Learn JavaScript—
the quick and easy way!

- Easy visual approach uses pictures to guide you through bringing your websites to life with JavaScript and show you what to do step by step.
- Concise steps and explanations let you get up and running in no time.
- Essential reference guide keeps you coming back again and again.
- Whether you’re a JavaScript newbie or an experienced veteran, this book will teach you all you need to know—from the basics of making your website interactive to adding advanced features with jQuery—and much more!
- Visit the companion website at www.javascriptworld.com to download sample scripts and more.
Dedication

To the memory of Bill Horwitz and Dorothy Negrino, because they loved learning.

Special Note

Way back in 1997, when we were writing Chapter 1 of our first edition of this book, we were searching for a way to make the concept of JavaScript objects clear, and found inspiration in the then-newest member of our family, our cat Pixel. Over the years since then, countless readers have told us how our “cat object” helped them to understand JavaScript better. Pixel became the mascot for many of our books. In the Fall of 2013, after a long and happy life, we lost him to old age. We miss him very much.

Pixel, on his last day with us.
Special Thanks to:

Big thanks to our editor Nancy Peterson; her expert touch, serenity, and compassion made this edition a pleasant one to create. Extra-special thanks for her above-the-call understanding when we were faced with a personal crisis.

Thanks also go to our other editor, Scholle McFarland, who stepped in and kept the project on an even keel when Nancy was overscheduled.

Thanks to Scout Festa for her skillful copy-editing. Our heartfelt thanks to Danielle Foster, the book’s production editor, who laid out the book and pulled off the job with grace and aplomb, and to the indexer, Emily Glossbrenner, who should be thanked for doing a thankless job.

As always, we’re grateful to Peachpit’s Nancy Ruenzel and Nancy Davis for their support.

We’d like to express our special thanks to all of the high school, college, and university instructors who chose to use the previous editions of this book as a textbook for their classes.
# Contents at a Glance

Introduction ........................................... xiii

**Chapter 1**  
Getting Acquainted with JavaScript .................. 1

**Chapter 2**  
Start Me Up! ........................................... 21

**Chapter 3**  
Your First Web App................................... 49

**Chapter 4**  
Working with Images .................................. 81

**Chapter 5**  
Windows and Frames ................................... 115

**Chapter 6**  
Form Handling ......................................... 133

**Chapter 7**  
Forms and Regular Expressions ....................... 171

**Chapter 8**  
Handling Events ....................................... 193

**Chapter 9**  
JavaScript and Cookies ................................ 219

**Chapter 10**  
Objects and the DOM .................................... 241

**Chapter 11**  
Making Your Pages Dynamic ............................ 261

**Chapter 12**  
Applied JavaScript ..................................... 285

**Chapter 13**  
Introducing Ajax .......................................... 325

**Chapter 14**  
Toolkits, Frameworks, and Libraries ................. 365

**Chapter 15**  
Designing with jQuery .................................. 385

**Chapter 16**  
Building on jQuery ...................................... 411

**Chapter 17**  
Scripting Mobile Devices ............................... 425

**Chapter 18**  
Bookmarklets ............................................. 441

**Appendix A**  
JavaScript Genealogy and Reference .................. 469

**Appendix B**  
JavaScript Reserved Words ............................. 491

**Appendix C**  
Cascading Style Sheets Reference .................... 495

**Appendix D**  
Where to Learn More .................................... 507

Index ....................................................... 515
# Table of Contents

## Introduction

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1</td>
<td>Getting Acquainted with JavaScript</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>What JavaScript Is</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>JavaScript Isn’t Java</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Where JavaScript Came From</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>What JavaScript Can Do</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>What JavaScript Can’t Do</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>JavaScript and More</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>The Snap-Together Language</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Handling Events</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>Values and Variables</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Writing JavaScript-Friendly HTML</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>What Tools to Use?</td>
<td>20</td>
</tr>
</tbody>
</table>

## Chapter 2 | Start Me Up! | 21

|         | Where to Put Your Scripts | 23 |
|         | About Functions | 25 |
|         | Using External Scripts | 26 |
|         | Putting Comments in Scripts | 29 |
|         | Alerting the User | 31 |
|         | Confirming a User’s Choice | 33 |
|         | Prompting the User | 35 |
|         | Redirecting the User with a Link | 37 |
|         | Using JavaScript to Enhance Links | 39 |
|         | Using Multi-Level Conditionals | 43 |
|         | Handling Errors | 46 |

## Chapter 3 | Your First Web App | 49

<p>|         | Around and Around with Loops | 50 |
|         | Passing a Value to a Function | 55 |
|         | Detecting Objects | 57 |
|         | Working with Arrays | 59 |
|         | Working with Functions That Return Values | 61 |
|         | Updating Arrays | 62 |</p>
<table>
<thead>
<tr>
<th>Chapter 4</th>
<th>Working with Images</th>
<th>81</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating Rollovers</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>Creating More Effective Rollovers</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>Building Three-State Rollovers</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>Triggering Rollovers from a Link</td>
<td>93</td>
<td></td>
</tr>
<tr>
<td>Making Multiple Links Change a Single Rollover</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>Working with Multiple Rollovers</td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>Creating Cycling Banners</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>Adding Links to Cycling Banners</td>
<td>106</td>
<td></td>
</tr>
<tr>
<td>Building Wraparound Slideshows</td>
<td>108</td>
<td></td>
</tr>
<tr>
<td>Displaying a Random Image</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>Cycling Images with a Random Start</td>
<td>113</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 5</th>
<th>Windows and Frames</th>
<th>115</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keeping a Page out of a Frame</td>
<td>117</td>
<td></td>
</tr>
<tr>
<td>Setting a Target</td>
<td>118</td>
<td></td>
</tr>
<tr>
<td>Loading iframes with JavaScript</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>Working with iframes</td>
<td>122</td>
<td></td>
</tr>
<tr>
<td>Creating Dynamic iframes</td>
<td>124</td>
<td></td>
</tr>
<tr>
<td>Sharing Functions Between Documents</td>
<td>126</td>
<td></td>
</tr>
<tr>
<td>Opening a New Window</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>Loading Different Contents into a Window</td>
<td>131</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 6</th>
<th>Form Handling</th>
<th>133</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select-and-Go Navigation</td>
<td>135</td>
<td></td>
</tr>
<tr>
<td>Changing Menus Dynamically</td>
<td>140</td>
<td></td>
</tr>
<tr>
<td>Making Fields Required</td>
<td>142</td>
<td></td>
</tr>
<tr>
<td>Checking Fields Against Each Other</td>
<td>147</td>
<td></td>
</tr>
<tr>
<td>Identifying Problem Fields</td>
<td>149</td>
<td></td>
</tr>
<tr>
<td>Putting Form Validation into Action</td>
<td>151</td>
<td></td>
</tr>
<tr>
<td>Working with Radio Buttons</td>
<td>156</td>
<td></td>
</tr>
<tr>
<td>Setting One Field with Another</td>
<td>159</td>
<td></td>
</tr>
<tr>
<td>Validating Zip Codes</td>
<td>162</td>
<td></td>
</tr>
<tr>
<td>Validating Email Addresses</td>
<td>166</td>
<td></td>
</tr>
</tbody>
</table>
### Chapter 7  
**Forms and Regular Expressions**  
- Validating an Email Address with Regular Expressions  
- Validating a File Name  
- Extracting Strings  
- Formatting Strings  
- Formatting and Sorting Strings  
- Formatting and Validating Strings  
- Replacing Elements Using Regular Expressions  

### Chapter 8  
**Handling Events**  
- Handling Window Events  
- Mouse Event Handling  
- Form Event Handling  
- Key Event Handling  
- Advanced Event Handling  

### Chapter 9  
**JavaScript and Cookies**  
- Baking Your First Cookie  
- Reading a Cookie  
- Showing Your Cookies  
- Using Cookies as Counters  
- Deleting Cookies  
- Handling Multiple Cookies  
- Displaying “New to You” Messages  

### Chapter 10  
**Objects and the DOM**  
- About Node Manipulation  
- Adding Nodes  
- Deleting Nodes  
- Deleting Specific Nodes  
- Inserting Nodes  
- Replacing Nodes  
- Writing Code with Object Literals  

### Chapter 11  
**Making Your Pages Dynamic**  
- Putting the Current Date into a Webpage  
- Working with Days  
- Customizing a Message for the Time of Day
<table>
<thead>
<tr>
<th>Chapter 12</th>
<th>Applied JavaScript</th>
<th>285</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Using Sliding Menus</td>
<td>286</td>
</tr>
<tr>
<td></td>
<td>Adding Pull-Down Menus</td>
<td>289</td>
</tr>
<tr>
<td></td>
<td>Enhancing Pull-Down Menus</td>
<td>293</td>
</tr>
<tr>
<td></td>
<td>A Slideshow with Captions</td>
<td>297</td>
</tr>
<tr>
<td></td>
<td>A Silly Name Generator</td>
<td>301</td>
</tr>
<tr>
<td></td>
<td>A Bar Graph Generator</td>
<td>306</td>
</tr>
<tr>
<td></td>
<td>Style Sheet Switcher</td>
<td>315</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 13</th>
<th>Introducing Ajax</th>
<th>325</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ajax: Pinning It Down</td>
<td>327</td>
</tr>
<tr>
<td></td>
<td>Reading Server Data</td>
<td>331</td>
</tr>
<tr>
<td></td>
<td>Parsing Server Data</td>
<td>339</td>
</tr>
<tr>
<td></td>
<td>Refreshing Server Data</td>
<td>346</td>
</tr>
<tr>
<td></td>
<td>Getting Data From a Server</td>
<td>349</td>
</tr>
<tr>
<td></td>
<td>Previewing Links with Ajax</td>
<td>353</td>
</tr>
<tr>
<td></td>
<td>Auto-Completing Form Fields</td>
<td>356</td>
</tr>
<tr>
<td></td>
<td>Checking Whether a File Exists</td>
<td>362</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 14</th>
<th>Toolkits, Frameworks, and Libraries</th>
<th>365</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adding jQuery</td>
<td>367</td>
</tr>
<tr>
<td></td>
<td>Updating a Page with jQuery</td>
<td>370</td>
</tr>
<tr>
<td></td>
<td>Interacting with jQuery</td>
<td>371</td>
</tr>
<tr>
<td></td>
<td>Interacting and Updating</td>
<td>374</td>
</tr>
<tr>
<td></td>
<td>Stripping Tables</td>
<td>376</td>
</tr>
<tr>
<td></td>
<td>Sorting Tables</td>
<td>380</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 15</th>
<th>Designing with jQuery</th>
<th>385</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Highlighting New Elements</td>
<td>386</td>
</tr>
<tr>
<td></td>
<td>Creating Accordion Menus</td>
<td>389</td>
</tr>
<tr>
<td></td>
<td>Creating Smarter Dialogs</td>
<td>392</td>
</tr>
</tbody>
</table>
## Table of Contents

Auto-Completing Fields ........................................... 396
Adding Sortable Tabs ................................................ 398
Using Check Boxes as Buttons ..................................... 401
Adding a Calendar to Your Page .................................... 404
Using ThemeRoller to Customize Your Look ....................... 409

### Chapter 1
Building on jQuery .................................................. 411
Using jQuery as a Foundation ....................................... 412
Dragging and Dropping Elements .................................. 414
Using jQuery with External Data ................................... 417
Using jQuery Plugins ................................................. 420
Adding a jQuery Audio Plugin ..................................... 423

### Chapter 17
Scripting Mobile Devices ........................................... 425
Changing Your Orientation .......................................... 426
Handling Touch Events .............................................. 433
Differentiating Devices .............................................. 436
Locating Your Device ................................................ 438

### Chapter 18
Bookmarklets .......................................................... 441
Your First Bookmarklet ............................................... 442
Resetting a Webpage’s Background ................................. 447
Changing a Page’s Styles ............................................. 448
Word Lookups ......................................................... 451
Viewing Images ....................................................... 454
Displaying ISO Latin Characters ................................... 456
Converting RGB Values to Hex ..................................... 459
Converting Values ..................................................... 461
A Bookmarklet Calculator .......................................... 463
Shortening URLs ...................................................... 465
Validating Pages ...................................................... 466
Mailing Pages ........................................................ 467
Resizing Pages ........................................................ 468

### Appendix A
JavaScript Genealogy and Reference .............................. 469
JavaScript Versions ................................................... 470
ECMAScript ............................................................ 472
The Big Object Table .................................................. 473
Appendix B  JavaScript Reserved Words  . . . . . . . . . . . . . 491

Appendix C  Cascading Style Sheets Reference  . . . . . . . . 495

Appendix D  Where to Learn More  . . . . . . . . . . . . . . . . 507
  Finding Help Online  . . . . . . . . . . . . . . . . . . . . . . . . . 508
  Offline Resources  . . . . . . . . . . . . . . . . . . . . . . . . . . 511
  Troubleshooting Tips  . . . . . . . . . . . . . . . . . . . . . . . . 512

Index  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 515
Welcome to JavaScript! Using this easy-to-learn programming language, you’ll be able to add interest and interaction to your webpages and make them more useful for you and for your site’s visitors. We’ve written this book as a painless introduction to JavaScript, so you don’t have to be a geek or a nerd to write a script. Pocket protectors will not be necessary at any time. As a friend of ours says, “We’re geeky, so you don’t have to be!”

We wrote this book for you

We figure that if you’re interested in JavaScript, then you’ve already got some experience in creating HTML pages and websites, and you want to take the next step by adding some interactivity to your sites. We don’t assume that you know anything about programming or scripting. We also don’t assume that you are an HTML expert (though if you are, that’s just fine). We do assume that you’ve got at least the basics of building webpages down, and that you have some familiarity with common HTML, such as links, images, and forms. Similarly, we assume basic knowledge of the other major building block of modern websites: CSS.

We include some extra explanation of HTML in sidebars called “Just Enough HTML.” You won’t find these sidebars in every chapter, just the ones where we think you’ll need a quick reference. Having this information handy means you won’t need multiple books or webpages open just to remember the syntax of a particular HTML attribute.

If you already know something about programming, you should be aware that we don’t take the same approach to JavaScript as you might have seen in other books. We don’t delve deeply into JavaScript’s syntax and structure, and we don’t pretend that this book is a comprehensive language reference (though you’ll find some valuable reference material in Appendix A in the back of the book). There are several other books on the market that do that job admirably, and we list them in Appendix D at the end of this book. The difference between
those books and this one is that instead of getting bogged down in formalism, we concentrate on showing you how to get useful tasks done with JavaScript without a lot of extraneous information.

In previous editions we added coverage of Ajax and jQuery, which use JavaScript and other common web technologies to add extra interactivity to webpages and to improve the user experience of your websites. In this edition, we’ve added even more examples and techniques using the popular jQuery framework.

**How to use this book**

Throughout the book, we’ve used some devices that should make it easier for you to work both with the book and with JavaScript itself.

In the step-by-step instructions that make up most of the book, we’ve used a special type style to denote either HTML, CSS, or JavaScript code, like this:

```html
<div id="thisDiv">
 → window.onload = initLinks;
</div>
```

You’ll also notice that we show the HTML and the JavaScript in lowercase. We’ve done that because all of the scripts in this edition are compliant with the HTML5 standard from the W3C, the World Wide Web Consortium. Whenever you see a quote mark in a JavaScript, it is always a straight quote (like ‘ or ”), never curly quotes (aka “smart” quotes, like ’ or ”). Curly quotes will prevent your JavaScript from working, so make sure that you avoid them when you write scripts.

In the illustrations accompanying the step-by-step instructions, we’ve highlighted the part of the scripts that we’re discussing in red, so you can quickly find what we’re talking about. We often also highlight parts of the screen shots of web browser windows in red, to indicate the most important part of the picture.

Because book pages are narrower than computer screens, some of the lines of JavaScript code are too long to fit on the page. When this happens, we’ve broken the line of code up into one or more segments, inserted this gray arrow → to indicate that it’s a continued line, and indented the rest of the line. Here’s an example of how we show long lines in scripts.

```javascript
dtString = "Hey, just what are you → doing up so late?";
```

**You say browser, we say kumbaya**

Beginning with the sixth edition of this book, we made a big change: we ended our support for browsers that are very old or that don’t do a good job of supporting web standards. We’d found that virtually all web users have upgraded and are enjoying the benefits of modern browsers, ones that do a good-to-excellent job of supporting commonly accepted web standards like HTML, CSS2, and the Document Object Model. That covers Internet Explorer 9 or later; all versions of Firefox; all versions of Safari and Chrome; and Opera 7 or later.

We’ve tested our scripts in a wide variety of browsers, on several different operating systems, including Windows (mostly
Windows 7 and, in a few cases, Windows 8; like Microsoft, we’ve dropped support for Windows XP and Vista), OS X (10.8.5 and later), and Ubuntu Linux (we tested scripts in Firefox, Ubuntu’s default browser).

We used the former 600-pound gorilla of the browser world, Microsoft Internet Explorer for Windows, to test virtually everything in the book (we used versions 9, 10, and 11). For this edition, we added testing in the frequently updated versions of Google Chrome for both Mac and Windows. We also tested the scripts with recent versions of Firefox (which updated every few weeks, ending with version 29) for Mac and Windows, and with Safari for Mac versions 6 and 7 (as Apple has discontinued development of Safari for Windows, we’ve dropped it from our testing regimen). Working with the latter browser means that our scripts should also work in any browsers based on the WebKit engine, and on browsers (such as Konqueror for Linux) based on KHTML, the open-source rendering engine from which Safari got its start. WebKit is also the basis for browsers in mobile operating systems, such as Apple’s iOS, Google’s Android, the Amazon Kindle Fire tablets, and BlackBerry Limited’s Blackberry 10. So far as mobile devices go, we mainly tested scripts on iPhones and iPads.

**Don’t type that code!**

Some JavaScript books print the scripts and expect you to type in the examples. We think that’s way too retro for this day and age. It was tough enough for us to do all that typing, and there’s no reason you should have to repeat that work.

So we’ve prepared a companion website for this book—one that includes all of the scripts in the book, ready for you to just copy and paste into your own web-pages. If we discover any mistakes in the book that got through the editing process, we’ll list the updates on the site, too. You can find our companion site at javascriptworld.com.

If for some reason you do plan to type in some script examples, you might find that the examples don’t seem to work, because you don’t have the supporting files that we used to create the examples. For example, in a task where an onscreen effect happens to an image, you’ll need image files. No problem. We’ve put all of those files up on the book’s website, nicely packaged for you to download. You’ll find one downloadable file that contains all of the scripts, HTML files, CSS files, and any media files we used. If you have any questions, please check the FAQ (Frequently Asked Questions) page on the companion website. It’s clearly marked.

If you’ve read the FAQ and your question isn’t answered there, you can contact us via email at js9@javascriptworld.com. We regret that because of the large volume of email that we get, we cannot, and will not, answer email about the book sent to our personal email addresses. We can only guarantee that messages sent to the js9@javascriptworld.com address will be answered.

On the other hand, typing code by hand is likely to give you a more thorough learning experience—so don’t rule it out entirely!
Time to get started

One of the best things about JavaScript is that it’s easy to start with a simple script that makes cool things happen on your webpage, then add more complicated stuff as you need it. You don’t have to learn a whole book’s worth of information before you can start improving your webpages. But by the time you’re done with the book, you’ll be adding advanced interactivity to your sites with JavaScript and jQuery.

Of course, every journey begins with the first step, and if you’ve read this far, your journey into JavaScript has already begun. Thanks for joining us; please keep your hands and feet inside the moving vehicle. And please, no flash photography.
One of the best (and most common) uses of JavaScript is to add visual interest to webpages by animating graphics, and that’s what this chapter is all about. Making an image on a webpage change when the user moves the mouse over the image, thereby making the page react to the user, is one of the most common—and effective—tricks you can learn in JavaScript. This rollover, as it is called, is easy to implement yet has many applications, as you’ll see.

Rollovers are a great tool, but you can do much more than rollovers with JavaScript, such as automatically change images, create ad banners, build slideshows, and display random images on a page.

In this chapter, you’ll learn how to make JavaScript do all of these image tricks. Let’s get started.

### In This Chapter

- Creating Rollovers: 83
- Creating More Effective Rollovers: 85
- Building Three-State Rollovers: 91
- Triggering Rollovers from a Link: 93
- Making Multiple Links Change a Single Rollover: 96
- Working with Multiple Rollovers: 99
- Creating Cycling Banners: 104
- Adding Links to Cycling Banners: 106
- Building Wraparound Slideshows: 108
- Displaying a Random Image: 111
- Cycling Images with a Random Start: 113
<table>
<thead>
<tr>
<th>Tag</th>
<th>Attribute</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>img</td>
<td>src</td>
<td>Contains the URL of the image, relative to the URL of the webpage</td>
</tr>
<tr>
<td></td>
<td>width</td>
<td>Contains the width (in pixels) at which the browser will display the image</td>
</tr>
<tr>
<td></td>
<td>height</td>
<td>Contains the height (in pixels) at which the browser will display the image</td>
</tr>
<tr>
<td></td>
<td>alt</td>
<td>Used for non-visual browsers in place of the image</td>
</tr>
</tbody>
</table>
Creating Rollovers

The idea behind rollovers is simple. You have two images. The first, or original, image is loaded and displayed along with the rest of the webpage by the user. When the user moves the mouse over the first image, the browser quickly swaps out the first image for the second, or replacement, image, giving the illusion of movement or animation.

Listing 4.1 gives you the bare-bones rollover; the whole thing is done within a standard image link. First a blue arrow is loaded, and then it is overwritten by a red arrow when the user moves the mouse over the image. The blue arrow is redrawn when the user moves the mouse away.

Some styles get applied to elements on the page, and we’ve broken those styles out into a separate CSS file, as seen in Listing 4.2.

To create a rollover:

1. `<a href="next.html"`  
   The link begins by specifying where the browser will go when the user clicks the image, in this case to the page next.html.

2. `onmouseover="document.images['arrow'].src='images/arrow_on.gif'"`  
   When the user moves the mouse over the image (the src of the arrow id), the replacement image arrow_on.gif, which is inside the images directory, is swapped into the document.

3. `onmouseout="document.images['arrow'].src='images/arrow_off.gif'"`  
   Then, when the mouse moves away, the image arrow_off.gif is swapped back in.

continues on next page
Disadvantages to This Kind of Rollover

This method of doing rollovers is very simple, but you should be aware that there are several problems and drawbacks with it.

- Because the second image is downloaded from the server at the time the user rolls over the first image, there can be a perceptible delay before the second image replaces the first one, especially for people browsing your site with a slower connection.

- Using this method causes an error message in ancient browsers, such as Netscape 2.0 or earlier, Internet Explorer 3.0 or earlier, or the America Online 2.7 browser. Since there are so few of these vintage browsers still in use, it’s not much of a problem these days.

Instead of using this method, we suggest that you use the following way to create rollovers, in the “Creating More Effective Rollovers” section, which solves all these problems and more.
Creating More Effective Rollovers

To make the illusion of animation work, you need to make sure that the replacement image appears immediately, with no delay while it is fetched from the server. To do that, you use JavaScript to place the images into variables used by your script, which preloads all the images into the browser’s cache (so that they are already on the user’s hard disk when they are needed). Then, when the user moves the mouse over an image, the script swaps one variable containing an image for a second variable containing the replacement image. Listing 4.3 shows how it’s done. The visible result is the same as in A and B from the previous exercise, but the apparent animation is smoother.

To keep your JavaScript more manageable, we’ll extract the JavaScript code from the HTML page and put it in an external .js file, as in Listing 4.4 (see Chapter 2 for more about .js files).

Listing 4.3 The only JavaScript on this HTML page is the pointer to the external .js file.

Listing 4.4 This is a better way to do rollovers than in Listing 4.1, because it is much more flexible.
To create a better rollover:

1. `<script src="script02.js"></script>`
   
   This tag is in Listing 4.3, the HTML page. It uses the `src` attribute to tell the browser where to find the external `.js` file, which is where the JavaScript resides.

2. `<a href="next1.html"><img src="images/button1_off.gif" alt="button1" id="button1"></a>&nbsp;&nbsp;<a href="next2.html"><img src="images/button2_off.gif" alt="button2" id="button2"></a>

   Still in Listing 4.3, these are two typical link tags for the buttons, with image tags embedded in them. The `href` attribute describes the destination of the link when the user clicks it. In the `img` tag, the `src` attribute provides the path to the image before the user rolls over it. The link tags also define the image’s `alt` text. Note that each of the two buttons also has an `id` attribute; as described in Chapter 1, the `id` must be unique for each object. The script uses the image’s `id` to make the rollover work.

3. `window.onload = rolloverInit;`
   
   Moving to Listing 4.4, the `window.onload` event handler is triggered when the page has finished loading. The handler calls the `rolloverInit()` function.

   This handler is used here to make sure that the script doesn’t execute before the page is done loading. That’s because referring to items on the page before the page has finished loading can cause errors if some of the page’s elements haven’t yet been loaded.
4. function rolloverInit() {
    for (var i=0; i<document.images.length; i++) {
        The rolloverInit() function scans each image on the page, looking to see if the tag around the image is an `<a>` tag, indicating that it is a link. The first of these two lines begins the function. The second begins a `for...next` loop that goes through all of the images. The loop begins by setting the counter variable `i` to 0. Then, each time the loop goes around, if the value of `i` is less than the number of images in the document, increment `i` by 1.

5. if (document.images[i].parentNode.tagName == "A") {
        This is where we test to see if the tag surrounding the image is an anchor tag. We do it by looking at an object and seeing if the object’s value is `A` (the anchor tag). Let’s break that object apart a bit. The first part of the object, `document.images[i]`, is the current image. Its `parentNode` property is the container tag that surrounds it, and `tagName` then provides the name of that container tag. So in English, you can read the part of the line in the parentheses as “For this particular image, is the tag around it an 'A'?"

6. setupRollover(document.images[i]);
        If the result of the test in step 5 is true, then the `setupRollover` function is called and passed the current image.

   continues on next page
7. function setupRollover(theImage) {

Take a minute to look at the whole function before we go through it line by line. Here’s the overview: this function adds two new properties to the image object that’s passed in. The new properties are `outImage` (the version of the image when you’re not on it) and `overImage` (the version of the image when you are on it), both of which are image objects themselves. Because they’re image objects, once they’re created, we can add their `src` property. The `src` for `outImage` is the current (off) image `src`. The `src` value for `overImage` is calculated based on the `id` attribute of the original image.

This line starts off the function with the image that was passed to it by the `rolloverInit()` function.

8. theImage.outImage = new Image();

This line takes the image object that was passed in and adds the new `outImage` property to it. Because you can add a property of any kind to an object, and because properties are just objects themselves, what’s happening here is that we’re adding an image object to an image. The parentheses for the new image object are optional, but it’s good coding practice to include them; if needed, you can set properties of the new image object by passing certain parameters.
9. \texttt{theImage.outImage.src = theImage.src;}

Now we set the source for the new \texttt{outImage} to be the same as the source of \texttt{theImage}. The default image on the page is always the version you see when the cursor is off the image.

10. \texttt{theImage.onmouseout = function() { this.src = this.outImage.src; }}

The first line here starts off what's called an \textit{anonymous function}—that is, it's a function without a name. We could name it (say, \texttt{rollOut}), but as it's only one line, why bother?

In this section, we're telling the browser to trigger what should happen when the user moves the mouse away from the image. Whenever that happens, we want to set the image source back to the initial source value, that is, the \texttt{outImage} version of the image.

11. \texttt{theImage.overImage = new Image(); theImage.overImage.src = "images/" + theImage.id + "_on.gif";}

In the first line, we create a new image object that will contain the \texttt{overImage} version of the image. The second line sets the source for \texttt{overImage}. It builds the name of the source file on the fly, concatenating \texttt{"images/"} with the \texttt{id} of the image (remember, in Listing 4.3, we saw that those \texttt{ids} were \texttt{button1} and \texttt{button2}) and adding \texttt{"_on.gif"}. 

\textit{continues on next page}
Here we have another anonymous function. This one tells the browser that when the user moves the cursor over the image, it should reset the current image’s source to that of the overImage version, as seen in A and B.

**TIP** When you prepare your graphics for rollovers, make sure that all your GIF or PNG images are not transparent. If they are, you will see the image you are trying to replace beneath the transparent image—and that’s not what you want.

**TIP** Both the original and the replacement images need to have identical dimensions. Otherwise, some browsers resize the images for you, and you probably won’t like the distorted result.

**TIP** In the previous example, the rollover happened when you moved the cursor over the link; here, the rollover happens when you move the cursor over the image—that is, the onmouseover and onmouseout are now attached to the image, not the link. While these methods usually give the same effect, there’s one big difference: some older browsers (Netscape 4 and earlier, IE 3 and earlier) don’t support onmouseover and onmouseout on the img tag.

**TIP** You might think that, because all of the tags on the HTML page are lowercase, tagName should be compared to a lowercase “a”. That’s not the way it works; tagName always returns an uppercase value.

**TIP** There are many different ways to script rollovers. We prefer this one due to its flexibility: images can be added to or subtracted from associated HTML pages without any code needing to be changed.
Building Three-State Rollovers

A three-state rollover is one where the rollover has three versions. Besides the original image and the version that appears when the user places the cursor over the image, there is a third version of the image when the button itself is clicked, as shown in A.

Listing 4.5, the HTML file, looks almost exactly the same as Listing 4.3 from the previous task. In fact, the only differences are the document’s title and the name of the external JavaScript file that is being called. That’s it. This is an example of why putting all your JavaScript into an external file is so powerful; you can add functionality to your pages without having to rework your HTML pages.

In Listing 4.6, the external JavaScript file, there are only a few changes from Listing 4.4. Rather than go through the whole script again, we’ll just focus on the changes. Remember, the parts of the script that we’re covering are shown in red in the code.

Listing 4.5

By putting your JavaScript in an external file, the HTML for a three-state rollover is virtually identical to a two-state rollover.

```html
<!DOCTYPE html>
<html>
<head>
    <title>Three-state Rollovers</title>
    <script src="script03.js"></script>
    <link rel="stylesheet" href="script01.css">
</head>
<body>
    <a href="next1.html"><img src="images/button1_off.gif" alt="button1" id="button1"></a>
    <a href="next2.html"><img src="images/button2_off.gif" alt="button2" id="button2"></a>
</body>
</html>
```
To build a three-state rollover:

1. `theImage.clickImage = new Image();
   theImage.clickImage.src = "images/" + theImage.id + 
   "_click.gif";

   In the `setupRollover()` function, we now need to add a third image property for the click state. In the first line, we create a new image object that will contain the `clickImage` version of the image. The second line sets the source for `clickImage`. It builds the name of the source file on the fly, concatenating "images/" with the `id` of the image, and adding "_click.gif".

2. `theImage.onclick = function() {
   this.src = this.clickImage.src;
}

   This tells the browser what to do when the user clicks the mouse on the image: in this case, we want to set the image source to its `clickImage` version.

   **Tip** If you're thinking about using a script like this on your own site, a more complete version is Listing 7.9, in “Replacing Elements Using Regular Expressions,” and its final version is Listing 13.19, in “Checking Whether a File Exists.”

```javascript
Listing 4.6 This script powers the three-state rollover.

window.onload = rolloverInit;

function rolloverInit() {
   for (var i=0; i<document.images.length; i++) {
      if (document.images[i].parentNode.tagName == "A") {
         setupRollover(document.images[i]);
      }
   }
}

function setupRollover(theImage) {
   theImage.outImage = new Image();
   theImage.outImage.src = theImage.src;
   theImage.onmouseout = function() {
      this.src = this.outImage.src;
   }

   theImage.clickImage = new Image();
   theImage.clickImage.src = "images/" + theImage.id + 
   "_click.gif";
   theImage.onclick = function() {
      this.src = this.clickImage.src;
   }

   theImage.overImage = new Image();
   theImage.overImage.src = "images/" + theImage.id + 
   "_on.gif";
   theImage.onmouseover = function() {
      this.src = this.overImage.src;
   }
}
```
Triggering Rollovers from a Link

In earlier examples, the user triggered the rollover by moving the mouse over an image. But you can also make a rollover occur when the user hovers over a text link, as in A and B. The HTML is an unexciting page with one link and one image, shown in Listing 4.7. We’ll do the rollover by modifying the script used in previous examples, as in Listing 4.8.

A The text link is the triggering device for this rollover.

B When the user points at the link, the graphic below changes.

Listing 4.7 This script shows the HTML for a rollover from a text link.

```html
<!DOCTYPE html>
<html>
<head>
  <title>Link Rollover</title>
  <script src="script04.js"></script>
  <link rel="stylesheet" href="script01.css">
</head>
<body>
  <h1><a href="next.html" id="arrow">Next page</a></h1>
  <img src="images/arrow_off.gif" id="arrowImg" alt="arrow">
</body>
</html>
```
To trigger a rollover from a link:

1. function rolloverInit() {
   for (var i=0; i<document.links.length; i++) {

   After beginning the rolloverInit() function, we start a loop, much like previous examples in this chapter. But there we were looking for images (document.images.length), and here we’re looking for links (document.links.length). The loop begins by setting the counter variable i to zero. Every time around, if the value of i is less than the number of links in the document, increment i by 1.

2. var linkObj = document.links[i];

   We create the linkObj variable and set it to the current link.

3. if (linkObj.id) {
   var imgObj = document.getElementById(linkObj.id + "Img");

   If linkObj has an id, then we check to see if there’s another element on the page that has an id that’s the same plus Img. If so, put that element into the new variable imgObj.

4. if (imgObj) {
   setupRollover(linkObj,imgObj);

   If imgObj exists, then call the setupRollover() function, passing it the link object and the image object.

Listing 4.8 Here is the JavaScript for a rollover from a text link.

window.onload = rolloverInit;

function rolloverInit() {
   for (var i=0; i<document.links.length; i++) {
      var linkObj = document.links[i];
      if (linkObj.id) {
         var imgObj = document.getElementById(linkObj.id + "Img");
         if (imgObj) {
            setupRollover(linkObj,imgObj);
         }
      }
   }
}

function setupRollover(theLink, theImage) {
   theLink.imgToChange = theImage;
   theLink.onmouseout = function() {
      this.imgToChange.src = this.outImage.src;
   }
   theLink.onmouseover = function() {
      this.imgToChange.src = this.overImage.src;
   }
   theLink.outImage = new Image();
   theLink.outImage.src = theImage.src;

   theLink.overImage = new Image();
   theLink.overImage.src = "images/" + theLink.id + "_on.gif";
}
5. function setupRollover
   ➝ (theLink, theImage) {
       theLink.imgToChange = theImage;
   }

   The setupRollover() function begins with the link and image parameters that were passed to it in step 4. Then we add a new property, imgToChange, to the link object. JavaScript needs some way of knowing what image is to be changed when the link is moused over, and this is where it's stored.

6. theLink.onmouseout = function() {
       this.imgToChange.src = ➝ this.outImage.src;
   }

   theLink.onmouseover = function() {
       this.imgToChange.src = ➝ this.overImage.src;
   }

   When the mouseover and mouseout are triggered, they're slightly different from the previous examples in this chapter: now, this.imgToChange.src is being reset instead of this.src itself.

   Tip: This technique is useful when you want to provide the user with a preview of what they will see if they click the link at which they are pointing. For example, say you have a travel site describing trips to Scotland, Tahiti, and Cleveland. On the left of the page could be a column of text links for each destination, while on the right could be a preview area where an image appears. As the user points at the name of a destination, a picture of that place appears in the preview area. Clicking the link takes the user to a page detailing their fabulous vacation spot.
Making Multiple Links Change a Single Rollover

Up to now, you've seen how mousing over a single area can trigger a rollover effect. But you can also have several different areas that trigger a rollover. This can be very useful, for example, when you have several images that you want to annotate; that is, where rolling over each of the images makes the description of that image appear. In this example, we've done just this with images of three of Leonardo da Vinci's inventions. As you roll over each image, the description of that image appears elsewhere. The description itself is another image. Actually, it's three images, one for each of the three inventions. A shows Listing 4.9

Listing 4.9 Note that the links and images on this page all have unique ids.

```html
<!DOCTYPE html>
<html>
<head>
    <title>Multiple Links, Single Rollover</title>
    <script src="script05.js"></script>
    <link rel="stylesheet" href="script02.css">
</head>
<body>
    <div id="captionDiv">
        <img src="images/DaVinci.jpg" width="144" height="219" alt="DaVinci">
        <img src="images/bg.gif" id="captionField" alt="Text Field">
    </div>
    <div id="inventionDiv">
        <img src="images/leoText.gif" id="heading" alt="Leonardo's Inventions">
        <a href="flyPage.html" class="captionField" id="flyer">
            <img src="images/flyer.gif" width="293" height="165" alt="Flying Machine" id="flyerImg"></a>
        <a href="tankPage.html" class="captionField" id="tank">
            <img src="images/tank.gif" width="325" height="92" alt="Tank" id="tankImg"></a>
        <a href="heliPage.html" class="captionField" id="helicopter">
            <img src="images/helicopter.gif" width="224" height="160" alt="Helicopter" id="helicopterImg"></a>
    </div>
</body>
</html>
```
Listing 4.10 In this CSS file, we define the classes we reference in the HTML.

```css
body {
    background-color: #EC9;
}

ing {
    border-width: 0;
}

#captionDiv {
    float: right;
    width: 210px;
    margin: auto 50px;
}

#captionField {
    margin: 20px auto;
    width: 208px;
    height: 27px;
}

#inventionDiv {
    width: 375px;
    margin-left: 20px;
}

#heading {
    margin-bottom: 20px;
    width: 375px;
    height: 26px;
}
```

(HTML), Listing 4.10 (CSS), and Listing 4.11 (JavaScript) in action. As with most of the scripts in this book, it builds on previous examples, so we’ll just explain the new concepts. There are just a few lines that are different between Listing 4.8 and Listing 4.11.
To make multiple links change a single rollover:

1. if (linkObj.className) {
   var imgObj = document.getElementById(linkObj.className);
}

   We can’t use the id of the rolled-over images to calculate the id of the changed image—that’s because an id has to be unique, and all of the rolled-over images have to come up with the same value for the changed image destination. Instead, we’re using the class attribute (because you can have multiple page elements sharing the same class). In this line, we’re looking for the className of the link object.

2. function setupRollover(theLink, textImage) {
   theLink.imgToChange = textImage;
}

   The setupRollover() function is passed the current link object (theLink) and the image object, which we’re calling textImage. Note that when we passed these objects (which can also be referred to as variables) in, we called them linkObj and imgObj, respectively.

   The rest of the script works the same way as the previous examples in this chapter.

Listing 4.11 This script shows you how to use multiple links to trigger a single rollover.

window.onload = rolloverInit;

function rolloverInit() {
   for (var i=0; i<document.links.length; i++) {
      var linkObj = document.links[i];
      if (linkObj.className) {
         var imgObj = document.getElementById(linkObj.className);
         if (imgObj) {
            setupRollover(linkObj, imgObj);
         }
      }
   }
}

function setupRollover(theLink, textImage) {
   theLink.imgToChange = textImage;
   theLink.onmouseout = function() {
      this.imgToChange.src = this.outImage.src;
   }
   theLink.onmouseover = function() {
      this.imgToChange.src = this.overImage.src;
   }
    
   theLink.outImage = new Image();
   theLink.outImage.src = textImage.src;

   theLink.overImage = new Image();
   theLink.overImage.src = "images/" + theLink.id + "Text.gif";"
When you roll over one of the images, a description appears and a drop shadow appears around the image itself.

---

**Working with Multiple Rollovers**

What if you want the image that triggers the rollover to also be a rollover itself? A builds on the last example and shows how we’ve added this feature. When you roll over one of the invention images, it makes the description image appear, as before, but this time also swaps out the invention image for another image with a drop shadow. This gives the user visual feedback about what they’re pointing at (as if the mouse pointer isn’t enough!).

Listing 4.12 is the HTML page (no changes except for the title and the name of the external JavaScript file being called), and Listing 4.13 shows the additions to the JavaScript from the previous example.

---

**Listing 4.12** This HTML is identical to Listing 4.9, except for the title and reference to the external script.

```html
<!DOCTYPE html>
<html>
<head>
<title>Multiple Links, Multiple Rollovers</title>
<script src="script06.js"></script>
<link rel="stylesheet" href="script02.css">
</head>
<body>

<div id="captionDiv">
  <img src="images/DaVinci.jpg" width="144" height="219" alt="DaVinci">
  <img src="images/bg.gif" id="captionField" alt="Text Field">
</div>

<div id="inventionDiv">
  <img src="images/leoText.gif" id="heading" alt="Leonardo's Inventions">
  <a href="flyPage.html" class="captionField" id="flyer"><img src="images/flyer.gif" width="293" height="165" alt="Flying Machine" id="flyerImg"></a>
  <a href="tankPage.html" class="captionField" id="tank"><img src="images/tank.gif" width="325" height="92" alt="Tank" id="tankImg"></a>
  <a href="heliPage.html" class="captionField" id="helicopter"><img src="images/helicopter.gif" width="224" height="160" alt="Helicopter" id="helicopterImg"></a>
</div>

</body>
</html>
```
Listing 4.13 This script handles the multiple rollovers.

```javascript
window.onload = rolloverInit;

function rolloverInit() {
    for (var i=0; i<document.links.length; i++) {
        var linkObj = document.links[i];
        if (linkObj.className) {
            var imgObj = document.getElementById(linkObj.className);
            if (imgObj) {
                setupRollover(linkObj, imgObj);
            }
        }
    }
}

function setupRollover(theLink, textImage) {
    theLink.imgToChange = new Array;
    theLink.outImage = new Array;
    theLink.overImage = new Array;

    theLink.imgToChange[0] = textImage;
    theLink.onmouseout = rollOut;
    theLink.onmouseover = rollOver;

    theLink.outImage[0] = new Image();
    theLink.outImage[0].src = textImage.src;

    theLink.overImage[0] = new Image();
    theLink.overImage[0].src = "images/" + theLink.id + "Text.gif";

    var rolloverObj = document.getElementById(theLink.id + "Img");
    if (rolloverObj) {
        theLink.imgToChange[1] = rolloverObj;

        theLink.outImage[1] = new Image();
        theLink.outImage[1].src = rolloverObj.src;

        theLink.overImage[1] = new Image();
        theLink.overImage[1].src = "images/" + theLink.id + "_on.gif";
    }
}

function rollOver() {
    for (var i=0; i<this.imgToChange.length; i++) {
        this.imgToChange[i].src = this.overImage[i].src;
    }
}

function rollOut() {
    for (var i=0; i<this.imgToChange.length; i++) {
        this.imgToChange[i].src = this.outImage[i].src;
    }
}
```
To work with multiple rollovers:

1. theLink.imgToChange = new Array;
   theLink.outImage = new Array;
   theLink.overImage = new Array;

   These lines were added because
   the script has more images to work
   with (two for each rollover). In each
   line, we’re creating a new property of
   theLink, each of which is an array.

2. theLink.imgToChange[0] =
   textImage;

   In the previous task, imgToChange
   was an image, but in this task, it’s an
   array that will contain images. Here,
   textImage is stored in the first element
   of imgToChange.

3. theLink.outImage[0] = new Image();
   theLink.outImage[0].src =
   textImage.src;

   As previously, we need to store the out
   (off) version of the image, but this time
   it’s stored in the first element of the
   outImage array.

4. theLink.overImage[0] =
   new Image();
   theLink.overImage[0].src =
   "images/" + theLink.id +
   "Text.gif";

   Similarly, the over (on) version of the
   image is calculated and stored in the
   first element of overImage.

   continues on next page
5. var rolloverObj = document.
   → getElementById(theLink.id +
   → "Img");
   if (rolloverObj) {
     Now we need to figure out if this roll-
     over will trigger multiple images, not
     just an individual image. If that's the
     case, there will be an element on the
     HTML page whose id is the same as
     this one, but with Img appended. That
     is, if we're working on flyer, we'll be
     checking to see if there's a flyerImg
     element on the page. If there is, it's
     saved in rolloverObj, and we should
     do the next three steps.

6. theLink.imgToChange[1] =
   → rolloverObj;
   In the same way that we set
   imgToChange[0] above, we now
   set imgToChange[1] (the second
   element in the array) to the new
   rolloverObj. When the onmouseout
   and onmouseover event handlers are
   triggered, both images swap to their
   alternate versions, as we'll see later.

7. theLink.outImage[1] = new Image();
   theLink.outImage[1].src =
   → rolloverObj.src;
   This sets the second array element
   of outImage to the out (off) version of
   the image.

8. theLink.overImage[1] =
   → new Image();
   theLink.overImage[1].src =
   → "images/" + theLink.id +
   → "_on.gif";
   And here, the over (on) version of the
   image is calculated and stored in the
   second element of overImage.
If, for some reason, we wanted a third image to also change during this same rollover, we'd repeat steps 6–8 with the third image object.

9. for (var i=0; i<this.imgToChange.length; i++) {
   this.imgToChange[i].src = this.overImage[i].src;
}

Here inside the rollOver() function is where the images get swapped. Because one or more images can be changed, we need to start by asking how many images we have stored—that's the value of this.imgToChange.length. Here, the value is 2, because we want two images to change. We then loop through two times, setting the source of imgToChange[0] and then imgToChange[1] to their respective over values.

10. for (var i=0; i<this.imgToChange.length; i++) {
    this.imgToChange[i].src = this.outImage[i].src;
}

This code in the rollOut() function is virtually the same as that in the previous step; the only difference is that we're now resetting those images to their out source values.

**Tip** It's important to remember that every image that ever gets rolled over must have a unique id.

**Tip** What if you want some of the links on your page to trigger multiple rollovers, but others to be individual rollovers? No problem—you don't even need to change a line of JavaScript. So long as the check in step 5 doesn't find the alternate id on the page, no second element is stored, and the rollOver() and rollOut() loops only animate the initial image.
Creating Cycling Banners

When you surf the web, it’s common to see advertising banners that periodically switch between images. Some of these are animated GIF files, which are GIF files that contain a number of frames that play in succession; others are Flash animations. If you want to have a page that cycles through a number of GIFs (either animated or not), you can use JavaScript to do the job, as in Listing 4.15. This example uses three GIFs and cycles repeatedly through them, as shown in A, B, and C. The simple HTML page is shown in Listing 4.14.

To create cycling banners:

1. var theAd = 0;
   var adImages = new Array
       ("images/reading1.gif",
        "images/reading2.gif",
        "images/reading3.gif");

   Our script starts by creating theAd, which is given its beginning value in this code. The next line creates a new array called adImages. In this case, the array contains the names of the three GIF files that make up the cycling banner.

2. function rotate() {
   theAd++;
}

   We start off with a new function called rotate().

3. theAd++;

   Take the value of theAd, and add one to it.

4. if (theAd == adImages.length) {
   theAd = 0;

   This code checks to see if the value of theAd is equal to the number of items in the adImages array; if it is, then set the value of theAd back to zero.
5. `document.getElementById("adBanner").src = adImages[theAd];`

The image on the web that is being cycled has the id `adBanner`; you define the name as part of the `img` tag, as shown in Listing 4.14. This line of code says that the new sources for `adBanner` are in the array `adImages`, and the value of the variable `theAd` defines which of the three GIFs the browser should use at this moment.

6. `setTimeout(rotate, 3 * 1000);`

This line tells the script how often to change GIFs in the banner. The built-in JavaScript command `setTimeout()` lets you specify that an action should occur on a particular schedule, always measured in milliseconds. In this case, the function `rotate()` is called every 3000 milliseconds, or every 3 seconds, so the GIFs will cycle in the banner every three seconds.

**Tip** You might be wondering why you would want to use JavaScript for a cycling banner, rather than just create an animated GIF. One good reason is that it lets you use JPEGs or PNGs in the banner, which gives you higher-quality images. With these higher-quality images, you can use photographs in your banners.

**Tip** Unlike in some of the previous examples in this chapter, the images in this task are not pre-cached. Each downloads from the server the first time that it’s displayed. This is because you might have any number of images in your ad array, and it’s not polite to force users to download, for example, 100 images if they’re only going to see 2 or 3 of them.

---

**Listing 4.14** The HTML loads the first image in the cycling banner; the JavaScript handles the rest.

```html
document.getElementById("adBanner").src = adImages[theAd];
```

**Listing 4.15** You can use JavaScript to cycle between images in a banner.

```javascript
window.onload = rotate;
var theAd = 0;
var adImages = new Array("images/reading1.gif","images/reading2.gif","images/reading3.gif");

function rotate() {
    theAd+=;
    if (theAd == adImages.length) {
        theAd = 0;
    }
    document.getElementById("adBanner").src = adImages[theAd];
    setTimeout(rotate, 3 * 1000);
}
```
Adding Links to Cycling Banners

Banners are often used in advertising, and you’ll want to know how to make a banner into a link that will take a visitor somewhere when the visitor clicks the banner. Listing 4.16 shows the HTML page, which differs from the last example only in that it adds a link around the `img` tag. Listing 4.17 shows a variation of the previous script. In this script, we’ll add a new array. This new array contains destinations that users will be sent to when they click the banner. In this case, the “Eat at Joe’s” banner takes you to `negrino.com`, “Drink More Java” goes to `sun.com`, and “Heartburn” goes to `microsoft.com`, as shown in A. No editorial comments implied, of course.

To add links to cycling banners:

1. `window.onload = initBannerLink;`

   When the window finishes loading, trigger the `initBannerLink()` function.

Listing 4.16 The HTML needed for an ad banner.

```html
<!DOCTYPE html>
<html>
<head>
<title>Cycling Banner with Links</title>
<script src="script08.js"></script>
<link rel="stylesheet" href="script01.css">
</head>
<body>

<div class="centered">
  <a href="linkPage.html"><img src="images/banner1.gif" id="adBanner" alt="ad banner"></a>
</div>

</body>
</html>
```
Listing 4.17 This script shows how you can turn cycling banners into real, clickable ad banners.

```javascript
window.onload = initBannerLink;
var theAd = 0;
var adURL = new Array("negrino.com", "sun.com", "microsoft.com");
var adImages = new Array("images/banner1.gif", "images/banner2.gif", "images/banner3.gif");

function initBannerLink() {
    if (document.getElementById("adBanner").parentNode.tagName == "A") {
        document.getElementById("adBanner").parentNode.onclick = newLocation;
    }
    rotate();
}

function newLocation() {
    document.location.href = "http://www." + adURL[theAd];
    return false;
}

function rotate() {
    theAd++;
    if (theAd == adImages.length) {
        theAd = 0;
    }
    document.getElementById("adBanner").src = adImages[theAd];
    setTimeout(rotate, 3 * 1000);
}
```

2. if (document.getElementById("adBanner").parentNode.tagName == "A") {
    document.getElementById("adBanner").parentNode.onclick = newLocation;
} rotate();

This code, inside the initBannerLink() function, first checks to see if the adBanner object is surrounded by a link tag. If so, when the link is clicked, the newLocation() function will be called. Finally, the rotate() function is called.

3. document.location.href = "http://www." + adURL[theAd];
   return false;

Inside newLocation(), we set the document.location.href object (in other words, the current document window) to the value of the text string "http://www." (notice the period), plus the value of one item from adURL. Since adURL is an array, you need to specify a member of the array. That’s stored in theAd, and the resulting string can be any of the three links, depending on when the user clicks. Last, it returns false, which tells the browser that it should not also load in the href. Otherwise, the browser would do both. We’ve handled everything within JavaScript, so the href doesn’t need to be loaded.

Tip The adURL array needs to have the same number of array items as the adImages array for this script to work correctly.
Building Wraparound Slideshows

Slideshows on websites present the user with an image and let the user control the progression (either forward or backward) of the images. JavaScript gives the user the interactive control needed. Listing 4.18 shows the HTML needed, and the JavaScript in Listing 4.19 has what you need to add slideshows to your pages.

This script builds a slideshow that wraps around—that is, if you go past the end of the list you go back to the beginning and vice versa. A shows the new slideshow.

Listing 4.18 This HTML page creates a slideshow.

```html
<!DOCTYPE html>
<html>
<head>
<title>Image Slideshow</title>
<script src="script09.js"></script>
<link rel="stylesheet" href="script01.css">
</head>
<body>
<div class="centered">
<h1>Welcome, Robot Overlords!</h1>
<img src="images/robot1.jpg" id="myPicture" width="200" height="400" alt="Slideshow">
<h2><a href="previous.html" id="prevLink"><< Previous</a> &nbsp;&nbsp;<a href="next.html" id="nextLink">Next &gt;&gt;</a></h2>
</div>
</body>
</html>
```

Listing 4.19 This script builds a slideshow that the user can click through using links to control movement forward and back.

```javascript
window.onload = initLinks;
var thePic = 0;
var myPix = new Array("images/robot1.jpg","images/robot2.jpg","images/robot3.jpg");

function initLinks() {
  document.getElementById("prevLink").onclick = processPrevious;
  document.getElementById("nextLink").onclick = processNext;
}

function processPrevious() {
  if (thePic == 0) {
    thePic = myPix.length;
  }
  thePic--;
  document.getElementById("myPicture").src = myPix[thePic];
  return false;
}

function processNext() {
  thePic++;
  if (thePic == myPix.length) {
    thePic = 0;
  }
  document.getElementById("myPicture").src = myPix[thePic];
  return false;
}
```
To build a wraparound slideshow:

1. `window.onload = initLinks;`
   - When the window finishes loading, trigger the `initLinks()` function.

2. `function initLinks() {`
   - `document.getElementById("prevLink").onclick = processPrevious;`
   - `document.getElementById("nextLink").onclick = processNext;`
}

This function sets up the `onclick` event handlers for the Previous and Next links.

`continue on next page`

Clicking the Previous or Next link calls the `processPrevious()` or `processNext()` function, respectively.
3. function processPrevious() {
    if (thePic == 0) {
        thePic = myPix.length;
    }

    This function makes the slideshow run in the Previous direction. This first part checks to see if thePic is equal to 0. If it is, the function gets the number of pictures in the myPix array.

4. thePic--;
    document.getElementById("myPicture").src = myPix[thePic];

    The first line reduces the value of thePic by 1. The next line sets the src of myPicture to the element of the myPix array represented by the current value of thePic.

5. thePic++;
    if (thePic == myPix.length) {
        thePic = 0;
    }
    document.getElementById("myPicture").src = myPix[thePic];

    This code, inside the processNext() function, makes the slideshow run in the Next direction and is much like the processPrevious() function. The first thing it does is increment the value of thePic by 1. Then it checks to see if the value of thePic is the same as the number of items in the myPix array. If so, it sets thePic back to 0. The next line sets the src of myPicture.
Displaying a Random Image

If your site is rich with graphics, or if you are displaying digital artwork, then you may want to have a random image from your collection appear when the user enters your site. Once again, JavaScript to the rescue! The extremely simple Listing 4.20 shows the required HTML, and Listing 4.21 provides the JavaScript. A shows the result of the script, in this case images of a stuffed lion, tiger, and bear (oh, my!).

Depending on the value of the random number generated by the script, the user is presented with the lion, the tiger, or the bear.
To display a random image:

1. `var myPix = new Array
   ➔ ("images/lion.jpg", "images/tiger.jpg", "images/bear.jpg");`
   Here we build an array of three images, and stuff it into the variable myPix.

2. `var randomNum = Math.floor
   ➔ (Math.random() * myPix.length);`
   The variable called randomNum gets the value of a math expression that’s best read from the inside outwards.
   `Math.random` generates a random number between 0 and 1, which is then multiplied by `myPix.length`, which is the number of items in the array (in this case, it’s 3). `Math.floor` rounds the result down to an integer, which means that the number must be between 0 and 2.

3. `document.getElementById
   ➔ ("myPicture").src =
   ➔ myPix[randomNum];`
   This says that the source of the image myPicture is set based on the array myPix, and the value at this moment is dependent on the value of randomNum.

Listing 4.20 This simple HTML creates the page for a random image.

```html
<!DOCTYPE html>
<html>
<head>
    <title>Random Image</title>
    <script src="script10.js"></script>
    <link rel="stylesheet" href="script01.css">
</head>
<body>
    <img src="images/spacer.gif" width="305" height="312" id="myPicture" alt="some image">
</body>
</html>
```

Listing 4.21 You can display random images on your page with this script, which uses JavaScript’s `Math.random` method to generate a random number.

```javascript
window.onload = choosePic;

var myPix = new Array("images/lion.jpg", "images/tiger.jpg", "images/bear.jpg");

function choosePic() {
    var randomNum = Math.floor
    ➔ (Math.random() * myPix.length);
    document.getElementById("myPicture").
    ➔ src = myPix[randomNum];
}
```
Cycling Images with a Random Start

If you have a number of images that you want to display, you may not want to display them beginning with the same image each time the page is loaded. **Listing 4.22** has the HTML, and **Listing 4.23** combines the code used earlier for the cycling ad banners with the random image code.

**Listing 4.22** There's a spacer GIF in the HTML file, which is a placeholder until the ad banner appears.

```html
<!DOCTYPE html>
<html>
<head>
    <title>Cycling Random Banner</title>
    <script src="script11.js"></script>
    <link rel="stylesheet" href="script01.css">
</head>
<body>
    <div class="centered">
        <img src="images/spacer.gif" id="adBanner" alt="Ad Banner">
    </div>
</body>
</html>
```

**Listing 4.23** This script allows you to start your cycling image show with a random image.

```javascript
window.onload = choosePic;

var theAd = 0;
var adImages = new Array("images/reading1.gif","images/reading2.gif","images/reading3.gif");

function choosePic() {
    theAd = Math.floor(Math.random() * adImages.length);
    document.getElementById("adBanner").src = adImages[theAd];
    rotate();
}

function rotate() {
    theAd++;
    if (theAd == adImages.length) {
        theAd = 0;
    }
    document.getElementById("adBanner").src = adImages[theAd];
    setTimeout(rotate, 3 * 1000);
}
```
To start images cycling from a random start:

1. `var adImages = new Array("images/reading1.gif", "images/reading2.gif", "images/reading3.gif");`

   As in previous examples, set up the array and the variable that contains the number of items in the array.

2. `function rotate() {`

   This function is similar to the `rotate()` function in Listing 4.15. See that explanation for the details of how it works.
Symbols

= (equals sign) assignment, 15, 16, 34
== (equivalence) comparison, 16
=== (identical) comparison, 16
> (greater than) comparison, 16
>= (greater than or equal to) comparison, 16
< (less than) comparison, 16
<= (less than or equal to) comparison, 16
% (modulo) operator, 15
%= (modulo) assignment, 16
! (exclamation point) character, 16
!= (not equal to) comparison, 16
!== (not identical) comparison, 16
" (quotes)
in bookmarklets, 447
enclosing string values in, 15, 59
in open() command, 130
single vs. double, 84, 447, 451
in user alerts, 32
# (hash symbol) character, 19
$ (dollar sign) character, 176, 177, 367–368
& (ampersand) operator, 70, 76
&& (and) comparison, 16, 74
&nbis; (non-breaking space), 51
() (parentheses)
in functions, 25, 28
in methods, 12
in regular expressions, 174–175, 177
* (asterisk) character, 15, 175, 177
*=(multiplication) assignment, 16
+ (plus sign) character, 15, 174, 177
++ (increment) operator, 15, 53
+= (addition) assignment, 16
- (minus sign) operator, 15
-= (decrement) operator, 15
-= (subtraction) assignment, 16
. (dot/period) character, 12, 19, 177
/ (slash) operator, 15, 173
// (comment indicators), 29
/= (division) assignment, 16
/* and */ (comment indicators), 29
; (semicolon) character, 23–24, 441, 450
? (question mark) character, 34, 175, 177
@media queries, 430
[] (brackets), 174, 177
\ (backslash), 123, 174, 177
^ (caret) character, 174, 177
{} (braces), 25, 34, 175, 177
| (or) character, 70, 76, 177
|| (or) comparison, 16, 74
1-up calendars, 404–405
2-digit year, 277, 405
2-up calendars, 404–405
3-state rollovers, 91–92, 191
4-digit year, 277, 405
12-hour time format, 272
24-hour time format, 272

A

<a> (anchor) tags, 22, 87, 118, 119
abs() method, 54
abort() method, 333
acceleration property, 432
accelerationIncludingGravity property, 432
accented characters, 456–458
accessibility, 192, 296
accordion menus, 389–391, 398
acos() method, 54
action attribute, 134
ActiveX, 332, 338
Adaptive Path, 327
addEventListener() method, 216, 218, 333
Adobe
   Dreamweaver, 20, 138, 172
   Fireworks, 459
   Flash, 4, 104
   Photoshop, 306, 459
advertising banners, 104–107, 113–114
Ajax, 325–364
   article about, 327
   auto-completing form fields with, 356–361
   and back buttons, 330
   browser considerations, 329
   and caching, 338, 348
   checking whether file exists with, 362–364
   coining of term, 9, 327
how it works, 328–329
and jQuery, 412–413
and JSON format, 260, 349
parsing server data with, 339–345, 349–352
previewing links with, 353–356
purpose of, 8–9
refreshing server data with, 346–348
requesting/reading server data with, 331–338,
   349–352
and server-side technologies, 330
testing, 338
ways of using, 325–326
web technologies included in, 9–10, 327, 361
alert boxes, 215
   alert() method, 32
   alert windows, 31–32
alpha property, 432
alphabetizing names, 186–187
alt attribute, 82, 84
altKey property, 432
and operator, 70, 76
and (&&) operator, 16, 74
anchor (<a>) tags, 22, 87, 118, 119
and (&) operator, 70, 76
and (&&) operator, 16, 74
Android devices, 431, 437, 440. See also mobile devices
animated GIFs, 104, 105
animation, 81, 83, 85, 104–105
annotating scripts, 29–30
anonymous functions, 89, 90, 136, 386
AOL (America Online), 42, 84
Apache, 172
appendChild() method, 244, 253
Apple
   iOS simulator, 431
   and Macworld Expo, 80
map services, 440
Maps app, 438
Safari browser (See Safari)
aplets, Java, 4, 5
apps, launching mobile, 440
arithmetic operators, 15
arrays
defined, 59
updating, 62–63
using string, 77–80
asin() method, 54
assignment operators, 16
asterisk (*), 15, 175, 177
Asynchronous JavaScript and XML, 9, 327. See also Ajax
atan() method, 54
Atom feeds, 340
attributes
   action, 134
   alt, 82, 84
   autocomplete, 357
class, 18–19, 69–70, 72, 74, 99
deprecated, 24
for, 134
height, 82
href, 22
language, 24
maxlength, 134
name, 116, 119, 134
selected, 134
size, 134
src, 22, 26, 82, 116
style, 70
target, 118–119
type, 24, 134
value, 134
width, 82
audio-player plugin, 423–424
autocomplete attribute, 357
auto-completing fields, 356–361, 396–397
back button, 117, 330
background color, changing page’s, 447, 448–450
background properties (CSS), 498
backslash (\), 123, 174, 177
banners, 104–107, 113–114
bar graph generator, 306–314
Bare Bones Software, 20
BBEdit, 20, 172
**beta** property, 432
binary math, 70, 74, 75–76
binary values, 70, 71
Bingo cards
adding interactivity to, 68–70
applying styles to, 52, 68–70
avoiding duplicate numbers in, 62–63, 64
checking for winning state, 71–74
creating skeleton for, 50–51
limiting range of values in, 59
possible winning patterns for, 75
range of allowable numbers for, 52, 59
using loop to create table for, 53–54
**bit.ly**, 465
bits, 70, 71, 75–76
bitwise arithmetic, 70, 72, 75–76
Blackberry devices, 437. See also mobile devices
blind users, 296
Blink, 472
block-level elements, 18
**blog.jquery.com**, 384
blogs
DailyJS, 510
jQuery, 384
Mozilla Hacks, 509
QuirksMode, 510
Safari, 509
blur() method, 132
body scripts, 23
**<body>** tags, 22, 23
bookmarklets, 441–468
for changing page’s styles, 448–450
for converting kilometers to miles, 461–462
for converting RGB values to hex, 459–460
creating
in Chrome, 444
in Firefox, 442
in Internet Explorer, 445
in Safari, 443
defined, 441
for displaying ISO Latin characters, 456–458
for doing complex calculations, 463–464
and IE security, 446
for looking up words, 451–453
for mailing webpages, 467
origin of, 443
repositioning, 443
for resetting page background, 447
for resizing pages, 468
for shortening URLs, 465
troubleshooting, 468
use of semicolons in, 441, 450
use of single vs. double quotes in, 447
for validating pages, 466
for viewing images, 454–455
vs. other JavaScript code, 441
**bookmarklets.com**, 443
**books**
*Dreamweaver: Visual QuickStart Guide*, 20, 138
*HTML and CSS: Visual QuickStart Guide*, 2, 430
*JavaScript, The Definitive Guide*, 511
**ppk on JavaScript**, 511
*Pro JavaScript Techniques*, 511
*Styling Web Pages with CSS: Visual QuickProject Guide*, 496
Boolean values, 15, 61, 63, 70, 77, 217
border properties (CSS), 499
box properties (CSS), 504
braces ({}), 25, 34, 175, 177
brackets ([ ]), 174, 177
browser compatibility, 412
browser detection, 58
browser objects, 11. See also objects
browser security settings, 129
browser windows, 128
browsers. See also specific browsers
and Ajax, 329, 348
and alert boxes, 32
and browser detection, 58
and caching, 348
and case-sensitivity, 494
and cookies, 219–220
and daylight savings time, 271
developer tools for, 473–474
and the DOM, 242
and ECMAScript, 472
and event handlers, 69
and external JavaScript files, 28
and JavaScript toolkits, 373
and JavaScript versions, 470
and jQuery versions, 366
mouse click codes for, 203
and name attribute, 119
performing word lookups in, 451–453
and pop-up windows, 127, 129
and resizing of images, 90
and rollovers, 84, 90
and security problems, 446
testing scripts in different, 130
viewing code in, 474
viewing document tree structure in, 13
and Year 2000 Problem, 277
bubbling, event, 216, 218
buttons
back, 117, 330
radio, 156–158
submit, 133, 139, 142, 209
updating with jQuery, 374–375
using check boxes as, 401–403
Buzzword Bingo game, 77–80

C
C#, 3
C/C++, 3
cache files, 28, 85, 136, 138, 338, 348
calculators, 463–464
calendar widget, 404–408
calendars
adding to webpages, 404–408
Google, 10
one-up, 404–405
two-up, 406–408
callback function, 352
calling functions, 25
caniuse.com, 496, 513
capitalizing names, 183–184
captions, slideshow, 297–300
capturing events, 216, 218
caret (^), 174, 177
Cascading Style Sheets. See CSS
case statements, 43–45
case-sensitivity, 15, 90, 494
Castro, Elizabeth, 2, 430
catch statements, 46, 47, 48, 464
CDN. See Content Delivery Network
cell() method, 54, 277
cell phones. See mobile devices; phones
characters, displaying ISO Latin, 456–458
charts, 306–314
code for drawing, 308–311
HTML page for generating, 306
script containing styles for, 307
source of statistics for, 314
check boxes, 11, 12, 151, 401–403, 410
checkers applet, 4
child frames, 117
Chrome
and browser detection, 58
creating bookmarklets in, 444, 446
developer tools, 473–474
and DOM-2, 242
and ECMAScript, 472
and external JavaScript files, 28
and JavaScript alert boxes, 32
and JavaScript toolkits, 373
and key events, 214
and mouse handling events, 202
and user prompts in new dialogs, 36
viewing document tree structure in, 13
window defaults, 130
and window events, 199
class attribute, 18–19, 69–70, 72, 74, 98
classes, CSS pseudo-, 496
clientX property, 432
clientY property, 432
client machines, reading/writing files on, 7
client-side languages, 7
client-side programs, 4, 6
Cocoa-based programs, 453
code-checking tool, 513
code-writing tool, 513
coding, for mobile vs. desktop devices, 426
color
  changing page background, 447, 448–450 properties (CSS), 506
color-picker script, 371–372
comment indicators (///, /*, and */), 29
commenting scripts, 29–30
Communicator, Netscape, 470
comparison operators, 16
_compile()_ method, 185
conditionals
  _if/then/else_, 33–34, 43
  multi-level, 43–45, 276
  _switch/case_, 43–45, 276
  use of _&&_ and _||_ in, 74
_confirm()_ method, 33–34
_constructor_ property, 185
container tags, 23
Content Delivery Network (CDN), 369, 394, 410, 422
cookies, 219–240
counting, 219, 228–230
defined, 219
deleting, 231–232
displaying “New to You” message with,
  235–240
format for typical, 221
handling multiple, 223, 233–234
how browsers handle, 219
misconceptions about, 219–220
reading, 225
setting, 6, 221–224
showing, 226–227
ways of using, 219
Coordinated Universal Time. See Greenwich Mean Time
Core JavaScript Reference/Guide, 508
cos() method, 54
countdown script, 274–277
counter programs, 230
counters, 50, 53, 228–230
country pop-up menus, 140
_createElement()_ method, 244
_createTextNode()_ method, 244
CSS (Cascading Style Sheets) and Ajax, 9, 327
background properties, 498
basic concepts, 496
border properties, 499
box properties, 504
color properties, 506
combining JavaScript and, 68–70
font properties, 500–501
generated content properties, 506
and jQuery, 366, 370, 387
list properties, 506
and object literals, 257
page properties, 496
pseudo-elements/classes, 496
purpose of, 17
recommended books on, 2, 430, 496
reference, 495–506
table properties, 500
text properties, 502–503
tools for creating, 20
units, 502
user interface properties, 497
visual effects properties, 498
visual formatting properties, 505
CSS 2.0 specification, 495
CSS 2.1 specification, 495
CSS 3 specification, 495–496
_.css_ file extension, 20
ctrlKey property, 432
curly braces (_{}_), 25, 34
cycling banners, 104–107, 113–114

D
DailyJS blog, 510
data
  accessing other people’s, 345
  automatic entry of, 356–361
  parsing, 339–345, 349–352
  presenting tabular, 49, 376–379
  refreshing, 346–348
  requesting/reading, 331–338, 349–352
  using jQuery with external, 417–419
date methods, 283–284
datepicker widget, 404–408
dates
  comparing two, 277
  displaying by time zone, 266–271
  distinguishing between weekdays/weekends, 264
dynamically displaying on webpage, 262–263
  how JavaScript stores, 277
daylight saving time, 271
day/month pop-up menus, 140–141
Debugger
  Firebug, 512
  Venkman, 508–509
debugging mobile devices, 431
decimal math, 74
deprecated attributes, 24
detection methods, 58
developer tools, browser, 473–474
DeviceMotionEvent object, 432
DeviceOrientationEvent object, 432
DHTML. See Dynamic HTML
dialing phone numbers, 440
dialogs, creating smarter, 392–395
dictionary-lookup script, 451–453
<div> tags, 18
Document Object Model. See DOM
document tree structure, 13, 243
documentation
  JavaScript, 508
  jQuery, 384
  node manipulation, 260
document.write() method, 28
Dojo, 373
dollar sign ($), 176, 177, 367–368
DOM (Document Object Model)
  and Ajax, 9, 327
defined, 13
  and nodes, 13, 241–243
  scripting, 42
  and W3C, 242, 243
  and web browsers, 242
DOM Inspector, 13
DOM-2 event handlers, 216
DOM-2, 242–243
DOM-3, 243
dot syntax, 12–13, 194
do/while loops, 64–65, 78
drag-and-drop page elements, 414–416
draggable dialogs, 393
Dreamweaver, 20, 138, 172
drop shadows, 99
Dynamic HTML, 42, 373
dynamic menus, 140–141
E
ECMA-262 specification, 472
ECMAScript
  and 4-digit years, 277
  bindings, 243
  and JavaScript versions, 470
  and Netscape, 472
  official specification for, 472
  reserved words, 492–493
  versions, 472
ejohn.org, 384
element nodes, 13, 243, 244
elements
  block-level vs. inline, 18
  CSS pseudo-, 496
  highlighting page, 386–388
  identifying, 19
  modifying, 19
  moving around on page, 433–435
else statements, 34, 43
Emacs, 20
email, sending webpages via, 467
email addresses
  validating, 166–170, 173–176
  verifying, 170
equals sign (=) assignment, 15, 16, 34
error messages, 46–48, 200
error-handling script, 46–48
escaping characters, 174
event bubbling, 216, 218
event capturing, 216, 218
event handlers, 195–218
  advanced, 216–218
defined, 14, 193
for form events, 209–212
  **onblur**, 210–211
  **onchange**, 209, 211
  **onclick**, 210
  **onfocus**, 212
  **onreset**, 209
  **onselect**, 209
  **onsubmit**, 209
importance of, 193
for key events, 213–215
  **onkeydown**, 213–215
  **onkeypress**, 215
  **onkeyup**, 215
list of common, 14
for mouse events, 201–208
  **onclick**, 208
  **oncontextmenu**, 201–203
  **ondblclick**, 208
  **onmousedown**, 201–203
  **onmousemove**, 204–206, 207
  **onmouseout**, 207
  **onmouseover**, 207, 353
  **onmouseup**, 204, 208
reference, 473–490
for window events, 194–200
  **onabort**, 199
  **onbeforeunload**, 198–199
  **onblur**, 200
  **onDOMContentLoaded**, 200
  **onerror**, 200
  **onfocus**, 200
  **onload**, 14, 195–197
  **onmove**, 199
  **onresize**, 199
  **onscroll**, 200
  **onunload**, 14, 136, 198
for XMLHttpRequest object, 333, 490
**event** property, 69
events. See also event handlers
  defined, 14
  form, 209–212
  key, 213–215
  mobile, 432
  mouse, 201–208
  window, 194–200
Excel, 306
**exec()** method, 184, 185
**exp()** method, 54
external data, using jQuery with, 417–419
external scripts, 26–28
extracting strings, 180–182

F
FaceTime app, 440
favelets, 441
favorites, 441
fields
  auto-completing, 356–361, 396–397
  checking one against another, 147–148
  identifying/marking problem, 147–148
  making them required, 142–146
  setting one with another, 159–161
  validating email addresses in, 166–170, 173–176
  validating zip codes in, 162–165
files, checking for existence of, 362–364
**finally{}** block, 48
Firebug Debugger, 512
Firebug Lite, 512
Firefox
  and alert boxes, 32
  creating bookmarklets in, 442
defaults, 258
  DOM, 28
  and external JavaScript files, 28
  and JavaScript, 470
  and JavaScript toolkits, 373
  and name attribute, 119
  non-standard window event handlers, 138
  and oncontextmenu events, 203
  and onkeydown events, 214
  and onload events, 136, 138
  and onmousedown events, 203
  and page caching, 136, 138
  performing word lookups in, 453
  user prompts in new dialogs, 36
  window defaults, 130
  and Year 2000 Problem, 277
Fireworks, 459
Flanagan, David, 511
Flash animations, 4, 104
Flickr
  displaying data from, 417–419
  popularity of, 325
  reading/parsing server data from, 339–345, 349–352
  refreshing server data from, 346–348
floor() method, 54, 112, 347
focus() method, 132, 211
font properties (CSS), 500–501
font-compatibility list, 437
fonts, 315, 318, 437
for attribute, 134
for loops, 50, 52, 53–54
form event handlers, 209–212
  onblur, 210–211
  onchange, 209, 211
  onclick, 210
  onfocus, 212
  onreset, 209
  onselect, 209
  onsubmit, 209
<form> tags, 134, 139
form validation, 133, 209
formatting
  properties (CSS), 505
  strings, 183–190
    list of names, 183–187
    phone numbers, 188–190
  time, 272–273
form-handling scripts
  changing menus dynamically, 140–141
  checking one field against another, 147–148
  creating select-and-go menu, 135–138
  identifying problem fields, 149–150
  making fields required, 142–146
  making sure user picks radio button, 156–158
  setting field value automatically, 159–161
  validating email addresses, 166–170, 173–176
  validating multi-element form, 151–155
  validating zip codes, 162–165
forms, 133–170
  auto-completing fields in, 356–361, 396–397
  checking one field against another in, 147–148
  how they work, 133
  identifying problem fields in, 149–150
  for jumping from one page to another, 139
  making fields required in, 142–146
  purpose of, 133
  and regular expressions, 171–172
  setting field values automatically in, 159–161
  tags/attributes, 134
  UI design considerations, 146
  validating email addresses in, 166–170, 173–176
  validating file names in, 178–179
  validating multi-element, 151–155
  validating URLs in, 178–179
  validating zip codes in, 162–165
forum.jquery.com, 384
forums, jQuery, 384
four-digit year, 277, 405
frames
  HTML tags/attributes, 116
  inline, 118 (See also iframes)
  keeping pages out of, 117
  reduced popularity of, 115
  setting target for, 118–119
  sharing functions between, 126–127
framesets, 117, 118
frameworks, JavaScript, 365–366, 369, 385
function values, 15
functions
  anonymous, 89, 136, 386
  calling, 25
  components of, 25
  defined, 25
  naming, 25, 494
  passing values to, 55–56
  sharing between documents, 126–127
  use of parentheses in, 28
G
  gamma property, 432
Garrett, Jesse James, 9, 327
generated content properties (CSS), 506
gelocation, 438–440
gesturechange event, 432
gestureend event, 432
Gesterevent object, 432
gesturestart event, 432
getAllResponseHeaders() method, 333
getDate() method, 283
getDay() method, 283
getElementById() method, 27
getElementsByTagname() method, 245, 246, 247
getFirebug.com, 512
getFullYear() method, 277, 283
getHours() method, 265, 283
getMilliseconds() method, 283
getMinutes() method, 283
getMonth() method, 283
getResponseHeader() method, 333
getSeconds() method, 283
time() method, 277, 283
timezoneOffset() method, 283
getUTCDate() method, 283
getUTCFullYear() method, 283
getUTCHours() method, 283
getUTCMilliseconds() method, 283
getUTCMilliseconds() method, 283
getUTCMonth() method, 283
getUTCSeconds() method, 283
getYear() method, 277, 283
Global property, 185
Global Statistics, StatCounter's, 314
Gmail, 10, 325
GMT. See Greenwich Mean Time
goo.gl, 465
Google
and Ajax, 10
Android emulator, 431
Calendar, 10
Docs, 10
Gmail, 10, 325
Instant, 361
Maps, 9, 10, 325, 440
Maps Mania, 10
googlemapsmania.blogspot.com, 10
graphics. See also images
animating, 81
displaying, 111
preparing for rollovers, 90
programs, 459
Greenwich Mean Time (GMT), 266, 283, 284
grep, 171
gs.statcounter.com, 314

H

<h1>...<h6> tags, 22
hash symbol (#), 19
<head> tags, 22, 23
header scripts, 23
highlighting new elements, 386–388
hijacking pages, 117
hit counters, 230
hover() method, 374
href attribute, 22, 440
HTML
and Ajax, 9, 327
attributes, 22, 49, 82, 116, 134
and case, 90
and CSS, 17
evolution of, 1
forms, 133
recommended book on, 2, 430
separating JavaScript from, 41, 43
tags, 22, 49, 82, 116, 134
tools for writing, 20
and W3C validation, 17
writing JavaScript-friendly, 17–19
HTML and CSS: Visual QuickStart Guide, 2, 430
.html file extension, 20
HTML Source mode, 20
<html> tags, 22
Hyslop, Bruce, 2, 430

I

id attribute
and frames, 116, 119
and images, 82, 103
manipulating cell contents with, 51
purpose of, 18, 19
identifier property, 432
IE. See Internet Explorer

if/else conditionals, 33–34, 43, 57

iframes
  creating content for, 122–123
  creating dynamic, 124–125
  defined, 118
  loading, with JavaScript, 120–121

ignoreCase property, 185

image (<img>) tags, 82, 86, 90
image rollovers, 191, 208, 362. See also rollovers

images, 81–114
  annotating, 96
  checking for alternate versions of, 362–364
  creating illusion of animation with, 85
  in cycling banners, 105
  cycling with random start, 113–114
  displaying random, 111–112
  forcing users to download, 105
  HTML tag/attributes for, 82
  preparing for rollovers, 90
  presenting as slideshows, 108–110
  for simple rollovers, 83–84
  for three-state rollovers, 91–92, 191, 192
  viewing table of, 454–455

increment step, for loop, 53

index number, 59

initDeviceMotionEvent() method, 432

initDeviceOrientationEvent() method, 432

initEvent() method, 217

initGestureEvent() method, 432

initTouchEvent() method, 432

initialization step, for loop, 53

inline elements, 18

innerHTML property, 27, 28, 42, 245, 355

input property, 185

<input> tags, 134

insertBefore() method, 253

interactivity, 1, 6, 8, 68–70, 371–375

internal scripts, 26

Internet Explorer (IE)
  and alert boxes, 32
  creating bookmarklets in, 445
  debugger, 512
  and DOM-2, 242
  and ECMAScript, 472

and event handlers, 69
and external JavaScript files, 28
and getFullYear() method, 277
and JavaScript toolkits, 373
and JScript, 471
mouse click codes, 203
and name attribute, 119
and oncontextmenu events, 203
and onkeydown events, 214
and onmousedown events, 203
and pop-up windows, 129
and rollovers, 84, 90
scripting capabilities, 5
and security, 129, 446
and tabbed browsing, 129
viewing document tree structure in, 13
window defaults, 130
and XMLHttpRequest object, 332, 338
Year 2000 Problem, 277

Internet time server, 271

interval property, 432

iOS devices, 431, 437, 440
iPad/iPhone, 80, 431, 438, 440. See also mobile devices

is.gd, 465

isNaN() method, 47

ISO Latin characters, 456–458

J

Java, 3–4, 5, 172

java.com, 3

JavaScript
  adding visual interest to webpages with, 81
  and AOL, 42
  applying styles with, 68–70
  and browser compatibility, 412
  calculator, 463–464
  case-sensitivity of, 15
  as client-side language, 7
  combining CSS and, 68–70
  and cookies, 219–220
  documentation, 508
  and the DOM, 13, 243
  enhancing links with, 39–41
  evolution of, 1
  frameworks, 365–366, 369, 385
hiding from users, 28, 201–203
how events are handled in, 14
inventor of, 1
limitations of, 7
loading iframes with, 120–121
Math object, 54, 463–464, 481
Microsoft version of, 5, 42
modifying document tree structure with, 13
month numbering in, 277
and Netscape, 1, 5, 42, 470
object table, 473–490
as object-oriented language, 11
operators, 15–16
as programming language, 2
purpose of, 1, 17
recommended books on, 511
and regular expressions, 171, 172
reserved words, 491–494
resources, 507–514
rewriting with object literals, 257–260
as scripting language, 2
terminology, 42
toolkits, 8, 365–366, 373
tools for writing, 20, 365–366
use of semicolons in, 24
using functions in, 25
value types, 15
versions, 470
vs. Java, 3
ways of using, 6
JavaScript, The Definitive Guide, 511
JavaScript Center, 508
JavaScript Guide, Netscape, 443
JavaScript Object Notation, 260
javascriptworld.com, 2, 507
Jobs, Steve, 80
jordanm.co.uk, 437
JPEG images, 105
jQuery, 365–424
adding to page, 367–368
adding user interaction with, 371–375
and Ajax, 412–413
alternatives to, 373
auto-completing fields with, 396–397
and browser compatibility, 412
calendar widget, 404–408
CDN, 369, 394, 410, 422
and CSS, 366, 370, 387
designing with
  adding calendar to page, 404–408
  adding sortable tabs, 398–400
  creating accordion menus, 389–391
  creating custom themes, 409–410
  creating smarter dialogs, 392–395
  highlighting new elements, 386–388
  using check boxes as buttons, 401–403
documentation, 384
and dollar sign ($), 367–368
downloading, 384
as foundation, 411, 412–413
and JSON, 411, 412–413
plugins, 384, 411, 413, 420–424
purpose of, 8, 366
resources, 384
serving, 369
sorting tables with, 380–383
strengths of, 366, 373, 412
striping tables with, 376–379
support for older browsers, 366
themes, 391, 394–395, 409–410
tutorials, 384
updating buttons with, 374–375
updating page with, 370
user interface (See jQuery UI)
using with external data, 417–419
versions, 366, 368, 369
and “yellow fade”, 385, 386
jQuery Mobile, 384
jQuery UI
  adding to pages, 386–388
  avoiding overhead of, 388
CSS files, 394
plugins, 422
purpose of, 385
resources, 384
themes, 389, 392, 394–395
versions, 394
.js file extension, 20, 26, 203
JSBin, 514
JScript, 5, 42, 471, 509
jscript.dll, 471
JSFiddle, 514
JSHint, 513
JSON format, 10, 260, 349–352, 411, 412–413
jsperf.com, 366
jump menus, 138

K
Kangas, Steve, 443
key event handlers, 213–216
keywords
  this, 41, 260
  var, 35, 36
kilometers-to-miles converter, 461–462
Koch, Peter-Paul, 510, 511

L
<label> tags, 134
landscape orientation, 426, 427
language attribute, 24
languages
  client-side, 7
  object-oriented, 11
  scripting, 2, 5
lastIndex property, 185
lastMatch property, 185
lastParen property, 185
Latin characters, ISO, 456–458
latitude attribute, 438, 439, 440
layers, hiding/displaying, 278–280
leap year, 141
left-click codes, 203
leftContext property, 185
length units (CSS), 502
<li> tags, 288
light-table script, 414–418
limiting step, for loop, 53
link enhancement script, 39–40
links. See also URLs
  adding to cycling banners, 106–107
  changing single rollover from multiple, 96–98
  enhancing with JavaScript, 39–41
mailto, 440, 467
  previewing, 95, 353–356
  redirecting users with, 37–38
  triggering rollovers from, 93–98
Linux, 3
list properties (CSS), 506
lists, 288
literal values, 16
LiveScript, 5, 470
log() method, 54
longitude attribute, 438, 439, 440
loops
  counters for, 50, 53
  creating Bingo card with, 50–54
  creating table’s contents with, 53–54
  importance of, 50
  specific types
    do/while, 64–65, 78
    for loops, 50, 52, 53–54
M
Mac OS X. See OS X
Macintosh
  browsers, 373
  and JScript, 471
  mouse click codes, 203
  toolkits, 373
Macworld Expo, 80
Mail app, 440
mailing webpages, 467
mailto links, 440, 467
map apps, 438, 440. See also Google Maps
mashups, 10
match() method, 185
math, binary vs. decimal, 74
Math object
  and bookmarklet calculator, 463–464
  methods/properties, 54, 481
max() method, 54
maxlength attribute, 134
@media queries, 430
menus
  accessibility considerations, 296
  accordion, 389–391, 398
  changing dynamically, 140–141
  horizontal vs. vertical, 293
  jump, 138
  outline-style, 285
  pop-up, 140–141
  pull-down, 289–296
Index

Index

select-and-go, 135–138
sliding, 286–288
sortable tabs in, 398–400
metaKey property, 432
meta characters, 176
methods
  combining with objects/properties, 12–13
defined, 12
for DeviceMotionEvent object, 432
for DeviceOrientationEvent object, 432
distinguishing from properties, 12
for GestureEvent object, 432
for Math object, 54, 481
reference, 473–490
for RegExp object, 185
for strings, 185
for TouchEvent object, 432
use of parentheses in, 12
for XMLHttpRequest object, 333, 490
Microsoft
  and ECMAScript, 472
Excel (See Excel)
Internet Explorer (See Internet Explorer)
  and Java, 3
  and JScript, 5, 42, 471
JScript Language site, 509
Windows (See Windows)
Word (See Word)
min() method, 54
mobile apps, launching, 440
mobile devices, 425–440
  changing orientation, 426–430
debugging, 431
differentiating, 436–437
  font considerations, 437
  handling touch events, 433–435
  help building websites for, 384
  launching apps for, 440
  locating, 438–440
popularity of, 425
mobile events, 432
modal dialogs, 392–393
modifiers, regular expression, 177
month/day pop-up menus, 140–141
MooTools, 373
mouse click codes, 203
mouse event handlers, 201–208
onclick, 208
oncontextmenu, 201–203
ondblclick, 2070
onmousedown, 201–203
onmousemove, 204–206, 207
onmouseout, 207, 375
onmouseover, 207, 353
onmouseup, 204, 208
Mozilla, 58, 470, 508–509
Mozilla Hacks blog, 509
MSIE. See Internet Explorer
multi-level conditionals, 43–45, 276
multiline property, 185
My Yahoo, 10

N
name attribute, 116, 119, 134
navigation menus, 135
Navigator, Netscape
  and ECMAScript, 472
  and JavaScript, 1, 5, 42, 470
  and LiveScript, 5, 470
  and Year 2000 Problem, 277
nested if statements, 43
Netscape
  Communicator (See Communicator)
  and external JavaScript files, 28
JavaScript Guide, 443
Navigator (See Navigator)
  and rollovers, 84, 90
node manipulation, 241, 242–243, 260
nodes, 241–260
  adding, 244–245
  defined, 13
deleting, 246–250
and the DOM, 13, 241–243
inserting, 251–253
replacing, 254–256
types of, 13, 243
vs. innerHTML, 245
non-breaking space (&nbsp;), 51
<noscript> tags, 32
Notepad, 20
null values, 15, 35
number sign (#), 19
numbers
   random, 54, 347
   validating, 189–190
numeric values, 15

O
object detection, 57–58
object literals, 257–260
   sample scripts, 258–259, 371, 375
   similarity to CSS, 257
   use of this with, 260
   vs. standard procedural JavaScript, 257, 260
object table, 473–490
<object> tags, 4
object values, 15
object-based languages, 11
object-oriented languages, 11
objects
   combining with properties/methods, 12–13
   defined, 11
   detecting, 57–58
   displaying/hiding, 280
   methods of, 12
   moving, 281–282
   naming, 11
   properties of, 12
   reference, 473–490
offline resources, 511
onabort events, 14, 199
onbeforeunload events, 198–199
onblur events, 14, 200, 210–211
onchange events, 14, 209, 211
onclick events, 14, 38, 208, 210
oncontextmenu events, 201–203
ondblclick events, 208
onDOMContentLoaded events, 200
onerror events, 14, 200
one-up calendars, 404–405
onfocus events, 14, 200, 212
onkeydown events, 213–215
onkeypress events, 215
onkeyup events, 215
online pastebins, 514
online resources, 508–510, 512–514
onload events, 14, 195–197, 333
onloadend events, 333
onloadstart events, 333
onmousedown events, 201–203
onmousemove events, 204–206, 207
onmouseout events, 14, 90, 207
onmouseover events, 14, 90, 207, 353
onmouseup events, 204, 208
onmove events, 199
onpagehide events, 138
onpageshow events, 138
onreadystatechange events, 333
onreset events, 209
onresize events, 199
onscroll events, 200
onselect events, 14, 209
onsubmit events, 14, 209
ontimeout events, 333
onunload events, 14, 136, 198
open() method, 128, 130, 333
Opera, 58, 512
operators, 15–16, 70, 171
<option> tags, 134
or (|) comparison, 16, 70, 74, 76, 177
orientationChange event, 432
orientation changes, 426–430
OS X
   alert boxes, 32
   and daylight saving time, 271
dictionary/thesaurus window, 453
text editors, 20
outline-style menus, 285
overrideMimeType() method, 333

P
pageX property, 432
pageY property, 432
paragraphs, 245
parameters, passing, 35, 55
parentheses. See ( ) (parentheses)
parse() method, 269, 283
parseInt() method, 228, 283
passing information, 55–56
password-checking script, 142, 147–148
paste bins, 514
period (.), 12, 19, 177
Perl, 7, 172
phone numbers, formatting/validating, 188–190
phones. See also mobile devices
  app for dialing, 440
differentiating between, 436–437
  font considerations, 437
  free simulators, 431
  handling orientation changes, 426–430
  locating, 438–440
Photoshop, 306, 459
PHP, 7, 172
plugins, jQuery, 384, 411, 413, 420–424
plugins.jquery.com, 384, 413
plus sign (+), 15, 53, 174, 177
PNG images, 105
pop-up killers, 127
pop-up menus, 140–141
pop-up windows, 127, 195, 198, 278
portrait orientation, 426, 427
postal codes, validating, 162–165
pow() method, 54
ppk on JavaScript, 511
preventDefault() method, 217
Pro JavaScript Techniques, 511
programming languages, 2, 3, 172
progressive enhancement, 42
prompt() method, 35
properties
  combining with objects/methods, 12–13
defined, 12
  for DeviceMotionEvent object, 432
  for DeviceOrientationEvent object, 432
distinguishing from methods, 12
  for GestureEvent object, 432
reference
  CSS, 496–506
  JavaScript, 473–490
  for RegExp object, 185
touch, 432
  for TouchEvent object, 432
  for XMLHttpRequest object, 333, 490
Prototype, 373
pseudo-classes (CSS), 496
pseudo-elements (CSS), 496
pull-down menus, 289–296
Python, 172
Q
question mark (?), 34, 175, 177
QuirksMode blog, 510
quotes. See " (quotes)
R
radio buttons, 156–158
random images, 111–114
random() method, 54, 112, 347
random numbers, 54, 127, 347
ready() method, 368
readyState property, 333, 334
redirection, 21, 37–38
RegExp object, 171, 185, 484
regular expressions, 171–192
  alternate names for, 171
defined, 171
  extracting strings with, 180–182
  formatting strings with, 183–190
  modifiers for, 177
  purpose of, 171
  replacing elements with, 191–192
  sorting strings with, 186–187
  special characters for, 177
  validating email addresses with, 173–176
  validating file names with, 178–179
  validating strings with, 188–190
  validating URLs with, 178–179
removeEventListener() method, 217
replace() method, 117, 185
replaceChild() method, 254
reserved words, 491–494
Resig, John, 384, 511
resizable dialogs, 393
resizeTo() method, 468
resizing windows, 468
resources
books
CSS, 2, 430, 496
Dreamweaver, 138
HTML, 2, 430
JavaScript, 511
@media queries, 430
websites
Android SDK, 431
Bare Bones Software, 20
bit.ly, 465
bookmarklets.com, 443
caniuse.com, 496, 513
dailyjs.com, 510
Dojo, 373
ECMA International, 472
Firebug Debugger, 512
font-compatibility list, 437
Google Maps Mania, 10
is.gd, 465
java.com, 3
JavaScript Center, 508
javascriptworld.com, 2, 507
jQuery, 384
jQuery Mobile, 384
jQuery plugins, 413
JSBin, 514
JScript Language, 509
JSFiddle, 514
JSHint, 513
jsperf.com, 366
Modernizr, 373
MooTools, 373
Mozilla Hacks, 509
Prototype, 373
QuirksMode, 510
Resig, John (ejohn.org), 384
slidesjs.com, 422
StatCounter’s Global Statistics, 314
Surfin’ Safari, 509
tablesorter.com, 383
tinyURL.com, 465
toolkits, 373
URL-shortening services, 465
Venkman Debugger, 508–509
W3C validation tool, 17, 466
Web Standards Project, 42
Wikipedia, 373
Willison, Simon (simonwillison.net), 197
Xcode developer tools, 431
YUI, 373
response property, 333
responseText property, 333, 335
responseXML property, 333, 335
right-click codes, 203
rightContext property, 185
rollovers, 83–103
browser considerations, 84, 90
building three-state, 91–92, 191
checking whether image exists, 362–364
defined, 6, 81
preparing images for, 90
rotation property, 432
rotationRate property, 432
round() method, 54
RSS feeds, 340
S
Safari
and alert boxes, 32
blog, 509
and browser detection, 58
creating bookmarklets in, 443
developer, 512
and debugging mobile devices, 431
development tools, 431
and DOM-2, 242
and ECMAScript, 472
and external JavaScript files, 28
and iPhone, 80
iPhone/iPad Simulator, 431
and JavaScript toolkits, 373
and onkeydown events, 214
and onload events, 136, 138
and page caching, 136, 138
performing word lookups in, 453
viewing document tree structure in, 13
Web Inspector, 431
window defaults, 130
sample scripts
Ajax
  auto-completing fields, 356–361
  checking whether file exists, 362–364
  parsing server data, 339–345, 349–352
  previewing links, 353–356
  refreshing server data, 346–348
  requesting/reading server data, 331–338, 349–352
bookmarklets
  changing page’s styles, 448–450
  converting kilometers to miles, 461–462
  converting RGB values to hex, 459–460
  creating in Chrome, 444
  creating in Firefox, 442
  creating in Internet Explorer, 445
  creating in Safari, 443
  displaying ISO Latin characters, 456–458
  looking up words, 451–453
  mailing webpages, 467
  resetting page background, 447
  resizing pages, 468
  shortening URLs, 465
  using JavaScript calculator, 463–464
  validating pages, 466
  viewing images, 454–455
cookies
  counting cookies, 228–230
  deleting cookies, 231–232
  displaying “New to You” message, 235–240
  handling multiple cookies, 233–234
  reading cookies, 225
  setting cookies, 221–224
  showing cookies, 226–227
cycling banners
  adding links, 106–107
  creating, 104–105
dynamic pages
  converting 24-hour to 12-hour time, 272–273
  creating countdown, 274–277
  customizing message for time of day, 265
  displaying dates by time zone, 266–271
  hiding/displaying layers, 279–280
  identifying weekday vs. weekend, 264
  moving objects, 281
  putting current date on webpage, 262–263
  event handlers
  checking for double clicks with ondblclick, 208
  hiding code with onmousedown, 201–203
  preventing wayward field entries with onfocus, 212
  setting multiple onload attributes, 194–197
  triggering slide change with addEventListener(), 216
  triggering slide change with onkeydown, 215
  using onblur to force field entry, 210–211
  using onblur to trigger action when user leaves field, 211
form handling
  changing menus dynamically, 140–141
  checking one field against another, 147–148
  creating select-and-go menu, 135–138
  identifying problem fields, 149–150
  making fields required, 142–146
  making sure user picks radio button, 156–158
  setting field value automatically, 159–161
  validating email addresses, 166–170, 173–176
  validating multi-element form, 152–155
  validating zip codes, 162–165
frames
  creating content for iframes, 122–123
  keeping pages out of frames, 117
  loading iframes with JavaScript, 120–121
  setting target for frames, 118–119
images
  displaying as slideshow, 108–110
  displaying random, 111–112
  rollover, making multiple links change single, 96–98
  rollovers, building three-state, 91–92
  rollovers, creating simple, 83–84
JavaScript applied
  adding pull-down menus, 289–292
  allowing user to switch between style sheets, 315–324
  generating bar graph, 306–314
  using sliding menus, 286–288
sample scripts (continued)
JavaScript basics
alerting users, 31–32
commenting scripts, 29–30
confirming user choice, 33–34
enclosing script in `<script>` and `</script>` tags, 23–24
enhancing links, 39–40
handling errors, 46–48
prompting users, 35–36
redirecting users with link, 37–38
referencing external JavaScript files, 26–28
using conditionals, 33–34, 43–45
JavaScript language essentials
applying styles with JavaScript, 68–70
calling scripts multiple ways, 66–67
checking states, 71–74
detecting objects, 57–58
passing values to functions, 55–56
returning values from functions, 61–62
updating arrays, 62–63
using arrays, 59–60
using `do/while` loops, 64–65, 78
using `for` loops, 50–54
using string arrays, 77–80
jQuery
adding calendar to page, 404–408
adding jQuery to page, 367
adding sortable tabs, 398–400
adding user interaction/updates, 371–373
audio-player plugin, 423–424
auto-completing fields, 396–397
creating accordion menus, 389–391
creating sortable tables, 380–383
dragging/dropping elements, 414–416
highlighting new elements, 386–388
slideshow plugin, 420–422
striping tables, 376–379
updating page, 370
using check boxes as buttons, 401–403
mobile devices
changing orientation, 426–430
differentiating devices, 436–437
handling touch events, 433–435
locating device, 438–440
objects and the DOM
adding text nodes, 244–245
deleting text nodes, 246–250
inserting nodes, 251–253
replacing nodes, 254–256
using object literals, 258–260
regular expressions
capitalizing names, 183–184
extracting strings, 180–182
formatting strings, 183–184
formatting/sorting strings, 186–187
formatting/validating strings, 188–190
replacing page elements, 191–192
sorting names, 186–187
validating email addresses, 173–176
validating file names, 178–179
validating phone numbers, 188–190
validating URLs, 178–179
windows
loading different contents into, 131–132
opening new, 128–129
sans-serif fonts, 315, 318
`scale` property, 432
scope, variable, 36, 450
`screenX` property, 432
`screenY` property, 432
screen size, 468
script errors, 129
`<script>` tags, 2, 23, 24, 117
scripting, unobtrusive, 41, 42
scripting languages, 2, 5
scripts. See also sample scripts
allowing users to run, 66–67
anticipating user actions in, 14
calling functions in, 25
defined, 2
how web browsers handle, 2
internal vs. external, 26
putting comments in, 29–30
testing, 130
triggering when page loads, 14
using external, 26–28
where to put, 23
writing your first, 23
`search()` method, 185
search-and-replace feature, 50
Searles, Nathan, 422
security problems, Internet Explorer, 446
security settings, browser, 129
<select> tags, 134
select-and-go navigation, 135–138
selected attribute, 134
semantic chunks, breaking content into, 18
semicolon (;), 23–24, 441, 450
send() method, 333
serif fonts, 315, 318
server data
  parsing, 339–345, 349–352
  refreshing, 346–348
  requesting/reading, 331–338, 349–352
server machines, writing files on, 7
server-side programs, 6, 241, 330
server-side scripts, 133, 139, 148
server-side technologies, 330, 361
setDate() method, 284
setFullYear() method, 284
setHours() method, 284
setMilliseconds() method, 284
setMinutes() method, 284
setMonth() method, 284
setRequestHeader() method, 333
setSeconds() method, 284
setTimeout() method, 197, 346
setUTCDate() method, 284
setUTCFullYear() method, 284
setUTCFullYear() method, 284
setUTCHours() method, 284
setUTCMilliseconds() method, 284
setUTCMinutes() method, 284
setUTCMonth() method, 284
setUTCSeconds() method, 284
setYear() method, 284
shiftKey property, 432
simonwillison.net, 197
sin() method, 54
size attribute, 134
slash (/), 15, 173
slideshows
  building wraparound, 108–110
  plugin, 420–422
  showing captions in, 297–300
  triggering slide changes in, 215, 216
SlidesJS plugin, 420–422
sliding menus, 286–288
smartphones. See mobile devices
SMS app, 440
sortable tabs, 398–400
sorting tables, 380–383
source property, 185
<span> tags, 18
special characters, 174, 177, 456–458
split() method, 185, 222, 223
sqrt() method, 46–47, 54
square root calculator, 46–48
src attribute, 22, 26, 82, 116
srcElement property, 69
standards
  ECMAscript, 472
  web, 17, 373, 466
StatCounter’s Global Statistics, 314
state names, auto-completing, 356–397
status property, 333
statusText property, 333
SteveNote Bingo, 80
stopPropagation() method, 217
string arrays, 77–80
string methods, 180, 185
strings, 180–190
  comparing, 16
  defined, 15
  returning, from functions, 61
  use of quotes with, 15, 59
striping tables, 376–379
style attribute, 70
style sheet switcher, 315–324
styles
  applying, with JavaScript, 68–70
  changing, with bookmarklet, 448–450
Styling Web Pages with CSS: Visual QuickProject Guide, 496
submit buttons, 133, 139, 142, 212
substring() method, 237, 240
Sun Microsystems, 3
Surfin’ Safari blog, 509
switch/case statements, 43–45, 276
syntax, dot, 12–13, 194
syntax errors, 129

t

tabbed browsing settings, 129
table properties (CSS), 500
<table> tags, 49
tables
  sorting, 380–383
  striping, 376–379
  viewing page images in, 454–455
tablesorter plugin, 383
tablets, 425, 431. See also mobile devices
tabs, sortable, 398–400	tabular data, 49, 376. See also tables
tagName, 87, 90
tags
  basic, 22
  form, 134
  frame, 116
  image, 82
  table, 49
tan() method, 54
target attribute, 118–119
target property, 432
<td> tags, 49
terminology
  DOM 2, 243
  JavaScript, 42
  node manipulation, 242–243
test() method, 185
testing scripts, 130
text editors, 20
text nodes, 13, 243, 244–250
text properties (CSS), 502–503
text property, 336
textContent property, 336
TextMate, 172
TextWrangler, 20
<th> tags, 49
ThemeRoller, 409–410
themes
  creating custom, 409–410
  jQuery UI's built-in, 391, 394–395
this keyword, 41, 260
three-state rollovers, 91–92, 191
throw statements, 46, 47, 464
Thunderbird, 509
time
  adding AM/PM to, 271, 273
  converting 24-hour to 12-hour, 272–273
  customizing messages for, 265
  dealing with daylight saving time, 271
  JavaScript's inconsistent handling of, 263
setTimeout property, 333
time server, 271
time zone, displaying dates by, 266–271
tinyURL.com, 465
<title> tags, 22
toggle() method, 388
toolkits, JavaScript, 8, 365–366, 373
toGMTString() method, 284
toLocaleString() method, 284
toSource() method, 185
toString() method, 185, 284
toUTCString() method, 284
touchcancel event, 432
touchend event, 432
touchmove event, 432
touchstart event, 432
touch events, 433–435
TouchEvent object, 432
<tr> tags, 49
tree structure, 13, 243
ttrue/false values, 15, 57, 70
try statements, 46, 47, 463, 464
tutorials, jQuery, 384
Twitter, 465	two-digit year, 277, 405	two-up calendars, 406–408	type attribute, 24, 134

U
UI, jQuery. See jQuery UI
<ul> tags, 288
units (CSS), 502
Universal Time (UT). See Greenwich Mean Time
Unix, 3, 20
unobtrusive scripting, 41, 42
unordered lists, 288, 398
upload property, 333
URLs. See also links
    shortening, 465
    updates to this book’s, 507
validating, 178–179
user interface, jQuery. See jQuery UI
user interface properties (CSS), 497
users
    alerting, 31–32
    allowing control of scripts by, 66–67
    confirming choices of, 33–34
    prompting for response, 35–36
    redirecting with links, 37–38
UT (Universal Time). See Greenwich Mean Time
UTC() method, 284
UTC (Coordinated Universal Time).
    See Greenwich Mean Time
V
validating
    email addresses, 166–170, 173–176
    file names, 178–179
    forms, 151–155, 209
    JavaScript, 512
    phone numbers, 188–190
    strings, 171, 188–190
    URLs, 178–179
    webpages, 17, 466
    zip codes, 162–165
validator.w3.org, 17, 466
value attribute, 134
valueOf() method, 185, 284
values
    adding, 15
    assigning to variables, 16
    binary, 70, 71
    checking variables against multiple, 43
    comparing, 16, 70
    literal, 16
    passing to functions, 55–56
    types of, 15
var keyword, 35, 36
variables
    assigning values to, 16
    checking against multiple values, 43
    comparing values of, 16
    declaring, 35
    defined, 15
    defining scope of, 36, 450
    naming, 15, 182, 494
    use of equals sign with, 15
Venkman Debugger, 508–509
verifying email addresses, 170
Vista, 446
visual effects properties (CSS), 498
visual formatting properties (CSS), 505
visually-impaired users, 296
void() method, 447, 450
W
W3C
    and CSS 3 properties, 495
    deprecation of attributes by, 24
    and DOM scripting, 42
    and DOM-2, 242
    and DOM-3, 243
    and innerHTML property, 28
    and node manipulation, 241, 242, 243
    validation tool, 17, 466
web
    browsers (See browsers)
    dynamic nature of, 1, 325
    sites (See websites)
    standard layout language for, 17
    standards, 17, 42, 373, 466
Web 2.0, 328, 342, 385
Web Inspector, Safari, 431
Web Standards Project, 42
web-based applications
    and Ajax, 9, 10
    and JavaScript, 6
    and jQuery, 413
web-based email, 10
web-based slideshows, 108–110, 414–416
webkitCompassAccuracy property, 432
webkitCompassHeading property, 432
WebKit, 472, 509
weblogs. See blogs
websites
debugging, 431
for specific companies/topics (See resources)
while statements, 64–65
width attribute, 82
Wikipedia, 373
Willison, Simon, 197
window event handlers, 194–200
onabort, 199
onbeforeunload, 198–199
onblur, 200
onDOMContentLoaded, 200
onerror, 200
onfocus, 200
onload, 14, 195–197
onmove, 199
onresize, 199
onscroll, 200
onunload, 14, 136, 198
Windows
browsers, 373
and Java, 3
and JScript, 471
Phone, 437
text editor, 20
Vista (See Vista)
XP Service Pack 2, 446
windows, 125–132
adding parameters to, 130
alert, 31–32
closing, 7
elements of standard browser, 128
how JavaScript deals with, 115
importance of, 115
loading different contents into, 131–132
opening new, 128–129
pop-up, 195, 198
sharing functions between, 126–127
withCredentials property, 333
Word, Microsoft, 20, 172
WYSIWYG editors, 20, 41, 138

X
Xcode developer tools, 431
XML
and Ajax, 9, 10, 327
benefits of using, 417
file request example, 331–338
forcing call to return, 338
reading/storing, 340, 348
vs. JSON, 351
XMLHttpRequest object, 331–338
and Ajax, 10, 327
event handlers, 333, 490
and Internet Explorer, 332, 338
methods, 333, 490
properties, 333, 490
purpose of, 10, 327
retrieving/displaying server data with, 413

Y
Yahoo, 10, 373
Yahoo Mail, 10
Year 2000 Problem, 277
yellow fade, 385, 386
YouTube app, 440
YUI, 373

Z
zebra-striped tables, 376–379
z-index, 278
zip codes, 162–165
zooming in/out, on maps, 440