

VISUAL **QUICKSTART** GUIDE

Get up and running in no time!



**Register your book
for free extras!**
www.peachpit.com/register

Dozens of downloadable scripts!

JavaScript

Eighth Edition

TOM NEGRINO • DORI SMITH

© LEARN THE QUICK AND EASY WAY!

VISUAL QUICKSTART GUIDE

JavaScript

EIGHTH EDITION

TOM NEGRINO • DORI SMITH

Visual QuickStart Guide

JavaScript, Eighth Edition

Tom Negrino and Dori Smith

Peachpit Press
1249 Eighth Street
Berkeley, CA 94710
(510) 524-2178
(510) 524-2221 (fax)

Find us on the Web at www.peachpit.com
To report errors, send a note to errata@peachpit.com
Peachpit Press is a division of Pearson Education

Copyright © 2012 by Tom Negrino and Dori Smith

Editor: Nancy Peterson
Production Editor: Tracey Croom
Copyeditor: Scout Festa
Compositor: Danielle Foster
Indexer: Emily Glossbrenner
Cover Design: RHDG / Riezebos Holzbaur Design Group, Peachpit Press
Interior Design: Peachpit Press
Logo Design: MINE™ www.minesf.com

Notice of rights

All rights reserved. No part of this book may be reproduced or transmitted in any form by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. For information on getting permission for reprints and excerpts, contact permissions@peachpit.com

Notice of liability

The information in this book is distributed on an “As is” basis, without warranty. While every precaution has been taken in the preparation of the book, neither the authors nor Peachpit Press, shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the instructions contained in this book or by the computer software and hardware products described in it.

Trademarks

Visual QuickStart Guide is a registered trademark of Peachpit Press, a division of Pearson Education. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Peachpit Press was aware of a trademark claim, the designations appear as requested by the owner of the trademark. All other product names and services identified throughout this book are used in editorial fashion only and for the benefit of such companies with no intention of infringement of the trademark. No such use, or the use of any trade name, is intended to convey endorsement or other affiliation with this book.

ISBN 13: 978-0-321-77297-8

ISBN 10: 0-321-77297-0

0 9 8 7 6 5 4 3 2 1

Printed in the United States of America

Dedication

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

Special Thanks to:

Big thanks to our editor Nancy Peterson; her expert touch, serenity, and compassion made this edition a pleasant one to create.

Thanks to Tracey Croom for her excellent production work and to Scout Festa for her skillful copyediting.

Our heartfelt thanks to Danielle Foster, the book's compositor, 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.

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.

Between the time we signed the contract for this book and when we began working on it, Dori was offered her dream job at Stack Exchange (<http://www.stackexchange.com>). She could not have worked two jobs *and* still maintained a semblance of sanity without a great deal of help from others, for which she is truly grateful. In particular:

- Thanks to all the great folks at Stack: Joel Spolsky, Jeff Atwood, Robert Cartaino, Rebecca Chernoff, and Josh Heyer, among others—who gave their time and patience to help me stay employed while finishing this book.
- Thanks also to the amazingly patient women at Peachpit—I've worked with you for 14 years, and I'm still in awe of what you're able to create.
- And in particular, I must thank my co-author, Tom Negrino, who did much more than his share of the load on this edition, and who postponed his (and my) tenth wedding anniversary celebration until the book was complete.

Contents at a Glance

	Introduction	xv
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	195
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	363
Chapter 15	Designing with jQuery	377
Chapter 16	Building on jQuery	397
Chapter 17	Bookmarklets	425
Appendix A	JavaScript Genealogy and Reference	453
Appendix B	JavaScript Reserved Words	477
Appendix C	Cascading Style Sheets Reference	481
Appendix D	Where to Learn More	489
	Index	497

This page intentionally left blank

Table of Contents

	Introduction.	xv
Chapter 1	Getting Acquainted with JavaScript.	1
	What JavaScript Is.	2
	JavaScript Isn't Java.	3
	Where JavaScript Came From	5
	What JavaScript Can Do	6
	What JavaScript Can't Do.	7
	What Is Ajax?.	8
	The Snap-Together Language	11
	Handling Events	14
	Values and Variables	15
	Writing JavaScript-Friendly HTML	17
	What Tools to Use?	20
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
	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
	Using Do/While Loops	64
	Calling Scripts Multiple Ways	66
	Combining JavaScript and CSS	68
	Checking State.	71
	Working with String Arrays	77
Chapter 4	Working with Images.	81
	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
Chapter 5	Windows and Frames	115
	Keeping a Page out of a Frame	117
	Setting a Target	118
	Loading Iframes with JavaScript.	120
	Working with Iframes	121

	Creating Dynamic Iframes	123
	Sharing Functions between Documents	125
	Opening a New Window	127
	Loading Different Contents into a Window	131
Chapter 6	Form Handling	133
	Select-and-Go Navigation	135
	Changing Menus Dynamically	140
	Making Fields Required.	142
	Checking Fields against Each Other	147
	Identifying Problem Fields	149
	Putting Form Validation into Action	151
	Working with Radio Buttons	156
	Setting One Field with Another	159
	Validating Zip Codes	162
	Validating Email Addresses	166
Chapter 7	Forms and Regular Expressions.	171
	Validating an Email Address with Regular Expressions	173
	Validating a File Name	178
	Extracting Strings	180
	Formatting Strings.	183
	Formatting and Sorting Strings	187
	Formatting and Validating Strings	189
	Replacing Elements using Regular Expressions	192
Chapter 8	Handling Events	195
	Handling Window Events.	196
	Mouse Event Handling	204
	Form Event Handling	212
	Key Event Handling	216

Chapter 9	JavaScript and Cookies	219
	Baking Your First Cookie	221
	Reading a Cookie	225
	Showing Your Cookies	226
	Using Cookies as Counters	228
	Deleting Cookies	231
	Handling Multiple Cookies	233
	Displaying “New to You” Messages	235
Chapter 10	Objects and the DOM	241
	About Node Manipulation	242
	Adding Nodes	244
	Deleting Nodes	246
	Deleting Specific Nodes	248
	Inserting Nodes	251
	Replacing Nodes	254
	Writing Code with Object Literals	257
Chapter 11	Making Your Pages Dynamic.	261
	Putting the Current Date into a Web Page	262
	Working with Days.	264
	Customizing a Message for the Time of Day	265
	Displaying Dates by Time Zone	266
	Converting 24-Hour Time to 12-Hour Time.	272
	Creating a Countdown	274
	Hiding and Displaying Layers	278
	Moving an Object in the Document	281
	Date Methods	283

Chapter 12	Applied JavaScript	285
	Using Sliding Menus	286
	Adding Pull-Down Menus.	289
	Enhancing Pull-Down Menus.	293
	A Slideshow with Captions.	297
	A Silly Name Generator.	301
	A Bar Graph Generator	306
	Style Sheet Switcher	315
Chapter 13	Introducing Ajax.	325
	Ajax: Pinning It Down	327
	Reading Server Data	331
	Parsing Server Data	339
	Refreshing Server Data	346
	Getting Data From a Server	349
	Previewing Links with Ajax	353
	Auto-Completing Form Fields	357
Chapter 14	Toolkits, Frameworks, and Libraries	363
	Adding jQuery	365
	Updating a Page with jQuery	368
	Interacting with jQuery	369
	Interacting and Updating.	371
	Auto-Completing Fields	374
Chapter 15	Designing with jQuery.	377
	Highlighting New Elements	378
	Creating Accordion Menus.	382
	Creating Smarter Dialogs.	386
	Striping Tables.	390
	Sorting Tables	393

Chapter 16	Building on jQuery	397
	Using jQuery as a Foundation	398
	Using ThemeRoller to Customize Your Look	400
	Adding a Calendar to Your Page.	403
	Dragging and Dropping Elements.	408
	Using jQuery with External Data.	411
	Using jQuery Plugins	420
Chapter 17	Bookmarklets.	425
	Your First Bookmarklet	426
	Resetting a Web Page’s Background	432
	Changing a Page’s Styles.	433
	Word Lookups	436
	Viewing Images	439
	Displaying ISO Latin Characters	441
	Converting RGB Values to Hex	444
	Converting Values.	446
	A Bookmarklet Calculator	447
	Shortening URLs.	449
	Validating Pages.	450
	Mailing Pages	451
	Resizing Pages.	452
Appendix A	JavaScript Genealogy and Reference	453
	JavaScript Versions	454
	ECMAScript	457
	Object Flowchart	459
	The Big Object Table	465

Appendix B	JavaScript Reserved Words	477
Appendix C	Cascading Style Sheets Reference	481
Appendix D	Where to Learn More	489
	Finding Help Online	490
	Offline Resources	493
	Troubleshooting Tips	494
	Index	497

This page intentionally left blank

Introduction

Welcome to JavaScript! Using this easy-to-learn programming language, you'll be able to add pizzazz to your Web pages 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 Web sites, 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 Web pages down, and

that you have some familiarity with common HTML, such as links, images, and forms.

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 Web pages 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, the color section in the back of the book). There are several other books on the market that do that job admirably, and we list them at the end of this book, in Appendix D. 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 an introduction to Ajax, a technique that uses JavaScript and other common Web technologies to add extra interactivity to Web pages, and to improve the user experience of your Web sites. We covered the basics of Ajax and added some practical examples to allow you to Ajax-ify your sites without getting an advanced degree in Web programming. In this edition, we've added even more examples and techniques, using the jQuery *framework*—which you can think of as building blocks that allow you to easily add useful features to your sites.

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 or JavaScript code, like this:

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

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 in-progress 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.

```
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 7 or later; Firefox 1.0 or later; 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 (both Vista and Windows 7; like Microsoft, we've dropped support for Windows XP), Mac OS X (10.4.11 and later), and Ubuntu Linux (we only tested scripts in Firefox, Ubuntu's default browser).

We used the 600-pound gorilla of the browser world, Microsoft Internet Explorer for Windows, to test virtually everything in the book (we used versions 7, 8, and 9). We also tested the scripts with Firefox 3 and 4, for Mac and Windows, and with Safari 5 for Mac and Windows. Working with the latter browser means that our scripts should also work in any browsers based on the WebKit engine (including Google Chrome), 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, Research in Motion's Blackberry 6 and later, and HP's WebOS. 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 Web site 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. The site also includes additional tips and scripts. 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 www.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 on-screen effect happens to an image, you'll need image files. No problem. We've put all of those files up on the book's Web site, 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 Web site. It's clearly marked.

If you've read the FAQ and your question isn't answered there, you can contact us via email at: js8@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 js8@javascriptworld.com address will be answered.

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 Web page, 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 Web pages. But by the time you're done with the book, you'll be adding advanced interactivity to your sites with Ajax 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.

4

Working with Images

One of the best (and most common) uses of JavaScript is to add visual interest to Web pages by animating graphics, and that's what this chapter is all about. Making an image on a Web page 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.

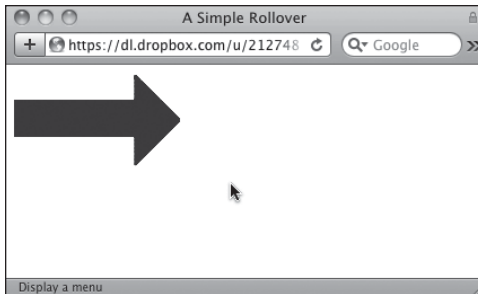
In this chapter, you'll learn how to make JavaScript do all of these image tricks. Let's get started.

In This Chapter

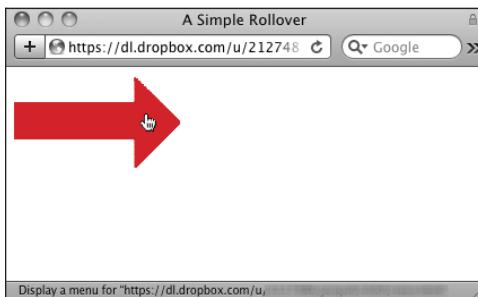
Creating Rollovers	83
Creating More Effective Rollovers	85
Building Three-State Rollovers	91
Triggering Rollovers from a Link	93
Making Multiple Links Change a Single Rollover	96
Working with Multiple Rollovers	99
Creating Cycling Banners	104
Adding Links to Cycling Banners	106
Building Wraparound Slideshows	108
Displaying a Random Image	111
Cycling Images with a Random Start	113

TABLE 4.1 Just Enough HTML—Images

Tag	Attribute	Meaning
img		Contains the attributes that describe the image to be displayed by the browser
	src	Contains the URL of the image, relative to the URL of the Web page
	width	Contains the width (in pixels) at which the browser will display the image
	height	Contains the height (in pixels) at which the browser will display the image
	alt	Used for non-visual browsers in place of the image
	id	A unique identifier, which JavaScript will use to manipulate the image



A The first image, before the user moves the mouse over it.



B When the mouse is over the image, the script replaces the first image with the second image.

Listing 4.1 Here's the simplest way to do a rollover, within a link tag.

```
<!DOCTYPE html>
<html>
<head>
  <title>A Simple Rollover</title>
  <link rel="stylesheet" href="script01.
    → css">
</head>
<body>
  <a href="next.html" onmouseover=
    → "document.images['arrow'].src=
    → 'images/arrow_on.gif'" onmouseout=
    → "document.images['arrow'].src=
    → 'images/arrow_off.gif'"><img src=
    → "images/arrow_off.gif" id="arrow"
    → alt="arrow"></a>
</body>
</html>
```

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 Web page 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 **A**, and then it is overwritten by a red arrow when the user moves the mouse over the image **B**. 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 written to the document window.

continues on next page

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.

4. ``

The image link defines the source of the original image for the page. We have included the `alt` attribute inside the image tag because `alt` attributes (which give non-graphical browsers a name or description of an image) are required if you want your HTML to be compliant with the W3C standards, and because using `alt` attributes helps make your page accessible to disabled users, such as visually impaired users who browse using screen readers.

Listing 4.2 This CSS file is used to style elements throughout many of the examples in this chapter.

```
body {
    background-color: #FFF;
}

img {
    border-width: 0;
}

img#arrow, img#arrowImg {
    width: 147px;
    height: 82px;
}

#button1, #button2 {
    width: 113px;
    height: 33px;
}

.centered {
    text-align: center;
}

#adBanner {
    width: 400px;
    height: 75px;
}
```

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.

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. ` `

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(thisImage) {`

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. `thisImage.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. thisImage.outImage.src =  
   → thisImage.src;
```

Now we set the source for the new **outImage** to be the same as the source of **thisImage**. The default image on the page is always the version you see when the cursor is off the image.

```
10. thisImage.onmouseout =  
    → function() {  
        this.src = this.outImage.src;  
    }
```

The first line here starts off what's called an *anonymous* function—that is, it's a function without a name. We could name it (say, **rollOut()**), but as it's only one line it's not so necessary.

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 **outImage** version of the image.

```
11. thisImage.overImage = new Image();  
    thisImage.overImage.src =  
    → "images/" + thisImage.id +  
    → "_on.gif";
```

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

continues on next page

```
12. thisImage.onmouseover =  
    → function() {  
        this.src = this.overImage.src;  
    }  
}
```

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.

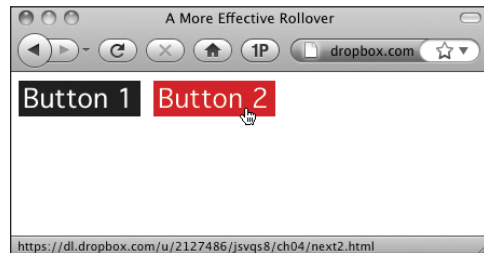
TIP Both the original and the replacement images need to have identical dimensions. Otherwise, some browsers resize the images for you, and you probably won't like the distorted result.

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.



A You can also put multiple rollovers on the same page.



B Hovering over the second rollover.

To build a three-state rollover:

```
1. thisImage.clickImage =  
   → new Image();  
   thisImage.clickImage.src =  
   → "images/" + thisImage.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. thisImage.onclick = function() {  
    this.src = this.clickImage.src;  
}
```

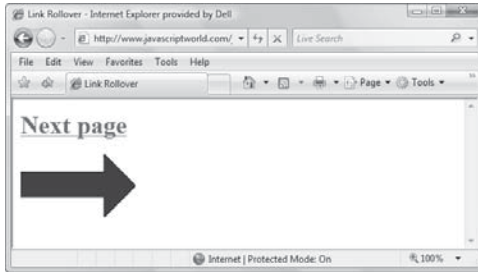
This tells the browser what to do when the user clicks the mouse on the image: in this case, we want to set the image source to its `clickImage` version.

Listing 4.6 This script powers the three-state rollover.

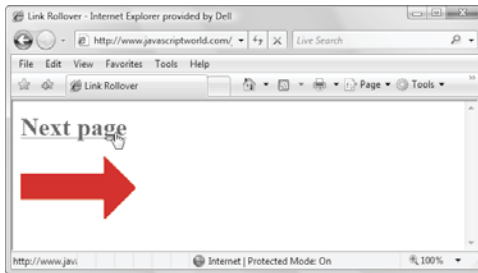
```
window.onload = rolloverInit;  
  
function rolloverInit() {  
    for (var i=0; i<document.images.length;  
        → i++) {  
        if (document.images[i].parentNode.  
            → tagName == "A") {  
            setupRollover(document.images[i]);  
        }  
    }  
}  
  
function setupRollover(thisImage) {  
    thisImage.outImage = new Image();  
    thisImage.outImage.src = thisImage.src;  
    thisImage.onmouseout = function() {  
        this.src = this.outImage.src;  
    }  
  
    thisImage.clickImage = new Image();  
    thisImage.clickImage.src = "images/" +  
    → thisImage.id + "_click.gif";  
    thisImage.onclick = function() {  
        this.src = this.clickImage.src;  
    }  
  
    thisImage.overImage = new Image();  
    thisImage.overImage.src = "images/" +  
    → thisImage.id + "_on.gif";  
    thisImage.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.

```
<!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>
  
</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(thisLink,thisImage) {  
    thisLink.imgToChange = thisImage;  
    thisLink.onmouseout = function() {  
        this.imgToChange.src =  
        → this.outImage.src;  
    }  
    thisLink.onmouseover = function() {  
        this.imgToChange.src =  
        → this.overImage.src;  
    }  
  
    thisLink.outImage = new Image();  
    thisLink.outImage.src = thisImage.src;  
  
    thisLink.overImage = new Image();  
    thisLink.overImage.src = "images/" +  
    → thisLink.id + "_on.gif";  
}
```

```
5. function setupRollover
   → (thisLink,thisImage) {
       thisLink.imgToChange =
       → thisImage;
```

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.

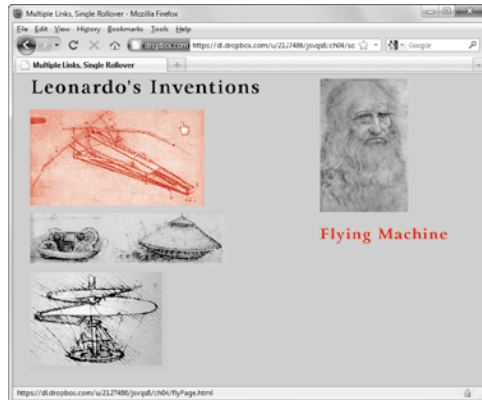
```
6. thisLink.onmouseout = function() {
    this.imgToChange.src =
    → this.outImage.src;
}
thisLink.onmouseover = function() {
    this.imgToChange.src =
    → this.overImage.src;
}
```

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 image (or actually, the link associated with that image) can trigger a rollover effect. But you can also have several different images that trigger a rollover. This can be very useful when you have several images that you want to annotate. 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 in a text box. The description itself is another image. Actually, it's three images, one for each of the three inventions. **A** shows **Listing 4.9** (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.



A This page has three interactive images, a flying machine, a tank, and a helicopter. When you roll over an image, its description appears under Leonardo's face.

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

```
<!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">
    
    
  </div>
  <div id="inventionDiv">
    
    <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"></a>
    <a href="heliPage.html" class=
    → "captionField" id="helicopter"></a>
  </div>
</body>
</html>
```

Listing 4.10 In this CSS file, we define the classes we reference in the HTML.

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

img {
  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;
}
```

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.

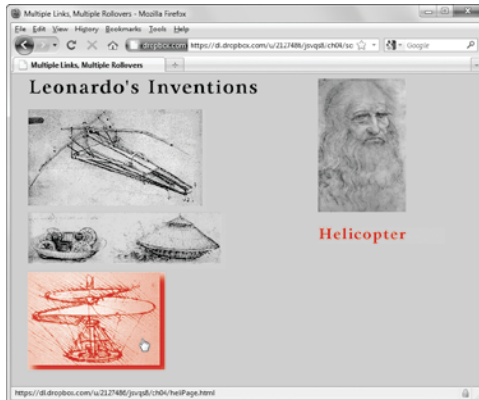
```
2. function setupRollover  
   → (thisLink,textImage) {  
     thisLink.imgToChange =  
     → textImage;
```

The **setupRollover()** function is passed the current link object (**thisLink**) 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(thisLink,textImage) {  
    thisLink.imgToChange = textImage;  
    thisLink.onmouseout = function() {  
        this.imgToChange.src =  
        → this.outImage.src;  
    }  
    thisLink.onmouseover = function() {  
        this.imgToChange.src =  
        → this.overImage.src;  
    }  
  
    thisLink.outImage = new Image();  
    thisLink.outImage.src = textImage.src;  
  
    thisLink.overImage = new Image();  
    thisLink.overImage.src = "images/" +  
    → thisLink.id + "Text.gif";  
}
```



A 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.

```
<!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">
    
    
  </div>
  <div id="inventionDiv">
    
    <a href="flyPage.html" class="captionField" id="flyer"></a>
    <a href="tankPage.html" class="captionField" id="tank"></a>
    <a href="heliPage.html" class="captionField" id="helicopter"></a>
  </div>
</body>
</html>
```

Listing 4.13 This script handles the multiple rollovers.

```
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(thisLink,textImage) {
    thisLink.imgToChange = new Array;
    thisLink.outImage = new Array;
    thisLink.overImage = new Array;

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

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

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

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

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

        thisLink.overImage[1] = new Image();
        thisLink.overImage[1].src = "images/" + thisLink.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. `thisLink.imgToChange = new Array;`
`thisLink.outImage = new Array;`
`thisLink.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 `thisLink`, each of which is an array.

2. `thisLink.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. `thisLink.outImage[0] =`
→ `new Image();`
`thisLink.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. `thisLink.overImage[0] =`
→ `new Image();`
`thisLink.overImage[0].src =`
→ `"images/" + thisLink.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  
    → (thisLink.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. thisLink.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. thisLink.outImage[1] =  
    → new Image();  
    thisLink.outImage[1].src =  
    → rolloverObj.src;
```

This sets the second array element of `outImage` to the out (off) version of the image.

```
8. thisLink.overImage[1] =  
    → new Image();  
    thisLink.overImage[1].src =  
    → "images/" + thisLink.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 thisAd = 0;`

Our script starts by creating `thisAd`, which is given its beginning value in this code.

2. `function rotate() {` `var adImages = new Array` `→ ("images/reading1.gif",` `→ "images/reading2.gif",` `→ "images/reading3.gif");`

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

3. `thisAd++;`

Take the value of `thisAd`, and add one to it.

4. `if (thisAd == adImages.length) {` `thisAd = 0;`

This code checks to see if the value of `thisAd` is equal to the number of items in the `adImages` array; if it is, then set the value of `thisAd` back to zero.



A The first image, which starts the cycling banner...



B ...the second image...



C ...the final image. Once the page loads and the banner begins cycling, the animation continues with no user intervention required.

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

```
<!DOCTYPE html>
<html>
<head>
  <title>Rotating Banner</title>
  <script src="script07.js"></script>
  <link rel="stylesheet" href="script01.
css">
</head>
<body>
  <div class="centered">
    
    </div>
</body>
</html>
```

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

```
window.onload = rotate;

var thisAd = 0;

function rotate() {
  var adImages = new Array("images/
  → reading1.gif", "images/reading2.
  → gif", "images/reading3.gif");

  thisAd++;
  if (thisAd == adImages.length) {
    thisAd = 0;
  }
  document.getElementById("adBanner").
  → src = adImages[thisAd];

  setTimeout(rotate, 3 * 1000);
}
```

5. `document.getElementById`

```
→ ("adBanner").src =
→ adImages[thisAd];
```

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 `thisAd` 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 3,000 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.

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.

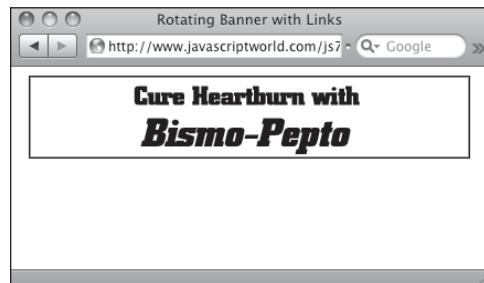
To add links to cycling banners:

1. `window.onload = initBannerLink;`

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

Listing 4.16 The HTML needed for an ad banner.

```
<!DOCTYPE html>
<html>
<head>
  <title>Rotating 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>
```



A Each of these three images is a link, and clicking each image takes you to one of three different Web sites.

Listing 4.17 This script shows how you can turn cycling banners into real, clickable ad banners.

```
window.onload = initBannerLink;

var thisAd = 0;

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

    rotate();
}

function newLocation() {
    var adURL = new Array("negrino.com",
        → "sun.com", "microsoft.com");
    document.location.href = "http://www." +
        → adURL[thisAd];
    return false;
}

function rotate() {
    var adImages = new Array("images/
        → banner1.gif", "images/banner2.gif",
        → "images/banner3.gif");

    thisAd++;
    if (thisAd == adImages.length) {
        thisAd = 0;
    }
    document.getElementById("adBanner").
        → src = adImages[thisAd];

    setTimeout(rotate, 3 * 1000);
}
```

TIP The `adURL` array needs to have the same number of array items as the `adImages` array for this script to work correctly.

```
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. function newLocation() {
    var adURL = new Array("negrino.
        → com", "sun.com", "microsoft.com");
```

In the new function `newLocation()`, the `adURL` variable gets assigned the three constituents of a new array. Just the domain names need to go in here, because we'll complete the URLs next.

```
4. document.location.href =
    → "http://www." + adURL[thisAd];
    return false;
```

Still 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 `thisAd`, 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.

Building Wraparound Slideshows

Slideshows on Web sites 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.

```
<!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>
    
    <h2><a href="previous.html"
      → id="prevLink">&lt;&lt; Previous
      → </a>&nbsp;&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.

```
window.onload = initLinks;

var myPix = new Array("images/robot1.jpg", "images/robot2.jpg", "images/robot3.jpg");
var thisPic = 0;

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

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

function processNext() {
  thisPic++;
  if (thisPic == myPix.length) {
    thisPic = 0;
  }
  document.getElementById("myPicture").src = myPix[thisPic];
  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.

continues on next page



A Clicking the Previous or Next link calls the `processPrevious()` or `processNext()` function, respectively.

```
3. function processPrevious() {  
    if (thisPic == 0) {  
        thisPic = myPix.length;
```

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

```
4. thisPic--;  
document.getElementById  
→ ("myPicture").src =  
→ myPix[thisPic];
```

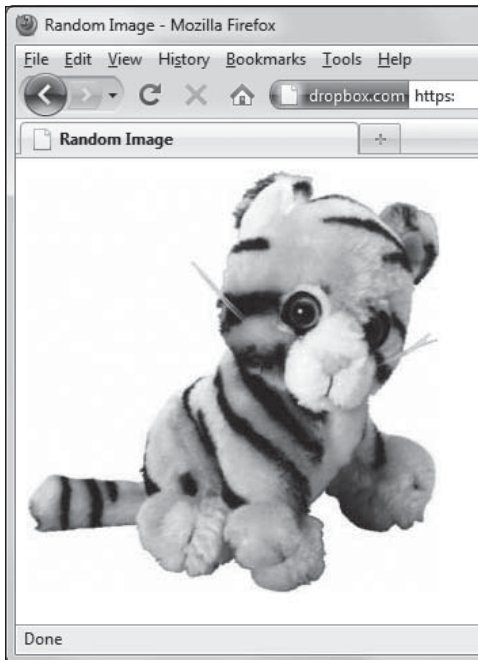
The first line reduces the value of **thisPic** by 1. The next line sets the **src** of **myPicture** to the element of the **myPix** array represented by the current value of **thisPic**.

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

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 **thisPic** by 1. Then it checks to see if the value of **thisPic** is the same as the number of items in the **myPix** array. If so, it sets **thisPic** 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!).



A 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");`

Inside the function `choosePic()`, as is now familiar, build an array of three images, and stuff it into the variable `myPix`.

2. `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.

```
<!DOCTYPE html>
<html>
<head>
  <title>Random Image</title>
  <script src="script10.js"></script>
  <link rel="stylesheet"
    → href="script01.css">
</head>
<body>
  
</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.

```
window.onload = choosePic;

function choosePic() {
  var myPix = new Array("images/lion.jpg",
    → "images/tiger.jpg", "images/bear.jpg");
  var randomNum = Math.floor
    → ((Math.random() * myPix.length));
  document.getElementById("myPicture").
    → src = myPix[randomNum];
}
```

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

```
<!DOCTYPE html>
<html>
<head>
  <title>Rotating Random Banner</title>
  <script src="script11.js"></script>
  <link rel="stylesheet"
    → href="script01.css">
</head>
<body>
  <div class="centered">
    
  </div>
</body>
</html>
```

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

```
window.onload = choosePic;

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

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

  rotate();
}

function rotate() {
  thisAd++;
  if (thisAd == adImages.length) {
    thisAd = 0;
  }
  document.getElementById("adBanner").src = adImages[thisAd];
  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 choosePic() {
```

This function is similar to the **choosePic()** function in Listing 4.21. See that explanation for the details of how it works.

```
3. function rotate() {
```

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

Index

Symbols

- ! (not), 16
- > (greater than), 16
- < (less than), 16
- % (modulus), 15
- " (quotes), 432
- # (hash symbol), 19
- \$ (dollar sign), 176, 365–366, 379
- sp;** (non-breaking space), 51
- &** (and) operator, 70
- &&** (and) operator, 16, 74
- () (parentheses)
 - in functions, 25, 28
 - in loops, 51
 - in methods, 12
 - in regular expressions, 174
- * (asterisk), 15, 175
- + (plus sign) operator, 15, 174
- ++** (increment) operator, 15, 53
- (minus sign) operator, 15
- (decrement) operator, 15
- . (dot/period), 12
- / (slash), 15, 122, 173
- // (comment indicator), 30
- /*** and ***/** (comment indicators), 29
- ; (semicolon), 24, 173, 425, 435
- = (equals sign), 15, 16, 34
- ==** (equivalence), 16, 34
- ? (question mark), 34, 175
- @ character, 175
- [] (brackets), 174
- \ (backslash), 122, 174
- ^ (caret), 174
- { } (braces), 25, 34
- | (or) operator, 70

- || (or) operator, 16, 74
- 3-state rollovers, 91–92, 192
- 12-hour time format, 272
- 24-hour time format, 272

A

- accented characters, 441–443
- accessibility, 296
- accordion menus, 382–385
- action** attribute, 134
- ActiveX, 332, 338
- Adaptive Path, 327
- Adobe
 - Dreamweaver. *See* Dreamweaver
 - Flash, 4, 104
 - Photoshop, 306, 444
- Ajax, 325–362
 - article about, 327
 - auto-completing form fields with, 357–362
 - and back buttons, 330
 - browser considerations, 329
 - and caching, 338, 348
 - coining of term, 9, 327
 - drawbacks/problems, 329–330, 338, 348
 - engine, 328
 - how it works, 328–329
 - and jQuery, 398–399
 - and JSON format, 260, 349
 - parsing server data with, 339–345, 349–352
 - popularity of, 327
 - previewing links with, 353–356
 - purpose of, 8
 - recommended video on, 493
 - refreshing server data with, 346–348

Ajax (*continued*)
 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, 327, 362
Ajax Essential Training, 493
Ajaxian blog, 492
alert boxes, 218
alert() method, 32
alert windows, 31–32
alphabetizing names, 187–188
alt attribute, 82
America Online (AOL), 456, 8442
AM/PM, adding to time, 271, 273
anchor tags (<a>), 22, 87
and (&) operator, 70
and (&&) operator, 16, 74
animated GIFs, 104, 105
animation, 81, 83, 85, 104–105
annotating scripts, 29–30
anonymous functions, 89, 90, 136
AOL (America Online), 42, 84, 456
Apache, 172
appendChild() method, 244, 253
Apple Safari. *See* Safari
applets, Java, 4, 5
arithmetic operators, 15
arrays
 declaring, 59
 defined, 59
 updating, 62–63
 using string, 77–80
assignment operators, 16
asterisk (*), 15, 175
Asynchronous JavaScript and XML, 9, 327.
 See also Ajax
<a> (anchor) tags, 22, 87
Atom feeds, 340
attributes
 action, 134
 alt, 82
 autocomplete, 358

class, 18–19, 70, 72, 74, 99, 134
 deprecated, 24
 for, 134
 height, 82
 href, 22
 id. *See* **id** attribute
 language, 24
 maxlength, 134
 name, 116, 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, 420–424
autocomplete attribute, 358

B

back button, 117, 330
background color, changing Web page, 432,
 433–435
background properties (CSS), 483
backslash (\), 122, 174
Bare Bones Software, 20
BBEdit, 20, 172
binary math, 74
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, 53, 59
 using loop to create table for, 53–54
bit.ly, 449
bits, 70, 71, 75–76
bitwise arithmetic, 70, 72, 75–76
blind users, 296

- block-level elements, 18
- blogs. *See* weblogs
- blur()** method, 132
- body scripts, 23
- <body>** tags, 22, 23
- bookmarklets, 425–452
 - for changing page's styles, 433–435
 - for converting kilometers to miles, 446
 - for converting RGB values to hex, 444–445
 - creating
 - in Firefox, 426
 - in Internet Explorer, 428–431
 - in Safari, 427
 - defined, 425
 - for displaying ISO Latin characters, 441–443
 - for doing complex calculations, 447–448
 - and IE security, 431
 - for looking up words, 436–438
 - for mailing Web pages, 451
 - origin of, 427
 - repositioning, 427
 - for resetting page background, 432
 - for resizing pages, 452
 - for shortening URLs, 449
 - use of semicolons in, 425, 435
 - use of single vs. double quotes in, 432
 - for validating pages, 450
 - for viewing images, 439–440
 - vs. other JavaScript code, 425
- bookmarklets.com**, 427
- books
 - Dreamweaver: Visual QuickStart Guide*, 138
 - HTML, XHTML, and CSS: Visual QuickStart Guide*, 2
 - JavaScript, The Definitive Guide*, 493
 - Perl and CGI for the World Wide Web: Visual QuickStart Guide*, 176
 - ppk on JavaScript*, 493
 - Pro JavaScript Techniques*, 493
 - Styling Web Pages with CSS: Visual QuickProject Guide*, 481
- Boolean values, 15, 61, 63, 70, 77
- box properties (CSS), 485
- braces (**{}**), 25, 34
- brackets (**[]**), 174
- browser compatibility, 398
- browser detection, 58
- browser objects, 11. *See also* objects
- browsers. *See also* specific browsers
 - and Ajax, 329, 348
 - and alert boxes, 32
 - and browser detection, 58
 - and caching, 348
 - and case-sensitivity, 480
 - and cookies, 219–220
 - and Daylight Savings Time, 271
 - and DOM, 242, 438
 - and ECMAScript, 458
 - and event handlers, 69
 - and external JavaScript files, 28
 - and JavaScript toolkits, 373
 - and JavaScript versions, 454
 - performing word lookups in, 436–438
 - and pop-up windows, 129
 - and resizing of images, 90
 - and rollovers, 84, 90
 - and security problems, 431
 - testing scripts in different, 130
 - viewing document tree structure in, 13
 - and Year 2000 Problem, 277
- browser security settings, 129
- browser windows, 127. *See also* windows
- buttons
 - back, 117, 330
 - radio, 156–158
 - submit, 133, 139, 142, 212
- Buzzword Bingo game, 77–80

C

- C#, 3
- C/C++, 3
- cache files
 - and Ajax, 338, 348
 - and external scripts, 28
 - and **onload/onunload** events, 136, 138
 - and rollovers, 85
 - and select-and-go menus, 136

- calculators, 447–448
- calendars
 - adding to Web pages, 403–407
 - Google, 10
- calendar widget, 403–407
- callback function, 352
- calling functions, 25
- Camino, 454
- capitalizing names, 183–185
- caret (^), 174
- Cascading Style Sheets, 9. *See also* CSS
- case-sensitivity, 15, 90, 480
- case** statements, 43–45
- Castro, Elizabeth, 2, 176
- catch** statement, 47
- CDN (Content Delivery Network), 367
- CGI programs
 - and forms, 133
 - for getting from one page to another, 139
 - and JavaScript, 7
 - meaning of acronym, 7
 - recommended book on, 176
- characters, displaying ISO Latin, 441–443
- charts, 306–314
 - code for drawing, 308–311
 - HTML page for generating, 306
 - script containing styles for, 307
 - source of statistics for, 314
- child frames, 117
- Chrome, 28, 58, 242, 373, 431, 458
- class** attribute, 18–19, 70, 72, 74, 98
- classes, CSS pseudo-, 482
- client machines, reading/writing files on, 7
- client-side languages, 4, 6, 7
- closing windows, 7
- Cocoa-based programs, 438
- code-checking tool, 494
- code-writing tool, 495
- color
 - changing Web page background, 432, 433–435
 - converting RGB to hexadecimal, 444–445
 - properties (CSS), 483
- color-picker script, 369–370
- comment indicators (*/** and **/*), 29
- commenting scripts, 29–30
- Common Gateway Interface, 133. *See also* CGI
- Communicator, Netscape, 454
- comparison operators, 16
- compile()** method, 186
- conditionals
 - if/else**, 33–34, 43
 - multi-level, 43–45, 276
 - switch/case**, 43–45, 276
 - use of **&&** and **||** in, 74
- confirm()** method, 33–34
- container tags, 23
- Content Delivery Network (CDN), 367
- converters
 - kilometers-to-miles, 446
 - RGB-to-hexadecimal, 444–445
- 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, 266, 283
- Core JavaScript Reference/Guide, 490
- countdown script, 274–277
- counter programs, 230
- counters, 50, 53, 228–230
- country pop-up menus, 140
- createElement()** method, 244
- createTextNode()** method, 244
- CSS
 - and Ajax, 9, 327
 - basic concepts, 482
 - color units, 488

- combining JavaScript and, 68–70
- and jQuery, 364, 368, 379
- length units, 488
- meaning of acronym, 9
- and object literals, 257
- properties, 483–488
- pseudo-elements/classes, 482
- purpose of, 17
- recommended books on, 2, 481
- reference, 481–488
- styling Bingo card with, 52
- tools for creating, 20
 - and HTML, 17

CSS 2.0 specification, 481

CSS 2.1 specification, 481

.css file extension, 20

customizing messages

- for time of day, 265
- for weekday vs. weekend, 264

cycling banners

- adding links to, 106–107
- creating, 104–105
- displaying random images in, 113–114

D

data

- accessing other people's, 345
- automatic entry of, 357–362
- parsing, 339–345, 349–352
- presenting tabular, 49, 390–392
- refreshing, 346–348
- requesting/reading, 331–338, 349–352
- using jQuery with external, 411–419

date methods, 283–284

datepicker widget, 403–407

dates. *See also* calendars

- comparing two, 277
- displaying by time zone, 266–271
- distinguishing between weekdays/
weekends, 264
- dynamically displaying on Web page,
262–263
- how JavaScript stores, 277

Daylight Savings Time, 271

day/month pop-up menus, 140–141

Debugger, Firebug, 494

Debugger, Venkman, 490–491

decimal math, 74

deprecated attributes, 24

detection methods, 58

DHTML, 42, 373

dialogs, 386–387

dictionary-lookup script, 436–438

<div> tags, 18

documentation

- JavaScript, 490
- jQuery, 376
- node manipulation, 260

Document Object Model, 9. *See also* DOM

document tree structure, 13, 243

document.write() method, 28

Dojo, 373

dollar sign (\$), 176, 365–366, 379

DOM

- and Ajax, 9, 327
- defined, 13
- meaning of acronym, 9
- and nodes, 13, 241–243
- scripting, 42
- and W3C, 242, 243
- and Web browsers, 242, 438

DOM-2, 242–243

DOM-3, 243

DOM Inspector, 13, 459

dots-follow-cursor script, 209

dot syntax, 12–13, 196

do/while loops, 64–65, 78

draggable dialogs, 387

dragging/dropping page elements, 408–410

Dreamweaver

- jump menus, 138
- recommended book on, 138
- and regular expressions, 172
- scripting in, 20

Dreamweaver: Visual QuickStart Guide, 138

drop shadows, 99

Dynamic HTML, 42, 373

dynamic iframes, 123–124

- dynamic pages, 261–282
 - converting 24-hour to 12-hour time, 272–273
 - creating countdowns, 274–277
 - customizing message for time of day, 265
 - customizing message for weekday vs. weekend, 264
 - defined, 261
 - and jQuery, 368
 - moving objects in, 281–282
 - putting current date on page, 262–263

E

- ECMA-262 specification, 454, 457.
 - See *also* ECMAScript

- ECMAScript
 - bindings, 243
 - flavors of, 457–458
 - and JavaScript versions, 454
 - and Netscape, 457
 - official specification for, 457
 - reserved words, 478–479

- ejohn.org**, 376

- element nodes, 13, 243, 244

- elements

- block-level vs. inline, 18
 - CSS pseudo-, 482
 - highlighting page, 378–381
 - identifying, 19
 - modifying, 19

- else** statements, 34, 43

- Emacs, 20

- email, sending Web pages via, 451

- email addresses

- validating, 166–170, 173–176
 - verifying, 169

- equals sign (=), 15, 16, 34

- error-handling script, 46–47

- error messages, 46–47, 201

- escaping characters, 174

- event handlers, 195–218

- defined, 14, 195
 - for form events, 212–215
 - onblur**, 213–214
 - onchange**, 212

- onclick**, 213

- onfocus**, 215

- onreset**, 212

- onselect**, 212

- onsubmit**, 212

- importance of, 195

- for key events, 216–218

- onkeydown**, 216–217

- onkeypress**, 218

- onkeyup**, 218

- list of common, 14

- for mouse events, 204–211

- onclick**, 211

- ondblclick**, 210

- onmousedown**, 204–206

- onmousemove**, 207–209

- onmouseout**, 210

- onmouseover**, 210, 353

- onmouseup**, 206, 211

- reference, 465–476

- for window events, 196–203

- onabort**, 201

- onblur**, 203

- onerror**, 201

- onfocus**, 202

- onload**, 14, 197–199

- onmove**, 201

- onresize**, 200–201

- onunload**, 14, 136, 200

- event** property, 69

- events. See *also* event handlers

- defined, 14

- form, 212–215

- key, 216–218

- mouse, 204–211

- window, 196–203

- evt** property, 69

- Excel. See Microsoft Excel

- exec()** method, 184, 186

- external scripts, 26–28

- extracting strings, 180–182

- eyeballs script, 207–209

F

favelets, 425

favorites, 425

fields

- auto-completing, 357–362, 374–375
- checking one against another, 147–148
- identifying/marketing problem, 149–150
- 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

file names, validating, 178–179

finally {} block, 47

Firebug Debugger, 494

Firebug Lite, 494

Firefox

- and alert boxes, 32
- creating bookmarklets in, 426
- debugger, 491, 494
- and DOM-2, 242
- DOM Inspector, 13, 459
- and ECMAScript, 458
- and external JavaScript files, 28
- and **focus()** method, 214
- and JavaScript, 454
- and JavaScript toolkits, 373
- non-standard window event handlers, 138
- and **oncontextmenu** events, 206
- and **onkeydown** events, 217
- and **onload** events, 136, 138
- and **onmousedown** events, 206
- and page caching, 136, 138
- performing word lookups in, 436–437
- window defaults, 130
- and Year 2000 Problem, 277

Flanagan, David, 493

Flash animations, 4, 104

Flickr, 325, 339–345, 349–352

focus() method, 132, 214

following-eyes script, 207–209

font properties (CSS), 483

fonts, serif vs. sans-serif, 315, 318

for attribute, 134

for loops, 50, 53, 54

formatting

properties (CSS), 486

strings, 183–191

list of names, 183–188

phone numbers, 189–191

time, 272–273

form event handlers, 212–215

onblur, 213–214

onchange, 212

onclick, 213

onfocus, 215

onreset, 212

onselect, 212

onsubmit, 212

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, 133–193

auto-completing fields in, 357–362, 374–375

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

setting field values automatically in, 159–161

tags/attributes, 134

using radio buttons in, 156–158

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

- <form> tags, 134, 139
- form validation, 133, 212
- forums, jQuery, 376
- frames, 115–126
 - 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, 125–126
- framesets, 117, 118. *See also* frames
- frameworks, JavaScript, 363–364, 367, 373, 377
- functions. *See also* specific functions
 - anonymous, 89, 136
 - calling, 25
 - components of, 25
 - defined, 25
 - naming, 25, 479
 - passing values to, 55–56
 - sharing between documents, 125–126
 - use of parentheses in, 28
- function values, 15

G

- Garrett, Jesse James, 9, 327
- generated content properties (CSS), 484
- getElementById()** method, 27, 459
- getElementsByTagName()** method, 245, 246, 247
- getFullYear()** method, 277
- getHours()** method, 265
- getTime()** method, 277
- getYear()** method, 277
- GIF images, 90, 104, 105
- GIFs, spacer, 113
- Gmail, 10, 325
- GMT, 266, 283
- goo.gl, 449
- Google
 - and Ajax, 10
 - browser. *See* Chrome
 - and jQuery, 367
- Google Calendar, 10

- Google Docs, 10
- Google Gmail, 10, 325
- Google Instant, 362
- Google Maps, 8, 10, 325
- Google Maps Mania, 10
- googlemapsmania.blogspot.com**, 10
- graphics. *See also* images
 - animating, 81
 - preparing for rollovers, 90
- Greenwich Mean Time, 266, 283
- grep, 171

H

- <h1>...<h6> tags, 22
- hash symbol (#), 19
- header scripts, 23
- <head> tags, 22, 23
- “Hello, Cleveland!” script, 2
- “Hello, world!” script, 23, 27, 29–30, 426
- hexadecimal, converting RGB values to, 444–445
- hide()** method, 379, 381
- highlighting new elements, 378–381
- hijacking pages, 117
- hit counters, 230
- hover()** method, 372–373
- href** attribute, 22
- HTML
 - and Ajax, 9, 327
 - attributes, 22, 49, 82, 116, 134
 - and case, 90
 - classic style of, 18
 - and CSS, 17
 - deprecation of attributes in, 24
 - evolution of, 1
 - forms, 133
 - modern, 17
 - purpose of, 17
 - recommended book on, 2
 - separating JavaScript from, 41, 43
 - tags, 22, 49, 82, 116, 134
 - tools for writing, 20
 - writing JavaScript-friendly, 17–19
 - and W3C validation, 17

HTML, XHTML, and CSS: Visual QuickStart Guide, 2

.html file extension, 20

HTML Source mode, 20

<html> tags, 22

Hypertext Markup Language. See HTML

I

id attribute

and forms, 134

and frames, 116

and “Hello, world!” script, 27

and images, 82, 103

manipulating cell contents with, 51

purpose of, 18, 19, 22

IE. See Internet Explorer

if/else conditionals, 33–34, 43, 57

iframes

creating content for, 122

creating dynamic, 123–124

defined, 118

loading, with JavaScript, 120

working with, 121–122

images, 81–114

annotating, 96

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, 192

viewing table of, 439–440

**** tags, 82, 86, 90

increment step, **for** loop, 54

index number, 59

initialization step, **for** loop, 54

inline elements, 18

innerHTML property, 27, 28, 42, 245

<input> tags, 134

insertBefore() method, 253

interactive Web applications, 8

interactivity, 1, 6, 68–70, 369–373

internal scripts, 26

Internet Explorer

and alert boxes, 32

and AOL, 456

creating bookmarklets in, 428–431

and DOM-2, 242

and ECMAScript, 458

and event handlers, 69

and external JavaScript files, 28

and **getFullYear()** method, 277

and JavaScript toolkits, 373

and **oncontextmenu** events, 206

and **onkeydown** events, 217

and **onmousedown** events, 206

performing word lookups in, 436–437

and pop-up windows, 129

and rollovers, 84, 90

scripting capabilities, 5

security problems, 431

and security settings, 129

and tabbed browsing, 129

versions of, 455, 456

viewing document’s tree structure in, 13

window defaults, 130

and **XMLHttpRequest** object, 332, 338

Year 2000 Problem, 277

Internet time server, 271

iPhone, 80

isNaN() method, 47

ISO Latin characters, 441–443

J

Java

applets, 4, 5

cross-platform compatibility of, 3, 4

developers of, 3

and Microsoft, 3

and regular expressions, 172

vs. JavaScript, 3

vs. other programming languages, 3

ways of using, 4

JavaScript

- adding visual interest to Web pages with, 81
- and AOL, 42, 456
- applying styles with, 68–70
- and browser compatibility, 398
- calculator, 447–448
- case-sensitivity of, 15
- as client-side language, 7
- combining CSS and, 68–70
- and cookies, 219–220. *See also* cookies
- documentation, 490
- and DOM, 13, 243
- enhancing links with, 39–41
- evolution of, 1
- frameworks, 363–364, 367, 373, 377
- hiding from users, 28, 204–206
- how events are handled in, 14
- inventor of, 1
- libraries. *See* JavaScript frameworks
- limitations of, 7
- loading iframes with, 120
- Math** object, 447
- Microsoft version of, 5, 42
- modifying document tree structure with, 13
- and Netscape, 1, 5, 42, 454
- object flowchart, 459–464
- object hierarchy, 460
- as object-oriented language, 11
- object table, 465–476
- operators, 15–16
- as programming language, 2
- purpose of, 1, 17
- recommended books on, 493
- recommended video on, 493
- and regular expressions, 171, 172
- reserved words, 477–480
- resources, 489–496
- rewriting with object literals, 257–260
- as scripting language, 2
- and standards-compliant HTML, 17
- terminology, 42
- toolkits. *See* JavaScript frameworks
- tools for writing, 20, 363–364

- tutorials, 493

- use of semicolons in, 24
- using functions in, 25
- value types, 15
- versions, 453, 454–456
- vs. Java, 3
- ways of using, 6

JavaScript, The Definitive Guide, 493

JavaScript Center, 490

JavaScript Essential Training, 493

JavaScript Guide, Netscape, 427

JavaScript Object Notation (JSON), 260.

See also JSON format

JavaScript World, 2, 489

javascriptworld.com, *see* JavaScript World

Jobs, Steve, 80

JPEG images, 105

jQuery, 363–424

- adding to page, 365–366

- adding user interaction with, 369–373

- and Ajax, 398–399

- alternatives to, 373

- auto-completing fields with, 374–375

- and browser compatibility, 398

- calendar widget, 403–407

- and CSS, 364, 368, 379

- designing with

 - creating accordion menus, 382–385

 - creating custom themes, 400–402

 - creating smarter dialogs, 386–387

 - highlighting new elements, 378–381

 - sorting tables, 393–396

 - striping tables, 390–392

- documentation, 376

- and dollar sign (?), 365–366

- downloading, 376

- as foundation, 398–399

- forums, 376

- and JSON, 398–399, 411

- plugins, 376, 399, 420–424

- purpose of, 364

- resources, 376

- serving, 367

- strengths of, 364, 373, 398
- themes, 385, 388, 400–402
- tutorials, 376
- updating buttons with, 372–373
- updating page with, 368
- use of dollar sign (\$) in, 379
- user interface (UI). See jQuery UI
- using with external data, 411–419
- versions, 366, 367
- and “yellow fade,” 377, 378

jQuery UI, 376, 377, 378, 381, 396, 400

- ThemeRoller. See ThemeRoller
- themes, 385, 388, 400–402

JSBin, 495

JScripT, 5, 42, 455, 491

JSFiddle, 495

.js file extension, 20, 26, 206

JSHint, 494

JSON format, 9, 260, 349–352, 398–399, 411

jump menus, 138

K

Kangas, Steve, 427

key event handlers, 216–218

- onkeydown**, 216–217
- onkeypress**, 218
- onkeyup**, 218

keywords

- this**, 41, 260
- var**, 35, 36

kilometers-to-miles converter, 446

Koch, Peter-Paul, 492, 493

L

<label> tags, 134

language attribute, 24

languages

- client-side, 7
- object-based, 11
- object-oriented, 11
- scripting, 2, 5

Latin characters, ISO, 441–443

layers, hiding/displaying, 278–280

leap year, 141

length units (CSS), 488

libraries, JavaScript. See frameworks, JavaScript

light table script, 408–410

limiting step, **for** loop, 54

link enhancement script, 39–40

links

- enhancing with JavaScript, 39–41
- previewing, 95, 353–356
- redirecting users with, 37–38
- triggering rollovers from, 93–98

Linux, 3

list properties (CSS), 487

lists, 292

**** tags, 292

literal values, 16

LiveScript, 5, 454

loops

- counters for, 50, 53
- importance of, 50
- specific types

 - do/while**, 64–65, 78
 - for** loops, 50, 53, 54

Lynda.com, 493

M

Macintosh, 455

Mac OS X

- and AOL, 456
- Camino, 454
- dictionary/thesaurus window, 438
- and ECMAScript, 458
- and Java, 3
- text editors, 20

Macworld Expo, 80

mailing Web pages, 451

mailto links, 451

MapQuest, 8

mashups, 10

match() method, 186

math, binary vs. decimal, 74

Math object

- abs()** method, 447
 - acos()** method, 447
 - asin()** method, 447
 - atan()** method, 447
 - ceil()** method, 447
 - cos()** method, 447
 - exp()** method, 447
 - floor()** method, 347, 447
 - log()** method, 447
 - max()** method, 447
 - min()** method, 447
 - pow()** method, 447
 - random()** method, 54, 126, 347
 - round()** method, 447
 - sqrt()** method, 447
 - tan()** method, 447
- maxLength** attribute, 134
- menus
- accessibility considerations, 296
 - accordion, 382–385
 - changing dynamically, 140–141
 - horizontal vs. vertical, 293
 - jump, 138
 - outline-style, 285
 - pop-up, 140–141
 - pull-down, 289–296
 - select-and-go, 135–138
 - sliding, 286–288
- methods
- combining with objects/properties, 12–13
 - defined, 12
 - distinguishing from properties, 12
 - reference, 465–476
 - for **RegExp** object, 186
 - for strings, 186
 - use of parentheses in, 12
- Microsoft
- and ECMAScript, 457–458
 - Excel, 306
 - Internet Explorer. *See* Internet Explorer
 - and Java, 3
 - and JScript, 5, 42, 455
 - JScript blog, 491

- JScript Language site, 491
 - Windows. *See* Windows
 - Word, 20, 172
- miles, converting kilometers to, 446
- military time, 272
- modifiers, regular expression, 177
- mouse click codes, 206
- mouse event handlers, 204–211
- onclick**, 211
 - ondblclick**, 210
 - onmousedown**, 204–206
 - onmousemove**, 207–209
 - onmouseout**, 210
 - onmouseover**, 210, 353
 - onmouseup**, 206, 211
- mouseout()** method, 372–373
- Mozilla. *See also* Firefox
- and browser detection, 58
 - debugger, 491
 - JavaScript Developer Center, 490
 - and JavaScript versions, 454
- Mozilla Hacks* blog, 491
- MSIE. *See* Internet Explorer
- multi-level conditionals, 43–45, 276

N

- name** attribute, 116, 134
- names
- capitalizing list of, 183–185
 - generating silly, 301–305
 - reversing list of, 180–182
 - sorting list of, 187–188
- navigation menus, 135
- Navigator
- and ECMAScript, 458
 - and JavaScript, 5, 454
 - and LiveScript, 5, 454
 - and Year 2000 Problem, 277
- nested **if** statements, 43
- Netscape
- and AOL, 456
 - Communicator, 454
 - and ECMAScript, 457, 458
 - and external JavaScript files, 28

- and following-eyes script, 209
- and JavaScript, 1, 5, 42, 454
- JavaScript Guide, 427
- and LiveScript, 5
- Navigator. See Navigator
- and rollovers, 84, 90
- “New to You” messages, 235–240
- node manipulation, 241, 242–243, 260
- nodes, 241–260
 - adding, 244–245
 - defined, 13
 - deleting, 246–250
 - and DOM, 13, 241–243
 - inserting, 251–253
 - replacing, 254–256
 - types of, 13, 243
 - vs. **innerHTML**, 245
- non-breaking space, 51
- <noscript>** tags, 32
- Notepad, 20
- null values, 15, 35
- numbers
 - random, 54, 347
 - validating, 190–191
- number sign (#), 19
- numeric values, 15

O

- object-based languages, 11
- object detection, 57–58
- object flowchart, 459–464
- object hierarchy, 460
- object literals, 257–260
 - sample scripts, 258–259, 372–373
 - similarity to CSS, 257
 - use of **this** with, 260
 - vs. standard procedural JavaScript, 257, 260
- object-oriented languages, 11
- objects
 - combining with properties/methods, 12–13
 - defined, 11
 - detecting, 57–58
 - methods of, 12
 - naming, 11

- properties of, 12
 - reference, 465–476
- object table, 465–476
- <object>** tags, 4
- object values, 15
- offline resources, 493
- onabort** events, 14, 201
- onblur** events, 14, 203, 213–214
- onchange** events, 14, 212
- onclick** events, 14, 38, 211, 213
- oncontextmenu** events, 206
- ondblclick** events, 210
- onerror** events, 14, 201
- onfocus** events, 14, 202, 215
- onkeydown** events, 216–217
- onkeypress** events, 218
- onkeyup** events, 218
- online pastebins, 495
- online resources, 490–492, 494–496
- onload** events, 14, 197–199
- onmousedown** events, 204–206
- onmousemove** events, 207–209
- onmouseout** events, 14, 90, 210
- onmouseover** events, 14, 90, 210, 353
- onmouseup** events, 206, 211
- onmove** events, 201
- onpagehide** events, 138
- onpageshow** events, 138
- onreadystatechange** events, 333
- onreset** events, 212
- onresize** events, 200–201
- onselect** events, 14, 212
- onsubmit** events, 14, 212
- onunload** events, 14, 136, 200
- open()** method, 127, 130, 333
- Opera, 58, 373, 494
- operators
 - arithmetic, 15
 - assignment, 16
 - comparison, 16
 - defined, 15
 - regular expression, 171
- <option>** tags, 134

or (|) operator, 70
or (||) operator, 16, 74
outline-style menus, 285

P

pageCount array, 123
page hit counters, 230
page properties (CSS), 488
paragraphs, 245
parameters, passing, 35, 55
parentheses
 in functions, 25, 28
 in loops, 51
 in methods, 12
 in regular expressions, 174
parseInt() method, 228, 240
parse() method, 269
passing information, 55–56
password-checking script, 142, 147–148
pastebins, 495
period (.), 12
Perl, 7, 172, 176
Perl and CGI for the World Wide Web: Visual QuickStart Guide, 176
phone numbers, formatting/validating, 189–191
Photoshop, 306, 444
PHP, 7, 172
plugins, jQuery, 399, 420–424
plus sign (+), 15, 174
PNG images, 105
pop-up killers, 129
pop-up menus, 140–141
pop-up windows, 129, 197, 200, 203, 278
postal codes, validating, 162–165
ppk on JavaScript, 493
programming languages, 2, 3, 172
progressive enhancement, 42
Pro JavaScript Techniques, 493
prompt() method, 35
properties
 combining with objects/methods, 12–13
 defined, 12

distinguishing from methods, 12
reference
 CSS, 483–488
 JavaScript, 465–476
for **RegExp** object, 186

Prototype, 373
pseudo-classes (CSS), 482
pseudo-elements (CSS), 482
pull-down menus, 289–296
Python, 172

Q

question mark (?), 175
QuirksMode blog, 492
quotes, in bookmarklets, 432

R

radio buttons, 156–158
random images, 111–114
random numbers, 54, 126, 347
ready() method, 366
readyState property, 334
redirection, 21, 37–38
RegExp object, 171, 186, 472
regular expressions, 171–193
 alternate names for, 171
 defined, 171
 extracting strings with, 180–182
 formatting strings with, 183–191
 geekiness of, 172
 modifiers for, 177
 purpose of, 171
 replacing elements with, 192–193
 sorting strings with, 187–188
 special characters for, 177
 validating email addresses with, 173–176
 validating file names with, 178–179
 validating strings with, 189–191
 validating URLs with, 178–179
 ways of using, 171, 172
replaceChild() method, 254
replace() method, 117, 186

- reserved words, 477–480
- Resig, John, 376, 493
- resizable dialogs, 387
- resizeTo()** method, 452
- resizing windows, 452
- resources
 - books
 - Ajax, 493
 - CGI programs, 176
 - CSS, 2, 481
 - Dreamweaver, 138
 - HTML, 2
 - JavaScript, 493
 - Perl, 176
 - videos, 493
 - Web sites
 - Ajaxian*, 492
 - Bare Bones Software, 20
 - bit.ly**, 449
 - bookmarklets.com**, 427
 - Dojo, 373
 - ECMA International, 457
 - Firebug Debugger, 494
 - Flickr, 325, 339–345, 349–352
 - Google Maps Mania (**googlemapsmania.blogspot.com**), 10
 - JavaScript Center, 490
 - JavaScript World (**javascriptworld.com**), 2, 489
 - jQuery (**jquery.com**), 376
 - JSBin, 495
 - JScript Language, 491
 - JSFiddle, 495
 - JSHint, 494
 - lynda.com**, 493
 - Mozilla Hacks*, 491
 - Prototype, 373
 - QuirksMode*, 492
 - Resig, John (**ejohn.org**), 376
 - Stack Exchange (**stackexchange.com**), 496
 - Stack Overflow (**stackoverflow.com**), 496
 - Surfin' Safari*, 491
 - tinyurl.com**, 449
 - Twitter (**twitter.com**), 411–419, 449
 - Venkman Debugger, 490–491
 - W3C validation tool (**validator.w3.org**), 17, 450
 - Web Standards Project (**webstandards.org**), 42
 - Wikipedia, 373
 - Willison, Simon (**simonwillison.net**), 199
 - YUI, 373
- responseText** property, 335
- responseXML** property, 335
- RGB-to-hex converter, 444–445
- rollovers, 83–103
 - browser considerations, 84, 90
 - building three-state, 91–92
 - creating more effective, 85–89
 - creating simple, 83–84
 - defined, 6, 81
 - making multiple links change single rollover, 96–98
 - preparing images for, 90
 - triggering from text links, 93–95
 - working with multiple, 99–103
- RSS feeds, 340

S

- Safari
 - and alert boxes, 32
 - blog, 491
 - and browser detection, 58
 - creating bookmarklets in, 427
 - and DOM-2, 242
 - and ECMAScript, 458
 - and external JavaScript files, 28
 - and iPhone, 80
 - and JavaScript toolkits, 373
 - and **onkeydown** events, 217
 - and **onload** events, 136, 138
 - and page caching, 136, 138
 - performing word lookups in, 436–437
 - viewing document's tree structure in, 13
 - window defaults, 130

sample scripts

Ajax

- auto-completing fields, 357–362
- 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, 433–435
- converting kilometers to miles, 446
- converting RGB values to hex, 444–445
- creating in Firefox, 427
- creating in Internet Explorer, 428–431
- creating in Safari, 427
- displaying ISO Latin characters, 441–443
- looking up words, 436–438
- mailing Web pages, 451
- resetting page background, 432
- resizing pages, 452
- shortening URLs, 449
- using JavaScript calculator, 447–448
- validating pages, 450
- viewing images, 439–440

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 Web page, 262–263

event handlers

- checking for double clicks with **ondblclick**, 210
- controlling window stacking with **onfocus**, 202
- fixing Netscape dynamic content redraw bug with **onresize**, 200–201
- hiding code with **onmousedown**, 204–206
- preventing wayward field entries with **onfocus**, 215
- setting multiple **onload** attributes, 196–199
- triggering slide change with **onkeydown**, 216–217
- using **onblur** to force field entry, 213–214
- using **onblur** to keep window in front, 203
- using **onblur** to trigger action when user leaves field, 214
- using **onmousemove** to display eyeballs that follow user, 207–209

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, 121–122
- keeping pages out of frames, 117
- loading dynamic iframes, 123–124
- loading iframes with JavaScript, 120
- setting target for frames, 118–119
- sharing functions between documents, 125–126

“Hello, Cleveland!”, 2

- “Hello, world!”, 23, 27, 29–30, 426
- images
 - cycling with random start, 113–114
 - 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 more effective, 85–89
 - rollovers, creating simple, 83–84
 - rollovers, triggering from text links, 93–95
 - rollovers, working with multiple, 99–103
- JavaScript applied
 - adding pull-down menus, 289–292
 - allowing user to switch between style sheets, 315–324
 - enhancing pull-down menus, 293–296
 - generating bar graph, 306–314
 - generating silly names, 301–305
 - showing captions in slideshows, 297–300
 - using sliding menus, 286–288
- 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, 46–47
 - 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, 403–407
 - adding jQuery to page, 365
 - adding user interaction/updates, 369–373
 - audio player plugin, 420–424
 - auto-completing fields, 374–375
 - creating accordion menus, 382–385
 - creating smarter dialogs, 386–387
 - creating sortable tables, 393–396
 - dragging/dropping elements, 408–410
 - highlighting new elements, 378–381
 - striping tables, 390–392
 - updating page, 368
- objects and 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, 183–185
 - extracting strings, 180–182
 - formatting/sorting strings, 187–188
 - formatting strings, 183–185
 - formatting/validating strings, 189–191
 - replacing page elements, 192–193
 - sorting names, 187–188
 - validating email addresses, 173–176
 - validating file names, 178–179
 - validating phone numbers, 189–191
 - validating URLs, 178–179
- windows
 - loading different contents into, 131–132
 - opening new, 127–129
- sans-serif fonts, 315, 318
- scope, variable, 36, 435
- screen size, 452
- script errors, 129
- 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
 - calling multiple ways, 66–67
 - 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
- <script>** tags
 - adding attributes to, 24
 - in book's sample scripts, 24
 - enclosing scripts in, 23
 - language** attribute of, 24
 - and page hijacking, 117
 - script text and, 2
 - where to put, 2
- search-and-replace feature, 50
- search()** method, 186
- security problems, Internet Explorer, 431
- security settings, browser, 129
- select-and-go menus, 135–138
- selected** attribute, 134
- <select>** tags, 134
- semantic chunks, breaking content into, 18
- semicolon (;), 24, 173, 425, 435
- serif fonts, 315, 318
- server data
 - parsing, 339–345, 349–352
 - refreshing, 346–348
 - requesting/reading, 331–338, 349–352
- server machines, writing of files on, 7
- server-side programs, 6, 241, 330
- setTimeout()** method, 199, 346
- silly name generator, 301–305
- simonwillison.net**, 199
- size** attribute, 134
- slash (/), 122, 173
- slideshows
 - building wraparound, 108–110
 - enabling dragging/dropping for, 408–410
 - showing captions in, 297–300
- sliding menus, 286–288
- sorting tables, 393–396
- spacer GIFs, 18, 113
- ** tags, 18
- special characters, 174, 177, 441–443
- speech synthesis, 481
- split()** method, 186, 222, 223
- square root calculator, 46–47
- src** attribute, 22, 26, 82, 116
- srcElement** property, 69
- Stack Exchange, 496
- Stack Overflow, 496
- standards. *See* Web standards
- state names, auto-completing, 357–362, 374–375
- SteveNote Bingo, 80
- string arrays, 77–80
- string methods, 186
- strings, 180–191
 - comparing, 16
 - defined, 15
 - extracting, 180–182
 - formatting, 183–191
 - returning, from functions, 61
 - sorting, 187–188
 - syntax for, 117
 - validating, 171, 189–191
- striped tables, 390–392
- style** attribute, 70
- styles. *See also* CSS
 - applying, with JavaScript, 68–70
 - changing, with bookmarklet, 433–435
- style sheet switcher, 315–324
- Styling Web Pages with CSS: Visual QuickProject Guide*, 481
- submit buttons, 133, 139, 142, 212
- substring()** method, 237, 240
- Sun Microsystems, 3
- Surfin' Safari* blog, 491

switch/case statements, 43–45, 276

syntax, dot, 12–13, 196

syntax errors, 129

T

tabbed browsing settings, 129

table-based layouts, 18

table properties (CSS), 487

tables

 sorting, 393–396

 striping, 390–392

 viewing page's images in, 439–440

<table> tags, 49

tabular data, 49, 390. *See also* tables

tag-along dots script, 209

tagName, 87, 90

tags. *See also* specific tags

 basic, 22

 form, 134

 frame, 116

 image, 82

 table, 49

target attribute, 118–119

<td> tags, 49

terminology

 DOM 2, 243

 JavaScript, 42

 node manipulation, 242–243

testing scripts, 130

test() method, 186

textContent property, 336

text editors, 20

text links, triggering rollovers from, 93–95

TextMate, 172

text nodes, 244–250

 adding, 244–245

 defined, 13, 243

 deleting, 246–250

text properties (CSS), 484

text property, 336

Text Wrangler, 20

ThemeRoller, 400–402

themes, jQuery, 385, 388–389

thesaurus-lookup script, 436–438

this keyword, 41, 260

three-state rollovers, 91–92, 192

throw statement, 46, 47

<th> tags, 49

Thunderbird, 491

time

 adding AM/PM to, 271, 273

 converting 24-hour to 12-hour, 272–273

 customizing messages for, 265

 dealing with Daylight Savings Time, 271

time server, 271

time zone, displaying dates by, 266–271

tinyurl.com, 449

<title> tags, 22

toggle() method, 380

toolkits, JavaScript. *See* frameworks,
 JavaScript

toSource() method, 186

toString() method, 186

transparent GIFs, 90

tree structure, 13, 243

<tr> tags, 49

true/false values, 15, 57, 70

try statement, 47

tutorials

 JavaScript, 493

 jQuery, 376

Twitter, 411–419, 449

type attribute, 24, 134

U

**** tags, 292

Universal Time, 266, 283

Unix, 3, 20

unobtrusive scripting, 41, 42

unordered lists, 292

URLs

 shortening, 449

 updates to this book's, 489

 validating, 178–179

URL-shortening services, 449

user interface properties (CSS), 488

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, 266

UTC, 266, 283

V

validating

- email addresses, 166–170, 173–176
- file names, 178–179
- forms, 151–155, 212
- JavaScript, 494
- phone numbers, 190–191
- strings, 171, 189–191
- URLs, 178–179
- vs. verifying, 169
- Web pages, 17, 450
- zip codes, 162–165

validator.w3.org, 17, 450

value attribute, 134

valueOf() method, 186

values

- adding, 15
- assigning to variables, 16
- binary, 70, 71
- checking variables against multiple, 43
- comparing, 16, 70
- converting kilometers to miles, 446
- converting RGB to hexadecimal, 444–445
- defined, 15
- 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, 435
- naming, 15, 182, 479
- use of equals sign with, 15

Venkman Debugger, 490–491

verifying email addresses, 169

video training resources, 493

videos

- Ajax Essential Training*, 493
- JavaScript Essential Training*, 493

Vista. *See* Windows Vista

visual effects properties (CSS), 487

visual formatting properties (CSS), 486

visually impaired users, 296

void() method, 432, 435

W

W3C

- deprecation of attributes by, 24
- and DOM-2, 242
- and DOM-3, 243
- and DOM scripting, 42
- and **innerHTML** property, 28
- and node manipulation, 241, 242, 243
- validation tool, 17, 450

Web

- browsers. *See* Web browsers
- dramatic changes in, 325
- dynamic nature of, 1
- standard layout language for, 17
- standards. *See* Web standards

Web 2.0, 328, 343, 377

Web-based email, 10

Web-based slideshows, 408–410

Web browsers. *See also* specific browsers

- and Ajax, 329, 348
- and alert boxes, 32
- and browser detection, 58
- and caching, 348
- and case-sensitivity, 480
- and cookies, 219–220
- and Daylight Savings Time, 271
- and DOM, 242, 438
- and ECMAScript, 458

- and event handlers, 69
 - and external JavaScript files, 28
 - and JavaScript toolkits, 373
 - and JavaScript versions, 454
 - performing word lookups in, 436–438
 - and pop-up windows, 129
 - and resizing of images, 90
 - and rollovers, 84, 90
 - and security problems, 431
 - testing scripts in different, 130
 - viewing document tree structure in, 13
 - and Year 2000 Problem, 277
- WebKit, 491
- weblogs
- Ajaxian*, 492
 - jQuery, 376
 - jScript, 491
 - Mozilla Hacks*, 491
 - QuirksMode*, 492
 - Surfin' Safari*, 491
- Web page hit counters, 230
- Web pages. *See also* Web sites
- adding interactivity to, 1, 6, 68–70, 369–373
 - adding visual interest to, 81
 - changing background color for, 432, 433–435
 - changing styles for, 433–435
 - characteristics of effective, 261
 - mailing, 451
 - preventing hijacking of, 117
 - reading information from opened, 7
 - reloading part of, 67
 - shortening URLs for, 449
 - standards-compliant, 17, 450
 - tree structure for, 13, 243
 - unobtrusive scripting of, 41, 42
 - validating, 17, 450
 - viewing all images on, 439–440
- Web sites. *See also* Web pages
- alerting users to new content on, 235–240
 - presenting slideshows on, 108
 - for specific topics/companies. *See* resources
- Web standards
- and browser DOMs, 438
 - and JavaScript toolkits, 373
 - validating pages for compliance with, 17, 450
- Web Standards Project, 42
- while** statements, 64–65
- width** attribute, 82
- Wikipedia, 373
- Willison, Simon, 199
- window** event handlers, 196–203
- onabort**, 201
 - onblur**, 203
 - onerror**, 201
 - onfocus**, 202
 - onload**, 14, 197–199
 - onmove**, 201
 - onresize**, 200–201
 - onunload**, 14, 136, 200
- Windows
- and Java, 3
 - and JScript, 455
 - text editor, 20
 - Vista, 431
 - XP Service Pack 2, 431
- windows, 125–132
- adding parameters to, 130
 - alert, 31–32
 - closing, 7
 - elements of standard browser, 127
 - how JavaScript deals with, 115
 - importance of, 115
 - loading different contents into, 131–132
 - opening new, 127–129
 - pop-up, 197, 200, 203
 - sharing functions between, 125–126
- Word, Microsoft. *See* Microsoft Word
- WYSIWYG editors, 20, 41, 138
- X**
- XHTML, and **target** attribute, 118, 119
- XHTML Strict, 118
- XML, 9, 327
- XMLHttpRequest** object, 9, 327, 331, 332, 338, 399

Y

Year 2000 Problem, 277

yellow fade, 377, 378

YUI (Yahoo! User Interface Library), 373

Z

zebra-striped tables, 390–392

z-index, 278

zip codes, 162–165