jQuery and Ajax in the Presentation Tier

“The measuring programming progress by lines of code is like measuring aircraft building progress by weight.”

—Bill Gates

The landscape of the Internet has changed dramatically over the past few years. Users expect more functionality out of web applications, to the point that the line between web and desktop is somewhat blurred. The issue is complicated by the nature of web browsers themselves; different manufacturers have implemented different parts of individual standards, added their own custom features that other browsers don’t support, and so on. A web developer is left to try to create identical functionality and appearance across any number of distinct browsers that may or may not support any number of features. It appears quite daunting indeed. Luckily, there exists a quality JavaScript framework called jQuery that aims to bridge developer and user desire with browser results. We’ll begin with a discussion of the problem domain, explore jQuery, and look at some clever ways to add Ajax functionality to the CMS. We’ll improve on the style and design of the public-facing site as well.

Note jQuery has a vibrant community built up around it, constantly evolving the framework with plug-ins, innovative applications, and so on. This chapter is not meant to be an exhaustive look at everything the jQuery library can do (which anyone will tell is “quite a bit”) but rather focuses on getting you up to speed on some core practices and applications of the library in the context of the CMS. For a very in-depth look at the nuances of jQuery, I recommend Embracing jQuery: User Experience Design by Brennan Stehling and Cindi Thomas (Apress, 2010).

An Introduction to jQuery

jQuery is an extremely popular JavaScript framework created by John Resig. The goal of jQuery is to provide a way for developers to leverage JavaScript across a variety of browsers without having to concern themselves with the low-level plumbing involved in writing JavaScript that operates
identically across those browsers. The real power of the framework is in its syntax and brevity coupled with its highly extensible nature.

The $() Factory

Arguably the most common action in jQuery is the selection of particular elements on the page that need to be modified in some fashion; this is accomplished via the $() factory. The $() factory is the window to jQuery’s selection capabilities, capable of accepting CSS classes, element IDs, types of elements, XPath queries, and more, making it an extremely flexible and versatile method of affecting changes upon a page’s markup. This is a key tenet of jQuery: simplicity makes code easier to understand and maintain.

For example, Listing 5–1 shows a snippet of JavaScript that uses jQuery to add some class information to elements on a page. In almost all cases, jQuery code that involves event handling or working with page elements is placed within the $(document).ready() function. This ensures that the entire document has been loaded but fires before the page contents themselves are displayed. If the document were not ready, actions could be triggered for elements that didn’t presently exist, which would could cause unpredictable behavior (or simply stop script execution altogether).

Listing 5–1. jQuery Code to Modify a Few Stylistic Elements and Display a Traditional Alert Box

```javascript
<script type="text/javascript">
  $(document).ready(function () {
    $('h1').addClass('underlined');
    $('h4').addClass('italicized');
    alert('Hello via jQuery!');
  });
</script>
```

In CSS, classes are prefixed with a period, while IDs are prefixed with a pound sign. For instance, `<div class="foo"></div>` would be accessed with `.foo { attribute: value; }`, while `<div id="bar"></div>` would be accessed with `#foo { attribute: value; }`. What’s the difference? IDs should be unique on a page, while classes can appear multiple times; we should have only one `<div>` with an ID of “bar,” but we could have many `<div>`s with classes of “foo.”

The jQuery Object

The $() factory is just a shorthand wrapper for the jQuery object. The previous code could be written like this:

```javascript
<script type="text/javascript">
  jQuery(document).ready(function () {
    jQuery('h1').addClass('underlined');
    jQuery('h4').addClass('italicized');
    alert('Hello via jQuery!');
  });
</script>
```