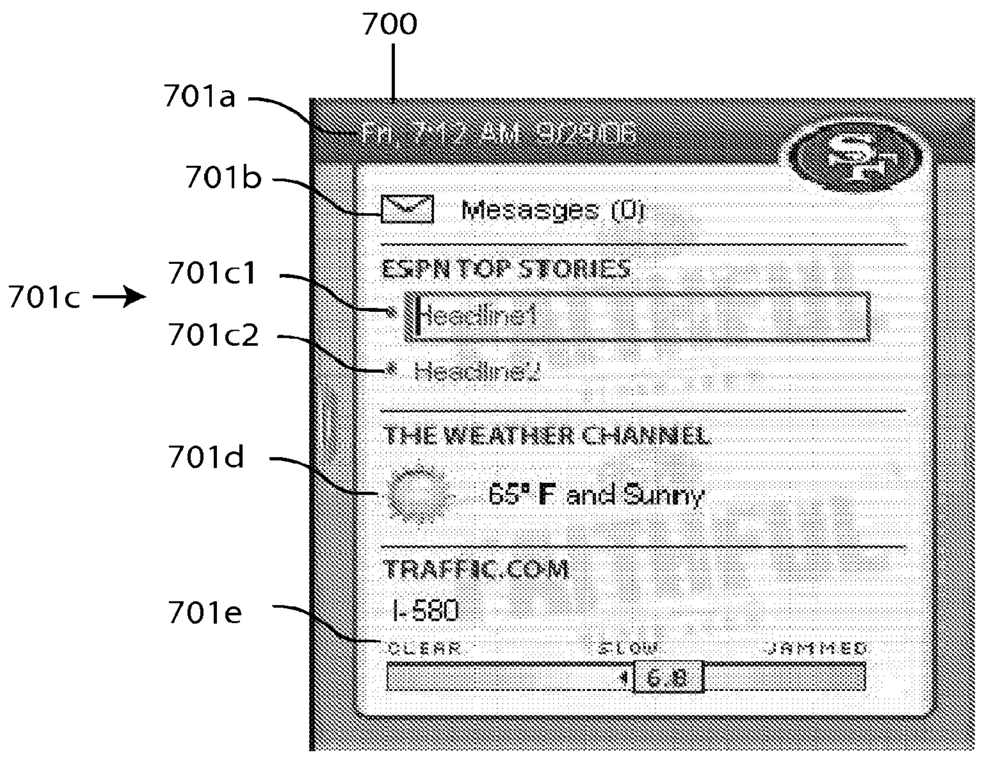

700
701a
701b
701c
701c1
701c2
701d
701e
Fri, 7:12 AM 9/29/06
Mesasges (0)
ESPN TOP STORIES
Headline1
Headline2
THE WEATHER CHANNEL
65° F and Sunny
TRAFFIC.COM
I-580
CLEAR
SLOW
JAMMED
6.8

FIG. 7

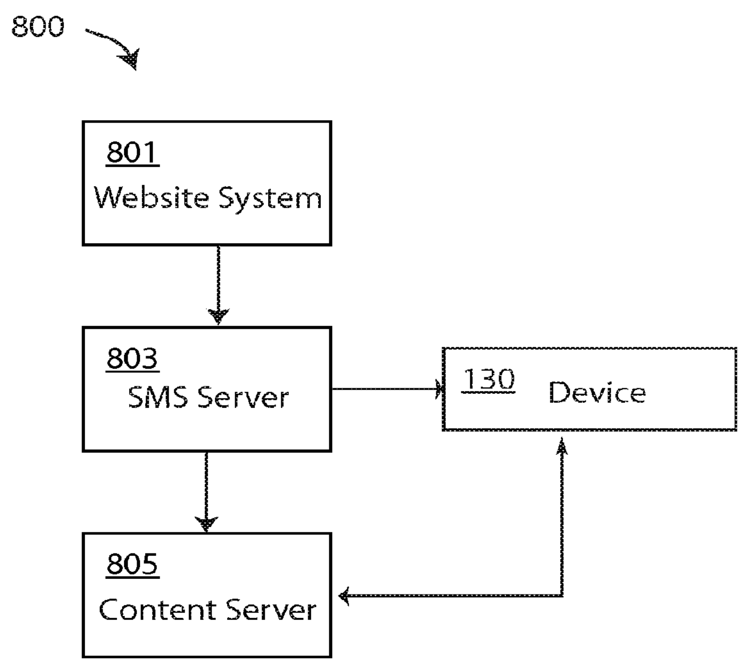

800
801 Website System
803 SMS Server
130 Device
805 Content Server

FIG. 8

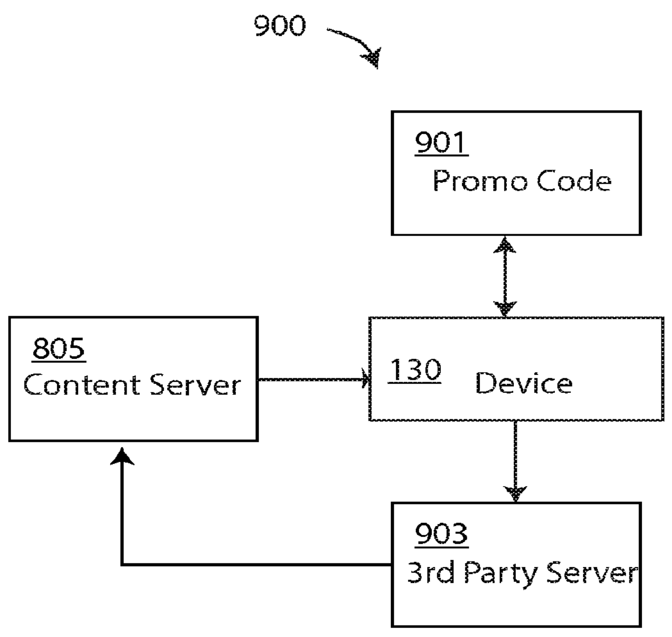

900
901 Promo Code
805 Content Server
130 Device
903 3rd Party Server

FIG. 9

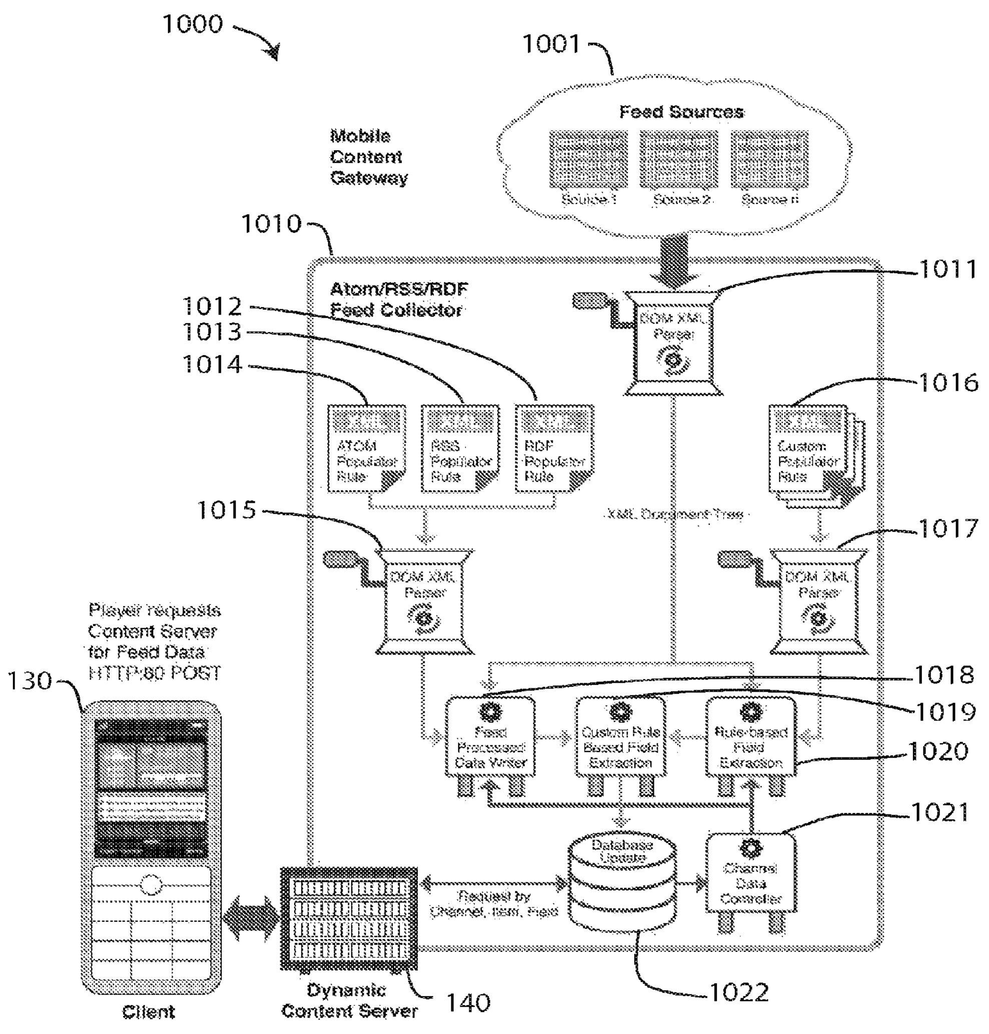

1000
1001
Feed Sources
Source 1
Source 2
Source n
Mobile Content Gateway
1010
Atom/RSS/RDF Feed Collector
1011
DOM XML Parser
1012
1013
1014
ATOM Populator Rule
RSS Populator Rule
RDF Populator Rule
1016
Custom Populator Rule
XML Document Tree
1015
DOM XML Parser
1017
DOM XML Parser
Player requests Content Server for Feed Data HTTP:80 POST
130
1018
1019
1020
Feed Processed Data Writer
Custom Rule Based Field Extraction
Rule-based Field Extraction
1021
Database Update
Channel Data Controller
Request by Channel, Item, Field
Client
Dynamic Content Server
140
1022

FIG. 10

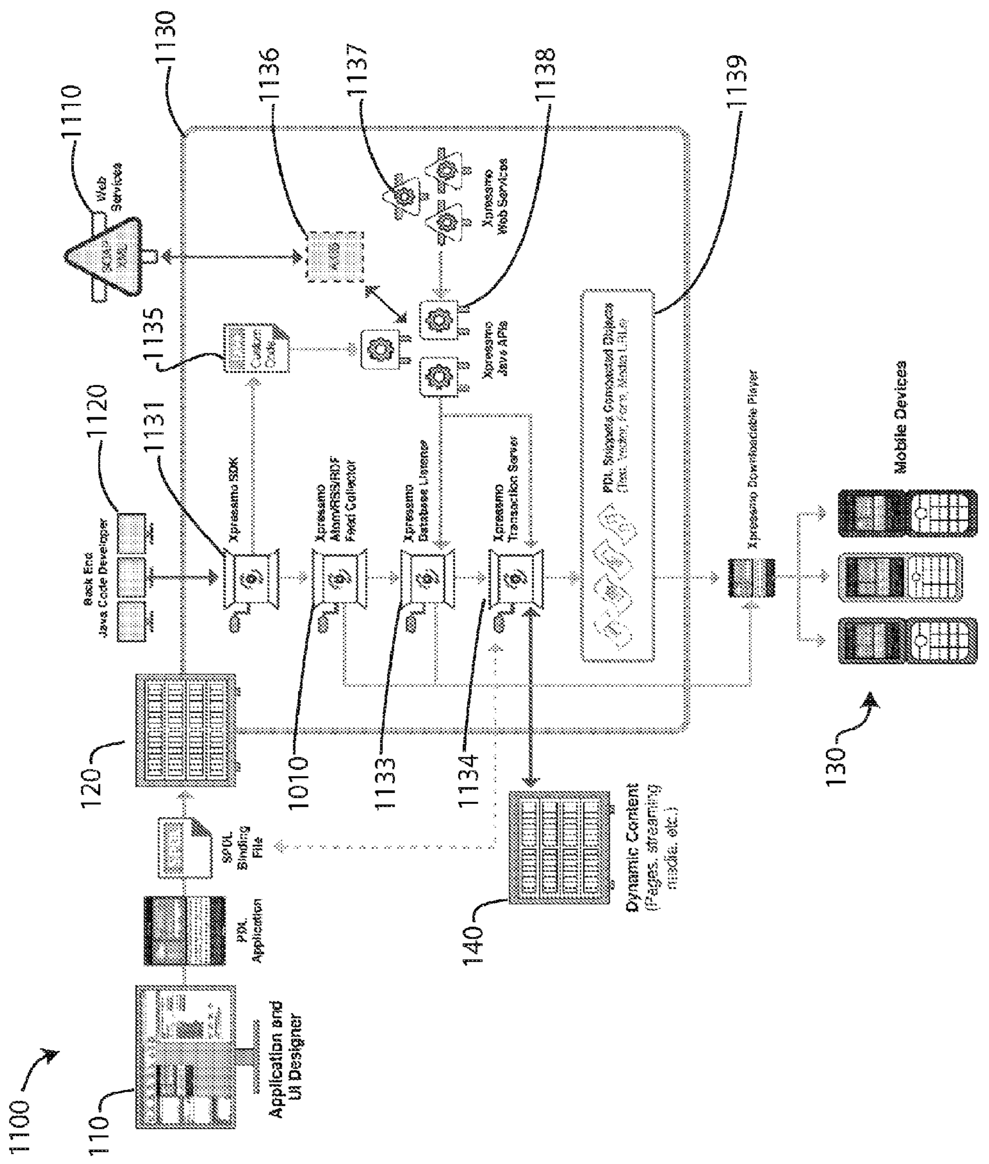

1100
110
Application and UI Designer
PDL Application
SPDL Binding File
120
1110
Web Services
SOAP XML
1120
Back End Java Code Developer
1130
1131
Xpressmo SDK
1135
Custom Code
1136
1137
Xpressmo Web Services
1138
Xpressmo Java APIs
1010
Xpressmo Atom/RSS/RDF Feed Collector
1133
Xpressmo Database Listener
1134
Xpressmo Transaction Server
1139
PDL Snippets Composited Objects (Text, Vector, Form, Media Files)
Xpressmo Downloadable Player
140
Dynamic Content (Pages, streaming media, etc.)
130
Mobile Devices

FIG. 11

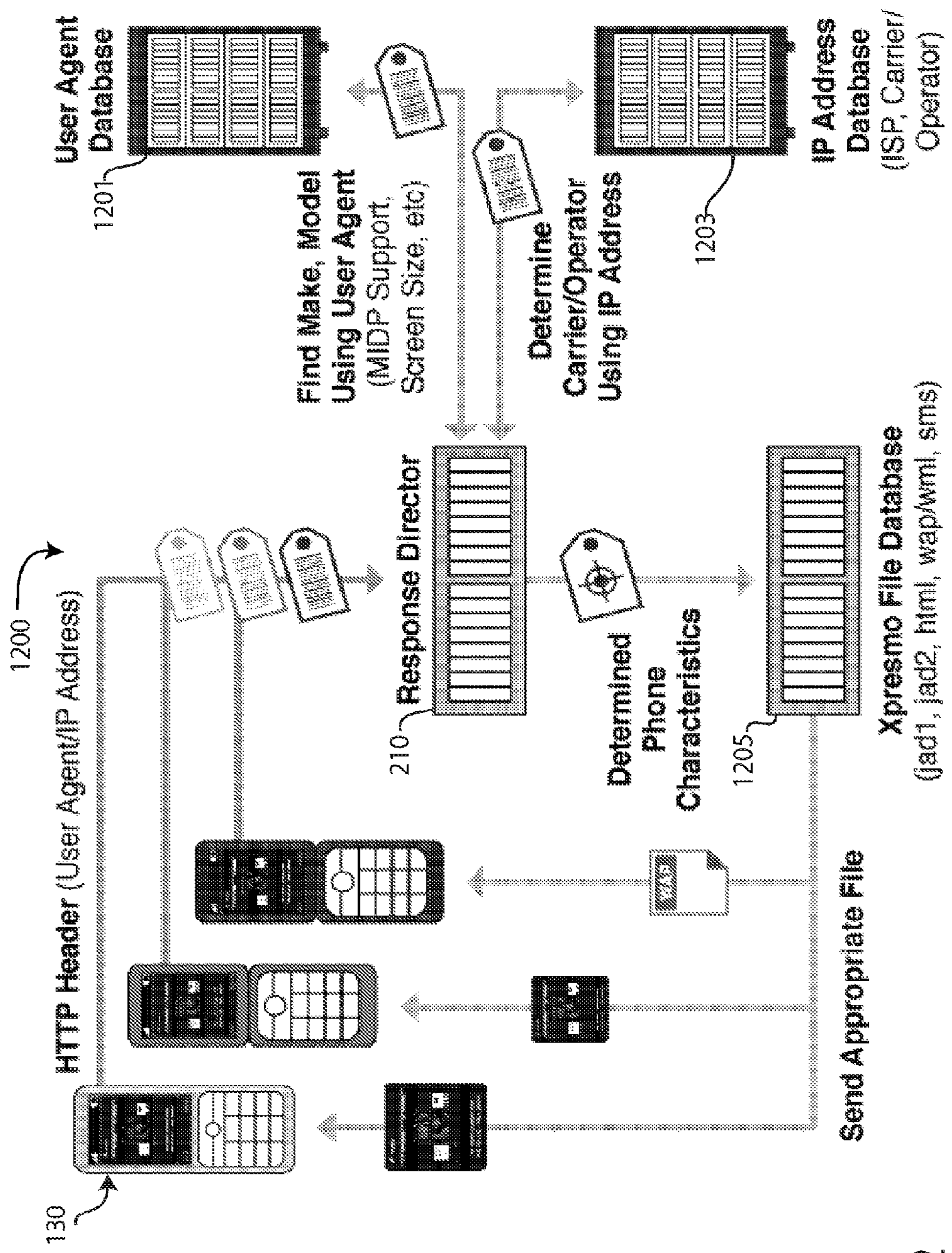

1200
HTTP Header (User Agent/IP Address)
130
Response Director
210
Find Make, Model Using User Agent (MIDP Support, Screen Size, etc)
User Agent Database
1201
Determine Carrier/Operator Using IP Address
IP Address Database (ISP, Carrier/ Operator)
1203
Determined Phone Characteristics
Xpresmo File Database (jad1, jad2, html, wap/wml, sms)
1205
Send Appropriate File

FIG. 12

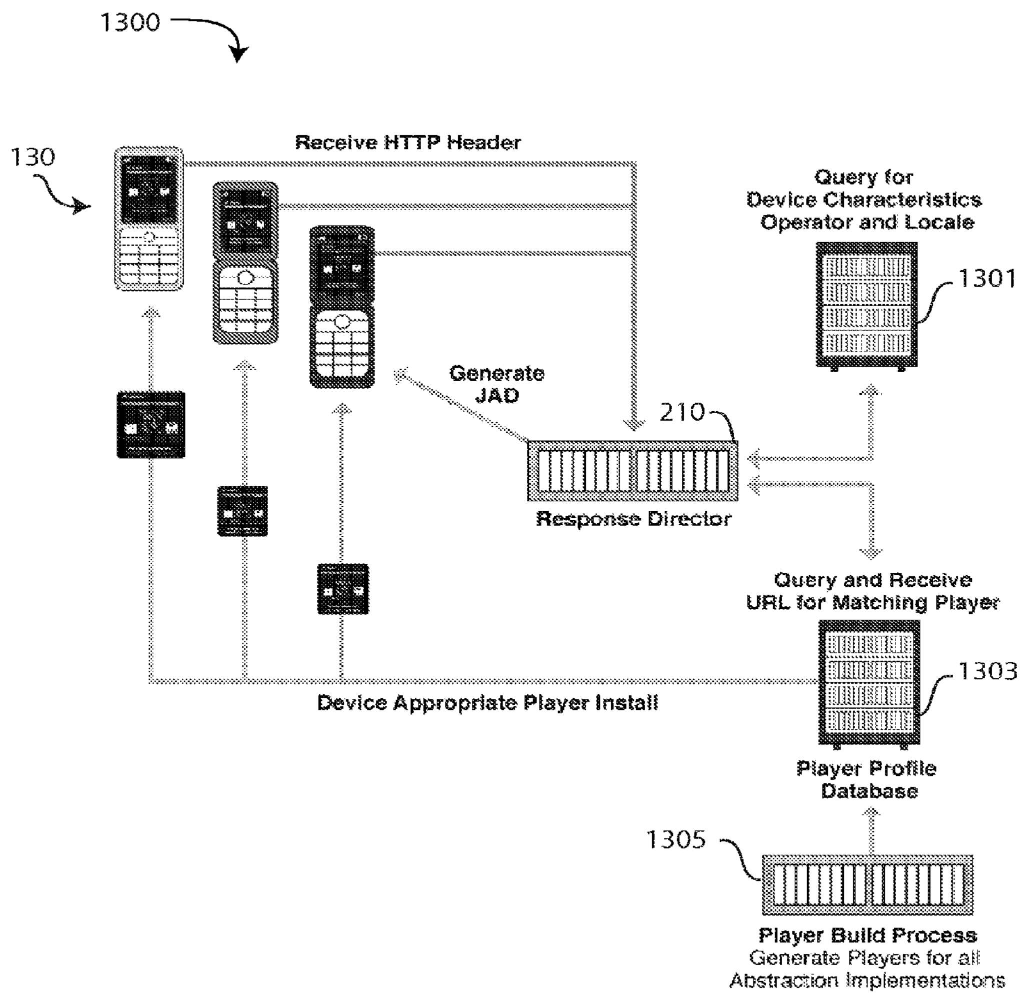

1300
130
Receive HTTP Header
Query for
Device Characteristics
Operator and Locale
1301
Generate
JAD
210
Response Director
Query and Receive
URL for Matching Player
1303
Device Appropriate Player Install
Player Profile
Database
1305
Player Build Process
Generate Players for all
Abstraction Implementations

FIG. 13

SYSTEMS AND METHODS FOR PROGRAMMING MOBILE DEVICES

TECHNICAL FIELD

The present invention generally relates to providing software for mobile devices, and more particularly to a method and system for authoring Applications for devices.

BACKGROUND ART

Internet-connected mobile devices are becoming ever more popular. While these devices provide portability to the Internet, they generally do not have the capabilities of non-mobile devices including computing, input and output capabilities.

In addition, the mobility of the user while using such devices provides challenges and opportunities for the use of the Internet. Further, unlike non-mobile devices, there are a large number of types of devices and they tend to have a shorter lifetime in the marketplace. The programming of the myriad of mobile devices is a time-consuming and expensive proposition, thus limiting the ability of service providers to update the capabilities of mobile devices.

Thus there is a need in the art for a method and apparatus that permits for the efficient programming of mobile devices. Such a method and apparatus should be easy to use and provide output for a variety of devices.

DISCLOSURE OF INVENTION

In certain embodiments, a system is provided to generate code to provide content on a display of a platform. The system includes a database of web services obtainable over a network and an authoring tool. The authoring tool is configured to define an object for presentation on the display, select a component of a web service included in said database, associate said object with said selected component, and produce code that, when executed on the platform, provides said selected component on the display of the platform.

In certain other embodiments, a method is provided for providing information to platforms on a network. The method includes accepting a first code over the network, where said first code is platform-dependent; providing a second code over the network, where said second code is platform-independent; and executing said first code and said second code on the platform to provide web components obtained over the network.

In certain embodiments, a method for displaying content on a platform utilizing a database of web services obtainable over a network is provided. The method includes: defining an object for presentation on the display; selecting a component of a web service included in said database; associating said object with said selected component; and producing code that, when executed on the platform, provides said selected component on the display of the platform.

In one embodiment, one of the codes is a Player, which is a thin client architecture that operates in a language that manages resources efficiently, is extensible, supports a robust application model, and has no device specific dependencies. In another embodiment, Player P is light weight and extends the operating system and/or virtual machine of the device to: Manage all applications and application upgrades, and resolve device, operating system, VM and language fragmentation.

In another embodiment, one of the codes is an Application that is a device independent code that interpreted by the Player.

These features together with the various ancillary provisions and features which will become apparent to those skilled in the art from the following detailed description, are attained by the system and method of the present invention, preferred embodiments thereof being shown with reference to the accompanying drawings, by way of example only, wherein:

BRIEF DESCRIPTION OF DRAWINGS

FIG. **1**A is an illustrative schematic of one embodiment of a system including an authoring platform and a server for providing programming instructions to a device over a network;

FIG. **1**B is schematic of an alternative embodiment system for providing programming instructions to device over a network;

FIG. **2**A is a schematic of an embodiment of system illustrating the communications between different system components;

FIG. **2**B is a schematic of one embodiment of a device illustrating an embodiment of the programming generated by authoring platform;

FIGS. **3**A and **3**B illustrate one embodiment of a publisher interface as it appears, for example and without limitation, on a screen while executing an authoring tool;

FIG. **3**C illustrates an embodiment of the Events Tab,

FIG. **3**D illustrates one embodiment of an Animation Tab;

FIG. **3**E illustrates one embodiment of Bindings Tab;

FIG. **3**F illustrates one embodiment of a pop-up menu for adding web components;

FIG. **4**A shows a publisher interface having a layout on a canvas; and FIG. **4**B shows a device having the resulting layout on a device screen;

FIG. **5** shows a display of launch strips;

FIG. **6**A is a display of a Channel Selection List;

FIG. **6**B is a display of a Widget Selection List;

FIG. **6**C is a display of a Phone List;

FIG. **7** shows a display of a mash-up;

FIG. **8** is a schematic of an embodiment of a push capable system;

FIG. **9** is a schematic of an alternative embodiment of a push capable system;

FIG. **10** is a schematic of one embodiment of a feed collector;

FIG. **11** is a schematic of an embodiment of a Mobile Content Gateway;

FIG. **12** is a schematic of one embodiment of a system that includes a response director, a user agent database, an IP address database, and a file database; and

FIG. **13** is a schematic of another embodiment of a system that includes a response director, a user agent database, an IP address database, and a file database.

Reference symbols are used in the Figures to indicate certain components, aspects or features shown therein, with reference symbols common to more than one Figure indicating like components, aspects or features shown therein.

MODE(S) FOR CARRYING OUT THE INVENTION

FIG. **1**A is an illustrative schematic of one embodiment of a system **100** including an authoring platform **110** and a server **120** for providing programming instructions to a

device **130** over a network N. In one embodiment, device **130** is a wireless device, and network N includes wireless communication to the device. Alternatively, system **100** may provide access over network N to other information, data, or content, such as obtainable as a web service over the Internet. In general, a user of authoring platform **110** may produce programming instructions or files that may be transmitted over network N to operate device **130**, including instructions or files that are sent to device **130** and/or server **120**. The result of the authoring process is also referred to herein, and without limitation, as publishing an Application.

Embodiments include one or more databases that store information related to one or more devices **130** and/or the content provided to the devices. It is understood that such databases may reside on any computer or computer system on network N, and that, in particular, the location is not limited to any particular server, for example.

Device **130** may be, for example and without limitation, a cellular telephone or a portable digital assistant, includes a network interface **131**, a memory **133**, a processor **135**, a screen **137**, and an input device **139**. Network interface **131** is used by device **130** to communication over a wireless network, such as a cellular telephone network, a WiFi network or a WiMax network, and then to other telephones through a public switched telephone network (PSTN) or to a satellite, or over the Internet. Memory **133** includes programming required to operate device **130** (such as an operating system or virtual machine instructions), and may include portions that store information or programming instructions obtained over network interface **131**, or that are input by the user (such as telephone numbers or images from a device camera (not shown). In one embodiment screen **137** is a touch screen, providing the functions of the screen and input device **139**.

Authoring platform **110** includes a computer or computer system having a memory **111**, a processor **113**, a screen **115**, and an input device **117**. It is to be understood that memory **111**, processor **113**, screen **115**, and input device **117** are configured such a program stored in the memory may be executed by the processor to accept input from the input device and display information on the screen. Further, the program stored in memory **111** may also instruct authoring platform **110** to provide programming or information, as indicated by the line labeled "A" and to receive information, as indicated by the line labeled "B."

Memory **111** is shown schematically as including a stored program referred to herein, and without limitation, as an authoring tool **112**. In one embodiment, authoring tool **112** is a graphical system for designing the layout of features as a display that is to appear on screen **137**. One example of authoring tool **112** is the CDER™ publishing platform (Express Mobile, Inc., Novato, Calif.).

In another embodiment, which is not meant to limit the scope of the present invention, device **130** may include an operating system having a platform that can interpret certain routines. Memory **111** may optionally include programming referred to herein, and without limitation, as routines **114** that are executable on device **130**.

Routines **114** may include device-specific routines—that is, codes that are specific to the operating system, programming language, or platform of specific devices **130**, and may include, but are not limited to, Java, Windows Mobile, Brew, Symbian OS, or Open Handset Alliance (OHA). Several examples and embodiments herein are described with reference to the use of Java. It is to be understood that the invention is not so limited, except as provided in the claims, and that one skilled in the art could provide Players for devices using routines provided on a platform. Thus as an example, routines **114** may include Java API's and an authoring tool System Development Kit (SDK) for specific devices **130**.

Server **120** is a computer or computer system that includes a network interface **121**, a memory **123**. and a processor **125**. Is to be understood that network interface **121**, memory **123**, and processor **125** are configured such that a program stored in the memory may be executed by the processor to: accept input and/or provide output to authoring platform **110**; accept input and/or provide output through network interface **121** over network N to network interface **131**; or store information from authoring platform **110** or from device **130** for transmission to another device or system at a later time.

In one embodiment, authoring platform **110** permits a user to design desired displays for screen **137** and actions of device **130**. In other words, authoring platform **110** is used to program the operation of device **130**. In another embodiment, authoring platform **110** allows a user to provide input for the design of one or more device displays and may further allow the user to save the designs as device specific Applications. The Applications may be stored in memory **123** and may then be sent, when requested by device **130** or when the device is otherwise accessible, over network N, through network interface **130** for storage in memory **133**.

In an alternative embodiment, analytics information from devices **130** may be returned from device **130**, through network N and server **120**, back to authoring platform **110**, as indicated by line B, for later analysis. Analytics information includes, but is not limited to, user demographics, time of day, and location. The type of analytic content is only limited by which listeners have been activated for which objects and for which pages. Analytic content may include, but is not limited to, player-side page view, player-side forms-based content, player-side user interactions, and player-side object status.

Content server **140** is a computer or computer system that includes a network interface **141**, a memory **143**. and a processor **145**. It is to be understood that network interface **141**, memory **143**, and processor **145** are configured such that a stored program in the memory may be executed by the processor to accepts requests R from device **130** and provide content C over a network, such as web server content the Internet, to device **130**.

FIG. **1**B is schematic of an alternative embodiment system **100** for providing programming instructions to device **130** over a network N that is generally similar to the system of FIG. **1**A. The embodiment of FIG. **1**B illustrates that system **100** may include multiple servers **120** and/or multiple devices **130**.

In the embodiment of FIG. **1**B, system **100** is shown as including two or more servers **120**, shown illustratively and without limitation as servers **120***a* and **120***b*. Thus some of the programming or information between authoring platform **110** and one or more devices **130** may be stored, routed, updated, or controlled by more than one server **120**. In particular, the systems and methods described herein may be executed on one or more server **120**.

Also shown in FIG. **1**B are a plurality of devices **130**, shown illustratively and without limitation as device **130-1**, **130-1**, . . . **130**-N. System **100** may thus direct communication between individual server(s) **120** and specific device(s) **130**.

As described subsequently, individual devices **130** may be provided with program instructions which may be stored in each device's memory **133** and where the instructions are executed by each device's processor **135**. Thus, for example, server(s) **120** may provide device(s) **130** with programming in response to the input of the uses of the individual devices. Further, different devices **130** may be operable using different sets of instructions, that is having one of a variety of different "device platforms." Differing device platforms may result, for example and without limitation, to different operating systems, different versions of an operating system, or different versions of virtual machines on the same operating system. In some embodiments, devices **130** are provided with some programming from authoring system **100** that is particular to the device.

In one embodiment, system **100** provides permits a user of authoring platform **110** to provide instructions to each of the plurality of devices **130** in the form of a device- or device-platform specific instructions for processor **135** of the device, referred to herein and without limitation as a "Player," and a device-independent program, referred to herein and without limitation as an "Application" Thus, for example, authoring platform **110** may be used to generate programming for a plurality of devices **130** having one of several different device platforms. The programming is parsed into instructions used by different device platforms and instructions that are independent of device platform. Thus in one embodiment, device **130** utilizes a Player and an Application to execute programming from authoring platform **110**. A device having the correct Player is then able to interpret and be programmed according to the Application.

In one alternative embodiment, the Player is executed the first time by device **130** ("activated") through an Application directory. In another alternative embodiment, the Player is activated by a web browser or other software on device **130**. In yet another alternative embodiment, Player is activated through a signal to device **130** by a special telephone numbers, such as a short code.

When the Application and the Player are provided to memory **133**, the functioning of device **130** may occur in accordance with the desired programming Thus in one embodiment, the Application and Player includes programming instructions which may be stored in memory **133** and which, when executed by processor **135**, generate the designed displays on screen **137**. The Application and Player may also include programming instructions which may be stored in memory **133** and which provide instructions to processor **135** to accept input from input device **139**.

Authoring tool **112** may, for example, produce and store within memory **111** a plurality of Players (for different devices **130**) and a plurality of Applications for displaying pages on all devices. The Players and Applications are then stored on one or more servers **120** and then provided to individual devices **130**. In general, Applications are provided to device **130** for each page of display or a some number of pages. A Player need be provided once or updated as necessary, and thus may be used to display a large number of Applications. This is advantageous for the authoring process, since all of the device-dependent programming is provided to a device only once (or possibly for some small number of upgrades), permitting a smaller Application, which is the same for each device **130**.

Thus, for example and without limitation, in one embodiment, the Player transforms device-independent instructions of the Application into device-specific instructions that are executable by device **130**. Thus, by way of example and without limitation, the Application may include Java programming for generating a display on screen **137**, and the Player may interpret the Java and instruct processor **135** to produce the display according to the Application for execution on a specific device **130** according to the device platform. The Application may in general include, without limitation, instructions for generating a display on screen **137**, instructions for accepting input from input device **139**, instructions for interacting with a user of device **130**, and/or instructions for otherwise operating the device, such as to place a telephone call.

The Application is preferably code in a device-independent format, referred to herein and without limitation as a Portable Description Language (PDL). The device's Player interprets or executes the Application to generate one or more "pages" ("Applications Pages") on a display as defined by the PDL. The Player may include code that is device-specific—that it, each device is provided with a Player that is used in the interpretation and execution of Applications. Authoring tool **112** may thus be used to design one or more device-independent Applications and may also include information on one or more different devices **130** that can be used to generate a Player that specific devices may use to generate displays from the Application.

In one embodiment, system **100** provides Players and Applications to one server **120**, as in FIG. **1**A. In another embodiment, system **100** provides Players to a first server **120***a* and Applications to a second server **120***b*, as in FIG. 1B.

In one embodiment, authoring tool **112** may be used to program a plurality of different devices **130**, and routines **114** may include device-specific routines. In another embodiment, the Player is of the type that is commonly referred to as a "thin client"—that is, software for running on the device as a client in client-server architecture with a device network which depends primarily on a central server for processing activities, and mainly focuses on conveying input and output between the user and the server.

In one embodiment, authoring platform **110** allows user to arrange objects for display on screen. A graphical user interface ("GUI," or "UI") is particularly well suited to arranging objects, but is not necessary. The objects may correspond to one or more of an input object, an output object, an action object, or may be a decorative display, such as a logo, or background color or pattern, such as a solid or gradient fill. In another embodiment, authoring platform **110** also permits a user to assign actions to one or more of an input object, an output object, or an action object. In yet another embodiment, authoring platform **110** also permits a user to bind one or more of an input object, an output object, or an action object with web services or web components, or permits a user to provide instructions to processor **135** to store or modify information in memory **133**, to navigate to another display or service, or to perform other actions, such as dialing a telephone number.

In certain embodiments, the applicant model used in developing and providing Applications is a PDL. The PDL can be conceptually viewed as a device, operating system and virtual machine agnostic representation of Java serialized objects. In certain embodiments, the PDL is the common language for authoring tool **112**, the Application, and Player. Thus while either designing the Application with the authoring tool **112**, or programming with the SDK, the internal representation of the programming logic is in Java. In one embodiment the SDK is used within a multi-language software development platform comprising an IDE and a plug-in system to extend it, such as the Eclipse Integrated Development Environment (see, for example, http://www.eclipse.org/). At publish time the Java code is translated into a PDL. This translation may also occur in real-time during the execution of any Web Services or backend business logic that interacts with the user.

One embodiment for compacting data that may be used is described in co-pending U.S. Pat. No. 6,546,397 to Rempell ("Rempell"), the contents of which are incorporated herein by reference. In that patent the compressed data is described as being a database. The terminology used here is a PDL, that is the "internal database" of Rempell is equivalent to the PDL of the present Application.

The use of a PDL, as described in Rempell, permits for efficient code and data compaction. Code, as well as vector, integer and Boolean data may be compacted and then compressed resulting in a size reduction of 40 to 80 times that of the original Java serialized objects. This is important not only for performance over the network but for utilizing the virtual memory manager of the Player more efficiently. As an example, the reassembled primitives of the Java objects may first undergo logical compression, followed by LZ encoding.

The use of a PDL also provides virtual machine and operating system independence. Since the reassembled primitives of the Application no longer have any dependencies from the original programming language (Java) that they were defined in. The PDL architecture takes full advantage of this by abstracting all the virtual machine and/or operating system interfaces from the code that processes the PDL.

In one embodiment, the PDL is defined by the means of nested arrays of primitives. Accordingly, the use of a PDL provides extensibility and compatibility, with a minimal amount of constraints in extending the Player seamlessly as market demands and device capabilities continue to grow. Compatibility with other languages is inherent based on the various Player abstraction implementations, which may be, for example and without limitation, Java CDC, J2SE or MIDP2 implementations.

In one embodiment, the architecture of Player P includes an abstraction interface that separates all device, operating system and virtual machine dependencies from the Player's Application model business logic (that is, the logic of the server-side facilities) that extend the Application on the Player so that it is efficiently integrated into a comprehensive client/server Application. The use of an abstraction interface permits the more efficient porting to other operating systems and virtual machines and adding of extensions to the Application model so that a PDL can be implemented once and then seamlessly propagated across all platform implementations. The Application model includes all the currently supported UI objects and their attributes and well as all of the various events that are supported in the default Player. Further, less robust platforms can be augmented by extending higher end capabilities inside that platform's abstraction interface implementation.

In one embodiment, authoring platform **110** provides one or more pages, which may be provided in one Application, or a plurality of Applications, which are stored in memory **123** and subsequently provided to memory **133**. In certain embodiments, the Application includes instructions R to request content or web services C from content server **140**. Thus, for example and without limitation, the request is for information over the network via a web service, and the request R is responded to with the appropriate information for display on device **130**. Thus, for example, a user may request a news report. The Application may include the layout of the display, including a space for the news, which is downloaded form content server **140** for inclusion on the display. Other information that may be provided by content server **140** may include, but is not limited to, pages, Applications, multimedia, and audio.

FIG. **2**A is a schematic of a system **200** of an embodiment of system **100** illustrating the communications between different system components. System includes a response director **210**, a web component registry **220**, and a web service **230**. System **200** further includes authoring platform **110**, server **120**, device **130** and content server **140** are which are generally similar to those of the embodiments of FIGS. **1**A and **1**B, except as explicitly noted.

Response director **210** is a computer or computer system that may be generally similar to server **120** including the ability to communicate with authoring platform **110** and one or more devices **130**. In particular, authoring platform **110** generates one or more Players (each usable by certain devices **130**) which are provided to response director **210**. Devices **130** may be operated to provide response director **210** with a request for a Player and to receive and install the Player. In one embodiment, device **130** provides response director **210** with device-specific information including but not limited to make, model, and/or software version of the device. Response director **210** then determines the appropriate Player for the device, and provides the device with the Player over the network.

Web service **230** is a plurality of services obtainable over the Internet. Each web service is identified and/or defined as an entry in web component registry **230**, which is a database, XML file, or PDL that exists on a computer that may be a server previously described or another server **120**. Web component registry **230** is provided through server **120** to authoring platform **110** so that a user of the authoring platform may bind web services **230** to elements to be displayed on device **130**, as described subsequently.

In one embodiment, authoring platform **110** is used in conjunction with a display that provides a WYSIWYG environment in which a user of the authoring platform can produce an Application and Player that produces the same display and the desired programming on device **130**. Thus, for example, authoring tool **112** provides a display on screen **115** that corresponds to the finished page that will be displayed on screen **137** when an Application is intercepted, via a Player, on processor **135** of device **130**.

Authoring platform **110** further permits a user of the authoring platform to associate objects, such as objects for presenting on screen **137**, with components of one or more web services **230** that are registered in web component registry **220**. In one embodiment, information is provided in an XML file to web component registry **220** for each registered components of each web service **230**. Web component registry **220** may contain consumer inputs related to each web service **230**, environmental data such as PIM, time or location values, persistent variable data, outputs related to the web service, and/or optional hinting for improving the user's productivity.

A user of authoring platform **110** of system **200** may define associations with web services as WebComponent Bindings. In one embodiment, authoring platform **110** allows a user to associate certain objects for display that provide input or output to components of web service **230**. The associated bindings are saved as a PDL in server **120**.

In one embodiment, an XML web component registry **220** for each registered web service **230** is loaded into authoring platform **110**. The user of system **200** can then assign components of any web service **230** to an Application without any need to write code. In one embodiment, a component of web service **230** is selected from authoring platform **110** which presents the user with WYSIWYG dialog boxes that enable the binding of all the inputs and outputs of component of web service **230** to a GUI component of the Application as will be displayed on screen **137**. In addition, multiple components of one or more web service **230** can be assigned to any Object or Event in order to facilitate mashups. These Object and/or Event bindings, for each instance of a component of any web service **230**, are stored in the PDL. The content server **140** handles all communication between device **130** and the web service **230** and can be automatically deployed as a web application archive to any content server.

Device **130**, upon detecting an event in which a component of a web service **230** has been defined, assembles and sends all related inputs to content server **240**, which proxies the request to web service **230** and returns the requested information to device **130**. The Player on device **130** then takes the outputs of web service **230** and binds the data to the UI components in the Application, as displayed on screen **137**.

In one embodiment, the mechanism for binding the outputs of the web service to the UI components is through symbolic references that matches each output to the symbolic name of the UI component. The outputs, in one embodiment, may include meta-data which could become part of the inputs for subsequent interactions with the web service.

For example, if a user of authoring platform **110** wants to present an ATOM feed on device **130**, they would search through a list of UI Components available in the authoring platform, select the feed they want to use, and bind the output of the feed summary to a textbox. The bindings would be saved into the PDL on server **120** and processed by device **130** at runtime. If the ATOM feed does not exist a new one can be added to the web component registry that contains all the configuration data required, such as the actual feed URL, the web component manager URL, and what output fields are available for binding.

In another embodiment, components of web services **230** are available either to the user of authoring platform **110** or otherwise accessible through the SDK and Java APIs of routines **114**. System **200** permits an expanding set of components of web services **230** including, but not limited to: server pages from content server **120**; third-party web services including, but not limited to: searching (such through Google or Yahoo), maps (such as through MapQuest and Yahoo), storefronts (such as through ThumbPlay), SMS share (such as through clickatel), stock quotes, social networking (such as through FaceBook), stock quotes, weather (such as through Accuweather) and/or movie trailers. Other components include web services for communication and sharing through chats and forums and rich messaging alerts, where message alerts are set-up that in turn could have components of Web Services **230** defined within them, including the capture of consumer generated and Web Service supplied rich media and textual content.

System **200** also permits dynamic binding of real-time content, where the inputs and outputs of XML web services are bound to GUI components provided on screen **137**. Thus, for example, a user of authoring platform **110** may bind attributes of UI Objects to a particular data base field on a Server. When running the Application, the current value in the referenced data base will be immediately applied. During the Application session, any other real time changes to these values in the referenced data base will again be immediately displayed.

As an example of dynamic binding of real-time content, an RSS feeds and other forms of dynamic content may be inserted into mobile Applications, such as device **130**, using system **200**. Authoring platform **110** may include a “RSS display” list which permits a user to select RSS channels and feeds from an extensible list of available dynamic content. Meta data, such as titles, abstracts and Images can be revealed immediately by the user as they traverse this RSS display list, bringing the PC experience completely and conveniently to mobile devices **130**. In addition, Authoring platform **110** may include a dialog box that dynamically links objects to data and feeds determined by RSS and chat databases. Any relevant attribute for a page view and/or object can be dynamically bound to a value in a server-side database. This includes elements within complex objects such as: any icon or text element within a graphical list; any icon within a launch strip; any feature within any geographical view of a GIS service object; and/or any virtual room within a virtual tour.

As an example of third-party web services **230** that may be provided using system **200**, a user of authoring platform **110** can place, for example, Yahoo maps into device **130** by binding the required component of the Yahoo Maps Web Service, such as Yahoo Map’s Inputs and/or Outputs to appropriate Objects of authoring platform **110**. System **200** also provides binding to web services for text, image and video searching by binding to components of those web services.

In one embodiment, an Application for displaying on device **130** includes one or more Applications Pages, each referred to herein as an “XSP,” that provides functionality that extends beyond traditional web browsers. The XSP is defined as a PDL, in a similar manner as any Application, although it defines a single page view, and is downloaded to the Player dynamically as required by the PDL definition of the Application. Thus, for example, while JSPs and ASPs, are restricted to the functionality supported by the web browser, the functionality of XSPs can be extended through authoring platform **110** having access to platform dependent routines **114**, such as Java APIs. Combined with dynamic binding functionality, an XSP, a page can be saved as a page object in an author’s “pages” library, and then can be dynamically populated with real-time content simultaneously as the page is downloaded to a given handset Player based on a newly expanded API. XSP Server Pages can also be produced programmatically, but in most cases authoring platform **110** will be a much more efficient way to generate and maintain libraries of dynamically changing XSPs.

With XSPs, Applications Pages that have dynamic content associated with them can be sent directly to device **130**, much like how a web browser downloads an HTML page through a external reference. Without XSPs, content authors would have to define each page in the Application. With XSPs, no pages need to be defined. Thus, for example, in a World Cup Application, one page could represent real-time scores that change continuously on demand. With polling (for example, a prompt to the users asking who they predict will win a game), a back-end database would tabulate the information and then send the results dynamically to the handsets. With a bar chart, the Application would use dynamic PDL with scaling on the fly. For example, the server would recalibrate the bar chart for every ten numbers.

Other combinations of components of web services **230** include, but are not limited to, simultaneous video chat sessions, inside an integrated page view, with a video or television station; multiple simultaneous chat sessions, each with a designated individual and/or group, with each of the chat threads visible inside an integrated page view.

Another extension of an XSP is a widget object. Widgets can be developed from numerous sources including, but not limited to, authoring platform **110**, a Consumer Publishing Tool, and an XML to Widget Conversion Tool where the SDK Widget Libraries are automatically populated and managed, or Widget Selection Lists that are available and can be populated with author defined Icons.

Applications, Players, and Processing in a Device

FIG. **2**B is a schematic of one embodiment of a device **130** illustrating an embodiment of the programming generated by authoring platform **110**. Memory **133** may include several different logical portions, such as a heap **133***a*, a record store **133***b* and a filesystem (not shown).

As shown in FIG. **2**B, heap **133***a* and record store **133***b* include programming and/or content. In general, heap **133***a* is readily accessible by processor **135** and includes, but is not limited to portions that include the following programming: a portion **133***a***1** for virtual machine compliant objects representing a single Page View for screen **137**; a portion **133***a***2** for a Player; a portion **133***a***3** for a virtual machine; and a portion **133***a***4** for an operating system.

Record store **133***b* (or alternatively the filesystem) includes, but is not limited to, portions **133***b***1** for Applications and non-streaming content, which may include portions **133***a***2** for images, portions **133***a***4** for audio, and/or portions **133***a***5** for video. and portions **133***b***2** for non-Application PDLs, such as a Master Page PDL for presenting repeating objects, and Alerts, which are overlayed on the current page view. Other content, such as streaming content may be provided from network interface **131** directly to the Media Codec of device **130** with instructions from Player on how to present the audio or video.

In one embodiment, the Player includes a Threading Model and a Virtual Memory Manager. The Threading Model first manages a queue of actions that can be populated based on Input/Output events, Server-side events, time-based events, or events initiated by user interactions. The Threading Model further manages the simultaneous execution of actions occurring at the same time. The Virtual Memory Manager includes a Logical Virtual Page controller that provides instructions from the record store to the heap, one page at time. Specifically, the Virtual Memory Manager controls the transfer of one of the Application Pages and its virtual machine compliant objects into portion **133***a***1** as instructions readable by the Player or Virtual Machine. When the Player determines that a new set of instructions is required, the information (such as one Application Page is retrieve from the Record store, converted into virtual machine compliant objects (by processor **135** and according to operation by the Player, Virtual Machine, etc). and stored in heap **133***a*. Alternatively, the Player may augment virtual machine compliant objects with its own libraries for managing user interactions, events, memory, etc.

The connection of portions **133***a***1**, **133***a***2**, **133***a***3**, **133***a***4**, record store **133***b* and processor **135** are illustrative of the logical connection between the different types of programming stored in Heap **133***a* and record store **133***b*, that is, how data is processed by processor **135**.

The Player determines which of the plurality of Application Pages in portion **133***b***1** is required next. This may be determined by input actions from the Input Device **139**, or from instructions from the current Application Page. The Player instructs processor **135** to extract the PDF from that Applications Page and store it in portion **133***a***1**. The Player then interprets the Application Page extracted from PDL

which in turn defines all of the virtual machine compliant Objects, some of which could have attributes that refer to images, audio, and/or video stored in portions **133***a***3**, **133***a***4**, **133***a***5**, respectively.

The Virtual Machine in portion **133***a***3** processes the Player output, the Operating System in portion **133***a***3** processes the Virtual Machine output which results in machine code that is processed by the Operating System in portion **133***a***4**.

In another embodiment, the Player is a native program that interacts directly with the operating system.

Embodiments of a Publishing Environment

In one embodiment, authoring platform **110** includes a full-featured authoring tool **112** that provides a what-you-see-is-what-you-get (WYSIWYG) full featured editor. Thus, for example, authoring tool **112** permits a user to design an Application by placing objects on canvas **305** and optionally assigning actions to the objects and save the Application. System **100** then provides the Application and Player to a device **130**. The Application as it runs on device **130** has the same look and operation as designed on authoring platform **110**. In certain embodiments, authoring platform **110** is, for example and without limitation, a PC-compatible or a Macintosh computer.

Authoring platform **110** produces an Application having one or more Applications Pages, which are similar to web pages. That is, each Applications Page, when executed on device **130** may, according to its contents, modify what is displayed on screen **137** or cause programming on the device to change in a manner similar to how web pages are displayed and navigated through on a website.

In one embodiment, authoring tool **112** allows a user to place one or more objects on canvas **305** and associate the objects with an Applications Pages. Authoring platform **110** maintains a database of object data in memory **111**, including but not limited to type of object, location on which page, and object attributes. The user may add settings, events, animations or binding to the object, from authoring tool **112**, which are also maintained in memory **111**. Authoring tool **112** also allows a user to define more than one Applications Page.

In another embodiment, authoring tool **112**, provides Java programming functions of the Java API for specific devices **130** as pull-down menus, dialog boxes, or buttons. This permits a user of authoring platform **110** to position objects that, after being provided as an Application to device **130**, activate such Java functions on the device.

In certain embodiments, authoring platform **110**, as part of system **100**, permits designers to include features of advanced web and web services Applications for access by users of device **130**. Some of the features of advanced web and web services include, but are not limited to: slide shows, images, video, audio, animated transitions, multiple chats, and mouse interaction; full 2-D vector graphics; GIS (advanced LBS), including multiple raster and vector layers, feature sensitive interactions, location awareness, streaming and embedded audio/video, virtual tours, image processing and enhancement, and widgets. In other embodiments the features are provided for selection in authoring platform **110** through interactive object libraries.

In certain embodiments, authoring platform **110**, as part of system **100**, allows the inclusion of child objects which may eventually be activated on device **130** by the user of the device or by time. The uses of the child objects on device **130** include, but are not limited to: mouse over (object selection), hover and fire events and launching of object-specific, rich-media experiences.

In certain other embodiments, authoring platform **110**, as part of system **100**, provides advanced interactive event models on device **130**, including but not limited to: user-, time- and/or location-initiated events, which allow content developers to base interactivity on specific user interactions and/or instances in time and space; timelines, which are critical for timing of multiple events and for animations when entering, on, or exiting pages of the Application; waypoints, which act similar to key frames, to allow smooth movement of objects within pages of the Application. Waypoints define positions on a page object's animation trajectory. When an object reaches a specific waypoint other object timelines can be initiated, creating location-sensitive multiple object interaction, and/or audio can be defined to play until the object reaches the next waypoint.

Authoring platform **110** may also define a Master Page, which acts as a template for an Applications Page, and may also define Alert Pages, which provide user alerts to a user of device **130**.

In certain embodiments, authoring platform **110**, as part of system **100**, provides full style inheritance on device **130**. Thus, for example and without limitation, both master page inheritance (for structural layout inheritance and repeating objects) and object styles (for both look and feel attribute inheritance) are supported. After a style has been defined for an object, the object will inherit the style. Style attributes include both the look and the feel of an object, including mouse interaction, animations, and timelines. Each page may include objects that may be a parent object or a child object. A child object is one that was created by first selecting a parent object, and then creating a child object. Child objects are always part of the same drawing layer as its parent object, but are drawn first, and are not directly selectable when running the Application. A parent object is any object that is not a child object, and can be selected when running the Application.

As an example, the user of authoring tool **112** may create various text objects on canvas **305** using a style that sets the font to red, the fonts of these objects will be red. Suppose user of authoring tool **112** changes the font color of a specific button to green. If later, the user of authoring tool **112** changes the style to blue; all other text objects that were created with that style will become blue except for the button that had been specifically set to green.

In certain other embodiments, authoring platform **110** provides page view, style, object, widget and Application template libraries. Authoring platform **110** may provide templates in private libraries (available to certain users of the authoring platform) and public libraries (available to all users of the authoring platform). Templates may be used to within authoring platform **110** to define the look and feel of the entire Application, specific pages, or specific slide shows and virtual tours a seen on device **130**.

FIGS. **3**A and **3**B illustrate one embodiment of a publisher interface **300** as it appears, for example and without limitation, on screen **115** while executing authoring tool **112**. In one embodiment, publisher interface **300** includes a Menu bar **301**, a Tool bar **303**, a Canvas **305**, a Layer Inspector **307** having subcomponents of a page/object panel **307***a*, an object style panel **307***b*, and a page alert panel **307***c*, and a Resource Inspector **309**.

In general, publisher interface **300** permits a user of authoring platform **110** to place objects on canvas **305** and then associate properties and/or actions to the object, which are stored in the Application. As described subsequently, publisher interface **300** permits a user to program a graphical interface for the screen **137** of device **130** on screen **115** of authoring platform **110**, save an Application having the programming instructions, and save a Player for the device. The intended programming is carried out on device **130** when the device, having the appropriate device platform Player, receives and executes the device-independent Application.

Thus, for example, authoring tool **112** maintains, in memory **111**, a list of every type of object and any properties, actions, events, or bindings that may be assigned to that object. As objects are selected for an Application, authoring tool **112** further maintains, in memory **111**, a listing of the objects. As the user selects objects, publisher interface **300** provides the user with a choice of further defining properties, actions, events, or bindings that may be assigned to each particular object, and continues to store the information in memory **111**.

In one embodiment, publisher interface **300** is a graphical interface that permits the placement and association of objects in a manner typical of, for example, vector graphics editing programs (such as Adobe Illustrator). Objects located on canvas **305** placed and manipulated by the various commands within publisher interface **300** or inputs such as an input device **117** which may be a keyboard or mouse. As described herein, the contents of canvas **305** may be saved as an Application that, through system **100**, provide the same or a similar placement of objects on screen **137** and have actions defined within publisher interface **300**. Objects placed on canvas **305** are intended for interaction with user of device **130** and are referred to herein, without limitation, as objects or UI (user interface) objects. In addition, the user of interface **300** may assign or associate actions or web bindings to UI objects placed on canvas **305** with result in the programming device **130** that cause it to respond accordingly.

Objects include, but are not limited to input UI objects, response UI objects. Input UI objects include but are not limited to: text fields (including but not limited to alpha, numeric, phone number, or SMS number); text areas; choice objects (including but not limited to returning the selected visible string or returning a numeric hidden attribute); single item selection lists (including but not limited to returning the selected visible string or returning a numeric hidden attribute); multi item selection lists (including but not limited to returning all selected items (visible text string or hidden attribute) or cluster item selection lists (returning the hidden attributes for all items).

Other input UI objects include but are not limited to: check boxes; slide show (including but not limited to returning a numeric hidden attribute, returning a string hidden attribute, or returning the hidden attributes for all slides); and submit function (which can be assigned to any object including submit buttons, vectors, etc.).

Response UI Objects may include, but are not limited to: single line text objects, which include: a text Field (including but not limited to a URL, audio URL, or purchase URL), a text button, a submit button, or a clear button. Another response UI objects include: a multiple line text object, which may include a text area or a paragraph; a check box; an image; a video; a slide show (with either video or image slides, or both); choice objects; list objects; or control lists, which control all the subordinate output UI objects for that web component. Control list objects include, but are not limited to: list type or a choice type, each of which may include a search response list or RSS display list.

As a further example of objects that may be used with authoring tool **112**, Table I lists Data Types, Preferred Input, Input Candidates, Preferred Output and Output Candidates for one embodiment of an authoring tool.

TABLE I

One embodiment of supported objects

Data Types	Preferred Input	Input Candidates	Preferred Output	Output Candidates
boolean	Check Box	Check Box	Check Box	Check Box
Int	Text Field (integer)	Text Field (integer) Text Field (Phone #) Text Field (SMS #) Choice List (single select)	Text Field (integer)	Text Field (integer) Text Field (Phone #) Text Field (SMS #) Choice List (single select) Text Button
String	Text Field (Alpha)	Any	Text Field (Alpha)	Any
multilineString	Text Area	Text Area	Text Area	Text Area Paragraph
ImageURL	N/A	N/A	Image	Image Slide Show
VideoURL	N/A	N/A	Video	Video Slide Show
List	Single Item List	Single Item List Multi-Select List Complex List Choice Slide Show	Single Item List	Any List Type Any Choice Type (see Complex List Specification)
ComplexList	Complex List	Single Item List Multi-Select List Complex List	Single Item List	Any List Type (see Complex List Specification)
Slideshow	Slide Show	Slide Show	Slide Show	Slide Show
SearchResponseList	N/A	N/A	Search Response List	Search Response List Control List Complex List Choice
RSSList	N/A	N/A	RSS Display List	RSS Display List Control List Complex List Choice
SingleSelectionList	Choice	Choice Complex List	Choice	Choice Complex List
MultiSelectionList	Multi-Selection List	Multi-Selection List	Multi-Selection List	Multi-Selection List
ServiceActivation	Submit Button	Any	N/A	N/A
ChannelImageURL	N/A	N/A	Image	Image Video Slide Show
ChannelDescription	N/A	N/A	Text Area	Text Area Paragraph Text Field Text Button List Choice
ChannelTitle	N/A	N/A	Text Field	Text Field Text Button Paragraph Text Area List Choice
URL			Text Field (URL request)	Text Field (URL request)
Audio URL			Text Field (Audio URL request)	Text Field (Audio URL request)
Purchase URL			Text Field (Purchase URL request)	Text Field (Purchase URL request)
Image Data			Image	Image Slide Show
Image List Data			Slide Show	Slide Show Image
Persistent Variable	N/A	N/A	N/A	N/A
Pipeline Multiple Select	Multi-select List	Multi-select List Complex List Slide Show	N/A	N/A
Phone Number	Text Field (numeric type)	Text Field Text Button	Text Field (numeric type)	Text Field Text Button
Hidden Attribute	Complex List	Complex List Slide Show	Complex List	Complex List Slide Show
Collection List	N/A	N/A	Slide Show	Complex List Slide Show

In general, publisher interface **300** permits a user to define an Application as one or more Applications Pages, select UI objects from Menu bar **301** or Tool bar **303** and arrange them on an Applications Page by placing the objects canvas **305**. An Application Page is a page that is available to be visited through any navigation event. Application Pages inherit all the attributes of the Master Page, unless that attribute is specifically changed during an editing session.

Authoring platform **110** also stores information for each UI object on each Application Page of an Application. Layer Inspector **307** provides lists of Applications Pages, UI objects on each Applications Page, and Styles, including templates. Objects may be selected from canvas **305** or Layer Inspector **307** causing Resource Inspector **309** to provide lists of various UI objects attributes which may be selected from within the Resource Inspector. Publisher interface **300** also permits a user to save their work as an Application for layer transfer and operation of device **130**. Publisher interface **300** thus provides an integrated platform for designing the look and operation of device **130**.

The information stored for each UI object depends, in part, on actions which occur as the result of a user of device **130** selecting the UI object from the device. UI objects include, but are not limited to: navigational objects, such as widget or channel launch strips or selection lists; message objects for communicating, such as a multiple chat, video chat, phone and/or SMS lists or fields or a pop-up alert; text fields or areas; check boxes; pull down menus; selection lists and buttons; pictures; slide shows; video or LBS maps; shapes or text defined by a variety of tools; a search response; or an RSS display.

In certain embodiments, publisher interface **300** permits a user to assign action to UI objects, including but not limited to, programming of the device **130** or a request for information over network N. In one embodiment, for example and without limitation, publisher interface **300** has a selection to bind a UI object to a web service—that is, associate the UI object or a manipulation or selection of UI object with web services. Publisher interface **300** may also include many drawing and text input functions for generating displays that may be, in some ways, similar to drawing and/or word processing programs, as well as toolbars and for zooming and scrolling of a workspace.

Each UI object has some form, color, and display location associate with it. Further, for example and without limitation, UI objects may have navigational actions (such as return to home page), communications actions (such as to call the number in a phone number field), or web services (such as to provide and/or retrieve certain information from a web service). Each of the these actions requires authoring platform **110** to store the appropriate information for each action. In addition, UI objects may have associated patent or child objects, default settings, attributes (such as being a password or a phone number), whether a field is editable, animation of the object, all of which may be stored by authoring platform **110**, as appropriate.

Menu bar **301** provides access features of publisher interface **300** through a series of pull-down menus that may include, but are not limited to, the following pull-down menus: a File menu **301***a*, an Edit menu **301***b*, a View menu **301***c*, a Project menu **301***d*, an Objects menu **301***e*, an Events menu **301***f*, a Pages menu **301***g*, a Styles menu **301***h*, and a Help menu **301***i*.

File menu **301***a* provides access to files on authoring platform **110** and may include, for example and without limitation, selections to open a new Application or master page, open a saved Application, Application template, or

style template, import a page, alert, or widget, open library objects including but not limited to an image, video, slide show, vector or list, and copying an Application to a user or to Server **120**.

Edit menu **301***b* may include, but is not limited to, selections for select, cut, copy, paste, and edit functions.

View menu **301***c* may include, but is not limited to, selections for zooming in and out, previewing, canvas **305** grid display, and various palette display selections.

Project menu **301***d* may include, but is not limited to, selections related to the Application and Player, such as selections that require a log in, generate a universal Player, generate server pages, activate server APIs and extend Player APIs. A Universal Player will include all the code libraries for the Player, including those that are not referenced by the current Application. Server APIs and Player APIs logically extend the Player with Server-side or device-side Application specific logic.

Objects menu **301***e* includes selections for placing various objects on canvas **305** including, but not limited to: navigation UI objects, including but not limited to widget or channel launch strips or selection lists; message-related UI objects, including but not limited to multiple chat, video chat, phone and/or SMS lists or fields, or a pop-up alert; shapes, which provides for drawing tools; forms-related objects, including but not limited to text fields; scrolling text box, check box, drop-down menu, list menu, submit button or clear button; media-related UI objects such as pictures, slide shows, video or LBS maps; text-related UI objects such as buttons or paragraphs; and variables, including but not limited to time, date and audio mute control.

Events menu **301***f* includes selections for defining child objects, mouse events, animations or timelines.

Pages menu **301***g* includes selection for handling multi-page Applications, and may include selections to set a master page, delete, copy, add or go to Applications Pages.

Styles menu **301***h* includes selections to handle styles, which are the underlying set of default appearance attributes or behaviors that define any object that is attached to a style. Styles are a convenient way for quickly creating complex objects, and for changing a whole collection of objects by just modifying their common style. Selections of Styles menu **301***h* include, but not limited to, define, import, or modify a style, or apply a template. Help menu **301***i* includes access a variety of help topics.

Tool bar **303** provides more direct access to some of the features of publisher interface **300** through a series of pull-down menus. Selections under tool bar **303** may include selections to:

- control the look of publisher interface **300**, such as a Panel selection to control the for hiding or viewing various panels on publisher interface **300**;
- control the layout being designed, such as an Insert Page selection to permit a user to insert and name pages;
- control the functionality of publisher interface **300**, such as a Palettes selection to choose from a variety of specialized palettes, such as a View Palette for zooming and controlling the display of canvas **305**, a Command Palette of common commands, and Color and Shape Palettes;
- place objects on canvas **305**, which may include selections such as: a Navigation selection to place navigational objects, such as widget or channel launch strips or selection lists), a Messages selection to place objects for communicating, such as a multiple chat, video chat, phone and/or SMS lists or fields, or a pop-up alert, a Forms selection to place objects such as text fields or

areas, check boxes, pull down menus, selection lists, and buttons, a Media selection to place pictures, slide shows, video or LBS maps, and a Shapes selection having a variety of drawing tools, a Text selection for placing text, a search response, or an RSS display, and Palettes.

In one embodiment, Tool bar **303** includes a series of pull-down menus that may include, but are not limited to, items from Menu bar **301** organized in the following pull-down menus: a Panel menu **303***a*, an Insert Page menu **303***b*, a Navigation menu **303***c*, a Messages menu **303***d*, a Forms menu **303***e*, a Media menu **303***f*, a Shapes menu **303***g*, a Text menu **303***h*, and a Palettes menu **301***i*.

Panel menu **303***a* permits a user of authoring platform **110** to change the appearance of interface **300** by, controlling which tools are on the interface or the size of canvas **305**. Insert Page menu **303***b* permits a user of authoring platform **110** to open a new Application Page. Navigation menu **303***c* displays a drop down menu of navigational-related objects such as a widget or channel launch strip or selection list. Messages menu **303***d* displays a drop down menu of messaging-related objects such as multiple chat, video chat, phone or SMS lists or fields, and pop-up alerts. Forms menu **303***e* displays a drop down menu of forms-related objects including, but not limited to, a text field, a text area, a check box, a drop down menu, a selection list, a submit button, and a clear button. Media menu **303***f* displays a drop down menu of media-related objects including, but not limited to, a picture, slide show, video or LBS map. Shapes menu **303***g* displays a drop down menu of draw tools, basic shapes, different types of lines and arrows and access to a shape library. Text menu **303***j* displays a drop down menu of text-related objects, including but not limited to a text button, paragraph, search response, RSS display and variables such as time and date.

Palettes menu **301***i* includes a selection of different palettes that can be moved about publisher interface **300**, where each palette has specialized commands for making adjustments or associations to objects easier. Palettes include, but are not limited to: a page view palette, to permit easy movement between Applications Pages; a view palette, to execute an Application or zoom or otherwise control the viewing of an Application; a commands palette having editing commands; a color palette for selection of object colors; and a shapes palette to facilitate drawing objects.

Layer inspector **307** permits a user of publisher interface **300** to navigate, select and manipulate UI objects on Applications Pages. Thus, for example, a Page/objects panel **307***a* of layer inspector **307** has a listing that may be selected to choose an Applications Pages within and Application, and UI objects and styles within an Applications Page. An Object styles panel **307***b* of layer inspector **307** displays all styles on the Applications Page and permits selection of UI objects for operations to be performed on the objects.

Thus, for example, when objects from Menu bar **301** or Tool bar **303** are placed on canvas **305**, the name of the object appears in Page/objects panel **307***a*. Page/objects panel **307***a* includes a page display **307***a***1** and an objects display **307***a***2**. Page display **307***a***1** includes a pull down menu listing all Applications Pages of the Application, and objects display **307***a***2** includes a list of all objects in the Applications Page (that is, objects on canvas **305**).

In general, page/objects panel **307***a* displays various associations with a UI object and permits various manipulations including, but not limited to, operations for parent and child objects that are assigned to a page, and operations for object styles, and permits navigating between page types and object styles, such as switching between the master page and Application pages and deselecting object styles and alerts, opening an Edit Styles Dialog Box and deselecting any master, Application or alert page, or selecting an alert page and deselecting any Master Page or Application Page. A parent or child object can also be selected directly from the Canvas. In either case, the Resource Inspector can then be used for modifying any attribute of the selected object.

Examples of operations provided by page/objects panel **307***a* on pages include, but are not limited to: importing from either a user's private page library or a public page library; deleting a page; inserting a new page, inheriting all the attributes of the Master Page, and placing the new page at any location in the Page List; editing the currently selected page, by working with an Edit Page Dialog Box. While editing all the functions of the Resource Inspector **309** are available, as described subsequently, but are not applied to the actual page until completing the editing process.

Examples of operations provided by of page/objects panel **307***a* on objects, which may be user interface (UI) objects, include but are not limited to: changing the drawing order layer to: bring to the front, send to the back, bring to the front one layer, or send to the back one layer; hiding (and then reshowing) selected objects to show UI objects obstructed by other UI Objects, delete a selected UI Page Object, and editing the currently selected page, by working with a Edit Page Dialog Box.

Object styles panel **307***b* of layer inspector **307** displays all styles on the Applications Page and permits operations to be performed on objects, and is similar to panel **307***a*. Examples of operations provided by object style panel **307***b* include, but are not limited to: importing from either a user's private object library or a public object library; inserting a new object style, which can be inherited from a currently selected object, or from a previously defined style object; and editing a currently selected object style by working with an Edit Style Dialog Box.

Style attributes can be assigned many attributes, including the look, and behavior of any object that inherits these objects. In addition, List Layout Styles can be created or changed as required. A layout style can define a unbounded set of Complex List Layouts, including but not limited to: the number of lines per item in a list, the number of text and image elements and their location for each line for each item in the last, the color and font for each text element, and the vertical and horizontal offset for each image and text element.

Alerts Panel **307***c* provides a way of providing alert pages, which can have many of the attributes of Application Pages, but they are only activated through an Event such as a user interaction, a network event, a timer event, or a system variable setting, and will be superimposed onto whatever is currently being displayed. Alert Pages all have transparent backgrounds, and they function as a template overlay, and can also have dynamic binding to real time content.

Resource inspector **309** is the primary panel for interactively working with UI objects that have been placed on the Canvas **305**. When a UI object is selected on Canvas **305**, a user of authoring platform **110** may associate properties of the selected object by entering or selecting from resource inspector **309**. In one embodiment, resource inspector **309** includes five tab selections: Setting Tab **309***a*, Events Tab **309***b*, Animation Tab **309***c*, Color Tab **309***d* which includes a color palette for selecting object colors, and Bindings Tab **309***e*.

Settings Tab **309***a* provides a dialog box for the basic configuration of the selected object including, but not limited to, name, size, location, navigation and visual settings. Depending upon the type of object, numerous other attributes could be settable. As an example, the Setting Tab for a Text Field may include dialog boxes to define the text field string, define the object style, set the font name, size and effects, set an object name, frame style, frame width, text attributes (text field, password field, numeric field, phone number, SMS number, URL request).

As an example of Setting Tab **309***a*, FIG. **3**B shows various selections including, but not limited to, setting **309***a***1** for the web page name, setting **309***a***2** for the page size, including selections for specific devices **130**, setting **309***a***3** indicating the width and height of the object, and setting **309***a***4** to select whether background audio is present and to select an audio file.

FIG. **3**C illustrates an embodiment of the Events Tab **309***b*, which includes all end user interactions and time based operations. The embodiment of Events Tab **309***b* in FIG. **3**C includes, for example and without limitation, an Events and Services **309***b***1**, Advanced Interactive Settings **309***b***2**, Mouse State **309***b***3**, Object Selected Audio Setting **309***b***4**, and Work with Child Objects and Mouse Overs button **309***b***5**.

Events and Services **309***b***1** lists events and services that may be applied to the selected objects. These include, but are not limited to, going to external web pages or other Applications pages, either as a new page or by launching a new window, executing an Application or JavaScript method, pausing or exiting, placing a phone call or SMS message, with or without single or multiple Player download, show launch strip, or go back to previous page. Examples of events and services include, but are not limited to those listed in Table II

TABLE II

Events and Services	
Goto External Web Page replacing Current Frame	ChoiceObject: Remove Icon from Launch Strip
Goto External Web Page Launched in a New Window	Goto a specific Internal Web Page with Alert. "Backend Synchronization"
Goto a specific Internal Web Page	Goto Widget Object
Goto the next Internal Web Page	Generate Alert. "With a Fire Event"
Goto External Web Page replacing the Top Frame	Send SMS Message from Linked Text Field
Execute JavaScript Method	Toggle Alert. "Display OnFocus, Hide OffFocus"
Pause/Resume Page Timeout	Execute an Application with Alert. "With a Fire Event"
Execute an Application	Goto Logical First Page
Goto a specific Internal Web Page with setting starting slide	Generate Alert with Backend Synchronization
Exit Application	Send SMS Message with Share (Player Download)
Exit Player	Place PhoneCall from linked Text Field with Share (Player Download)
Place PhoneCall from linked Text Field	Send IM Alert from linked Text Field or Text Area
Text Field/Area: Send String on FIRE	Set and Goto Starting Page
ChoiceObject: Add Icon to Launch Strip	Populate Image
Text Field/Area: Send String on FIRE or Numeric Keys	Preferred Launch Strip

Advanced Interactive Settings **309***b***2** include Scroll Activation Enabled, Timeline Entry Suppressed, Enable Server Listener, Submit Form, Toggle Children on FIRE, and Hide Non-related Children, Mouse State **309***b***3** selections are Selected or Fire. When Mouse State Selected is chosen,

Object Selected Audio Setting **309***b***4** of Inactive, Play Once, Loop, and other responses are presented. When Mouse State Fire is chosen, Object Selected Audio Setting **309***b***4** is replaced with FIRE Audi Setting, with appropriate choices presented.

When Work with Child Objects and Mouse Overs button **309***b***5** is selected, a Child Object Mode box pops up, allowing a user to create a child object with shortcut to Menu bar **301** actions that may be used define child objects.

FIG. **3**D illustrates one embodiment of an Animation Tab **309***c*, which includes all animations and timelines. The Color Tab includes all the possible color attributes, which may vary significantly by object type.

Animation Tab **309***c* includes settings involved in animation and timelines that may be associated with objects. One embodiment of Animation Tab **309***c* is shown, without limitation, in FIG. **3**D, and is described, in Rempell ("Rempell").

A Color Tab **309***d* includes a color palette for selecting object colors.

Bindings Tab **309***e* is where web component operations are defined and dynamic binding settings are assigned. Thus, for example, a UI object is selected from canvas **305**, and a web component may be selected and configured from the bindings tab. When the user's work is saved, binding information is associated with the UI object that will appear on screen **137**.

FIG. **3**E illustrates one embodiment of Bindings Tab and includes, without limitation, the following portions: Web Component and Web Services Operations **309***e***1**, Attributes Exposed list **309***e***2**, panel **309***e***3** which includes dynamic binding of server-side data base values to attributes for the selected object, Default Attribute Value **309***e***4**, Database Name **309***e***5**, Table Name **309***e***6**, Field Name **309***e***7**, Channel Name **309***e***8**, Channel Feed **309***e***9**, Operation **309***e***10**, Select Link **309***e***11**, and Link Set checkbox **309***e***12**.

Web Component and Web Services Operations **309***e***1** includes web components that may be added, edited or removed from a selected object. Since multiple web components can be added to the same object, any combination of mash-ups of 3rd party web services is possible. When the "Add" button of Web Component and Web Services Operations **309***e***1** is selected, a pop-up menu **319**, as shown in FIG. **3**F, appears on publisher interface **300**. Pop-up menu **319** includes, but is not limited to, the options of: Select a Web Component **319***a*; Select Results Page **319***b*; Activation Options **319***c*; Generate UI Objects **319***d*; and Share Web Component **319***e*.

The Select a Web Component **319***a* portion presents a list of web components. As discussed herein, the web components are registered and are obtained from web component registry **220**.

Select Results Page **319***b* is used to have the input and output on different pages—that is, when the Results page is different from Input page. The default selected results page is either the current page, or, if there are both inputs and outputs, it will be set provisionally to the next page in the current page order, if one exists.

Activation Options **319***c* include, if there are no Input UI Objects, a choice to either "Preload" the web component, similar to how dynamic binding, or have the web component executed when the "Results" page is viewed by the consumer.

Generate UI Objects **319***c*, if selected, will automatically generate the UI objects. If not selected, then the author will bind the Web Component Inputs and Results to previously created UI Objects.

Share Web Component **319***e* is available and will become selected under the following conditions: 1) Web Component is Selected which already has been used by the current Application; or 2) the current Input page is also a “Result” page for that Web component. This permits the user of device **130**, after viewing the results, to extend the Web Component allowing the user to make additional queries against the same Web Component. Examples of this include, but are not limited to, interactive panning and zooming for a Mapping Application, or additional and or refined searches for a Search Application.

Dynamic Binding permits the binding of real time data, that could either reside in a 3rd party server-side data base, or in the database maintained by Feed Collector **1010** for aggregating live RSS feeds, as described subsequently with reference to FIG. **10**.

Referring again to FIG. **3**E, Attributes Exposed list **309***e***2** are the attributes available for the selected object that can be defined in real time through dynamic binding.

Panel **309***e***3** exposes all the fields and tables associated with registered server-side data bases. In one embodiment, the user would select an attribute from the “Attributes Exposed List” and then select a data base, table and field to define the real time binding process. The final step is to define the record. If the Feed Collector data base is selected, for example, then the RSS “Channel Name” and the “Channel Feed” drop down menus will be available for symbolically selected the record. For other data bases the RSS “Channel Name” and the “Channel Feed” drop down menus are replaced by a “Record ID” text field.

Default Attribute Value **309***e***4** indicates the currently defined value for the selected attribute. It will be overridden in real time based on the dynamic linkage setting.

Database Name **309***e***5** indicates which server side data base is currently selected.

Table Name **309***e***6** indicates which table of the server side data base is currently selected.

Field Name **309***e***7**, indicates which field form the selected table of the server side data base is currently selected.

Channel Name **309***e***8** indicates a list of all the RSS feeds currently supported by the Feed Collector. This may be replaced by “Record ID” if a data base other than the Feed Collector **1010** is selected.

Channel Feed **309***e***9** indicates the particular RSS feed for the selected RSS Channel. Feed Collector **1010** may maintain multiple feeds for each RSS channel.

Operation **309***e***10**, as a default operation, replaces the default attribute value with the real time value. In other embodiments this operation could be append, add, subtract, multiply or divide.

Select Link **309***e***11** a button that, when pressed, creates the dynamic binding. Touching the “Select Link” will cause the current data base selections to begin the blink is some manner, and the “Select Link” will change to “Create Link”. The user could still change the data base and attribute choices. Touching the “Create Link” will set the “Link Set” checkbox and the “Create Link” will be replaced by “Delete Link” if the user wishes to subsequently remove the link. When the application is saved, the current active links are used to create the SPDL.

Link Set checkbox **309***e***12** indicates that a link is currently active.

An example of the design of a display is shown in FIGS. **4**A and **4**B according the system **100**, where FIG. **4**A shows publisher interface **300** having a layout **410** on canvas **305**, and FIG. **4**B shows a device **130** having the resulting layout **420** on screen **137**. Thus, for example, authoring platform **110** is used to design layout **410**. Authoring platform **110** then generates an Application and a Player specific to device **130** of FIG. **4**B. The Application and Player are thus used by device **130** to produce layout **420** on screen **137**.

As illustrated in FIG. **4**A, a user has placed the following on canvas **305** to generate layout **410**: text and background designs **411**, a first text input box **413**, a second text input box **415**, and a button **417**. As an example which is not meant to limit the scope of the present invention, layout **410** is screen prompts a user to enter a user name in box **413** and a password in box **415**, and enter the information by clicking on button **417**.

In one embodiment, all UI objects are initially rendered as Java objects on canvas **305**. When the Application is saved, the UI objects are transformed into the PDL, as described subsequently.

Thus, for example, layout **410** may be produced by the user of authoring platform **110** selecting and placing a first Text Field as box **413** then using the Resource Inspector **309** portion of interface **300** to define its attributes.

Device User Experience

Systems **100** and **200** provide the ability for a very large number of different types of user experiences. Some of these are a direct result of the ability of authoring platform **110** to bind UI objects to components of web services. The following description is illustrative of some of the many types of experiences of using a device **130** as part of system **100** or **200**.

Device **130** may have a one or more of a very powerful and broad set of extensible navigation objects, as well as object- and pointer-navigation options to make it easy to provide a small mobile device screen **137** with content and to navigate easily among page views, between Applications, or within objects in a single page view of an Application.

Navigation objects include various types of launch strips, various intelligent and user-friendly text fields and scrolling text boxes, powerful graphical complex lists, as well as Desktop-level business forms. In fact, every type of object can be used for navigation by assigning a navigation event to it. The authoring tool offers a list of navigation object templates, which then can be modified in numerous ways.

Launch Strips and Graphical List Templates Launch Strips

Launch strips may be designed by the user of authoring platform **110** with almost no restrictions. They can be stationary or appear on command from any edge of the device, their size, style, audio feedback, and animations can be freely defined to create highly compelling experiences.

FIG. **5** shows a display **500** of launch strips which may be on displayed canvas **305** or on screen **137** of device **130** having the proper Player and Application. Display **501** includes a portal-type Launch Strip **501** and a channel-type Launch Strip **502**, either one of which may be included for navigating the Application.

Launch Strip **501** includes UI objects **501***a*, **501***b*, **501***c*, **501***d*, and **501***e* that that becomes visible from the left edge of the display, when requested. UI objects **501***a*, **501***b*, **501***c*, **501***d*, and **501***e* are each associated, through resource inspector **309** with navigational instructions, including but not limited to navigating to a different Applications Page, or requesting web content. When the Applications Page, having been saved by authoring platform **110** and transferred to display **130**, is executed on device **130**, a user of the device may easily navigate the Application.

Launch Strip **502** includes UI objects **502***b*, **502***c*, **502***d*, and **503***e* that that becomes visible from the bottom of the display, when requested. UI objects **501***a*, **501***b*, **501***c*, **501***d*, and **501***e* are each associated, through resource inspector

309 with navigational instructions, including but not limited to navigating to a different Applications Page, or requesting web content. Launch Strip **502** also includes UI objects **502***a* and **503***g*, which include the graphic of arrows, and which provide access to additional navigation objects (not shown) when selected by a user of device **130**. Launch strip **502** may also include sound effects for each channel when being selected, as well as popup bubble help.

Additional navigational features are illustrated in FIG. **6**A as a display of a Channel Selection List **601***a*, in FIG. **6**B as a display of a Widget Selection List **601***b*, and in FIG. **6**C as display of a Phone List **601***c*. Lists **601***a*, **601***b*, and **601***c* may be displayed on canvas **305** or on screen **137** of device **130** having the proper Player and Application. As illustrated, graphical lists **601***a*, **601***b*, and **601***c* may contain items with many possible text and image elements. Each element can be defined at authoring time and/or populated dynamically through one or more Web Service **250** or API. Assignable Navigation Events. All objects, and/or all elements within an object, can be assigned navigation events that can be extended to registered web services or APIs. For example, a Rolodex-type of navigation event can dynamically set the starting slide of the targeted page view (or the starting view of a targeted Application).

In the embodiment of FIGS. **6**A, **6**B, and **6**C, each list **601***a*, **601***b*, and **601***c* has several individual entries that are each linked to specific actions. Thus Channel Selection List **601***a* shows three objects, each dynamically linked to a web service (ESPN, SF 49ers, and Netflix) each providing a link to purchase or obtain items from the Internet. Widget Selection List **601***b* includes several objects presenting different widgets for selecting. Phone List **601***c* includes a list phone number objects of names that, when selected by a user of device **130** cause the number to be dialed Entries in Phone List **601***c* may be generated automatically from either the user's contact list that is resident on the device, or though a dynamic link to any of user's chosen server-side facilities such as Microsoft Outlook, Google Mail, etc. In one embodiment, Phone List **601***c* may be generated automatically using a web component assigned to the Application, which would automatically perform those functions.

In another embodiment, authoring platform **110** allows a navigation selection of objects with a Joy Stick and/or Cursor Keys in all 4 directions. When within a complex object the navigation system automatically adopts to the navigation needs for that object. For coordinate sensitive objects such as geographical information services (GIS) and location-based services (LBS) or virtual tours a soft cursor appears. For Lists, scrolling text areas and chats, Launch strips, and slide shows the navigation process permits intuitive selection of elements within the object. Scroll bars and elevators are optionally available for feedback. If the device has a pointing mechanism then scroll bars are active and simulate the desktop experience.

Personalization and Temporal Adoption

System **100** and **200** permit for the personalization of device **130** by a variety of means. Specifically, what is displayed on screen **137** may depend on either adoption or customization. Adoption refers to the selection of choices, navigation options, etc. are based on user usage patterns. Temporal adoption permits the skins, choices, layouts, content. widgets, etc. to be further influenced by location (for example home, work or traveling) and time of day (including season and day of week). Customization refers to user selectable skins, choices, layouts, dynamic content, widgets, etc. that are available either through a customization on the phone or one that is on the desktop but dynamically linked to the user's other internet connected devices.

To support many personalization functions there must be a convenient method for maintaining, both within a user's session, and between sessions, memory about various user choices and events. Both utilizing a persistent storage mechanism on the device, or a database for user profiles on a server, may be employed.

FIG. **7** shows a display **700** of a mash-up which may be on displayed canvas **305** or on screen **137** of device **130** having the proper Player and Application. Display **700** includes several object **701** that have been dynamically bound, including an indication of time **701***a*, an indication of unread text messages **701***b*, an RSS news feed **701***c* (including 2 "ESPN Top Stories" **701***c***1** and **701***c***2**), components **701***d* from two Web Services—a weather report ("The Weather Channel"), and a traffic report **701***e* ("TRAFFIC.COM").

In assembling the information of display **700**, device **130** is aware of the time and location of the device—in this example the display is for a workday when a user wakes. Device **130** has been customized so that on a work day morning the user wishes to receive the displayed information. Thus in the morning, any messages received overnight would be flagged, the user's favorite RSS sports feeds would be visible, today's weather forecast would be available, and the current traffic conditions between the user's home and office would be graphically depicted. User personalization settings may be maintained as persistent storage on device **130** when appropriate, or in a user profile which is maintained and updated in real-time in a server-side data base.

Push Capable Systems

In another embodiment system **100** or **200** is a push-capable system. As an example, of such systems, short codes may be applied to cereal boxes and beverage containers, and SMS text fields can be applied to promotional websites. In either case, a user of device **130** can text the short code or text field to an SMS server, which then serves the appropriate Application link back to device **130**.

FIG. **8** is a schematic of an embodiment of a push enabled system **800**. System **800** is generally similar to system **100** or **200**. Device **130** is shown as part of a schematic of a push capable system **800** in FIG. **8**. System **800** includes a website system **801** hosting a website **801**, a server **803** and a content server **805**. System **801** is connected to servers **803** and/or **805** through the Internet. Server **803** is generally similar to server **120**, servers **805** is generally similar to server **140**.

In one embodiment, a user sets up a weekly SMS update from website system **801**. System **801** provides user information to server **803**, which is an SMS server, when an update is ready for delivery. Server **803** provides device **130** with an SMS indication that the subscribed information is available and queries the user to see if they wish to receive the update. Website **801** also provides content server **805** with the content of the update. When a user of device **130** responds to the SMS query, the response is provided to content server **805**, which provides device **130** with updates including the subscribed content.

In an alternative embodiment of system **800**, server **803** broadcasts alerts to one or more devices **130**, such as a logical group of devices. The user is notified in real-time of the pending alert, and can view and interact with the massage without interrupting the current Application.

FIG. **9** is a schematic of an alternative embodiment of a push enabled system **900**. System **900** is generally similar to system **100**, **200**, or **800**. In system **900** a user requests information using an SMS code, which is delivered to device

130. System **900** includes a promotional code **901**, a third-party server **903**, and content server **805**. Server **803** is connected to servers **803** and/or **805** through the Internet, and is generally similar to server **120**.

A promotional code **901** is provided to a user of device **130**, for example and without limitation, on print media, such as on a cereal box. The use of device **130** sends the code server **903**. Server **903** then notifies server **805** to provide certain information to device **130**. Server **805** then provides device **130** with the requested information.

Device Routines

Device routines **114** may include, but are not limited to: an authoring tool SDK for custom code development including full set of Java APIs to make it easy to add extensions and functionality to mobile Applications and tie Applications to back-end databases through the content server **140**; an expanding set of web services **250** available through the authoring tool SDK; a web services interface to SOAP/XML enabled web services; and an RSS/Atom and RDF feed collector **1010** and content gateway **1130**.

Authoring Tool SDK for Custom Code Development Including Full Set of Java APIs

In one embodiment, authoring platform **110** SDK is compatible for working with various integrated development environments (IDE) and popular plug ins such as J2ME Polish. In one embodiment the SDK would be another plug in to these IDEs. A large and powerful set of APIs and interfaces are thus available through the SDK to permit the seamless extension of any Application to back end business logic, web services, etc. These interfaces and APIs may also support listeners and player-side object operations.

There is a large set of listeners that expose both player-side events and dynamically linked server side data base events. Some examples of player side events are: player-side time based event, a page entry event, player-side user interactions and player-side object status. Examples of server-side data base events are when a particular set of linked data base field values change, or some filed value exceeds a certain limit, etc.

A superset of all authoring tool functionality is available through APIs for layer-side object operations. These include, but are not limited to: page view level APIs for inserting, replacing, and or modifying any page object; Object Level APIs for modifying any attribute of existing objects, adding definitions to attributes, and adding, hiding or replacing any object.

Authoring Tool SDK Available Web Services

The APIs permit, without limit, respond, with or without relying on back-end business logic, that is, logic that what an enterprise has developed for their business, to any player-side event or server-side dynamically linked data-base, incorporating any open 3rd party web service(s) into the response.

RSS/ATOM and RDF Feed Conversion Web Service

FIG. **10** is a schematic of one embodiment a system **1000** having a feed collector **1010**. System **1000** is generally similar to system **100**, **200**, **800**, or **900**. Feed collector **1010** is a server side component of system **100** that collects RSS, ATOM and RDF format feeds from various sources **1001** and aggregates them into a database **1022** for use by the Applications built using authoring platform **110**.

Feed collector **1010** is a standard XML DOM data extraction process, and includes Atom Populator Rule **1012**, RSS Populator Rule **1013**, RDF Populator Rule **1014**, and Custom Populator Rule **1016**, DOM XML Parsers **1011**, **1015**, and **1017**, Feed Processed Data Writer **1018**, Custom Rule

Based Field Extraction **1019**, Rule-based Field Extraction **1020**, Channel Data Controller **1021**, and Database **1022**.

The feed collector is primarily driven by two sets of parameters: one is the database schema (written as SQL DDL) which defines the tables in the database, as well as parameters for each of the feeds to be examined. The other is the feed collection rules, written in XML, which can be used to customize the information that is extracted from the feeds. Each of the feeds is collected at intervals specified by the feed parameter set in the SQL DDL.

Feed collector **1010** accepts information from ATOM, RDF or RSS feed sources **1001**. Using a rules-based populator, any of these feeds can be logically parsed, with any type of data extraction methodology, either by using supplied rules, or by the author defining their own custom extraction rule. The rules are used by the parser to parse from the feed sources, and the custom rule base field extraction replaces the default rules and assembles the parsed information into the database

In particular, Atom Populator Rule **1012**, RSS Populator Rule **1013**, RDF Populator Rule **1014**, Custom Populator Rule **1016**, and DOM XML Parsers **1011**, **1015**, and **1017** are parse information from the feeds **1001**, and Feed Processed Data Writer **1018**, Custom Rule Based Field Extraction **1019**, Rule-based Field Extraction **1020**, and Channel Data Controller **1021**, supply the content of the feeds in Database **1022**, which is accessible through content server **140**.

FIG. **11** is a schematic of an embodiment of a system **1100** having a Mobile Content Gateway **1130**. System **1100** is generally similar to system **100**, **200**, **800**, **900**, or **1000**. System **1100** includes an SDK **1131**, feed collector **1010**, database listener **1133**, transaction server **1134**, custom code **1135** generated from the SDK, Java APIs, Web Services **1137**, and PDL snippets compacted objects **1139**. System **1100** accepts input from Back End Java Code Developer **1120** and SOAP XML from Web Services **1110**, and provides dynamic content to server **140** and Players to devices **130**.

In one embodiment authoring platform **110** produces a Server-side PDL (SPDL) at authoring time. The SPDL resides in server **120** and provides a logical link between the Application's UI attributes and dynamic content in database **1022**. When a user of device **130** requests dynamic information, server **120** uses the SPDL to determine the link required to access the requested content.

In another embodiment Web Services **1137** interface directly with 3rd party Web Services **1110**, using SOAP, REST, JAVA, JavaScript, or any other interface for dynamically updating the attributes of the Application's UI objects.

XSP Web Pages as a Web Service

In one embodiment, a PDL for a page is embedded within an HTML shell, forming one XSP page. The process of forming XSP includes compressing the description of the page and then embedding the page within an HTML shell.

In another embodiment, a PDL, which contains many individual page definitions, is split into separate library objects on the server, so that each page can to presented as a PDL as part of a Web Service.

Prior to compression the code has already been transformed so that there are no dependencies on the original programming language (Java), and The code and data have been reduced by 4 to 10 times.

Compression has two distinct phases. The first takes advantage of how the primitive representations had been assembled, while the second utilizes standard LZ encoding.

The final result is an overall reduction of 40 to 100 times the original size as represented by Java serialized objects.

One embodiment for compacting data that may be used is described in Rempell. In that patent the compressed data is described as being a database. The terminology used here is a PDL, that is the "internal database" of Rempell is equivalent to the PDL of the present Application.

In Rempell, a process for compacting a "database" (that is, generating a compact PDL) is described, wherein data objects, including but not limited to, multi media objects such as colors, fonts, images, sound clips, URLs, threads, and video, including multi level animation, transformation, and time line are compacted. As an extension to Rempell in all cases these objects are reduced and transformed to Boolean, integer and string arrays.

The compression technique involves storing data in the smallest arrays necessary to compactly store web page information. The technique also includes an advanced form of delta compression that reduces integers so that they can be stored in a single byte, a as high water marks.

Thus, for example, the high water mark for different types of data comprising specific web site settings are stored in a header record as Boolean and integer variables and URL and color objects. Data that defines web page, paragraph, text button, and image style and text button, image and paragraph high watermark settings can be stored in one-dimensional arrays as Boolean, integer and string variables and URL, font, image or thread objects at. The URL, color, font, image and thread objects can also be created as required

Data that defines text button, image, paragraph, or other parent objects and paragraph line high watermark settings can be stored in two-dimensional arrays (by web page and by object number) as Boolean, integer, string, floating point variables and URLs. Again, the URL, color, font, image, audio clip, video clip, text area and thread objects can also be created as required. Data that defines a paragraph line and paragraph line segment high watermarks can be stored in three-dimensional arrays (by web page, by paragraph number, and by line number) as Boolean, integer or string variables. Again, the URL, color or font objects can be created as required. Data that defines a paragraph line segment can be stored into four-dimensional arrays (by web page, by paragraph number, by line number and by line number segment) as Boolean, integer or string variables or URL, color and font objects.

As a data field is added, changed or deleted, a determination is made at on whether a value for a given high watermark needs to be changed. If so, it is updated. As a specific method in the build engine is called, a determination is made on whether a feature flag needs to be set. For example, if a particular JAVA method is called, which requires an instance of a certain JAVA Class to be executed by the run time engine, then that JAVA Class is flagged, as well as any supporting methods, variables and/or object definitions.

In one implementation, the header record, the style record, the web page record, and the object records, are carefully defined in a specific order, written in that order, and explicitly cast by object type when read by the run time engine. Exception handling can be implemented to recover from any errors. This helps assure that data integrity is maintained throughout the build and run time processes.

Also described in Rempell is the "run generation process." This is equivalent generating a Player in the present application. This process starts when the build process detects that the user is finished defining the web site (user has saved the web site and invokes the run generation process), and concludes with the actual uploading of all the necessary web site run time files to the user's server.

In one embodiment, the PDL includes a first record, a "Header" record, which contains can include the following information:

1: A file format version number, used for upgrading database in future releases.

2: The default screen resolution, in virtual pixels, for both the screen width and height. This is usually set to the web designer's screen resolution, unless overwritten by the user.

3: Whether the Application is a web site.

4: Virtual web page size settings. A calculation is performed by the build engine method, in order to calculate what the maximum web page length is, after reformatting all paragraphs on all internal web pages, based on the default screen resolution.

5: Web page and styles high watermarks.

6: The Websitename.

As new web pages or new objects are created by the user, or as text is added to or deleted from a paragraph, or as new styles are created or deleted, appropriate high watermarks are set, in order to show the current number of each of these entities. Thus, the values for the number of active web pages and the number of text button, image, paragraph or other styles are written as high watermarks in the header. The high watermarks for the number of text button, image, paragraph or other objects that exist for each web page, the number of lines for each paragraph object, and the number of line segments for each paragraph line are written within the body of the PDL, and used as settings for each of the loops in the four-dimensional data structure. Because no structural limits are set on the number of web pages, objects per web page, styles, or paragraph size, these high watermarks greatly reduce the external database file size, and the time it takes for the run time engine to process the data stored in its database.

The settings for all paragraph, text button and image styles are then written as a style record based on their high watermark. This data includes Boolean and integer variables, and font and color objects, written as a one-dimensional array, based on the high watermark values for the number of styles that exist.

The body of the PDL is then written. All Boolean values are written inside a four-dimensional loop. The outside loop contains the Boolean values used to define web pages (i.e. a one-dimensional array definition) as well as the high watermarks for the number of text button, image, paragraph or other objects per web page, with the loop set at the high watermark which defines the number of existing web pages for this web site structure. The second level consists of three or more two dimensional loops with the loops set to the high watermarks defining the actual number of text button, image, and paragraph or other objects that appear on any given web page and contains the values used to define web page objects ((i.e. a two-dimensional array definition; web page number by object number). Included within the loop for paragraph objects are the high watermarks for the number of lines for each paragraph object. The third loop is set by the high watermark defining the actual number of paragraph lines that for all paragraphs on any web page and contains the values used to define paragraph lines (i.e. a three-dimensional array definition; web page number by object number by paragraph line.) Included within the loop for paragraph lines are the high watermarks for the number of line segments for each paragraph line. The inner most loop is set by the high watermarks defining the number of line segments per paragraph line and contains the values used to define paragraph line segments (i.e. a four-dimensional array definition; web page number by object number by paragraph line by paragraph line segment).

All integer values are written inside a four-dimensional loop. Their four loops are controlled by the same high watermark settings as used for the Boolean records, and they describe the same logical entities.

Multimedia objects are written inside a two-dimensional loop. They include URL, color, and font objects, and can include other types of objects. A URL object is the encoded form of a URL Address, used by a web browser or a JAVA method to access files and web addresses. All multimedia objects must be serialized before they can be written. This means that the objects are converted into a common external definition format that can be understood by the appropriate deserialization technique when they are read back in and cast into their original object structure. The outside loop contains web page related objects, and the inner loop contains image, text button, paragraph, etc. related URL, color, and font objects. The outer loop is defined by the web page high watermark and the inner loops by the high watermarks for the actual number of text button, image, paragraph or other objects on a web page.

String records are written inside a four-dimensional loop. The outer loop may be empty. The second loop can include the string values for text button objects, audio and video filenames, and audio and video channel names. The third loop contains values for paragraph line related data, and the innermost loop contains the values for paragraph line segment definitions. The string records are controlled by the same high watermarks as those used for Boolean and integer records. String records are stored utilizing an appropriate field delimiter technology. In one implementation, a UTF encoding technology that is supported by JAVA is utilized.

Single and double floating-point, and long integer records are written inside a two-dimensional loop. The outer loop may be empty. The inner loop contains mathematical values required for certain animations and image processing algorithms. The single and double floating-point, and long integer records are controlled by the same high watermarks as those used for Boolean and integer records.

In one embodiment, a versionizing program analyzes the feature flags, and only those variable definitions, defined in the "Main" object class, relating to the object classes and methods that will be executed at run time, are extracted. All references to object classes that will be called at run time are extracted, creating the source code for the run engine "Main" object class that is ready for compilation.

All external image, video and audio files are resolved. The external references can be copied to designated directories, either on the user's local disk or file server. The file Pathnames can be changed to reflect these new locations. During the installation of the build tools, the necessary class libraries are either installed on the local system or made available on the server where the build tools can be optionally located. The necessary environmental variables are set to permit normal access to the required class libraries.

The customized run engine and a library of the referenced run time classes are compiled and converted into byte code. Finally, the run time engine for the web site is created. The required set of class objects required at run time is flagged for inclusion into the CAB/JAR file.

Next, an HTML Shell File (HSF) is constructed. The first step of this process is to determine whether the dynamic web page and object resizing is desired by testing the Application setting. If the Application was a web page, and thus requiring dynamic web page and object resizing, virtual screen resolution settings are placed in an appropriate HTML compliant string. If the Application is a banner or other customized Application, the absolute values for the run time object (applet size) height and width are placed in an appropriate HTML compliant string as absolute width and height values.

An analysis is made for the background definition for the first internal web page. If a background pattern is defined, an appropriate HTML compliant string for setting the HTML "background" to the same background image is generated. If the first web page definition is a color instead, then the RGB values from those colors are converted to hexadecimal and an appropriate HTML compliant String is generated setting the "bgcolor" to the required hexadecimal value. This process synchronizes the web page background with the background that will be drawn by the web browser when it first interprets the HSF.

Thereafter, a JAVA method generates HTML and JavaScript compliant strings, that when executed by a web browser, generate additional sets of HTML and JavaScript compliant strings that are again executed by the web browser. More specifically, if the Application required dynamic web page and object resizing then JavaScript and HTML compliant strings are generated so that, when interpreted by the web browser at the time the HTML Shell File is initialized, the screen resolution sensing JAVA applet (SRS) will be executed. JavaScript code is generated in order to enable JavaScript to SRS applet communication. In one implementation, the code is generated by performing the following functions:

1: Determine the current web browser type.
2: Load the SRS from either a JAR or CAB File, based on web browser type.
3: Enter a timing loop, interrogating when the SRS is loaded.
4: When the SRS returns an "available" status, interrogate the SRS, which will return the current screen and window's actual height and width.
5: Convert the virtual screen resolution settings into appropriate absolute screen width and height values.

Strings defining additional JavaScript code are generated that perform the following steps at the time the HSF is initialized by the web browser:

1: Generate HTML compliant strings that set the run time engine's applet size to the appropriate values.
2: Generate an HTML complaint string that contains a "param" definition for linking the run time engine to the PDL.
3: Generate an HTML complaint string, dependent upon the type of web browser, which causes the current web browser to load either the JAR or the CAB File(s).
4: Generate JavaScript Code compliant strings that create and dynamically write the applet size defining HTML strings utilizing the JavaScript "document.write" function. This dynamically created code causes the web browser to execute the run time engine, in the correctly sized window, from the correct JAR or CAB file, and linked to the external database.

The writing out the above-generated HTML and JavaScript compliant strings creates the HSF. The necessary security policy permissions are asserted, and a "Websitename".html file is created.

In one embodiment, the processes for creating the CAB and JAR Files is as follows. The image objects, if any, which were defined on the first internal web page are analyzed. If they are set to draw immediately upon the loading of the first web page, then they are flagged for compression and inclusion in the CAB and JAR Files. The feature flags are analyzed to determine which JAVA classes have been compiled. These class files are flagged for compression and inclusion in the library CAB and JAR Files. Strings that are BAT compliant definitions are created that will, when executed in DOS, create compressed CAB and JAR Files. These CAB and JAR Files contain the compressed versions of all necessary JAVA class files, image files, the "Websitename".class, customized run time engine file, and the "Websitename".dta database file. In one implementation of the invention, two BAT files are created. The first, when executed, will create a CAB/JAR file with the "Websitename".dta database file and the customized "main" run time engine, excluding all the image and button object animation, transformation, and image processing code. The second BAT file, when executed, will create a CAB/JAR file with all the library of all the referenced image and button object animation, transformation, and image processing code.

The necessary security policy permissions for file creation are then asserted, and "Websitename".bat and "Websitenamelib".bat files are written. The "Websitename".bat and "Websitename".bat files are then executed under DOS, creating compressed "Websitename".cab and "Websitenamelib".cab files and compressed "Websitename" jar and "Websitenamelib" jar files. The HTML Shell File and the JAR and CAB files are then, either as an automatic process, or manually, uploaded to the user's web site. This completes the production of an XSP page that may be accessed through a web browser.

Displaying Content on a Device

Decompression Management

Authoring platform **110** uses compaction to transform the code and data in an intelligent way while preserving all of the original classes, methods and attributes. This requires both an intelligent server engine and client (handset) Player, both of which fully understand what the data means and how it will be used.

The compaction technology described above includes transformation algorithms that deconstruct the logic and data into their most primitive representations, and then reassembles them in a way that can be optimally digested by further compression processing. This reassembled set of primitive representations defines the PDL of authoring platform **110**.

Prior to compression the code has already been transformed so that there are no dependencies on the original programming language (Java). The data is then compressed by first taking advantage of how the primitive representations had been assembled, and then by utilizing standard LZ encoding. The final result is an overall reduction of 40 to 100 times the original size as represented by Java serialized objects.

The Player, when preparing a page view for execution, decompresses and then regenerate the original objects, but this time in compliance with the programming APIs of device **130**. Specifically, device **130** operates on compacted image pages, one at a time. The cache manager retrieves, decompresses, and reassembles the compacted page images into device objects, which are then interpreted by device **130** for display on screen **137**.

Response Director

In one embodiment, system **100** includes a Response Director, which determines a user's handset, fetches the correct Application from different databases, and delivers a respective highly compressed Application in a PDL format over the air (OTA).

In one embodiment, the Response Director operates on a network connected computer to provide the correct Player to a given device based on the information the device sent to it. As an example, this may occur when a device user enters their phone number into some call-to-action web page. The response director is called and sends an SMS message to the device, which responds, beginning the recognition process.

FIG. **12** illustrates one embodiment of a system **1200** that includes a response director **210**, a user agent database **1201**, an IP address database **1203**, and a file database **1205**. System **1200** is generally similar to system **100**, **200**, **800**, **900**, **1000**, or **1100**.

Databases **1201**, **1203**, and **1205** may reside on server **120**, **210**, or any computer system in communication with response director **210**. System **1200**, any mobile device can be serviced, and the most appropriate Application for the device will be delivered to the device, based on the characteristics of the device.

User agent database **1201** includes user agent information regarding individual devices **130** that are used to identify the operating system on the device. IP address database **1203** identifies the carrier/operator of each device **130**. File database **1205** includes data files that may operate on each device **130**.

The following is an illustrative example of the operation of response director **210**. First, a device **1300** generates an SMS message, which automatically sends an http:// stream that includes handset information and its phone number to response director **210**. Response director **210** then looks at a field in the http header (which includes the user agent and IP address) that identifies the web browser (i.e., the "User Agent"). The User Agent prompts a database lookup in user agent database **1201** which returns data including, but not limited to, make, model, attributes, MIDP 1.0 MIDP 2.0, WAP and distinguishes the same models from different countries. A lookup of the IP address in IP address 1203 identifies the carrier/operator.

File database **1205** contains data types, which may include as jad1, jad2, html, wml/wap2, or other data types, appropriate for each device **130**. A list of available Applications are returned to a decision tree, which then returns, to device **130**, the Application that is appropriate for the respective device. For each file type, there is an attributes list (e.g., streaming video, embedded video, streaming audio, etc.) to provide enough information to determine what to send to the handset.

Response director **210** generates or updates an html or jad file populating this text file with the necessary device and network dependent parameters, including the Application dependent parameters, and then generate, for example, a CAB or JAD file which contains the necessary Player for that device. For example, the jad file could contain the operator or device type or extended device-specific functions that the player would then become aware of

If there is an Application that has a data type that device **130** cannot support, for example, video, response director **210** sends an alternative Application to the handset, for example one that has a slide show instead. If the device cannot support a slide show, an Application might have text and images and display a message that indicates it does not support video.

Another powerful feature of response director **210** is its exposed API from the decision tree that permits the overriding of the default output of the decision tree by solution providers. These solution providers are often licensees who want to further refine the fulfillment of Applications and Players to specific devices beyond what the default algorithms provide. Solution providers may be given a choice of Applications and then can decide to use the defaults or force other Applications.

Authoring platform **110** automatically scales Applications at publishing time to various form factors to reduce the amount of fragmentation among devices, and the Response Director serves the appropriately scaled version to the device. For example, a QVGA Application will automatically scale to the QCIF form factor. This is important because one of the most visible forms of fragmentation resides in the various form factors of wireless, and particularly mobile, devices, which range from 128×128, 176×208, 240×260, 220×220, and many other customized sizes in between.

FIG. **13** is a schematic of an embodiment of a system **1300**. System **1300** is generally similar to system **1200**. System **1300** is an overview of the entire Player fulfillment process, starting with the generation of players during the player build process.

System **1300** includes response director **210**, a device characteristics operator and local database **1301**, a player profile database **1303** and a player build process **1305**, which may be authoring platform **110**.

As an example of system **1300**, when response director **210** receives an SMS message from device **130**, the response director identifies the device characteristics operator and locale from database **1301** and a Player URL from database **1303** and provides the appropriate Player to the device.

In another embodiment, Player P extend the power of response director **210** by adapting the Application to the resources and limitations of any particular device. Some of these areas of adaptation include the speed of the device's microprocessor, the presence of device resources such as cameras and touch screens. Another area of adaptation is directed to heap, record store and file system memory constraints. In one embodiment, the Player will automatically throttle down an animation to the frame rate that the device can handle so that the best possible user experience is preserved. Other extensions include device specific facilities such as location awareness, advanced touch screen interactions, push extensions, access to advanced phone facilities, and many others

Memory Management

In one embodiment, Player P includes a logical page virtual memory manager. This architecture requires no supporting hardware and works efficiently with constrained devices. All page view images, which could span multiple Applications, are placed in a table as highly compacted and compressed code. A typical page view will range from 500 bytes up to about 1,500 bytes. (See, for example, the Rempell patent) When rolled into the heap and instantiated this code increases to the more typical 50,000 up to 250,000 bytes. Additional alert pages may also be rolled into the heap and superimposed on the current page view. Any changes to any page currently downloaded are placed in a highly compact change vector for each page, and rolled out when the page is discarded. Note that whenever an Application is visited that had previously been placed in virtual memory the Server is interrogated to see if a more current version is available, and, if so, downloads it. This means that Application logic can be changed in real-time and the results immediately available to mobile devices.

To operate efficiently with the bandwidth constraints of mobile devices, authoring platform **110** may also utilize anticipatory streaming and multi-level caching. Anticipatory streaming includes multiple asynchronous threads and IO request queues. In this process, the current Application is scanned to determine if there is content that is likely to be required in as-yet untouched page views. Anticipatory streaming also looks for mapping Applications, where the user may zoom or pan next so that map content is retrieved prior to the user requesting it. For mapping applications, anticipatory streaming downloads a map whose size is greater than the map portal size on the device and centered within the portal. Any pan operation will anticipatory stream a section of the map to extend the view in the direction of the pan while, as a lower priority, bring down the next and prior zoom levels for this new geography. Zooming will always anticipatory stream the next zoom level up and down.

Multi-level caching determines the handset's heap through an API, and also looks at the record store to see how much memory is resident. This content is placed in record store and/or the file system, and may, if there is available heap, also place the content there as well. Multi-level caching permits the management of memory such that mobile systems best use limited memory resources. Multi-level caching is a memory management system with results similar to embedding, without the overhead of instantiating the content. In other words, with multi-level caching, handset users get an "embedded" performance without the embedded download. Note that when content is flagged as cacheable and is placed in persistent storage, a digital rights management (DRM) solution will be used.

One embodiment of each of the methods described herein is in the form of a computer program that executes on a processing system. Thus, as will be appreciated by those skilled in the art, embodiments of the present invention may be embodied as a method, an apparatus such as a special purpose apparatus, an apparatus such as a data processing system, or a carrier medium, e.g., a computer program product. The carrier medium carries one or more computer readable code segments for controlling a processing system to implement a method. Accordingly, aspects of the present invention may take the form of a method, an entirely hardware embodiment, an entirely software embodiment or an embodiment combining software and hardware aspects. Furthermore, the present invention may take the form of carrier medium (e.g., a computer program product on a computer-readable storage medium) carrying computer-readable program code segments embodied in the medium. Any suitable computer readable medium may be used including a magnetic storage device such as a diskette or a hard disk, or an optical storage device such as a CD-ROM.

It will be understood that the steps of methods discussed are performed in one embodiment by an appropriate processor (or processors) of a processing (i.e., computer) system executing instructions (code segments) stored in storage. It will also be understood that the invention is not limited to any particular implementation or programming technique and that the invention may be implemented using any appropriate techniques for implementing the functionality described herein. The invention is not limited to any particular programming language or operating system. It should thus be appreciated that although the coding for programming devices has not be discussed in detail, the invention is not limited to a specific coding method. Furthermore, the invention is not limited to any one type of network architecture and method of encapsulation, and thus may be utilized in conjunction with one or a combination of other network architectures/protocols.

Reference throughout this specification to "one embodiment," "an embodiment," or "certain embodiments" means that a particular feature, structure or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, appearances of the phrases "in one embodiment," "in an embodiment," or "in certain embodiments" in various places throughout this specification are not necessarily all referring to the same embodiment. Furthermore, the particular features, structures or characteristics may be combined in any suitable manner, as would be apparent to one of ordinary skill in the art from this disclosure, in one or more embodiments.

Throughout this specification, the term "comprising" shall be synonymous with "including," "containing," or "characterized by," is inclusive or open-ended and does not exclude additional, unrecited elements or method steps. "Comprising" is a term of art which means that the named elements are essential, but other elements may be added and still form a construct within the scope of the statement. "Comprising" leaves open for the inclusion of unspecified ingredients even in major amounts.

Similarly, it should be appreciated that in the above description of exemplary embodiments, various features of the invention are sometimes grouped together in a single embodiment, figure, or description thereof for the purpose of streamlining the disclosure and aiding in the understanding of one or more of the various inventive aspects. This method of disclosure, however, is not to be interpreted as reflecting an intention that the claimed invention requires more features than are expressly recited in each claim. Rather, as the following claims reflect, inventive aspects lie in less than all features of a single foregoing disclosed embodiment, and the invention may include any of the different combinations embodied herein. Thus, the following claims are hereby expressly incorporated into this Mode(s) for Carrying Out the Invention, with each claim standing on its own as a separate embodiment of this invention.

Thus, while there has been described what is believed to be the preferred embodiments of the invention, those skilled in the art will recognize that other and further modifications may be made thereto without departing from the spirit of the invention, and it is intended to claim all such changes and modifications as fall within the scope of the invention. For example, any formulas given above are merely representative of procedures that may be used. Functionality may be added or deleted from the block diagrams and operations may be interchanged among functional blocks. Steps may be added or deleted to methods described within the scope of the present invention.

We claim:

1. A system for generating code to provide content on a display of a device, said system comprising:

computer memory storing:

a) symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network, where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service, where each symbolic name has an associated data format class type corresponding to a subclass of User Interface (UI) objects that support the data format type of the symbolic name, and where each symbolic name has a preferred UI object, and

b) an address of the web service;

an authoring tool configured to:

define a UI object for presentation on the display,

where said defined UI object corresponds to a web component included in said computer memory selected from a group consisting of an input of the web service and an output of the web service, where each defined UI object is either:

1) selected by a user of the authoring tool; or

2) automatically selected by the system as the preferred UI object corresponding to the symbolic name of the web component selected by the user of the authoring tool,

access said computer memory to select the symbolic name corresponding to the web component of the defined UI object,

associate the selected symbolic name with the defined UI object, where the selected symbolic name is only available to UI objects that support the defined data format associated with that symbolic name,

store information representative of said defined UI object and related settings in a database;

retrieve said information representative of said one or more said UI object settings stored in said database; and

build an application consisting of one or more web page views from at least a portion of said database utilizing at least one player, where said player utilizes information stored in said database to generate for the display of at least a portion of said one or more web pages,

wherein when the application and player are provided to the device and executed on the device, and

when the user of the device provides one or more input values associated with an input symbolic name to an input of the defined UI object, the device provides the user provided one or more input values and corresponding input symbolic name to the web service, the web service utilizes the input symbolic name and the user provided one or more input values for generating one or more output values having an associated output symbolic name,

and the player receives the output symbolic name and corresponding one or more output values and provides instructions for the display of the device to present an output value in the defined UI object.

2. The system of claim **1**, where said system stores information in a registry, and wherein the registry includes definitions of input and output related to said web service.

3. The system of claim **1**, where said web component is a text chat, a video chat, an image, a slideshow, a video, or an RSS feed.

4. The system of claim **1**, where said UI object is an input field for a chat.

5. The system of claim **1**, where said UI object is an input field for a web service.

6. The system of claim **1**, where said UI object is an input field usable to obtain said web component, where said input field includes a text field, a scrolling text box, a check box, a drop down-menu, a list menu, or a submit button.

7. The system of claim **1**, where said web component is an output of a web service, is the text provided by one or more simultaneous chat sessions, is the video of a video chat session, is a video, an image, a slideshow, an RSS display, or an advertisement.

8. The system of claim **1**, where said authoring tool is further configured to:

define a phone field or list; and

generate code that, when executed on the device, allows a user to supply a phone number to said phone field or list.

9. The system of claim **1**, where said authoring tool is further configured to:

define a SMS field or list; and
generate code that, when executed on the device, allows a user to supply an SMS address to said SMS field or list.

10. The system of claim **1**,
where said code includes three or more codes, where one of said three or more codes is device specific, and where two of said three or more codes is device independent.

11. The system of claim **1**, where said code is provided over said network.

12. The system of claim **1**, wherein said defined UI object corresponds to a widget.

13. The system of claim **1**, where said player is activated and runs in a web browser.

14. The system of claim **1**, where said player is a native program.

15. A method of displaying content on a display of a device having a player and non-volatile computer memory storing symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network, where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service, where each symbolic name has an associated data format class type corresponding to a subclass of User Interface (UI) objects that support the data format type of the symbolic name, and where each symbolic name has a preferred UI object, and an address of the web service, said method comprising:
defining a UI object for presentation on the display, where said UI object corresponds to a web component included in the computer memory, where said web component is selected from a group consisting of an input of a web service and an output of the web service, where each defined UI object is either: 1) selected by a user of the authoring tool; or 2) automatically selected by the system as the preferred UI object corresponding to a symbolic name of the web component selected by the user of the authoring tool;
selecting the symbolic name corresponding to the web component of the defined UI object;
associating the selected symbolic name with the defined UI object, where the selected symbolic name is only available to UI objects that support the defined data format associated with that symbolic name;
storing information representative of said defined UI object and related settings in a database;
retrieving said information representative of said one or more said UI object settings stored in said database; and
building an application consisting of one or more web page views from at least a portion of said database utilizing the player, where said player utilizes information stored in said database to generate for the display of at least a portion of said one or more web pages,

wherein, when the application and player are provided to the device and executed on the device, and when the user of the device provides one or more input values associated with an input symbolic name to an input of the defined UI object, 1) the device provides the user provided one or more input values and corresponding input symbolic name to the web service, 2) the web service utilizes the input symbolic name and the user provided one or more input values for generating one or more output values having an associated output symbolic name, and 3) the player receives the output symbolic name and corresponding one or more output values and provides instructions for the display of the device to present an output value in the defined UI object.

16. The method of claim **15**, where said method stores information in a registry, and wherein the registry includes definitions of input and/or output related to said web service.

17. The method of claim **15**, where said web component is a text chat, a video chat, an image, a slideshow, a video, or an RSS feed.

18. The method of claim **15**, where said UI object is an input field for a chat.

19. The method of claim **15**, where said UI object is an input field for a web service.

20. The method of claim **15**, where said UI object is an input field usable to obtain said web component, where said input field includes a text field, a scrolling text box, a check box, a drop down-menu, a list menu, or a submit button.

21. The method of claim **15**, where said web component is an output of a web service, is the text provided by one or more simultaneous chat sessions, is the video of a video chat session, is a video, an image, a slideshow, an RSS display, or an advertisement.

22. The method of claim **15**, further comprising:
defining a phone field or list; and
generating code that, when executed on the device, allows a user to supply a phone number to said phone field or list.

23. The method of claim **15**, further comprising:
defining a SMS field or list; and
generating code that, when executed on the device, allows a user to supply an SMS address to said SMS field or list.

24. The method of claim **15**, and such that said player interprets dynamically received, device independent values of the web component defined in the application.

25. The method of claim **15**, further comprising:
providing said application and player over said network.

26. The method of claim **15**, wherein said UI object corresponds to a widget.

27. The method of claim **15**, where said player is activated and runs in a web browser.

28. The method of claim **15**, where said player is a native program.

* * * * *

EXHIBIT F

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

EXPRESS MOBILE, INC., Plaintiff, v. CODE AND THEORY LLC, Defendant.	Case No. 18-cv-04679-RS
EXPRESS MOBILE, INC., Plaintiff, v. PANTHEON SYSTEMS INC., Defendant.	Case No. 18-cv-04688-RS **ORDER DENYING MOTIONS TO DISMISS**

I. INTRODUCTION

These are two in a number of related patent cases involving allegations that the various defendants infringe U.S. Patent Nos. 6,546,397 (the ’397 patent) and 7,594,168 (the ’168 patent), which share a specification. Certain terms of the patents have been the subject of claim construction by this court in *X.Commerce, Inc. v. Express Mobile, Inc.,* Case No. 17-cv-02605-RS, a declaratory relief action where the patent holder, plaintiff Express Mobile in these cases, appears as the defendant. Claim construction has also taken place in at least one other district in other litigation brought by Express Mobile on the patents.

Defendants in these two actions move to dismiss contending the patent claims are drawn only to abstract ideas, ineligible for protection under Section 101 of the Patent Act, as elucidated in *Alice Corp. Pty. Ltd. v. CLS Bank Int'l*, 134 S. Ct. 2347 (2014) and its progeny. Because the patents purport to describe a novel technological approach to creating websites on the internet, defendants' characterization of the patents as claiming only an abstract idea fails, and the motions to dismiss must be denied.

II. BACKGROUND

As described in the patents' shared specification and explained in the briefing, Express Mobile contends the patents "bring together a number of disparate ideas and concepts, to create a new paradigm for creating, storing, and building web pages." According to Express Mobile, prior to the invention of the patents, web pages were created, stored and rendered in a fundamentally different manner. Individual web pages were typically created by either programming directly in HTML or JavaScript code, or by using a visual editor that output HTML formatted files. These approaches allegedly were cumbersome and inflexible, in various respects.

The inventive methodology purportedly described in the patents involves building a web page by defining it as a set of user-selected "objects" and/or "settings." The result is not a markup language code file for the web page, but instead a collection of user selected objects and object attributes. These can be saved in a database, for ease of access and efficient storage. Express Mobile explains that because complete code files for each page do not need to be stored, the page structure—the full HTML code itself—is created on the fly each time the page is loaded in a user's Web browser. This is achieved in part through a browser-appropriate "run time engine" and related files.

Defendants contend claim 1 of the '397 patent is representative of both patents for purposes of *Alice* analysis.[1] It provides:

[1] Express Mobile asserts claim 1 is not representative, and discusses at least one other claim (Claim 1 of the '168) patent, but it has not shown the result would be any different for any of the

> A method to allow users to produce Internet websites on and for computers having a browser and a virtual machine capable of generating displays, said method comprising:
>
> (a) presenting a viewable menu having a user selectable panel of settings describing elements on a website, said panel of setting being presented through a browser on a computer adapted to accept one or more of said selectable settings in said panel as inputs therefrom, and where at least one of said user selectable settings in said panel corresponds to commands in said virtual machine;
>
> (b) generating a display in accordance with one or more user selected settings substantially contemporaneously with the selection thereof;
>
> (c) storing information representative of said one or more user selected settings in a database;
>
> (d) generating a website at least in part by retrieving said information representative of said one or more user selected settings stored in said database; and
>
> (e) building one or more webpages to generate said website from at least a portion of said database and at least one run time file, where said at least one run time file utilizes information stored in said database to generate virtual machine commands for the display of at least a portion of said one or more web pages.

III. DISCUSSION

As explained in *Alice*, the Supreme Court has "interpreted § 101 and its predecessors ... for more than 150 years" to "'contain[ ] an important implicit exception: Laws of nature, natural phenomena, and abstract ideas are not patentable.' " The *Alice* court applied a two-step framework for determining patent eligibility, previously articulated in *Mayo Collaborative Servs. v. Prometheus Labs.,* Inc., 132 S.Ct. 1289 (2012):

> First, we determine whether the claims at issue are directed to one of those patent-ineligible concepts. If so, we then ask, "[w]hat else is

---

other asserted claims.

> there in the claims before us?" To answer that question, we consider the elements of each claim both individually and "as an ordered combination" to determine whether the additional elements "transform the nature of the claim" into a patent-eligible application. We have described step two of this analysis as a search for an "inventive concept"—i.e., an element or combination of elements that is sufficient to ensure that the patent in practice amounts to significantly more than a patent upon the [ineligible concept] itself.

Alice, 134 S.Ct. at 2355.

Alice also explained, "The 'abstract ideas' category embodies "the longstanding rule that '[a]n idea of itself is not patentable.' " *Id*. at 2355; *see also Le Roy v. Tatham*, 14 How. 156, 175, 14 L.Ed. 367 (1853). ("A principle, in the abstract, is a fundamental truth; an original cause; a motive; these cannot be patented, as no one can claim in either of them an exclusive right").

Alice repeated the caution given in *Mayo*, however, that the exclusion for "abstract ideas" must not be applied too broadly, "we tread carefully in construing this exclusionary principle lest it swallow all of patent law." 134 S.Ct. at 2354 (*citing Mayo*, 132 S.Ct. at 1293–1294.) At some level, "all inventions . . . embody, use, reflect, rest upon, or apply laws of nature, natural phenomena, or abstract ideas." *Mayo*, 132 S.Ct. at 1293.

On the facts before it, the *Alice* court also expressly declined to "labor to delimit the precise contours of the 'abstract ideas' category." 134 S.Ct. at 2357. Instead, it merely found that the concept of providing an "intermediated settlement" was not meaningfully distinguishable from the idea of "risk hedging" at issue in *Bilski v. Kappos*, 561 U.S. 593 (2010). In both instances, the idea involved was "a fundamental economic practice long prevalent in our system of commerce." *Id*. at 2356.

Here, defendants rely primarily on *Intellectual Ventures I LLC v. Capital One Bank (USA)*, 792 F.3d 1363 (Fed. Cir. 2015). They argue the patents here are not meaningfully distinguishable from one held invalid in that case, U.S. Patent No. 7,603,382, entitled "Advanced Internet Interface Providing User Display Access of Customized Webpages." Defendants' insistence to the contrary notwithstanding, the patents are simply not directly comparable. The '382 patent in *Intellectual Ventures* "generally relate[d] to customizing web page content as a function of navigation history and information known about the user." 792 F.3d at 1369. The representative claim described "[a] system for providing web pages accessed from a web site in a manner which presents the web pages tailored to an individual user." The *Intellectual Ventures* court had little trouble concluding that merely tailoring the information presented to a website user based on information about that user or when the website was being viewed represented patent-ineligible abstract ideas.

> This sort of information tailoring is "a fundamental . . . practice long prevalent in our system. . . ." . . . There is no dispute that newspaper inserts had often been tailored based on information known about the customer—for example, a newspaper might advertise based on the customer's location. Providing this minimal tailoring—e.g., providing different newspaper inserts based upon the location of the individual—is an abstract idea . . . .
>
> Tailoring information based on the time of day of viewing is also an abstract, overly broad concept long-practiced in our society. There can be no doubt that television commercials for decades tailored advertisements based on the time of day during which the advertisement was viewed.

Id.

The patents here are not comparable merely because they also involve webpages that reflect information provided by a "user." Indeed, the patents do not even involve the same category of "user"—here the "user" is the person who is trying to *create* webpages, in *Intellectual Ventures* the user is a person viewing the webpage to whom customized content will be delivered. The patents here are directed at a purportedly revolutionary technological solution to a technological problem—how to create webpages for the internet in a manner that permits "what you see is what you get" editing, and a number of other alleged improvements over the then-existing methodologies.

A more apt comparison is *Enfish, LLC v. Microsoft Corp*., 822 F.3d 1327 (Fed. Cir. 2016), which reversed a district court's finding of ineligibility under *Alice.* At issue in *Enfish* was "an innovative logical model for a computer database." *Enfish* supports the notion that a dividing line can be drawn between patents which merely describe using a computer and/or the internet to carry out pre-existing and well-known tasks and techniques, and those that relate to the functioning of computers themselves. The former will virtually always fail under *Alice* unless some "inventive concept" can be found in the second step of the analysis; the latter are substantially less easily characterized as merely abstract ideas.

Enfish explains:

> The first step in the *Alice* inquiry in this case asks whether the focus of the claims is on the specific asserted improvement in computer capabilities (*i.e.*, the self-referential table for a computer database) or, instead, on a process that qualifies as an "abstract idea" for which computers are invoked merely as a tool. As noted *infra*, in *Bilski* and *Alice* and virtually all of the computer-related § 101 cases we have issued in light of those Supreme Court decisions, it was clear that the claims were of the latter type—requiring that the analysis proceed to the second step of the *Alice* inquiry, which asks if nevertheless there is some inventive concept in the application of the abstract idea. *See Alice*, 134 S.Ct. at 2355, 2357–59. In this case, however, the plain focus of the claims is on an improvement to computer functionality itself, not on economic or other tasks for which a computer is used in its ordinary capacity.

822 F.3d at 1335–36.

Enfish drew a line between "improvement[s] to computer functionality itself," and "economic or other tasks for which a computer is used in its ordinary capacity." The court concluded:

> we find that the claims at issue in this appeal are not directed to an abstract idea within the meaning of *Alice*. Rather, they are directed to a specific improvement to the way computers operate . . . ."

Id. at 1336.

So too here.

Finally, to the extent that defendants are arguing that any potentially patent-eligible technological improvements set out in the specification are not reflected in the actual claims, dismissal under *Alice* is not appropriate, at least at this juncture. Although some claim construction has taken place, it simply cannot be said on the present record that the claims are drawn so broadly as to be divorced from the potentially patent-eligible purported technological improvements described in the specification. Accordingly, the motions to dismiss are denied.

IT IS SO ORDERED.

Dated: January 29, 2019

[illegible]

RICHARD SEEBORG
United States District Judge

EXHIBIT G

IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

Express Mobile, Inc,	§	
	§	
Plaintiff,	§	
	§	
v.	§	Case No. 2:17-cv-128-JRG-RSP
	§	
KTree Computer Solutions Inc.,	§	
	§	
Defendant.	§	
	§	
	§	

REPORT AND RECOMMENDATION

Before the Court is Defendant's Motion for Judgment on the Pleadings (Dkt. No. 9) (the "Motion"). The Court recommends the Motion be **DENIED WITHOUT PREJUDICE**.

First, the claims appear to address a problem particular to the internet: dynamically generating websites and displaying web pages based on stored user-selected settings. *See DDR Holdings, LLC v. Hotels.com*, L.P., 773 F.3d 1245, 1257 (Fed. Cir. 2014). On the face of the asserted patents, the claims appear to be necessarily rooted in computer technology and directed to specific problems of and improvements to then-existing web publishing applications. *See, e.g.*, Dkt. No. 22-1 at 1:6–8, 1:22–24 ("As such it is virtually impossible to write a web publishing application in HTML and JavaScript. All conventional implementations must, and do, utilize a full-featured programming language, such as C++ or Visual Basic."); 1:36–39 ("As such, a conventional web publishing application written in one of these languages suffers from the severe performance problems inherent in these languages."); 2:60–64 ("Because of the implementation of a variety of performance and file reduction techniques, the entire run time environment can range from as low as 12K, and no larger than

5OK."); 3:4–6 ("The present invention provides a real time, dynamic linkage between JAVA and HTML including two-way communications, in real time, between JAVA and JavaScript.").

Second, the asserted claims do not bear all of the hallmarks of claims that have been invalidated on the pleadings by other courts in the past. For example, the claims are not merely do-it-on-a-computer claims. *See Alice Corp. Pty. v. CLS Bank Int'l*, 134 S. Ct. 2347, 2358, 189 L. Ed. 2d 296 (2014). The Court has also reviewed *Internet Patents Corp. v. Active Network, Inc.*, 790 F.3d 1343 (Fed. Cir. 2015), cited by Defendant, and is not persuaded that the case is directly on point. For one, it is unclear from this record whether a run time file or run time engine as claimed in the asserted patents is analogous to the "template file" recited in the *Internet Patents* claims. The Court also does not have a firm conviction at this early stage that these claims contain no inventive concept.

Thus, the Court is not persuaded that claim construction and a fuller factual record would be unhelpful to the patent eligibility determination. *See Autumn Cloud LLC v. TripAdvisor, Inc.*, Dkt. No. 66 at 2–4, Case No. 2:16-cv-853-JRG-RSP (E.D. Tex. April 3, 2017).

Accordingly, the Court recommends the Motion be **DENIED WITHOUT PREJUDICE.**[1]

SIGNED this 4th day of May, 2017.

Roy S. Payne

ROY S. PAYNE
UNITED STATES MAGISTRATE JUDGE

[1] A party's failure to file written objections to the findings, conclusions, and recommendations contained in this report within fourteen days after being served with a copy shall bar that party from de novo review by the district judge of those findings, conclusions, and recommendations and, except on grounds of plain error, from appellate review of unobjected-to-factual findings and legal conclusions accepted and adopted by the district court. Fed. R. Civ. P. 72(b)(2); *see Douglass v. United Servs. Auto. Ass'n*, 79 F.3d 1415, 1430 (5th Cir. 1996) (en banc).

EXHIBIT H

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

EXPRESS MOBILE, INC.,

Plaintiff,

v.

DREAMHOST LLC,

Defendant.

Civil Action No. 1:18-cv-01173-RGA

EXPRESS MOBILE, INC.,

Plaintiff,

v.

HOSTWAY SERVICES, INC.,

Defendant.

Civil Action No. 1:18-cv-01175-RGA

MEMORANDUM ORDER

Presently before me are Defendants' Motions to Dismiss. (C.A. 18-1173, D.I. 13; C.A. 18-1175, D.I. 13). The Parties have briefed the issues. (C.A. 18-1173, D.I. 14, 18, 24; C.A. 18-1175, D. I. 14, 17, 23). For the reasons set out below, I will grant Defendants' motions as to past damages and willful infringement. I will deny Defendant Hostway Service Inc.'s motion as to direct infringement and Defendants' motions as to patent eligibility.

Plaintiff filed these lawsuits on August 4, 2018. It alleges that Defendants infringe U.S. Patent Nos. 6,546,397 ("'397 Patent"), 7,594,168 ("'168 Patent), 9,471,287, and 9,928,044 through their use of certain website building tools such as Wordpress or Joomla. (*See* C.A. 18-1173, D.I. 10; C.A. 18-1175, D.I. 9).

When reviewing a motion to dismiss pursuant to Rule 12(b)(6), the court must accept the complaint's factual allegations as true. *See Bell Atl. Corp. v. Twombly*, 550 U.S. 544, 555-56 (2007). Rule 8(a) requires "a short and plain statement of the claim showing that the pleader is entitled to relief." *Id.* at 555. The factual allegations do not have to be detailed, but they must provide more than labels, conclusions, or a "formulaic recitation" of the claim elements. *Id.* ("Factual allegations must be enough to raise a right to relief above the speculative level . . . on the assumption that all the allegations in the complaint are true (even if doubtful in fact)."). Moreover, there must be sufficient factual matter to state a facially plausible claim to relief. *Ashcroft v. Iqbal*, 556 U.S. 662, 678 (2009). The facial plausibility standard is satisfied when the complaint's factual content "allows the court to draw the reasonable inference that the defendant is liable for the misconduct alleged." *Id.* ("Where a complaint pleads facts that are merely consistent with a defendant's liability, it stops short of the line between possibility and plausibility of entitlement to relief." (internal quotation marks omitted)).

To satisfy the *Iqbal* pleading standard in a patent case, "[s]pecific facts are not necessary." *Disc Disease Solutions Inc. v. VGH Solutions, Inc.*, 888 F.3d 1256, 1260 (Fed. Cir. 2018) (quoting *Erickson v. Pardus*, 551 U.S. 89, 93 (2007)). The Complaint need only give defendant "fair notice of what the [infringement] claim is and the ground upon which it rests." *Id.*

Defendant Hostway Services, Inc. ("Hostway") argues that the direct infringement claims of the 67-page First Amended Complaint should be dismissed for failure to meet the pleading standard. (C.A. 18-1175, D.I. 14 at 7-10). In three related cases, I have found that Plaintiff's essentially identical complaints are sufficient. (*See* C.A. 18-1176, D.I. 19; C.A. 18-1177, D.I. 26; C.A. 18-1181, D.I. 39). Hostway's argument for dismissal of Plaintiff's direct infringement claims are largely identical to those made in the related cases. (*Compare* C.A. 18-1175, D.I. 14 at 7-10, *with* C.A. 18-1177, D.I. 14 at 4-6, *and* C.A. 18-1181, D.I. 28 at 5-7). Accordingly, I will deny Hostway's motion to dismiss Plaintiff's direct infringement claims.[1]

Defendants next argue that Plaintiff's willfulness claims should be dismissed. I agree. "[T]o state a claim of willful infringement, the patentee must allege facts in its pleading plausibly demonstrating that the accused infringer had committed subjective willful infringement as of the date of the filing of the willful infringement claim." *Välinge Innovation AB v. Halstead New England Corp.*, 2018 WL 2411218, at *10-12 (D. Del. May 29, 2018) (discussing *Mentor Graphics Corp. v. EVE-USA, Inc.*, 851 F.3d 1275, 1282 (Fed. Cir. 2017), *cert. dismissed*, No. 17-804, 2018 WL 3978434 (U.S. Aug. 17, 2018)). The complaints allege only post-filing knowledge of the alleged infringement. (C.A. 18-1173, D.I. 10 at ¶¶ 54, 72, 143, 210; C.A. 18-1175, D.I. 9 at ¶¶ 54, 72, 143, 210). Thus, they fail to meet the pleading standard for willful infringement, which requires allegations of willful conduct prior to the filing of the claim. I will grant Defendants' motion to dismiss Plaintiff's willful infringement claims. I will, however, also grant Plaintiff's request for leave to amend the complaints to plead post-filing conduct. (*See* C.A. 18-1175, D.I. 17 at 8).

Defendants separately note that Plaintiff has failed to plead compliance with the marking statute. (C.A. 18-1173, D.I. 14 at 16-19; C.A. 18-1175, D.I. 14 at 13-14). They argue that this is a basis to dismiss Plaintiff's claims to the extent that they seek past damages. 35 U.S.C. § 287(a) requires that a patentee who makes or sells a patented article mark the articles to recover past damages. It is the patentee's burden to plead compliance with § 287(a). *Arctic Cat Inc. v.*

[1] Plaintiff improperly included a screenshot of a content management service detection website, https://builtwith.com, as evidence that Hostway uses WordPress. (C.A. 18-1175, D.I. 17 at 4). I have not considered this new evidence in resolving this motion.

Bombardier Recreational Prod. Inc., 876 F.3d 1350, 1366 (Fed. Cir. 2017). Plaintiff avers that there is no evidence that there was anything for it to mark but does not argue that it pled compliance. (*See* C.A. 18-1173, D.I. 16 at 19-20; C.A. 18-1175, D.I. 17 at 5-7). At the motion to dismiss stage, I am only concerned with the sufficiency of the claims. A claim for past damages requires pleading compliance with the marking statute—even when compliance is achieved, factually, by doing nothing at all. Thus, as Plaintiff has failed to state a claim for past damages, I will grant Defendants' motions.

Defendants also argue for dismissal of half of Plaintiff's claims on the basis that the claims of the '397 and '168 Patents are invalid under Section 101. (C.A. 18-1173, D.I. 14 at 5-15; *see also* C.A. 18-1175, D.I. 14 at 14 (incorporating Section 101 argument)). Patentability under 35 U.S.C. § 101 is a threshold legal issue. *Bilski*, 561 U.S. at 602. Accordingly, the § 101 inquiry is properly raised at the pleading stage if it is apparent from the face of the patent that the asserted claims are not directed to eligible subject matter. *See Cleveland Clinic Found. v. True Health Diagnostics LLC*, 859 F.3d 1352, 1360 (Fed. Cir. 2017), *cert. denied*, 138 S. Ct. 2621 (2018). This is, however, appropriate "only when there are no factual allegations that, taken as true, prevent resolving the eligibility question as a matter of law." *Aatrix Software, Inc. v. Green Shades Software, Inc.*, 882 F.3d 1121, 1125 (Fed. Cir. 2018). In response to Defendants' argument, Plaintiff identified factual allegations of inventiveness in the complaint and submitted an expert declaration explaining inventiveness of the claims. (*See* C.A. 18-1173, D.I. 10 at ¶¶ 12, 13, 60, 61; C.A. 18-1173, D.I. 20 (expert declaration discussing inventiveness of the claims)). I find that these factual issues preclude a finding of invalidity on a motion to dismiss. Thus, I will deny Defendants' Section 101 motion without prejudice to Defendants raising the issue again on summary judgment.

Defendants' Motions to Dismiss (C.A. 18-1173, D.I. 13; C.A. 18-1175, D.I. 13) are **GRANTED-IN-PART** and **DENIED-IN-PART**. Plaintiff's claims for past damages and willful infringement are dismissed. Plaintiff may file an amended complaint within 14 days of the entry of this order.

Entered this 18 day of June 2019.

Richard G. Andrews
United States District Judge

EXHIBIT I

STEVEN J. RIZZI (*pro hac vice*)
srizzi@kslaw.com
RAMY HANNA (*pro hac vice* forthcoming)
rhanna@kslaw.com
KING & SPALDING LLP
1185 Avenue of the Americas, 35th Floor
New York, NY 10036
Telephone: (212) 556-2100
Facsimile: (212) 556-2222

RYAN A. SCHMID (*pro hac vice* forthcoming)
rschmid@kslaw.com
KING & SPALDING LLP
1700 Pennsylvania Avenue, N.W.
Washington, D.C. 20006
Telephone: (202) 737-0500
Facsimile: (202) 626-3737

RAMON A. MIYAR (CA SBN 284990)
rmiyar@kslaw.com
KING & SPALDING LLP
50 California Street, Suite 3300
San Francisco, CA 94111
Telephone: (415) 318-1200
Facsimile: (415) 318-1300

Attorneys for Plaintiff
EXPRESS MOBILE, INC.

[*Additional Counsel Listed on Signature Page*]

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

EXPRESS MOBILE, INC.

Plaintiff,

v.

BOOKING HOLDINGS, INC.

Defendants.

Civil Action No. 3:20-cv-08491-RS

JOINT STIPULATION REGARDING FILING OF AMENDED COMPLAINT AND WAIVER OF SERVICE OF SUMMONS

JOINT STIPULATION

Plaintiff Express Mobile, Inc. ("Plaintiff") and Defendant Booking Holdings, Inc. ("Defendant" or "Booking Holdings"), by and through their respective counsel, hereby stipulate as follows:

RECITALS

WHEREAS, Plaintiff filed its Complaint against Defendants on December 1, 2020;

WHEREAS, service of process on Booking Holdings' registered agent for service of process was effectuated on December 3, 2020;

WHEREAS, following the appearance of Booking Holdings' former counsel, Fish & Richardson LLP, the Parties stipulated to continue Booking Holdings' deadline to respond to the Complaint to January 27, 2021 (ECF No. 13);

WHEREAS, since filing that stipulation, Booking Holdings retained new counsel and is now represented in this action by Baker Botts LLP;

WHEREAS, the Parties stipulated to a further extension of Booking Holdings' responsive pleading deadline to February 11, 2021, in order to accommodate its change in counsel (ECF No. 18);

WHEREAS, Booking Holdings represents that it is a holding company that does not presently—and did not at any point—own, operate or control the allegedly infringing www.booking.com, www.agoda.com, and www.priceline.com websites accused of infringement in the Complaint ("Accused Instrumentalities") (with the exception of any indirect ownership interest(s) it may have by virtue of its status as an ultimate parent company of the Booking Subsidiaries);

WHEREAS, Booking Holdings represents that its subsidiaries: (1) Agoda Company Pte. Ltd.; (2) Booking.com B.V.; and (3) priceline.com LLC (collectively, the "Booking Subsidiaries") possess relevant and discoverable information pertaining to the Accused Instrumentalities;

WHEREAS, counsel for Booking Holdings represents that it is also counsel for the

Case 3:20-cv-08491-RS Document 26 Filed 02/26/21 Page 5 of 6

Booking Subsidiaries for the purposes of this action, and that it is duly authorized to act on their behalf and bind them to this Stipulation;

WHEREAS, Booking Holdings represents that the website at www.booking.com referenced in Plaintiff's Complaint is developed and owned by Booking.com B.V., a Netherlands company, which is wholly owned by Booking Holdings;

WHEREAS, Booking Holdings represents that the website at www.agoda.com referenced in Plaintiff's Complaint is developed and owned by Agoda Company Pte. Ltd., a Singapore company, which is wholly owned by Booking Holdings;

WHEREAS, Booking Holdings represents that the website at www.priceline.com referenced in Plaintiff's Complaint is developed and owned by priceline.com LLC, which is wholly owned by Booking Holdings;

WHEREAS, Booking Holdings represents that it is a holding company, which is a separate corporate entity from the Booking Subsidiaries, and that it has no discoverable information in its possession, custody, and/or control related to the Accused Instrumentalities beyond that which is available from the Booking Subsidiaries;

WHEREAS, Booking Holdings represents that it (a) has no ownership interest in the Accused Instrumentalities (with the exception of any indirect ownership interest(s) it may have by virtue of its status as an ultimate parent company of the Booking Subsidiaries), (b) does not operate or control the Accused Instrumentalities, and (c) would not bear financial or legal responsibility for satisfying any judgment against the Booking Subsidiaries on any of the causes of actions asserted against them in the Complaint;

WHEREAS, each of the Booking Subsidiaries stipulates and represents that it will not use the absence of Booking Holdings as a basis to limit or withhold discovery in this matter concerning the Accused Instrumentalities;

WHEREAS, the Parties wish to avoid the burden and expense of motion practice;

WHEREAS, in the interests of judicial economy and efficiency, the Parties stipulate to Plaintiff's filing of its First Amended Complaint, removing Booking Holdings as a defendant, and substituting the Booking Subsidiaries as defendants to this action and allegations related to those entities;

WHEREAS, Plaintiff agrees to withdraw all outstanding discovery requests against Booking Holdings;

WHEREAS, Booking.com B.V. is headquartered and incorporated under the laws of the Kingdom of the Netherlands and Agoda Company Pte. Ltd. is headquartered and incorporated under the laws of the Republic of Singapore;

WHEREAS, the Bookings Subsidiaries, by and through their counsel, agree to waive service of process under Rule 4(d) of the Federal Rule of Civil Procedure ("FRCP"), provided that all three of the Booking Subsidiaries are provided 60 days to answer or otherwise plead in response to the First Amended Complaint;

NOW, THEREFORE, the Parties hereby stipulate and as follows:

1. Pursuant to FRCP 15(a), the Parties agree that Plaintiff shall be permitted to file a First Amended Complaint;

2. Each of (i) Agoda Company Pte. Ltd., (ii) Booking.com B.V., and (iii) priceline.com LLC shall execute and provide to Plaintiff's counsel, within 7 days of the execution of this Stipulation, a waiver of service of the Summons in the form prescribed by the Court;

3. Each of (i) Agoda Company Pte. Ltd., (ii) Booking.com B.V., and (iii) priceline.com LLC consents to the filing of the First Amended Complaint, and further consents to and waives any objection to venue in the U.S. District Court for the Northern District of California; provided, however, that each reserves any and all rights to raise specific and/or affirmative defenses other than those related to venue;

4. Booking Holdings shall be contemporaneously dismissed without prejudice; and

5. The language of the waiver of the service of summons form and FRCP 4(d) notwithstanding, the Booking Subsidiaries' deadline to file an answer or other pleading or motion in response to the Complaint shall be 60 days after the date on which Plaintiff files its First Amended Complaint via the Court's ECF system. Agoda Company Pte. Ltd. and Booking.com

B.V. expressly waive the 90-day answer/responsive pleading period provided to defendants who are served in a judicial district outside the United States under FRCP 4(d)(3).

6. In light of the above, the Parties agree to jointly seek (through a separate stipulation) a fourteen-day continuance of the Initial Case Management Conference presently scheduled for March 4, 2021 to March 18, 2021.

IT IS SO STIPULATED.

Dated: February 26, 2021

KING & SPALDING LLP

/s/ Ramon A. Miyar

Steven J. Rizzi (*pro hac vice*)
Ramy Hanna (*pro hac vice* forthcoming)
Ryan A. Schmid (*pro hac vice* forthcoming)
Ramon A. Miyar

Attorneys for Plaintiff Express Mobile, Inc.

Dated: February 26, 2021

BAKER BOTTS LLP

/s/ Jeremy J. Taylor

JEREMY J. TAYLOR
101 California Street, ste. 3600
San Francisco, CA 94111
Tel.: (415) 291-6202
Facsimile: (415) 291-6302

Attorneys for Defendants Booking Holdings, Inc.; Agoda Company Pte. Ltd.; Booking.com B.V.; and priceline.com LLC

Case 3:20-cv-08491-RS Document 26 Filed 02/26/21 Page 548 of 548

ATTESTATION

Pursuant to Civil Local Rule 5-1(i)(3), I hereby attest that all signatories to this document concur in its filing.

Dated: February 26, 2021

/s/ Ramon A. Miyar
Ramon A. Miyar