PART I

INTRODUCING JQUERY AND JAVASCRIPT

CHAPTER 1

INTRODUCING JQUERY

JQUERY IS A JAVASCRIPT LIBRARY created to help Web designers and developers write and extend JavaScript interactions quickly and concisely using a defined set of methods wrapped around the native JavaScript functions. jQuery does not offer any new functionality, but it takes existing hard-to-understand-and-write JavaScript APIs (application programming interfaces) and makes them available to a wider audience through easy-to-understand-and-write jQuery syntax.

In this chapter, I guide you through the benefits of using a JavaScript library, show you the different libraries that are commonly grouped into the same category as jQuery, and give you a good background on the features of jQuery and why it's a great library.

DISCOVERING JAVASCRIPT LIBRARIES

JavaScript libraries allow Web designers and developers to extend Web page interactivity and usability by employing a framework of commonly used JavaScript functions built using native JavaScript primitives.

Think of libraries as frameworks or blueprints with a set of rules and guidelines to help you build your Web site. JavaScript libraries make writing JavaScript much easier for Web designers and developers — they are a starting point. Many popular libraries such as Prototype, MooTools, Dojo, YUI, and the main focus of this book, jQuery, are used widely on the Web today. Each library has a specific feature set, with jQuery owning the DOM (Document Object Model) manipulation space.

The Document Object Model is the actual HTML code that represents a Web page, structured like a tree, with each branch being a node tied together in a hierarchical sense. Each node can be accessed most commonly through CSS and also through JavaScript using selectors. The DOM is the API (application programming interface) for how Web designers and developers can manipulate the Web page using methods created by the HTML standards committee. HTML 5 offers a new set of APIs for interacting with the DOM and creating a richer Internet experience for users. After a Web page is fully loaded, the DOM is ready to be interacted with.

A JavaScript framework allows a Web designer or developer to extend the DOM by adding a JavaScript include (`library.js`) to a page and then using special functions set up within the library.

REALIZING THE BENEFITS OF USING A JAVASCRIPT LIBRARY VERSUS THE TRADITIONAL APPROACH

The greatest benefit of using a JavaScript library is being able to tap into a huge assortment of functions to extend your Web pages beyond dull, non-interactive content.

JavaScript libraries can offer ways for Web designers and developers to work with effects, animations, events, Ajax, and interactive user interface widgets for faster and rapid Web development. Designers and developers are not limited to those functions provided by the library. You can also write your own.

The beauty of JavaScript libraries for Web designers who understand the DOM is that manipulating the DOM with a library becomes inherently easier than manipulating it by using the limited API of JavaScript.

In order to get the same features by writing your own JavaScript, you would have to spend countless hours and long nights programming, testing, and bug fixing, which would probably result in massive amounts of code. JavaScript libraries help greatly in this area by reducing the amount of code it takes you to do something that might normally be four times as big if it was done with native JavaScript.

Avoiding repetition is another benefit of using JavaScript libraries. As you start writing JavaScript functions to do similar tasks, you end up with a lot of very similar code — by using a library, you can eliminate that repetition.

GETTING TO KNOW THE MAIN LIBRARY PLAYERS

When you select a framework, you have about 20 JavaScript libraries to choose from, with five of those libraries being the main players. These five main players — YUI, Prototype, MooTools, Dojo, and the topic of this book, jQuery — have risen up above the rest because of their ease of use and the enormous audiences they have. The main differentiators between most of these libraries include size and browser support.

The five libraries I discuss are open source, which means that anyone can contribute to the source code that makes up these libraries. Microsoft software, for example, is not open source — it is propriety software owned by Microsoft. Microsoft employs its own programmers to develop it. Microsoft then sells their software for a licensing fee. The licensing fee allows you to use the software, usually for a set period of time, and gives you access to support from Microsoft if you have problems.

Open source software is different. Anyone can download the source code and contribute changes to it — which makes for better code because it's all created on a volunteer basis with a goal of writing better software, not making money. Because you are not paying a licensing fee, you are free to do whatever you please with the library. The open source community on the Web is huge, with millions of users contributing through blogs and forums, and it's now quite easy for designers and developers to find support when they need it.

One important thing to keep in mind is that when you are learning a JavaScript library, you are learning to read and write in what feels like another language — it is another interpretation of the JavaScript language.

YUI

The YUI (Yahoo! User Interface) JavaScript library was created by the Yahoo! Developer Network in 2005 and is under the BSD (Berkeley Software Distribution) license. The BSD license allows software to be distributed as permissive-free software, which has the least amount of restrictions for distribution purposes compared to other similar licensing options such as GNU General Public Licenses. YUI is fully supported on Internet Explorer 6 and later versions, Firefox 3 and later, Safari 3 and later, and Opera 10 and later.

The total file size of the YUI library is 31 KB.

To give you an idea of what YUI code looks like, here is some sample JavaScript code showing how to implement a click event using YUI. There are two parts to the `click` event here: the function that is called when the click occurs and the actual `click` event itself. The code is not that elegant and uses a lot of YUI-specific syntax.

```
function handleClick(e) {
  Y.log(e);
}
YUI().use('node-base', function(Y) {
  Y.on("click", handleClick, "#foo");
});
```

Prototype

Prototype, a JavaScript library created by Sam Stevenson, became popular because it was the first JS framework to be bundled with the also-popular rapid development-programming framework called Ruby on Rails. Because it's incorporated in Ruby on Rails, I have always felt that it wasn't meant for Web designers, but more for hardcore Web developers to use in conjunction with Ruby on Rails.

The Prototype library is a base library with Ajax functionality, which becomes more feature-rich with the addition of Scriptaculous. Scriptaculous offers effects and user interface elements and only works in conjunction with Prototype. The major downside to this library is the size: Both .js files total up to 278 KB.

The documentation for the Prototype and Scriptaculous libraries can be hard to understand for inexperienced front-end developers. As with other libraries, a community of supporters exists, but Prototype can still be hard to learn because of some of its more complicated syntax. To give you an idea of what Prototype code looks like, here is some sample JavaScript code showing how Prototype handles a `click` event. The `click` event is very similar to setting up a `click` event in jQuery, but looks can be deceiving — many of the other methods in Prototype are actually more difficult and look less like jQuery:


```
$("foo").observe("click", function() {
  alert('Clicked!');
});
```


MooTools

MooTools was first released in 2006, and is a similar JavaScript library to Prototype — the syntax is aimed at intermediate to advanced Web designers and developers. The MooTools JavaScript library allows designers and developers work with an object-oriented framework to extend the JavaScript API and provide interactivity on the Web pages. MooTools is for those looking for a library similar to pure JavaScript.

Here is some sample code showing how MooTools handles `click` events:

```
$('foo').addEvent('click', function() {}));
```

Dojo

Dojo was first released in 2004 as a JavaScript framework for creating cross-browser compatible Web applications and for adding seamless interactivity to Web sites. Dojo's

syntax can be quite confusing; it feels much more like writing native JavaScript and is aimed at experienced front-end developers, making it harder for beginners to use and understand.

Here is some sample code showing how Dojo handles `click` events:

```
fooNode = dojo.byId("foo");
fooConnections = [];
fooConnections.push(dojo.connect(fooNode, 'onclick', foo));
```

As you can see from all of the preceding examples, JavaScript libraries can have pretty confusing syntax. Now take a look at an example of how jQuery handles `click` events:

```
$('#foo').click(function() {
  //click event
});
```


THE JQUERY ADVANTAGE

jQuery comes with a lot of advantages. You can see by comparing it to the previous syntax examples that jQuery is the most concise and easy to understand. That's the advantage of jQuery — it just gets it done. No fluff, no confusing code, and you don't need to be a back-end programmer to write it, but that's not to say that jQuery doesn't have an advanced side.

Figure 1-1 shows the jQuery homepage at `http://jquery.com`.

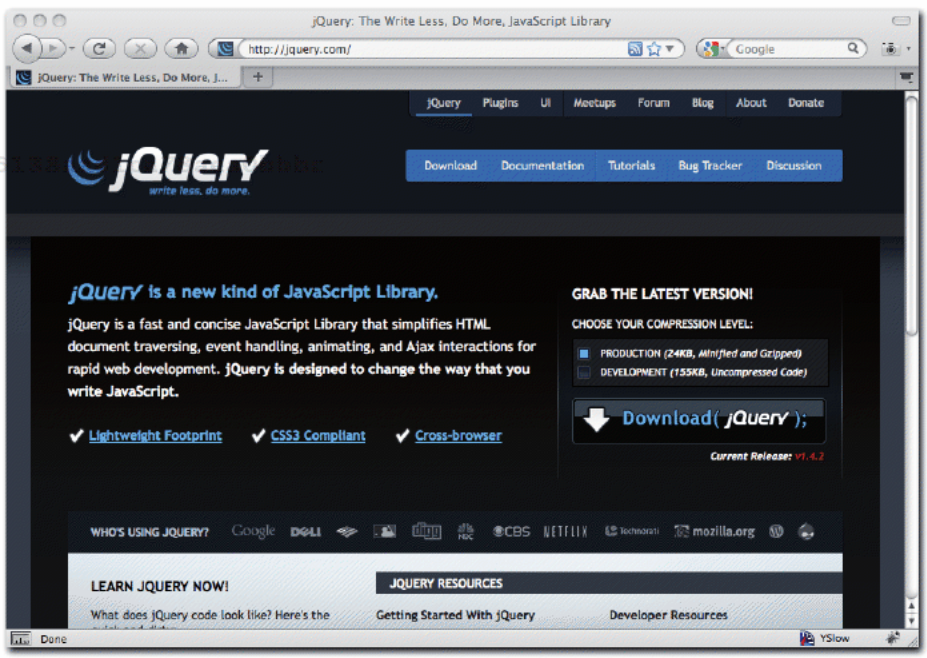

Figure 1-1: The jQuery Web site

A Brief History of jQuery

jQuery was created in 2006 by John Resig as an alternative to more complicated JavaScript libraries. jQuery enables Web designers and developers to write simpler JavaScript that still enables them to perform advanced JavaScript functions on their Web sites.

jQuery is awesome because no hardcore programming knowledge is required to perform DOM manipulation. jQuery does have some advanced areas that require prior JavaScript knowledge, such as working with the Ajax methods in forms to get and post content, as reviewed in Chapter 9, creating custom jQuery plug-ins, as reviewed in Chapter 11, and working with mobile Web sites, as reviewed in Chapter 10.

Most designers and developers whom I know use, or have used, jQuery at some point in the past four years. When I ask why, they usually answer, "Is there anything easier?" Its ease of use attracts so many people to use jQuery; you don't need a master's degree in computer science to make a form submit via Ajax.

You may be wondering what sort of things you can do with the jQuery library. The answer is, everything you can do with the native JavaScript API. I dive deeper into what you can do with jQuery throughout the book, but here is just a quick overview of the main features of jQuery:

- Events that include mouse, keyboard, form and user interactions
- Effects that include show/hide, sliding, fading, and custom animations
- Animation that allows you to make things move by utilizing CSS and native effects
- Ajax methods to interface with server-side form processing using XML and JSON
- Extensibility to create your own plug-ins that extend the jQuery API core
- DOM manipulation
- Cascading Style Sheet (CSS) manipulation
- Utilities that provide browser detection and easier interfaces for common JavaScript functions

Who uses jQuery

Web designers and developers are the main users of jQuery. I've seen jQuery used on a wide range of sites: everything from small-town mom and pop Web sites to full-blown enterprise sites. Because jQuery is free, all kinds of designers and developers use it. It brings the benefits of JavaScript to Web designers who may not know a thing about programming, but want to add cool effects to their Web site.

jQuery became more popular when Google and Microsoft started to offer hosted solutions. A hosted solution is when a file is hosted from a Web server — in this case, through a CDN (content delivery network) — to offer increased performance from those Web sites that use the file. This move by Google and Microsoft indicated that jQuery would be a preferred library and a major player within the open source JavaScript library community. Google,

BBC, Dell, Bank of America, Major League Baseball, NBC, and Netflix are part of the growing number of companies who are using jQuery on their Web sites. Netflix, a Web site that offers movie rentals to customers through direct mail and online channels, has been using advanced JavaScript to drive the user interface for a number of years. Figure 1-2 shows a menu creating using jQuery that appears as you hover over a movie title, allowing the user to view more information without leaving the page.

Figure 1-2: The Netflix Web site

Seeing How jQuery Works with Web Pages

jQuery is easy to set up. Just as with other JavaScript libraries, you include the JavaScript library at the top of your page between the `<head>` `</head>` tags. Here's how to include the jQuery JavaScript library in your site:

```
<!doctype html>
<html>
  <head>
    <script type="text/javascript" src="jquery-1.4.2.js"></script>
    <script type="text/javascript"></script>
  </head>
  <body>
    <a href="http://jquery.com/">jQuery</a>
  </body>
</html>
```

After you've added the jQuery library to your page, you write JavaScript code using the jQuery API (application programming interface) to access different parts of your page through the DOM (Document Object Model), which most Web designers and developers should be familiar with. If you are a Web designer, you work with it on a daily basis but may not even be aware of it.

You can also use jQuery to add and remove HTML from your page and respond to actions users perform on your page, such as clicking a link or filling in a form. You can also create animations and use Ajax to send and load content through Web services without having to refresh the page.

Exploring the Advantages of jQuery

Web sites have become much more than just text, images, and links to other pages. Internet users expect greatness from a Web site and the bar is constantly being raised by Web sites and companies such as Facebook, Google, Microsoft, Twitter, and Foursquare, to name a few. The technology is constantly changing, and using jQuery helps you keep up with the fast pace. jQuery is a library that promotes rapid development of Web sites or applications and allows you to focus on user interaction and interface design without having to write long, bloated code.

Writing jQuery is easier than writing JavaScript code because you are following an API. If you are proficient in writing HTML and CSS, you can understand and write jQuery because most jQuery functionality is based on interactivity within the HTML and CSS.

Open Source

JavaScript libraries are supported by the open source community and are well tested and updated. The open source community a huge support network. Web designers and developers are continually creating tutorials, books, and plug-ins to help and extend the jQuery library.

Great Documentation

By far, one of the greatest benefits of jQuery is its documentation, which is what makes a great library. The team behind jQuery has spent a great deal of their time documenting how the library works and how to navigate around their API. The jQuery documentation site has tutorials complete with code examples, plus a huge community of supporters across the Web. Figure 1-3 shows the Documentation section of the jQuery Web site.

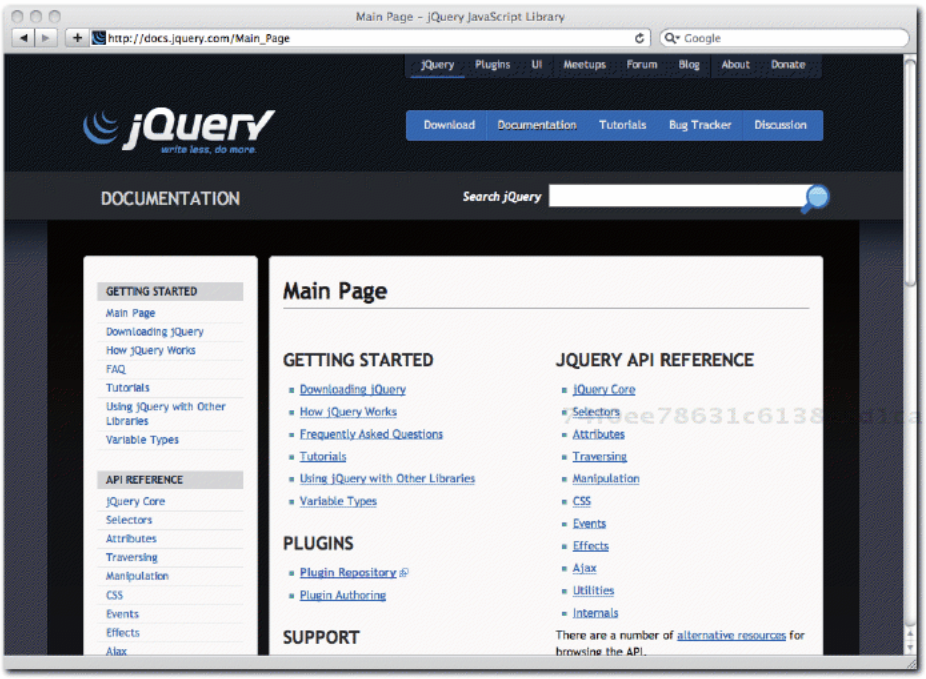

Figure 1-3: The Documentation section of the jQuery Web site

The community of developers and programmers that has created the jQuery library are constantly improving and releasing new versions. jQuery was first released 2006 as v1.0. Since then, the code has been updated 23 times, which brings us to the current release, v1.4.2. jQuery is continuously being improved, which is one of the reasons it has become so popular. Libraries that aren't updated as often are not as popular.

As updates occur, the documentation is updated for methods that have become deprecated (slated for removal in the next release) and to ensure that the library will be backward-compatible — that is, that it works with older versions. When a library is updated to a new version, the process of upgrading is painless — you just drop the new JavaScript library on your server. In addition, you should usually look over the changelog, a section that outlines each release and the changes that have been made to the library, to see if any methods you are using have become deprecated.

jQuery is released under the MIT License or the GNU General Public License (GPL), version 2. This basically means that it's free, and as long as you give credit to the author within the jQuery plug-in itself, you are free to use the code as you wish.

Same JavaScript with Less Code

jQuery is JavaScript: Everything you can do in JavaScript, you can also do in jQuery. The possibilities are endless. What I love about jQuery is that it gives you a base you can build on, but you're not limited by what jQuery offers. When you're using jQuery, you have three options when coding:

- Use the extensive jQuery API
- Use or create a jQuery plug-in
- Write regular JavaScript

Another attractive benefit of jQuery is the brevity of the code. If I want to change the background color in plain JavaScript, the code looks like this:


```
document.getElementById('mydiv').style.backgroundColor = 'red';
```

By using the powerful selection engine, jQuery needs only one shorter line to achieve the same outcome:

```
$('#mydiv').css('background-color','red');
```


The syntax is easier to understand than JavaScript and was created with Web designers in mind. When you compare this syntax to other libraries such as Prototype or YUI, you can see why jQuery has become the choice for many Web professionals. The selector engine is the most prominent and loved feature of the jQuery library. It allows you to use CSS2 selectors, which makes it incredibly easy for Web designers with CSS knowledge to pick up.

Chaining

One of jQuery's greatest features is chaining, which allows you to chain multiple methods one after the other. This helps to keep your amount of code smaller and therefore improves the speed with which your jQuery code is retrieved from the Web server and executed.

Here's an example of jQuery code that uses chaining:

```
$('#foo').addClass('active').prev().removeClass('active');
```

Here's an example that doesn't use chaining:

```
$('#foo').addClass('active');
$('#foo').next().removeClass('.active');
```

The example that uses chaining is a cleaner and more concise way to write jQuery. I use chaining in my code examples throughout the book.

Cross-Browser Compatibility

With the recent updates to Safari, Firefox, Internet Explorer, Google Chrome, and Opera, creating pages that work across all of the major browsers is the top priority. Browser wars have become a part of every Web designer's daily struggle.

When you use jQuery, you can rest assured that it's cross-browser compatible with all the popular browsers, such as Internet Explorer 6.0+, Mozilla Firefox 2+, Safari 3.0+, Opera 9.0+, and Google Chrome.

Often, a major issue with JavaScript is that you need to create different code to support multiple browsers. Some Web designers and developers choose to create alternate browser-specific style sheets to support CSS in different browsers, mainly Internet Explorer versus the rest. The same issue often occurs with JavaScript. The following code represents how to set up an Ajax request that works in multiple browsers:


```
If(window.XMLHttpRequest)
  {
  xhr = new XMLHttpRequest(); //Code for Firefox/Safari
  }
  else
    if(window.ActiveXObject) //Active X Version
    {
      xhr = new ActiveXObject("Microsft.XMLHTTP"); // For IE
}
```


Using plain JavaScript, you have to write two different functions, test those functions, and fix any bugs. It's quite a lot to manage, not to mention the repetition that occurs when you have to write several functions to do the same thing to support multiple browsers, instead of one script that supports them all, such as in jQuery.

By contrast, the following piece of code shows just how easy an Ajax request is in jQuery:

```
$.ajax();
```

CSS3-Compliant

All modern browsers support CSS1 and CSS2 (the first two versions of Cascading Style Sheets), and most Web designers and developers these days use CSS2. CSS3 is in development and offers enhanced features such as embedded fonts, rounded corners, advanced background images, and colors, text effects, and transitions. Only a handful of browsers currently support the full CSS3 spec as of July 2010 — Firefox 4, Internet Explorer 9, Opera 9, and Safari 4. Some older versions of these browsers do support certain features of CSS3.

jQuery has CSS3 support for new selectors only. What does that mean? One of the new features of CSS3 is additional attribute selectors, which are an improvement from the attribute selectors included in CSS2 and are similar to the attribute selectors in jQuery. These selectors allow you to style content based on its attributes, so you can filter based on specific values found in the attributes. Check out the following sample code:

```
p[title=*foo] {background:black;color:white}
<p class="text" title="food is good foo you">This is my sample text</p>
```

Practicing Unobtrusive JavaScript

Many of you have probably created pop-up windows by embedding JavaScript directly into your HREF tags, as shown in the following example. The biggest problem with this code is that it's embedded into your link HREF. If the user happens to have JavaScript disabled, which is rarely that case, this link won't work and there is no fallback method to enable them to view the help.

```
<a href="javascript:window.open('help.html', 'help window', 'height=800,width=600,to
 olbar=no');return false;">Help</a>
```


This HTML is an example of obtrusive JavaScript. For you Web designers, it's like when you write inline styles instead of separating the presentation layer from the content. To contrast with the example of obtrusive JavaScript, here is an example of how to use jQuery to provide an unobtrusive solution using similar code. When JavaScript is disabled, this version, instead of wrapping the code inside of a `click` function, offers users the fallback of clicking the link and being brought to the help page.

```
<!doctype html>
<html>
  <head>
    <title>Unobtrusive jQuery</title>
    <script type="text/javascript">
    $(document).ready(){
      $(".help-link").click(function() {
      var linkHref = $(this).attr('href');
      window.open(linkHref,'help window', 'height=800,width=600,toolbar=no');
      return false;
    });
  });
  </script>
  </head>
  <body>
    <a href="help.html" class="help-link">Help</a>
  </body>
</html>
```

Graceful degradation and progressive enhancement are strategies dealing with how to support newer browser features and older unsupported browsers while offering the best user

experience. Progressive enhancement is the newer strategy of the two, but the main difference is the approach each of them takes. I cover these in the next couple of sections.

Graceful Degradation

With the graceful degradation approach, you get your Web site working in all modern, popular browsers and then work with older browsers to make sure they support those features. Most Web designers and developers have practiced graceful degradation by setting up specific style sheets or browser hacks for various versions of Internet Explorer (ahem, not pointing any fingers, IE6) because of layout issues with the box model.

```
<a href="javascript:window.open('help.html', 'help window', 'height=800,width=600,to
 olbar=no');">Help</a>
<noscript>
Please upgrade your browser or turn on JavaScript, as your browser is not working
 with our Website.
</noscript>
```


Progressive Enhancement

Progressive enhancement refers to a strategy of starting with a baseline of features supported by all browsers and then adding more features for the modern browsers that support them. Progressive enhancement is a great practice to adopt because it makes your sites more accessible. It's better for your users if you deliver one set of features for everyone and add special upgrades for those using more compliant browsers — that is, those compatible with features such as CSS3 and HTML 5. Currently only Safari 4 and Opera 10.6 support HTML 5 and CSS3.

The progressive enhancement approach doesn't assume that everyone has JavaScript enabled and always gives the user an alternative way of accessing the content. Consider the pop-up window used in the discussion about practicing unobtrusive JavaScript. Instead, you can use the target attribute in the anchor tag to tell the browser to open up the link in a new browser window, instead of creating a pop-up. This is supported by all popular browsers.


```
<a href="help.html" target="_blank">Help</a>
```

In this book, I focus on using jQuery with the progressive enhancement strategy to give more modern browsers a slightly better experience, while still supporting the older browsers with a baseline experience.

Unobtrusive JavaScript and jQuery

jQuery makes practicing both of these strategies (graceful degradation and progressive enhancement) easier because all of the jQuery (JavaScript) code lives outside the HTML in an external JavaScript file or in the head of the HTML file you're working with, unless you are using a hosted solution delivered by a CDN. The HTML elements contain no embedded JavaScript code, therefore a fallback action is always an option as long as the developer keeps these practices in mind when setting up their Web sites.

CHAPTER 2

GETTING STARTED WITH JQUERY

JQUERY IS A POPULAR CHOICE among Web designers and developers because the steps required to get started developing are minimal. You just have to download a copy of the core JavaScript library file and include a `script` tag link to it on the top of your Web site. As with all JavaScript libraries, a JavaScript library `include` must be added to your Web site before you can start using its features in your Web site or application.

This chapter guides you through how to set up an optional local development environment, choose the right jQuery download, and set up the jQuery library for inclusion in your Web site. I also explain what the jQuery wrapper does.

SETTING UP YOUR DEVELOPMENT ENVIRONMENT

To get started setting up your development environment, first, choose which code editor that you want to use. There are many popular choices such as Dreamweaver, Coda, TextMate, and EditPlus to name a few. I primarily use Coda, a Mac OS X software application built for Web designers and developers to code Web sites and applications. This software is an integrated toolset that has features such as FTP, terminal (command prompt), file management, CSS and code editors, syntax highlighting, auto completion, extended find and replace functions, preview, and multi-language support. If you don't want to use a code editor, you are also are free to use regular old Notepad (Windows) or TextEdit (Mac), but you won't get all the increased features that a code editor can provide.

Before you can start writing jQuery code, you need a place to test your work: a development environment. A development environment can be either a local development setup consisting of a local Web server and Web browser or an external Web host. It basically allows you to test any work you are doing in a simulated live environment. The beauty of using an application like Coda or Dreamweaver is that you can set up your external Web host in the application, which allows you to work directly off the server and makes testing a breeze.

Some might argue that you can just work from a local folder and open each file in a Web browser, but this won't give you an accurate view of a live environment. You might end up developing a few jQuery functions like this, but when they go live, you get a different output. I believe it's better to work in an environment that's as close to live as possible from the start.

Local development environment are easy to set up and beneficial for when you aren't able to access the Internet. For Mac users, the most popular choice is MAMP (`www.mamp.info/`), which stands for Mac/Apache/MySQL/PHP, as shown in Figure 2-1. It is an all-in-one development application that you can run locally and test as if you were on a live Web server. For Windows users, the Windows version is called WampServer, which stands for Windows/Apache/mySQL/PHP Server (`www.wampserver.com/en/`). I would suggest whenever possible that you use Apache as a Web server. It's a very stable open source Web server that runs primarily on Linux.

Another alternative for Web developers using Mac OS X is to use the built-in Apache Web server on Mac OS X. To set up Apache server on a Mac, follow these steps:

1. Open System Preferences. You should see a group of icons that allow you to control settings such as Personal, Hardware, Internet and Wireless, System, and Other.
2. Click the Sharing icon to open the Sharing pane and then select the Web Sharing check box in the list of sharing services. The Sharing pane displays settings to allow you to share files, screens, printers, and so on. If the check box is already selected, skip this step.
3. Make sure the Web Sharing check box is selected, as shown in Figure 2-2. You have now started your Apache Web server. You should see a red icon turn green on the right side of the pane and text that states, `Your personal website, in the Sites folder in your home folder, is available at this address: http://xx.xx.xx.xx/~yourname`.

4. Click the IP address to open your default Web browser. It loads your default page. The Web site files (HTML, CSS, JavaScript, Images) are located in your –/Sites directory, similar to the way a Web server is set up. I do all of my local development using this type of setup.

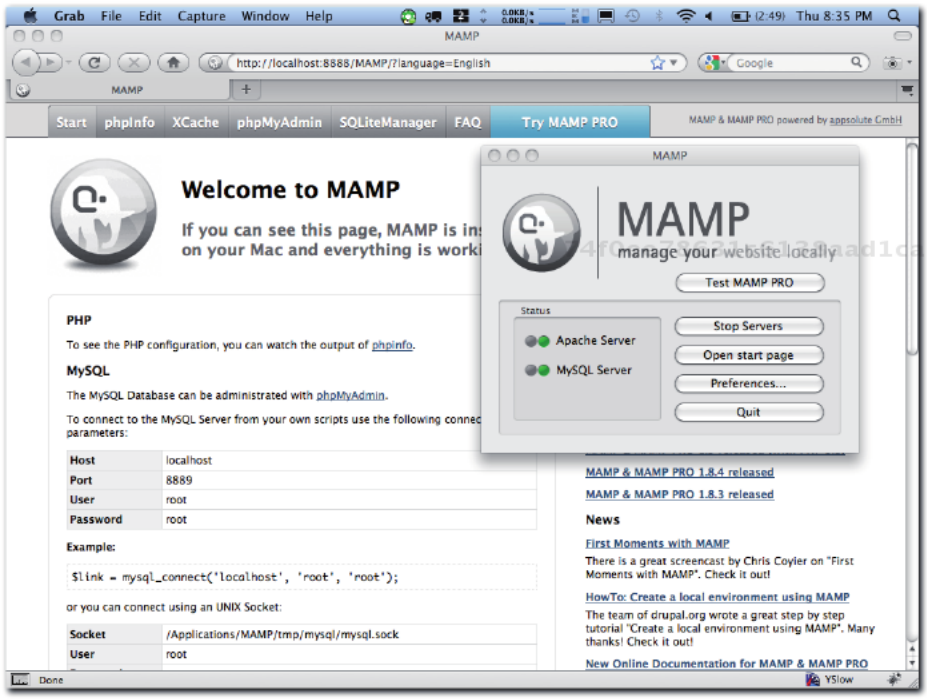

Figure 2-1: MAMP in action on my desktop

USING FIREBUG IN FIREFOX

If you aren't currently using Firefox as one of your main development browsers, I highly suggest that you download it before you proceed any further in this book. For the examples in this book, I use version 3.6.8 of Firefox and version 1.5.4 of Firebug. Joe Hewitt originally created Firebug, which is now an open-source development project, in December of 2006. Since that time, a number of updates to Firebug have been made and more than 1 million developers now use it.

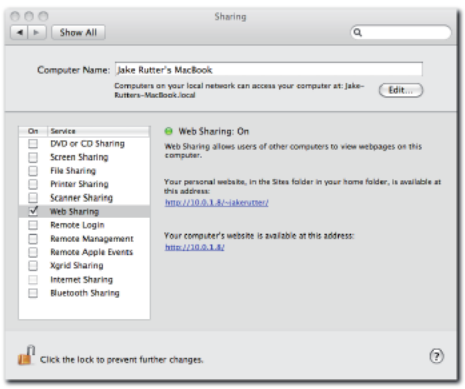

Figure 2-2: The Sharing preferences dialog in Mac OS X

Firebug is an extension that provides a toolset to Web designers and developers who work with HTML, CSS, and JavaScript. Firebug is a free and open-source tool available to everyone through the Firefox extensions Web site (`addons.mozilla.org`).

Firebug allows you to view and edit your HTML and CSS on the fly (see Figure 2-3), and it also has a very powerful JavaScript debugger, which helps in finding errors. The console is a nice feature as you can execute JavaScript directly onto you page from within the console, and it comes in very handy!

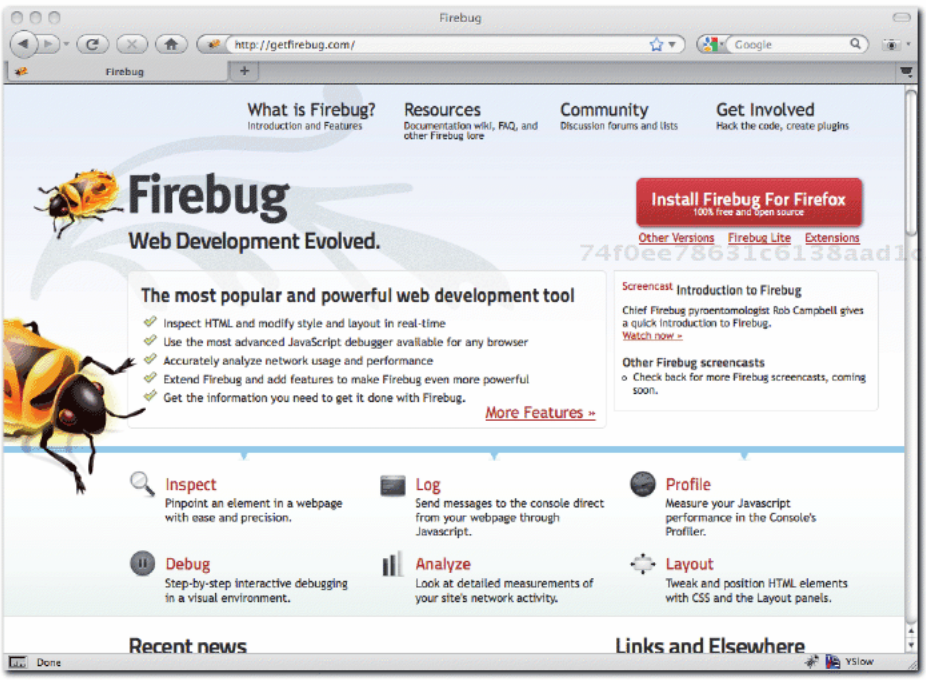

Figure 2-3: The Firebug home page

How to Install Firebug and Enable It

To install and enable Firebug, follow these steps:

1. Open your Firefox browser and go to `www.getFirebug.com`.
2. Click Install Firebug for Firefox.
3. A prompt window, as shown in Figure 2-4, appears that says Install Add-Ons Only from Authors Whom You Trust. On the Install button, a number begins counting down. When it gets to 0, the Install Now button becomes enabled. Click it.
4. Next you see a progress bar that indicates that the plug-in is being installed on your browser. After the plug-in has been installed, you see a confirmation message and a button that says Restart Firefox, as seen in Figure 2-5.
5. Congratulations, you are now ready to begin using Firebug! Figure 2-6 shows the final step of the installation process.

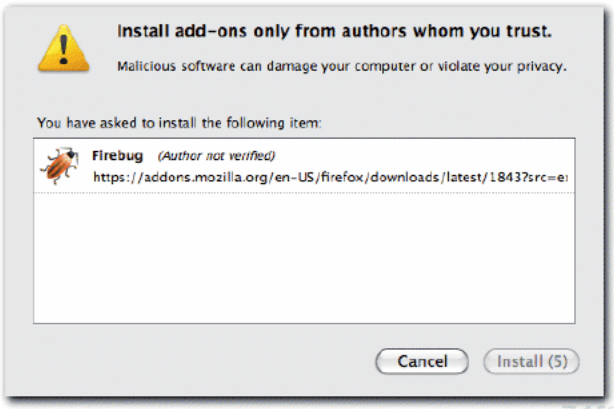

Figure 2-4: The Firebug install prompt

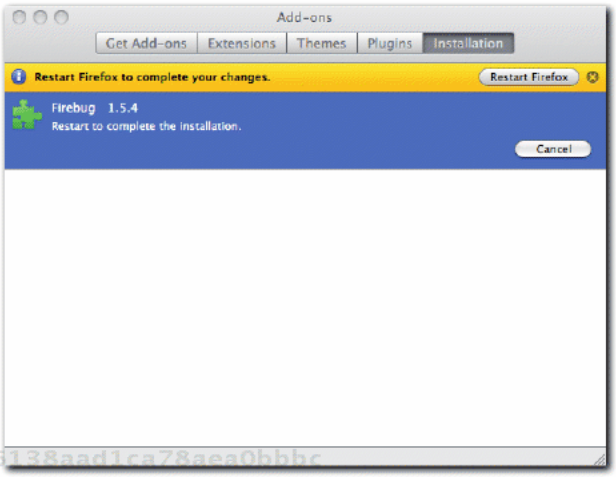

Figure 2-5: The Firebug installation complete window

How to Enable Firebug

To enable Firebug, follow these steps:

1. Open up a Web page in Firefox. For demonstration purposes, I'm going to use `www.mozilla.com`.
2. After you have loaded the page, there are a few ways to open Firebug. The easiest way is by clicking on the Firebug icon in the bottom right of the browser. Figure 2-7 shows an example of Firebug installed and the Firebug icon on the bottom right of the browser.

   You can also start Firebug by right-clicking within the browser window and choosing Inspect Element from the drop-down menu. See Figure 2-8 for an example of the drop-down menu within the browser.

3. After you open Firebug, you see a series of tabs: Console, HTML, CSS, Script, and DOM, among others.

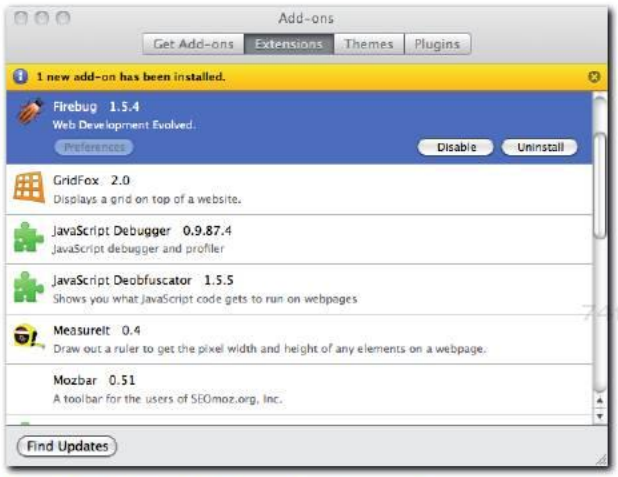

Figure 2-6: The Firebug installation confirmation prompt after you restart your browser

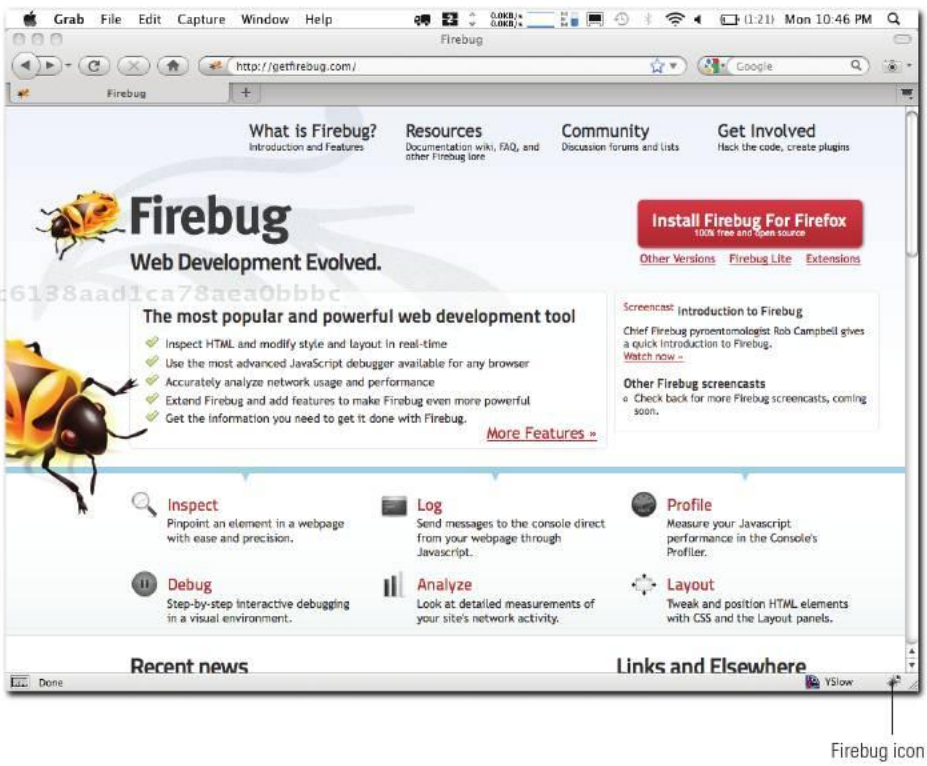

Figure 2-7: A Web page and Firebug installed

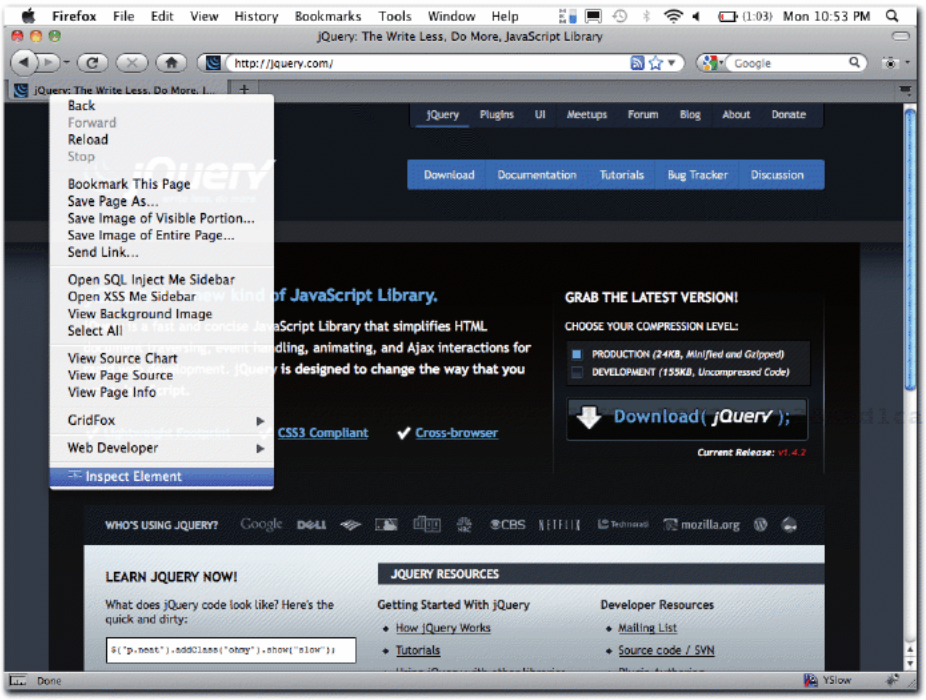

Figure 2-8: How to access Inspect Element to open Firebug

How to Inspect and Edit HTML

The inspect and edit feature of Firebug is very powerful — it makes debugging HTML and JavaScript much easier, especially when you are changing the DOM on the fly. If you have a script that adds/changes HTML, you can open up the inspect window and see the HTML changes in real-time. It's always the first step I take when debugging JavaScript. I always like to first make sure that the HTML is being created properly before proceeding. See Figure 2-9 for an example of what your screen should look like after you have enabled Firebug with the HTML section open.

How to Use the Console

The Firebug console is the second step I take in debugging my JavaScript. After I have fixed any issues with the DOM, I use the console to try running my script live on the page. You see two panels in the console — the left panel is used for showing errors and the right panel is otherwise known as the command line.

1. Open up Firebug and click the Console tab.
2. If there are any errors with your JavaScript, you see them displayed on the left pane. See Figure 2-10 for an example of Firebug open with errors displayed.

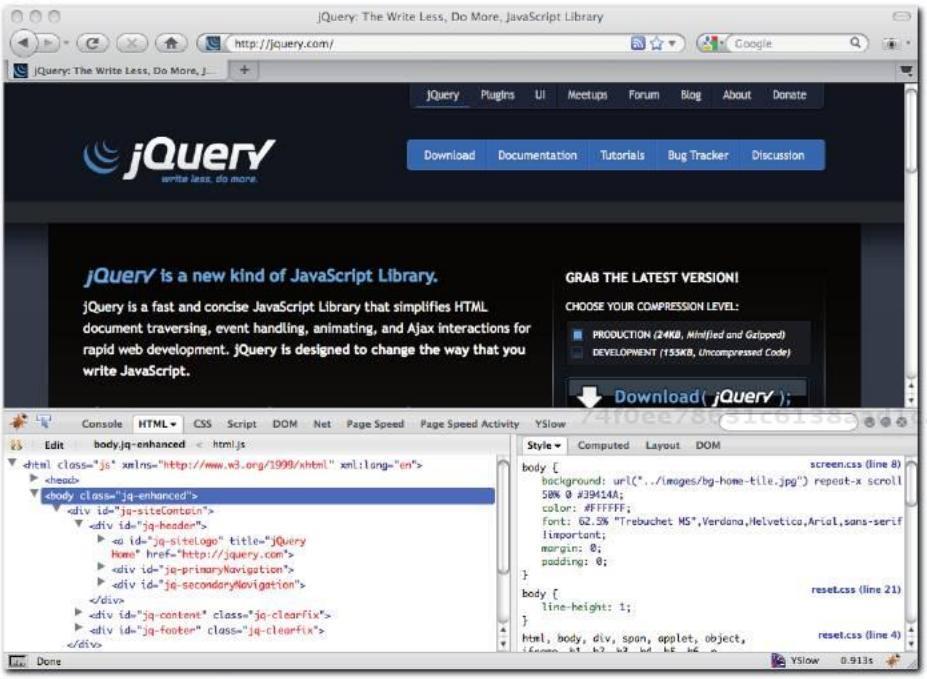

Figure 2-9: The Firebug Edit HTML section

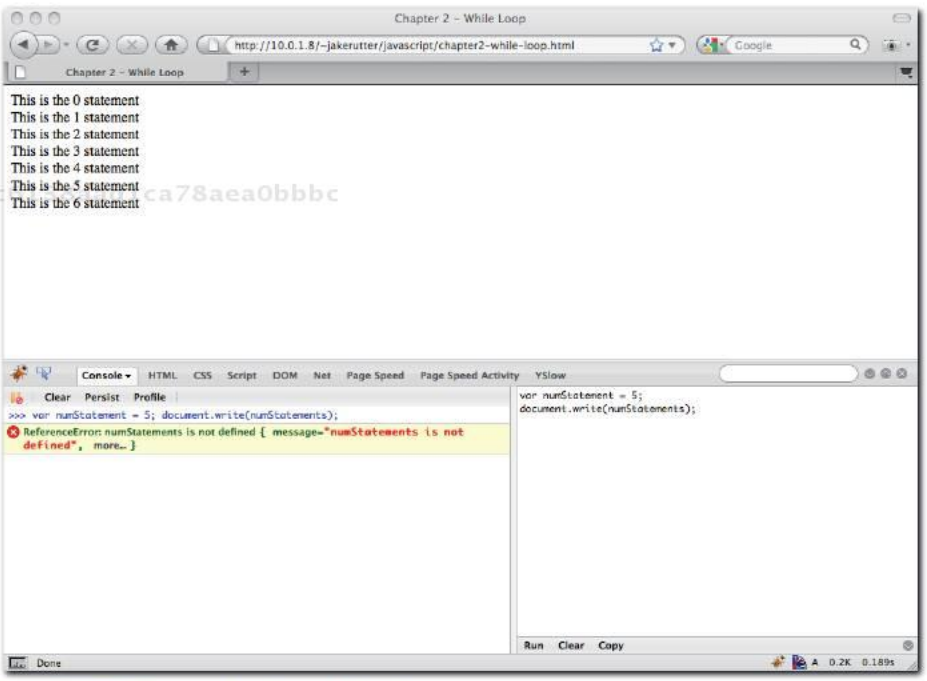

Figure 2-10: The Firebug console with errors displayed

How to Run JavaScript Live in Firefox Through the Firebug Console

The Firebug console is a great way to test jQuery or JavaScript on a page without having to add it to your HTML, run it on a Web server, and so on. The Firebug console also gives feedback in the form of error messages if your JavaScript has a problem. It's a great way to test code before committing to writing it all out in your HTML.

Advanced JavaScript Debugging with Firebug

For more advanced JavaScript applications, you can use the JavaScript debugger tool in Firebug. The JavaScript debugger is a very powerful tool that allows you to add breakpoints to different parts of the script so you can stop, start, and pause your script and take a closer look at variables and objects. I don't focus too much on the JavaScript debugger in this book because it's for more advanced JavaScript programmers.

JavaScript Debugging with Other Web Browsers

Firefox is not the only browser with Web developer tools. Apple Safari, Google Chrome, and Internet Explorer all have similar toolsets, but they are not as powerful as Firefox's Firebug. The Safari/Chrome debugger shares some common features with Firebug, including the inspect element feature and resource management tab, but they lack a powerful debugger such as the one included with Firebug. I have limited experience working with the other tools. I have used IE/Safari for spot-checking, but Firebug is always my main choice for development.

The basics of JavaScript are all concepts that Web designers and developers should be familiar with prior to entering the world of jQuery and JavaScript libraries. That being said, you can use and create basic scripts without needing a solid base in JavaScript, but knowing how JavaScript works accelerates your rate of development and understanding, which leads to high productivity.

DOWNLOADING THE JQUERY LIBRARY

Before you can start developing with jQuery, you first need to download the jQuery library from the jQuery Web site. The jQuery library is a JavaScript file that can be accessed in either of two ways:

- Download the jQuery.js and host it locally on your Web site
- Use a hosted version from a CDN (content delivery network)

I recommend downloading a copy of jQuery to your local computer for development work and testing, and specifically for when an Internet connection is unavailable. To download jQuery, follow these steps:

1. Point your Web browser to the jQuery site at `www.jquery.com`.
2. Click the Download link located in the main navigation bar at the top of the page, which takes you to a page that offers many different ways to access the jQuery Library. Figure 2-11 shows the download page.

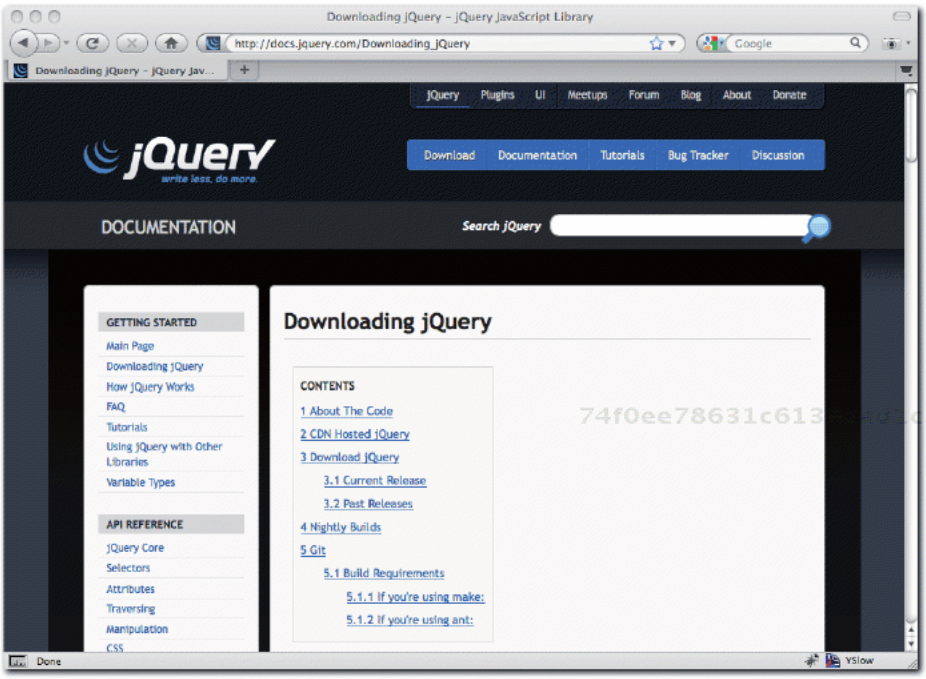

Figure 2-11: The jQuery download page

Many Web sites opt for free CDN-hosted solutions because they have proven to be reliable and fast-loading solutions that free up the bandwidth from your own site. CDN stands for content delivery network, which is a hosted solution provided by companies with large networks such as Google, Microsoft, Akamai, and so on.

Content delivery networks offer the benefit of a larger, high-speed network, which serves up the jQuery library from many locations. When a user brings up your site in their browser, their location triggers the server closest to them geographically to deliver the jQuery library, which in turn decreases the load time. The Google-hosted Ajax libraries page in Figure 2-12 has various options to include jQuery and many other libraries in your Web site or application.

When using the hosted solution, you can specify in the URL which version of jQuery to use, as shown in the following code example using Google Hosted jQuery:

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></
 script>
```

Google and Microsoft are large companies with high market penetration, and many Web sites already use their hosted jQuery libraries. The more Web sites that use hosted versions, the higher the chance that you have already grabbed that file from another Web site, so it's already cached in your browser. (Browser caching occurs when you grab a file from a Web site.)

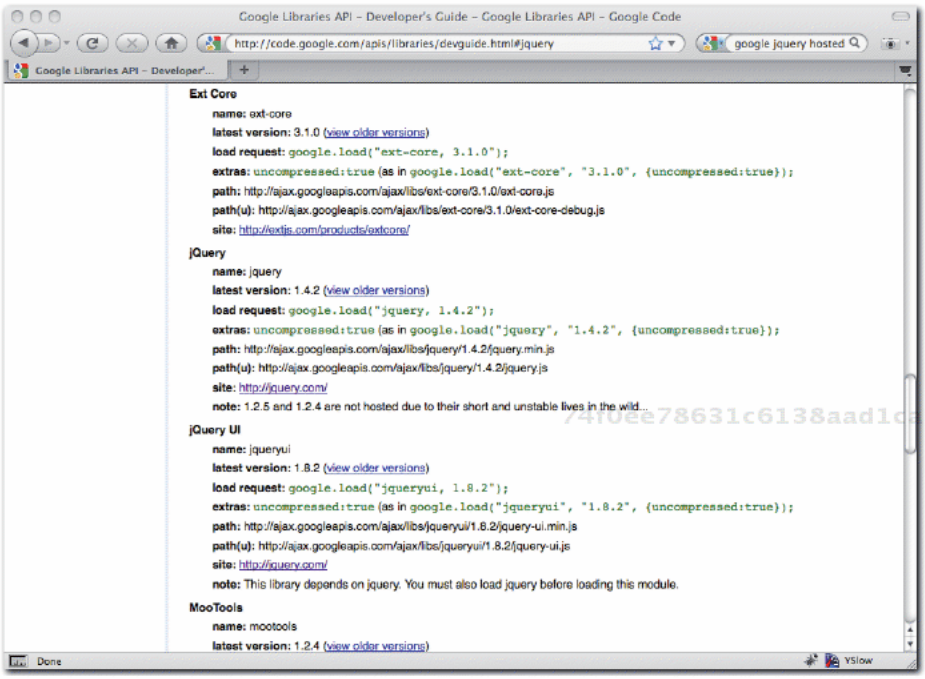

Figure 2-12: The Google-hosted Ajax libraries page

For example, say I'm a daily user of Digg (`www.digg.com`), an online news portal that displays news based on the popularity voted on by its users, which uses the hosted jQuery library from Google, and I decide to add the jQuery library to my Web site. When I pull up my site in the browser, it grabs a cached version of jQuery from my browser cache, unless, when requesting the library, Google sends a modified response back. In that case, the latest jQuery library would be returned instead of the file from my cache.

Table 2.1 outlines the two available versions of jQuery.

Table 2.1 jQuery Library Versions

Format	Description
Uncompressed (Estimated* 155 KB)	Main purpose is for when working in development, for debugging, and for advanced developers who would like to take a look at the code to understand how it works.
Compressed (Estimated* 55 KB)	A much smaller footprint meant for production code. It is compressed using a technique called minification, which removes unnecessary characters such as comments, line breaks, white space, and tabs to improve loading times.

* Estimated for version 1.4.2

Advanced users may want to use Git (source control software) to access the latest version of the jQuery library. I won't be going into any further detail about Git, so if you would like to learn more about it, check out `www.git.com`.

INCLUDING THE JQUERY LIBRARY IN YOUR WEB PAGE

After you have decided which approach you want to take — either downloading a library or using a hosted version of jQuery — you need to set it up in your Web page. The jQuery library can be included within script tags and live in between the `<head></head>` tags of your HTML document, or you can include the jQuery library before your closing `</body>` tag.

You must include any CSS (cascading style sheets) before the jQuery library and jQuery custom code because you want to ensure that all CSS is applied to the DOM (Document Object Model) before you try to change it with jQuery.

You should always include a `doctype` in your HTML pages. Failing to include a `doctype` causes all sorts of erratic behavior within your Web browser and causes your Web page to fail the validation suite check. It helps to ensure that your code works across all browsers. jQuery won't always render correctly if a `doctype` is not present on the page. CSS also renders incorrectly when a `doctype` is not present on the page.

All of the examples in this book use the HTML 5 `doctype` to ensure progressive enhancement and graceful degradation with older Web browsers. (See Chapter 1 if you need a reminder of what progressive enhancement and graceful degradation are.) The `doctype` for HTML5 is very simple to set up compared to previous `doctypes`, which are hard to remember.

The following code is the `doctype` for HTML 5:

```
<!DOCTYPE html>
```


The following example is a very basic HTML document that outlines how to include your jQuery at the top of the page. Always include all CSS files first to ensure that the page has rendered correctly before you try to manipulate the DOM:

```
<!DOCTYPE html>
<html>
  <head>
    <title>My jQuery Example</title>
    <link href="css/global.css"/>
    <script src="js/jquery.js" type="text/javascript"></script>
    <script type="text/javascript">
    //script goes here
    </script>
  </head>
  <body>
  </body>
</html>
```

Alternatively, you can also include your jQuery library at the bottom of the page. This can sometimes increase the speed at which your page loads because the JavaScript doesn't get hung up on loading at the top while the rest of the page continues to load. Also, this way, the entire DOM is guaranteed to load before the JavaScript is applied.

```
<html>
  <head>
    <title>My jQuery Example</title>
    <link href="css/global.css"/>
    </head>
  <body>
  <h1>Hello jQuery!</h1>
  <div id="page-container">
    <p>You can place your jQuery at the end of the page too!</p>
</div>
  <script src="js/jquery.js" type="text/javascript"></script>
  <script type="text/javascript">
//script goes here
  </script>
  </body>
</html>
```

If you are using the Google-hosted version of the jQuery `include`, instead of including a relative path, insert the direct path to the library provided by Google:

```
<html>
  <head>
  <title>My jQuery Example</title>
  <link href="css/global.css"/>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"
 type="text/javascript"></script>
  <script type="text/javascript">
   //script goes here
  </script>
  </head>
  <body>
  </body>
</html>
```


UNDERSTANDING THE JQUERY WRAPPER

Before you get started programming with jQuery, you need to understand what the jQuery wrapper is and how it applies to the DOM. A wrapper, in most programming languages, is something that wraps something else to extend the functionality, most often an object. To put this in perspective, the jQuery wrapper attaches itself to the DOM by using selectors and allows you to extend the DOM. jQuery doesn't actually offer any new methods; it just takes methods that already exist in native JavaScript and makes them much easier to interact with.

The power of the wrapper is being able to extend the DOM with much less code than native JavaScript. The following code is an example of the jQuery selector statement:

```
$.(selector)
```

jQuery has many event methods to choose from, but one very important one is called the `document.ready()` event handler method, which executes only after the DOM is fully loaded. In its simplest form, a method is just another way of describing a function, but in other OOP (object-oriented programming) languages, methods have increased benefits compared to functions. The power behind jQuery is in manipulating the DOM; therefore, you want to make sure the DOM is ready before you do anything with it.

The `document.ready()` event handler method allows you to put all of your JavaScript jQuery code within this event to make sure the code is executed when the DOM is ready. This event is similar to the JavaScript `onLoad` event, except the `document.ready()` event handler method only fires after the DOM has loaded.

The following code is an inline example of how to set up the `document ready` event handler method:

```
<html>
  <head>
  <title>My jQuery Example</title>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"
 type="text/javascript"></script>
  <script type="text/javascript"></>
  $(document).ready(function() {
  //script goes here
  });
  </script>
  </head>
  <body>
  </body>
</html>
```


You should get in the habit of setting up the `document.ready()` event handler and any other custom jQuery code in an external file. This is the method I prefer because it keeps all your code separate in its own JavaScript include. I usually call my external file with all of my jQuery code, jquery.function.js, and always make sure it's included last, after all jQuery core library file(s).

You can also use a shortened version of the `document.ready()` event handler method when you are trying to optimize the size of the jQuery functions file for increased performance, such as for a mobile application. The shortened version is set up like this:

```
<html>
  <head>
```

```
  <title>My jQuery Example</title>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"
 type="text/javascript"></script>
  <script type="text/javascript"></>
  $(function() {
  //script goes here
  });
  </script>
  </head>
  <body>
  </body>
</html>
```

To explain the jQuery wrapper, I need to walk you through how to set up the `document.ready()` statement. The first step involves setting up a selector that is preceded by the dollar sign ($), which is the alias for accessing the jQuery itself. You pass the selector between the two parentheses; in this case, I'm passing the document selector for the DOM. The alias and the selector make up the jQuery wrapper.

```
$(document)
```

The ready event gets attached after the selector statement and is interchangeable with other events.

```
.ready()
```


The function is the piece that holds the code, which is applied after the DOM is loaded and ready and does not include graphics. The function is placed within the parentheses of the ready event because you are passing the function you want to be run to the ready event:

```
.ready(function() {
  //jQuery DOM code goes here
  alert("The DOM is fully loaded and ready");
});
```

The `window.load` function is very similar to the `document.ready()` function, except that it also waits for all of the graphics on the page to load before executing any jQuery code:

```
$(window).load {
    //jQuery Code Goes Here
    alert("The window has been loaded");
});

 $(window).load {
    //jQuery Code Goes Here
    alert("The window has been loaded");
});
```

RUNNING CODE OUTSIDE OF DOCUMENT READY HANDLER

Most jQuery-specific code needs to be set up within a `document.ready()` event handler method. But native JavaScript such as variables, arrays, and so on can be set up outside of the document ready event handler because they don't need to wait for the DOM to be ready and are hidden from the DOM as they are specifics within the actual script.

The following code example outlines a script that relies on the DOM to be loaded before new content can be added. There are three variables being set in this script — two of them outside of the `document.ready()` event and one being set inside for the `document.ready()` event as it needs access to the `for` loop set up inside on it.

```
<!doctype html>
<html>
  <head>
    <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></
  script>
    <script>
    var numShows = 10;
    var numTickets = 100;
    $(document).ready(function() {
     for(i=0; i < numTickets; i++)
     {
     var numTotal = i + 1;
$('.container').append("<p>There are " + numTotal + " tickets available</p>");
     }
    });
</script>
<body>
  <div class="container">
  </div>
</body>
</html>
```


PREVENTING CONFLICTS WITH OTHER LIBRARIES

Conflicts can occur with other JavaScript libraries if you don't take the proper precautions when writing your jQuery. Most conflicts occur with the use of the $ alias, which Prototype also shares as an alias. You need to take two steps to eliminate conflicts with others libraries:

1. Add the `noConflict` function at the end of your jQuery Library. The `noConflict` function releases all dependability of the jQuery on the `$` alias back to any other libraries that are also using it.

   ```
   $.noConflict();
   ```

2. Change all references of the `$` alias to the jQuery alias as demonstrated in the following example. Change this:

   ```
   $(document).ready() {
      //code goes here
   ```

```
});
```

To this:

```
jQuery(document).ready() {
   //code goes here
});
```

You can also define your own alias if you don't wish to use the jQuery alias. This is done by adding a line of JavaScript to define your own alias. In the following example, I set up the new alias as `$alien`, instead of just $. It's that easy!

```
var $alien = jQuery;

$alien(document).ready() {
   // code goes here
});
```


USING JAVASCRIPT WITH JQUERY

Variables are a great way to store types of information, especially when writing JavaScript. I use variables frequently with jQuery and do so throughout the examples later in the book. Using variables when writing jQuery is no different than using them with JavaScript. You can set variables and call them within the jQuery wrapper because it's all basically just JavaScript.

The beauty of jQuery is that it is JavaScript, so if you have any prior knowledge of JavaScript, you can directly apply that knowledge to jQuery. You don't have to worry about learning new syntax, conventions, or methods because most of jQuery is based on JavaScript functionality, but the syntax is much easier to understand.

You may be wondering, if jQuery is JavaScript, why not just learn JavaScript? The answer is that jQuery takes everything that JavaScript does and makes it much easier to implement. The jQuery tagline — "Write less, do more" — definitely holds true. You can take 20 lines of native JavaScript and turn it into 5 lines of jQuery without having to know JavaScript. For those of you who are curious about JavaScript, learning jQuery can help you to understand the JavaScript API.

JavaScript can be hard to understand. jQuery makes it much easier for Web designers to implement features from the JavaScript API without having to understand all the complexities of JavaScript itself. jQuery has really opened up the door for so many Web designers who previously had little programming experience to add interactivity to their Web sites. It's a great time to be working with jQuery: The support and community around this library is growing at an astounding rate.