Proceedings in
Scientific Conference

SCIECONF 2013

10. – 14. June 2013

ScieConf
SCIENTIFIC CONFERENCE 2013

The 1st International Virtual Scientific Conference

Published by:	EDIS - Publishing Institution of the University of Zilina Univerzitná 1 01026 Žilina Slovak Republic
Editors:	Ing. Michal Mokryš; Ing. Štefan Badura, Ph.D.; Ing. Anton Lieskovský, Ph.D.
ISBN:	978-80-554-0726-5
ISSN:	1339-3561
Pages:	529
Printed in:	200 copies
Publication year:	2013
Other:	All published papers undergone single blind peer review.

The evolution of web browser architecture

Tedo Vrbanec
University of Zagreb
Faculty of Teacher Education
Zagreb, Croatia
tedo.vrbanec@gmail.com

Nenad Kirić
APIS IT d.o.o.
Zagreb, Croatia
nenad@chaos.com.hr

Matija Varga
University of Zagreb
Faculty of Teacher Education
Zagreb, Croatia
maavarga@gmail.com

Abstract - **Web developers, who strive to maintain the high quality of its products, spend considerable amount of their development time on maintaining compatibility with currently popular web browsers. They need to insure compatibility with several standards; HTML, CSS, JavaScript and video. To insure quality, they need to allocate significant portion of their resources for maintaining cross browser compatibility. Resources used represent costs. This paper presents the concept of web browser architecture that could cut websites development cost, improve browsing experience for surfers. The paper explains concept advantages and potentials for global wide acceptance.**

Keywords - web browsers, software architecture, software implementation, tire architecture

I. INTRODUCTION

With current web technologies, web developers spend approximately 50% [1]–[3] of their development time for quality assurance and maintaining cross-browser compatibility. Overload in development and testing stages of projects implies financial costs.

Ensuring compatibility mean compatibility with current web standards; HTML, CSS, JavaScript, video and more, ensuring cross browser consistency of design and code so every visitor gets a coherent experience.

HTML, CSS and JS are separate technologies. Their standards are left to interpretation of browser vendors. The standards are complex and there is no strict testing process or certification such as ISO quality systems. Sometimes the interpretations of standards differ for the same version of web browser for different operating systems, if the same functionality was developed by different teams or if developers do not use the same versioning on different operating systems. The difference may arise because same features are developed by different people, or because of differences in the architecture and technical solutions of different operating systems.

Is it time for the web browser architecture evolution? The authors advocate this thesis and present the concept which could drastically lower costs of web development and simultaneously improve the browsing experience. After the intro chapter the paper reviews the current state of desktop and mobile browsers and the aftermath of a browser vendor wars. Some of the major consequences are the differences in interpretations of standards. Paper also covers a brief overview of compatibility tests for browsers and web pages as a very useful tool for websites quality assessment. Perils of modern web development are presented in the second chapter. The solution for web developing problems is offered in the third chapter; its advantages are presented and possible obstacles in its way to acceptance are described. The fourth section provides an overview of scientific and professional papers. The fifth chapter brings conclusions.

During research, the following research methods were used: web surveys, observation method, analysis (content analysis, analysis of the results, business analysis and analysis of documents, data analysis and mathematical analysis), graphical and mathematical modeling and descriptive statistics. Authors also used methods and tools for web browsers testing.

II. THE CURRENT STATE OF WEB BROWSERS AND WEB DEVELOPMENT

Every web browser vendor forms their own interpretation of web standards and there lie the immense costs of development for web developers. Cutting costs in web developing business means sacrificing compatibility with some browsers. Will something revolutionary happen in the near future and for example is W3C (World Wide Web Consortium) going to come up with a certification plan for browsers? An educated guess would be it won't happen because that would mean higher costs for browser vendors since they'll need to get in line with the certification requirements and let's not forget that all the vendors are W3C members.

A. Web browsers: trends and actual market shares

At the start of website or application development, a web developer needs to make a choice what browsers will his product support, because supporting all existing browsers and their version is just not feasible. A starting point in this decision making are statistics of preexisting website that undergoes reconstruction or generic stats based on anticipated user base characteristics or geo location or what is the average age of the target audience, or maybe will the solution require intensive interaction with the user even when the user is away from the desktop (need for mobile web browsers support)?

Depending on the final solution, even functionalities and versions of virtual machines like Flash or Java have to be taken into consideration and sometimes even more exotic virtual machines like Unity3D, a 3D engine for the browser.

1) Desktop web browsers

According to comparative tests of web browsers features, safety, speed, compatibility, the ease of use and the support of manufacturers [4]–[7], the best rated are Google Chrome and Mozilla Firefox. They are followed by Microsoft Internet Explorer, Opera and Safari in various proportions, but in the almost same ranks. According to the rates of users - web pages visitors [7]–[9], the best rated browsers are classified at almost identical order: Google Chrome and Mozilla Firefox with over 33% of votes, Opera and Microsoft Internet Explorer with a little over 10% of votes and Safari with under 10% of votes. Even technologically dedicated websites such as [10] have almost identical ranking of web browsers. Some differences exist only in the order of the third and the fourth place, Microsoft Internet Explorer and Opera. However, according to the number of users [11]–[15], the most used are Google Chrome and Microsoft Internet Explorer (>30%), followed by Mozilla Firefox (20%) and Safari (<10%). Let's complete this statistical review about desktop web browsers with confirming data about trends and current state with statistics from StatCounter, which are most often used by secundary sources, shown at Fig. 1 (presenting five year trend) and Fig. 2 (presenting data on April 2013). Other sources mentioned have very similar data, but StatCounter is selected as their representative.

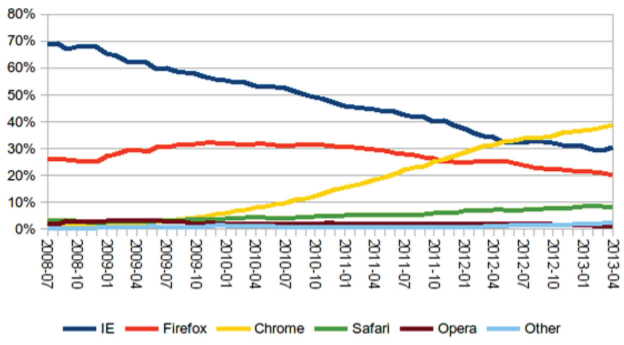

Figure 1. Top 5 Browsers from July 2008 to Apr 2013

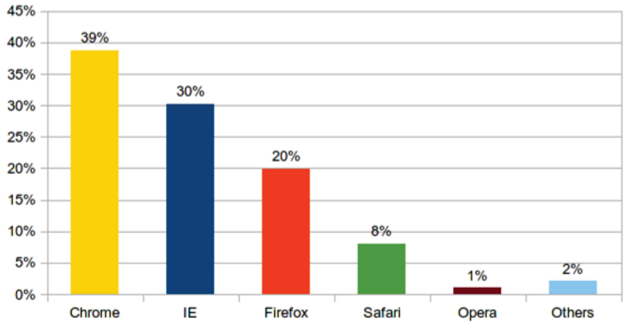

Figure 2. The market share of web browser on April 2013

In the period 7/2008 - 4/2013, a proportion of MSIE had fallen from 68.6% to 30.2%. Proportion of Mozilla Firefox was relatively stable: it grew from 26.1% to 32.2% (12/2009), after which it began falling to its current state of 19.98% (4/2013). Google Chrome was released in 9/2008 (beta version), and 12/2008 (stable version) and it grew from 0% to 38.8% which got him on the first place according to the number of users. The other two significant web browsers are Safari and Opera. Although very respected desktop web browsers, they aren't showing any big makings or losses, but they do have their own niche of users. Safari is respected on the iOS, Opera is thin on desktop but instead it's respected on all mobile platforms because it uses proxy which reduces the amount of data that must be retrieved from the Internet and it adjusts web browser to the platform on which the content is reviewed.

There are over hundreds of desktop web browsers and a partial list (about 75 browsers) can be found on Webdevelopersnotes.com [8].

2) *Non-desktop web browsers*

Web browser versatility is further enhanced by the massive use of smart phones, tablets and all sorts of smart (TV) devices (Apple TV, Google TV and others) which users can and do access the web. The world of mobile web browser is even more crowded than the desktop one. In total, there is less browsers but the most represented have a smaller percentage than those on the desktop market (Fig. 3).

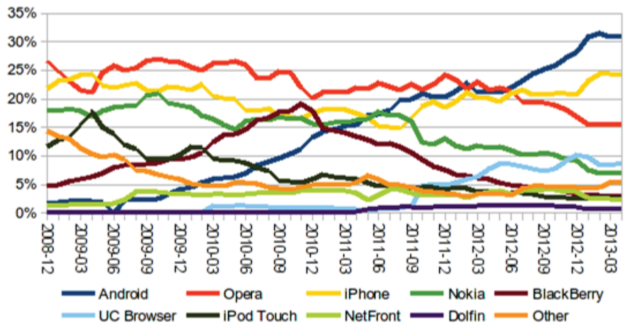

Figure 3. Top 9 Mobile Browsers from 12/2008 to 4/2013

In 3/2012, the three most widely used mobile web browser were used by less than 24% of users, all three were within the range of 4%: Opera, Android and iPhone. A few months later, in 12/2012, there was a significant change in the leading trio: Android became the leader with 27.4%, followed by the iPhone (20.9%), Opera (17.1%) and others. Since then, up to 4/2013 [16], Android has increased its primacy (30.93%), the iPhone has increased its share (24.06%), but Opera (15.35%) and all other significant mobile web browsers have reduced their share (Fig. 4).

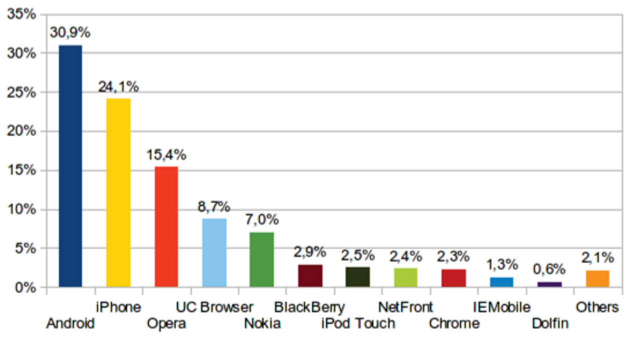

Figure 4. The market share of mobile web browser on 4/2012

Most of these mobile web browsers use the same basis (4/2013) [17]: over 75% WebKit Mobile [18] – an open source engine which was initially made by Apple for Safari, over 15% Opera mobile and rest share all others mobile web browsers.

B. *Considerations regarding the Browser Wars*

On the free market every product fights for its share of users. Since the browsers are free, developers/owners have to find some way to earn money to their company/project/organisation. They do that in a way to promote the name of the company or the project. When we say Internet Explorer, we know it's created by Microsoft, for Firefox we know it's an open source and that Mozilla works on it and so on.

In the early years of the Browser Wars, after promoting brands, the next step was to captivate users for their platform by their own upgrades of HTML/CSS/JS standards which led so far that Microsoft published their own version of Java virtual machine, their own web scripting language Vbscript and their version of JavaScript. The initial versions of HTML standard were rudiment, but the new tag which supported one browser didn't show itself as predicted on the other browser. The users were irritated because the competition of manufacturers resulted [19] in:

1. "Adding new features instead of fixing bugs.
2. Adding proprietary features instead of obeying standards.
3. Inadvertently creating security loopholes".

When the authority for management of web standards showed up, it was necessary to find a new source of income other than self-promotion and this is where search engines came. Search engines are the channel for promoting platforms for distributing web advertisement. The more visitors and the more advertisements on pages that use them, the more money is made. Of course, in that world, Google was interested to promote its search engine through other browsers as a default page and through search bars by which browsers became the directors of users and got money for the actual job. Therefore, a browser is strategic software for its company and, hidden from the view of users, makes money [20] based on the activity of users.

C. If there aren't enough problems: Vendor Prefixes

Even if most web browsers mostly respect web standards, some of their manufacturers are in a hurry with implementation of new specifications. Currently, CSS3 and HTML5 are tempting, even if they are not at Working Draft status [21] any more. The attempts of defining trends with new features are increasing in popularity, especially in the community of developers. In order to avoid drastic differences in sites of various browsers they agreed on a concept named Vendor Prefixes for CSS [22], [23]. Vendor prefix is an addition to CSS property which marks a feature in development, meaning: in experimental faze. Developers can use them, but their behavior in next versions could drastically change. In that way they distinguish new features from the standard. When the same or modified possibilities become standard from W3C, developers of web browsers should ignore standard Vendor Prefixes. This is just another attempt to control implementation of new possibilities.

However, the use of Vendor Prefixes has its downside [21], [24]–[27]. It turns out that massive use of WebKit as an engine on mobile web browsers and the use of its prefix has been brought to the point of web developers using more of those prefixes since the development team of WebKit wants to impose as a trend setter and include more new options in the syntax. By that, they put pressure on other browsers because they want them to support their Vendor Prefixes.

As it can be seen, process of web development and web browser development are cyclic because most companies see that the only way of getting advantage in front of the competition is to lure users by standard solutions and then turning to nonstandard ones to gain advantages before the competition [28], [29].

D. (In)compatibility Tests

Web developers tried different ways to harmonize their websites with different web browsers and their versions on different operating systems, because a slight difference in the version of the same browser or the same browser on a different operating system can very differently show the same sites. Parts of HTML, CSS and JS code called hacks, were in the fight for compatibility. Developers had most problems with Internet Explorer browsers, especially until the version eight. Problems which become obvious in cases of intensively using JavaScript, are speed and consistency of implementation of JS engine even in the same browser. There are also some fundamental differences in ways that the JS walks through DOM and AJAX methods which brought Mozilla and Microsoft. Parts of engines incompatibility are also problems of various JS frameworks such as jQuery or YUI and their inconsistency.

"Compatibilisation" wastes a lot of time but it is necessary to some extent. While websites already exist for some time, the extent of supported platforms can reduce, due to the statistics of visits and information about versions of browsers and operating systems. To support as much platforms as possible and versions of browsers for each, especially a small company, a significant expenditure is needed so compromises are needed. There are more possible ways of compatibilisation.

One of the ways is opening sites on the same machine with more recent versions of web browsers; with the restriction that one web browser can be installed in only one version.

The second way is a subscription to some online service which is specialized to show sites in different browsers and different versions on different platforms [30]–[35].

The third way is multiplication of hardware, meaning having test machines – that is financial inefficient. The same output can be made by virtualised technologies on production machines, such as [36]–[40].

Besides compatibility with different versions of different web browsers for different platforms, web developers have to make effort to keep compatibility with accepted web standards [41]. In that, they get help by on-line tests.

ACID1, ACID2 and ACID3 are online web tests [42]–[44] sponsored by The Web Standards Project (WaSP) [45], "a grassroots coalition fighting for standards which ensure simple, affordable access to web technologies for all." ACID tests measure how much the browsers follow the CSS standard [46].

HTML5 Test is an online web test whose [5] "score is an indication of how well" ... "browser supports the upcoming

HTML5 standard and related specifications." HTML 5 updates whenever the HTML 5 specifications change. Therefore, this test is not complete and it won't be until the specifications are determined as a standard. Indeed, considering the word is actually about a collection of tests that are performed sequentially, there are often new components of tests added. Therefore, the maximum number of point's changes (increases) by adding new components to tests. In addition to compliance with standards and specifications of standards, browsers can be tested and be compared about speed [47], memory use [48], but also an umpteen of many other features. Those testings' usually come out of some online non-commercial services and earlier mentioned online services.

Web browser tests and their compliance with web standards are important to web developers in order to comply and optimize their work with requirements. Checking the compliance of their products – websites – with standards is important too. Some of websites tests who are considered to be necessary in making websites (so called validators) are [49]: "Unicorn - W3C's Unified Validator, The MarkUp Validator - also known as the HTML validator, The Link Checker - checks anchors (hyperlinks) in a HTML/XHTML document and The CSS Validator - validates CSS stylesheets or documents using CSS stylesheets", ... The other, optional validators, are [49]: "Semantic Extractor - sees a Web page from a semantic point of view - extracts such information as outline, description, languages used, etc.; RDF Validator - checks and Visualize RDF documents; Feed Validator - checks newsfeeds in formats like ATOM and RSS; P3P Validator - checks whether a site is P3P enabled and controls protocol and syntax of Policy-Reference-File and Policy; XML Schema Validator" and two "human-centered test tools: the Mobile Test Harness is 'Web-based harness for browsers test suites, that offers users the possibility to record results on whether the browser they're using passes or not a set of test suites'; and MUTAT - an (older) human-centered testing framework developed in perl (code)".

III. SOLUTION FOR WEB DESIGN NIGHTMARE

In a world where everyone wants to make money on the web, the differences in the implementation of web standards in some browsers come to the fore and developers have to adapt to it. The solution for web design problems and consistency of the user surfing experience is in web browser architecture evolution i.e. new web browsers architecture.

Every browser has a standard set of components: GUI, browsing engine, rendering engine, output renderer, networking interface, persistancy engine and plug-ins. Rendering engine has its primary components: HTML, CSS and JavaScript interpreters, as shown on the Fig. 5: GUI, Display Backend, Browser Engine, Rendering Engine, Data Persistance, Networking and many possible modules, such as XML Parser, JavaScript Interpreter, Flash plug-in and others.

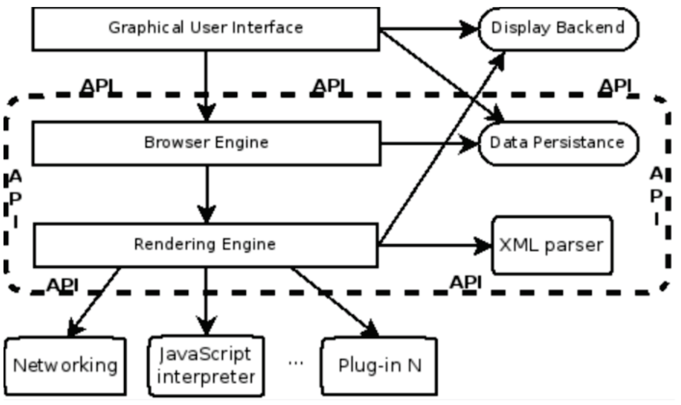

Figure 5. Web browser software architecture

What would happen if the HTML interpreter was declared as a plug-in? What if the same thing happens to the JavaScript engine? In that suggestion not only would the HMTL and the JavaScript engine become user configurable components but that would give developers the power to choose main plug-ins which supports the objectives of their project even before the actual development. Also, in the heading of HTML they could define which engine they want for technology, specified to the smallest decimal version if it suits them. The solution showed on Fig. 5 is on the borders of objects which with the browsers next to them communicate through the standard interface defined through the API. The authors propose to add and define standardized interfaces (API's) in addition to the already defined software architecture browser [50], [51]. That would allow components to talk in a standard way and, interchangeability of components get possible and defined, i.e., that would allow the HTML to influence the selection modules that will be used to display the page and users would see only GUI. Every GUI would have a standard interface regarding primary plug-ins which it runs as needed. When surfers come to a specific website written in the HTML code, in the heading they take over the definition which engines were used. If they don't already have it in the repository of engine, it initiates the download of it and performs silent installations, followed by the download of HTML and other parts of the site, in a similar way how they downloaded Flash and Java virtual machines if they found content needing a new version.

Existing browsers use separate projects which make JS and HTML engines as their components. In most cases the HTML engine integrates the browsing engine which is closer to user interface, the CSS and XHTML interpreters and doesn't care about the standardization of interfaces with the outside world because every browser has its private GUI. The modularization of browsers should be going through stages where in every next stage browser have more and more components with standard interfaces which solve specific functions. In the first stage the rendering engine, the browsing engine and JS engine should be separated. The other components such as output, persistency and networking can stay connected on a nonstandard way. In the next stage, parts of the rendering engine could become modular so that the HTML, XML and CSS get their modules. In the third stage the persistency engine would also get modules. Bookmarking, file cache and web app user data storage is to be standardized and centralized so that all other engines could use them and be unique on the system level. Now users already suffer transferring bookmarks from one browser to another on the same machine. The central file cache will reduce the space needed for it because it will no longer exist as many cache instances as there are installed browsers. When used HTML and JS engines (which interprets the site) changes, the user will still be able to see same information as before with including central web app user data store. In next stages, some subsystems for networking, for the visualization of sites should be singled out too, as well as managing the utility plug-ins which are mostly written in JS but aren't cross browser compatible. Regarding the sophistication of applications and websites which are served over the Internet and the level of sophistication of security threats, in one of the next stages there's a need to separate a security component in the subsystem for networking which would represent an additional proxy between network protocols and the rest of the system and will improve security. That proxy could have an API toward the proxy software of other manufacturers and toward other antivirus software. It would also be a proxy between the HTML engine and data persistency engine to secure that the web code doesn't compromise data in the user bases or the file system.

New iterations of the API standard would bring the support for sandboxed native applications which could be taken over from the internet without any rigorous installation procedures. Native applications would mean they are compiled for a platform of users, for example JS or C++ compiling. Around the sandboxes would also be a security layer and in architecture it would take place between the HTML engine and data persistency components. By bringing in applications it would be necessary to bring in an app store component which would take care of install/uninstall and monitor the appearance of new versions of applications and notify the user about it over GUI.

Possible candidates for modularization would be all SVG engines, handling micro formatting, printing pages or something else.

The suggested direction of development has big possibilities and its power is on the one hand in specialization of developer teams and essential components and on the other hand approaching developers to browsers, browser plug-ins and developers of web.

Clear definition of interfaces (the APIs) between components would enable various enthusiasts, specialists and companies to focus even more on their field and make quality specialized components. The most potentials for improvement is visible in JavaScript interpreters which are heavily (ab)used in the production of web applications and utilization in graphics for which JavaScript wasn't designed for. The APIs need to be defined as they must be unambiguous in a way to prescribe tests, inputs and outputs on which components can be marked as certified for some versions of API. For example, the hardware acceleration of graphics is made differently on Linux and on Windows platforms, as the first one leans to OpenGL and the second one on DirectX. In fact, the developers of rendering plug-ins would have a favor done and shouldn't get worried if under there it's an OpenGL or DirectX or the graphics related to the HTML and JavaScript because more of 3D is also pushing to that field and the websites aren't native applications. The subsystems for acceleration of graphics are a part of the operating system or separated projects in the open source community but someone who is developing a robustious JavaScript engine in most cases shouldn't be losing time or money on question such as the hardware acceleration of graphics which the JS can support.

A. Advantages of suggested solution

Development teams can choose plug-ins of various developers depending on their characteristics and performances and depending on work that need to be done. The time for testing functionalities of browsers, detecting differences and implementation or testing fixes which annul differences will decreasing. The times spent will also decreasing because the developing teams will focus on a specific version of HTML and JS engine. Eventually the only thing it might remain is testing the differences on various operating systems. If the developer of the plug-in doesn't subscribe a product parallel on different platforms, in the heading of HTML would be definitions for various operating systems.

Until now, developers could choose which monolithic browsers and their versions will support. Those browsers weren't specialized for their current business problem but for the universal solution which needs to cover a bigger number of user scenarios.

Engines which would be used would get referenced in the heading of HTML in a similar way as today Flash Player. References should be possible in the header of each web page which means that two different web pages on the same website can have references to different engines, as appropriate for each task.

Let's explain the consequence of accepting our suggestion on an example: let one site, let's call it A, be developed in 2012 for a WebKit version x.y and let it be visited by "our browser" with a WebKit x.y HTML/CSS plug-in in 2015: that site will look identical. Let the second site, let's call it B, be developed in 2013 with a new engine, for example the NN version t.z, considering that "our browser" has the NN plug in, and both sites are showing to users as the authors imagined.

Therefore, the proposed solution permanently removes the current problems of development and use of the website if the visitor has the engine that is needed. Of course, in the beginning most sites in the heading won't have a reference to plug-ins, but in the GUI we can have an option that one engine is used as a default choice. Modularization means decreasing work for development teams of module browsers. By that, maintenance is easier, especially when the HTML renderer is separated from the JavaScript engine. Focusing on a smaller amount of source code means decreasing potential attack surface for the development team which means they have a smaller amount of code for security testing.

GUI components would change slower than other components, so users would always have one acceptable interface which changes when they want, not when the changes are dictated by the components under. Currently the GUI browser and the arrangement of options differ significantly. The differences come through the life cycle of several browsers where developers usually impose their vision of ergonomics. The GUI component would also take care of following versions of components so they could be warned about new security patches and with adequate architecture, new versions could be downloaded and used without restarting the whole browser. API between components contributes to easier standardization.

By everything stated, the time of developing websites/applications decreases, which decreases expenses, but it also facilitates the development of web browser components what cheapens their development, makes them have better quality, safety and compatibility with other browser components.

B. Predicted barriers for accepting solutions

There are a couple of barriers that we will discuss, concerning the standardization and earnings.

Is the concept of standardization disturbed?

The first logical question being asked is: who needs or can accept this new concept of developing web browsers?

Developers of browsers would rebel because they set up business strategies around their monolithic products which are their marketing tool and their source of earnings on the market of online search engines and advertising. However, on the market, customers and users have the last word. For them, this alternative might be the best news since the invention of web.

W3C could provide resistance because this suggestion brings in referencing vendors and versions in the heading of HTML and by that disturbs the neutrality of the source code. Yet, vendor specific references are nowadays present on Flash and Java VM.

The pressure needed for changes in the way of browser developers' thinking will have to be made by organized users and developers. Therefore, the concept should first be shown to web developers and to users. When they get loud enough, the W3C won't be able to ignore the trends and it will get involved in the definition of the API call among the components.

Developers and surfers are winners in these changes. The prospective losers, at least in the beginning of adaptation to new standards would be developers of the HTML engine which developers neglect for objective or subjective reasons.

Browser vendors will resist to new solutions because they are being forced to accept the APIs they didn't create.

Resistance would also be made by former contractors about the income from directing users to specific browsers because that means signing new contracts with developers of various components.

In those conditions, developers of the GUI components would have to sign contracts with browser component developers, for example, for statuses of default components but also for sharing income of browsers.

Earnings through web browsers

Companies earn money by using the mechanisms built into the web browser interface. Towards our suggestion, they could keep earning in that way (imposing default search engines, gathering statistics of user surfing and so on), but if the opponents in browser development are considered mortal enemies, the question is how much would they be ready to approach work in developing plug-ins by the defined standard. Of course, everyone will defer the trend if, for example, someone makes a GUI by this proposal, and shapes the existing source of Firefox and Chrome into plug-ins. That project will then enable people to use both browsers through the same interface and Opera, MS and Apple for sure won't ignore it.

Leaders of new modular trends could be universities, therefore the academic community which is not strictly tied to terms of profitability or enthusiasm by starting new projects which grew out on the foundations of frustrations on current state.

Money will adapt to new trends and web search engines will still be interested in advertising through browsers [20], [52]. Therefore, there will be money for browser and web developers but with less spent time, which also means potentially bigger earnings or margins space in calculations.

IV. RELATED WORKS

Unfortunately, the problem and the nature of the relationship between web development on the one hand, and the concept of making and using a web browser on the other hand, has remained largely unnoticed among the scientific community and researchers. There is relatively small number of authors who deal with the software architecture of web browsers, but some works are inspiring. There are several authors mainly concerned with the architecture of web applications.

Windrum [53] brings interesting and quality analysis of the (first) browser war. Oshri and de Vries [54] explain the evolution of web browsers through the analysis of the competitive relationship between the main players.

Grosskurth and Godfrey are the most significant researchers who discuss the software architecture of web browsers. By studying the open source of web browsers they [52] developed a reference architecture for web browsers and in the conclusions noted that it could be different than the current one. The same authors [50] additionally widened gatherings of web browsers on which they combined the reverse engineering and by studying the source code determined their architecture: Mozilla, Konqueror, Epiphany, Safari, Lynx, Mosaic, Firefox by which they confirmed their referent architecture and defined key subsystems of listed browsers.

Li and Chen [55] propose an innovative Web browser interface called TopicBrowser, in which learners can make an Information Gathering plan with a hierarchical concept map in the context of learning.

Huang et al. [56] proposed a new type of middleware, which is embedded in web browsers and encapsulates reusable solutions for common problems to the composition of Web-delivered services.

Bohannon and Pierce [57] bring formal specification of the core data structures and operations of a web browser.

J. P. Espada et al. [58] proposed that web applications should be allowed to access context information in a simple and fast manner: "The proposed system consists of a modular web browser context aware and a set of specific XML tags that can be used on web applications."

M. Šilić et al. [59] find that modern web applications are demanding "a new web browser architecture design that will meet new security requirements arisen with Web 2.0." Authors are focused on web browser vulnerabilities; they analyze popular web browsers' architecture and present "how they cope with potential security threats".

V. CONCLUSION AND FUTURE WORK

Results of research can and should be used in developing and optimizing web browsers, and implicitly websites. Dissatisfaction of current states or inertness in solving problems is motivator that cannot be neglected when deciding about changes. The number of surfers by far exceeds the number of browser developers and if the unique platform is ensured to them to show their dissatisfaction by current state of browsers, corporations and organizations which think of browsers as their strategic products will start paying attention. They will start paying attention if the developers are also given a platform on one global place to show their dissatisfaction. Therefore, even before anyone invests money in lobbying or in the development of alternative solutions, the attention could be drawn by feedback information from those who use browser for consuming information and everyday fun and by feedback information of those who use browsers as media for distributing content and solutions.

If anyone "dare" to make a browser by the suggested concept, no matter how much money would be spent on modifying an open source code of existing open source browsers, return on investment would be guaranteed. Imagine offering developers a platform which various versions of HTML and JS interpreters treat as plug-ins without problems of coexistence of various version on the same instance of operating systems. Such a piece of software would be virally promoted in the development community. The commercial would bring users to browsers and in time profitable contracts with search engines which are associated with markets of web commercials.

By reducing time needed for the development of cross browser compatible webs and time needed for testing the same ones, developers, the people and business customers who orders web projects and web users/visitors would become happier.

The authors advocate that users obtain browsers which have IE, FF, Chrome and Opera engines in one interface. To make this possible, it would be necessary to make the API that all the engines would follow so that the GUI can communicate with them in a consistent manner. Modularization is great, but it is clear that current browsers are not built with such a dose of modularity in mind, which means that it will take iterations to define standards and compatible solutions. However, the basic components of web browsers architecture stay the same. The difference proposed is in standardization of communication between components so they could be changeable on the user side.

The next step is promoting the idea (raising awareness) on the Internet and the attempt of getting people with the same thinking in the developers' branch but also in the web browser development branch.

REFERENCES

[1] O. White, "Developer Productivity Report - Part 1: Developer Timesheet," *Zeroturnaround.com*, 2012. [Online]. Available: http://zeroturnaround.com/blog/developer-productivity-report-part-1-developer-timesheet/. [Accessed: 25-Apr-2013].

[2] C. Jenke and O. Loy, "Ratio of time spent on coding versus unit testing - Stack Overflow," *Stackoverflow.com*, 2008. [Online]. Available: http://stackoverflow.com/questions/174880/ratio-of-time-spent-on-coding-versus-unit-testing. [Accessed: 25-Apr-2013].

[3] J. Pan, "Software Testing," *Carnegie Mellon University*, 1999. [Online]. Available: http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/. [Accessed: 25-Apr-2013].

[4] TopTenReviews.com, "Internet Browser Software Review 2013," *Toptenreviews.com*, 2013. [Online]. Available: http://internet-browser-review.toptenreviews.com/. [Accessed: 22-Apr-2013].

[5] Browserscope.com, "The HTML5 test - How well does your browser support HTML5?," *Browserscope.com*, 2013. [Online]. Available: http://html5test.com/results.html. [Accessed: 23-Apr-2013].

[6] M. Muchmore, "Browser Wars: Chrome vs. IE9 vs. Firefox," *PCmag.com*, 2012. [Online]. Available: http://www.pcmag.com/article2/0,2817,2389248,00.asp. [Accessed: 23-Apr-2013].

[7] P. Wayner, "The best Web browser: Chrome, Firefox, Internet Explorer, Opera, or Safari?," *InfoWorld.com*, 2010. [Online]. Available: http://www.infoworld.com/d/applications/the-best-web-browser-chrome-firefox-internet-explorer-opera-or-safari-516. [Accessed: 23-Apr-2013].

[8] WebDevelopersNotes.com, "Browsers List," *WebDevelopersNotes.com*, 2013. [Online]. Available: http://www.webdevelopersnotes.com/design/browsers_list.php3. [Accessed: 22-Apr-2013].

[9] Classora.com, "Ranking of the Best Web Browsers," *Classora Knowledge Base*, 2011. [Online]. Available: http://en.classora.com/reports/y58864/ranking-of-the-best-web-browsers. [Accessed: 23-Apr-2013].

[10] GeekAdda.com, "Top 5 Web Browsers based on Speed and Flexibility," *Geek Adda*, 2012. [Online]. Available: http://www.geekadda.com/top-5-web-browsers-based-on-speed-and-flexibility/. [Accessed: 22-Apr-2013].

[11] NetApplications.com, "Browser market share," *NetMarketShare*, 2013. [Online]. Available: http://marketshare.hitslink.com/browser-market-share.aspx?qprid=0&qpcustomd=0. [Accessed: 22-Apr-2013].

[12] W3Counter.com, "W3Counter - Global Web Stats," *W3Counter*, 2013. [Online]. Available: http://www.w3counter.com/globalstats.php?year=2013&month=3. [Accessed: 22-Apr-2013].

[13] E. Zachte, "Wikimedia Traffic Analysis Report - Browsers e.a.," *Wikimedia Traffic Analysis Report*, 2013. [Online]. Available: http://stats.wikimedia.org/archive/squid_reports/2013-03/SquidReportClients.htm. [Accessed: 22-Apr-2013].

[14] Clicky Web Analytics, "Web browsers (Global marketshare)," *Clicky Web Analytics*, 2013. [Online]. Available: http://www.getclicky.com/marketshare/global/web-browsers/. [Accessed: 22-Apr-2013].

[15] Classora.com, "Ranking of the World's Most Popular Web Browsers (01/2012) - Classora Knowledge Base," *Classora Knowledge Base*, 2012. [Online]. Available: http://en.classora.com/reports/s87934/ranking-of-the-worlds-most-popular-web-browsers. [Accessed: 22-Apr-2013].

[16] StatCounter.com, “Top 9 Mobile Browsers on Apr 2013,” 2013. [Online]. Available: http://gs.statcounter.com/#mobile_browser-ww-monthly-201304-201304-bar. [Accessed: 26-Apr-2013].

[17] Clicky Web Analytics, “Mobile web browsers (Global marketshare),” *Clicky Web Analytics*, 2013. [Online]. Available: http://clicky.com/marketshare/global/web-browsers/mobile/. [Accessed: 26-Apr-2013].

[18] Webkit.org, “WebKit,” *Webkit*, 2013. [Online]. Available: http://trac.webkit.org/wiki. [Accessed: 25-Apr-2013].

[19] Wikipedia.org, “Browser wars,” *Wikipedia, the free encyclopedia*, 2013. [Online]. Available: http://en.wikipedia.org/wiki/Browser_wars#Consequences. [Accessed: 26-Apr-2013].

[20] S. Anthony, “How browsers make money, or why Google needs Firefox,” *ExtremeTech*, 2011. [Online]. Available: http://www.extremetech.com/internet/92558-how-browsers-make-money-or-why-google-needs-firefox. [Accessed: 30-Mar-2013].

[21] R. Andrew, “Call for action on Vendor Prefixes,” *The Web Standards Project*, 2012. [Online]. Available: http://www.webstandards.org/2012/02/09/call-for-action-on-vendor-prefixes/. [Accessed: 24-Apr-2013].

[22] P. Beverloo, “Vendor-prefixed CSS Property Overview,” *Peter Beverloo*, 2013. [Online]. Available: http://peter.sh/experiments/vendor-prefixed-css-property-overview/. [Accessed: 26-Apr-2013].

[23] T. Olsson and P. O’Brien, “Vendor-specific Properties,” *Sitepoint.com*, 2012. [Online]. Available: http://reference.sitepoint.com/css/vendorspecific. [Accessed: 24-Apr-2013].

[24] D. Glazman, “Call for Action,” *Glazblog*, 2012. [Online]. Available: http://www.glazman.org/weblog/dotclear/index.php?post/2012/02/09/CALL-FOR-ACTION:-THE-OPEN-WEB-NEEDS-YOU-NOW. [Accessed: 26-Apr-2013].

[25] P.-P. Koch, “CSS vendor prefixes considered harmful,” *QuirksBlog*, 2010. [Online]. Available: http://www.quirksmode.org/blog/archives/2010/03/css_vendor_pref.html. [Accessed: 06-Apr-2013].

[26] H. Sivonen, “Vendor Prefixes Are Hurting the Web,” *Henri Sivonen’s pages*, 2011. [Online]. Available: http://hsivonen.iki.fi/vendor-prefixes/. [Accessed: 24-Apr-2013].

[27] P. Irish, “Vendor prefixes are not developer-friendly,” *Paul Irish*, 2012. [Online]. Available: http://paulirish.com/2012/vendor-prefixes-are-not-developer-friendly/. [Accessed: 24-Apr-2013].

[28] J. Wilcox, “Microsoft limits XML in Office 2003,” *CNET News.com*, 2003. [Online]. Available: http://web.archive.org/web/20050922005808/http://news.zdnet.com/2100-3513_22-996528.html. [Accessed: 13-Apr-2013].

[29] K. Noyes, “Look Who’s Working on Linux Now: Microsoft,” *PCWorld*, 2012. [Online]. Available: http://www.pcworld.com/businesscenter/article/253100/look_whos_working_on_linux_now_microsoft.html. [Accessed: 23-Apr-2013].

[30] Adobe Systems Inc., “Adobe® BrowserLab,” *Adobe® BrowserLab*, 2011. [Online]. Available: https://browserlab.adobe.com/en-us/index.html#. [Accessed: 24-Mar-2012].

[31] Spoon.net, “Browser Sandbox - Run any browser instantly from the web,” *Spoon.net App Virtualization*, 2013. [Online]. Available: http://spoon.net/browsers/. [Accessed: 24-Apr-2013].

[32] Browsershots.org, “Check Browser Compatibility, Cross Platform Browser Test,” *Browsershots.org*, 2013. [Online]. Available: http://browsershots.org/. [Accessed: 24-Apr-2013].

[33] CrossBrowserTesting.com, “Cross Browser Testing. Pick an OS - Pick a Browser - Test website,” *Crossbrowsertesting.com*, 2013. [Online]. Available: http://crossbrowsertesting.com/. [Accessed: 24-Apr-2013].

[34] Microsoft.com, “Expression Web SuperPreview,” *Microsoft Expression - Features and Articles*, 2013. [Online]. Available: http://expression.microsoft.com/en-us/dd565874.aspx. [Accessed: 24-Apr-2013].

[35] Litmus.com, “Landing Page and Browser Tests,” *Litmus*, 2013. [Online]. Available: http://litmus.com/page-tests. [Accessed: 24-Apr-2013].

[36] Oracle, “Oracle VM VirtualBox,” *Oracle VM VirtualBox*, 2013. [Online]. Available: https://www.virtualbox.org/. [Accessed: 24-Apr-2013].

[37] QEMU, “QEMU,” 2013. [Online]. Available: http://wiki.qemu.org/Main_Page. [Accessed: 24-Apr-2013].

[38] VMware, “VMware Fusion: Run Windows on Mac for Desktop Virtualization,” *VMware Fusion: Run Windows on Mac for Desktop Virtualization*, 2013. [Online]. Available: http://www.vmware.com/products/fusion/overview.html. [Accessed: 24-Apr-2013].

[39] Xen.org, “What is Xen?,” Welcome to xen.org, home of the Xen® hypervisor, the powerful open source industry standard for virtualization., 2013. [Online]. Available: http://www.xen.org/. [Accessed: 24-Apr-2013].

[40] Microsoft.com, “Windows Virtual PC: Home Page,” 2013. [Online]. Available: http://www.microsoft.com/windows/virtual-pc/default.aspx. [Accessed: 24-Apr-2013].

[41] W3C, “Standards - W3C,” *W3C*, 2012. [Online]. Available: http://www.w3.org/standards/. [Accessed: 24-Apr-2013].

[42] W3.org, “The First Acid Test,” *The First Acid Test - W3*, 1998. [Online]. Available: http://www.w3.org/Style/CSS/Test/CSS1/current/test5526c.htm. [Accessed: 24-Apr-2013].

[43] Acidtests.org, “The Second Acid Test,” *The Second Acid Test*, 2005. [Online]. Available: http://acid2.acidtests.org/#top. [Accessed: 23-Apr-2013].

[44] Acidtests.org, “The Acid3 Test,” *The Acid3 Test*, 2008. [Online]. Available: http://acid3.acidtests.org/. [Accessed: 24-Apr-2013].

[45] Webstandards.org, “The Web Standards Project,” *The Web Standards Project*, 2013. [Online]. Available: http://www.webstandards.org/. [Accessed: 22-Apr-2013].

[46] Wikipedia.org, “Comparison of layout engines (Cascading Style Sheets),” *Wikipedia, the free encyclopedia*, 2013. [Online]. Available: http://en.wikipedia.org/wiki/Comparison_of_layout_engines_%28Cascading_Style_Sheets%29. [Accessed: 24-Apr-2013].

[47] How To Create, “Browser speed comparisons,” *Browser speed comparisons*, 2010. [Online]. Available: http://www.howtocreate.co.uk/browserSpeed.html. [Accessed: 30-Mar-2013].

[48] S. Villinger, “What’s the fastest browser? Maybe you’re measuring wrong,” *ITworld*, 2012. [Online]. Available: http://www.itworld.com/software/266362/whats-fastest-browser-maybe-youre-measuring-wrong?page=0,2. [Accessed: 13-Apr-2013].

[49] O. Thereaux, “The W3C QA Toolbox - Validators, checkers and other tools for Webmasters and Web Developers,” *W3C Blog*, 2012. [Online]. Available: http://www.w3.org/QA/Tools/. [Accessed: 24-Apr-2013].

[50] A. Grosskurth and M. W. Godfrey, “Architecture and evolution of the modern web browser,” *Can. David R Cheriton Sch. Comput. Sci. Univ. Waterloo*, 2006.

[51] A. Grosskurth and M. W. Godfrey, “A reference architecture for web browsers,” in *Software Maintenance, 2005. ICSM’05. Proceedings of the 21st IEEE International Conference on*, 2005, pp. 661–664.

[52] G. Duncan, “Can Mozilla survive without Google?,” *Digital Trends*, 2011. [Online]. Available: http://www.digitaltrends.com/computing/mozillas-delicate-dance-with-google/. [Accessed: 23-Apr-2013].

[53] P. Windrum, Back from the brink: Microsoft and the strategic use of standards in the Browser Wars. MERIT, 2000.

[54] I. Oshri et al., “The rise of Firefox in the web browser industry: The role of open source in setting standards,” *Bus. Hist.*, vol. 52, no. 5, pp. 834–856, 2010.

[55] L. Y. Li and G. D. Chen, “A Web Browser Interface to Manage the Searching and Organizing of Information on the Web by Learners,” *Subscr. Prices Ordering Inf.*, p. 86, 2010.

[56] G. Huang et al., “Towards service composition middleware embedded in web browser,” in *Cyber-Enabled Distributed Computing and Knowledge Discovery, 2009. CyberC’09. International Conference on*, 2009, pp. 93–100.

[57] A. Bohannon and B. C. Pierce, “Featherweight Firefox: formalizing the core of a web browser,” in *Proceedings of the 2010 USENIX conference on Web application development*, Berkeley, CA, USA, 2010, pp. 11–11.

[58] J. P. Espada et al., “Extensible architecture for context-aware mobile web applications,” *Expert Syst. Appl.*, vol. 39, no. 10, pp. 9686–9694, 2012.

[59] M. Šilić et al., “Security vulnerabilities in modern web browser architecture,” in MIPRO 2010 - 33rd International Convention on Information and Communication Technology, Electronics and Microelectronics, Proceedings, 2010, pp. 1240–1245.