CHAPTER 4

JavaScript in Action

Now that we’ve covered the basics of JavaScript and the DOM, let’s work with our new tools. In this chapter we have picked seven projects that will help you build your own projects, as well as illustrate many of the techniques and JavaScript features we have covered in other chapters:

- Working with JavaScript
- Loading Scripts Efficiently
- Asynchronous Communication using XMLHttpRequest
- Cross-Domain Techniques
- Data Caching
- Choosing a JavaScript Library
- Using jQuery
- Building Your Own JavaScript Library

Working with JavaScript

Although not technically a “project,” we wanted to discuss some important aspects of working with JavaScript. Probably the most common questions we get from JavaScript novices have to do with working with JavaScript: which editors are good? How do you debug? What’s the best environment to work with? Are there any tricks to working with the language? We wanted to take this opportunity to answer these questions.

Over the years, we’ve written JavaScript in just about every environment imaginable. One of the great things about JavaScript is that you don’t need a lot of tools to work with it. A simple text editor and a browser will suffice to get you started, and for basic projects that’s really all you need. Once you start working on projects with a little complexity, though, you’ll quickly find yourself wanting more advanced tools.

In this section we want to cover the basic tools of the JavaScript trade. To start we wanted to go over the trinity of JavaScript development tools: integrated development environments, browsers, and personal web servers.

We’ll start by talking a little about some of the more popular integrated development environments (IDEs) with JavaScript support. Having a solid IDE with features like syntax highlighting, code completion, refactoring support, and collaboration capabilities can help tame a complex project. There are many available, and it’s hard to know which one to pick.

We also want to cover the developer support provided by web browsers. Modern web browsers provide a wide variety of very useful tools for monitoring and debugging the JavaScript that they’re running.

Finally we’ll cover the most commonly-used personal web servers. You can use your browser’s Open File feature to test your scripts, and that’s okay for basic work. Asynchronous communication, however, is one of the cornerstones
of building JavaScript applications, and it requires a web server. (We cover asynchronous communication in the Asynchronous Communication with XMLHttpRequest section, below.)

Once we’ve covered the tools of the trade we will talk about how to use them. We’ll provide some insights into our usual workflow when working with JavaScript, and then talk a little about methods for debugging your scripts.

**JavaScript IDEs**

Since JavaScript is essentially text, all you really need to write it is a text editor. Any text editor can serve the purpose, even something simple like the Notepad application on Windows. There are also several code editors available that work well with JavaScript, and some even provide basic features like syntax highlighting. We’ve used vi to create projects, and know several colleagues who are die-hard emacs users. We’re also very fond of TextMate and Sublime Edit, two great editing programs that support a variety of languages.

When you start working with complex projects with many JavaScript files, you’ll quickly find that you’ll need more features than simple code editors can provide. That’s where integrated development environments (IDEs) come into play. (And if you’re already familiar with code editing environments for other languages, you’ll want the same features for your JavaScript projects.)

An integrated development environment takes a code editor to the next level. A typical IDE will provide features for managing multiple files and file types, grouping them together in projects or applications (the exact term varies), and will often provide features for collaborating with other developers, such as integration with source control systems.

There are several IDEs that support JavaScript development. Typically they all provide a basic set of editing features (e.g. file creation, deletion, renaming, moving; find/replace in a single file or across multiple files; auto indenting). The better IDEs will provide more advanced features like code assist (a feature that acts as a dynamic assistant, providing suggestions or autocompletion based on what you have typed).

In addition, much of the time you won’t be working exclusively with JavaScript. If you’re working on a typical web project you’ll also be working with HTML and CSS, so you’ll want the IDE to support those as well. We find, however, that we can live with fewer features for HTML and CSS in an IDE if the JavaScript support is particularly good.

If you’re new to using IDEs entirely, we recommend looking through this list and trying a couple of the options. Many developers are passionate about their IDEs and will claim that their choice is the only logical one, but really the choice is one of personal preference. Trying a few IDEs will help you figure out what your preferences are—which features you really like, which ones you can work without—and you can make a choice at that point.

**aptana studio**

http://www.aptana.com

Aptana Studio is our preferred IDE. It’s built on Eclipse (see below) and is available as either a standalone install or as an Eclipse plugin. Aptana has all of the features of the base Eclipse IDE (build integration, cross-platform, scriptability, etc). In addition, Aptana has several very useful features for web development in general, and JavaScript development in particular:

- Code assist with JavaScript, HTML, and CSS.
- Code assist with jQuery (which has its own syntax, see Using jQuery, below).
- Out-of-the-box Git integration (Git is a highly popular source control system; see [http://git-scm.com/](http://git-scm.com/) for details).
- Built-in CLI, for working with script interpreters like node (or ruby if you’re working with Rails).
- JSLint integration (JSLint is a JavaScript syntax and style checker; see [http://www.jslint.com/lint.html](http://www.jslint.com/lint.html) for details).
- Open Source, with an active community. Bug reports are answered quickly and fixes are pushed regularly via the internal updating system.