(12) STANDARD PATENT (11) Application No. **AU 2006291331 B2**
(19) AUSTRALIAN PATENT OFFICE

(54) Title
Script markup

(51) International Patent Classification(s)
G06F 17/00 (2006.01) ***G06F 17/21*** (2006.01)
G06F 3/14 (2006.01)

(21) Application No: **2006291331** (22) Date of Filing: **2006.08.29**

(87) WIPO No: **WO07/032925**

(30) Priority Data

(31) Number	(32) Date	(33) Country
60/716,293	**2005.09.12**	**US**
11/318,305	**2005.12.23**	**US**

(43) Publication Date: **2007.03.22**
(44) Accepted Journal Date: **2011.03.31**

(71) Applicant(s)
Microsoft Corporation

(72) Inventor(s)
Kothari, Nikhil;Le Roy, Bertrand

(74) Agent / Attorney
Davies Collison Cave, 1 Nicholson Street, Melbourne, VIC, 3000

(56) Related Art
US 6748569 B1
WO 2001/042977 A1

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

PCT

(43) International Publication Date
22 March 2007 (22.03.2007)

(10) International Publication Number
WO 2007/032925 A1

(51) International Patent Classification:
G06F 17/00 (2006.01) *G06F 3/14* (2006.01)
G06F 17/21 (2006.01)

(21) International Application Number: PCT/US2006/034121

(22) International Filing Date: 29 August 2006 (29.08.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/716,293 12 September 2005 (12.09.2005) US
11/318,305 23 December 2005 (23.12.2005) US

(71) Applicant *(for all designated States except US)*: **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) Inventors: **KOTHARI, Nikhil**; One Microsoft Way, Redmond, Washington 98052-6399 (US). **LE ROY, Bertrand**; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

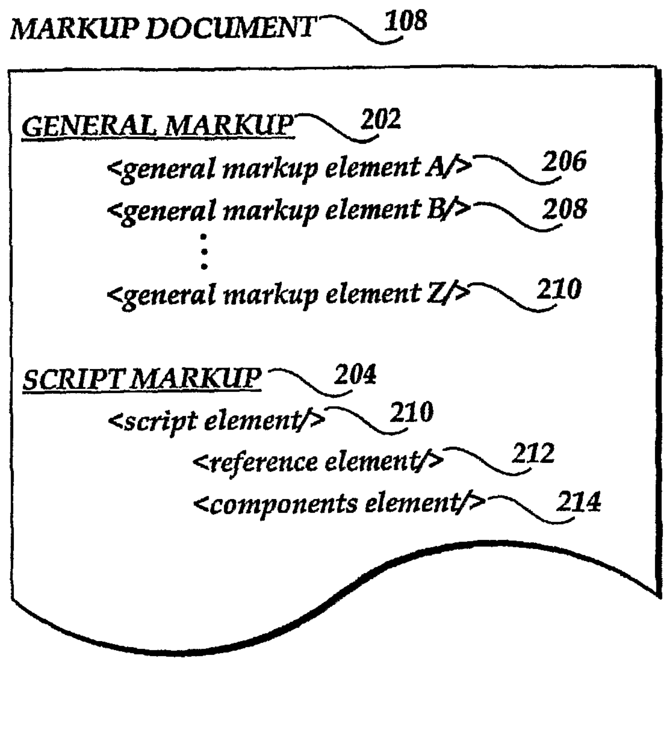
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:
— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*
— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SCRIPT MARKUP


MARKUP DOCUMENT 108
GENERAL MARKUP 202
<general markup element A/> 206
<general markup element B/> 208
<general markup element Z/> 210
SCRIPT MARKUP 204
<script element/> 210
<reference element/> 212
<components element/> 214

(57) **Abstract:** A script markup language provides a declarative mechanism for defining script based interactive behavior and application logic associated with a document. The script markup defining the interactive behavior and application logic is presented as an independent portion of the markup for the document, separated from any markup concerning the content and presentation of the document.

WO 2007/032925 A1

SCRIPT MARKUP

BACKGROUND

Historically, markup was used to refer to the process of marking manuscript copy for typesetting with directions for formatting such as use of type fonts and sizes, spacing, indentation, etc. In today's digital age, markup refers to electronic markup, i.e., the internal and sometimes invisible codes in an electronic document that describe the formatting of the document. Generally, a user can view the markup of an electronic document by looking at the source code of the document with the browser displaying the electronic document. The electronic markup of a document generally provides encoding of text as well as details about the structure, appearance and presentation of the text and content in the document.

The markup of an electronic document usually is programmed using a markup language. A markup language provides syntax and procedures for embedding in a document tags that control the formatting of the text when the document is viewed by a special application such as a Web browser. Commonly used electronic markup languages include HTML, XML, and ASP.NET. Traditionally, markup languages are used to design the content and appearance of a static document.

However, for an interactive application such as a Web application, the content and/or presentation of a document such as a Web page may change, for example, based on user input. The markup of the document thus needs to be accompanied by information governing the behavior of the document. Traditionally, document behavior has been implemented procedurally in a script. To provide dynamic document behavior, a markup of the document may call on methods in the script at the appropriate time. The intermingling of markup and calls to script methods thus makes it difficult to independently design the markup for a document. Meanwhile, because a script language traditionally has been procedural and imperative, a user of a document usually cannot use the script language to design a specific behavior for the document.

2006291331 23 Feb 2011

It is desired, therefore, to provide a computer program product, or a method which facilitates designing the markup for an electronic document that alleviate one or more of the above difficulties, or at least provide a useful alternative.

SUMMARY

In accordance with the present invention, there is provided a computer program product comprising a computer storage medium containing computer-executable instructions for implementing a method which facilitates designing the markup for an electronic document such as a Web page which is stored as a markup document in a database associated with a server computing system, and which is later retrieved for viewing and design of the markup document at a browser of a client computing system, and wherein the method facilitates the design by separating script markup language that defines interactive behavior and application logic associated with the electronic document from general markup language of the electronic document that defines content and presentation of the electronic document, the method comprising:

receiving input at a type manager associated with a browser for registering a custom script object model, the custom script object model containing one or more user defined attributes comprising properties, methods, or event attributes for use by the browser in interpreting script objects that conform to the custom script object model and that are contained within electronic documents that are executed by the browser;

registering the custom script object model with the browser to enable the browser to interpret script objects that conform to the custom script object model;

retrieving from a database associated with a server computing system an electronic document stored in the form of a markup document, for display at the browser of a client computing system, the retrieved electronic document having a markup language that defines in a first part of the retrieved electronic document a general markup portion comprised of one or more general markup elements that define formatting of the content and/or the overall appearance of the electronic document when displayed on a Web page, the retrieved electronic document having a markup language that defines in a second part of the same retrieved electronic document a script markup portion comprised of a reference element and a components element, wherein the reference element references script files external to the retrieved electronic document, and wherein the components element contains one or more script objects for implementing interactive behavior and application logic associated with the electronic document when displayed as a Web page, wherein at least one of the script objects is a custom script object that conforms to the custom script object model that was registered with the browser;

upon retrieving the electronic document, processing the components element to instantiate the one or more script objects, including accessing the custom script object model to determine how to instantiate the at least one script object that conforms to the custom script object model registered with the browser;

wherein one or more of the script markup elements of the script markup portion reference at least one general markup element contained in the general markup portion of the retrieved electronic document, but script elements of the script markup portion are not referenced by any of the general markup elements of the general markup portion of the retrieved electronic document so that the script portion of the retrieved document is kept separate from the general markup portion when presented for viewing and design on a browser of a client computing system;

presenting the retrieved document for display at the browser of the client computing system with the separate general and script markup portions;

receiving user input that interacts with a portion of the displayed document that is represented by one or more general markup elements, and in response, accessing the custom script object model to perform functionality defined by one or more attributes associated with a custom script object that references the one or more general markup elements; and

performing the functionality to modify the appearance of the portion of the displayed document that is represented by the one or more general markup elements.

The present invention also provides a method which facilitates designing the markup for an electronic document such as a Web page which is stored as a markup document in a database associated with a server computing system, and which is later retrieved for viewing and design of the markup document at a browser of a client computing system, and wherein the method facilitates the design by separating script markup language that defines interactive behavior and application logic associated with the

2006291331 23 Feb 2011

electronic document from general markup language of the electronic document that defines content and presentation of the electronic document, the method comprising:

receiving input at a type manager associated with a browser for registering a custom script object model, the custom script object model containing one or more user defined attributes comprising properties, methods, or event attributes for use by the browser in interpreting script objects that conform to the custom script object model and that are contained within electronic documents that are executed by the browser;

registering the custom script object model with the browser to enable the browser to interpret script objects that conform to the custom script object model;

retrieving from a database associated with a server computing system an electronic document stored in the form of a markup document, for display at the browser of a client computing system, the retrieved electronic document having a markup language that defines in a first part of the retrieved electronic document a general markup portion comprised of one or more general markup elements that define formatting of the content and/or the overall appearance of the electronic document when displayed on a Web page, the retrieved electronic document having a markup language that defines in a second part of the same retrieved electronic document a script markup portion comprised of a reference element and a components element, wherein the reference element references script files external to the retrieved electronic document, and wherein the components element contains one or more script objects for implementing interactive behavior and application logic associated with the electronic document when displayed as a Web page, wherein at least one of the script objects is a custom script object that conforms to the custom script object model that was registered with the browser;

upon retrieving the electronic document, processing the components element to instantiate the one or more script objects, including accessing the custom script object model to determine how to instantiate the at least one script object that conforms to the custom script object model registered with the browser;

wherein one or more of the script markup elements of the script markup portion reference at least one general markup element contained in the general markup portion of the retrieved electronic document, but script elements of the script markup portion are not referenced by any of the general markup elements of the general markup portion of the retrieved electronic document so that the script portion of the retrieved document is kept separate from the general markup portion when presented for viewing and design on a browser of a client computing system;

presenting the retrieved document for display at the browser of the client computing system with the separate general and script markup portions;

receiving user input that interacts with a portion of the displayed document that is represented by one or more general markup elements, and in response, accessing the custom script object model to perform functionality defined by one or more attributes associated with a custom script object that references the one or more general markup elements; and

performing the functionality to modify the appearance of the portion of the displayed document that is represented by the one or more general markup elements.

DESCRIPTION OF THE DRAWINGS

Embodiments of the present invention are hereinafter described, by way of example only, with reference to the accompanying drawings, wherein:

FIGURE 1 is a block diagram illustrating an exemplary computing system for implementing embodiments of the invention;

FIGURE 2 is a block diagram illustrating an exemplary partition of a markup document according to one embodiment of the invention; and

FIGURE 3 is a text diagram illustrating an exemplary markup document implementing embodiments of the invention.

DETAILED DESCRIPTION

The following text illustrates and describes exemplary embodiments of the invention. However, those of ordinary skill in the art will appreciate that various changes can be made therein without departing from the scope of the invention.

FIGURE 1 illustrates an exemplary computing system 100 for implementing embodiments of the invention. The computing system 100 includes a server component 102 and a client component 104. Generally, a browser 106 is associated with the client 104 for displaying a document such as a Web page. In a typical scenario, when the browser 106 requests to display a document, e.g., a Web page, the client 104 sends a document request to the server 102. The server 102 then sends the client 104 the markup document 108 containing markup information for displaying the requested document. The markup document 108 may exist in a database 110 associated with the server 102. Often, the server 102 and the client 104 exist on the same computer system. Alternatively, they may exist on different computer systems and communicate through a network (not shown).

In embodiments of the invention, upon receiving the markup document 108, the browser 106 parses and interprets the markup document 108 to display the requested document according to the definitions provided in the markup document 108.

In exemplary embodiments of the invention, the markup document 108 for a document such as a Web page provides general markup that defines the content and/or presentation of the document. The markup document 108 further includes or references script markup that defines the behavior of the document. FIGURE 2 illustrates exemplary blocks of information presented in the markup document 108. As shown in FIGURE 2, the markup document 108 includes a general markup portion 202 and a script markup portion 204.

The general markup portion 202 defines the formatting of the content and/or the overall appearance of the document to be displayed. The general markup portion 202 may define one or more general markup elements. For example, FIGURE 2 illustrates that the general markup portion 202 includes multiple general markup elements such as a general markup element A (206), a general markup element B (208), and a general markup element Z (210).

On the other hand, content of the script markup portion 204 defines interactive behavior and application logic associated with the document to be displayed. In embodiments of the invention, the content of the script markup portion 204 defines or references one or more script objects, and instantiates the script objects along with attributes defining the states, property values of the script objects. As shown in FIGURE 2, in embodiments of the invention, the script markup portion 204 is separated from the general markup portion 202 and is an independent portion of the markup document 108. Alternatively, in some embodiments of the invention, the script markup portion 204 can be included in a separate file, which is then referenced by the markup document 108. As shown in FIGURE 2, the content of the script markup portion 204 includes multiple script markup elements such as a script element 210, a reference element 212, and a components element 214. Both the general markup elements and the script markup elements are called markup elements.

In an exemplary embodiment of the invention, the script element 210 defines the overall scope of the script markup portion 204. All other elements in the script markup portion 204, such as the reference element 212 and the components element 214, are contained within the script element 210. Referring back to FIGURE 1, while interpreting the script markup portion 204, the browser 106 navigates through the script element 210 to interpret the included definitions, so to decide the behavior of the document to be displayed.

In embodiments of the invention, the reference element 212 references script files external to the markup documents 108 that are used by markup elements in the markup documents 108. The external script files may detail dependency information that the markup elements may use. Preferably, the external script files may also provide implementation details of script markup elements defined or referenced in the script markup portion 204.

The components element 214 contains one or more script object definitions that actually define the behavior of the document to be displayed. In exemplary embodiments of the invention, one or more of the script objects defined in the components element 214 may reference and hence define behaviors of one or more of the general markup elements included in the general markup portion 202.

FIGURE 3 illustrates an exemplary markup document 108 implementing the exemplary markup elements illustrated in FIGURE 2. As shown in FIGURE 3, the exemplary markup document 108 contains a hierarchical structure, in which one markup element may be contained by another markup element. Each markup element includes tags, as denoted by, for example, < > symbols, with the actual element being detailed between the tags. Each markup element includes a start tag and an end tag, wherein a start tag begins a markup element and an end tag ends the corresponding markup element. For example, as shown in FIGURE 3, the script element 210 begins with the start tag < > on line 3 and ends with the end tag </> on line 34. As will be described in detail below, the markup elements in the markup document 108 further contain one or more attributes with assigned values.

The exemplary markup document 108 shown in FIGURE 3 illustrates script-defined behavior of two counters. As shown in FIGURE 3, lines 1-2 illustrate an exemplary general markup portion 202. Here, two general markup elements—Counter#1 and Counter#2—are defined, wherein Counter#1 has an "id" attribute with the value "counterLabel1" and Counter#2 has an "id" attribute with the value "counterLabel2."

Lines 3-34 illustrate an exemplary script markup portion 204 that specifies the behavior of the two counters defined in lines 1-2. Specifically, line 3 signals the beginning of a definition for an exemplary script element 210 and line 34 signals the end of the definition. The exemplary script element 210 includes an exemplary reference element 212 (lines 5-8) that links in two JavaScript files—AtlasUI.js and AtlasControls.js. Lines 9-32 illustrate an exemplary components element 214 that defines a plurality of script objects. For example, line 10 defines a script object Counter 302 that is identified as "counter1," while line 11 defines a script object Counter 304 that is identified as "counter2" and has a value of "10000." The code between lines 12-16 and lines 17-21 each defines a script object Timer (306, 316) that periodically, e.g., every 500 seconds, enables an event object Tick (308, 318). In embodiments of the invention, a script object may include one or more sub-script objects. For example, the script object Timer 306 includes an event object Tick 308, which further includes an action object invokeMethod 310. For another example, the script object Label 312 defined in lines 22-26 includes a binding object 314.

In exemplary embodiment of the invention, a script object may be associated with one or more attributes whose values are used to define the behavior of the script object. An attribute can be, for example, a property, a method, or an event associated with the script object. An attribute may also be a reference to another markup element. For example, the script object Counter 304 defined in line 11 has a property attribute "id" and a property attribute "value". The action object invokeMethod 310 defined in line 14 has an method attribute "Method" that is set to an exemplary "increment" method. For example, instead of using an event object Tick 308, the script object Timer 306 may have an event attribute "Tick". The scrip object Label 312 defined in line 22 has an attribute "targetElement" that references the general markup element Counter#1 identified as "counterLabel1" in line 1.

In exemplary embodiments of the invention, a script object may reference a general markup element defined in the general markup portion 202 of the markup document 108 and define document behavior associated with the referenced general markup element. For example, the code between lines 22-26 defines a script object

Label 312 that references the general markup element Counter#1 defined in line 1. The code between lines 27-31 defines a script object Label 320 that references the general markup element Counter#2 defined in line 2. Consequently, the script objects Label 312 and Label 320 may specify the behaviors of the general markup elements Counter#1 and Counter#2 in the general markup portion 202.

In embodiments of the invention, a script object may communicate with another script object by performing a specific action upon occurrence of a specific event. For example, in embodiments of the invention, a script object may be associated with an event, the occurrence of which initiates a corresponding event handler, which may link to developer-defined code for markup elements in the markup document 108. In an exemplary embodiment of the invention, the event handler includes one or more specific actions to be performed on one of the script objects in the components element 214. An exemplary action can be to invoke a method associated with another script object. Another exemplary action can be to configure a property associated with another script object. In a typical embodiment of the invention, both the event and the action are also script objects including one or more attributes. For example, the script object Timer 306 contains an event object Tick 308, the enablement of which initiates an action object invokeMethod 310. The action object invokeMethod 310 has an attribute "target" specifying a target script object—"counter1", for example—and an attribute "method" specifying the function to be performed on the target script object.

Another exemplary mechanism for one script object to communicate with another script object is a binding mechanism that connects a property of one script object with a property of another script object; the change of one property thus is reflected on the other property. For example, as shown in FIGURE 3, the script object Label 312 includes a binding object 314. The binding object 314 has an attribute "dataContext" that specifies the script object and an attribute "dataPath" that specifies one of the script object's properties with which the script object Label 312 will bind its property "text". As a result of the binding, the value of the script object Counter 302 defined in line 10 is reflected in the "text" property associated with the script object Label 312 and hence is displayed in the general markup element Counter#1 defined in line 1. In an exemplary embodiment of the invention, a binding object provides a transform functionality that transforms the type of the property that provides the data into the type of the property that receives the data.

For example, the transform functionality for the binding object 314 may convert the type of the property specified by "dataPath" into the type of the property specified by "text".

It is to be understood that FIGURE 3 illustrates only exemplary formats of a script markup language for implementing aspects of the invention. These exemplary formats should be used for illustration purposes only. These exemplary formats do not limit the script markup language offered by embodiments of the invention to the specific formats, syntax, and functionalities illustrated. For example, the exemplary markup document 108 has been illustrated using XML syntax and formats. However, those of ordinary skill in the art will appreciate that aspects of the invention may be implemented in different markup languages such as HTML, ASP.NET, JavaScript Object Notation, etc.

In embodiments of the invention, a developer may custom define a script object model. The script object model, for example, specifies attributes, such as property, method, and/or event attributes, and any sub-script object models that may be associated with the script object model. The script object model then is registered with the browser 106, for example, through a type manager associated with the browser 106. The browser 106 thus knows how to interpret and process a script object instantiated based on the script object model. As a result, the script markup language provided by aspects of the invention is extensible in that new script object models can be defined and registered with a browser for interpreting script markups containing script objects instantiated based on the script object models.

Although aspects of the invention have been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims.

The reference in this specification to any prior publication (or information derived from it), or to any matter which is known, is not, and should not be taken as an acknowledgment or admission or any form of suggestion that that prior publication (or information derived from it) or known matter forms part of the common general knowledge in the field of endeavour to which this specification relates.

Throughout this specification and claims which follow, unless the context requires otherwise, the word "comprise", and variations such as "comprises" and "comprising", will be understood to imply the inclusion of a stated integer or step or group of integers or steps but not the exclusion of any other integer or step or group of integers or steps.

2006291331 23 Feb 2011

THE CLAIMS DEFINING THE INVENTION ARE AS FOLLOWS:

1. A computer program product comprising a computer storage medium containing computer-executable instructions for implementing a method which facilitates designing the markup for an electronic document such as a Web page which is stored as a markup document in a database associated with a server computing system, and which is later retrieved for viewing and design of the markup document at a browser of a client computing system, and wherein the method facilitates the design by separating script markup language that defines interactive behavior and application logic associated with the electronic document from general markup language of the electronic document that defines content and presentation of the electronic document, the method comprising:

receiving input at a type manager associated with a browser for registering a custom script object model, the custom script object model containing one or more user defined attributes comprising properties, methods, or event attributes for use by the browser in interpreting script objects that conform to the custom script object model and that are contained within electronic documents that are executed by the browser;

registering the custom script object model with the browser to enable the browser to interpret script objects that conform to the custom script object model;

retrieving from a database associated with a server computing system an electronic document stored in the form of a markup document, for display at the browser of a client computing system, the retrieved electronic document having a markup language that defines in a first part of the retrieved electronic document a general markup portion comprised of one or more general markup elements that define formatting of the content and/or the overall appearance of the electronic document when displayed on a Web page, the retrieved electronic document having a markup language that defines in a second part of the same retrieved electronic document a script markup portion comprised of a reference element and a components element, wherein the reference element references script files external to the retrieved electronic document, and wherein the components element contains one or more script objects for implementing interactive behavior and application logic associated with the electronic document when displayed as a Web page, wherein at least one of the script objects is a custom script object that conforms to the custom script object model that was registered with the browser;

upon retrieving the electronic document, processing the components element to instantiate the one or more script objects, including accessing the custom script object model to determine how to instantiate the at least one script object that conforms to the custom script object model registered with the browser;

wherein one or more of the script markup elements of the script markup portion reference at least one general markup element contained in the general markup portion of the retrieved electronic document, but script elements of the script markup portion are not referenced by any of the general markup elements of the general markup portion of the retrieved electronic document so that the script portion of the retrieved document is kept separate from the general markup portion when presented for viewing and design on a browser of a client computing system;

presenting the retrieved document for display at the browser of the client computing system with the separate general and script markup portions;

receiving user input that interacts with a portion of the displayed document that is represented by one or more general markup elements, and in response, accessing the custom script object model to perform functionality defined by one or more attributes associated with a custom script object that references the one or more general markup elements; and

performing the functionality to modify the appearance of the portion of the displayed document that is represented by the one or more general markup elements.

2. The computer program product of Claim 1, wherein one of the one or more script objects references one of the one or more general markup elements in the general markup portion.

3. The computer program product of Claim 1 or 2, wherein one of the one or more script objects includes one or more sub-script objects.

4. The computer program product of Claim 3, wherein one of the sub-script objects is a binding object for connecting an attribute of the script object with an attribute of another script object, wherein both attributes are property attributes.

5. The computer program product of Claim 4, wherein the binding object includes a function for converting type of the attribute of the script object into type of the attribute of the another script object.

6. The computer program product of any one of Claims 3 to 5, wherein one of the sub-script objects is an event object.

7. The computer program product of Claim 6, wherein the event object further includes an event handler detailing what to do when the event occurs.

8. The computer program product of Claim 7, wherein the event handler includes an action object that initiates a specific action when the event occurs.

9. The computer program product of Claim 8, wherein the action involves executing an attribute of another script object, wherein the attribute is a method attribute.

10. The computer program product of Claim 8, wherein the action involves configuring an attribute of another script object, wherein the attribute is a property attribute.

11. A method which facilitates designing the markup for an electronic document such as a Web page which is stored as a markup document in a database associated with a server computing system, and which is later retrieved for viewing and design of the markup document at a browser of a client computing system, and wherein the method facilitates the design by separating script markup language that defines interactive behavior and application logic associated with the electronic document from general

markup language of the electronic document that defines content and presentation of the electronic document, the method comprising:

receiving input at a type manager associated with a browser for registering a custom script object model, the custom script object model containing one or more user defined attributes comprising properties, methods, or event attributes for use by the browser in interpreting script objects that conform to the custom script object model and that are contained within electronic documents that are executed by the browser;

registering the custom script object model with the browser to enable the browser to interpret script objects that conform to the custom script object model;

retrieving from a database associated with a server computing system an electronic document stored in the form of a markup document, for display at the browser of a client computing system, the retrieved electronic document having a markup language that defines in a first part of the retrieved electronic document a general markup portion comprised of one or more general markup elements that define formatting of the content and/or the overall appearance of the electronic document when displayed on a Web page, the retrieved electronic document having a markup language that defines in a second part of the same retrieved electronic document a script markup portion comprised of a reference element and a components element, wherein the reference element references script files external to the retrieved electronic document, and wherein the components element contains one or more script objects for implementing interactive behavior and application logic associated with the electronic document when displayed as a Web page, wherein at least one of the script objects is a custom script object that conforms to the custom script object model that was registered with the browser;

upon retrieving the electronic document, processing the components element to instantiate the one or more script objects, including accessing the custom script object model to determine how to instantiate the at least one script object that conforms to the custom script object model registered with the browser;

wherein one or more of the script markup elements of the script markup portion reference at least one general markup element contained in the general markup portion of the retrieved electronic document, but script elements of the script markup portion are not referenced by any of the general markup elements of the general markup portion of the

2006291331 23 Feb 2011

retrieved electronic document so that the script portion of the retrieved document is kept separate from the general markup portion when presented for viewing and design on a browser of a client computing system;

presenting the retrieved document for display at the browser of the client computing system with the separate general and script markup portions;

receiving user input that interacts with a portion of the displayed document that is represented by one or more general markup elements, and in response, accessing the custom script object model to perform functionality defined by one or more attributes associated with a custom script object that references the one or more general markup elements; and

performing the functionality to modify the appearance of the portion of the displayed document that is represented by the one or more general markup elements.

12. The computer program product of any one of claims 1 to 10 wherein the custom script object model further specifies a sub-script object model that is associated with the custom script object model.

13. A computer program product, substantially as hereinbefore described with reference to the accompanying drawings.

14. A method which facilitates designing the markup for an electronic document, substantially as hereinbefore described with reference to the accompanying drawings.

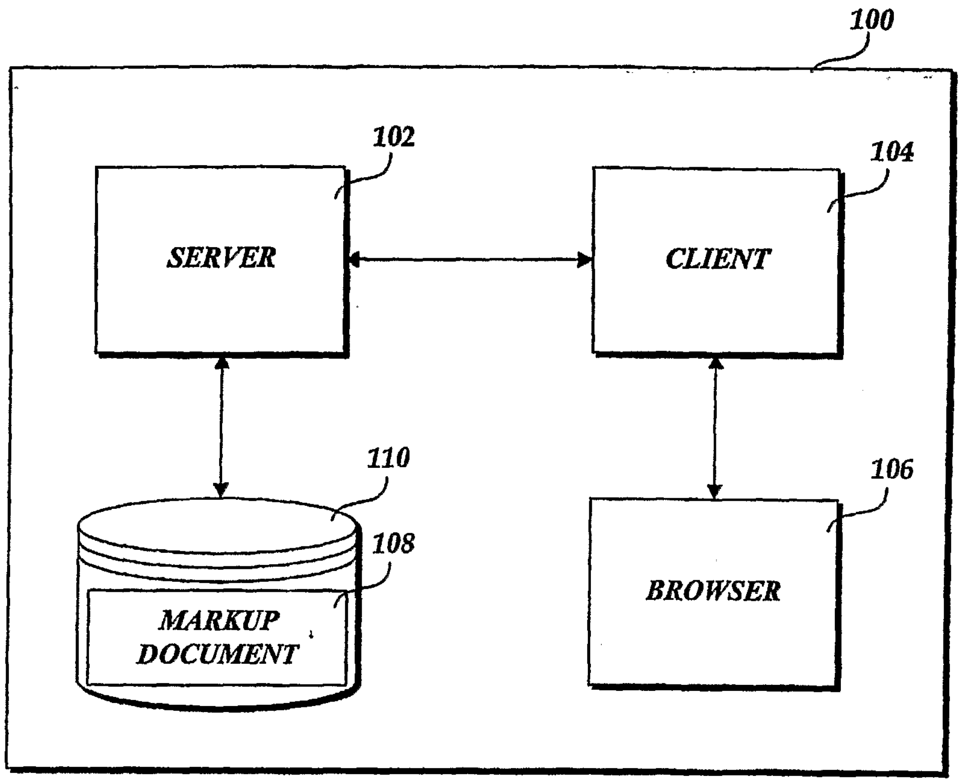

100
102
SERVER
104
CLIENT
110
108
MARKUP DOCUMENT
106
BROWSER

Fig.1.

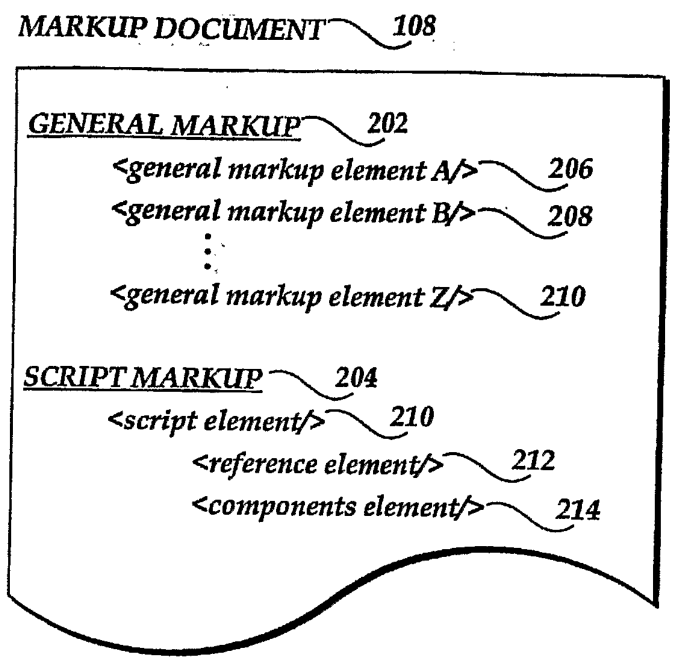
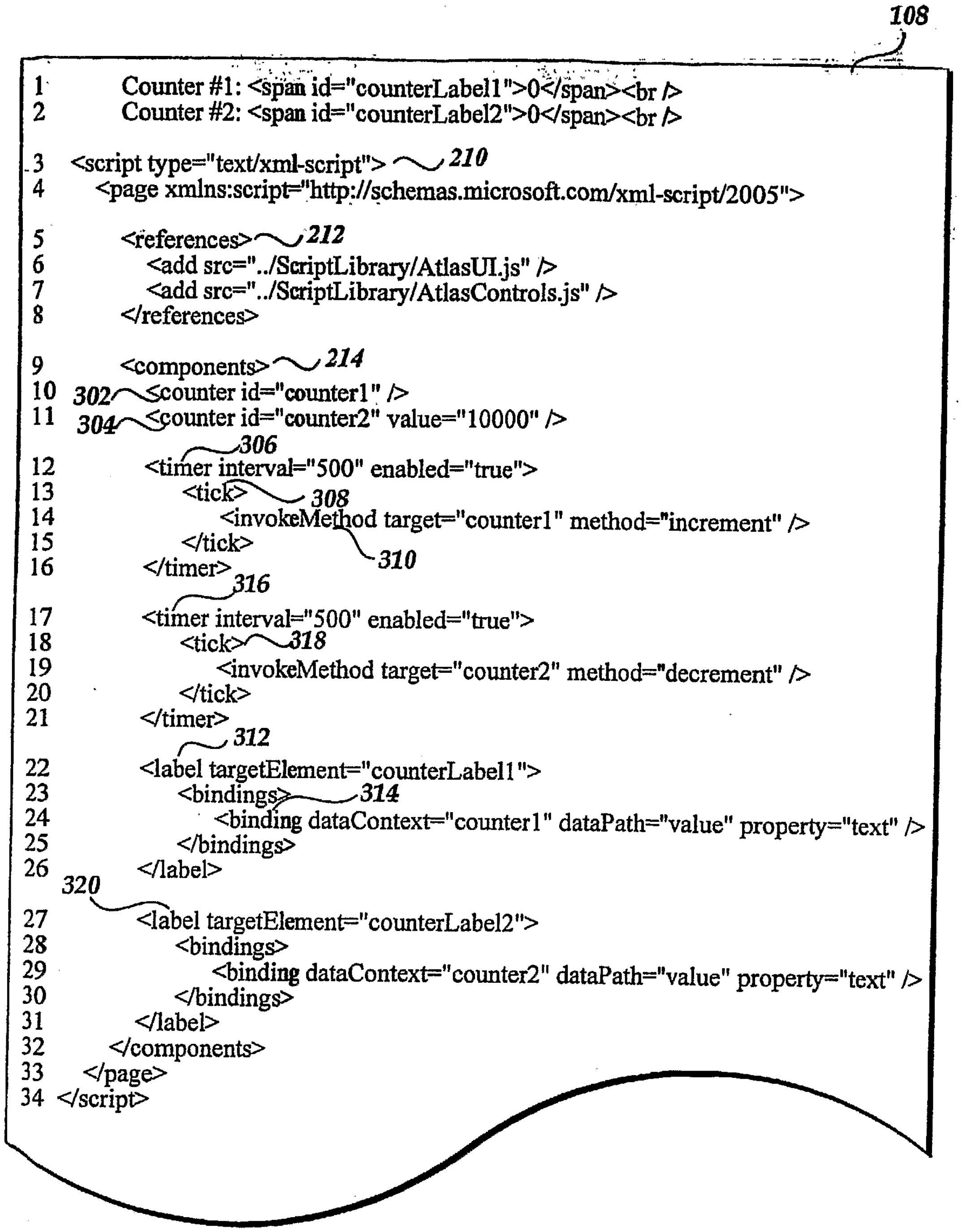

MARKUP DOCUMENT 108
GENERAL MARKUP 202
<general markup element A/> 206
<general markup element B/> 208
<general markup element Z/> 210
SCRIPT MARKUP 204
<script element/> 210
<reference element/> 212
<components element/> 214

Fig.2.


108
210
212
214
302
304
306
308
310
316
318
312
314
320

```
    Counter #1: <span id="counterLabel1">0</span><br />
    Counter #2: <span id="counterLabel2">0</span><br />

<script type="text/xml-script">
  <page xmlns:script="http://schemas.microsoft.com/xml-script/2005">

    <references>
      <add src="../ScriptLibrary/AtlasUI.js" />
      <add src="../ScriptLibrary/AtlasControls.js" />
    </references>

    <components>
      <counter id="counter1" />
      <counter id="counter2" value="10000" />

      <timer interval="500" enabled="true">
        <tick>
          <invokeMethod target="counter1" method="increment" />
        </tick>
      </timer>

      <timer interval="500" enabled="true">
        <tick>
          <invokeMethod target="counter2" method="decrement" />
        </tick>
      </timer>

      <label targetElement="counterLabel1">
        <bindings>
          <binding dataContext="counter1" dataPath="value" property="text" />
        </bindings>
      </label>

      <label targetElement="counterLabel2">
        <bindings>
          <binding dataContext="counter2" dataPath="value" property="text" />
        </bindings>
      </label>
    </components>
  </page>
</script>
```

Fig.3.