Adding in Ajax

I imagine that, in the days before most programming languages included garbage collection, way back in the 1980s when people were working with C and Pascal, programming was a much more existential profession. People programmed computer applications that were never intended to be made public, working on programs that were only ever intended to be run on a small number of computers not connected to a big open network like the Internet. Much more time was spent on code to perform data processing and making a program run than went into the user interface.

Obviously, a lot of things have changed since then. A lot of software development has moved out of the basement and onto the web. Because of this, a lot of effort now goes into a user interface that is publicly available to anyone with a computer and a network connection. This means that development is a lot more engaging to developers these days because we’re able to get visual feedback from our coding efforts a lot quicker than you could when people were coding up computer programs in C.

A large portion of increased interactivity via the web can be attributed to the rise in popularity of a technology called Ajax. Developers use Ajax to make their Web pages more dynamic and more responsive to a user’s actions on the site. Don’t worry, this has nothing to do with those Flash intros that cropped up all over pages in the late 1990s. It’s far better than that. With Ajax, we can add dynamic functionality to the site, capturing input from the user and then updating the current page without having to reload the entire page.

And the great thing is, the work we do can be viewed by anyone with a computer and an Internet connection. It’s a great time to be a web developer.

In this chapter, we’re going to add two little functions to our site using Ajax. We’re going to briefly discuss Ajax, what it can do for us, and how it works. We’ll look at how we can use the jQuery JavaScript library in order to handle most of the Ajax details for us. Then, using these new concepts, we’re going to set up a means for registered customers to write product reviews on the site and set up tagging functionality so that users can tag products with keywords.

The Ajax Pros and Cons

“Ajax” stands for Asynchronous JavaScript and XML, after two of the technologies that were primarily used in the initial implementations. However, the term “Ajax” itself has come to embody more than simply JavaScript working in conjunction with XML. Ajax represents an entirely different means of requesting data from a server than a typical web page. Whereas typical requests are made for entire pages of HTML, an Ajax request typically returns a smaller amount of data, which is then injected into the current page using the Document Object Model (DOM).

One of the main advantages of this is the ability to process data and requests with a great deal less bandwidth. When you make a request to any given web site for a certain page, the HTML is downloaded and rendered by the browser. Then, the browser makes a bunch of subsequent requests to the server for...
all of the extra components on that page, such as style sheets, JavaScript files, and any images. All put together, each of these subsequent HTTP requests for pictures and other static files can comprise a great deal of the overall load time for a page, although it’s quite easy to forget this point in the era of high-speed Internet connections, which make it seem as though everything on an HTML page is downloaded in one fell swoop.

Ajax requests, on the other hand, are triggered by events within pages, and are able to send and retrieve data to and from the server without the need to incur all of this extra page overhead of downloading components. Remember that HTTP is stateless, so each HTTP request is independent. Imagine that you’re a door-to-door salesperson selling some new kind of vacuum cleaner that you hitch to the back of your housecat, and let the cat clean the house for you. You knock on someone’s door and, if the person living there doesn’t want to buy the cat vacuum cleaner (hey, it’s their loss), they shut the door on you, mid-sentence. You’re cut off. The only way to get back in is to knock again.

Let’s transfer this analogy to the context of the web: you would just need to refresh the page, and the new, updated page will be sent back to you. However, if all you want to do is update a small portion of the page, having to reload the entire page and everything on it can be overkill.

As one example, let’s say you want to allow a user to reload the list of messages posted on a message board page by having the user click a “Reload” button somewhere on the page. Using Ajax, the new content could be loaded and the page could be updated, all without having to reload the entire page. This is all done using JavaScript, which can make requests to the server, parse out data it retrieves, and then update the part of the page with the new content. And you save the overhead of a page refresh.

The main problem with Ajax is that it’s still dependent on the user of the site having JavaScript enabled in their browser. There’s simply no guarantee that this will always be the case. Some of the people using your site may have JavaScript turned off. When you’re programming an e-commerce site, there’s generally one of two things that should keep you from going crazy and adding Ajax to everything:

- Content that is accessible only via JavaScript, like in DHTML navigation menus, cannot be crawled by search engines. If you use a good deal of content that is only available via Ajax events, then this content is invisible to spiders.

- Mission-critical parts of your site, like the checkout process, cannot rely on any JavaScript at all. The last thing you want to do is stop someone from buying stuff from your site.

I really can’t give you firm advice about when you should be using Ajax functionality on your site and when you shouldn’t. I will, however, offer one guideline that not only makes good business sense, but will also help save you countless hours of development time:

- Code for your customers and not for your fellow developers.

Stuff that is all Ajaxy and dynamic is admittedly very cool. I remember using Google Maps for the first time, and it was superb. However, you shouldn’t preoccupy yourself with tasks that are mainly to add zing to your site without adding any value for your customers. Ajax, for all its wonders, does bring up a whole different slew of security issues that you need to consider, such as Cross-Site Scripting attacks (I’ll cover those in Chapter 12). It can also make the content of your site less accessible to people who have JavaScript disabled in their browsers.

As I’ve been saying all along, you should be coding an application that best serves your customers, and that might include some Ajax functionality. By default, though, it’s not necessarily a great idea. Imagine that you get the urge to code up a series of checkout pages that slide by the user as they go from page to page; yeah, that’s cool, but are your customers even going to notice it? Do they care? Is the extra functionality interfering with anyone trying to check out? It all depends on your audience and their expectations. So far, in this book, I haven’t used Ajax for other stuff, such as the checkout process or the shopping cart, merely because it’s in your best interest not to use Ajax for anything mission-critical, and because Ajax and accessibility are, at the time I’m writing this, something of an oil and water combo. They don’t mix well.