PART II

LEARNING THE JQUERY FUNDAMENTALS

3 USING SELECTORS, FILTERS, AND CSS: JQUERY AT ITS CORE

SELECTORS ARE THE CORE building blocks of jQuery: Everything you do with the DOM (Document Object Model) in jQuery incorporates the use of a selector because you need to choose which elements in the DOM you are selecting and manipulating. jQuery uses common CSS and XPATH selectors that most Web designers and developers are already familiar with, as well as a few custom jQuery selectors. This makes jQuery flexible and easy to learn. Understanding how selectors work is a key component in utilizing jQuery to its fullest capability.

Filters give you more flexibility in selecting elements based on a characteristics in the DOM, where CSS selectors can't help. Filters are often used in conjunction with selectors to give you great depth of control when selecting specific elements based on, say, their position in a group of elements, or their visibility, or a characteristic of a form input such as checked or disabled. jQuery also offers a nice range of methods for adding and removing CSS classes and also applying styling directly to a DOM element.

In this chapter, I review how to use selectors, filters, and CSS with jQuery through various real-world tutorials, which gives you a great understanding of these fundamental jQuery methods.

WORKING WITH DOM ELEMENTS USING JQUERY SELECTORS

Selectors, an essential feature of the jQuery library, are powered by the jQuery Sizzle selector engine. Sizzle can be used with other languages, but its real power is best used with all the other jQuery methods. The syntax is easy to understand for Web designers who have a solid understanding of CSS and HTML. The jQuery Sizzle selector engine is JavaScript code written to handle selectors in jQuery. The selectors are common CSS and XPATH selectors with the addition of a few custom selectors.

The jQuery selector is a string expression that classifies a single or set of DOM elements to be known as the matched set and is ready to be worked with in jQuery. The selector is always declared directly after the jQuery alias ($). After DOM elements are selected and methods have been applied, the matched set becomes a jQuery object. jQuery objects allow you to add many different types of methods including events, effects, traversal, manipulation, etc. A jQuery object is something that you may or may not encounter during your jQuery coding, but it's always there and it's important to be aware of.

The following code example shows how a selector is laid out in jQuery:

```
$(selector).method().
```

With a little practice, working with selectors in jQuery becomes second nature because many selectors are those you have worked with before using CSS. A selector is a way for you to navigate the Document Object Model (DOM), and, in its most basic form, allows you to select an element and the syntax is identical to CSS selector syntax, whether it's a ID, class, tag, or an attribute. Selectors are an essential component of any jQuery statement that you create.

When you use a selector, the statement you create automatically loops through all of the nodes in the DOM looking for the elements you have specified in your selector. The result of this loop is also known as the matched set. Web designers who are proficient with CSS and have a solid understanding of the DOM can pick up selectors quickly.

The JavaScript parts that make up a selector are

- The jQuery alias (`jQuery` or `$`).
- The DOM elements, which you are selecting, wrapped in quotes within the two parentheses.
- Anything after the selector is the jQuery method, which you are applying. The jQuery method can do anything from adding CSS to animating elements on the page. The functions, also known as methods, perform the specified action and accept arguments in the parentheses.

Table 3.1 outlines the anatomy of a jQuery selector statement:

Table 3.1 The Anatomy of a jQuery Statement

The jQuery Alias	The Selector	The jQuery Method or Action
$ or jQuery	('div')	.css('border','1px solid #333');

SELECTING PAGE ELEMENTS BY USING CSS SELECTORS

JavaScript has native functions that can select elements by ID and tag. The downside of these functions is that you have to use a different function for each of the three types of elements. Also, this creates repetition and code bloat, which can become a nightmare to manage. When you use selectors in jQuery, one selector can handle multiple types of elements. This makes writing clean and manageable code much easier.

All examples in this book show the browser output as well as the Firebug output so you can see what is being added or changed to the DOM. (See Chapter 2 for more about installing and using Firebug.) When you view the page source, you don't see the rendered source. The rendered source is the actual code that is created after the browser loads and the JavaScript has fired, which, as you are manipulating the DOM, can change quite noticeably. Firebug shows the rendered source, and can even show you the source as it changes, so it's the perfect way to test your JavaScript and jQuery.

The following list outlines the most common jQuery CSS selectors, which I explain in more depth with code and browser throughout in this chapter:

- `$('*')`
- `$('p')`
- `$('.class')`
- `$('#id')`
- `$('.parent ul li')`

In this chapter, I apply CSS to the elements using the jQuery `.css()` method. The CSS method in jQuery works by allowing you to pass any CSS properties to the CSS method that are then applied to the element matched by the selector. The CSS is added to the element, as an inline style after the DOM is loaded and ready.

In each example, you are not limited to the CSS I am applying. I'm just showing individual examples of problems you may encounter yourself and how to quickly solve them using selectors.

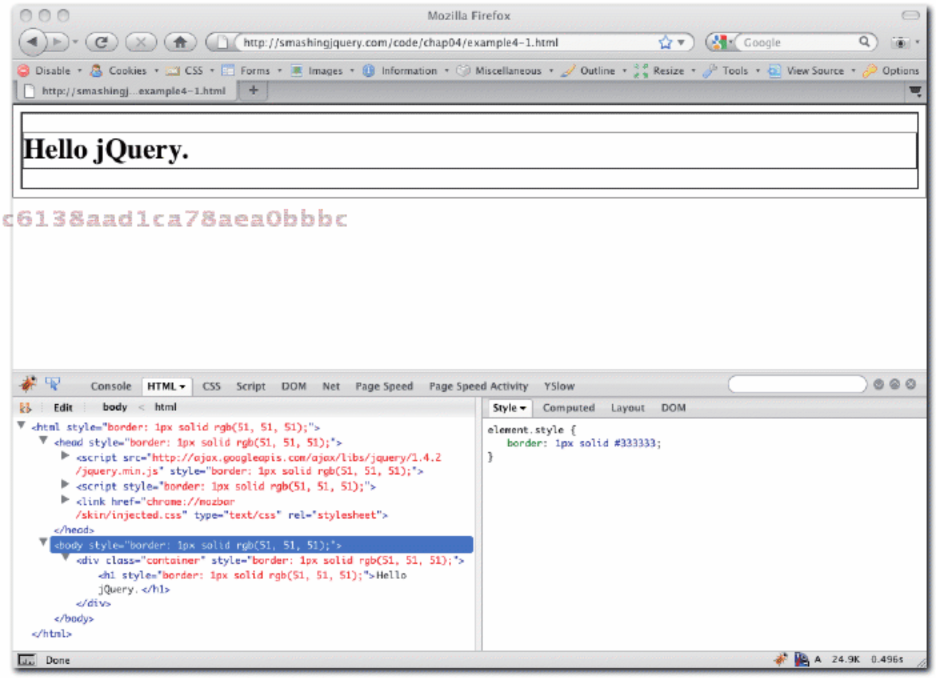
Selecting Elements by Using the Wildcard (*) Selector

If you would like to select all the elements in your DOM or within other elements, use the wildcard (*) selector. The wildcard is wrapped in quotes between the parentheses directly after the alias.

In the following HTML example, I select all the elements in the page with the wildcard selector and add a CSS border to each element. To add a CSS border, I use the method `.css('border','1px solid #333');`.

Figure 3-1 shows the output of the following script example with the Firebug tab open. This is where you can observe how jQuery added the borders inline to each element on the page.

```
<!doctype html>
<html>
  <head>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'></
 script>
  <script>
  $(document).ready(function() {
    $('*').css('border','1px solid #333');
  });
  </script>
  <body>
    <div class="container">
    <h1>Hello jQuery.</h1>
    </div>
  </body>
</html>
```



Hello jQuery.

```
<html style="border: 1px solid rgb(51, 51, 51);">
  <head style="border: 1px solid rgb(51, 51, 51);">
    <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2
    /jquery.min.js" style="border: 1px solid rgb(51, 51, 51);">
    <script style="border: 1px solid rgb(51, 51, 51);">
    <link href="chrome://mozbar
    /skin/injected.css" type="text/css" rel="stylesheet">
  </head>
  <body style="border: 1px solid rgb(51, 51, 51);">
    <div class="container" style="border: 1px solid rgb(51, 51, 51);">
      <h1 style="border: 1px solid rgb(51, 51, 51);">Hello
      jQuery.</h1>
    </div>
  </body>
</html>
```

```
element.style {
    border: 1px solid #333333;
}
```

Figure 3-1: The browser output of adding a border to every element on the page using the wildcard selector

Selecting Elements by Using the HTML Tag

After you understand how the wildcard selector works, you can see that the other CSS selectors work the same way. You can select any element within the DOM using the element selector — you need to pass a tag name to the selector, which is present in the page. This selector uses the native JavaScript method `getElementsByTagName()`.

The native JavaScript function `getElementsByTagName()` retrieves all elements by their tag name. The code is set up like this:

```
document.getElementsByTagName('h1');
```

In the following HTML example, I need to set the `font-family` property of my `h1` tags. I can do so by changing the CSS, but I don't want the `h1` tags on other pages to be affected. Instead, I can use the tag selector together with the `css()` method to change the `font-family` property for `H1` tags on this page only. I use the CSS method to apply the `font-family` property to just the `H1` tag that I select.

Figure 3-2 shows the output of the following code in the browser with Firebug open. This example demonstrates how the DOM is altered by the `font-family` property being added to the `h1` tag after the DOM has loaded.

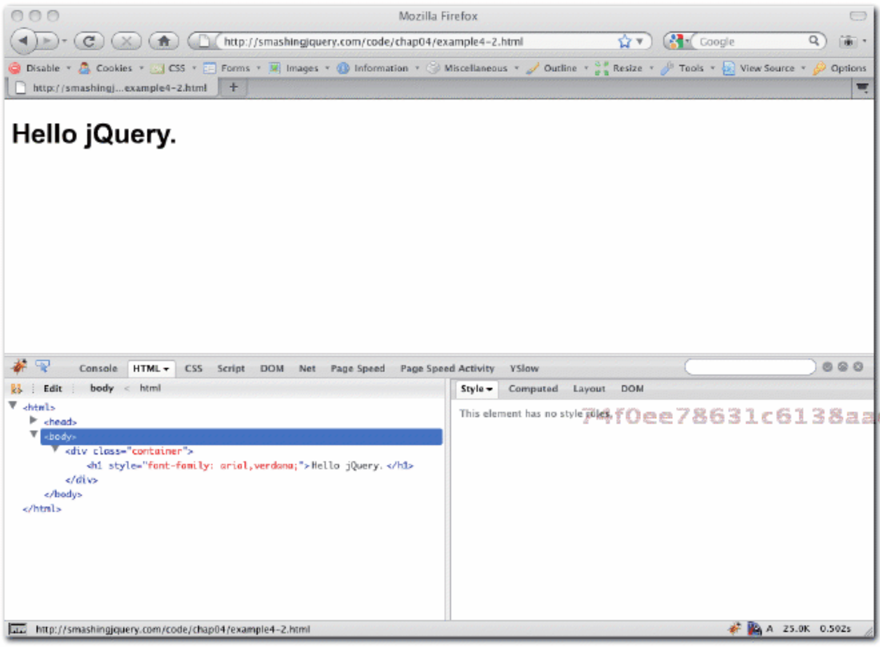
```
<!doctype html>
<html>
  <head>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'></
 script>
  <script>
  $(document).ready(function() {
    $('h1').css('font-family','arial,verdana');
  });
  </script>

  <body>
    <div class='container'>
    <h1>Hello jQuery.</h1>
    </div>
  </body>
</html>
```

Selecting Elements by Using the ID Selector

You can select any ID within the page using the `ID ('#')` selector. The ID selector uses the native JavaScript method `getElementById()`. To select a class using the native JavaScript function `getElementById()`, the code is set up in the following manner:

```
document.getElementById('sidebar');
```


Mozilla Firefox
http://smashingjquery.com/code/chap04/example4-2.html
Hello jQuery.
Console HTML CSS Script DOM Net Page Speed Page Speed Activity YSlow
Edit body < html
Style Computed Layout DOM
This element has no style rules.

```
<html>
  <head>
  <body>
    <div class="container">
      <h1 style="font-family: arial,verdana;">Hello jQuery. </h1>
    </div>
  </body>
</html>
```

Figure 3-2: The element selector in action

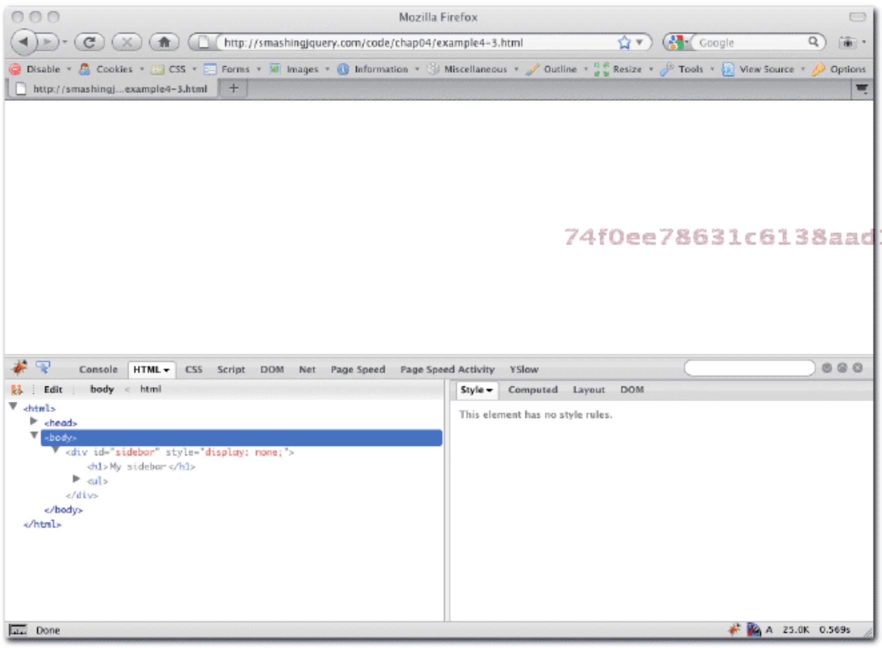
The ID selector always includes the # (hash) symbol when referencing the ID in the selector. Without that symbol, the selector won't work correctly. The ID selector returns only the one ID that it matches. Remember that with CSS no two elements on a page should share the same ID.

In the following HTML example, I want to hide the #sidebar div using CSS. I use a jQuery selector to select the #sidebar id from the document. The ID is wrapped in quotes between the parentheses, directly after the alias. In order to hide the div, I use the CSS method and apply the following: 'display','none'.

Figure 3-3 shows the browser output of using the ID selector to hide a div with the ID "#sidebar":

```
<!doctype html>
<html>
  <head>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'></
 script>
  <script>
  $(document).ready(function() {
    $('#sidebar').css('display','none');
  });
  </script>
  <body>
  <div id='sidebar'>
  <h1>My sidebar</h1>
  <ul>
```

```
    <li><a href='/nav'>Navigation</a></li>
  </ul>
  </div>
  </body>
</html>
```



```
<html>
  <head>
  <body>
    <div id="sidebar" style="display: none;">
      <h1>My sidebar</h1>
      <ul>
    </div>
  </body>
</html>
```

This element has no style rules.

Figure 3-3: The browser output after selecting the #sidebar ID

Selecting Elements by Class

Similar to selecting by ID, you can also select elements in your page by class (`.class`). This selector uses the native JavaScript method `getElementsByClassName()`. The class selector selects all elements of the given class in the DOM. If you want to select a class using the native JavaScript method `getElementsByClassName()`, the code is set up as follows:

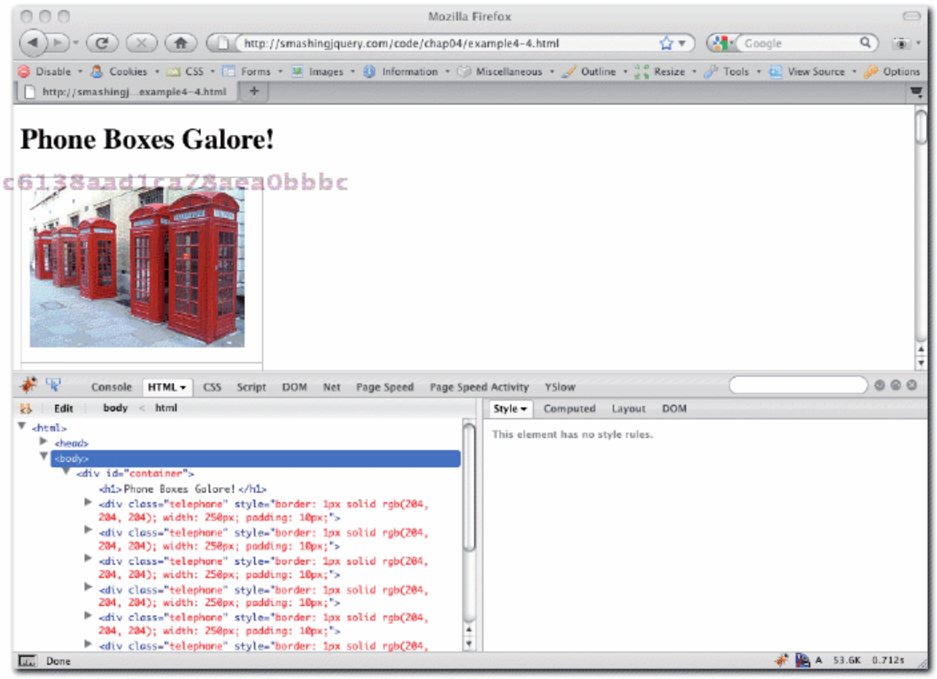
```
document.getElementsByClassName('product-image');
```

By using jQuery, I can achieve the same result as the `getElementsByClassName()` method, but with much less code.

In the following HTML example, I want to add a gray 1 pixel border, include 5 pixels of padding, and set a width of 150 pixels. The CSS method allows multiple CSS properties to be passed in using an object literal (a comma-separated list composed of name and value pairs that helps keep your code organized), which helps to keep this statement clean and simple.

I use a class selector to select all instances of the (`.telephone`) class on the page. I pass three sets of CSS properties to the CSS method, therefore I need to also enclose those properties in brackets `{}`. Figure 3-4 shows the output of the browser after the class selector has been called.

```
<!doctype html>
<html>
  <head>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></
 script>
  <script>
  $(document).ready(function() {
    $('.telephone').css({'padding':'5px','border':'1px solid
 #ccc','width':'150px'});
  });
  </script>
  <body>
    <div id='container'>
    <h1>Hello jQuery</h1>
    <div class='telephone'><img src="images/product.jpg"></div>
    <div class='telephone'><img src="images/product.jpg"></div>
    <div class='telephone'><img src="images/product.jpg"></div>
    <div class='telephone'><img src="images/product.jpg"></div>
    <div class='telephone'><img src="images/product.jpg"></div>
    <div class='telephone'><img src="images/product.jpg"></div>
    <div class='telephone'><img src="images/product.jpg"></div>
    <div class='telephone'><img src="images/product.jpg"></div>
    </div>
  </body>
</html>
```



Phone Boxes Galore!

```
<html>
  <head>
  <body>
    <div id="container">
      <h1>Phone Boxes Galore!</h1>
      <div class="telephone" style="border: 1px solid rgb(204,
      204, 204); width: 250px; padding: 10px;">
      <div class="telephone" style="border: 1px solid rgb(204,
      204, 204); width: 250px; padding: 10px;">
      <div class="telephone" style="border: 1px solid rgb(204,
      204, 204); width: 250px; padding: 10px;">
      <div class="telephone" style="border: 1px solid rgb(204,
      204, 204); width: 250px; padding: 10px;">
      <div class="telephone" style="border: 1px solid rgb(204,
      204, 204); width: 250px; padding: 10px;">
      <div class="telephone" style="border: 1px solid rgb(204,
```

This element has no style rules.

Figure 3-4: The browser output after selecting the eight elements with the class name telephone

Selecting One or Many Elements with Multiple Classes

In some cases, you may have applied multiple classes to the same element and you may want to select only elements with those classes applied. The class selector accepts multiple classes.

In the following example, I have six elements with multiple classes. I am hiding the two elements using CSS that have the `"book"` and `"inactive"` classes applied to them by using the jQuery class selector and CSS method. Figure 3-5 shows the output of the browser after the multiple elements in the selector have been called.

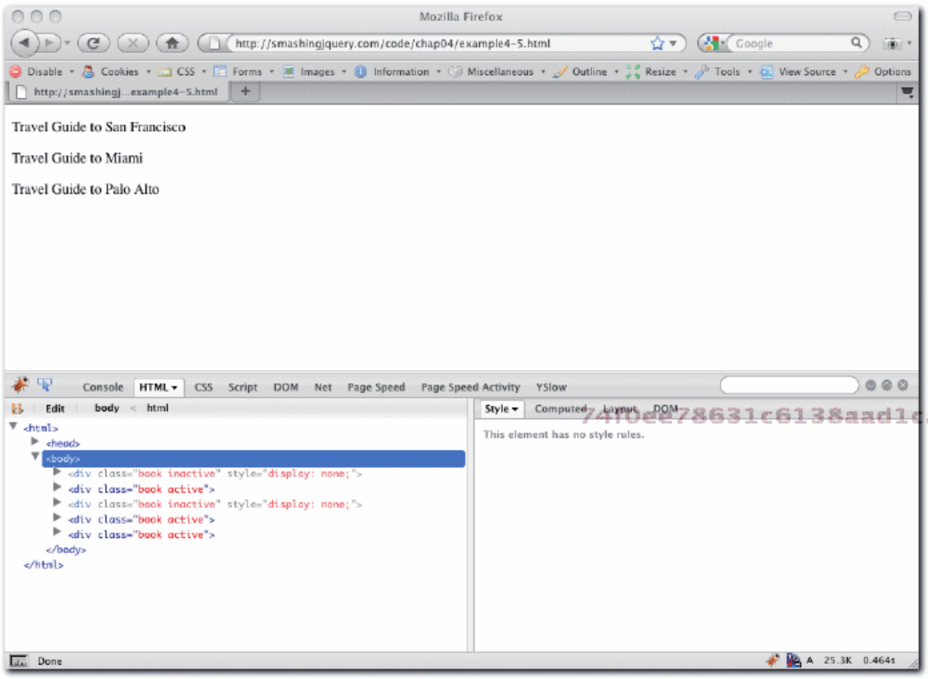
```
<!doctype html>
<html>
  <head>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'></
 script>
  <script>
  $(document).ready(function() {
    $('.book.inactive').css('display','none');
  });
  </script>

  <body>
    <div class='book inactive'>
        <p>Travel Guide to NYC</p>
    </div>
    <div class='book active'>
        <p>Travel Guide to San Francisco</p>
    </div>
    <div class='book inactive'>
        <p>Travel Guide to Seattle</p>
    </div>
    <div class='book active'>
        <p>Travel Guide to Miami</p>
    </div>
    <div class='book active'>
        <p>Travel Guide to Palo Alto</p>
    </div>

  </body>
</html>
```

Selecting Page Elements by Using Parent-Child Selectors

Parent-child selectors are a useful way to select elements within your page, when `tag`, `CSS`, and `ID` elements cannot be used. The `parent-child` CSS property is available in CSS to all popular browsers except IE6. Are you surprised? I'm not, but the beauty of using this CSS selector with jQuery is that IE6 is supported. The parent-child selector can be very useful when working with nested elements such as navigation menus.



Travel Guide to San Francisco

Travel Guide to Miami

Travel Guide to Palo Alto

```
<html>
  <head>
  <body>
    <div class="book inactive" style="display: none;">
    <div class="book active">
    <div class="book inactive" style="display: none;">
    <div class="book active">
    <div class="book active">
  </body>
</html>
```

This element has no style rules.

Figure 3-5: The browser output after selecting the two elements with the class names book and inactive

The parent-child selector allow you to select direct descendents of a given parent element using the > operator. Consider the following jQuery example. This statement selects all of the paragraph (p) elements found within the body as long as they are a child (that is, nested inside of the body tag):

```
$('body > p')
```

But if you had a scenario where you were trying to select a specific paragraph element inside of a specific div element, you would have to be more specific by using their class or ID names in order to select the correct descendent element.

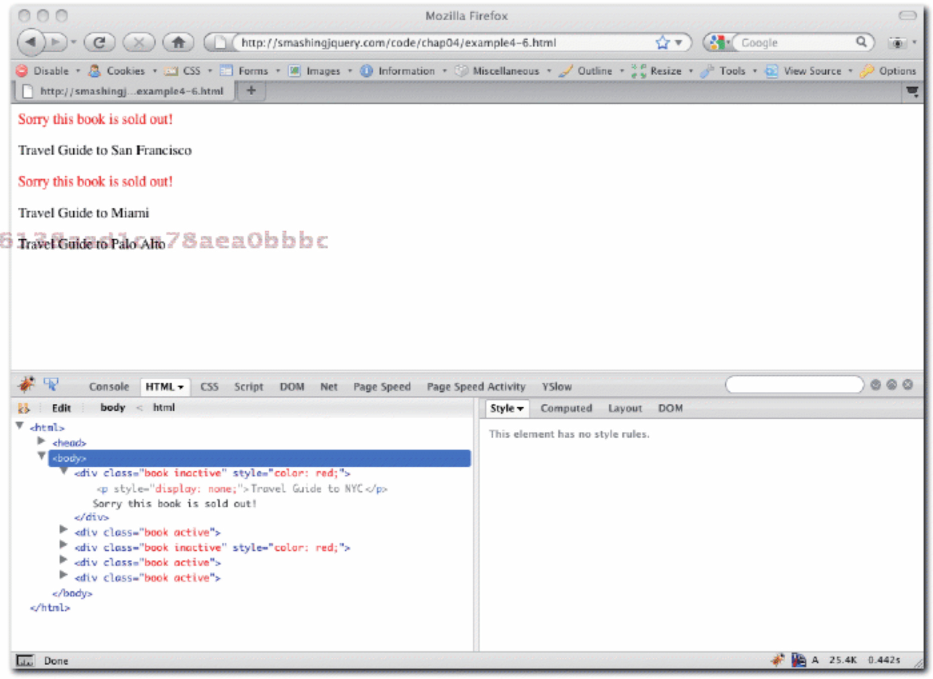
In the following HTML example, I need to select the p tags within the elements that have the class (.inactive) and append "Sorry, this book is sold out" to the end of the p tag in the color red. I use chaining to add multiple methods to one statement. Figure 3-6 shows the output of the browser after the selector statement has been applied.

```
<!doctype html>
<html>
  <head>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'></
 script>
  <script>
  $(document).ready(function() {
    $('.book.inactive > p').css('display','none');
```

```
        $('.book.inactive').append('Sorry this book is sold out!').css('color','red');
      });
      </script>

      <body>
        <div class='book inactive'>
            <p>Travel Guide to NYC</p>
        </div>
        <div class='book active'>
            <p>Travel Guide to San Francisco</p>
        </div>
        <div class='book inactive'>
            <p>Travel Guide to Seattle</p>
        </div>
        <div class='book active'>
            <p>Travel Guide to Miami</p>
        </div>
        <div class='book active'>
            <p>Travel Guide to Palo Alto</p>
        </div>

      </body>
    </html>
```



Sorry this book is sold out!

Travel Guide to San Francisco

Sorry this book is sold out!

Travel Guide to Miami

Travel Guide to Palo Alto

```
<html>
  <head>
  <body>
    <div class="book inactive" style="color: red;">
      <p style="display: none;">Travel Guide to NYC</p>
      Sorry this book is sold out!
    </div>
    <div class="book active">
    <div class="book inactive" style="color: red;">
    <div class="book active">
    <div class="book active">
  </body>
</html>
```

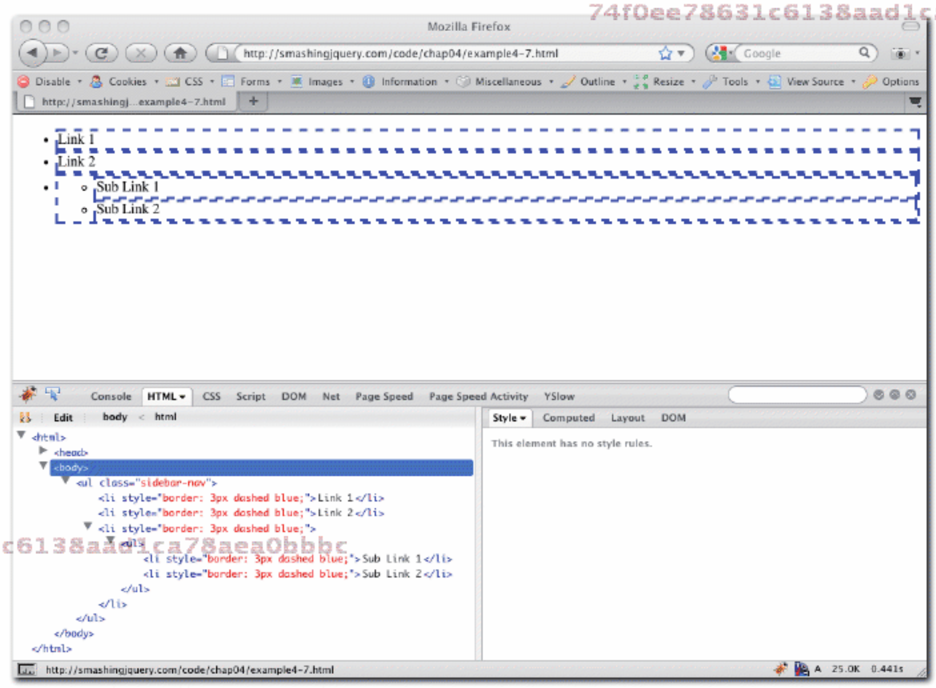
This element has no style rules.

Figure 3-6: The output of the browser after the selector statement has been applied

Selecting Page Elements by Using Descendent Selectors

Parent-child selectors only work if the child is directly related to the parent, like `<li>` tags that are between two `<ul>` tags. If you run into a situation where you are trying to select an element that is two or three levels down from the parent, you need to use descendent selectors. The difference between the parent-child and descendent selector is the usage of the > operator — if this operator is not included, the selector matches not only child but also descendent elements.

In the following HTML example, I select the `<li>` tag that is nested within the `<ul class="sidebar-nav">` tags. After I have a matched set, I apply a CSS border to it. Figure 3-7 shows the output of the browser after the class selector has been called.



```
Link 1
Link 2
  Sub Link 1
  Sub Link 2
```

```
<html>
  <head>
  <body>
    <ul class="sidebar-nav">
      <li style="border: 3px dashed blue;">Link 1</li>
      <li style="border: 3px dashed blue;">Link 2</li>
      <li style="border: 3px dashed blue;">
        <ul>
          <li style="border: 3px dashed blue;">Sub Link 1</li>
          <li style="border: 3px dashed blue;">Sub Link 2</li>
        </ul>
      </li>
    </ul>
  </body>
</html>
```

This element has no style rules.

http://smashingjquery.com/code/chap04/example4-7.html

Figure 3-7: The browser output after selecting descendent elements

```
<!doctype html>
<html>
  <head>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'></
 script>
  <script>
  $(document).ready(function() {
    $('ul li').css('border','3px dashed blue');
  });
  </script>

  <body>
```