The Push for Web Standards:

Developing Uniformly-Accessible Web Sites

Matthew J. Ring
November 11, 2003

Introduction

Statement of the Problem

Although the push for web standards has been around since 1998, internet developers and designers have been extremely slow in adapting this highly beneficial and efficient concept into their work methods. Leading browsers were also slow to adapt a uniform support for web standards, but now the technology has emerged to a point where "Best viewed with Browser X" is no longer a necessary option.

The Importance of Standards

This proposal studies the effects of "inaccessible" web sites – or web sites that only cater to certain browsers - and its effect on the internet public. In addition, we will examine how the push for web standards has grown considerably and the benefits that current and future internet developers will obtain from supporting the use of web standards in their projects.

We will also review the current methods of web development used by many programmers. Since most people don't code sites by hand, it is important to look at how today's WYSIWYG[1] authoring tools generate web pages and see which ones best promote well-structured, accessible code. We will explain the meaning and use of accessibility later in the document.

Sources

Most of the information and data used in this proposal is web-based, coming from e-zines (electronic magazines) and article reviews. In addition, we refer somewhat heavily to the ideas discussed in *Designing with Web Standards* by Jeffrey Zeldman, which is a book designed to introduce novice and veteran webmasters to the world of web standards. It also gives a detailed account of where today's browsers are in terms of supporting standards.

[1] Acronym: "What You See Is What You Get" – used to describe web authoring programs such as Microsoft FrontPage and Macromedia Dreamweaver

Details of the Problem

The idea of web standards has been around since the mid-1990s, but because of slow browser support and general hesitation to accept change, developers have continued to use old, outdated methods in the creation of their web sites.

Two common methodologies are still used today in creating web sites. On one end, developers can design their sites so that even the oldest browsers view the sites the same as new browsers. Unfortunately, this leads to a lot of redundant coding and re-coding. If something works in four browsers, but doesn't quite work in the fifth, the idea is scrapped. The other way is to design for only one browser, one that a majority of visitors use. "In a misguided effort to reduce expenses, an increasing number of sites [were] designed to work only in Internet Explorer, and sometimes only on the Windows platform, thus locking out 15-25% of [a company's] potential visitors and customers" (Zeldman 34).

In addition to slow browser support, web developers approached the idea of web standards with skepticism and hesitation. The idea that current popular coding methods would need to be unlearned in order to code to web standard turned a lot of people off, which is why, even today, several commercial and organizational web sites still use old, outdated methods.

Fortunately, using web standards is much easier than previously thought. Conforming to web standards means that developers must completely separate content from design. This way, the content will be the same in any browser that views it, but the developer has the power to make the site "look and feel" however they wish in any browser *without* changing the content.

This will be covered in detail later in the proposal, but we must first look at the current problems facing web developers who use outdated methods.

Creating Inconsistent (and unprofessional) Web Sites

For the unfortunate few who designed commercial or organizational web sites for disabilities groups, it was often common to sacrifice any sense of design so that the content would be easy enough for a screen reader program to read. Therefore, any sighted person visiting the sight would be turned off by the unprofessional appearance of the company or group. It was a problem where design and content intermixed. Without the idea of using web standards, there seemed to be no simple, viable solution.

One solution did come about when internet scripting became popular. Developers used browser-sniffing programs – code that would determine what kind of browser the visitor was using and return a specific web page to them. This method seemed to work at first. For sight-disabled people using a text-only browser such as Lynx and a screen reader program, the browser-sniffer would send that user a page created with little or no visual design. For everybody else, they would get a beautifully crafted, professional site. However, there was no 100% guarantee that everyone would be handed the best page their particular browser could handle.

Doubling (or Tripling) the Work for Developers

The above solution may seem good for the end-user, but what about the work of the developer? If each page of a web site requires two separate pages (one visually-pleasing and one dressed down) then the developer must maintain twice the number of pages he/she originally had to deal with. For simple spelling or wording changes, this may seem more of a small annoyance than a problem, but suppose a new department is being developed. That means one or more new pages must be designed for that department. In addition, those new pages must be redesigned and tested to work on less-capable browsers. As you can see, this can make for quite a headache.

Alienation of Some Visitors

Let us return to the issue of sight- or hearing-disabled visitors. These visitors often rely on separate programs to "read" pages to them. This may not seem like a big deal to most commercial web sites, but what about medical sites? Disabled visitors need just as much access to information on these sites as perfectly healthy visitors.

By using proprietary code designed for specific browsers (such as IE or Netscape), visitors viewing the site through other means can risk having the site breaking for them, or displaying code fragments not meant to be seen by the audience. How do you think a sight-disabled person would react if his/her screen reader said the following: "`Take one pill with food. [inline] [usemap] To consult doctor for more questions, please contact us at the following number: [image].`" I would be truly frustrated and concerned about what to do if I had questions.

Case Study: A Prime Example

Let us look at Microsoft homepage, located at http://www.microsoft.com. This site is set up to detect a visitor's browser using the browser-sniffing technique described above. If the visitor is using Microsoft's Internet Explorer, he/she receives a nicely illustrated, professional web page with menus for easy navigation (see Figure 1.1).

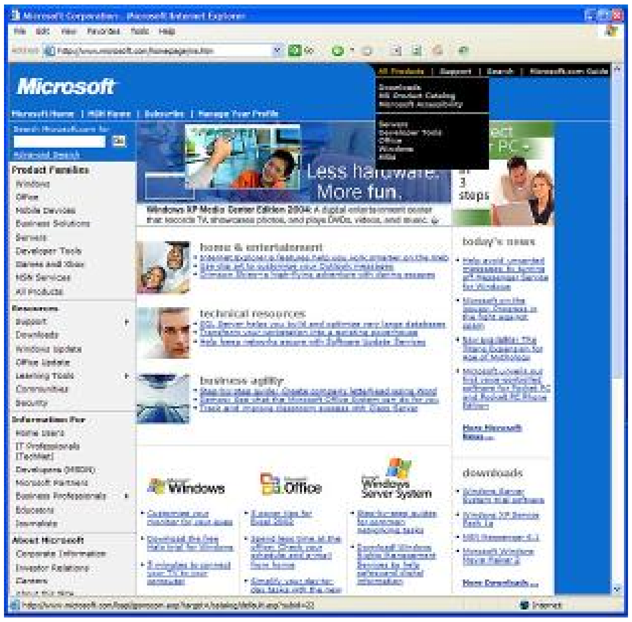


Figure 1.1: Microsoft's Site viewed in IE6

However, if the visitor is using Netscape or Mozilla, the browser is fed another page designed to work for Netscape Navigator 4.x (which was released in 1994. See Figure 1.2). This page does not include drop down menus for easy navigation. Nor does it include some of the header illustrations shown on the IE page, indicating separate sections.

For example, on the IE-generated page, pictures appear to the left of the links on the center of the page. Also, the links at the bottom center of Figure 1.1 have illustrated headings with Microsoft's signature logo. However, the pictures in Figure 1.2 are removed on the non-IE generated page and the illustrated headings are replaced by simple text, such as "Windows" or "Office." It's not that this version of Netscape cannot handle the type of things Microsoft is attempting to use (i.e. menus, image headings), it's that Microsoft's developers are taking a "one-size-fits-all" approach to visitors using Netscape's browser.

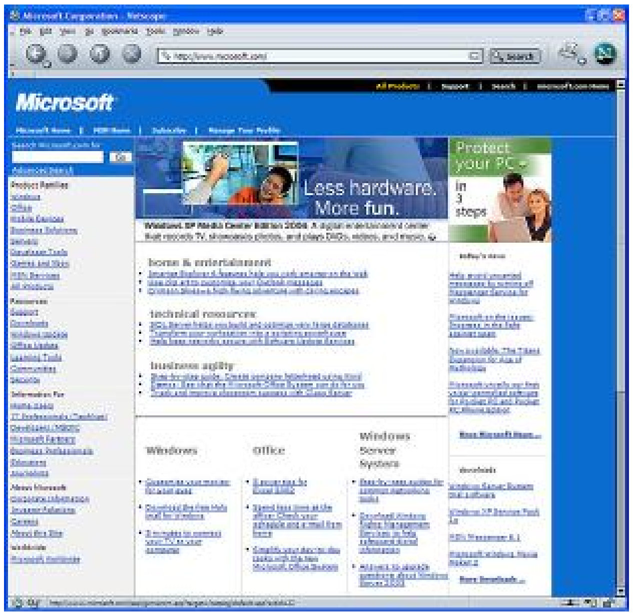
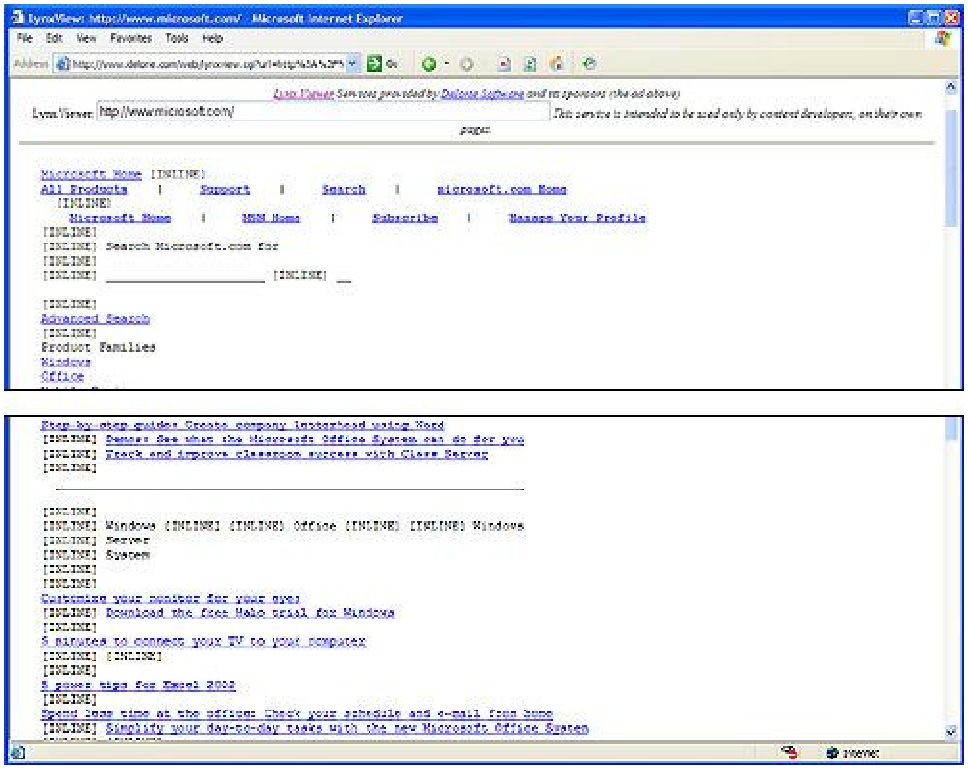


Figure 1.2: Microsoft's site viewed in Netscape 7.1

Let’s take this a step further and see how the site looks in text-only browser. In this case, we chose to use Lynx Viewer[2], an online emulator of the Lynx text browser. The output of the site is shown in Figure 1.3. As you can see, it is full of code fragments, making it very difficult for a sight- or hearing-disabled user to navigate the site.



```
Lynx Viewer Services provided by Delorie Software and its sponsors (the ad above)
Lynx Viewer: http://www.microsoft.com/    This service is intended to be used only by content developers, on their own pages.

Microsoft Home [INLINE]
All Products    |    Support    |    Search    |    microsoft.com Home
  [INLINE]
    Microsoft Home    |    MSN Home    |    Subscribe    |    Manage Your Profile
[INLINE]
[INLINE] Search Microsoft.com for
[INLINE]
[INLINE] ____________________ [INLINE] __

[INLINE]
Advanced Search
[INLINE]
Product Families
Windows
Office
```

```
Step-by-step guides Create company letterhead using Word
[INLINE] Demos: See what the Microsoft Office System can do for you
[INLINE] Track and improve classroom success with Class Server
[INLINE]

  ______________________________________________________________

[INLINE]
[INLINE] Windows [INLINE] [INLINE] Office [INLINE] [INLINE] Windows
[INLINE] Server
[INLINE] System
[INLINE]
[INLINE]
Customize your monitor for your eyes
[INLINE] Download the free Halo trial for Windows
[INLINE]
5 minutes to connect your TV to your computer
[INLINE] [INLINE]
[INLINE]
5 power tips for Excel 2002
[INLINE]
Spend less time at the office: Check your schedule and e-mail from home
[INLINE] Simplify your day-to-day tasks with the new Microsoft Office System
```

Figure 1.3: Microsoft’s web site in an emulated text-only browser

Proposed Plan

Before we can begin talking about how the use of web standards resolves these problems and more, we must first remove the general misconceptions surrounding this concept.

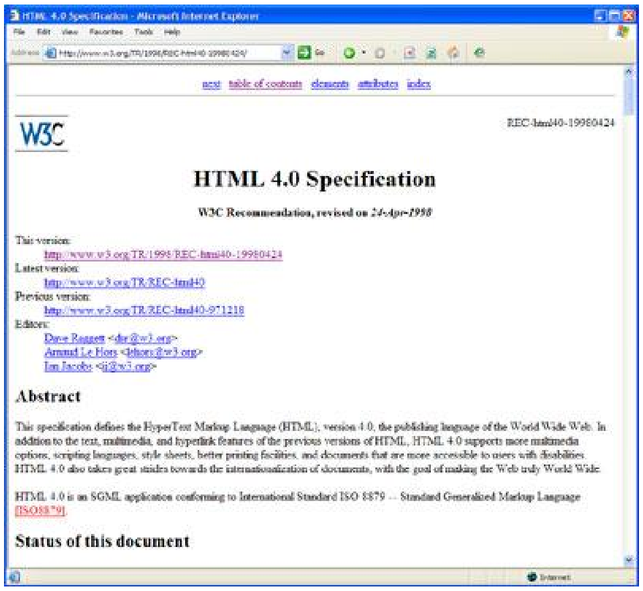
Cleaning Up the Misconceptions

“[An] obstacle to widespread acceptance of web standards is the mistaken belief that standards will somehow diminish creativity, tie the developer’s hands, or result in lessened user experiences…” (Zeldman 97). Zeldman sums up these misconceptions nicely, and he’s 100% correct. But since web standards are meant to promote the exact

[2] found at http://www.delorie.com/web/lynxview.html

opposite of this idea, why is there a general misconception? Zeldman attributes it to the early developers who tried using the technology back in the day of version 4.0 browser, when browser support was almost non-existent. Who can blame them; if something new didn't work right at the start, who was going to wait around for it to gain supportability?

The idea that using web standards somehow diminished creativity was actually the blame of the proponents of web standards themselves. Although very well-educated in standards coding, there was a major lack of design ability that would truly inspire other developers to pick up the practice. For example, Figure 1.4 shows a web page last revised in 1998 from the World Wide Web Consortium (W3C) organization, the very organization that developed a set of web standards. From the developer's perspective, the idea of using web standards meant their creativity would force them to revert back to the stone age of internet design.


HTML 4.0 Specification - Microsoft Internet Explorer
File Edit View Favorites Tools Help
next table of contents elements attributes index
W3C
Internet

REC-html40-19980424

HTML 4.0 Specification

W3C Recommendation, revised on *24-Apr-1998*

This version:
http://www.w3.org/TR/1998/REC-html40-19980424
Latest version:
http://www.w3.org/TR/REC-html40
Previous version:
http://www.w3.org/TR/REC-html40-971218
Editors:
Dave Raggett <dsr@w3.org>
Arnaud Le Hors <lehors@w3.org>
Ian Jacobs <ij@w3.org>

Abstract

This specification defines the HyperText Markup Language (HTML), version 4.0, the publishing language of the World Wide Web. In addition to the text, multimedia, and hyperlink features of the previous versions of HTML, HTML 4.0 supports more multimedia options, scripting languages, style sheets, better printing facilities, and documents that are more accessible to users with disabilities. HTML 4.0 also takes great strides towards the internationalization of documents, with the goal of making the Web truly World Wide.

HTML 4.0 is an SGML application conforming to International Standard ISO 8879 -- Standard Generalized Markup Language [ISO8879].

Status of this document

Figure 1.4: Technical Report page from W3C

Thankfully, there are now several web sites in existence that show off the actual creative freedom that web standards allows for developers and shows that users can enjoy visually-appealing and welcoming web sites. Of course, the W3C has had a nice facelift or two since 1998, making it a little more appealing and easier on the eyes.

Make Standards Work for the Developer

Here, we will discuss solutions to the problems described above. In our introduction, we discussed the technique of browser-sniffing, where several web sites are created and the visitor's browser type and version determines which one he/she will see. We also explained how this makes the developer's work double, or even triple the amount it used to be.

With the introduction to web standards came the concept of completely separating content (i.e. structure) from design. In this way, HTML[3] (or XHTML) would be used to code the actual content of the web pages (i.e. headings, text, header illustrations, etc.). Then, we would create a CSS[4] document that would control the look and feel of our web site. This way, we are setting ourselves up for much less maintenance work in the future.

First, we notice that we only need to create one CSS document for an entire web site. This makes it extremely efficient for a web developer to change anything from color effects to typography or even placement of objects with only one file. Beforehand, developers would have to sift through several lines of code ***per page*** to accomplish this same task.

Second, web standards allows for "alternative style sheets," which means developers can create several documents depicting different "looks and feels" of their web site without even touching the (X)HTML document. With this, we can send one style sheet to users with modern browsers, another style sheet to those with older browsers, and no style sheet whatsoever to text-only browsers or handheld devices. This way, we don't have to worry about needless design code, such as `[INLINE]` and `[USEMAP]`, interrupting our HTML documents since our HTML code no longer controls the design.

Make Standards Work for the User

Now that we've shown the power of using web standards, let's see its effect on the end-user, the actual visitor of your web pages. After all, the number one priority of commercial web sites is to have an effective and professional web site that visitors and potential customers can easily navigate and use effectively.

We've already mentioned how different users will be able to view the same site in the way that best compliments their browser (via the developer's coded style sheets). Another coding tool gaining popularity is the user's ability to manually select different style sheets for a web site. Some sites allow the user to select between two or three typographical web sites. Other sites allow users to completely change the look and feel of an entire web site. "These style sheets can be selected by the visitor as alternatives to a preferred style sheet. This allows the visitor to personalize a site and choose his or her favorite scheme." (Sowden 2)

[3] HyperText Markup Language (or eXtensible HyperText Markup Language)
[4] Cascading Style Sheets – a programming language can describes design attributes of HTML tags

Update Existing Web Sites

It's understandable that developers don't have endless amounts of time to completely redesign a site. However, it's important to do *something*. Make a series of small changes over a given timeline until there is time to do the big overhaul.

- Clean up your existing HTML code, even if you didn't directly hand-code it.
- Convert some proprietary methods to standards or phase them out.
- Begin using a Style sheet. Gradually modify your existing code to rely more on the style sheet. Get rid of all those annoying `font` tags.

As newer browsers are released, their support for web standards will continue to grow and support for outdated techniques will (hopefully) fade out. "A common feature of standards-compliant browsers is that they are much stricter about syntax. When a page breaks in one of these browsers, before you start blaming 'bugs' in the program, take a look at your HTML." (Itoh 3)

Choose Standard-Friendly Authoring Tools

With several authoring tools like Macromedia's Dreamweaver and Microsoft's FrontPage on the market, it's not surprising that many web developers create web sites using these tools. Because the authoring tools auto-generate much of the underlying code for the developer, this can free up a considerable amount of time for other work.

However, it's important to realize that, though authoring tools are designed to construct the underlying code for you, not all of them doing a good job at it, and they should. "If web builders must rely on software to do the grunt work of site building, the tools they use should be designed to generate standards-compliant code *by default.*" (Schmitt 2)

In the following sections, we will look at a couple authoring tools on the market today and see how well-structured and standards-compliant the code they generate really is.

Microsoft FrontPage 2002

Popularized by its easy-to-use interface which resembles Microsoft Word, FrontPage is seen as the tool of choice for mostly amateur and some professional developers. However, taking one look at the source code generated after five minutes of coding can make your head start to hurt immediately.

Though FrontPage's front-end interface is relatively simple, its underlying code-generators still use proprietary markup and bloat some sections. Opening `P` tags do not always have closing `P` tags, and it’s fairly difficult to set up a style sheet without manually coding it in.

Although I applaud Microsoft for making huge improvements since previous versions, FrontPage 2002 still has some problems putting in more code than is necessary. In addition, FrontPage still has a few problems writing code that works well for Internet Explorer, but not so well for anything else. These problems have also been worked on since previous versions, but not quite enough.

All in all, FrontPage is a good authoring program to get your feet wet or to design a simple web site. But to make sure your sites conform to standards, it’s almost impossible not to spend half the time cleaning up the source code.

Macromedia Dreamweaver MX 2004

Dreamweaver has quickly gained popularity over Microsoft’s FrontPage in the last few years, and for good reason. Along with its support for non-Microsoft programming languages such as PHP, JSP and ColdFusion, the product’s developers have made a lot of effort to keep their product conforming to web standards, including improved CSS integration. “The CSS editing in Dreamweaver MX 2004 is a lot more intuitive than [previous versions], and... if you are having conniptions about developing CSS for cross-browser compatibility, the browser check feature can help immensely.” (Kay 6)

What makes Dreamweaver stand out so much above the rest is that, in 2001, Macromedia engineers met with members of WaSP (Web Standards Project) to create the Dreamweaver Task Force. This group was formed to make Macromedia’s product the most standards-compliant on the market, focusing on valid markup and accessibility. Macromedia released Dreamweaver MX in May 2002, offering much improved standards compliance and accessibility over its previous versions.

Conclusion

Where to Go From Here

If you are a consultant, or develop web sites for a number of different clients, *do* make standards compliancy a selling point in your service. *Don’t* try to cram it down your clients’ throats. For one thing, many clients aren’t familiar with what “standards-compliancy” means. They haven’t just read this paper like you have, and aren’t interested in hearing a back story about your views on the benefits of web standards. “The biggest mistake you can make is to broadcast to your clients that you are now doing [standard-compliant] design. [It] sounds as if you are using them as an experiment.” (Kise 2)

Start coding with standards in mind. Read what information you can about standards and accessibility. Convert an existing web site to standards-compliant or start one from scratch. Remember, the best way to learn is by doing. The biggest stepping stone is to rid yourself of old, proprietary coding habits. Once you do, the quality of your sites will be all the more impressive.

Bibliography

Itoh, Makiko. "Preparing for Standard-Compliant Browsers, Part 1." April 2000. *Digital Web Magazine*. 9 Sep. 2003. <http://www.digital-web.com/tutorials/tutorial_2000-4.shtml>.

Kay, Michael. "Studio MX 2004 Overview." 25 Aug. 2003. *Webmonkey: The Web Developer's Resource.* 11 Sep. 2003. <http://hotwired.lycos.com/webmonkey/03/29/index0a.html?tw=multimedia>.

Kise, Greg. "CSS Talking Points: Selling Your Clients Web Standards." 6 July 2001. *A List Apart: For People who Make Websites.* 22 Sep 2003. <www.alistapart.com/stories/csstalking/>.

Schmitt, Christopher. "Accessibility & Authoring Tools." 22 March 2002. *A List Apart: For People who Make Websites.* 22 Sep 2003. <www.alistapart.com/stories/tools/>.

Sowden, Paul. "Alternative Style" Working with Alternative Style Sheets." 2 Nov. 2001. *A List Apart: For People who Make Websites.* 10 Nov 2003 <www.alistapart.com/articles/alternate/>.

Zeldman, Jeffrey. *Designing with Web Standards.* 1st ed. Indianapolis, ID: New Riders Publishing., 2003.

Contribute to web-push-libs/web-push-php development by creating an account on GitHub.Â Payload is encrypted according to the Message Encryption for Web Push standard, using the user public key and authentication secret that you can get by following the Web Push API specification. Internally, WebPush uses the WebToken framework or OpenSSL to handle encryption keys generation and encryption. For web push the audience is the push service, so we set it to the origin of the push service. The exp value is the expiration of the JWT, this prevent snoopers from being able to re-use a JWT if they intercept it. The expiration is a timestamp in seconds and must be no longer 24 hours.Â (This is why the web-push library needed an email address). Just like the JWT Info, the JWT Data is encoded as a URL safe base64 string. The third string, the signature, is the result of taking the first two strings (the JWT Info and JWT Data), joining them with a dot character, which we'll call the "unsigned token", and signing it. Refer browser-push repo for guide lines to implement web push notification for your web-app with your own back-end. It explains with examples how you can add web push notification support for your web application without any third party services. Share. Share a link to this answer.Â As of now GCM only works for chrome and android. similarly firefox and other browsers has their own api. Now coming to the question how to implement push notification so that it will work for all common browsers with own back end. You need client side script code i.e service worker,refer(Google push notification). Though this remains same for other browsers. 2.after getting endpoint using Ajax save it along with browser name. All major browsers support Web Push. Android devices may also receive Web Push notifications in addition to notifications from apps as long as they can connect to the push gateway servers. Below is a list of platforms and browsers that offer support for web push notifications. Browser Support by Operating System. Incognito Mode, Private Browsing Mode, and Guest Browser Mode do not support Web Push. HTTP Web Push. Notifications API and Push API are two different things which should be noted by now. Later on in this blog series we will cover how push on iOS can work, since there are some limitations using Mobile Safari.Â Christian Liebel is consultant at Thinktecture, focuses on web standards and Progressive Web Applications, and is Thinktecture's representative at the W3C. GÃ¸ran Homberg. GÃ¸ran Homberg is Consultant at Thinktecture and works with our clients on requirements engineering, project definition, and project management.

Web Push Notifications or Browser Notifications are clickable rich content messages sent to your device by a website or a web app. Web Push notifications can be delivered to your device, mobile or desktop, even when the user is not on your website.Â Web Push notifications can be delivered to your user's device (mobile or desktop) even when the user is not on your website. These notifications can only be sent to users who have opted-in to receive these notifications. Web push notifications or Browser push notifications are supported by Chrome, Firefox, Safari, Opera, and Edge on Desktop and by Chrome, Firefox, Opera on Android Mobile. It is important to note that iOS does not support web push notifications yet. Contribute to web-push-libs/web-push-php development by creating an account on GitHub.Â WebPush Requirements Installation Usage Full examples of Web Push implementations Authentication (VAPID) Reusing VAPID headers Notifications and default options TTL urgency topic batchSize Server errors Payload length, security, and performance Why is it more secure? Why does it decrease performance? How can I disable or customize automatic padding? Customizing the HTTP client Common questions Is there any plugin/bundle/extension for my favorite PHP framework? Is the API stable? What about security? HTTP Web Push. Notifications API and Push API are two different things which should be noted by now. Later on in this blog series we will cover how push on iOS can work, since there are some limitations using Mobile Safari.Â Christian Liebel is consultant at Thinktecture, focuses on web standards and Progressive Web Applications, and is Thinktecture's representative at the W3C. GÃ¸ran Homberg. GÃ¸ran Homberg is Consultant at Thinktecture and works with our clients on requirements engineering, project definition, and project management. Web pushes are permission-based, alert-style notifications that have long turned into a powerful tool of email marketing. They allow to deliver short quick messages and reach the customers on a different level. If you havenâ€™t used push notifications before because you think the technology requires too much technical knowledge and skills, wait no more.Â 1. Single Opt-In: a one-step permission request. Consists of a standard prompt with default text and buttons, without editing options. A click on Allow automatically equals subscription confirmation. 2. Double Opt-In: a two-step permission request. There are two types of Double Opt-In: Standard Double Opt-In: consists of a customized prompt (for which you can edit text, buttons, and color) that appears first. Web Push â€“ an informal term referring to the process or components involved in the process of pushing messages from a server to a client on the web. Push Service â€“ a system for routing push messages from a server to a client.Â When the user grants permission for Push on your site, you can then subscribe the app to the browser's push service. This creates a special subscription object that contains the "endpoint URL" of the push service, which is different for each browser, and a public key (see the example below).Â The Web Push protocol is the formal standard for sending push messages destined for the browser. It describes the structure and flow of how to create your push message, encrypt it, and send it to a Push messaging platform.