Debug iPhone Web applications with Eclipse

Diagnose and repair iPhone Web apps using Eclipse, Aptana's iPhone Development Plug-in, Aptana's Firefox JavaScript debugger, and Firebug

Skill Level: Intermediate

John Fronckowiak
President
IDC Consulting Inc.

15 Jul 2008

Learn how to debug Asynchronous JavaScript + XML (Ajax) Web applications using Eclipse, Aptana's iPhone Development plug-in, Aptana's Firefox JavaScript debugger, and Firebug.

Section 1. Before you start

About this tutorial

This tutorial shows how to debug iPhone Web applications using Eclipse, Aptana's iPhone Development plug-in, Aptana's Firefox JavaScript debugger, and Firebug.

Objectives

Use the tools and utilities provided by the Eclipse-based Aptana Studio to debug iPhone-optimized Web applications. Much of this tutorial is focused on how to optimally debug JavaScript and Ajax applications.
Prerequisites

You should have a good understanding of iPhone Web application development, JavaScript, and Ajax.

System requirements

To follow the examples, you need Aptana Studio and one of the operating systems Eclipse supports: Mac OS X, Microsoft® Windows®, or Linux®. You also need a Java™ Runtime Environment (JRE); JRE for Java V5 is recommended.

Section 2. Aptana Studio

The Aptana Studio integrated development environment (IDE) is an Eclipse-based open source development environment targeted to building Web applications, with a focus on HTML, Cascading Style Sheets (CSS), and JavaScript. Aptana Studio includes an iPhone Development plug-in that provides Web developers with a set of tools that assist with the development of iPhone-optimized Web applications.

This tutorial shows how to get started with debugging using Aptana Studio. It discusses debugging and logging with the debugger capabilities in Aptana Studio. Aptana Studio also provides integration with the Firefox Firebug extension for integrated inspecting and debugging.

The example code used in this article is a demo application included as part of Aptana Studio IDE, so you do, and it shows how the Aptana Studio IDE facilitates the debugging process.

Testing and debugging Web applications can be a difficult task that can be complicated to perform natively on the iPhone. Aptana Studio provides a localized development Web server you can use for testing. This tutorial shows how to configure Aptana Studio and your wireless network to locally access and test iPhone Web applications on a localized development Web server.

Obtaining and installing Aptana

Aptana Studio is available as a free Community Edition or as a paid Professional Edition, which includes additional support. The Aptana site describes the

Aptana Studio provides the following features for iPhone Web development:

- Previewing iPhone projects in horizontal and vertical iPhone visualizations
- Starter files to help you quickly begin your projects
- The ability to easily import Ajax libraries into your iPhone projects
- A local Web server that enables you to run Web applications on the iPhone over your local WiFi network
- Firebug integration for logging and command execution

Aptana Studio system requirements:

- Aptana Studio is supported under Windows, Mac OS X, and Linux operating systems.
- Under Windows, 512 MB of RAM and an Intel® Pentium® 4 processor or higher is required.
- Under Mac OS X, 512 MB of RAM and a G5 or Intel processor is required.
- Under Linux, 512 MB of RAM and a Pentium 4 processor or higher is required.
- V1.5.0 or later of the Java JRE must be installed.

See Resources for information about downloading Aptana.

After you've downloaded and installed Aptana Studio, you're ready to install the iPhone plug-in, which is not installed by default. Launch Aptana Studio. As shown in Figure 1, the Aptana Start Page is displayed in the primary file pane. Scroll down in the Plug-ins section to view the Aptana iPhone Development Plug-in.

**Figure 1. Aptana Studio Start Page**
Click the iPhone plug-in link to download and install the plug-in. The Updates dialog box is displayed. Select the **Apple iPhone Support** item, then click **Next**.

**Figure 2. The Updates dialog box**
Accept the license agreement, then click **Next**.

**Figure 3. iPhone plug-in license agreement**
The final dialog box in the plug-in installation process is displayed. Click **Finish** to download the iPhone plug-in.

**Figure 4. Final dialog box in plug-in installation process**
The Feature Verification dialog box is displayed. Click **Install All** to complete the installation process.

**Figure 5. Feature Verification dialog box**
Restart Aptana Studio to complete the iPhone plug-in installation process.

iPhone Web application development overview

Prior to Apple’s release of the iPhone software development kit (SDK), the only way to develop applications for the iPhone was through the Web. While the iPhone SDK offers the promise of native applications that run using local resources, Web applications can provide the benefits of server-side resources.

The iPhone is an optimal mobile platform for Web 2.0 applications. iPhone provides the most desktop-like browsing experience available on a mobile platform. JavaScript and Ajax provide the essentials for Web 2.0 application interactivity. iPhone Web application developers should understand the following about the Safari Web experience on the iPhone:

• Tabbed browsing experience
• Support for Extensible Hypertext Markup Language (XHTML)
• Support for CSS V2.1 and parts of CSS V3
• JavaScript V1.4 support
• Document Object Model (DOM) support
• Canvas tag support
• Support for the XMLHR (XMLHttpRequest) application program interface (API)
• PDF and QuickTime support

The iPhone version of Safari also presents several limitations to be aware of, including:

• A 5-second execution time limit
• A 10-MB memory allocation limit
• A limit of eight documents open simultaneously

Section 3. Aptana JavaScript debugger

The Aptana JavaScript debugger provides a set of tools that permits complete examination of your JavaScript while it is executing. Before you can use the debugger, you must install it. Currently, the Aptana JavaScript debugger is only supported for Firefox. While iPhone Web applications are optimized for the iPhone Safari Web browser, they are still simply Web applications that can be accessed in any browser. The debugging tools provided by Aptana can still be of assistance from Firefox.

To install the debugger, create a new Debug configuration for Firefox. Click on the Debug toolbar icon, then click on the Debug menu item.

Figure 6. The Debug toolbar icon
The Debug window is displayed. Click on the **New Launch Configuration** icon.

**Figure 7. The Debug dialog box**

In the **Name** text box, enter a name for the new configuration. Under **Start Action**, choose the option for the debugger start page. You'll want to keep the default — **Use current page**. You can use the built-in Web server. Now click **Debug**, as shown
Figure 8. New debug launch configuration definition

If the Debugger Extension has not been installed, the confirmation dialog box shown in Figure 9 is displayed. Click **Yes** to install the Debugger Extension. When complete, the confirmation dialog box is displayed.

Figure 9. Confirm Aptana Debugger Extension installation
Using the Aptana Debug perspective

When you install the Aptana Debugger Extension, the Aptana Studio Debug perspective is launched. Aptana Studio asks for your confirmation to switch perspectives.

**Figure 10. Confirm Perspective Switch dialog box**

![Confirm Perspective Switch dialog box]

Shown in Figure 11, the Debug perspective contains a number of views, including:

- **Debug view** — Displays the debugging targets and currently executing processes
- **Variables view** — Scope of variables when suspended
- **Breakpoints view** — Lists the breakpoints set in the current file
- **Scripts view** — Scripts currently being debugged
- **Expressions view** — Value of current expressions
- **Ajax Requests view** — Displays current Ajax requests
- **Editor view** — File currently being debugged
- **Outline view** — Displays an outline of the code for the file currently being debugged
- **Console view** — Log output from the debugger.

**Figure 11. Aptana Studio Debug perspective**
Running the debugger

After the JavaScript debugger is installed, you're ready to begin debugging your JavaScript-based Web applications. Make sure you have a project loaded and ready to launch. For this example, I use the base iPhone project provided by Aptana Studio when creating a new iPhone Web application. As shown in Figure 12, select the Project view when in the default Aptana perspective. Right-click on the file you want to debug, then select **Debug As > JavaScript Web Application**.

**Figure 12. Select the file to debug**
As discussed, you can begin debugging your Web application by switching to the Debugging perspective. Click on the Debug toolbar icon.

**Figure 13. Debug toolbar icon**
Because you configured Aptana Studio to use its built-in local Web server, as shown in Figure 14, Firefox is opened to the local Web server at http://127.0.0.1:8000. If you know the local IP address of your computer — for example, my server is running at 192.168.1.106 on my local network — you can access your application directly from your iPhone's WiFi connection. Open your iPhone Safari browser and navigate to your local IP address to open your Aptana Web server. In my case, I navigate to http://192.168.1.106:8000. Click the Stop icon in the toolbar to end your debugging session.

**Figure 14. iPhone application running in Aptana Studio local Web server**
iPhone application developers can also view Web applications in a client-based iPhone Web-browser simulator. Shown in Figure 15, iPhoney gives a pixel-accurate Web-browsing environment to test Web applications. iPhoney is free and open source. I used it to take all the iPhone application screenshots in this tutorial. iPhoney is a Mac OS X-only application.

**Figure 15. iPhoney Web-browser simulator**
About iPhone

Here's iPhoney 1.2, the pixel-accurate iPhone web browser simulator. Build your web creations and see how they might look on iPhone.

— Thanks from the Marketcircle team.

A word about size.
In hand, everything will appear about half the size it does to you here, depending on your display resolution. That's because iPhone packs 160 pixels into each inch of its display, compared to somewhere around 72 to 96 for your desktop or laptop display.

To-do.
A few things aren't quite finished in iPhoney 1.2, but it's open source so you can help!

- Better looking/less intrusive scrollbars
- Drag scrolling
- Better zooming/scrolling of the web view
- Proper rotation animation for the web view
- Adding and saving bookmarks
Section 4. Breakpoints

One of the sample iPhone applications provided with Aptana Studio is the iPhone snowstorm simulator.

**Figure 16. iPhone snowstorm application**

The snowstorm effect is implemented in JavaScript. In the snowstorm project, the JavaScript logic is contained in the snowstorm.js file, which is shown in the default Aptana view in Figure 17.

**Figure 17. Snowstorm.js file in default Aptana view**
Breakpoints can be set in any JavaScript. When that line of code is reached, the debugger will pause execution, and you can inspect current variable values, the call stack, and more. To set a breakpoint, right-click in the gray area to the left of the line of code where you want the breakpoint, as shown in Figure 18. Select **Toggle Breakpoint** on the context menu to set or clear a breakpoint at that location.

**Figure 18. Setting a breakpoint**
To debug your application, click on the Debug toolbar icon, shown in Figure 13. This launches the debugger and changes Aptana Studio into the Debug perspective, as shown in Figure 19. Execution is paused at the breakpoint. The values of variables currently in scope are displayed in the Variables view.

Figure 19. Execution paused at a breakpoint
It's also important to note that breakpoints can be toggled in the same manner while in the Debugging perspective. You can set and remove breakpoints while your Web application is running in debugging mode.

Section 5. Stepping though code

When a breakpoint has been toggled and the debugging perspective launched, execution pauses as soon as a breakpoint is reached. From the paused execution point, you can do one of the following:

- **Step Into** — Continue execution one step, entering a function if necessary
- **Step Over** — Continue execution one step, skipping over functions
• **Step Return** — Continue execution, but break again at the return point of the current function

You can access the stepping functions through function keys:

• F5 — Step Into
• F6 — Step Over
• F7 — Step Return

Alternatively, you can use the toolbar icons shown in Figure 20. You can use the Stop icon to terminate execution. By default, Eclipse also provides Step Filters and Drop To Frame functionality, but this is not currently supported for JavaScript debugging in Aptana Studio.

**Figure 20. Toolbar debug step icons**

---

**Section 6. Console logging and the console object**

To use the debugger, your Web application must run in the Firefox debugger. While this can be useful for interactively executing and examining your code, there are times when you simply need to run your application on your physical iPhone. I previously discussed how the local Web server in Aptana Studio allows you to access your Web application from your iPhone’s WiFi connection. While running your Web application on the iPhone device itself, you can still access debugging information in Aptana Studio through console logging.

To view the console, select **Show View > Console** from the Window menu. The Console view is displayed.

**Figure 21. The Console view**
Your Web application can now send messages to the console using the `console.log` function. As shown in Figure 22, I have added a call to the `console.log` function right before my breakpoint.

**Figure 22. console.log call and output**
The console object allows JavaScript logic to display various information in the console.

```javascript
console.log(object[, object, ...])
```

Writes a message to the console. Pass as many arguments as you’d like, and they will be joined together in a space-delimited line.

```javascript
console.debug(object[, object, ...])
```

Writes a message to the console, including a link to the line where it was called.

```javascript
console.info(object[, object, ...])
```

Writes a message to the console with the visual "info" icon and color coding and a link to the line where it was called.

```javascript
console.warn(object[, object, ...])
```

Writes a message to the console with the visual "warning" icon and color coding and a link to the line where it was called.

```javascript
console.error(object[, object, ...])
```
Wants a message to the console with the visual "error" icon and color coding and a link to the line where it was called.

```javascript
console.assert(expression[, object, ...])
```
Tests to see if an expression is true. If not, it writes a message to the console and throws an exception.

```javascript
console.dir(object)
```
Prints an interactive listing of all properties of the object.

```javascript
console.dirxml(node)
```
Prints the XML source tree of an HTML or XML element.

```javascript
console.trace()
```
Prints an interactive stack trace of JavaScript execution at the point where it is called.

```javascript
console.group(object[, object, ...])
```
Writes a message to the console and opens a nested block to indent all future messages sent to the console. Call `console.groupEnd()` to close the block.

```javascript
console.groupEnd()
```
Closes the most recently opened block created by a call to `console.group`.

```javascript
console.time(name)
```
Creates a new timer under the given name.

```javascript
console.timeEnd(name)
```
Stops a timer created by a call to `console.time(name)` and writes the time elapsed.

```javascript
console.profile([title])
```
Turns on the JavaScript profiler.

```javascript
console.profileEnd()
```
Turns off the JavaScript profiler and prints its report.

```javascript
console.count([title])
```
Writes the number of times that the line of code where count was called was executed.

---

Section 7. Firebug
Firebug is a Firefox extension that allows you to debug and inspect your pages in the Firefox browser. Firebug is integrated with the Aptana Studio debugger. In fact, when the debugger was configured, Aptana installed a version of the Firebug plug-in that is integrated with the Aptana debugger. Firebug provides many of the same features as the Aptana debugger, including:

- HTML element inspection
- CSS inspection
- Ajax request monitoring
- Interactive console
- Console logging
- Breakpoints and interactive debugging (breakpoints set in Firebug are transferred to the Aptana debugger, and vice-versa)
- Web-page profiling

To access Firebug, launch your application in the Debug perspective. This opens your application in Firefox. Select **Firebug > Open Firebug** on the Tools menu to launch Firebug in your Firefox Web browser.

**Figure 23. Firebug view in Firefox**
Section 8. Inspecting HTML and CSS

To inspect the HTML on a page in Firebug, click the HTML tab.

**Figure 24. HTML inspector**
Clicking **Inspect** enables inspection on hover, as shown in Figure 25. Simply pointing to a page element displays the underlying HTML in the HTML inspector.

**Figure 25. HTML hover-mode inspection enabled**
Clicking the CSS tab in Firebug displays the CSS inspector, as shown in Figure 26. If your Web page contains multiple stylesheets, you can view them from the drop-down box.

**Figure 26. CSS inspection**
Section 9. Debugging Ajax requests with Firebug

Firebug also allows you to monitor network activity, including Ajax requests, through the XMLHttpRequest object. Click the Net tab, shown in Figure 27, to view the network activity.
Using the options, you can view the following:

- **All** — View all network activity
- **HTML** — View HTML files loaded into the page
- **CSS** — View all CSS files loaded into the page
- **JS** — View all JavaScript files loaded into the page
- **XHR** — View all XMLHttpRequest objects for the page
Section 10. Profiling code performance with Firebug

Code profiling is the process of benchmarking the execution of one or more code units to understand where the time is being spent in terms of code execution. The proper use of profiling helps to answer these questions and more:

- What lines of code are responsible for the bulk of execution time?
- How many times is this looping construct executed?
- Which approach to coding a block of logic is more efficient?

Profiling on iPhone JavaScript applications is critically important because the iPhone version of Safari imposes a 5-second JavaScript execution limit. After the limit is met, all JavaScript execution is terminated.

You can determine how long a function takes to execute with the `console.time()` and `console.timeEnd()` functions. You can also interactively profile your entire script in Firebug. Click the **Profile** button to start the profiling process, then click again to stop it. The profiling results are displayed in the Firebug console.

**Figure 28. Entire script profiling**
Finally, you can simply profile a function using the `console.profile()` and `console.profileEnd()` functions. The result of the profiling is displayed in the console.

**Figure 29. Profiling using the console.profile() and console.profileEnd() functions**
Section 11. Conclusion

This tutorial illustrated how to get started using the tools and utilities provided by Aptana Studio to debug iPhone-optimized Web applications, including using the console for logging, the console object, and Firebug. You should have a better understanding of configuring Aptana Studio for iPhone debugging, how to use the
debugger in Aptana Studio and Firebug, and how to profile the execution of your Ajax application.
Resources

Learn

- The iPhone Developer Program is a helpful resource at the Apple Developer Connection for learning about iPhone Web development.

- The iPhone Human Interface Guidelines at the iPhone Dev Center is a set of metrics and guidelines.

- New to Eclipse? Read the developerWorks article "Get started with the Eclipse Platform" to learn its origin and architecture, and how to extend Eclipse with plug-ins.

- Check out the "Recommended Eclipse reading list."

- Browse all the Eclipse content on developerWorks.

- New to Eclipse? Read the developerWorks article "Get started with Eclipse Platform" to learn its origin and architecture, and how to extend Eclipse with plug-ins.

- Expand your Eclipse skills by checking out IBM developerWorks' Eclipse project resources.

- To listen to interesting interviews and discussions for software developers, check out developerWorks podcasts.

- Stay current with developerWorks' Technical events and webcasts.

- Watch and learn about IBM and open source technologies and product functions with the no-cost developerWorks On demand demos.

- Check out upcoming conferences, trade shows, webcasts, and other Events around the world that are of interest to IBM open source developers.

- Visit the developerWorks Open source zone for extensive how-to information, tools, and project updates to help you develop with open source technologies and use them with IBM's products.

Get products and technologies

- Download the version of Aptana Studio for your operating system from the Aptana Web site.

- Download the latest version of the Aptana Eclipse plug-in.

- Download the latest version of the iUi framework.

- Download the iPhoney Web browser simulator.

- Check out the latest Eclipse technology downloads at IBM alphaWorks.
• Download Eclipse Platform and other projects from the Eclipse Foundation.
• Download IBM product evaluation versions, and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
• Innovate your next open source development project with IBM trial software, available for download or on DVD.

Discuss
• The Eclipse Platform newsgroups should be your first stop to discuss questions regarding Eclipse. (Selecting this will launch your default Usenet news reader application and open eclipse.platform.)
• The Eclipse newsgroups has many resources for people interested in using and extending Eclipse.
• Participate in developerWorks blogs and get involved in the developerWorks community.

About the author

John Fronckowiak
John Fronckowiak is president and founder of IDC Consulting Inc. and also a Clinical Assistant Professor in Information Systems at Medaille College, at the School of Adult and Graduate Education. He is the author of several books and articles about Web application development, programming, database design and development, and networking. You can reach John at john@idcc.net.