Web Development Software Development

Adrian Kosmaczewski

2007-08-02

I have been developing web applications since 1996, and I still do a fairly large amount of web development nowadays. During these years I have seen some common misconceptions and myths about web development, that ultimately have a direct impact in the usability of the system. I will outline these in this article, providing at the end my opinion on how to achieve a proper QA process.

Website Development IS Software Development

A website is a piece of software, whether the team acknowledges the fact or not; as such, all the best practices for software development apply to web design and development as well, particularly in those projects that aim to build interactive "web applications" instead of simple "brochure websites" based on some CMS, usually displaying rather static information.

Acknowledging this fact is something that many teams do not do (or want to do); in my opinion, just because JavaScript lacks a compiler it does not mean that it is not a programming language, and sooner or later, nearly all websites use JavaScript - arguably the "world's most misunderstood programming language" (Crockford). Thus, the following tasks apply even for the smallest website:

- Have a specification;
- Have a schedule;
- Have an architecture (albeit small) that separates the UI from the rest of the application; this allows the designers to change the look & feel without disturbing those working in the functionality;
- Make a prototype; preferably many of them; keep them for future reference;
- Have (human) testers;
- Automate your tests (and builds, if you use a compiled technology like Java);
- Use a bug database;
- Use version control;
- Be agile: get your client involved and have small development-testing-deployment cycles;
- Know your audience;

- Scope your target.

I will provide a small comment on the last two items in the following sections.

Know Your Audience

A very important point to know is the target audience of the website: is it a public or intranet system? What is the average age of the users? What are the legal and accessibility requirements? And last but not least, what languages should we support? These informations must come from a simple analysis of the target users, and the answers to these questions must be stated in the design document of the website project.

Knowing the target languages is fundamental, since it means choosing a target encoding for the system output; Unicode UTF-8 is a canonical choice, but for English-only websites ISO-8859-1 is more than enough. On the other side, some languages like Arabic are displayed in a "right-to-left" fashion, and this means that your application must look good and work properly in this mode too.

Finally, some clients (particularly governments and federal agencies) require providers to comply with accessibility rules, for supporting users with disabilities. The use of text-to-speech browsers, large font sizes and other media must then be taken into account during analysis, design and development of the application.

Scope Your Target

When developing web applications, one common mistake that I've seen many development teams do is to forget to set up a list of minimum target browser requirements for the application. In my experience, relying on HTML, CSS or other standards is fundamental but sadly not enough: setting a scope means creating just a short list of browsers, including screen size, browser and operating system versions, scoping the sandbox where the QA operations will be applied. This list must be part of the specification of the system being created, and must be visible and known by everyone. A sample list could be the following:

- Internet Explorer 5.5+ for Windows XP Service Pack 2
- Firefox 1.5 and all Gecko 1.8-based browsers (many different OS)
- Konqueror 3 for Linux
- Safari 2 for Mac OS 10.3
- Opera 9 (any OS)

Using the name of the layout engine also helps (like the "Gecko" reference above) since several browsers use the same layout engine, providing the same characteristics to otherwise different browsers: for example, Gecko 1.8 is used in Mozilla, Camino (for Mac OS X), Galeon, Firefox 1.5 and 2.0, and other browsers (Wolter, 2007)

It is also important to scope the minimum supported screen size:

- Minimum screen size: 1024 x 768 pixels.

Of course, if the website must support other types of devices (game consoles like the Wii or the Xbox, cell phones or PDAs), this must be specified as well, with detailed version information, and details about the supported screen sizes.

Last but not least, it is fundamental to define the "DOCTYPE" to be used in the website; this setting defines the set of valid HTML tags to be used in the page:

> Per HTML and XHTML standards, a DOCTYPE (short for "document type declaration") informs the validator which version of (X)HTML you're using, and must appear at the very top of every web page. DOCTYPEs are a key component of compliant web pages: your markup and CSS won't validate without them.

(Zeldman, 2002)

Another good practice is to add an explicit list of definitely non-supported browsers or operating systems:

- Netscape 4.x (pre-Gecko rendering engine)
- Opera (versions prior to 9)
- Internet Explorer for the Macintosh
- Mac OS versions prior to 10.3
- Linux kernels prior to 2.4

Perhaps surprisingly, these simple actions are often forgotten, which leads to confusion and lack of coherence in the QA activities of the team. Managers avoid having developers refusing to fix some incompatibility because of the lack of standards support in some browser (which always happens). On the other side, the whole team is shielded against some change in the scope; for example, the marketing team might come up with a requirement to support Opera 8, and this can have a negative impact on the schedule if it was not specified upfront, since all the CSS and JavaScript code might have to be tweaked to support the new browser.

Different Dimensions

As shown in the above paragraphs, websites are complex beasts, that can have several (often conflicting) dimensions:

- Different languages;
- "Right-to-Left" against "Left-to-Right" layouts;
- Different target browsers;
- Different target operating systems;
- Users with potential disabilities;
- Finally, the website functionality itself!

How to manage this complexity? Unfortunately, given the high fragmentation of the web market, there is not a simple answer to this problem; I think that (at least currently) the best approach is a mix of automated and manual testing procedures:

- Use standards, everywhere, all the time. No exceptions to this rule.
- Have your website tested by human beings, particularly to find spelling or grammar errors, and to verify that the websites does what it is supposed to do, in all the target browsers and operating systems.
- Use JsUnit (http://www.jsunit.net/) to unit test your JavaScript code. Run the unit tests as often as possible, usually every night.
- Use JsDoc Toolkit (http://www.jsdoctoolkit.org/) to document your code and make this documentation available to the team.
- Compress your JavaScript files using tools like Dojo ShrinkSafe (http://alex.dojotoolkit.org/shrinksafe/) to make files lighter and speed up their execution.
- Use Selenium (http://www.openqa.org/selenium/) for browser compatibility and functional testing.
- Use online validators, particularly those of the World Wide Web consortium (http://validator.w3.org/). Use standards and fix your code to eliminate warnings and errors in the validation process.
- Have your team use several browsers in their development workflow (all those that are specified as mandatory in the specs); every new feature means a unit test, a functional test, and a "smoke test" reloading the page on the browser.
- Have the developers use tools like FireBug (http://www.getfirebug.com/) and the Web Development Extension for Firefox (http://chrispederick.com/work/web-developer/).

Conclusions

Web development is a fascinating and constantly evolving activity. We are reaching a point where the technologies are getting more and more stable, secure and affordable, but I think that the web industry lacks of comprehensive and reliable integrated solutions yet.

References

Crockford, D.; "JavaScript: The World's Most Misunderstood Programming Language", [Internet] http://javascript.crockford.com/javascript.html (Accessed May 12th, 2007)

Wolter, J.; "Javascript Madness: Layout Engines", February 2007 [Internet] http://www.unixpapa.com/js/gecko.html (Accessed May 12th, 2007)

Zeldman, J.; "Fix Your Site With the Right DOCTYPE!", [Internet] http://alistapart.com/stories/doctype/ (Accessed May 12th, 2007)

Update, September 10th, 2007: I am not the only one with this opinion.