B

Using Smart Tools to Write Good Code

All software development methodologies, with no exception, do include at least one stage of testing of the code. This is because the code most programmers write, the authors of this book included, doesn't usually run perfectly from its first version.

No matter what technology or platform you choose, you'll find many tools in the market that can help you test, debug, and test the performance of your application. In this appendix, we cover a few tools that can make your life easier when writing AJAX applications.

Getting a Good Code Editor

Having a good editor is the first step for increasing your coding efficiency. The list of code editors is endless, and each programmer seems to have his or her own preferences, so we can't make a definitive recommendation here. We suggest, however, that you take a look at the following:

- **PSPad** is a freeware editor that knows how to highlight the syntax for almost any existing file format. Additional plug-ins can add integrated CSS editing functionality and spell checking. This editor is available only for the Windows platform, and you can download it from `http://www.pspad.com`.
- **SciTE** is a free source-code editor based on the **Scintilla** source-code editing component (`http://www.scintilla.org`), and it is available for Intel Win32 and Linux-compatible operating systems with **GTK+**. You can see screenshots and download it at `http://scintilla.sourceforge.net/SciTE.html`.
- **PHP Designer 2006** is a freeware **PHP IDE** with integrated debugger. Find more details about it at `http://www.mpsoftware.dk`.

There are many, many more editors around, both free and commercial, and Google may help you find some better than these listed here. Once you know exactly what you're looking for, the solution will be a few mouse clicks away.

Debugging Your Code

While the most popular debugging method for JavaScript code is by displaying messages using `alert`, a number of better tools are available when more power is needed.

The JavaScript Console available in many web browsers is perhaps the first tool you should get accustomed to. The JavaScript Console displays the errors and warnings found while running the JavaScript code in a web page. Access this feature in Firefox from the Tools | JavaScript Console menu, in Opera by opening Tools | Advanced | JavaScript console, and in Mozilla you need to click Tools | Web Development | JavaScript Console. Other web browsers may have this feature too. Take note that the JavaScript Console has different behavior in each web browser, and that the console catches errors from all opened web pages. Figures B.1 and B.2 show the same error intercepted by the JavaScript Console in Firefox and Opera.

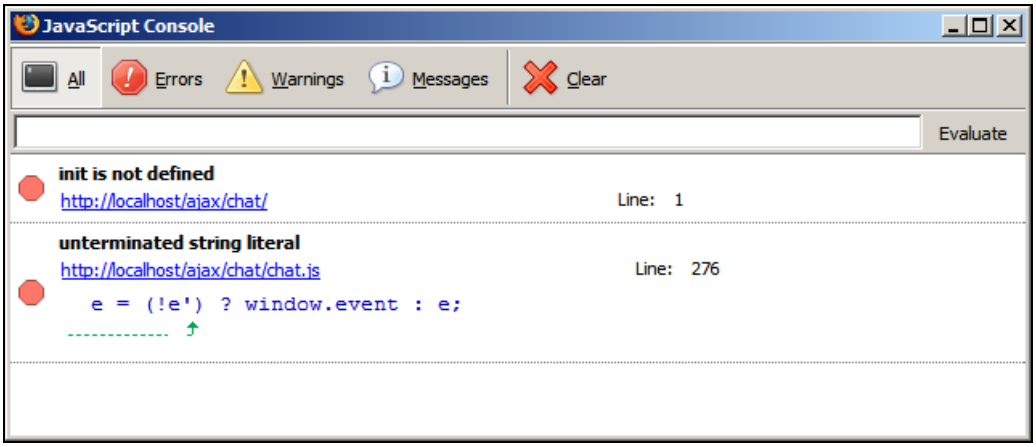


Figure B.1: The JavaScript Console in Firefox 1.5

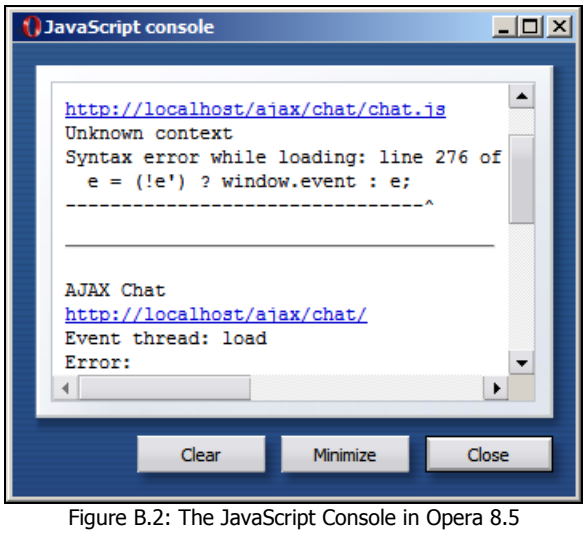


Figure B.2: The JavaScript Console in Opera 8.5

The DOM Inspector is an important tool that shows the DOM of the loaded page. This feature comes by default with Mozilla (Tools | Web Development | DOM Inspector). The DOM Inspector is packaged with Firefox as well, but it isn't installed by default. When installing Firefox, choose Custom install (instead of the default Standard install), and check the Developer Tools checkbox when asked about additional components. Amongst the very useful features of the DOM Inspector are that it allows you to dissect a page that was dynamically generated on the client, and it highlights in the web page the nodes that you select in the inspector window. Figure B.3 shows the DOM Inspector in action, inspecting the AJAX Chat web application.

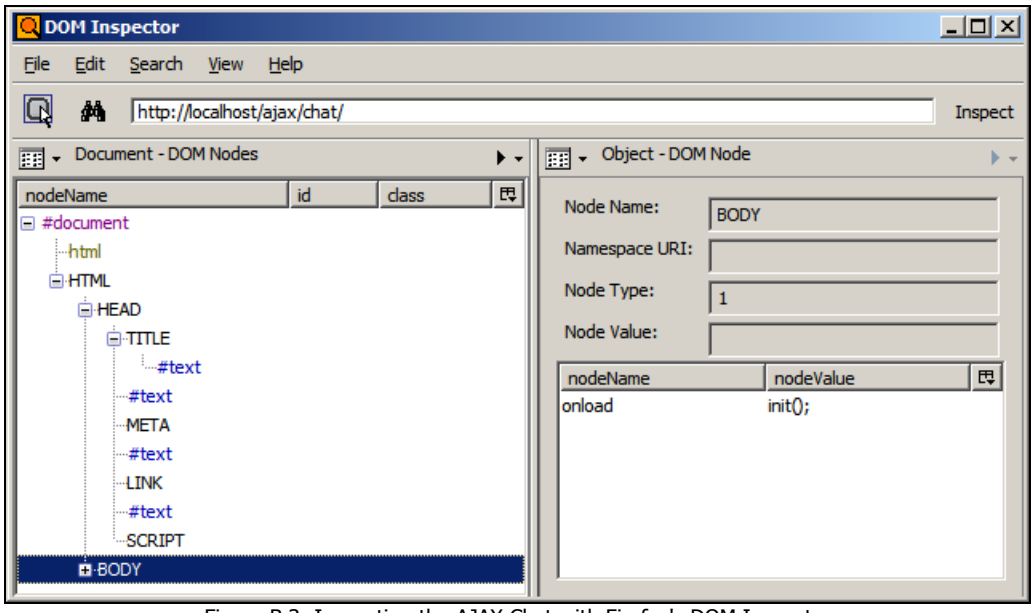


Figure B.3: Inspecting the AJAX Chat with Firefox's DOM Inspector

The Venkman Debugger is the heavy weaponry in your JavaScript debugger arsenal. It allows you effectively debug the JavaScript code, and it includes functionality that you could expect from a respectable debugger, such as setting breakpoints, allowing executing code in an interactive session window, reading local variable data, and proceeding line by line through the code. At the moment of writing, the Venkman Debugger is available for Firefox 1.0 and 1.5, and can be downloaded from `http://www.mozilla.org/projects/venkman`. Figure B.4 shows the Venkman Debugger in action, debugging the AJAX Chat application

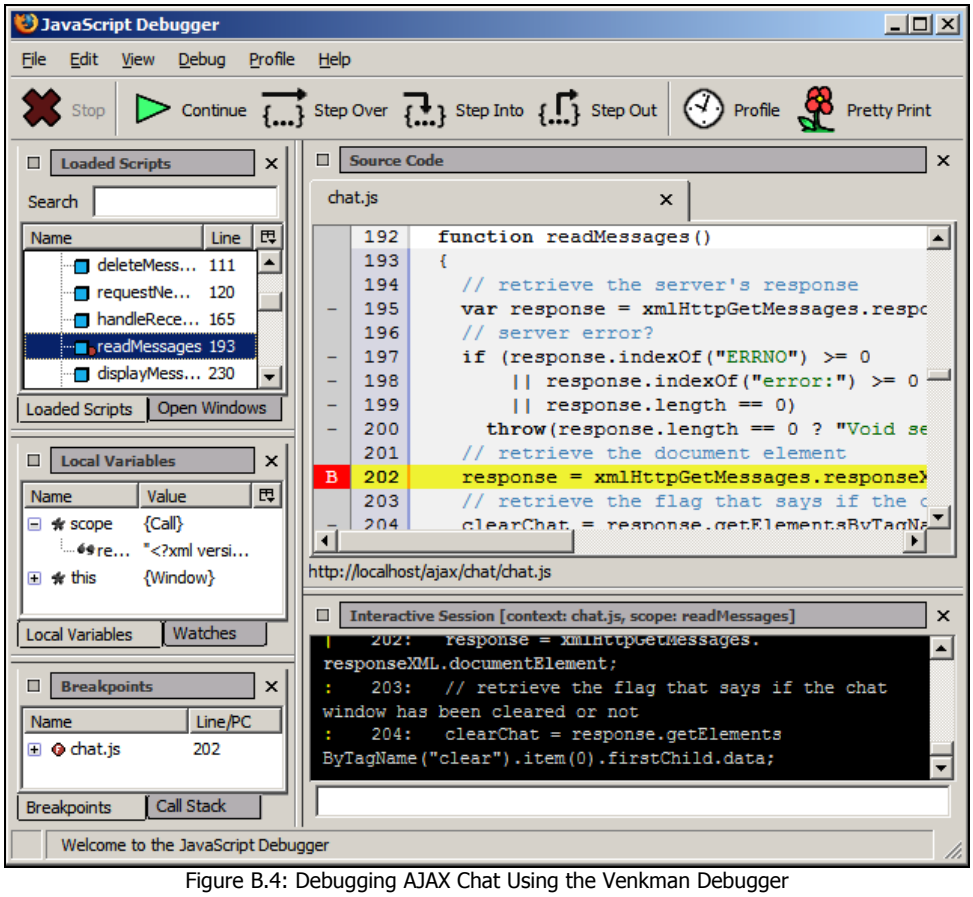


Figure B.4: Debugging AJAX Chat Using the Venkman Debugger

The Venkman Debugger also has a **code profiling** option, which you can use to analyze how many times each JavaScript code executed, and how much time it took. You can start and stop the profiling engine by clicking the Profile button on the toolbar, or by selecting the Collect Profile Data entry in the Profile menu. Also in the Profile menu you can choose to Clear Profile Data (which you may want after performing changes in the code and needing new data), and Save Profile Data. When saving profile data, it will format the output depending on the file type you choose. The profiler collects data about all JavaScript events that happen in all open browser windows, so you will need to scroll down to get at the files and functions you're interested in. Figure B.5 shows sample output from the AJAX Whiteboard demo (Please refer to the Whiteboard ebook that you can find on the book' mini-site at `http://ajaxphp.packtpub.com`).

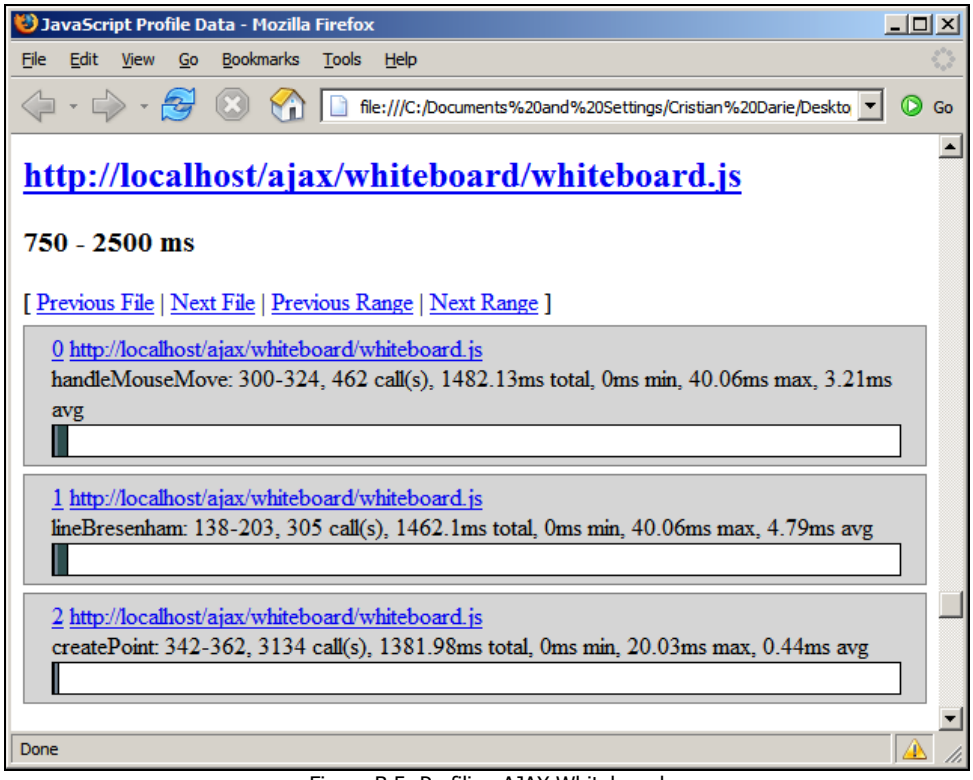


Figure B.5: Profiling AJAX Whiteboard

JavaScript debugging is possible with Internet Explorer as well, in which case you also need an external debugger. First you need to enable debugging, by un-checking Tools | Internet Options | Advanced | Browsing | Disable Script Debugging (Internet Explorer). After enabling debugging, you need a debugger tool. **Visual Studio** (see Figure B.6) is very powerful, and it supports, among many other features, debugging JavaScript code. **Visual Web Developer 2005 Express Edition** is freely available as a trial version, at `http://msdn.microsoft.com/vstudio/express/vwd`. You can also use Microsoft's **Script Debugger** (the download link is so long that's impossible to type, but you can easily search for the tool at `http://www.microsoft.com/downloads`).

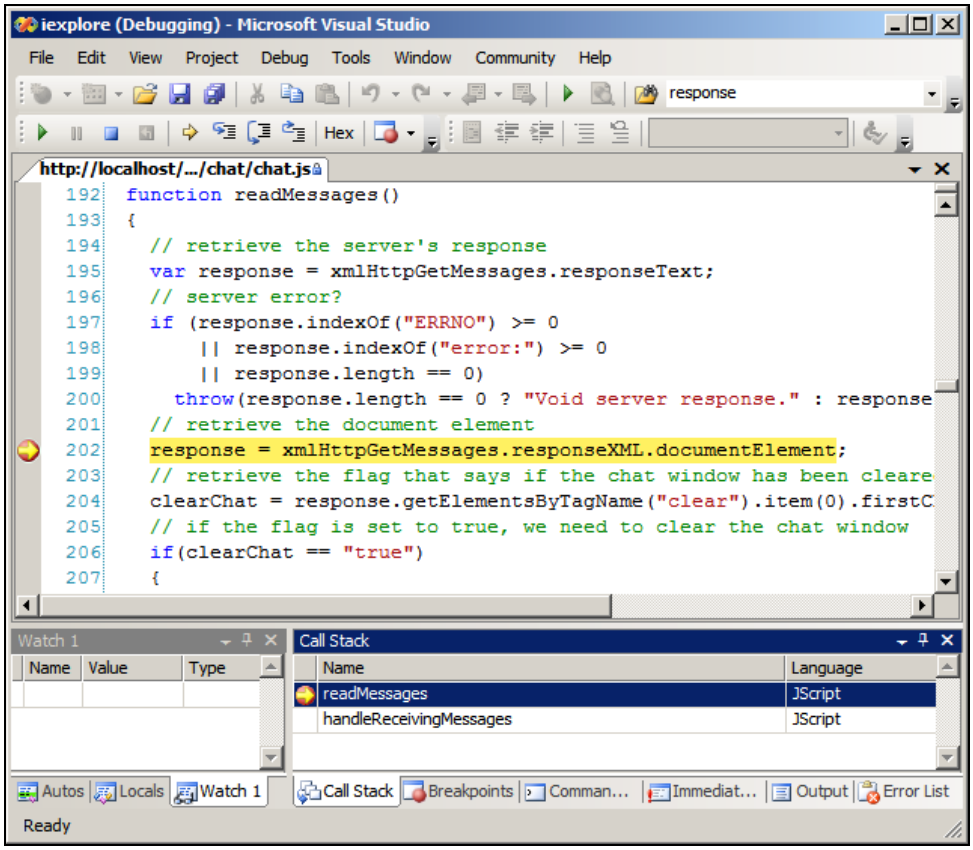


Figure B.6: Debugging AJAX Chat using Visual Web Developer 2005 Express Edition

To debug the server-side PHP script, you can use one of the many PHP editors that support this feature. The most powerful IDE for PHP code editing and debugging is **Zend Studio**, which you can buy from `http://www.zend.com`.

Other Useful Tools

Web Developer Extension is a plug-in that works with Firefox, Flock, and Mozilla and includes many features for web developers. Figure B.7 shows one of Web Developer Extension's menus. You can download the Web Developer Extension from `http://chrispederick.com/work/webdeveloper`.

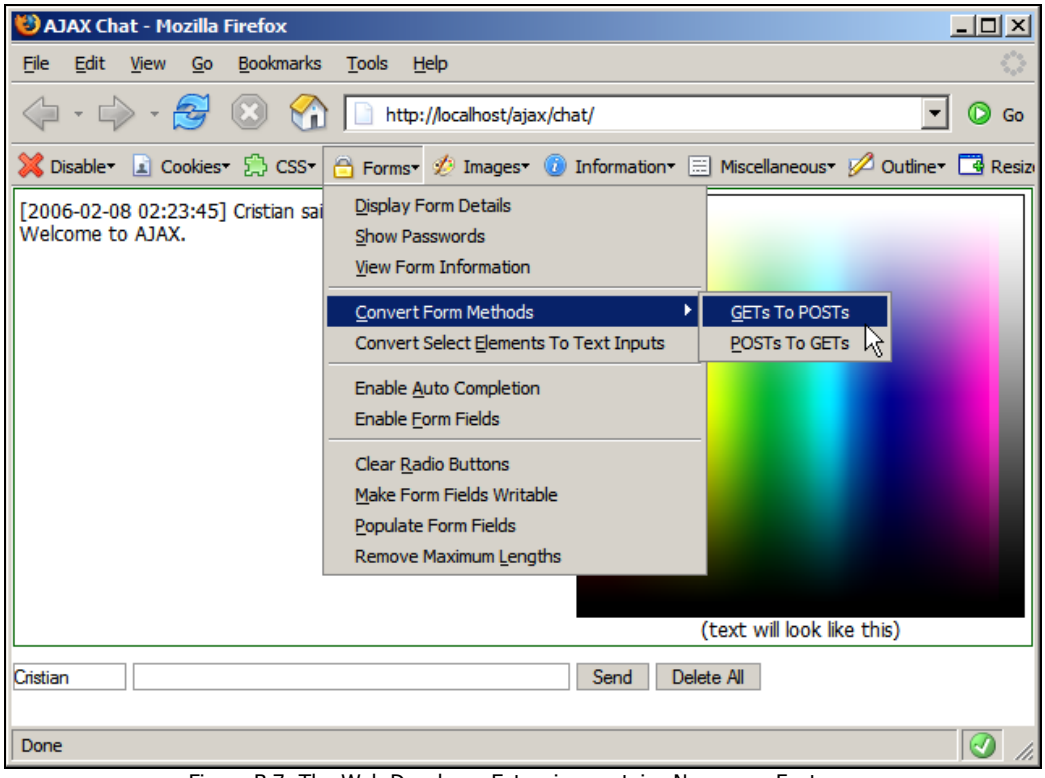


Figure B.7: The Web Developer Extension contains Numerous Features

HTML Validator is a Mozilla plug-in that validates your web pages for W3C compliance. You can download and install this plug-in from `http://users.skynet.be/mgueury/mozilla`. The validator displays a simple symbol in the browser's status bar showing if the page is valid, and if you double-click that symbol, a window such as the one in Figure B.8 appears to show the problematic lines.

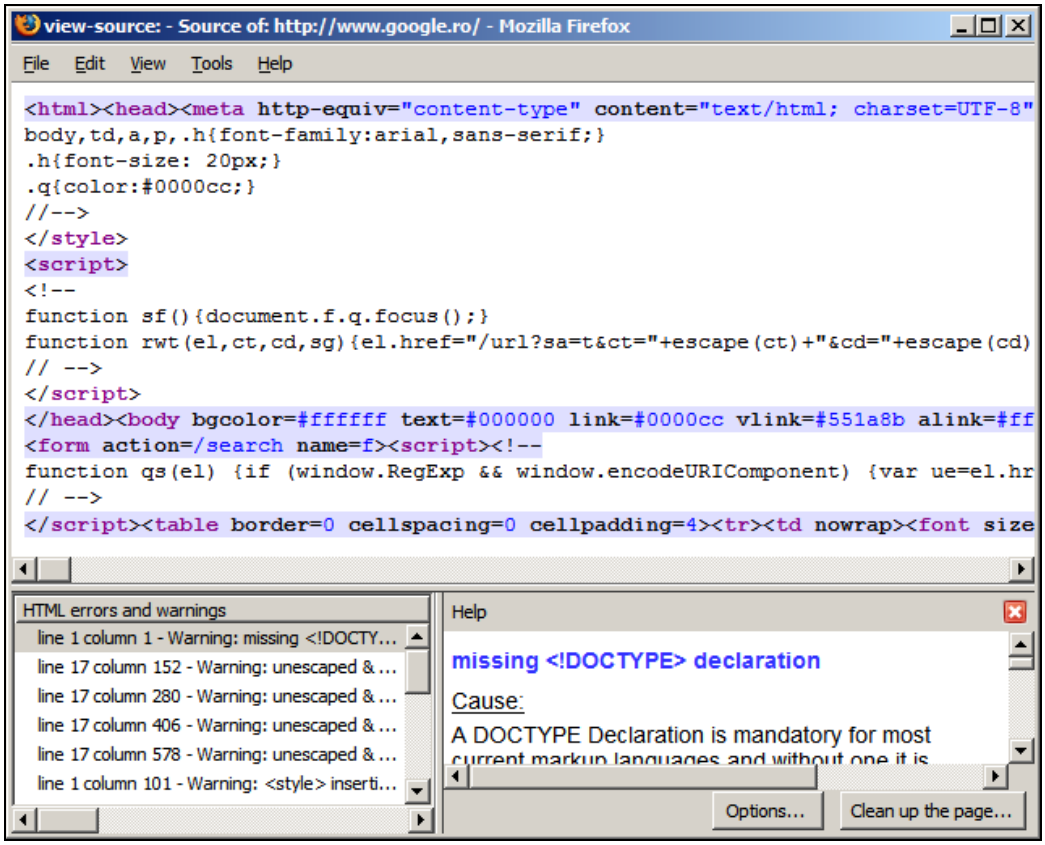
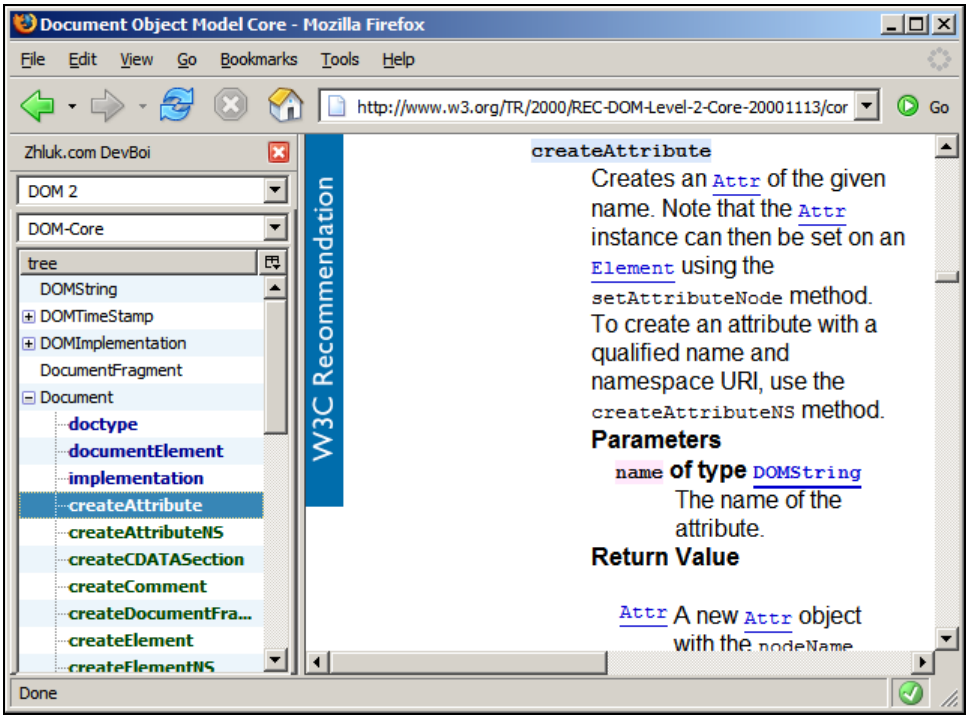


Figure B.8: Google Generates a Few Compliance Warnings

Checky is another validator for Mozilla browsers, which unlike the HTML Validator, uses online resources to perform validation instead of doing it locally. However, it supports validation of a much wider selection of formats, including HTML, XHTML, CSS, RDF, RSS, XML, and many more. You can download this plug-in at `http://checky.sourceforge.net`.

DevBoi (`http://devboi.mozdev.org`) adds a sidebar with documentation for HTML, CSS, DOM, and XUL (XML User Interface Language). You can chose to install the documentation locally, which occupies 4 MB, or let DevBoi fetch the data from online sources. After installing, the *Ctrl + F9* shortcut (Tools | Zhluk.com DevBoi in Firefox) will execute the program. The following figure shows the visual appearance of DevBoi:



B.9: DevBoi Loaded the W3C Page for createAttribute

UrlParams (`http://urlparams.blogwart.com`) is a small plug-in that shows the POST and GET parameters of a web page. Once installed, you can make it show from View | Sidebar | UrlParams (Alt+U). You can find this plug-in useful for web development, although it can't be of much help for AJAX development because the POST/GET parameters you're interested in are passed in the background.

Depending on your project at hand, you might want to have a look at other Firefox plug-ins, such as the **XSD Schema Validator** (`http://xsdvalidator.mozdev.org`), **XPather** (`http://xpath.alephzarro.com`), and **XPath Checker** (`http://slesinsky.org/brian/code/xpath_checker.html`).

Using the Tools

Here are the very first steps for writing and debugging code efficiently:

- When your application doesn't work as expected, the first step to make is to open the JavaScript console. Quite frequently, you will find this window flooded with errors from other pages, so you might want to clear it and re-run the problematic code.
- If you suspect the problem is at the server side, then check the Apache error log file. This file is by default installed as `Apache2/logs/error.log`.

- Passing data using GET makes debugging easier, because it allows you to simply test the server scripts by calling them directly from the browser. This technique is demonstrated in some of the case studies.
- Showing debugging messages using alert in JavaScript, or echo and exit in PHP, will always be handy, no matter how many advanced debugging tools you have around.

These are just a few basic suggestions, and I'm sure you'll find many more for your own projects, depending on the project complexity and your skill level.