ROLE OF WEB BROWSING LAYOUT ENGINE EVALUATION IN DEVELOPMENT PROCESS OF MORE USABLE WEB INFORMATION SYSTEM

Gatis Vitols, **Irina Arhipova**
Latvia University of Agriculture
gatis.vitols@llu.lv; irina.arhipova@llu.lv

Abstract. This paper focuses on evaluation of Web browsing layout engines used as a backbone in Web browsing software. Study of commonly used Web browsers and Web information system developing languages has been performed. The role of Web browsing layout engine evaluation in Web information system development process is identified as a critical matter for bringing business processes online in a form of usable and accessible information system. By analyzing Web browsing layout engines as a tool that renders elements on a particular Web page of Web information system, key tendencies and emphasis for Web information system developers are revealed and discussed.
Key words: Web browser, Web browsing layout engine, Hyper Text Markup Language, Web information system.

Introduction

Bringing everyday business processes online has become a consistently growing tendency that goes side by side with developing possibilities of the Web (Lane et al., 2008). A key tool to carry out business process online is to develop information systems that are able to work online serving large number of users. Web users typically are not restricted by geographic location and specific accessibility environments (Taniar and Rahayu, 2004).

One of the key software solutions for accessing Web services, including Web information systems, are Web browsers. Web browser is a software application for retrieving, presenting, and traversing information resources on the World Wide Web (William, 2009). When users read a specific page on Web information system, it means, they use some type of Web browser. Most browsers that are available on the market have a common nuance: browsers read Hypertext Markup Language, HTML and Cascading Style Sheets, CSS codes and interpret them to data which are more understandable to users in a form of visual layout. HTML, CSS and other commonly met Web page development languages, such as JavaScript, are client-side languages (Scott, 2006). It means in a client – server architecture, which is a core architecture in the Web, most of operations are processed in the client - user side (see Figure 1.).

In the client side information processing model, for example, if a mathematics statement such as 1+1 is made, the server will not perform the calculation. In such a model, the server does not interact with data. The server's main function is to send data to the client side. The main interaction, in Figure 1 example is made on a client computer. In such interaction Web browser has a huge impact on how the server sent information is processed. The browser receives data then performs operation taking into account the client hardware and software environment and displays the result.

Opposite to the client-side information processing, server-side processing is performed on the server. Data are processed and then the result is sent to the client.

In such a model, the client does not know how the data, which are sent by the server, are processed. In this model, the main task for the client is to receive and display data.

Each model with according languages has its main targeted tasks in the Web information system development. Sometimes the tasks can overlap, which requires developer skills to choose the solution for specific problems. One of the key factors in information system accessibility and usability is the designed presentation of information. System users usually are not interested how the information is processed or by which algorithm the task is completed. Most attention is paid on how the information system can be used in

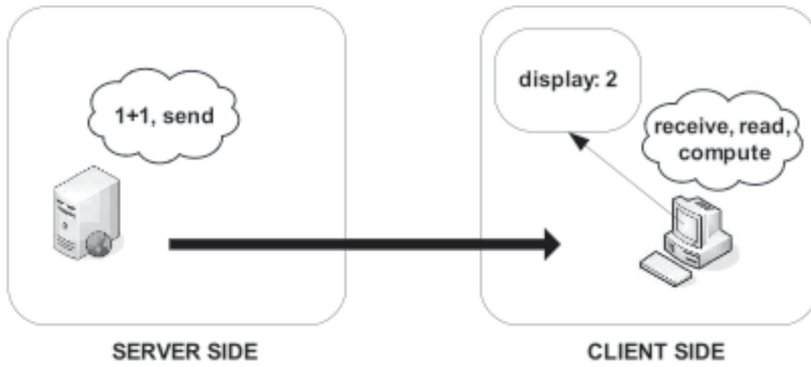

Figure 1. Client side information processing in the client-server architecture.

an effective, easy and beneficial way. It is seen that in order to meet such demands, focus on the client side information processing process examination has to be put.

The aim of this paper is to evaluate commonly used Web browsing layout engines and bring forward tendencies for more accessible and usable Web information system design. To reach the aim the following tasks have been brought forward:

1. Determine Web browsing layout engine diversity in various Web browsers.
2. Evaluate commonly used Web browsing layout engine similarities and differences regarding processing Hypertext Markup Language tags.
3. Identify tendencies and factors in Web browsers for information system developers that could allow creating more accessible and usable Web information systems.

Materials and Methods

When the analysis and summary of the leading Web standardization organization W3C provided browser usage statistics are made, it is seen, that in the year 2009, three leading Web browsers were Mozilla Firefox 3.x (46.94%), Microsoft Internet Explorer 6-8.x (40.52%) and Google Chrome 3-4.x (6.28%) which all together make 93.74% from all browser usage. In the beginning of the year 2010, tendencies remain the same and companies Mozilla, Microsoft and Google are developing and own most widely used Web browsers. According to published data (World Wide Web Consortium, 2010), there is an obvious tendency that popularity of Google Chrome and Mozilla Firefox has slightly increased, while Microsoft Internet Explorer has lost its stable position in the market. For example, Google browser popularity has increased for more than 5% in January 2010 compared with December 2009 (World Wide Web Consortium, 2010).

Next in a queue for the three most used Web browsers is Apple owned Web browser Safari, whose popularity increases with its integration into company manufactured, popularity gaining products, such as iPod portable music player and iPhone model mobile phones.

Web browser market consists of more than 10 universal, widely known Web browsers, not including many locally known or existing in developmental stage. For this research three named Web browsers with corresponding layout engines were chosen: Microsoft Internet Explorer 6.0, Mozilla Firefox 3.5 and Google Chrome 4.0.

To examine the role of Web browser layout engine in development of accessible and usable Web information system, one of the backbone client side languages of the Web - Hypertext Markup Language was chosen. Since creation of HTML, this language has been updated to various versions. In the year 2000, HTML language version 4.01 specification (World Wide Web Consortium, 2009) was published as an ISO/IEC international standard ISO/IEC 15445:2000(E) with a title 'Information technology — Document description and processing languages — Hyper Text Markup Language' (International Standards for Business, Government and Society, 2006). This is the latest approved version of HTML, (Johansson, 2010) however, there is an intense work at HTML 5.0 version development which specification still, until February 2010 appears to be in a draft version, and possibly will be released during the year 2010 or 2011 (Johansson, 2010).

When using HTML 4.01, there is a need to define the format subtypes of a language usage or more precisely DOCTYPEs, such as Strict and Transitional. Usage of document types relates to the Web browser rendering methods. If the Strict is used, rendering will be more precise and close to the language fundamental aim. If Transitional is chosen, exceptions in HTML document formatting are allowed. (Johansson, 2005) As the Strict document type promotes, a Web page separation in structure and presentation, which makes more usable and easier accessible Web page (Johansson, 2005) to design, the document type 'Strict' is used for research published in this paper.

The test of HTML 4.01 Strict specification tag usage possibilities and Web browsers layout engine rendering have been performed. A sample page of Web information system containing each of elements across chosen browsers has been created and tested.

Browsers use the following Web layout engines: Microsoft Internet Explorer 6.0 use Trident unversioned, Mozilla Firefox 3.5 use Gecko 1.9.3 and Google Chrome 4.0 use WebKit unversioned.

There are 91 markup tags named in HTML 4.01 specification (Holzsschlag, 2004; World Wide Web Consortium, 2009). As the Trident layout engine is fully available only in Microsoft Windows operation system (Lane et al., 2008), an operation system for testing Web browsers, Windows family operating system version XP SP2 were chosen. Rendering results on chosen Web browsers are ordered in a HTML tag functional groups.

Results and Discussion

HTML top level elements can be described by tags - body, frameset, head and html. Execution and rendering examination results show that each top level element is supported by all three analyzed Web layout engines.

HTML head elements can be described by tags – base, isindex, link, meta, script, style and title. Execution and rendering examination results show that only isindex element is not supported by any of three analyzed Web layout engines. Unsupported isindex element relates to the chosen document type Strict. Specification of Strict describes this element as deprecated or not supported by document type specification. Therefore, Web developers should avoid using isindex element.

HTML generic block-level elements execution results are summarized in Table 1.

Table 1

Layout engine evaluation based on block-level elements rendering examination on various Web browsers

Tag name	Commentary	Trident	Gecko	WebKit
address	Contact information for a document	+	+	+
blockquote	Quotations	+	+	+
center	Center alignment	-*	-*	-*
del	Defines text that has been deleted from a document	+	+	+
div	Generic language container	+	+	+
h1	Used to define HTML headings, largest size	+	+	+
h2	Used to define HTML headings, size 2	+	+	+
h3	Used to define HTML headings, size 3	+	+	+
h4	Used to define HTML headings, size 4	+	+	+
h5	Used to define HTML headings, size 5	+	+	+
h6	Used to define HTML headings, smallest size	+	+	+
hr	Horizontal rule	+	+	+
ins	Defines the text that has been inserted into a document	+	+	+
isindex	Creates a single-line text input control	-*	-*	-*
noscript	Alternate content container for non script rendering	+	+	+
p	Paragraph	+	+	+
pre	Pre-formatted text	+	+	+

From the results included in Table 1 is seen, that from HTML block-level elements, only center and isindex are not supported. Mark '+' shows that the browser layout engine supports element rendering, but the mark '-*' means that the element rendering in document type Strict, is deprecated. Deprecated elements may become obsolete in the future, but browsers continue to support deprecated elements for backward compatibility. (World Wide Web Consortium, 2009) HTML list elements execution results are summarized in Table 2.

Mark '+' shows that the browser layout engine supports element rendering, but mark '-*' means that element rendering in document type Strict, is deprecated. Deprecated elements may become obsolete in the future, but browsers continue to support deprecated elements for backward compatibility (World Wide Web Consortium, 2009). HTML table elements execution results are summarized in Table 3.

Table 2

Layout engine evaluation based on list elements rendering examination on various Web browsers

Tag name	Commentary	Trident	Gecko	WebKit
dd	Description of items in a definition list	+	+	+
dir	List directory titles	-*	-*	-*
dl	Defines definition list	+	+	+
dt	Definition term	+	+	+
li	List item	+	+	+
menu	Menu list	-*	-*	-*
ol	Ordered list	+	+	+
ul	Unordered list	+	+	+

Table 3

Layout engine evaluation based on table elements rendering examination on various Web browsers

Tag name	Commentary	Trident	Gecko	WebKit
caption	Table caption	+	+	+
col	Table column for formatting	+	-	-
colgroup	Table column group for formatting	+	-	-
table	Table definition	+	+	+
tbody	Table body definition	+	+	+
td	Table cell	+	+	+
tfoot	Table footer	+	+	+
th	Table header cell	+	+	+
thead	Table header	+	+	+
tr	Table row	+	+	+

Mark '+' shows that the browser layout engine supports element rendering, but mark '–' shows that a tag is neither supported nor rendered. From the rendering results in Table 3, it is seen that Web information system developers should avoid using col and colgroup tags, as they are only supported by Trident layout engine and can lead to distortion of the whole rendered table on Mozilla Firefox and Google Chrome family Web browsers.

HTML form elements execution was successful for all analyzed Web layout engines. All HTML form elements, including a button, fieldset, form, input, label, legend, optgroup, option, select and textarea, are supported by all three analyzed Web layout engines. HTML special inline elements execution results are summarized in Table 4.

Mark '+' shows that the browser layout engine supports element rendering, but mark '–' shows that the tag is neither supported nor rendered. Mark '-*' means that the element rendering in document type Strict, is deprecated. A deprecated element is one that has been outdated. Deprecated elements may become obsolete in future, but browsers continue to support deprecated elements for backward compatibility (World Wide Web Consortium, 2009).

Table 4

Layout engine evaluation based on special inline elements rendering examination on various Web browsers

Tag name	Commentary	Trident	Gecko	WebKit
a	Anchor	+	+	+
applet	Allows designers to embed a Java applet	-*	-*	-*
basefont	Specifies a default font attributes for text in a document	-*	-*	-*
bdo	Specifies text direction, overriding the bidirectional algorithm	+	+	+
br	Line break	+	+	+
font	Specifies the font attributes	-*	-*	-*
iframe	Defines inline frame	-*	-*	-*
img	Embedded image	+	+	+
map	Defines image map	+	+	+
object	Generic embedded object	-	+	+
param	Named property value	+	-	+
q	Defines a short quotation	-	+	+
script	Defines a client-side script	+	+	+
span	Generic language container	+	+	+
sub	Subscript	+	+	+
sup	Superscript	+	+	+

Table 5

Layout engine evaluation based on phrase elements rendering examination on various Web browsers

Tag name	Commentary	Trident	Gecko	WebKit
abbr	Abbreviation	-	+	+
acronym	Acronym	+	+	+
cite	Defines a citation	+	+	+
code	Computer code fragment	+	+	+
del	Defines the text that has been deleted from a document	+	+	+
dfn	Defines a definition term	+	+	-
em	Emphasis	+	+	+
ins	Defines the text that has been inserted into a document	+	+	+
kbd	Defines keyboard text	+	+	+
samp	Defines sample computer code	+	+	+
strong	Strong emphasis	+	+	+
var	Defines a variable part of a text	+	+	+

From the rendering results in Table 4, it is seen that inline elements support varies from used Web layout engines. Trident shows the poorest performance by not supporting quotation and object elements, unlike Google Chrome Webkit can render all non deprecated special inline elements. Web developers should pay attention to widely used elements, such as iframe and font. These elements are deprecated and should be avoided.

HTML phrase elements execution results are summarized in Table 5.

Mark '+' shows that the browser layout engine supports element rendering, but mark '–' shows that tag is neither supported nor rendered. From the rendering results in Table 5, it is seen that Web developers should be aware of using abbreviation and definition tags in developing projects as they are only partly supported by analyzed Web layout engines. Gecko shows the best performance in phrase element execution as it supports all phrase tags.

HTML font style elements execution results are summarized in Table 6.

Mark '+' shows that the browser layout engine supports element rendering. Mark '-*' means that the element rendering in document type Strict, is deprecated. A deprecated element is one that has been outdated. Deprecated elements may become obsolete in the future, but browsers continue to support deprecated elements for backward compatibility (World Wide Web Consortium, 2009). From the rendering results in Table 6, it is seen that Web developers should avoid using strike-through and underlined text elements as they are deprecated. As an alternative Cascade Style Sheet font styles could be used.

HTML frame elements execution was successful for all analyzed Web Layout engines. All HTML frame elements, including frame, frameset, noframes, are supported.

Even if Web developers follow mentioned analysis of HTML element rendering, there can still be some sort of distortions that relate Web layout engine tag attribute rendering or specific non-HTML element rendering, so in the project development stage it is useful to test Web page on different browsers. Besides installing different Web browsers on a system, there are also other possibilities to test Web page against various Web layout engines.

Table 6

Layout engine evaluation based on font style elements rendering examination on various Web browsers

Tag name	Commentary	Trident	Gecko	WebKit
b	Bold text	+	+	+
big	Larger text size	+	+	+
i	Italic text style	+	+	+
s	Strike-through text	-*	-*	-*
small	Small text	+	+	+
strike	Strike-though text	-*	-*	-*
tt	Teletype text	+	+	+
u	Underlined text	-*	-*	-*

In the Web browser market, a research for solutions to make more universal Web browser that could work with multiple layout engines is in progress. There are solutions named Hybrid Web browsers. These browsers include multiple layout engines in one software product. For example, Lunascape Orion that includes triple Web layout engine support – Trident, Gecko and Webkit (Lunascape Author Group, 2009). But until 2010, hybrid Web browsers are still in developmental stage. Also hybrid Web browsers face serious problems. It is known that parts of Web layout engine family are proprietary and owned by specific company which means that a hybrid Web browser cannot implement such layout engines without a license from its owners. For example, Opera and Microsoft Internet Explorer owns its own Web layout engines. So the fact that, for example, Lunascape Orion operational requirements are Windows operation system and Internet Explorer 6 or later (Lunascape Author Group, 2009) clearly shows that in order to support Trident in Lunascape, a user should have Internet Explorer installed on their devices.

Evaluation also can be done by using some virtualization service that is provided on Web. Adobe BrowserLab (Adobe BrowserLab Author Group, 2009) is one of examples. This service allows testing Web pages in various browsers on different operation systems using one Web page opened in Web browser. Although testing service is limited by amount of browsers and operation systems, there are few options that can be helpful for Web developers. For example, Adobe Browser lab supports the so called Onion skinning (Adobe BrowserLab Author Group, 2009) feature that allows to overlay two different browser rendering results, identifies faults and evaluates offsets. These types of services usually require Flash technologies support and at least one particular version Web browser that is already functioning on a system (Adobe BrowserLab Author Group, 2009).

Conclusions

1. Layout engines that are analyzed in this paper are not the only existing ones, but as the data by World Wide Web Consortium shows these are mostly used in the year 2009. From the layout engine analysis summarized in result tables, it is seen that two layout engines - Gecko and Webkit perform with similar results. Trident HTML tag rendering possibilities are more limited, although the main HTML elements can be executed and rendered properly. According to the information by Microsoft (Lane et al., 2008), Trident Web layout engine has been dramatically improved in future versions; unfortunately statistics and practices still show that Internet Explorer 6 with its corresponding layout engine is still widespread in use.
2. The fact that one has to take into consideration when designing Web information system and facing layout engine rendering testing, is that the creation of draft specifications of HTML 5 is also in progress. There is a need for Web developers to take into account and be aware of new possibilities which will be available by release of HTML 5.0 specification. There will be also a need to analyze and evaluate HTML 5.0 elements rendering on various layout engines in the nearest future.
3. Analysis in this research is not complete as it only covers HTML main element rendering results, not including, for example, attribute rendering evaluation.
4. Web developers should consider testing developing project on various browsers as a critical matter. For testing, the most stable method is installation of various browsers on a system, but alternatives as hybrid Web browsers and virtualization services can be considered. The tendency for hybrid Web layout engine development is still under a question mark and shows that market will remain scattered with existence of various stand-alone layout engines, but virtualization services such as Adobe BrowserLab, can be a good option for evaluation of Web layout engine performance.
5. In conclusion, it can be seen that Web browsing layout engine evaluation against programming language element execution has a serious role in Web information system development process. Web developers should practice testing Web pages of Web information system on various browsers, follow guidelines and published practices, as it directly addresses developed system accessibility and usability problems.

Acknowledgements

Funding support for this research is provided by Europe Social Fund program “Support for Doctoral Studies in LUA”, agreement 2009/0180/1DP/ 1.1.2.1.2/09/IPIA/ VIAA/017

References

1. Adobe BrowserLab Author Group (2009) Adobe Browser Lab: An Introduction. Available at: https://browserlab.adobe.com, 10 December 2009.
2. Holzsschlag E.M. (2004) 250 HTML and Web Design Secrets, Willey Publishing, Inc. USA, pp. 115-142.
3. International Standards for Business, Government and Society (2006) ISO/IEC 15445:2000 - Information technology - Document description and processing languages - HyperText Markup Language (HTML) Available at: http://www.iso.org/iso/catalogue_detail.htm?csnumber=27688, 3 February 2010.
4. Johansson R. (2005) Transitional vs. Strict Markup. Available at: http://24ways.org/2005/transitional-vs-strict-markup, 12 December 2009.
5. Johansson R. (2010) Seven HTML related working drafts published. Available at: http://www.456bereastreet.com/archive/201003/seven_html_related_working_drafts_published, 4 March 2010.

6. Lane J., Moscovitz M. and Lewis R.J. (2008) Website Creation with CSS, XHTML, and JavaScript, Friendsof, Apress, USA, 362 p.
7. Lunascape Author Group (2009) Lunascape Orion 6: First triple engine browser. Available at: http://www.lunascape.tv/Products/tabid/111/Default.aspx, 3 January 2010.
8. Scott M.L. (2006) Programming Language Pragmatics: Second Edition. Elsevier Inc., USA pp. 701-722.
9. Taniar D. and Rahayu J.W. (2004) Web Information Systems, Idea Group Publishing, USA, 372 p.
10. William S. (2009) Web Browser History. Available at: http://www.livinginternet.com/w/wi_browse.htm, 5 May 2009.
11. World Wide Web Consortium (2009) Index of the HTML 4 Elements. Available at: http://www.w3.org/TR/html4/index/elements.html, 8 January 2010.
12. World Wide Web Consortium (2010) Web Statistics and Trends. Available at: http://www.w3schools.com/browsers/browsers_stats.asp, 14 February 2010.