BEGINNER TO PRO

murach's JavaScript and jQuery

3RD EDITION

Mary Delamater

Zak Ruvalcaba

MIKE MURACH & ASSOCIATES, INC.

4340 N. Knoll Ave. • Fresno, CA 93722

www.murach.com • murachbooks@murach.com

Editorial team

Authors: Mary Delamater
Zak Ruvalcaba

Editor: Anne Boehm

Production: Maria Spera

Books for web developers

Murach's HTML5 and CSS3 (3rd Edition)
Murach's JavaScript (2nd Edition)
Murach's PHP and MySQL (2nd Edition)
Murach's Java Servlets and JSP (3rd Edition)
Murach's ASP.NET 4.6 Web Programming with C# 2015
Murach's ASP.NET Web Programming with VB

Books on core Python, Java, C#, and VB

Murach's Python Programming
Murach's Beginning Java with NetBeans
Murach's Beginning Java with Eclipse
Murach's Java Programming (4th Edition)
Murach's C# 2015
Murach's Visual Basic 2015

Books for database programmers

Murach's MySQL (2nd Edition)
Murach's Oracle SQL and PL/SQL (2nd Edition)
Murach's SQL Server 2016 for Developers

For more on Murach books, please visit us at www.murach.com

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1
ISBN: 978-1-943872-05-3

Contents

Introduction xv

Section 1 JavaScript essentials

Chapter 1 Introduction to web development 3
Chapter 2 Getting started with JavaScript 51
Chapter 3 The essential JavaScript statements 85
Chapter 4 How to work with JavaScript objects, functions, and events 111
Chapter 5 How to test and debug a JavaScript application 141
Chapter 6 How to script the DOM with JavaScript 163
Chapter 7 How to work with links, images, and timers 201

Section 2 jQuery essentials

Chapter 8 Get off to a fast start with jQuery 225
Chapter 9 How to use effects and animations 261
Chapter 10 How to work with forms and data validation 289
Chapter 11 How to use jQuery plugins and jQuery UI widgets 307
Chapter 12 How to use Ajax, JSON, and Flickr 335

Section 3 Advanced JavaScript skills

Chapter 13 How to work with numbers, strings, and dates 373
Chapter 14 How to work with control structures, exceptions, and regular expressions 401
Chapter 15 How to work with browser objects, cookies, and web storage 435
Chapter 16 How to work with arrays 465
Chapter 17 How to create and use your own objects 497
Chapter 18 How to create and use closures, IIFEs, the module pattern, and plugins 543

Reference Aids

Appendix A How to set up your computer for this book 589
Index 599

8

Get off to a fast start with jQuery

In this chapter, you'll quickly see how jQuery makes JavaScript programming easier. Then, you'll learn a working subset of jQuery that will get you off to a fast start. Along the way, you'll study four complete applications that will show you how to apply jQuery.

Introduction to jQuery 226
What jQuery is 226
How jQuery can simplify JavaScript development 228

The basics of jQuery programming 230
How to include jQuery in your web pages 230
How to code jQuery selectors 232
How to call jQuery methods 234
How to use jQuery event methods 236

The Email List application in jQuery 238
The user interface and HTML 238
The jQuery 240

A working subset of selectors, methods, and event methods 242
The most useful selectors 242
The most useful methods 244
The most useful event methods 246
Other event methods that you should be aware of 248

Three illustrative applications 250
The FAQs application in jQuery 250
The Image Swap application in jQuery 252
The Image Rollover application in jQuery 256

Perspective 258

Introduction to jQuery

In this introduction, you'll learn what jQuery is, how to include it in your applications, and how jQuery, jQuery UI, and plugins can simplify JavaScript development.

What jQuery is

As figure 8-1 summarizes, *jQuery* is a free, open-source, JavaScript library that provides dozens of methods for common web features that make JavaScript programming easier. Beyond that, the jQuery functions are coded and tested for cross-browser compatibility, so they will work in all browsers.

Those are just two of the reasons why jQuery is used by the majority of the one million most-visited websites today. And that's why jQuery is commonly used by professional web developers. In fact, you can think of jQuery as one of the four technologies that every web developer should know how to use: HTML, CSS, JavaScript, and jQuery. But don't forget that jQuery is actually JavaScript.

The jQuery website at jquery.com

What jQuery offers

- Dozens of selectors, methods, and event methods that make it easier to add JavaScript features to your web pages
- Cross-browser compatibility
- Selectors that are compliant with CSS3
- A compressed library that loads quickly so it doesn't degrade performance

Description

- *jQuery* is a free, open-source, JavaScript library that provides methods that make JavaScript programming easier.
- Today, jQuery is used by the majority of the one million most-visited websites, and its popularity is still growing.

Figure 8-1 What jQuery is

How jQuery can simplify JavaScript development

To show you how jQuery can simplify JavaScript development, figure 8-2 shows the jQuery for the FAQs application that you learned how to develop in chapter 6. If you're like most people, you probably found the JavaScript code for that application both complicated and confusing.

In contrast, this jQuery code takes significantly fewer lines of code. You'll also find that it is much easier to understand once you learn how to use the JQuery selectors, methods, and event methods. And you'll start learning those skills right after this introduction.

Incidentally, jQuery uses CSS selectors to select the HTML elements that the methods should be applied to. For instance,

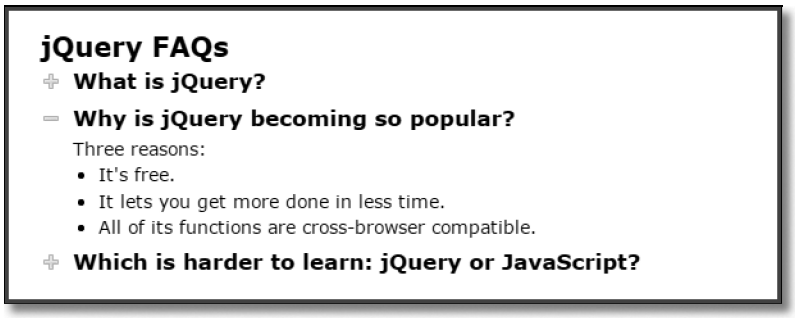
```
$("#faqs h2")
```

is a jQuery selector for the CSS selector

```
#faqs h2
```

which selects all of the h2 elements in the element with "faqs" as its id. In fact, jQuery supports all of the CSS selectors including the CSS3 selectors, even in browsers that don't support all of the CSS3 selectors. This is another reason why developers like jQuery.

The FAQs application in a browser



The HTML

```
<main id="faqs">
    <h1>jQuery FAQs</h1>
    <h2><a href="#">What is jQuery?</a></h2>
    <div>
        <p>jQuery is a library of the JavaScript functions that you're most
           likely to need as you develop websites.</p>
    </div>
    <h2><a href="#">Why is jQuery becoming so popular?</a></h2>
    <div>
        ...
    </div>
    ...
    ...
</main>
```

The critical CSS

```
h2 { background: url(images/plus.png) no-repeat left center; }
h2.minus { background: url(images/minus.png) no-repeat left center; }
div { display: none; }
```

The jQuery for the application

```
$(document).ready(function() {
    $("#faqs h2").click(function() {
        $(this).toggleClass("minus");
        if ($(this).attr("class") != "minus") {
            $(this).next().hide();
        }
        else {
            $(this).next().show();
        }
    }); // end click
}); // end ready
```

Description

- The application illustrates the use of the ready event method, a jQuery selector that's just like a CSS selector (#faqs h2"), and five jQuery methods. All of these make jQuery programming easier than JavaScript programming.

Figure 8-2 How jQuery can simplify JavaScript development

The basics of jQuery programming

In the next four figures, you're going to learn the basics of jQuery programming. Then, you'll study an application that uses these skills. That will show you how jQuery simplifies JavaScript programming.

How to include jQuery in your web pages

If you go to the web page that's shown in figure 8-3, you'll see that it contains links that let you download various releases of the jQuery core library. At this writing, the most current release is jQuery 3.1.1, and it comes in four versions: uncompressed, compressed, slim uncompressed, and slim compressed.

The uncompressed versions allow you to study the JavaScript code that's used in the library. But beware, this code is extremely complicated. In contrast, all whitespace has been removed from the compressed versions, which makes it almost impossible for a human to read them. But they are much smaller, which improves performance. The slim version of jQuery was introduced with jQuery 3.0. It doesn't include the ajax or effects modules, so it's even smaller.

Once you've downloaded the jQuery library you want, you can include it in a web page by coding a script element like the first one in this figure. Then, if you store the file on your own computer or a local web server, you'll be able to develop jQuery applications without being connected to the Internet.

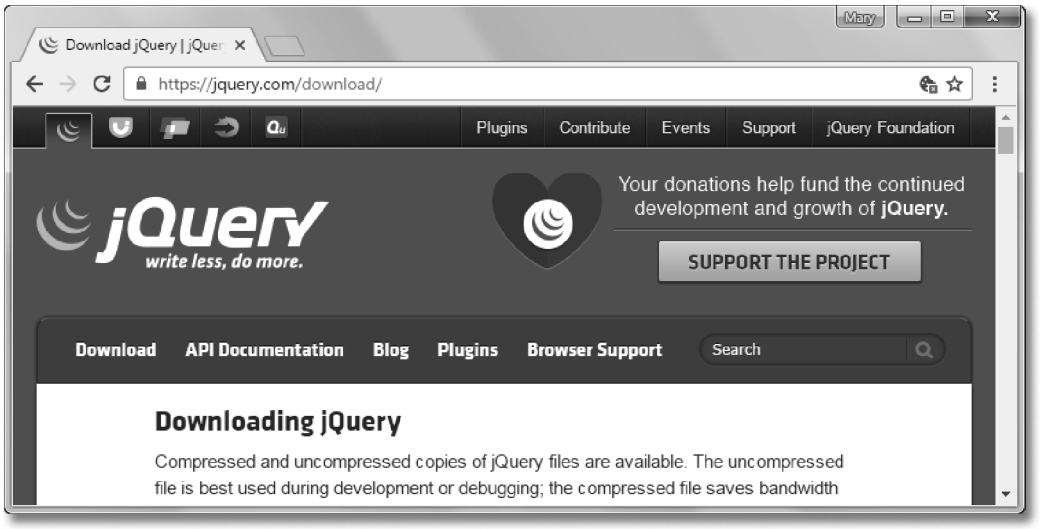
The other way to include the jQuery library in your web applications is to get the file from a *Content Delivery Network* (*CDN*). A CDN is a web server that hosts open-source software, and the download page shown here contains a link to the jQuery CDN. The next two examples in this figure show script elements that use the jQuery CDN with URLs that get version 3.1.1. Because using a CDN is a best practice, all the applications in this book use this technique.

Although you might think that you should always use the most current release of jQuery, that's not necessarily the case. As you can see in the table in this figure, different versions of jQuery implemented various changes over time. For example, the 2.x and 3.x branches of jQuery don't provide support for IE6, IE7, or IE8. So if you think that some of your users may be using those browsers, you'll want to use jQuery 1.12.4 since it's the most current release of the 1.x branch that supports them.

Also, jQuery versions 1.9 and 3.0 dropped certain features that were included in earlier versions. So if you have old code that uses those features, you'll need to keep using the older versions of the library. Or, you can use the jQuery Migrate plugins presented here to upgrade your application. To use these plugins, go to jquery.com/upgrade-guide, click on the upgrade you want, and then follow the step-by-step instructions.

Finally, you should know that jQuery recommends that you use *Subresource Integrity (SRI) checking* when you link to files on their CDN. This helps ensure that resources hosted on the CDN haven't been tampered with. The last example in this figure shows how this works. You can get more information on the jQuery CDN website and at https://www.srihash.org/. For brevity, the applications in this book don't use SRI checking.

The jQuery download page at jquery.com/download



How to include jQuery 3.1.1 after you've downloaded it to your computer

```
<script src="jquery-3.1.1.min.js"></script>
```

How to include jQuery 3.1.1 from a Content Delivery Network (CDN)

```
<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
```

How to include the slim version of jQuery 3.1.1 from a CDN

```
<script src="https://code.jquery.com/jquery-3.1.1.slim.min.js"></script>
```

The most important releases of jQuery

Version	Release date	Description
1.0	Aug 2006	First stable release.
1.9.1	Feb 2013	Deprecated interfaces removed.
1.12.4	May 2016	Last updated version of 1.x branch, which supports IE6-8.
2.0	April 2013	First version of 2.x branch. Dropped support for IE6-8.
2.2.4	May 2016	Last updated version of 2.x branch.
3.0	June 2016	First version of 3.x branch. A smaller, faster version of the 2.x branch, but with some breaking changes. Adds a slim version.
3.1.1	Sept 2016	Current version of jQuery.

Two jQuery migrate plugins

Version	Description
1.4.1	Restores features dropped by version 1.9.
3.0.0	Restores features dropped by version 3.0.

How to include SRI checking with the jQuery CDN

```
<script src="https://code.jquery.com/jquery-3.1.1.min.js"
integrity="sha256-hVVnYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGmIJLv2b8="
crossorigin="anonymous"></script>
```

Figure 8-3 How to include jQuery in your web pages

How to code jQuery selectors

When you use jQuery, you start by selecting the element or elements that you want to apply a jQuery method to. To do that, you can use jQuery *selectors* as shown in figure 8-4.

To code a jQuery selector, you start by coding the dollar sign (`$`) followed by a set of parentheses that contains a set of quotation marks. Then, within the quotation marks, you code the CSS selector for the element or elements that you want to select. This is shown by the syntax summary at the top of this figure.

The HTML and the examples that follow show how easy it is to select one or more elements with jQuery. For instance, the first selector in the first group of examples selects all <p> elements within the entire document. The second selector selects the element with "faqs" as its id. And the third selector selects all elements with "minus" as the value of its class attribute.

In the second group of examples, you can see how other types of CSS selectors are coded with jQuery. Here, you can see how descendants, adjacent siblings, general siblings, and children are coded. For instance, the first selector gets all <p> elements that are descendants of the element with "faqs" as its id. That includes all of the <p> elements in the HTML in this figure.

In contrast, the second selector gets the div elements that are adjacent siblings to the h2 elements, which includes all of the div elements. The third selector gets all <p> elements that are siblings of ul elements, which selects the one <p> element in the second div element. And the fourth selector gets all ul elements that are children of div elements, which selects the ul element in the second div element.

The third group of examples shows how to code multiple selectors. To do that, you separate them with commas, just as you do with CSS.

The syntax for a jQuery selector

```
$("selector")
```

The HTML for the elements that are selected by the examples

```
<main id="faqs">
    <h1>jQuery FAQs</h1>
    <h2 class="minus"><a href="#">What is jQuery?</a></h2>
    <div>
        <p>jQuery is a library of the JavaScript functions that you're most
           likely to need as you develop websites.
        </p>
    </div>
    <h2><a href="#">Why is jQuery becoming so popular?</a></h2>
    <div>
        <p>Three reasons:</p>
        <ul>
            <li>It's free.</li>
            <li>It lets you get more done in less time.</li>
            <li>All of its functions are cross-browser compatible.</li>
        </ul>
    </div>
</main>
```

How to select elements by element, id, and class

By element type: All <p> elements in the entire document

```
$("p")
```

By id: The element with "faqs" as its id

```
$("#faqs")
```

By class: All elements with "minus" as a class

```
$(".minus")
```

How to select elements by relationship

Descendants: All <p> elements that are descendants of the main element

```
$("#faqs p");
```

Adjacent siblings: All div elements that are adjacent siblings of h2 elements

```
$("h2 + div")
```

General siblings: All <p> elements that are siblings of ul elements

```
$("ul ~ p")
```

Children: All ul elements that are children of div elements

```
$("div > ul")
```

How to code multiple selectors

```
$("#faqs li, div p")
$("p + ul, div ~ p")
```

Description

- When you use jQuery, the dollar sign (**$**) is used to refer to the jQuery library. Then, you can code *selectors* by using the CSS syntax within quotation marks within parentheses.

Figure 8-4 How to code jQuery selectors

How to call jQuery methods

Once you've selected the element or elements that you want to apply a *method* to, you call the method using the syntax shown at the top of figure 8- 5. This is the same way that you call a method of any object. You code the selector that gets the element or elements, the dot, the method name, and any parameters within parentheses.

To get you started with jQuery, the table in this figure summarizes some of the jQuery methods that you'll use the most. For instance, the val() method without a parameter gets the value from a selected text box or other form control, and the val() method with a parameter sets the value in a selected text box or other form control. The first two examples after the table show how this works.

Similarly, the text() method without a parameter can be used to get the text of a selected element, and the text() method with a parameter can be used to set the text of a selected element. Methods like these are often referred to as *getter* and *setter* methods. Here, the third example illustrates the setter version of the text() method, which sets the text of an element to "Email address is required".

The fifth method in the table is the next() method, which is used to get the next (or adjacent) sibling of an element. This method is often followed by another method. To do that, you use *object chaining*, which works just as it does with JavaScript. This is illustrated by the fourth example. Here, the next() method gets the next sibling after the element that has been selected, and the text() method sets the text for that sibling.

In most cases, you'll use the val(), text(), and next() methods with selectors that select a single control or element. However, you can also use them with selectors that select two or more controls or elements. In that case, the set form of the val() and text() methods set the values of all the matching controls or elements. In contrast, the get form of the val() method gets the value of just the first matching form control. The get form of the text() method gets the combined text of all the matching elements. And the next() method gets the siblings of all the matched elements.

The last two methods in the table are the submit() and focus() methods, which are just like the JavaScript submit() and focus() methods. The submit() method submits the data for a selected form to the server, and the focus() method moves the focus to the selected form control or link.

In a moment, you'll see how these selectors and methods work in an application. But first, you need to learn how to set up the event handlers for an application.

The syntax for calling a jQuery method

```
$("selector").methodName(parameters)
```

Some common jQuery methods

Method	Description
`val()`	Get the value of a text box or other form control.
`val(value)`	Set the value of a text box or other form control.
`text()`	Get the text of an element.
`text(value)`	Set the text of an element.
`next([type])`	Get the next sibling of an element or the next sibling of a specified type if the parameter is coded.
`submit()`	Submit the selected form.
`focus()`	Move the focus to the selected form control or link.

Examples

How to get the value from a text box

```
var gallons = $("#gallons").val();
```

How to set the value for an input element

```
$("#gallons").val("");
```

How to set the text in an element

```
$("#email_address_error").text("Email address is required");
```

How to set the text for the next sibling with object chaining

```
$("#last_name").next().text("Last name is required");
```

How to submit a form

```
$("#join_list").submit();
```

How to move the focus to a form control or link

```
$("#email_address").focus();
```

Description

- To call a jQuery *method*, you code a selector, the dot operator, the method name, and any parameters within parentheses. Then, that method is applied to the element or elements that are selected by the selector.
- When you use *object chaining* with jQuery, you code one method after the other. This works because each method returns the appropriate object.
- Although you'll typically use the val(), text(), and next() methods with selectors that select a single form control or element, you can also use them with selectors that select two or more form controls or elements. The result depends on the method you use.

Figure 8-5 How to call jQuery methods

How to use jQuery event methods

When you use jQuery, you use *event methods* to attach event handlers to events. To do that, you use the syntax shown at the top of figure 8-6. First, you code the selector for the element that will initiate the event like a button that will be clicked. Then, you code the name of the event method that represents the event that you want to use. Last, you code a function that will be the event handler for the event within parentheses.

In the table in this figure, the two event methods that you'll use the most are summarized. The ready event is the jQuery alternative to the JavaScript load event. The ready event works better than the load event, though, because it's triggered as soon as the DOM is built, even if other elements like images are still being loaded into the browser. This means that the user can start using the web page faster.

Because the DOM usually has to be built before you can use JavaScript or jQuery, you'll probably use the ready() event method in every JavaScript application that you develop. The examples in this figure show two ways to do that. In the long form, you use document as the selector for the web page followed by the dot, the method name (ready()), and the function for the event handler.

In the short form, you can omit the selector and event method name and just code the function in parentheses after the dollar sign. Although this form is often used by professional developers, all of the examples in this book use the long form. That way, it's clear where the ready event handler starts.

The next example in this figure shows an event handler for the click event of all h2 elements. This is coded just like the event handler for the ready event except h2 is used as the selector and click() is used as the name of the event method.

The last example in this figure shows how you code an event handler within the ready() event handler. Note here that the closing brace, parenthesis, and semicolon for each event handler is critical. As you can guess, it's easy to omit one of these marks or get them out of sequence, so this is a frequent source of errors. That's why professional programmers often code inline comments after the ending marks for each event handler to identify which event handler the marks are for.

The syntax for a jQuery event method

```
$(selector).eventMethodName(function() {
    // the statements of the event handler
});
```

Two common jQuery event methods

Event method	Description
`ready(handler)`	The event handler runs when the DOM is ready.
`click(handler)`	The event handler runs when the selected element is clicked.

Two ways to code an event handler for the jQuery ready event

The long way

```
$(document).ready(function() {
    alert("The DOM is ready");
});
```

The short way

```
$(function(){                 // (document).ready is assumed
    alert("The DOM is ready");
});
```

An event handler for the click event of all h2 elements

```
$("h2").click(function() {
    alert("This heading has been clicked");
});
```

The click event handler within the ready event handler

```
$(document).ready(function() {
    $("h2").click(function() {
        alert("This heading has been clicked");
    }); // end of click event handler
}); // end of ready event handler
```

Description

- To code a jQuery event handler, you code a selector, the dot operator, the name of the jQuery *event method*, and an anonymous function that handles the event within parentheses.
- The event handler for the ready event will run any methods that it contains as soon as the DOM is ready, even if the browser is loading images and other content for the page. This works better than the JavaScript onload event, which doesn't occur until all of the content for the page is loaded.
- In this book, the ready event is always coded the long way that's shown above. In practice, though, many programmers use the short way.
- When coding one event handler within another, the use of the closing braces, parentheses, and semicolons is critical. To help get this right, many programmers code inline comments after these punctuation marks to identify the ends of the handlers.

Figure 8-6 How to use jQuery event methods

The Email List application in jQuery

With that as background, you're ready to see how jQuery can be used in the Email List application that you studied in section 1. That will show you how jQuery can simplify coding.

The user interface and HTML

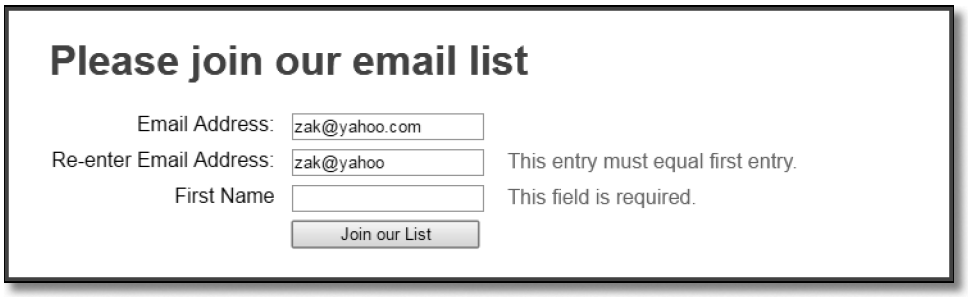
To refresh your memory, figure 8-7 presents the user interface and HTML for the Email List application. To use the application, the user enters text into the first three text boxes and clicks on the Join our List button. Then, the JavaScript validates the entries and displays appropriate error messages if errors are found. If no errors are found, the data in the form is submitted to the web server for processing.

In the HTML, note first the script element that loads jQuery. It is followed by the script element that identifies the file that holds the JavaScript for this application. That sequence is essential because the JavaScript file is going to use the jQuery file.

Also note that the span elements are adjacent siblings to the input elements for the text boxes. The starting text for each of these span elements is an asterisk that indicates that the text box entry is required. Later, if the JavaScript finds errors in the entries, it displays error messages in these span elements.

Finally, note that, unlike the Email List application you saw earlier, the span elements in this version don't have id attributes. That's because the jQuery code will use some of the methods you just learned to locate the span element it needs. You'll see how that works next.

The user interface for the Email List application



The HTML

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Join Email List</title>
    <link rel="stylesheet" href="email_list.css">
    <script src="http://code.jquery.com/jquery-3.1.1.min.js"></script>
    <script src="email_list.js"></script>
</head>
<body>
    <main>
        <h1>Please join our email list</h1>
        <form id="email_form" name="email_form"
              action="join.html" method="get">
            <label for="email_address1">Email Address:</label>
            <input type="text" id="email_address1">
            <span>*</span><br>

            <label for="email_address2">Re-enter Email Address:</label>
            <input type="text" id="email_address2">
            <span>*</span><br>

            <label for="first_name">First Name:</label>
            <input type="text" id="first_name">
            <span>*</span><br>

            <label> </label>
            <input type="button" id="join_list" value="Join our List">
        </form>
    </main>
</body>
</html>
```

Figure 8-7 The user interface and HTML for the Email List application

The jQuery

Figure 8-8 presents the jQuery for this application. This is the code in the email_list.js file that's included by the HTML. Here, all of the jQuery is highlighted. The rest of the code is JavaScript code.

To start, you can see that an event handler for the click event of the Join our List button is coded within the event handler for the ready event. Then, if you look at the last three lines of code, you can see the ending punctuation marks for these handlers. Within the click event handler, the first two statements show how jQuery selectors and the val() method can be used to get the values from text boxes.

In the first if statement, you can see how an error message is displayed if the user doesn't enter an email address in the first text box. Here, the next() method gets the adjacent sibling for the text box, which is the span element, and then the text() method puts an error message in that span element. This changes the DOM, and as soon as it is changed, the error message is displayed in the browser.

The next(), text(), and val() methods are used in similar ways in the next two if statements. Then, the fourth if statement tests to see whether the isValid variable is still true. If it is, the submit() method of the form is issued, which sends the data to the web server.

That ends the event handler for the click event of the Join our List button. But that handler is followed by one more statement. It moves the focus to the first text box, the one with "email_address1" as its id.

As you review this code, note that it doesn't require the standard $ function that gets an element object when the element's id is passed to it. Instead, the $ sign is used to start a jQuery selector that gets elements by their ids. This simplifies the coding.

Although this jQuery code illustrates how jQuery can simplify a data validation application, it doesn't begin to show the power of jQuery. For that, you need to learn more selectors, methods, and event methods, and then see how they can be used in other types of applications. You'll do that next.

The jQuery for the Email List application (email_list.js)

```
$(document).ready(function() {
    $("#join_list").click(function() {
        var emailAddress1 = $("#email_address1").val();
        var emailAddress2 = $("#email_address2").val();
        var isValid = true;

        // validate the first email address
        if (emailAddress1 == "") {
            $("#email_address1").next().text("This field is required.");
            isValid = false;
        } else {
            $("#email_address1").next().text("");
        }

        // validate the second email address
        if (emailAddress2 == "") {
            $("#email_address2").next().text("This field is required.");
            isValid = false;
        } else if (emailAddress1 != emailAddress2) {
            $("#email_address2").next().text(
                "This entry must equal first entry.");
            isValid = false;
        } else {
            $("#email_address2").next().text("");
        }

        // validate the first name entry
        if ($("#first_name").val() == "") {
            $("#first_name").next().text("This field is required.");
            isValid = false;
        }
        else {
            $("#first_name").next().text("");
        }

        // submit the form if all entries are valid
        if (isValid) {
            $("#email_form").submit();
        }
    }); // end click
    $("#email_address1").focus();
}); // end ready
```

Figure 8-8 The jQuery for the Email List application

A working subset of selectors, methods, and event methods

The next three figures present a working subset of the most useful jQuery selectors, methods, and event methods. This is a lot to take in, but once you understand them, you'll be able to write practical jQuery applications of your own.

The most useful selectors

In figure 8-4, you were introduced to the basic selectors that you can use with jQuery. Now, figure 8-9 presents the other selectors that you're most likely to use in your jQuery applications. The only selectors of significance that are missing are ones that you'll learn about in later chapters, like the animate selector that you use with animations and the form control selectors that you use with forms.

If this summary seems daunting, just read the list and realize that these selectors let you select just about any element that you need to select. Then, make sure that you understand the examples in this figure. Later, when you need to make a specific type of selection for an application, you can refer back to this summary and to figure 8-4.

To illustrate the use of these selectors, the first example shows a selector that gets the li elements that are the first children of their parent elements. If the HTML contains more than one list, this selects the first li element of each list. The second example shows how to get the even tr (row) elements in an HTML table.

The third example shows how to use the :eq selector to get a specific element within an array of elements. If, for example, there are four <p> elements that are descendants of the "faqs" element, :eq(2) will return the third <p> element because the index values start with zero.

The last example shows a selector that gets all input elements with type attributes that have "text" as the value. In other words, this selector gets all text boxes. Note, however, that you can also get the text boxes by using this selector:

```
$("input[type=text]")
```

This just shows that you can often select the elements that you want in more than one way.

A summary of the most useful jQuery selectors

Selector	Selects
`[attribute]`	All elements with the named attribute.
`[attribute=value]`	All elements with the named attribute and value.
`:contains(text)`	All elements that contain the specified text.
`:empty`	All elements with no children including text nodes.
`:eq(n)`	The element at index n within the selected set.
`:even`	All elements with an even index within the selected set.
`:first`	The first element within the set.
`:first-child`	All elements that are first children of their parent elements.
`:gt(n)`	All elements within the selected set that have an index greater than n.
`:has(selector)`	All elements that contain the element specified by the selector.
`:header`	All elements that are headers (h1, h2, ...).
`:hidden`	All elements that are hidden.
`:last`	The last element within the selected set.
`:last-child`	All elements that are the last children of their parent elements.
`:lt(n)`	All elements within the selected set that have an index less than n.
`:not(selector)`	All elements that aren't selected by the selector.
`:nth-child`	All elements that are the nth children of their parent elements.
`:odd`	All elements with an odd index within the selected set.
`:only-child`	All elements that are the only children of their parent elements.
`:parent`	All elements that are parents of other elements, including text nodes.
`:text`	All input elements with the type attribute set to "text".
`:visible`	All elements that are visible.

Examples

How to select the li elements that are the first child of their parent element

```
$("li:first-child")
```

How to select the even tr elements of a table

```
$("table > tr:even")     // numbering starts at 0, so first tag is even
```

How to select the third descendant <p> element of an element

```
$("#faqs p:eq(2)")       // numbering starts at 0
```

How to select all input elements with "text" as the type attribute

```
$(":text")
```

Description

- Figure 8-4 and the table above summarize the selectors that you are most likely to need.
- Not included are six attribute selectors that let you select attributes with attribute values that contain specific substrings.
- In chapter 9, you'll learn about a selector that's used with animation, and in chapter 10, you'll learn about other selectors that are used for form controls.

Figure 8-9 The most useful selectors

The most useful methods

The table in figure 8-10 represents a collection of methods that are taken from several jQuery categories. For instance, the prev(), next(), and find() methods are DOM traversal methods. The attr(), css(), addClass(), removeClass(), toggleClass(), and html() methods are DOM manipulation methods. The hide() and show() methods are effect methods. And the each() method is a miscellaneous method.

These are some of the most useful jQuery methods, and they will get you off to a fast start with jQuery. Then, you'll add methods to your repertoire as you read the rest of the chapters in this section.

Here again, if this table seems daunting, just read through these methods to see what's available. Then, study the examples to see how they can be used. Later, when you need a specific method for an application, you can refer back to this summary.

In the first example, the attr() method is used to get the src attribute of an element with "image" as its id. The second example uses the attr() method to add an src attribute to the selected element with the value that's stored in the variable named imageSource. If the element already has an src attribute, this method changes its value. The third example uses the css() method to change the color property of the selected elements to blue.

In the fourth example, the addClass() method is used to add a class to all of the h2 elements within the element that has "faqs" as its id. That's similar to what you've done using JavaScript.

In the fifth example, the html() method adds an h2 element to an aside element. This jQuery method works like the innerHTML property that you learned how to use in chapter 6.

In the last example, the each() method is used to perform a function for each element in an array. In this case, the array contains all of the <a> elements within the element with "image_list" as its id, and the each() method loops through these elements. As you will see, this simplifies the handling of the elements in the array.

In most cases, the jQuery methods operate on all of the selected elements. There are some exceptions, however. For example, if you use the attr() method to get the value of an attribute, it will get the value only for the first selected element. Similarly, the css() method will get the value of a property only for the first selected element.

A summary of the most useful jQuery methods

Method	Description
`next([selector])`	Get the next sibling of each selected element or the first sibling of a specified type if the parameter is coded.
`prev([selector])`	Get the previous sibling of each selected element or the previous sibling of a specified type if the parameter is coded.
`find(selector)`	Search the selected element and return descendant elements.
`attr(attributeName)`	Get the value of the specified attribute from the first selected element.
`attr(attributeName, value)`	Set the value of the specified attribute for each selected element.
`css(propertyName)`	Get the value of the specified property from the first selected element.
`css(propertyName, value)`	Set the value of the specified property for each selected element.
`addClass(className)`	Add one or more classes to the selected elements and, if necessary, create the class. If you use more than one class as the parameter, separate them with spaces.
`removeClass([className])`	Remove one or more classes. If you use more than one class as the parameter, separate them with spaces.
`toggleClass(className)`	If the class is present, remove it. Otherwise, add it.
`html(htmlString)`	Sets the HTML contents of each selected element to the specified HTML string.
`hide([duration])`	Hide the selected elements. The duration parameter can be "slow", "fast", or a number giving the time in milliseconds. By default, the duration is 400 milliseconds, "slow" is 600 milliseconds, and "fast" is 200 milliseconds.
`show([duration])`	Show the selected elements. The duration parameter is the same as for the hide() method.
`each(function)`	Run the function for each element in an array.

Get the value of the src attribute of an image

```
$("#image").attr("src");
```

Set the value of the src attribute of an image to the value of a variable

```
$("#image").attr("src", imageSource);
```

Set the value of the color property of the h2 elements to blue

```
$("h2").css("color", "blue");
```

Add a class to the h2 descendants of the "faqs" element

```
$("#faqs h2").addClass("minus");
```

Put an h2 element into an aside element

```
$("aside").html("<h2>Table of Contents</h2>");
```

Run a function for each <a> element within an "image_list" element

```
$("#image_list a").each(function() {
    // the statements of the function
});
```

Figure 8-10 The most useful methods

The most useful event methods

Figure 8-11 summarizes some of the most useful event methods, and this summary includes the ready() and click() methods that were introduced in figure 8-6. As you can see, most of these event methods provide for a single event handler that runs when the event occurs. Those event methods work like the ready() and click() methods, but with different events.

This is illustrated by the first example, which works just like the click() event method except that it handles the double-click event. Of note here is the use of the *this* keyword within the handler. It is coded as a jQuery selector, and it refers to the text box that has been double-clicked. This is similar to the way the *this* keyword works with JavaScript. Note that it isn't enclosed within quotation marks, even though it's a selector. What this function does is use the val() method to set the value of the double-clicked text box to an empty string.

In contrast, the hover() event method provides for two event handlers: one for when the mouse pointer moves into an element and another for when the mouse pointer moves out of an element. This is illustrated by the second example in this figure. Note here that the first function is the first parameter of the hover() method, and it is followed by a comma. Then, the second function is the second parameter of the hover() method. To end the parameters, the last line of code consists of a right parenthesis followed by a semicolon.

The hover() method is typical of jQuery coding, which often requires the use of one or more functions within another function. That's why you need to code the functions in a way that helps you keep the punctuation straight. It also helps to code inline comments that mark the ends of functions and methods.

In these examples, note that when a selector gets more than one element, the event method sets up the event handler for each of the selected elements. This makes it much easier to set up event handlers with jQuery than it is with JavaScript.

The last example in this figure shows how to use the preventDefault() method of the event object that's passed to the event handler to cancel the default action of the event. When you include the jQuery library, this jQuery method is executed in place of the preventDefault() method of the browser that you learned about in figure 7-1. Then, because the jQuery version of this method is cross-browser compatible, it simplifies the JavaScript code. You'll see how this method is used in some of the applications that are presented later in this chapter.

A summary of the most useful jQuery event methods

Event method	Description
ready(*handler*)	The handler runs when the DOM is ready.
click(*handler*)	The handler runs when the selected element is clicked.
dblclick(*handler*)	The handler runs when the selected element is double-clicked.
mouseenter(*handler*)	The handler runs when the mouse pointer enters the selected element.
mouseover(*handler*)	The handler runs when the mouse pointer moves over the selected element.
mouseout(*handler*)	The handler runs when the mouse pointer moves out of the selected element.
hover(*handlerIn, handlerOut*)	The first event handler runs when the mouse pointer moves into an element. The second event handler runs when the mouse pointer moves out.
event.preventDefault()	Stops the default action of an event from happening.

A handler for the double-click event of all text boxes that clears the clicked box

```
$(":text").dblclick(function () {
    $(this).val("");
});
```

A handler for the hover event of each img element within a list

```
$("#image_list img").hover(
    function() {
        alert("The mouse pointer has moved into an img element");
    },
    function() {
        alert("The mouse pointer has moved out of an img element);
    }
);  // end hover
```

A preventDefault() method that stops the default action of an event

```
$("#faqs a").click(function(evt) {  // the event object is named evt
    evt.preventDefault();           // the method is run on the event object
}); // end click
```

Description

- The table above presents the event methods that you'll use the most. Not included are: keydown, keypress, keyup, mousedown, mouseup, mouseleave, and mousemove.
- The preventDefault() method is executed on the event object that gets passed to a function when an event occurs. You can name this object whatever you want.
- Event methods named load(), unload(), and error() were deprecated in version 1.8 and removed in version 3.0. You may see them sometimes in older code examples online, but you shouldn't try to use them.

Figure 8-11 The most useful event methods

Other event methods that you should be aware of

For most applications, you'll use the event methods that have already been presented. You'll also code the function or functions for the event handlers as the parameters of the event methods.

Sometimes, though, you'll want to attach event handlers in other ways, remove an event handler, or trigger an event that starts an event handler. Then, you can use the methods that are presented in the table in figure 8-12.

The on() method lets you attach an event handler to one or more events. In the first set of examples, it's used to attach an event handler to the click event of an element with "clear" as its id. You can also do this using the click() event method as shown here. A method like this is called a *shortcut method* because it actually uses the on() method internally. In a case like this where you're attaching a single event handler, you can use either the on() method or the shortcut method.

One advantage of the on() method is that you can use it to attach an event handler to two different events. This is illustrated by the first statement in the second set of examples. Here, two events—click and mouseover—are included in the first parameter of the on() method. That means that the event handler will be triggered whenever either of these events occurs on the selected elements.

In contrast, if you want to attach the same event handler to two different events of two different elements, you can use shortcut events as shown here. Notice that because both event methods refer to the same event handler, you typically assign the event handler to a variable. Then, you can use the name of that variable as the parameter of the shortcut methods. In this example, the event handler is assigned to a variable named clearClick. Then, the click() method is used to attach this event handler to an element with "clear" as its id, and the dblclick() method is used to attach this event handler to all text boxes.

For some applications, you may want to remove an event handler when some condition occurs. To do that, you can use the off() method, as illustrated by the third example in this figure. If the event handler is assigned to a variable, you can also name the variable on the handler parameter of the off() method.

Or, if you want to remove an event handler after it runs just one time, you can use the one() event method. This is illustrated by the fourth example.

If you want to initiate an event from your jQuery code, you can use the trigger() method in either the long or short form, as illustrated by the fifth set of examples. Both of those examples trigger the click event of the element with "clear" as its id, which in turn causes the event handler for that event to be run.

That provides another way to run the same event handler for two different events. All you have to do is trigger the event for one event handler from another event handler. That's illustrated by the last example in this figure. There, the event handler for the double-click event of all text boxes triggers the click event of the element with "clear" as its id, and that starts the event handler.

Other event methods that you should be aware of

Event method	Description
`on(events, handler)`	Attach an event handler to one or more events.
`off(events, [handler])`	Remove an event handler from one or more events.
`one(event, handler)`	Attach an event handler and remove it after it runs one time.
`trigger(event)`	Trigger the event for the selected element.

How to attach an event handler to an event

With the on() method

```
$("#clear").on("click", function() {...});
```

With the shortcut method

```
$("#clear").click(function() {...});
```

How to attach an event handler to two different events

Of the same element

```
$("image_list img").on("click mouseover", function() {...});
```

Of two different elements

```
var clearClick = function() {...};
$("#clear").click(clearClick);
$(":text").dblclick(clearClick);
```

How to remove an event handler from an event

```
$("#clear").off("click");
```

How to attach and remove an event handler so it runs only once

```
$("#clear").one("click", function() {...});
```

How to trigger an event

With the trigger() method

```
$("#clear").trigger("click");
```

With the shortcut method

```
$("#clear").click();
```

How to use the shortcut method to trigger an event from an event handler

```
$(":text").dblclick(function() {
    $("#clear").click();     // triggers the click event of the clear button
});
```

Description

- When you use a *shortcut method* to attach an event handler to an event, you're actually using the on() method.
- Event methods named bind() and unbind() were deprecated in version 3.0. You may see them sometimes in older code examples online, but you shouldn't try to use them.

Figure 8-12 Other event methods that you should be aware of

Three illustrative applications

Now, to help you see how the selectors, methods, and event methods work in actual applications, this chapter presents three typical JavaScript applications.

The FAQs application in jQuery

Figure 8-13 presents a FAQs application similar to the one that you studied in chapter 6, but this time it uses jQuery. When this applications starts, the div elements after the h2 elements are hidden by the CSS, and the h2 elements are preceded by a plus sign. Then, if the user clicks on an h2 element, the div element below it is displayed and the plus sign is changed to a minus sign. And if the user clicks on the heading again, the process is reversed.

The jQuery code for this application consists of a click() event method that's within the ready() method. This click() event method sets up the event handlers for every h2 element within the section that has "faqs" as its id:

```
$("#faqs h2").click
```

Within the function for the event handler, the *this* keyword is used to refer to the current h2 element because that's the object of the click() method. Then, the toggleClass() method is used to add a class named "minus" to the h2 element if it isn't present or to remove it if it is. If you refer to the CSS that's shown here, you can see that this class determines whether a plus or minus sign is displayed before the heading.

Next, an if statement is used to check the value of the class attribute of the h2 element. To do that, it uses the attr() method. If the class isn't equal to "minus", it means that the div element that follows the h2 element shouldn't be displayed. Then, the statement within the if clause chains the next() and hide() methods to hide the div element that is a sibling to the h2 element.

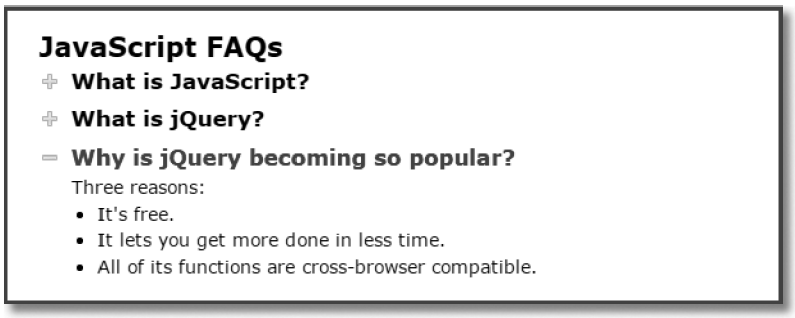
If the class attribute of the h2 element is equal to "minus", it means that the div element that follows the h2 element should be displayed. Then, the statement within the else clause uses the next() and show() methods to show the div element that is a sibling to the h2 element.

The click event ends by cancelling the default action of the <a> element that's within the h2 element. The next application will discuss how this works in more detail. For now, just know that this line of code, which is similar to the code you saw in figure 7-1, works across browsers, not just on DOM-compliant browsers.

Finally, the ready event ends by using the find() method to get the <a> elements that are descendants of the element whose id is "faqs". It then uses the first selector to get the first <a> element, and it sets the focus on that element.

This application begins to show the power of jQuery. Here, it takes just 13 lines of code to do the same thing that the JavaScript application in figure 6-6 needed 23 lines of code to do. Furthermore, these lines of code are easier to read because jQuery methods like show(), hide(), and find() are more descriptive and therefore easier to understand.

The FAQs application in a browser



The HTML

```
<main id="faqs">
    <h1>JavaScript FAQs</h1>
    <h2><a href="#">What is jQuery?</a></h2>
    <div>
        <p>jQuery is a library of the JavaScript functions that you're most
           likely to need as you develop websites.
        </p>
    </div>
    <h2><a href="#">Why is jQuery becoming so popular?</a></h2>
    <div>
        <p>Three reasons:</p>
        <ul>
            <li>It's free.</li>
            <li>It lets you get more done in less time.</li>
            <li>All of its functions are cross-browser compatible.</li>
        </ul>
    </div>
    ...
    ...
</main>
```

The critical CSS

```
h2 { background: url(images/plus.png) no-repeat left center; }
h2.minus { background: url(images/minus.png) no-repeat left center; }
div { display: none; }
```

The jQuery

```
$(document).ready(function( evt ) {
    $("#faqs h2").click(function() {
        $(this).toggleClass("minus");
        if ($(this).attr("class") != "minus") {
            $(this).next().hide();
        }
        else {
            $(this).next().show();
        }
        evt.preventDefault();           // cancel default action
    }); // end click
    $("#faqs").find("a:first").focus();   // set focus on first <a> tag
}); // end ready
```

Figure 8-13 The FAQs application in jQuery

The Image Swap application in jQuery

Figure 8-14 presents another application that shows the power of jQuery. It is the Image Swap application that you studied in chapter 7. In this application, when the user clicks on one of the thumbnail images at the top of the browser window, the caption and image below the thumbnails are changed.

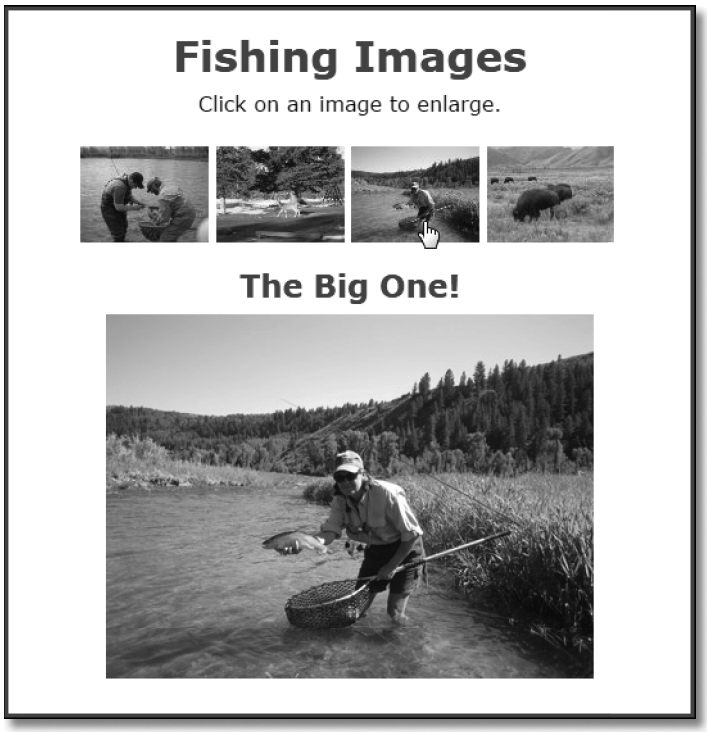
In the HTML shown here, img elements are used to display the four thumbnail images. However, these elements are coded within <a> elements so the images are clickable and they can receive the focus. In the <a> elements, the href attributes identify the images to be swapped when the links are clicked, and the title attributes provide the text for the related captions. In this case, both the <a> elements and the img elements are coded within a ul element.

After the ul element, you can see the h2 element for the caption and the img element for the main image on the page. The ids of these elements are highlighted because the jQuery will use those ids as it swaps captions and images into them.

For the motor-impaired, this HTML provides accessibility by coding the img elements for the thumbnails within <a> elements. That way, the user can access the thumbnail links by pressing the Tab key, and the user can swap the image by pressing the Enter key when a thumbnail has the focus, which starts the click event.

Of note in the CSS for this page is the rule set for the li elements. Their display properties are set to inline so the images go from left to right instead of from top to bottom.

The user interface for the Image Swap application



The HTML

```
<main>
    <h1>Fishing Images</h1>
    <p>Click on an image to enlarge.</p>
    <ul id="image_list">
        <li><a href="images/release.jpg" title="Catch and Release">
            <img src="thumbnails/release.jpg" alt=""></a></li>
        <li><a href="images/deer.jpg" title="Deer at Play">
            <img src="thumbnails/deer.jpg" alt=""></a></li>
        <li><a href="images/hero.jpg" title="The Big One!">
            <img src="thumbnails/hero.jpg" alt=""></a></li>
        <li><a href="images/bison.jpg" title="Grazing Bison">
            <img src="thumbnails/bison.jpg" alt=""></a></li>
    </ul>
    <h2 id="caption">Catch and Release</h2>
    <p><img id="main_image" src="images/release.jpg" alt=""></p>
</main>
```

The CSS for the li elements

```
li {
    display: inline;
}
```

Figure 8-14 The user interface, HTML, and CSS for the Image Swap application

Figure 8-15 presents the JavaScript and jQuery for this application. Within the ready event handler, the each() method is used to run a function for each <a> element in the unordered list. This function preloads the images that will be swapped so they are in the browser cache when the user starts using the application. If an application uses many images, this can make the application run faster.

Within the function for each <a> element, the first statement creates a new Image object. Then, the second statement uses the *this* keyword and the attr() method to set the src attribute of the Image object to the value of the href attribute in the <a> element. As soon as that's done, the image is loaded into the browser.

The each() method is followed by a click() event method that sets up the event handlers for the click events of the <a> elements that contain the thumbnail images. Note here that evt is coded as the parameter for the click() event method, which receives the event object when the event occurs. This object will be used later to cancel the default action of each link.

The first two statements in this event handler swap the image in the main portion of the browser window. The first statement uses the *this* keyword and attr() method to get the value of the href attribute of the <a> element. This value gives the location of the image to be swapped. Then, the second statement sets the src attribute of the main img element to this value. As soon as that's done, the image is swapped in the browser.

The next two statements work similarly. They swap the caption of the image in the main portion of the browser window. This time, the caption is taken from the title attribute of the <a> element.

The last statement in the click event handler uses the preventDefault() method of the evt object that is passed to the event handler to cancel the default action of clicking on the link. This is the same statement you saw in the FAQs application, but it's more important here. If you didn't cancel the default action of clicking on the <a> element in the FAQs application, the application would still work. The page would just be reloaded, which would make it less efficient. If the default action isn't cancelled when you click on a link in this application, though, the image that's referred to by the href attribute will be displayed in a new window or tab, which isn't what you want.

After the click event handler, the last statement in the ready() method shows another way to set the focus to the first <a> element. Because that element contains the first thumbnail image, that will make it easier for the users to tab to the thumbnails that they want to swap. To identify the first <a> element, the :first-child selector is used to get the first li element of the unordered list. Then, a descendant selector is used to get the <a> element within the li element.

The JavaScript

```
$(document).ready(function() {
    // preload images
    $("#image_list a").each(function() {
        var swappedImage = new Image();
        swappedImage.src = $(this).attr("href");
    });

    // set up event handlers for links
    $("#image_list a").click(function(evt) {
        // swap image
        var imageURL = $(this).attr("href");
        $("#main_image").attr("src", imageURL);

        //swap caption
        var caption = $(this).attr("title");
        $("#caption").text(caption);

        // cancel the default action of the link
        evt.preventDefault();  // jQuery cross-browser method
    }); // end click

    // move focus to first thumbnail
    $("li:first-child a").focus();
}); // end ready
```

Description

- When you attach an event handler to a link, you often need to cancel the default action of the link, which is to open the page or image that's identified by its href attribute.
- To cancel the default action of a link, you can use the jQuery preventDefault() method of the event object that is passed to the event handler. The jQuery version of this method is cross-browser compatible.
- To get the event object that's passed to a method, you need to code a parameter for the event handler function. You can use whatever name you want for this parameter, like evt or event. Then, you must use that name whenever you refer to the event object.
- When you use jQuery to work with images that aren't included in the HTML, preloading the images can improve the performance of the application because the user doesn't have to wait for a new image to be loaded into the browser.
- To preload an image, you create a new Image object and assign the URL for the image to the Image object's src attribute. As soon as that's done, the image is loaded into the browser.

Figure 8-15 The JavaScript for the Image Swap application

The Image Rollover application in jQuery

Figure 8-16 presents another application that shows the power of jQuery. It's the Rollover application from the exercises for chapter 7. In this application, when the user hovers the mouse pointer over one of the starting images in the Rollover application, it's replaced by another image. And when the user moves the mouse pointer out of the image, the original image is again displayed.

In the HTML shown here, img elements are coded within the li elements of an unordered list. In these img elements, the src attribute identifies the image that is displayed when the application is loaded into the browser, and the id attribute identifies the image that should be displayed when the mouse hovers over the img element.

In the JavaScript shown here, an each() method is used to perform a function for each occurrence of an img element within the ul element that has "image_rollovers" as its id:

```
$("#image_rollovers img").each
```

The function for the each() method starts by using the *this* keyword and the attr() method to get the values of the src and id attributes of the current image and store them in variables named oldURL and newURL. Remember that these attributes hold the values that locate the starting image and its rollover image. Then, these variables can be used to preload the rollover image and to set up the hover event handlers for each image.

Note that the each() method applies the function to all img elements in the element whose id is "image_rollovers". That means that you can add more images to this element, and as long as the img elements have an src attribute with the original image and an id attribute with the new image, they will work as rollovers without adding any more jQuery or JavaScript code.

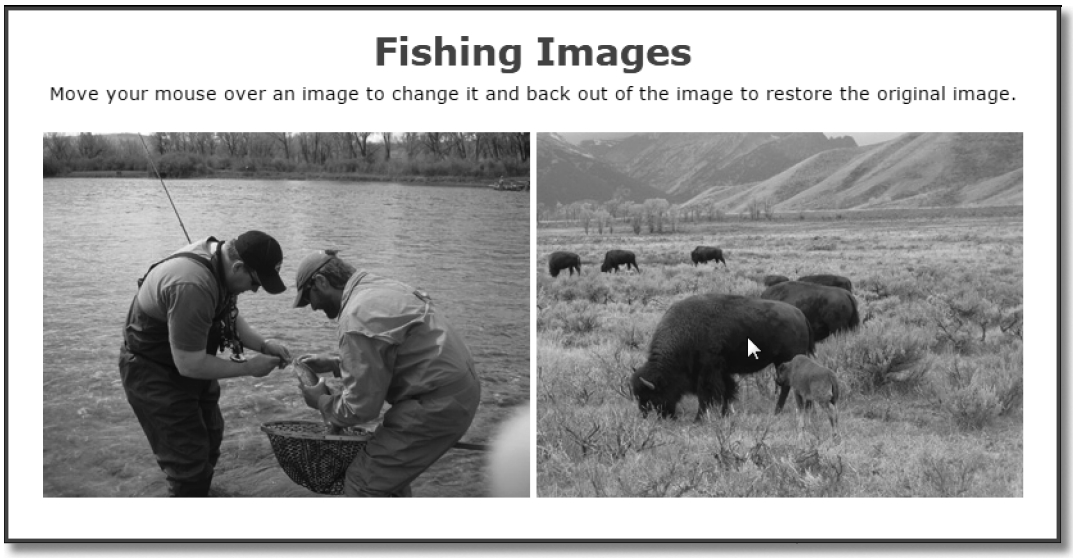
The next two statements preload the rollover image by creating a new Image object and assigning the value of the newURL variable to its src attribute. As soon as the src attribute is set, the browser loads the image. Although preloading isn't essential, it can improve the user experience because the users won't have to wait for a rollover image to be loaded.

Next, the hover() event method is used to set up the event handlers for the current image (this). Remember, this event method has two event handlers as its parameters. The first event handler is run when the mouse pointer moves into the element, and the second is run when the mouse pointer moves out of the element.

In the function for the first event handler, you can see that the *this* keyword and the attr() method are used to set the src attribute of the image to the value of the newURL variable. That causes the rollover image to be displayed. The function for the second event handler reverses this process by restoring the src attribute of the image to the value of the oldURL variable.

Of course, there's more than one way to code an application like this. For instance, you could implement this application by using the mouseover and mouseout events, similar to how the chapter 7 exercise worked. Then, you'd code the mouseover event like the first function of the hover() event method, and the mouseout event like the second function of the hover() event method.

Two images with the second image rolled over



The HTML

```
<main>
    <h1>Fishing Images</h1>
    <p>Move your mouse over an image to change it and back out of the
        image to restore the original image.</p>
    <ul id="image_rollovers">
        <li><img src="images/release.jpg" alt="" id="images/deer.jpg"></li>
        <li><img src="images/hero.jpg" alt="" id="images/bison.jpg"></li>
    </ul>
</main>
```

The JavaScript

```
$(document).ready(function() {
    $("#image_rollovers img").each(function() {
        var oldURL = $(this).attr("src");       // gets the src attribute
        var newURL = $(this).attr("id");        // gets the id attribute

        // preload rollover image
        var rolloverImage = new Image();
        rolloverImage.src = newURL;

        // set up event handlers
        $(this).hover(
            function() {
                $(this).attr("src", newURL);    // sets the src attribute
            },
            function() {
                $(this).attr("src", oldURL);    // sets the src attribute
            }
        );  // end hover
    }); // end each
}); // end ready
```

Figure 8-16 The Image Rollover application in jQuery

Perspective

To get you off to a fast start, this chapter has presented a working subset of the most useful jQuery selectors, methods, and event methods. That should give you some idea of what you have to work with when you use jQuery. And you'll add to those selectors and methods as you read other chapters in this book.

The trick of course is being able to apply the right selectors, methods, and event methods to the application that you're trying to develop. To get good at that, it helps to review many different types of jQuery applications. That's why this chapter has presented four applications, and that's why you'll see many more before you complete this book.

Terms

jQuery
CDN (Content Delivery Network)
Subresource Integrity (SRI) checking
selector
method
getter method
setter method
object chaining
event method
shortcut method

Summary

- *jQuery* is a JavaScript library that provides methods that make JavaScript programming easier. These methods have been tested for cross-browser compatibility.
- To use jQuery, you code a script element in the head section that includes the file for the jQuery core library. This file can be downloaded and stored on your computer or server, or you can access it through a *Content Delivery Network* (*CDN*).
- If you use the jQuery CDN, the jQuery website recommends that you use *Subresource Integrity (SRI) checking* to help ensure that resources from a third-party server aren't tampered with.
- When you code statements that use jQuery, you use *selectors* that are like those for CSS. You also use a dot syntax that consists of the selector for one or more elements, the dot, and the name of the *method* that should be executed.
- You can use *object chaining* to call a jQuery method on the object that's returned by another method.
- To set up event handlers in jQuery, you use *event methods*. Most of these methods have one parameter that is the event handler that will be run when the event occurs. But some event methods like the hover() method take two event handler parameters.

Exercise 8-1 Add a Clear button to the Email List application

In this exercise, you'll add a Clear button to the Email List application of figures 8-7 and 8-8. That will give you practice creating and attaching another event handler.

1. Use your text editor or IDE to open the index.html and email_list.js files in this folder:
   javascript_jquery\exercises\ch08\email_list
2. Run the application to refresh your memory about how it works. Note that a Clear button has been added below the Join our List button, but the button doesn't work.
3. Add an event handler for the click event of the Clear button that clears all of the text boxes by setting them to an empty string (""). This can be done in one statement. To select just the text boxes, you use a selector like the one in the last example in figure 8-9. To set the values to empty strings, you use the val() method like the second example in figure 8-5. Now, test this change.
4. This event handler should also put the asterisks back in the span elements that are displayed to the right of the text boxes to show that entries are required. That requires just one statement that uses the next() and the text() methods.
5. Add one more statement to this event handler that moves the focus to the first text box. Then, test this change.
6. Add another event handler to this application for the double-click event of any text box. This event handler should do the same thing that the event handler for the click event of the Clear button does. The easiest way to do that is to trigger the click event of the Clear button from the handler for the double-click event, as in the last example in figure 8-12. Test this change.
7. Comment out the line of code that you just used to trigger the click event of the Clear button. Then, add a statement to the double-click event handler that only clears the text from the text box that the user double-clicks in. To do that, you'll have to use the *this* keyword, as in the first example in figure 8-11. Test the application one more time to make sure this works.

Exercise 8-2 Use different event methods for the Image Rollover application

This exercise asks you to modify the Image Rollover application of figure 8-16 so it uses different events for the event handlers.

1. Use your text editor or IDE to open the HTML and JavaScript files that are in this folder:
   javascript_jquery\exercises\ch08\rollover
2. Run the application to refresh your memory about how it works.

3. Comment out the hover() method in the JavaScript, and rewrite the code so it uses the mouseover() and mouseout() event methods to implement this application.
4. Add the two images that are displayed when the two existing images are rolled over to the ul element whose id is image_rollovers. Make sure that the src attributes of the img elements contain the original image for the rollover, and that the id attributes contain the image that should display when the mouse is over the image.

Exercise 8-3 Develop a Book List application

In this exercise, you'll start with the HTML and CSS for the user interface that follows. Then, you'll develop the jQuery code that makes it work.

Development guidelines

1. You'll find the HTML, CSS, and image files for this application in this folder:

   javascript_jquery\exercises\ch08\book_list

   You'll also find an empty JavaScript file named book_list.js. You can add your code to this file.
2. This application works like the FAQs application you saw in this chapter, except that a list of book links is displayed below each heading. If the user clicks on one of these links, an image for the book is displayed to the right of the list. In addition, any time the user clicks on a heading with a plus or minus sign before it, the image should no longer be displayed.
3. The HTML for the links of this application is like the HTML for the Image Swap application. However, the links for this application don't require the title attribute since no caption is displayed for the image.
4. The images that are referred to by the href attributes of the links in this application should be preloaded. To do that, you can loop through all the links in the main element. Also, be sure to cancel the default actions of the links.
5. Feel free to copy and paste code from any of the applications that are available to you. That's the most efficient way to build a new application.