Julie C. Meloni

Sams Teach Yourself

HTML, CSS and JavaScript

All in One
Contents at a Glance

PART I: Getting Started on the Web
CHAPTER 1: Publishing Web Content
CHAPTER 2: Understanding HTML and XHTML Connections
CHAPTER 3: Understanding Cascading Style Sheets
CHAPTER 4: Understanding JavaScript

PART II: Building Blocks of Practical Web Design
CHAPTER 5: Working with Fonts, Text Blocks, and Lists
CHAPTER 6: Using Tables to Display Information
CHAPTER 7: Using External and Internal Links
CHAPTER 8: Working with Colors, Images, and Multimedia

PART III: Advanced Web Page Design with CSS
CHAPTER 9: Working with Margins, Padding, Alignment, and Floating
CHAPTER 10: Understanding the CSS Box Model and Positioning
CHAPTER 11: Using CSS to Do More with Lists, Text, and Navigation
CHAPTER 12: Creating Fixed or Liquid Layouts

PART IV: Getting Started with Dynamic Web Sites
CHAPTER 13: Understanding Dynamic Websites
CHAPTER 14: Getting Started with JavaScript Programming
CHAPTER 15: Working with the Document Object Model (DOM)
CHAPTER 16: Using JavaScript Variables, Strings, and Arrays
CHAPTER 17: Using JavaScript Functions and Objects
CHAPTER 18: Controlling Flow with Conditions and Loops
CHAPTER 19: Responding to Events
CHAPTER 20: Using Windows and Frames

PART V: Advanced JavaScript Programming
CHAPTER 21: Using Unobtrusive JavaScript
CHAPTER 22: Using Third-Party Libraries
CHAPTER 23: Greasemonkey: Enhancing the Web with JavaScript
CHAPTER 24: AJAX: Remote Scripting

PART VI: Advanced Website Functionality and Management
CHAPTER 25: Creating Print-Friendly Web Pages
CHAPTER 26: Working with Web-Based Forms
CHAPTER 27: Organizing and Managing a Website
CHAPTER 28: Helping People Find Your Web Pages

Index
## Table of Contents

### CHAPTER 1: Publishing Web Content
- A Brief History of HTML and the World Wide Web .......................... 1
- Creating Web Content ........................................................................ 2
- Understanding Web Content Delivery ................................................. 3
- Selecting a Web Hosting Provider ......................................................... 6
- Testing with Multiple Web Browsers .................................................... 8
- Creating a Sample File ....................................................................... 9
- Using FTP to Transfer Files ................................................................. 10
- Distributing Content Without a Web Server ....................................... 18
- Tips for Testing Web Content ............................................................. 19

### CHAPTER 2: Understanding HTML and XHTML Connections
- Getting Prepared .............................................................................. 25
- Getting Started with a Simple Web Page ............................................ 26
- HTML Tags Every XHTML Web Page Must Have ............................ 29
- Organizing a Page with Paragraphs and Line Breaks ....................... 31
- Organizing Your Content with Headings .......................................... 34
- Validating Your Web Content .......................................................... 36
- The Scoop on HTML, XML, XHTML, and HTML5 ......................... 38

### CHAPTER 3: Understanding Cascading Style Sheets
- How CSS Works .................................................................................. 46
- A Basic Style Sheet ........................................................................... 47
- A CSS Style Primer .......................................................................... 52
- Using Style Classes .......................................................................... 57
- Using Style IDs .................................................................................. 59
- Internal Style Sheets and Inline Styles .............................................. 59

### CHAPTER 4: Understanding JavaScript
- Learning Web Scripting Basics .......................................................... 65
- How JavaScript Fits into a Web Page ................................................. 67
- Exploring JavaScript’s Capabilities ................................................... 70
- Displaying Time with JavaScript ....................................................... 71
- Beginning the Script .......................................................................... 71
- Adding JavaScript Statements ............................................................ 72
- Creating Output ................................................................................ 73
- Adding the Script to a Web Page ....................................................... 73
- Testing the Script ............................................................................... 74

### CHAPTER 5: Working with Fonts, Text Blocks, and Lists
- Boldface, Italics, and Special Text Formatting ................................. 81
- Tweaking the Font ............................................................................ 82
- Working with Special Characters ....................................................... 85
- Aligning Text on a Page ................................................................. 89
- The Three Types of HTML Lists ...................................................... 95
- Placing Lists Within Lists ............................................................... 97

### CHAPTER 6: Using Tables to Display Information
- Creating a Simple Table .................................................................... 107
- Controlling Table Sizes .................................................................... 110
- Alignment and Spanning Within Tables ......................................... 113
- Page Layout with Tables ............................................................... 116

### CHAPTER 7: Using External and Internal Links
- Using Web Addresses ....................................................................... 123
- Linking Within a Page Using Anchors ............................................. 126
- Linking Between Your Own Web Content ...................................... 129
- Linking to External Web Content .................................................... 131
- Linking to an Email Address .......................................................... 132
- Opening a Link in a New Browser Window .................................. 134
- Using CSS to Style Hyperlinks ........................................................ 134

### CHAPTER 8: Working with Colors, Images, and Multimedia
- Best Practices for Choosing Colors ................................................. 141
- Understanding Web Colors ............................................................. 143
- Using Hexadecimal Values for Colors ............................................. 145
- Using CSS to Set Background, Text, and Border Colors ................ 146
- Choosing Graphics Software ......................................................... 148
- The Least You Need to Know About Graphics ................................ 149
- Preparing Photographic Images ..................................................... 150
- Creating Banners and Buttons ....................................................... 155
- Reducing the Number of Colors in an Image ................................. 157
- Working with Transparent Images .................................................. 158
- Creating Tiled Backgrounds ............................................................. 159
- Creating Animated Web Graphics .................................................... 160
- Placing Images on a Web Page ....................................................... 161
- Describing Images with Text .......................................................... 163
- Specifying Image Height and Width ............................................... 165
- Aligning Images .............................................................................. 165
CHAPTER 9: Working with Margins, Padding, Alignment, and Floating 191
Using Margins ............................. 192
Padding Elements ........................ 199
Keeping Everything Aligned ........ 203
Understanding the Float Property 204

CHAPTER 10: Understanding the CSS Box Model and Positioning 209
The CSS Box Model ....................... 209
The Whole Scoop on Positioning .... 213
Controlling the Way Things Stack Up 217
Managing the Flow of Text ........ 220

CHAPTER 11: Using CSS to Do More with Lists, Text, and Navigation 225
HTML List Refresher ..................... 226
How the CSS Box Model Affects Lists 226
Placing List Item Indicators ........ 229
Creating Image Maps with List Items and CSS ... 231
How Navigation Lists Differ from Regular Lists 235
Creating Vertical Navigation with CSS ... 236
Creating Horizontal Navigation with CSS ... 245

CHAPTER 12: Creating Fixed or Liquid Layouts 253
Understanding Fixed Layouts ........ 254
Understanding Liquid Layouts ...... 255
Creating a Fixed/Liquid Hybrid Layout 258

CHAPTER 13: Understanding Dynamic Websites 273
Understanding the Different Types of Scripting 273
Including JavaScript in HTML ...... 274
Displaying Random Content ...... 276
Understanding the Document Object Model 280
Changing Images Based on User Interaction 281

CHAPTER 14: Getting Started with JavaScript Programming 287
Basic Concepts .......................... 287
JavaScript Syntax Rules ............... 291
Using Comments ....................... 293
Best Practices for JavaScript ........ 293

CHAPTER 15: Working with the Document Object Model (DOM) 299
Understanding the Document Object Model (DOM) 299
Using window Objects ................ 300
Working with the document Object ... 300
Accessing Browser History .......... 303
Working with the location Object ... 305
More About the DOM Structure ... 306
Working with DOM Nodes .......... 309
Creating Positionable Elements (Layers) 311
Hiding and Showing Objects ...... 316
Modifying Text Within a Page ...... 317
Adding Text to a Page ............... 319

CHAPTER 16: Using JavaScript Variables, Strings, and Arrays 325
Using Variables ......................... 325
Understanding Expressions and Operators 328
Data Types in JavaScript .......... 330
Converting Between Data Types .... 331
Using String Objects ................. 332
Working with Substrings .......... 335
Using Numeric Arrays .............. 337
Using String Arrays ................. 338
Sorting a Numeric Array .......... 340

CHAPTER 17: Using JavaScript Functions and Objects 347
Using Functions ....................... 347
Introducing Objects .................. 352
Using Objects to Simplify Scripting .. 354
Extending Built-in Objects .......... 356
Using the Math Object .............. 360
Working with Math Functions ...... 361
Using the with Keyword .......... 363
Working with Dates .................. 364
<table>
<thead>
<tr>
<th>CHAPTER 18: Controlling Flow with Conditions and Loops</th>
<th>369</th>
</tr>
</thead>
<tbody>
<tr>
<td>The if Statement</td>
<td>369</td>
</tr>
<tr>
<td>Using Shorthand Conditional Expressions</td>
<td>372</td>
</tr>
<tr>
<td>Testing Multiple Conditions with if and else</td>
<td>373</td>
</tr>
<tr>
<td>Using Multiple Conditions with switch</td>
<td>375</td>
</tr>
<tr>
<td>Using for Loops</td>
<td>377</td>
</tr>
<tr>
<td>Using while Loops</td>
<td>379</td>
</tr>
<tr>
<td>Using do-while Loops</td>
<td>380</td>
</tr>
<tr>
<td>Working with Loops</td>
<td>380</td>
</tr>
<tr>
<td>Looping Through Object Properties</td>
<td>382</td>
</tr>
<tr>
<td>CHAPTER 19: Responding to Events</td>
<td>389</td>
</tr>
<tr>
<td>Understanding Event Handlers</td>
<td>389</td>
</tr>
<tr>
<td>Using Mouse Events</td>
<td>394</td>
</tr>
<tr>
<td>Using Keyboard Events</td>
<td>397</td>
</tr>
<tr>
<td>Using the onLoad and onUnload Events</td>
<td>399</td>
</tr>
<tr>
<td>Using onclick to Change &lt;div&gt; Appearance</td>
<td>400</td>
</tr>
<tr>
<td>CHAPTER 20: Using Windows and Frames</td>
<td>409</td>
</tr>
<tr>
<td>Controlling Windows with Objects</td>
<td>409</td>
</tr>
<tr>
<td>Moving and Resizing Windows</td>
<td>413</td>
</tr>
<tr>
<td>Using Timeouts</td>
<td>414</td>
</tr>
<tr>
<td>Displaying Dialog Boxes</td>
<td>417</td>
</tr>
<tr>
<td>Working with Frames</td>
<td>418</td>
</tr>
<tr>
<td>Building a Frameset</td>
<td>420</td>
</tr>
<tr>
<td>Linking Between Frames and Windows</td>
<td>423</td>
</tr>
<tr>
<td>Using Inline Frames</td>
<td>426</td>
</tr>
<tr>
<td>CHAPTER 21: Using Unobtrusive JavaScript</td>
<td>433</td>
</tr>
<tr>
<td>Scripting Best Practices</td>
<td>433</td>
</tr>
<tr>
<td>Reading Browser Information</td>
<td>440</td>
</tr>
<tr>
<td>Cross-Browser Scripting</td>
<td>443</td>
</tr>
<tr>
<td>Supporting Non-JavaScript Browsers</td>
<td>445</td>
</tr>
<tr>
<td>CHAPTER 22: Using Third-Party Libraries</td>
<td>453</td>
</tr>
<tr>
<td>Using Third-Party Libraries</td>
<td>453</td>
</tr>
<tr>
<td>Other Libraries</td>
<td>456</td>
</tr>
<tr>
<td>CHAPTER 23: Greasemonkey: Enhancing the Web with JavaScript</td>
<td>463</td>
</tr>
<tr>
<td>Introducing Greasemonkey</td>
<td>463</td>
</tr>
<tr>
<td>Working with User Scripts</td>
<td>466</td>
</tr>
<tr>
<td>Creating Your Own User Scripts</td>
<td>468</td>
</tr>
<tr>
<td>CHAPTER 24: AJAX: Remote Scripting</td>
<td>479</td>
</tr>
<tr>
<td>Introducing AJAX</td>
<td>479</td>
</tr>
<tr>
<td>Using XMLHttpRequest</td>
<td>483</td>
</tr>
<tr>
<td>Creating a Simple AJAX Library</td>
<td>485</td>
</tr>
<tr>
<td>Creating an AJAX Quiz Using the Library</td>
<td>487</td>
</tr>
<tr>
<td>Debugging AJAX Applications</td>
<td>491</td>
</tr>
<tr>
<td>CHAPTER 25: Creating Print-Friendly Web Pages</td>
<td>499</td>
</tr>
<tr>
<td>What Makes a Page Print-Friendly?</td>
<td>500</td>
</tr>
<tr>
<td>Applying a Media-Specific Style Sheet</td>
<td>503</td>
</tr>
<tr>
<td>Designing a Style Sheet for Print Pages</td>
<td>505</td>
</tr>
<tr>
<td>Viewing a Web Page in Print Preview</td>
<td>508</td>
</tr>
<tr>
<td>CHAPTER 26: Working with Web-Based Forms</td>
<td>513</td>
</tr>
<tr>
<td>How HTML Forms Work</td>
<td>513</td>
</tr>
<tr>
<td>Creating a Form</td>
<td>514</td>
</tr>
<tr>
<td>Accepting Text Input</td>
<td>519</td>
</tr>
<tr>
<td>Naming Each Piece of Form Data</td>
<td>519</td>
</tr>
<tr>
<td>Exploring Form Input Controls</td>
<td>521</td>
</tr>
<tr>
<td>Submitting Form Data</td>
<td>527</td>
</tr>
<tr>
<td>Accessing Form Elements with JavaScript</td>
<td>528</td>
</tr>
<tr>
<td>Displaying Data from a Form</td>
<td>528</td>
</tr>
<tr>
<td>Sending Form Results by Email</td>
<td>530</td>
</tr>
<tr>
<td>CHAPTER 27: Organizing and Managing a Website</td>
<td>537</td>
</tr>
<tr>
<td>When One Page Is Enough</td>
<td>538</td>
</tr>
<tr>
<td>Organizing a Simple Site</td>
<td>540</td>
</tr>
<tr>
<td>Organizing a Larger Site</td>
<td>543</td>
</tr>
<tr>
<td>Writing Maintainable Code</td>
<td>546</td>
</tr>
<tr>
<td>Thinking About Version Control</td>
<td>548</td>
</tr>
<tr>
<td>CHAPTER 28: Helping People Find Your Web Pages</td>
<td>553</td>
</tr>
<tr>
<td>Publicizing Your Website</td>
<td>553</td>
</tr>
<tr>
<td>Listing Your Pages with the Major Search Sites</td>
<td>555</td>
</tr>
<tr>
<td>Providing Hints for Search Engines</td>
<td>556</td>
</tr>
<tr>
<td>Additional Tips for Search Engine Optimization</td>
<td>562</td>
</tr>
<tr>
<td>INDEX</td>
<td>567</td>
</tr>
</tbody>
</table>
About the Author

Julie C. Meloni is the Lead Technologist and Architect in the Online Library Environment at the University of Virginia. Before coming to the library, she worked for more than 15 years in web application development for various corporations large and small in Silicon Valley. She has written several books and articles on Web-based programming languages and database topics, including the bestselling Sams Teach Yourself PHP, MySQL, and Apache All in One.
We Want to Hear from You!

As the reader of this book, you are our most important critic and commentator. We value your opinion and want to know what we’re doing right, what we could do better, what areas you’d like to see us publish in, and any other words of wisdom you’re willing to pass our way.

You can email or write directly to let us know what you did or didn’t like about this book—as well as what we can do to make our books stronger.

Please note that we cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail we receive, we might not be able to reply to every message.

When you write, please be sure to include this book’s title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: feedback@samspublishing

Mail: Sams Publishing
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.
In the previous chapter, you learned the basics of HTML and XHTML, including how to set up a skeletal HTML template for all your web content. In this chapter, you will learn how to fine-tune the display of your web content using Cascading Style Sheets (CSS).

The concept behind style sheets is simple: You create a style sheet document that specifies the fonts, colors, spacing, and other characteristics that establish a unique look for a website. You then link every page that should have that look to the style sheet, instead of specifying all those styles repeatedly in each separate document. Therefore, when you decide to change your official corporate typeface or color scheme, you can modify all your web pages at once just by changing one or two entries in your style sheet rather than changing them in all of your static web files. So, a style sheet is a grouping of formatting instructions that controls the appearance of several HTML pages at once.

Style sheets enable you to set a great number of formatting characteristics, including exacting typeface controls, letter and line spacing, and margins and page borders, just to name a few. Style sheets also enable sizes and other measurements to be specified in familiar units, such as inches, millimeters, points, and picas. You can also use style sheets to precisely position graphics and text anywhere on a web page, either at specific coordinates or relative to other items on the page.

In short, style sheets bring a sophisticated level of display to the Web. And they do so—you’ll pardon the expression—with style.

WHAT YOU’LL LEARN IN THIS CHAPTER:

- How to create a basic style sheet
- How to use style classes
- How to use style IDs
- How to construct internal style sheets and inline styles

NOTE
If you have three or more web pages that share (or should share) similar formatting and fonts, you might want to create a style sheet for them as you read this chapter. Even if you choose not to create a complete style sheet, you’ll find it helpful to apply styles to individual HTML elements directly within a web page.
CHAPTER 3  Understanding Cascading Style Sheets

How CSS Works

The technology behind style sheets is called CSS, which stands for Cascading Style Sheets. CSS is a language that defines style constructs such as fonts, colors, and positioning, which are used to describe how information on a web page is formatted and displayed. CSS styles can be stored directly in an HTML web page or in a separate style sheet file. Either way, style sheets contain style rules that apply styles to elements of a given type. When used externally, style sheet rules are placed in an external style sheet document with the file extension .css.

A style rule is a formatting instruction that can be applied to an element on a web page, such as a paragraph of text or a link. Style rules consist of one or more style properties and their associated values. An internal style sheet is placed directly within a web page, whereas an external style sheet exists in a separate document and is simply linked to a web page via a special tag—more on this tag in a moment.

The cascading part of the name CSS refers to the manner in which style sheet rules are applied to elements in an HTML document. More specifically, styles in a CSS style sheet form a hierarchy in which more specific styles override more general styles. It is the responsibility of CSS to determine the precedence of style rules according to this hierarchy, which establishes a cascading effect. If that sounds a bit confusing, just think of the cascading mechanism in CSS as being similar to genetic inheritance, in which general traits are passed from parents to a child, but more specific traits are entirely unique to the child. Base style rules are applied throughout a style sheet but can be overridden by more specific style rules.

A quick example should clear things up. Take a look at the following code to see whether you can tell what’s going on with the color of the text:

```
<div style="color:green">
  This text is green.
  <p style="color:blue">This text is blue.</p>
  <p>This text is still green.</p>
</div>
```

In the previous example, the color green is applied to the <div> tag via the color style property. Therefore, the text in the <div> tag is colored green. Because both <p> tags are children of the <div> tag, the green text style

NOTE
You might notice that I use the term element a fair amount in this chapter (and I will for the rest of the book, for that matter). An element is simply a piece of information (content) in a web page, such as an image, a paragraph, or a link. Tags are used to code elements, and you can think of an element as a tag complete with descriptive information (attributes, text, images, and so on) within the tag.
cascades down to them. However, the first <p> tag overrides the color style and changes it to blue. The end result is that the first line (not surrounded by a paragraph tag) is green, the first official paragraph is blue, and the second official paragraph retains the cascaded green color.

If you made it through that description on your own, congratulations. If you understood it after I explained it in the text, congratulations to you as well. Understanding CSS isn’t like understanding rocket science, although many people will try to convince you that it is (so that they can charge high consultation fees, most likely!).

Like many web technologies, CSS has evolved over the years. The original version of CSS, known as Cascading Style Sheets Level 1 (CSS1) was created in 1996. The later CSS 2 standard was created in 1998, and CSS 2 is still in use today. All modern web browsers support CSS 2, and you can safely use CSS 2 style sheets without too much concern. So when I talk about CSS throughout the book, I’m referring to CSS 2.

You’ll find a complete reference guide to CSS at http://www.w3.org/Style/CSS/. The rest of this chapter explains how to put CSS to good use.

**A Basic Style Sheet**

Despite their intimidating power, style sheets can be simple to create. Consider the web pages shown in Figure 3.1 and Figure 3.2. These pages share several visual properties that could be put into a common style sheet:

- They use a large, bold Verdana font for the headings and a normal size and weight Verdana font for the body text.
- They use an image named logo.gif floating within the content and on the right side of the page.
- All text is black except for subheadings, which are purple.
- They have margins on the left side and at the top.
- There is vertical space between lines of text.
- The footnotes are centered and in small print.
CHAPTER 3  Understanding Cascading Style Sheets

Listing 3.1 shows the code for the style sheet specifying these properties.
LISTING 3.1  A Single External Style Sheet

body {
    font-size: 10pt;
    font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
    color: black;
    line-height: 14pt;
    padding-left: 5pt;
    padding-right: 5pt;
    padding-top: 5pt;
}

h1 {
    font: 14pt Verdana, Geneva, Arial, Helvetica, sans-serif;
    font-weight: bold;
    line-height: 20pt;
}

p.subheader {
    font-weight: bold;
    color: #593d87;
}

img {
    padding: 3pt;
    float: right;
}

a {
    text-decoration: none;
}

a:link, a:visited {
    color: #8094d6;
}

a:hover, a:active {
    color: #FF9933;
}

div.footer {
    font-size: 9pt;
    font-style: italic;
    line-height: 12pt;
    text-align: center;
    padding-top: 30pt;
}

This might initially appear to be a lot of code, but if you look closely, you’ll see that there isn’t a lot of information on each line of code. It’s fairly standard to place individual style rules on their own line to help make style
sheets more readable, but that is a personal preference; you could put all the rules on one line as long as you kept using the semicolon to separate each rule (more on that in a bit). Speaking of code readability, perhaps the first thing you noticed about this style sheet code is that it doesn’t look anything like normal HTML code. CSS uses a language all its own to specify style sheets.

Of course, the listing includes some familiar HTML tags. As you might guess, body, h1, p, img, a, and div in the style sheet refer to the corresponding tags in the HTML documents to which the style sheet will be applied. The curly braces after each tag name contain the specifications for how all content within that tag should appear.

In this case, the style sheet says that all body text should be rendered at a size of 10 points, in the Verdana font (if possible), with the color black, and 14 points between lines. If the user does not have the Verdana font installed, the list of fonts in the style sheet represents the order in which the browser should search for fonts to use: Geneva, then Arial, and then Helvetica. If the user has none of those fonts, the browser will use whatever default sans serif font is available. Additionally, the page should have left, right, and top margins of 5 points each.

Any text within an <h1> tag should be rendered in boldface Verdana at a size of 14 points. Moving on, any paragraph that uses only the <p> tag will inherit all the styles indicated by the body element. However, if the <p> tag uses a special class named subheader, the text will appear bold and in the color #593d87 (a purple color).

The pt after each measurement in Listing 3.1 means points (there are 72 points in an inch). If you prefer, you can specify any style sheet measurement in inches (in), centimeters (cm), pixels (px), or widths-of-a-letter-m, which are called ems (em).

You might have noticed that each style rule in the listing ends with a semicolon (;). Semicolons are used to separate style rules from each other. It is therefore customary to end each style rule with a semicolon, so you can easily add another style rule after it.

To link this style sheet to HTML documents, include a <link /> tag in the <head> section of each document. Listing 3.2 shows the HTML code for the page shown in Figure 3.1. It contains the following <link /> tag:

<link rel="stylesheet" type="text/css" href="styles.css"/>

**NOTE**

You can specify font sizes as large as you like with style sheets, although some display devices and printers will not correctly handle fonts larger than 200 points.
This assumes that the style sheet is stored under the name styles.css in the same folder as the HTML document. As long as the web browser supports style sheets—and all modern browsers do support style sheets—the properties specified in the style sheet will apply to the content in the page without the need for any special HTML formatting code. This confirms the ultimate goal of XHTML, which is to provide a separation between the content in a web page and the specific formatting required to display that content.

LISTING 3.2 HTML Code for the Page Shown in Figure 3.1

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
'http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd'>

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<title>About BAWSI</title>
<link rel="stylesheet" type="text/css" href="styles.css" /> 
</head>
<body>
<h1>About BAWSI</h1>
<p><img src="logo.gif" alt="BAWSI logo"/>The Bay Area Women's Sports Initiative (BAWSI) is a public benefit, nonprofit corporation with a mission to create programs and partnerships through which women athletes bring health, hope and wholeness to our community. Founded in 2005 by Olympic and World Cup soccer stars Brandi Chastain and Julie Foudy and Marlene Bjornsrud, former general manager of the San Jose CyberRays women's professional soccer team, BAWSI provides a meaningful path for women athletes to become a more visible and valued part of the Bay Area sports culture.</p>
<p class="subheader">BAWSI's History</p>
<p>The concept of BAWSI was inspired by one of the most spectacular achievements in women's sports history and born out of one its biggest disappointments... </p>
<p>a href="secondpage.html">[continue reading] </a></p>
<div class="footer">Copyright &copy; 2005-2009 BAWSI (www.bawsi.org). All rights reserved. Used with permission.</div>
</body>
</html>
```

The code in Listing 3.2 is interesting because it contains no formatting of any kind. In other words, there is nothing in the HTML code that dictates how the text and images are to be displayed—no colors, no fonts, nothing. Yet the page is carefully formatted and rendered to the screen, thanks to the link to the external style sheet, styles.css. The real benefit to this

TIP
In most web browsers, you can view the style rules in a style sheet by opening the .css file and choosing Notepad or another text editor as the helper application to view the file. (To determine the name of the .css file, look at the HTML source of any web page that links to it.) To edit your own style sheets, just use a text editor.

NOTE
Although CSS is widely supported in all modern web browsers, it hasn’t always enjoyed such wide support. Additionally, not every browser’s support of CSS is flawless. To find out about how major browsers compare to each other in terms of CSS support, take a look at this website: http://www.quirksmode.org/css/contents.html.
CHAPTER 3  Understanding Cascading Style Sheets

approach is that you can easily create a site with multiple pages that maintains a consistent look and feel. And you have the benefit of isolating the visual style of the page to a single document (the style sheet) so that one change impacts all pages.

TRY IT YOURSELF

Create a Style Sheet of Your Own

Starting from scratch, create a new text document called mystyles.css and add some style rules for the following basic HTML tags: <body>, <p>, <h1>, and <h2>. After your style sheet has been created, make a new HTML file that contains these basic tags. Play around with different style rules and see for yourself how simple it is to change entire blocks of text in paragraphs with one simple change in a style sheet file.

A CSS Style Primer

You now have a basic knowledge of CSS style sheets and how they are based on style rules that describe the appearance of information in web pages. The next few sections of this chapter provide a quick overview of some of the most important style properties and allow you to get started using CSS in your own style sheets.

CSS includes various style properties that are used to control fonts, colors, alignment, and margins, to name just a few. The style properties in CSS can be generally grouped into two major categories:

- **Layout properties**—Consist of properties that affect the positioning of elements on a web page, such as margins, padding, alignment, and so on
- **Formatting properties**—Consist of properties that affect the visual display of elements within a website, such as the font type, size, color, and so on

Layout Properties

CSS layout properties are used to determine how content is placed on a web page. One of the most important layout properties is the `display` property, which describes how an element is displayed with respect to other elements. There are four possible values for the `display` property:
A CSS Style Primer

- **block**—The element is displayed on a new line, as in a new paragraph.
- **list-item**—The element is displayed on a new line with a list-item mark (bullet) next to it.
- **inline**—The element is displayed inline with the current paragraph.
- **none**—The element is not displayed; it is hidden.

It’s easier to understand the display property if you visualize each element on a web page occupying a rectangular area when displayed—the display property controls the manner in which this rectangular area is displayed. For example, the block value results in the element being placed on a new line by itself, whereas the inline value places the element next to the content just before it. The display property is one of the few style properties that can be applied in most style rules. Following is an example of how to set the display property:
```css
display:block;
```

You control the size of the rectangular area for an element with the **width** and **height** properties. Like many size-related CSS properties, width and height property values can be specified in several different units of measurement:

- **in**—Inches
- **cm**—Centimeters
- **mm**—Millimeters
- **px**—Pixels
- **pt**—Points

You can mix and match units however you choose within a style sheet, but it’s generally a good idea to be consistent across a set of similar style properties. For example, you might want to stick with points for font properties or pixels for dimensions. Following is an example of setting the width of an element using pixel units:
```css
width: 200px;
```

### Formatting Properties

CSS formatting properties are used to control the appearance of content on a web page, as opposed to controlling the physical positioning of the content. One of the most popular formatting properties is the border property,

---

**NOTE**

The display property relies on a concept known as *relative positioning*, which means that elements are positioned relative to the location of other elements on a page. CSS also supports *absolute positioning*, which enables you to place an element at an exact location on a page independent of other elements. You’ll learn more about both of these types of positioning in Part III, “Advanced Web Page Design with CSS.”
which is used to establish a visible boundary around an element with a box or partial box. The following border properties provide a means of describing the borders of an element:

- **border-width**—The width of the border edge
- **border-color**—The color of the border edge
- **border-style**—The style of the border edge
- **border-left**—The left side of the border
- **border-right**—The right side of the border
- **border-top**—The top of the border
- **border-bottom**—The bottom of the border
- **border**—All the border sides

The border-width property is used to establish the width of the border edge. It is often expressed in pixels, as the following code demonstrates:

```css
border-width: 5px;
```

Not surprisingly, the border-color and border-style properties are used to set the border color and style. Following is an example of how these two properties are set:

```css
border-color: blue;
borders-style: dotted;
```

The border-style property can be set to any of the following values:

- **solid**—A single-line border
- **double**—A double-line border
- **dashed**—A dashed border
- **dotted**—A dotted border
- **groove**—A border with a groove appearance
- **ridge**—A border with a ridge appearance
- **inset**—A border with an inset appearance
- **outset**—A border with an outset appearance
- **none**—No border
The default value of the `border-style` property is `none`, which is why elements don’t have a border unless you set the border property to a different style. The most common border styles are the `solid` and `double` styles.

The `border-left`, `border-right`, `border-top`, and `border-bottom` properties enable you to set the border for each side of an element individually. If you want a border to appear the same on all four sides, you can use the single `border` property by itself, which expects the following styles separated by a space: `border-width`, `border-style`, and `border-color`. Following is an example of using the `border` property to set a border that consists of two (double) red lines that are a total of 10 pixels in width:

```css
border:10px double red;
```

Whereas the color of an element’s border is set with the `border-color` property, the color of the inner region of an element is set using the `color` and `background-color` properties. The `color` property sets the color of text in an element (foreground) and the `background-color` property sets the color of the background behind the text. Following is an example of setting both color properties to predefined colors:

```css
color:black;
background-color:orange;
```

You can also assign custom colors to these properties by specifying the colors in hexadecimal (covered in more detail in Chapter 8, “Working with Colors, Images, and Multimedia”) or as RGB (Red, Green, Blue) decimal values, just as you do in HTML:

```css
background-color:#999999;
color:rgb(0,0,255);
```

You can also control the alignment and indentation of web page content without too much trouble. This is accomplished with the `text-align` and `text-indent` properties, as the following code demonstrates:

```css
text-align:center;
text-indent:12px;
```

After you have an element properly aligned and indented, you might be interested in setting its font. The following font properties are used to set the various parameters associated with fonts:

- `font-family`—The family of the font
- `font-size`—The size of the font
CHAPTER 3  Understanding Cascading Style Sheets

- **font-style**—The style of the font (normal or italic)
- **font-weight**—The weight of the font (light, medium, bold, and so on)

The **font-family** property specifies a prioritized list of font family names. A prioritized list is used instead of a single value to provide alternatives in case a font isn’t available on a given system. The **font-size** property specifies the size of the font using a unit of measurement, usually points. Finally, the **font-style** property sets the style of the font and the **font-weight** property sets the weight of the font. Following is an example of setting these font properties:

```css
font-family: Arial, sans-serif;
font-size: 36pt;
font-style: italic;
font-weight: medium;
```

Now that you know a whole lot more about style properties and how they work, refer back at Listing 3.1 and see whether it makes a bit more sense. Here’s a recap of the style properties used in that style sheet, which you can use as a guide for understanding how it works:

- **font**—Lets you set many font properties at once. You can specify a list of font names separated by commas; if the first is not available, the next is tried, and so on. You can also include the words bold and/or italic and a font size. Each of these font properties can be specified separately with **font-family**, **font-size**, **font-weight**, and **font-style** if you prefer.

- **line-height**—Also known in the publishing world as leading. This sets the height of each line of text, usually in points.

- **color**—Sets the text color using the standard color names or hexadecimal color codes (see Chapter 8 for more details).

- **text-decoration**—Useful for turning link underlining off—simply set it to none. The values of underline, italic, and line-through are also supported. The application of styles to links is covered in more detail in Chapter 7, “Using External and Internal Links.”

- **text-align**—Aligns text to the left, right, or center, along with justifying the text with a value of justify.

- **padding**—Adds padding to the left, right, top, and bottom of an element; this padding can be in measurement units or a percentage of the page width. Use padding-left and padding-right if you want to add padding to the left and right of the element independently. Use
Using Style Classes

This is a "teach yourself" book, so you don’t have to go to a single class to learn how to give your pages great style, although you do need to learn what a style class is. Whenever you want some of the text on your pages to look different from the other text, you can create what amounts to a custom-built HTML tag. Each type of specially formatted text you define is called a style class. A style class is a custom set of formatting specifications that can be applied to any element in a web page.

Before showing you a style class, I need to take a quick step back and clarify some CSS terminology. First off, a CSS style property is a specific style that can be assigned a value, such as color or font-size. You associate a style property and its respective value with elements on a web page by using a selector. A selector is used to identify tags on a page to which you apply styles. Following is an example of a selector, a property, and a value all included in a basic style rule:

```css
h1 { font: 36pt Courier; }
```

In this code, h1 is the selector, font is the style property, and 36pt Courier is the value. The selector is important because it means that the font setting will be applied to all h1 elements in the web page. But maybe you want to differentiate between some of the h1 elements—what then? The answer lies in style classes.

Suppose you want two different kinds of <h1> headings for use in your documents. You would create a style class for each one by putting the following CSS code in a style sheet:

```css
h1.silly { font: 36pt Comic Sans; }
h1.serious { font: 36pt Arial; }
```

Notice that these selectors include a period (.) after h1, followed by a descriptive class name. To choose between the two style classes, use the class attribute, like this:

```html
<h1 class="silly">Marvin’s Munchies Inc. </h1>
<p>Text about Marvin’s Munchies goes here. </p>
```
CHAPTER 3  Understanding Cascading Style Sheets

Or you could use this:

```html
<h1 class="serious">MMI Investor Information</h1>
<p>Text for business investors goes here.</p>
```

When referencing a style class in HTML code, simply specify the class name in the `class` attribute of an element. In the previous example, the words *Marvin’s Munchies Inc.* would appear in a 36-point Comic Sans font, assuming that you included a `<link />` to the style sheet at the top of the web page and assuming that the user has the Comic Sans font installed. The words *MMI Investor Information* would appear in the 36-point Arial font instead. You can see another example of classes in action in Listing 3.2; look for the subheader `<p>` class and the footer `<div>` class.

What if you want to create a style class that could be applied to any element, rather than just headings or some other particular tag? You can associate a style class with the `<div>` tag, as in Listing 3.2, which is used to enclose any text in a block that is somewhat similar to a paragraph of text; the `<div>` tag is another useful container element.

You can essentially create your own custom HTML tag by using the `div` selector followed by a period (.) followed by any style class name you make up and any style specifications you choose. That tag can control any number of font, spacing, and margin settings all at once. Wherever you want to apply your custom tag in a page, use a `<div>` tag with the `class` attribute followed by the class name you created.

For example, the style sheet in Listing 3.1 includes the following style class specification:

```css
div.footer {
  font-size: 9pt;
  font-style: italic;
  line-height: 12pt;
  text-align: center;
  padding-top: 30pt;
}
```

This style class is applied in Listing 3.2 with the following tag:

```html
<div class="footer">

```

Everything between that tag and the closing `</div>` tag in Listing 3.2 appears in 9-point, centered, italic text with 12-point vertical line spacing and 30 points of padding at the top of the element.

What makes style classes so valuable is how they isolate style code from web pages, effectively allowing you to focus your HTML code on the actual
content in a page, not how it is going to appear on the screen. Then you can focus on how the content is rendered to the screen by fine-tuning the style sheet. You might be surprised by how a relatively small amount of code in a style sheet can have significant effects across an entire website. This makes your pages much easier to maintain and manipulate.

**Using Style IDs**

When you create custom style classes, you can use those classes as many times as you would like—they are not unique. However, there will be some instances when you want to have precise control over unique elements for layout or formatting purposes (or both). In such instances, look to IDs instead of classes.

*A style ID* is a custom set of formatting specifications that can be applied only to one element in a web page. You can use IDs across a set of pages but only once per time within each page.

For example, suppose you have a title within the body of all your pages. Each page has only one title, but all the pages themselves include one instance of that title. Following is an example of a selector with an ID indicated, plus a property and a value:

```html
p#title {font: 24pt Verdana, Geneva, Arial, sans-serif}
```

Notice that this selector includes a hash mark, or pound sign (#), after `p`, followed by a descriptive ID name. When referencing a style ID in HTML code, simply specify the ID name in the `id` attribute of an element, like so:

```html
<p id="title">Some Title Goes Here</p>
```

Everything between the opening and closing `<p>` tags will appear in 24-point Verdana text—but only once on any given page. You will often see style IDs used to define specific parts of a page for layout purposes, such as a header area, footer area, main body area, and so on. These types of areas in a page will appear only once per page, so using an ID rather than a class is the appropriate choice.

**Internal Style Sheets and Inline Styles**

In some situations, you might want to specify styles that will be used in only one web page, in which case you can enclose a style sheet between `<style>` and `</style>` tags and include it directly in an HTML document.
CHAPTER 3  Understanding Cascading Style Sheets

Style sheets used in this manner must appear in the <head> of an HTML document. No <link /> tag is needed, and you cannot refer to that style sheet from any other page (unless you copy it into the beginning of that document, too). This kind of style sheet is known as an internal style sheet, as you learned earlier in the chapter.

Listing 3.3 shows an example of how you might specify an internal style sheet.

LISTING 3.3  A Web Page with an Internal Style Sheet

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
 "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
  <title>Some Page</title>
  <style type="text/css">
    div.footer {
      font-size: 9pt;
      line-height: 12pt;
      text-align: center;
    }
  </style>
</head>
<body>
  ...
  <div class="footer">
    Copyright 2009 Acme Products, Inc.
  </div>
</body>
</html>
```

In the listing code, the `div.footer` style class is specified in an internal style sheet that appears in the head of the page. The style class is now available for use within the body of this page. And, in fact, it is used in the body of the page to style the copyright notice.

Internal style sheets are handy if you want to create a style rule that is used multiple times within a single page. However, in some instances you might need to apply a unique style to one particular element. This calls for an inline style rule, which allows you to specify a style for only a small part of a page, such as an individual element. For example, you can create and apply a style rule within a `<p>`, `<div>`, or `<span>` tag via the `style` attribute. This type of style is known as an *inline style* because it is specified right there in the middle of the HTML code.

**NOTE**

`<span>` and `</span>` are dummy tags that do nothing in and of themselves except specify a range of content to apply any style attributes that you add. The only difference between `<div>` and `<span>` is that `<div>` is a block element and therefore forces a line break, whereas `<span>` doesn’t. Therefore, you should use `<span>` to modify the style of any portion of text that is to appear in the middle of a sentence or paragraph without any line break.
Here’s how a sample style attribute might look:

```html
<p style="color:green">
  This text is green, but <span style="color:red">this text is red.</span>
</p>
Back to green again, but...
<p>
...now the green is over, and we’re back to the default color for this page.
</p>
```

This code makes use of the `<span>` tag to show how to apply the `color` style property in an inline style rule. In fact, both the `<p>` tag and the `<span>` tag in this example use the `color` property as an inline style. What’s important to understand is that the `color:red` style property overrides the `color:green` style property for the text appearing between the `<span>` and `</span>` tags. Then in the second paragraph, neither of the `color` styles applies because it is a completely new paragraph that adheres to the default color of the entire page.

To validate your stylesheet, you can use a CSS validation tool such as the W3C CSS Validator. Just like the validation tool discussed in Chapter 2, “Understanding HTML and XHTML Connections,” you can point the tool to a web address, upload a file, or paste content into the form field provided. The ultimate goal is a result such as that shown in Figure 3.3: valid!

![W3C CSS Validator](image.png)

**FIGURE 3.3**
The W3C CSS Validator shows there are no errors in the style sheet contents of Listing 3.1.
CHAPTER 3  Understanding Cascading Style Sheets

Summary

In this chapter, you learned that a style sheet can control the appearance of many HTML pages at once. It can also give you extremely precise control over the typography, spacing, and positioning of HTML elements. You also discovered that by adding a style attribute to almost any HTML tag, you can control the style of any part of an HTML page without referring to a separate style sheet document.

You learned about three main approaches to including style sheets in your website: a separate style sheet file with the extension .css that is linked to in the <head> of your documents, a collection of style rules placed in the head of the document within the <style> tag, and as rules placed directly in an HTML tag via the style attribute.

Table 3.1 summarizes the tags discussed in this chapter. Refer to the CSS 2 style sheet standards at http://www.w3c.org for details on what options can be included after the <style> tag or the style attribute.

<table>
<thead>
<tr>
<th>Tag/Attributes</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;style&gt;...&lt;/style&gt;</td>
<td>Allows an internal style sheet to be included within a document. Used between &lt;head&gt; and &lt;/head&gt;.</td>
</tr>
<tr>
<td>Attribute</td>
<td></td>
</tr>
<tr>
<td>type=&quot;contenttype&quot;</td>
<td>The Internet content type. (Always &quot;text/css&quot; for a CSS style sheet.)</td>
</tr>
<tr>
<td>&lt;link /&gt;</td>
<td>Links to an external style sheet (or other document type). Used in the &lt;head&gt; section of the document.</td>
</tr>
<tr>
<td>Attribute</td>
<td></td>
</tr>
<tr>
<td>href=&quot;url&quot;</td>
<td>The address of the style sheet.</td>
</tr>
<tr>
<td>type=&quot;contenttype&quot;</td>
<td>The Internet content type. (Always &quot;text/css&quot; for a CSS style sheet.)</td>
</tr>
<tr>
<td>rel=&quot;stylesheet&quot;</td>
<td>The link type. (Always &quot;stylesheet&quot; for style sheets.)</td>
</tr>
<tr>
<td>&lt;span&gt;...&lt;/span&gt;</td>
<td>Does nothing but provide a place to put style or other attributes. (Similar to &lt;div&gt;...&lt;/div&gt; but does not cause a line break.)</td>
</tr>
<tr>
<td>Attribute</td>
<td></td>
</tr>
<tr>
<td>style=&quot;style&quot;</td>
<td>Includes inline style specifications. (Can be used in &lt;span&gt;, &lt;div&gt;, &lt;body&gt;, and most other HTML tags.)</td>
</tr>
</tbody>
</table>
Q&A

Q. Say I link a style sheet to my page that says all text should be blue, but there’s a `<span style="font-color:red">` tag in the page somewhere. Will that text display as blue or will it display as red?

A. Red. Local inline styles always take precedence over external style sheets. Any style specifications you put between `<style>` and `</style>` tags at the top of a page will also take precedence over external style sheets (but not over inline styles later in the same page). This is the cascading effect of style sheets that I mentioned earlier in the chapter. You can think of cascading style effects as starting with an external style sheet, which is overridden by an internal style sheet, which is overridden by inline styles.

Q. Can I link more than one style sheet to a single page?

A. Sure. For example, you might have a sheet for formatting (text, fonts, colors, and so on) and another one for layout (margins, padding, alignment, and so on)—just include a `<link />` for both. Technically speaking, the CSS standard requires web browsers to give the user the option to choose between style sheets when multiple sheets are presented via multiple `<link />` tags. However, in practice, all major web browsers simply include every style sheet. The preferred technique for linking in multiple style sheets involves using the special `@import` command. Following is an example of importing multiple style sheets with `@import`:

```css
@import url(styles1.css);
@import url(styles2.css);
```

Similar to the `<link />` tag, the `@import` command must be placed in the head of a web page. You learn more about this handy little command in Chapter 25, “Creating Print-Friendly Web Pages,” when you learn how to create a style sheet specifically for printing web pages.

Workshop

The workshop contains quiz questions and exercises to help you solidify your understanding of the material covered. Try to answer all questions before looking at the “Answers” section that follows.
CHAPTER 3 Understanding Cascading Style Sheets

Quiz

1. What code would you use to create a style sheet to specify 30-point blue Arial headings and all other text in double-spaced, 10-point blue Times Roman (or the default browser font)?

2. If you saved the style sheet you made for Question 1 as corporate.css, how would you apply it to a web page named intro.html?

3. How many different ways are there to ensure style rules can be applied to your content?

Answers

1. Your style sheet would include the following:

   h1 { font: 30pt blue Arial; }
   body { font: 10pt blue; }

2. Put the following tag between the <head> and </head> tags of the intro.html document:

   <link rel="stylesheet" type="text/css" href="corporate.css" />

3. Three: externally, internally, and inline.

Exercises

- Using the style sheet you created earlier in this chapter, add some style classes to your style sheet. To see the fruits of your labor, apply those classes to the HTML page you created as well. Use classes with your <h1> and <p> tags to get the feel for things.

- Develop a standard style sheet for your website and link it into all your pages. (Use internal style sheets and/or inline styles for pages that need to deviate from it.) If you work for a corporation, chances are it has already developed font and style specifications for printed materials. Get a copy of those specifications and follow them for company web pages, too.

- Be sure to explore the official style sheet specs at http://www.w3.org/Style/CSS/ and try some of the more esoteric style properties not covered in this chapter.
INDEX

**Symbols**

/ (forward slashes), HTML and, 124
+ (plus signs) in JavaScript statements, 73
+= operator, 327
(;) semicolons, JavaScript statements, 72, 287

**Numbers**

140 cross-browser color names, 143-144

**A**

A Small Orange web hosting provider, 7
<a> tags (HTML), 170
anchor locations, linking to, 126-129
naming, 127
web pages, identifying locations within, 126
absolute addresses, web pages and, 124-125
absolute links, web pages and, 124
absolute positioning, 213
display property (CSS), 53
positioning property, 214-217
accessibility, JavaScript best practices, 439
Adaptive Path, AJAX, 480
addEventListener function, 392
Adjust Hue/Lightness/Saturation tool (GIMP), 154
Adobe Photoshop, 148
AJAX, 71, 479. See also JavaScript
AJAX Frameworks JavaScript library, 456
debugging applications, 491-496
examples of, 481
frameworks, 482
libraries, 482
ajaxRequest function, 486
ajaxResponse function, 486
creating, 485-486
quiz building example, 487-491
using, 486
limitations of, 482
live search forms, creating, 496
front end, 495
HTML file example, 493
HTML forms, 492
JavaScript front end, 494-495
PHP back end, 493-494
quiz building example
HTML files, 487-488
JavaScript files, 489-490
testing, 490-491
XML files, 488-489
requests
awaiting responses, 484
back end, 480, 493-494
creating, 483
front end, 480, 494-495
interpreting response data, 484
JavaScript client, 480
sending, 484
server-side scripts, 480, 493-495
XML and, 481
XMLHttpRequest
awaiting responses to requests, 484
creating requests, 483
interpreting request response data, 484
opening URLs, 483
sending requests, 484
alert() function, event handlers and, 295
alerts (dialog boxes), 417-418
aligning
align property (CSS), 191, 203-204
images
horizontal alignment, 165-166
vertical alignment, 167-168
text
attributes, 92-93
block-level elements, 93-95
paragraphs, 93-95
tables, 113-115
text-align property (CSS), 56
text-align style rule (CSS), 93-95
text-decoration property (CSS), 56
ASCII text, 26, 34, 82, 397
.asp file extensions, 27
ASP scripting language, 274
attributes, 92-93, 123
auto image loading, web browsers, 20

B
<b> tags (HTML), 81-83
back end (AJAX requests), 480, 493-494
Back/Forward buttons, adding to documents, 304-305
backgrounds
background-position style property, 172
background-repeat style property, 172
colors,
background-color property (CSS), 55
background-color style property, 171
CSS and 146-148
images, 171-172
tiled backgrounds, 159-160
bad website examples, 144
bandwidth, web hosting providers, 6
banners, creating, 156
Barry's Clipart Server website, 149
BAWSI.org, website organization, 545
behavior, Javascripting scripting best practices, 434
Berners-Lee, Sir Tim, 1
<br /> tags, 32-33, 85
break statements (JavaScript), 381
browsers (web)
140 cross-browser color names, 143-144
cross-browser scripting
debugging browsers, 444
feature sensing, 437, 443-444
CSS, support for, 51
debugging, 444
development of, 2
dialog boxes, displaying, 417-418
distributing, 18
Firefox, 9, 464
frames, 418-420
   adding individual frames, 423
   creating frameset documents, 421-423
   frames array, 426
   inline frames, 426-429
   JavaScript objects and, 425
   linking windows to frames, 423-424
   modifying borders, 424-425
   nested framesets, 425
Google Chrome, 9, 465
history, accessing, 2, 303-305
HTML development, 2
images, auto image loading, 20
information, reading via JavaScript
dishonest browsers, 442-443
displaying information, 440-441
Internet Explorer, 9
event properties, 393
Trixie, 465-468
links, opening in new browser
tables, 134
lists, displaying in, 97
margins and, 226-229
non-Internet Explorer event properties, 393-394
non-JavaScript browsers
   avoiding errors, 446-449
detecting, 445-446
JavaScript optionality, 446
<noscript> tag, 445-446
<script> tag, 457
non-viewable window areas, 254
Opera, 9, 465
padding and, 226-228
pop-up windows, 134
popularity of, 26
Safari, 9
search engines, 445
sensing. See feature sensing
servers, basic browser server interaction, 3-5
text, adjusting font size
settings, 20
websites
   comparing, 26
testing, 8-9, 26
windows
   creating, 410-411
   linking frames to windows, 423-424
   moving, 413-414
   opening/closing, 411-412
   resizing, 413-414
   timeouts, 414-416
built-in objects (JavaScript), 289, 352, 356-359
date object
   converting date formats, 366
   creating, 364
   local time values, 365-366
   reading date values, 365
   setting date values, 364-365
time zones, 365
definitions, extending, 359
math object, 361
generating random numbers, 360-363
rounding decimal numbers, 360
truncating decimal numbers, 360
buttons, creating, 156
buying domain names, 6
color
   140 cross-browser color names, 143-144
   analogous color schemes, 143
   background color
      background-color style property, 171
   CSS and, 146-148
   best practices, 141-143
   border color (tables), CSS and, 146-148
   color property (CSS), 56
Color Scheme Generator website, 143
color style rule (CSS), fonts and, 86-87
color theory, 142
color wheel, 142
Colorblind Web Page Filter tool, 148
complementary color schemes, 143
graphics, adjusting color in, 154
decimal color codes, 144-146
links and, 144
lists, styling, 227-228
monitors and, 144
tables and, 115
text
  CSS and, 146-148
  formatting in style sheets, 46
triadic color schemes, 143
using, 141-143
W3C color standards, 143
columns, fixed/liquid hybrid layouts
defining, 260-262
height, setting, 264-268
combining
  string object values, 332-334
tasks. See functions (JavaScript)
comments
  HTML, JavaScript and, 276
  JavaScript, 293
websites, maintaining code via, 546-547
complementary color schemes, 143
compression
  graphics, 150
  JPEG, 155
conditional expressions, 317, 370-373
conditional operators (JavaScript), 370
conditional statements (JavaScript), 289
conditions (for loops), 377
confirmations (dialog boxes), 417-418
constructor functions (JavaScript), 354
containers, 307
content (web)
  creating, 2-3
  CSS box model, 210
delivering, 3-5
  Javascripting scripting best practices, 434
  publishing
    locally, 18-19
to blogs, 19
  viewing locally, 5
  web hosting providers, selecting, 6-8, 26
continue statements (JavaScript), 382
continuing loops (JavaScript), 382
control panels, selecting web hosting providers, 7-8
converting
  data formats (JavaScript date objects), 366
data types (JavaScript), 331
  string case (JavaScript), 334
copyrights, graphics and, 149
counters (JavaScript), for loops, 377
Creammonkey add-on, Greasemonkey support in Safari, 465
Creative Commons licenses, 149
cropping images, 151-152
cross-browser scripting
  debugging browsers, 444
event handlers, JavaScript and, 437
  feature sensing, 437, 443-444
CSS
  align property, 191, 203-204
  box model, 209
    borders, 210
    content, 210
    lists and, 226-229
    margins, 210
    padding, 210
    sizing elements, 210-212
  cascading component of, 46
clear property, text flow and, 220
color, specifying via CSS, 146-148
CSS 2, 47
CSS Zen Garden, 191-192, 253
CSS1, 47
definition of, 45-46
<div> tags, 46
DOCTYPE declarations, 212
float property, 191, 204-207, 220
formatting properties, 53
  background-color property, 55
  border property, 54-55
  border-bottom property, 54-55
  border-color property, 54-55
  border-left property, 54-55
  border-right property, 54-55
  border-style property, 54-55
  border-top property, 54-55
  border-width property, 54
  color property, 56
  font property, 56
  font-family property, 55
  font-size property, 55
  font-style property, 56
  font-weight property, 56
  line-height property, 56
  padding property, 56
  text-align property, 55-56
text-decoration property, 56
text-indent property, 55
image maps, creating, 231-235
inline styles, 60-61
layouts
  display property, 52-53
  fixed layouts, 254
fixed/liquid hybrid layouts, 258-268
height property, 53
liquid layouts, 253-257
width property, 53
<link /> tag, 50
links, styling, 134-138
lists
  horizontal navigation, 245-248
  list-style-image property, 226
  list-style-position property, 226, 229-231
  list-style-type property, 226
  navigation lists, 236-248
  vertical navigation, 236-244
margin property, 191-199
media-specific style sheets, 503-504
overflow property, text flow and, 220
padding property, 191, 199-202
positioning
  absolute positioning, 213-217
  overlapping elements, 213, 217-219
  positioning property, 213
  relative positioning, 213-215
  z-index property, 217-219
print pages, designing style sheets for, 505-508
properties, hyphenating, 312
reference guide online resource, 47
selectors, 57
style classes, 57-58
style IDs, 59
style properties, 57
style rules, 46, 50
color style rule, 86-87
font weight style rule, 83
font-family style rule, 86
font-size style rule, 86-87
list-style-type style rule, 99-101
multiple style properties in, 58
text-align style rule, 93-95
viewing, 51
style sheets
  creating, 47-52
definition of, 45
  external style sheets, 46-51
  formatting properties, 53-56
  formatting text color, 46
  inline styles and, 61
  internal style sheets, 46, 59-60
  italic text, 56
  layout properties, 52-53
  line-through text, 56
  linking to HTML documents, 50
  sizing text, 50
  strikethrough text, 56
  underline text, 56
tags, 50
  elements and, 46
  selectors, 57
validating, 61
web browsers, CSS support, 51
z-index property, 217-219
current line, text flow and, 220
custom HTML tags, creating, 58
custom objects (JavaScript), 289
customer service, web hosting providers, 6
Cyberduck FTP client, 11
data types (JavaScript), 330-331
date object (JavaScript)
  creating, 364
date formats, converting, 366
date values, reading, 365
date values, setting, 364-365
local time values, 365-366
time zones, 365
Date objects (JavaScript), time display example, 72
debugging
  AJAX applications, 491-496
  browsers, 444
  user scripts, 474-476
decimal numbers
  rounding, 360
  truncating, 360
declaring variables (JavaScript), 326
decrementing/incrementing variables (JavaScript), 327
definition lists, 96, 226
design patterns, JavaScript best practices, 439
dialog boxes, displaying, 417-418
Digg, 481
directories (web content), 123-124
display property (CSS), 52-53
displaying
document information, 301-302
time (JavaScript example)
  adding scripts to web pages, 73-74
  creating output, 73
  Date objects, 72
  error handling, 76-78
  modifying scripts, 74-76
<script> tags, 71-72
  statements, 72
testing scripts, 74
  variables, 72

D

DailyRazor web hosting provider, 7
dashed value (CSS border-style properties), 54
DisplayKey function, keyboard events, 398-399

distributing web browsers, 18
dithering, 158
<div> tags (CSS), 46, 170
do, while loops (JavaScript), 380
DOCTYPE declarations, 212
document objects (DOM), 300
  anchor objects, 303
document information, displaying, 301-302
  link objects, 303, 306
  methods of, 302, 310
  properties of, 301
  text, writing within documents, 302
document roots, 13-16
document.write statements
  (JavaScript), 68, 73
documenting code, 546-547
documents
  anchors, 303
  Back/Forward buttons, adding, 304-305
  information, displaying in, 301-302
  links, 303, 306
  text, writing within documents, 302
Dojo JavaScript library, 456
DOM, 280-281
  children, 308
  layers
    controlling positioning via JavaScript, 311-315
    creating, 311
    moveable layers, 311-315
  nodes, 307, 309-310
  objects, 352
    anchor objects, 303
    document objects, 300-303, 306
    hiding/showing, 316-317
    history objects, 303-305
  link objects, 303, 306
  location objects, 305-306
  methods, 299
  naming, 299
  parents, 308
  properties, 299
  referencing, 299
  showing/hiding, 316-317
  siblings, 308
  window objects, 300, 409-418
  parents, 308
  siblings, 308
  structure of, 306-307
  text, web pages
    adding to, 319-321
    modifying in, 317-318
DOM objects (JavaScript), 289
domain names, purchasing, 6
dotted value (CSS border-style properties), 54
double value (CSS border-style properties), 54-55
downloading JavaScript libraries, 457
dynamic websites
  client-side scripting, 274
  DOM, 280-281
  images, changing via user interaction, 281-283
  server-side scripting, 274
  text, printing via JavaScript, 275
  web content, displaying random content via JavaScript, 276-280
else keyword (JavaScript), testing multiple conditions, 372-373
HTML files, 374
JavaScript files, 374-375
<em> tags (HTML), 83
e-mail
  addresses
    email address encoders, 133
    linking to, 132-133
    form data, sending via, 530-531
<embed> tags (XHTML), 183
embedded multimedia files, 180-183
emphasized text. See italic text
empty tags (HTML), 30, 33
error handling, 448
  JavaScript scripts, 76-78
  non-JavaScript browsers, 446-449
error messages, displaying, 77
escaping loops (JavaScript), 381
ESPN.com, 254, 540-542
European languages, formatting text for, 89-91
event handlers, 389
  creating, 390-391
  defining, 390-391
  event objects, 392
  functions and, 484
  JavaScript and, 68-70, 290-291, 435
    alert() function, 295
    best practices, 294-295
    cross-browser scripting, 437
    W3C event model, 436-437
keyboard events, 397-399
mouse events
  mousestatus function, 396
  onClick, 394-397, 400-404
  onDblClick, 395
  onMouseDown, 395-397
  onMouseOut, 394
  onMouseOver, 394
E

effect (JavaScript), 70
elements, definition of, 46
formatting

Flicker, 161-163
float property (CSS), 191, 204-207, 220
float style property, 166
flow control (JavaScript), 369
break statements, 381
case statements, 376
continue statements, 382
do, while loops, 380
for loops, 377-378
for, in loops, 382-384
if statements, 369
  conditional expressions, 370-373
  conditional operators, 370
  else keyword, 372-375
  logical operators, 371
  testing multiple conditions, 373-375
infinite loops, 380-381
loops
  break statements, 381
  continue statements, 382
  escaping, 381
  switch statements, using multiple conditions, 375-376
  while loops, 379-380
flowing text, 220
fluid layouts. See liquid layouts
folders (web content), 123-125
<font> tags (HTML), 81, 86
fonts (text)
  Arial font, 86
  CSS
    color style rule, 86-87
    font property, 56
    font-family property, 55
    font-family style rule, 86
    font-size property, 55
    font-size style rule, 86-87
    font-style property, 56
    font-weight property, 56
    font weight style rule, 83
    foreign languages, 89-91
    HTML, customizing in, 85-89
    resumes, creating, 87-89
    sans-serif font, 86
    sizing, style sheets, 50
    special characters, 89-91
    Times Roman font, 86
    typewriter font. See monospaced text
    web browsers, adjusting font size settings, 20
for loops (JavaScript), 377-378
for statements (JavaScript), 290
for, in loops (JavaScript), 382-384
foreign languages, formatting text for, 89-91
formatting
  CSS, 53
    background-color property, 55
    border property, 54-55
    border-bottom property, 54-55
    border-color property, 54-55
    border-left property, 54-55
    border-right property, 54-55
    border-style property, 54-55
    border-top property, 54-55
    border-width property, 54
    color property, 56
    font property, 56
    font-family property, 55
    font-size property, 55
    font-style property, 55
    font-weight property, 56
    font-weight style property, 56
    line-height property, 56
    padding property, 56
    text-align property, 55-56
    text-decoration property, 56
    text-indent property, 55
style sheets, 46, 50

- text, 82
  - aligning text, 92-95
  - big text, 83
  - boldface text, 82-83
  - color, style sheets, 46
  - customizing fonts in HTML, 85-89
- definition lists, 96, 226
- foreign languages, 89-91
- italic text, 82-83
- monospaced text, 84-85
- multilevel lists, 100-101
- nested lists, 97-100, 226
- older HTML tags, 81
- ordered lists, 95-96, 226
- outlines, 98-100
- resumes, creating, 87-89
- sizing, style sheets, 50
- small text, 83
- special characters, 89-91
- subscript text, 83
- superscript text, 83
- unordered lists, 95-96, 226
- web page creation, 33-35

forms, 513

- check boxes, 521-523
- creating, 514-519
- data
  - displaying in pop-up windows, 528-530
  - naming, 519-520
  - sending via email, 530-531
  - submitting, 527
- elements, accessing via JavaScript, 528
- hidden data, 520
- JavaScript events, 527-528
- pull-down pick lists, 524-525
- radio buttons, 523-524
- scrolling lists, 524-525

constructor functions, 354

- defining, 347-349
- function call statements, 288
- math functions, 361-363
- naming, 292
- parameters, 288, 348
- returning values, 351-352

G

Garrett, Jesse James, 480

- get methods (JavaScript objects), 365
- Gickr, animated graphics, 161
- GIFs, 157-158
  - animated graphics, 160-161
  - tiled backgrounds, 159-160
  - transparent images, 159
- GIMP, 149
  - Adjust Hue/Lightness/Saturation tool, 154
  - banners, creating, 156
  - buttons, creating, 156
  - images
    - adjusting color, 154
    - cropping, 151-152
    - JPEG compression, 155
    - resizing, 153
    - Red Eye Removal, 154
- Git website version control, 550
- global variables (JavaScript), 326
- GMT (Greenwich Mean Time), 71
  - JavaScript time displaying example, 71
- Google
  - Chrome, 9
    - displaying error messages, 77
  - Greasemonkey, 465
  - Gmail, 446, 481
  - Images, 161
  - listing websites with, 555
Picasa, 149
searches, 4
graceful degradation, web design and, 434

graphics
Adobe Photoshop, 148
aligning
horizontal alignment, 165-166
vertical alignment, 167-168
animated graphics, 160-161
backgrounds
background-image style property, 172
graphics, 171-172
tiled backgrounds, 159-160
banners, creating, 156
buttons, creating, 156
CD-ROM, transferring graphics to, 150
clip art, 149
color, adjusting, 154
compression, 150
copyrights and, 149
Creative Commons licenses, 149
cropping, 151-152
dithering, 158
file sizes, 150
Flickr, 161-163
GIFs, 157-158
animated graphics, 160-161
tiled backgrounds, 159-160
transparent images, 159
GIMP, 149
adjusting image color, 154
banners, 156
buttons, 156
cropping images, 151-152
JPEG compression, 155
resizing images, 153
Google Images, 161
Google Picasa, 149

height/width, specifying, 165
image maps, 173-178
JPEGs
  compression, 155
tiled backgrounds, 159-160
links, turning graphics into, 169-171
Picnik, 149
PNGs, 158-159
Red Eye Removal, 154
republishing, 163
resizing, 153
resolution, 150
rollover images, 394
software, choosing, 149
text descriptions, 163-164
transparent graphics, 158
uses of, 150
web pages
  grabbing graphics from, 149
  placing graphics on web pages, 161-163

Greasemonkey, 463
API functions, 471
browser support, 465
installing, 464
metadata, 470
turning on/off, 468
user scripts
  creating, 468, 475
debugging user scripts, 474-476
describing, 469-470
finding, 466
installing, 466
managing, 466-467
metadata and, 469-470
site-specific user scripts, 472-473
testing, 468-471
text area macro user scripts, 475-476
groove value (CSS border-style properties), 54
grouping statements (JavaScript). See loops

H

<head> tags, 27, 31-33, 68
heading tags (HTML), 34-36
headings (tables), creating, 108
height
  CSS box model, adjusting height in, 210-212
  fixed/liquid hybrid layouts, setting column height in, 264-268
  height property (CSS), 53
  images, specifying height in, 165
Hello World HTML file, creating, 9-10
help
  CSS, reference guide online resource, 47
  web hosting providers, selecting, 6
helper applications, 180
hexadecimal color codes, 144-146
hidden form data, 520
hiding/showing DOM objects, 316-317
history objects (DOM), 303-305
horizontal image alignment, 165-166
horizontal navigation, 245-248
horizontal rule tags (HTML), 33
<hr /> tags, 33
HTML
AJAX
  live search forms, 492
  quiz building example, 487-488
attributes, 92-93, 123
comments, JavaScript and, 276
containers, 307
CSS
CSS box model, 209-212
external style sheets, 51
linking style sheets to HTML documents, 50
development of, 1-2, 38
event handlers
JavaScript, 435-437
Yahoo! UI Library, 437
file creation
boilerplate code, 28-30
comparing web page HTML codes, 37
formatting text, 33-35
html file extensions, 27
HTML tags, 27-33
HTML-compatible word processors, 27
indenting text, 35
line breaks, 32-33
naming files with HTML tags, 27
Notepad, 26
organizing content via headings, 34-36
overview of, 29
paragraphs, 32-33
saving files with HTML tags, 27
templates, 31
TextEdit, 27
Word, 27
WYSIWYG editors, 27
files
creating, 9-10
index pages, 16-18
managing, 14-16
organizing, 14-16
transferring, 10-13, 29
viewing, 29
forms, 513
accessing elements via JavaScript, 528
check boxes, 521-523
creating, 514-519
displaying data in pop-up windows, 528-530
hidden data, 520
JavaScript events, 527-528
naming form data, 519-520
pull-down pick lists, 524-525
radio buttons, 523-524
scrolling lists, 524-525
selection lists, 524-525
sending data via email, 530-531
submitting form data, 527
text areas, 526
text fields, 526
text input, 519
user input, 519
validating, 531-532
forward slashes (/), 124
frames, 418-420
adding individual frames, 423
creating frameset documents, 421-423
frames array, 426
JavaScript objects and, 425
linking windows to frames, 423-424
modifying borders, 424-425
nested framesets, 425
FTP
selecting FTP clients, 10-11
using FTP clients, 11-13
future of, 28
graphics, image maps, 176-178
<head> tags, functions and, 68
Hello World sample file, creating, 9-10
history of, 1-2
html file extensions, 27
HTML-compatible word processors, creating HTML files, 27
HTML4, empty tags, 33
HTML5, 28, 40
images, placing on web pages, 162-163
JavaScript and, 274
adding libraries to HTML documents, 457
adding scripts to HTML documents, 73-74
changing images via user interaction, 281-283
displaying random web content via, 276-280
DOM, 280-281
HTML comments, 276
printing text via, 275
JavaScript's advantages over, 299
.js files, linking to, 69
layouts
fixed layouts, 254
fixed/liquid hybrid layouts, 258-268
liquid layouts, 253-257
links
absolute links, 124
anchor tags, 126-129
images as, 134
linking between web content, 129-131
linking to email addresses, 132-133
linking to external web content, 131-132
opening links in new browser windows, 134
relative-root links, 124
styling via CSS, 134-138
lists
definition lists, 96, 226
multitiered lists, 100-101
nested lists, 97-100, 226
ordered lists, 95-96, 226
HTML

outlines, creating via lists, 98-100
unordered lists, 95-96, 226
"marked up" text, 2
multiple conditions, testing, 374
Notepad, creating HTML files, 26
outlines, creating, 98-100
pseudoclasses, 134-138
tables
aligning within, 113-115
cells, creating, 107
cells, sizing, 111-113
color in, 115
creating, 107-110
headings, creating, 108
images in, 116
page layout via, 116-117
rows, creating, 107
sizing, 110-113
sizing borders, 107
spacing borders, 116
spanning within, 115
uses for, 107
tags, 5, 28
<a> tags, 126-129, 170
attributes, 92-93, 123
</b> tags, 81-83
<big> tags, 83
<body> tag, 27, 31-33
<br /> tag, 32-33, 85
<b> tags, 81-83
closing tags, 30
creating custom tags, 58
<div> tags, 170
<em> tags, 83
empty tags, 30, 33
<em> tags, 83
event handlers, 68, 291
<font> tags, 81, 86
formatting and older HTML tags, 81
<head> tag, 27, 31-33
heading tags, 34-36
horizontal rule tag, 33
<hr /> tag, 33
<html> tag, 27, 31-33
<i> tags, 81-83
<img> tags, 162-163
line breaks, 32-33
naming files with, 27
nested tags, 97
opening tags, 30
<p> tag, 31-33
paragraphs, 32-33
<pre> tags, 84-85
pseudoclasses, 134-138
saving files with, 27
<small> tags, 83
</strike> tags, 84
</strong> tags, 83
</sub> tags, 83
</sup> tags, 83
<table> tags, 107
<td> tags, 107
</th> tags, 108
<title> tags, 27, 31-33, 36
<tr> tags, 107
</tt> tags, 84-85
<u> tags, 84
xml : lang attribute, 31
xmlns attribute, 31
text, formatting
aligning text, 92-95
big text, 83
boldface text, 82-83
customizing fonts, 85-89
definition lists, 96, 226
foreign languages, 89-91
italic text, 82-83
monospaced text, 84-85
multitiered lists, 100-101
nested lists, 97-100, 226
older HTML tags, 81
ordered lists, 95-96, 226
outlines, creating, 98-100
resumes, creating, 87-89
small text, 83
special characters, 89-91
subscript text, 83
superscript text, 83
unordered lists, 95-96, 226
whitespace, 32
TextEdit, creating HTML files, 27
validating, 36-37
web content
absolute addresses, 124-125
absolute links, 124
anchor tags, 126-129
creating, 2-3
delivering, 3-5
directories, 123-124
folders, 123-125
images as links, 134
linking between, 129-131
linking to email addresses, 132-133
linking to external web content, 131-132
opening links in new browser windows, 134
organizing, 123-124
publishing locally, 18-19
publishing to blogs, 19
relative addresses, 124-125
relative-root addresses, 124
relative-root links, 124
selecting web hosting providers, 6-8, 26
styling links via CSS, 134-138
website architectures, creating, 125
whitespace, 32
Word, creating HTML files, 27
<i> tags (HTML), 81-83
if statements (JavaScript), 369

conditional expressions, 317, 370-373
conditional operators, 370
else keyword, 372-375
logical operators, 371
multiple conditions, testing, 373
HTML files, 374
JavaScript files, 374-375

<iframe>. See inline frames
image maps, creating
creating, web resources, 231
CSS, 231-235
tutorials, 231

images
aligning
horizontal alignment, 165-166
vertical alignment, 167-168
animated images, 160-161
auto image loading, web browsers, 20
background images, 171-172
backgrounds
background images, 171-172
tiled backgrounds, 159-160
banners, creating, 156
buttons, creating, 156
CD-ROM, transferring images to, 150
clip art, 149
color, adjusting, 154
compression, 150
copyrights and, 149
Creative Commons licenses, 149
cropping, 151-152
dithering, 158
dynamic websites, changing images via user interaction, 281-283
file sizes, 150
Flickr, 161-163
GIFs, 157-158
animated images, 160-161
tiled backgrounds, 159-160
transparent images, 159
Google Images, 161
Google Picasa, 149
height/width, specifying, 165
image maps, 173-178
JPEGs
compression, 155
tiled backgrounds, 159-160
links, images as, 134, 169-171
Picnik, 149
PNGs, 158-159
Red Eye Removal, 154
republishing, 163
resizing, 153
resolution, 150
rollover images, 394
tables and, 116
text descriptions, 163-164
transparent images, 158
uses of, 150
web pages
grabbing images from, 149
placing images on web pages, 161-163

Images (Google), 161

<img> tags (HTML), 162-163
in value (CSS height/width properties), 53
increment expressions (for loops), 377
incrementing/decrementing variables (JavaScript), 327
indenting
code, 548
text, web page creation, 35
index pages, HTML file management, 16-18
indexes (JavaScript), for loops, 377
infinite loops (JavaScript), 380-381
initial expressions (for loops), 377
inline frames, 426-429
inline styles (CSS), 60-61
inline value (CSS display property), 53
inset value (CSS border-style properties), 54
installing
Greasemonkey, 464
user scripts, 466
internal style sheets (CSS), 46, 59-61
Internet, ISP selection, 25-26
Internet Explorer, 9
DOCTYPE declarations, 212
error messages, displaying, 77
event properties, 393
Internet Explorer 6.0, JavaScript testing, 74
Trixie, 465
installing user scripts, 466
managing user scripts, 467
turning on/off, 468
interpreted languages, 66
ISP (Internet service providers), selecting, 25-26
italic text, 56, 82-83
JavaScript, 274. See also AJAX
accessibility, 439
AJAX, 71
  live search forms, creating, 494-496
  quiz building example, 489-490
requests, 480, 494-495
arrays
  accessing elements of, 338
  declaring, 337
length of, 338
numeric arrays, 337, 340-342
string arrays, 338-340
best practices, 293-295, 433
  accessibility, 439
  behavior, 434
  comments, 438
  content, 434
design patterns, 439
documenting code, 438
event handlers, 435-437
presentation, 434
  “progressive enhancement”, 435
usability, 438-439
web standards and browser specificity, 437-438
break statements, 381
browser specificity, web standards, 437-438
browsers, reading information on, 440-443
capabilities of, 66, 70
case statements, 376
comments, 293, 438
conditional expressions, 370-373
conditional operators, 370
continue statements, 382
cross-browser scripting
debugging browsers, 444
feature sensing, 437, 443-444
data types, 330-331
Date objects, time display example, 72
design patterns, 439
development of, 66
do, while loops, 380
document.write statements, 68, 73
documenting code, 438
DOM, 280-281
  adding text to web pages, 319-321
  anchor objects, 303
  children, 308
document objects, 300-303, 306
  hiding/showing objects, 316-317
history objects, 303-305
layers, 311-315
link objects, 303, 306
location objects, 305-306
modifying text in web pages, 317-318
naming objects, 299
objects, 352
parents, 308
referencing objects, 299
showing/hiding objects, 316-317
siblings, 308
structure of, 306-307
window objects, 300, 409-418
else keyword, 372-375
error handling, 76-78, 446-449
event handlers, 68-70, 290-291, 389, 435
  alert() function, 295
  best practices, 294-295
  creating, 390-391
cross-browser scripting, 437
defining, 390-391
  event objects, 392
  keyboard events, 397-399
  mouse events, 394-397, 400-404
  multiple event handlers, 391-392
  naming, 390
  onLoad events, 399
  onUnload events, 400
  quotation marks and, 390
  syntax of, 390
W3C event model, 436-437
events, 69
expressions, 328
external scripts, 69
flow control
  break statements, 381
  case statements, 376
  continue statements, 382
  continuing loops, 382
  do, while loops, 380
  escaping loops, 381
  for loops, 377-378
  for, in loops, 382-384
  if statements, 369-375
  infinite loops, 380-381
  switch statements, 375-376
  while loops, 379-380
for loops, 377-378
for, in loops, 382-384
form elements, accessing, 528
form events, 527-528
function call statements, 288
functions, 68, 288
  alert() function, 295
  arguments, 348
calling, 349-350
constructor functions, 354
defining, 347-349
math functions, 361-363
naming, 292
parameters, 288, 348
returning values, 351-352
Gmail and, 446
Greasemonkey, 463
API functions, 471
browser support, 465
creating user scripts, 468, 475
debugging user scripts, 474-476
descrining user scripts, 469-470
finding user scripts, 466
installing, 464
installing user scripts, 466
managing user scripts, 466-467
metadata and user scripts, 469-470
site-specific user scripts, 472-473
testing user scripts, 468-471
text area macro user scripts, 475-476
turning on/off, 468
history of, 66
HTML, 274
comments, 276
JavaScript's advantages over, 299
if statements, 369
conditional expressions, 370-373
conditional operators, 370-371
else keyword, 372-375
testing multiple conditions, 373-375
images, changing via user interaction, 281-283
infinite loops, 380-381
.js files, 69
JSON, 481
layers, controlling positioning of, 311-315
libraries
adding effects via, 457
adding to HTML documents, 457
AJAX Frameworks, 456
building scripts, 458-459
Dojo, 456
downloading, 457
jQuery, 454-455
Mochikit, 456
MooTools, 456
Prototype, 453-454
Script.aculo.us, 455-459
using effects via, 457
Yahoo! UI Library, 456
logical operators, 371
loops
break statements, 381
continue statements, 382
escaping, 381
modifying scripts, 74-76
modulo operators, 362
multiple conditions, testing, 374-375
non-JavaScript browsers
avoiding errors, 446-449
detecting, 445-446
JavaScript optionality, 446
<noscript> tag, 445-446
<script> tag, 457
objects, 288
built-in objects, 289, 352, 356-361, 364-366
child objects, 353
creating, 353
creating instances of, 356
custom objects, 289
date object, 364-366
defining, 354-355
defining methods, 355-356
DOM objects, 289, 352
frames and, 425
math object, 360-363
methods, 289, 353, 365
naming, 292
properties, 288, 353
prototypes, 357-358
simplifying scripting via, 354-356
operators, 328-330
order of script operation, determining, 291
output, creating, 73
parseFloat() function, 331
parseInt() function, 331
programming language versus scripting language, 66
"progressive enhancement" strategies, 294
random web content, displaying via, 276-280
remote scripting, 71
<script> tags, 67-72, 457
scripting language versus programming language, 66
special effects, 70
statements
conditional statements, 289
for statements, 290
function calls, 288
loops, 290
objects, 288-289, 292
plus signs (+) in, 73
semicolons, 72
time display example, 72
variables, 288, 292
strings
calculating length of, 334
converting case, 334
string arrays, 338-340
string objects, 332-334, 357
substrings, 335-337
switch statements
declaring, 326
global variables, 326
incrementing/decrementing, 327
local variables, 326
loclime variable, 72
naming, 292, 325-326
scope of, 326
time display example, 72
UTC variable, 72
web pages, adding JavaScript to, 67-68, 73-74
websites, navigating, 70
while loops, 379-380
with keyword, 363-364

JPEG
compression, 155
tiled backgrounds, 159-160

jQuery JavaScript library, 454-455
js files, 69
JSON, 481
jsp file extension, 27
JSP scripting language, 274

kJ - L

Koch, Peter-Paul, 444

languages (foreign), formatting text for, 89-91

layers (DOM)
creating, 311
moveable layers, 311-315
positioning, controlling via JavaScript, 311-315

layouts
CSS layout properties, 52-53
fixed layouts, 254
fixed/liquid hybrid layouts, 258
defining columns in, 260-262
setting column height, 264-268
setting minimum width, 262-263
structure of, 258-259
liquid layouts, 255-257
web resources, 253

leading (text), line-height property (CSS), 56

libraries (AJAX), 482
ajaxRequest function, 486
ajaxResponse function, 486
creating, 485-486
quiz building example
HTML files, 487-488
JavaScript files, 489-490
testing, 490-491
XML files, 488-489
using, 486

libraries (JavaScript)
AJAX Frameworks, 456
Dojo, 456
effects, using via, 457
HTML documents, adding libraries to, 457
jQuery, 454-455
Mochikit, 456
MooTools, 456
Prototype, 453-454
Script.aculo.us, 455-459
Yahoo! UI Library, 456

line breaks, web page creation, 32-33
line-height property (CSS), 56
line-through text (CSS), 56

<link /> tags (CSS), 50
link objects (DOM), 303, 306

Koch, Peter-Paul, 444

languages (foreign), formatting text for, 89-91

layers (DOM)
creating, 311
moveable layers, 311-315
positioning, controlling via JavaScript, 311-315

layouts
CSS layout properties, 52-53
fixed layouts, 254
fixed/liquid hybrid layouts, 258
defining columns in, 260-262
setting column height, 264-268
setting minimum width, 262-263
structure of, 258-259
liquid layouts, 255-257
web resources, 253

leading (text), line-height property (CSS), 56

libraries (AJAX), 482
ajaxRequest function, 486
ajaxResponse function, 486
creating, 485-486
quiz building example
HTML files, 487-488
JavaScript files, 489-490
testing, 490-491
XML files, 488-489
using, 486

libraries (JavaScript)
AJAX Frameworks, 456
Dojo, 456
effects, using via, 457
HTML documents, adding libraries to, 457
jQuery, 454-455
Mochikit, 456
MooTools, 456
Prototype, 453-454
Script.aculo.us, 455-459
scripts, building, 458-459
Yahoo! UI Library, 456

line breaks, web page creation, 32-33
line-height property (CSS), 56
line-through text (CSS), 56

<link /> tags (CSS), 50
link objects (DOM), 303, 306
links

anchor tags
identifying locations within web pages via, 126
linking to anchor locations, 126-129
naming, 127
color and, 144
documents, 303, 306
email addresses, 132-133
images, 134, 169-171
.js files, 69
multimedia/website integration, 179-180
opening in new browser windows, 134
relative-root links, 124
styling via CSS, 134-138
web content
linking between, 129-131
linking to external web content, 131-132
liquid layouts, 253-257
list-item value (CSS display property), 53
lists
borders, styling, 227-228
color, styling, 227-228
CSS box model and, 226-229
definition lists, 96, 226
list item indicators, placing, 229-231
list-style-image property (CSS), 226
list-style-position property (CSS), 226, 229-231
list-style-type property (CSS), 226
list-style-type style rule (CSS), 99-101
multitiered lists, 100-101
navigation lists
horizontal navigation, 245-248
primary navigation, 236
regular lists versus, 235
secondary navigation, 236
vertical navigation, 236-244
nested lists, 97-100, 226
ordered lists, 95-96, 226
outlines, creating via lists, 98-100
unordered lists, 95-96, 226
live search forms
AJAX, creating via, 496
HTML forms, 492
JavaScript front end, 494-495
PHP back end, 493-494
example, requirements for, 496
LiveScript, JavaScript development, 66
loading web content, timing, 20
local time values, date object (JavaScript) and, 365-366
local variables (JavaScript), 326
localtime variable (JavaScript), 72
location objects (DOM), 305-306
logical operators (JavaScript), 371
loops (JavaScript), 290
break statements, 381
continue statements, 382
continuing, 382
do, while loops, 380
escaping, 381
for loops, 377-378
for, in loops, 382-384
infinite loops, 380-381
while loops, 379-380
LunarPages web hosting provider, 7

M

Macintosh computers, HTML file creation, 27
managing
domain names, 6
HTML files, 14
document roots, 15-16
index pages, 16-18
user scripts, 466-467
web pages, headings, 34-36
websites
coding clarity, 548
comments, 546-547
documenting code, 546-547
indenting code, 548
maintainable code, 546-548
version control, 548-550
margin property (CSS), 191-199
margins
browsers and, 226-229
CSS box model, 210
marked up text in HTML, 2
Mashable.com, publicizing websites via, 554
math object (JavaScript)
decimal numbers
rounding, 360
truncating, 360
math functions, 361-363
random numbers, generating, 360-363
media-specific style sheets, 503-504
metadata, user scripts, 469-470
methods
DOM objects, 299
document objects, 302, 310
history objects, 303
location objects, 306
JavaScript objects, 289, 353
adding to string objects, 357
defining, 355-356
get methods, 365
prototypes, 357-358
MIME types, 182
mm value (CSS height/width properties), 53
Mochikit JavaScript library, 456
objects (DOM) 583

<object> tags (XHTML), 180-183
objects (built-in), extending definitions, 359
objects (DOM), 352
  anchor objects, 303
  document objects, 300
  anchor objects, 303
  displaying document information, 301-302
  link objects, 303, 306
  methods of, 302
  properties of, 301
  writing text within documents, 302
  hiding/showing, 316-317
  history objects, 303-305
  link objects, 303, 306
  location objects, 305-306
  methods, 299
  document objects, 302
  history objects, 303
  location objects, 306
  naming, 299
  parents, 308
  properties, 299
  document objects, 301
  history objects, 303
  location objects, 305-306
  referencing, 299
  showing/hiding, 316-317
  siblings, 308
window objects, 300
  creating browser windows, 410-411
  displaying dialog boxes, 417-418
  moving browser windows, 413-414
  opening/closing browser windows, 411-412

N
naming
  anchor tags, 127
  DOM objects, 299

modifying
  frame borders, 424-425
  JavaScript scripts, 74-76
text in web pages, 317-318
module operators (JavaScript), 362
monitors and color, 144
monospaced text, 84-85
MooTools JavaScript library, 456
mouse events
  mousestatus function, 396
  onClick event handler, 394-397, 400-404
  onDblClick event handler, 395
  onMouseDown event handler, 395-397
  onMouseOut event handler, 394
  onMouseOver event handler, 394
  onMouseUp event handler, 395-397
  rollover images, 394
mousestatus function, mouse events and, 396
moveable layers (DOM), 311-315
moving browser windows, 413-414
Mozilla Firefox web browser, 9
multimedia
  QuickTime, 180
  website integration with, 178-179
  embedded multimedia files, 180-183
  links, 179-180
  streaming multimedia, 181
  tips for using, 184-185
multiple event handlers, 391-392
multitiered lists, 100-101

O
object tags (XHTML), 180-183
objects (built-in), extending definitions, 359
objects (DOM), 352
  anchor objects, 303
document objects, 300
  anchor objects, 303
displaying document information, 301-302
link objects, 303, 306
methods of, 302
properties of, 301
writing text within documents, 302
hiding/showing, 316-317
history objects, 303-305
link objects, 303, 306
location objects, 305-306
methods, 299
document objects, 302
history objects, 303
location objects, 306
naming, 299
parents, 308
properties, 299
document objects, 301
history objects, 303
location objects, 305-306
referencing, 299
showing/hiding, 316-317
siblings, 308
window objects, 300
  creating browser windows, 410-411
  displaying dialog boxes, 417-418
  moving browser windows, 413-414
  opening/closing browser windows, 411-412

NaN (not a number), 331
navigating websites, JavaScript, 70
navigation lists
  horizontal navigation, 245-248
  primary navigation, 236
  regular lists versus, 235
  secondary navigation, 236
  vertical navigation, 236-238
    multilevel vertical navigation, 240-244
    single-level vertical navigation, 239-241
nested framesets, 425
nested lists, 97-100, 226
nested tags (HTML), 97
nodes (DOM), 307
  methods of, 310
  properties, 309
non-viewable window areas (browsers), 254
none value (CSS)
  border-style properties, 54
  display property, 53
<noscript> tag (JavaScript), detecting non-JavaScript browsers, 445-446
Notepad, creating HTML files, 26
null values (JavaScript), 330
numbers
  arithmetic mean, 363
decimal numbers
  rounding, 360
  truncating, 360
random numbers, generating, 360-363
numeric arrays (JavaScript), 337, 340-342
numeric data types (JavaScript), 330

N
584  objects (DOM)

string objects
   adding methods to, 357
   assigning values, 332-334
   combining values, 332-334
   creating, 332

Office Online Clip Art and Media website (Microsoft), 149
onClick event handler, 394-397, 400-404
onDbClick event handler, 395
online resources, CSS reference guide, 47
onLoad events, 399
onMouseDown event handler, 395-397
onMouseOut event handler, 394
onMouseOver event handler, 394
onMouseUp event handler, 395-397
onUnload events, 400
opening tags (HTML), 30
opening/closing, browser windows, 411-412
Opera, 9, 465
operators (JavaScript), 328-330
ordered lists, 95-96, 226
organizing
   HTML files, 14
     document roots, 15-16
     index pages, 16-18
   web content, 123-124
   web pages, headings, 34-36
   websites, 538
     Amazon.com, 543
     BAWSI.org, 545
     ESPN.com, 540-542
     larger websites, 543-546
     simple websites, 540-542
     single-page websites, 538-539
     Starbucks.com, 544
outlines, creating via lists, 98-100
outset value (CSS border-style properties), 54

overflow property (CSS), text flow and, 220
overlapping elements, 213, 217-219

P
<p> tags, 31-33
padding
   browsers and, 226-228
   CSS box model, 210
padding property (CSS), 56, 191, 199-202
paragraphs
   aligning, 93-95
   web page creation, 32-33
parameters (JavaScript functions), 288, 348
parent folders, 125
parents (DOM), 289, 352
instances, creating, 356
math object
   generating random numbers, 360-363
   math functions, 361-363
   rounding decimal numbers, 360
   truncating decimal numbers, 360
methods, 289, 353
   adding to string objects, 357
   get methods, 365
   prototypes, 357-358
   methods, defining, 355-356
naming, 292
properties, 288
   prototypes, 357-358
   values, 353
prototypes, 357-358
scripting, simplifying
   creating object instances, 356
   defining object methods, 355-356
   defining objects, 354-355
photos
   aligning
      horizontal alignment, 165-166
      vertical alignment, 167-168
   background photos, 171-172
   background image style property, 172
   CD-ROM, transferring photos to, 150
   cropping, 151-152
   Flickr, 161-163
   Google Images, 161
   height/width, specifying, 165
   image maps, 173-178
   links, turning images into, 169-171
   Red Eye Removal, 154
   republishing, 163
   resizing, 153
text descriptions, 163-164
web pages, placing photos on, 161-163
Photoshop (Adobe), 148
PHP, 274, 514
AJAX live search forms, creating, 493-494
.php file extensions, 27
Picasa (Google), 149
Picnik, 149
“plain” text, 26, 34, 82
plug-ins, 180
plus signs (+), JavaScript statements, 73
PNGs, 158-159
pop-up windows, 134, 528-530
positionable elements. See layers positioning
absolute positioning, 213-217
overlapping elements, 213, 217-219
positioning property, 213
relative positioning, 213-215
presentation. JavaScript scripting best practices, 434
<pre> tags (HTML), 84-85
pricing, web hosting providers, 6
primary navigation, 236
printing
Print Preview, viewing web pages in, 508-509
print-friendly web pages, 499
criteria for print-friendliness, 500-503
designing style sheets for print pages, 505-508
media-specific style sheets, 503-504
reviewing content for print-friendliness, 500
viewing web pages in Print Preview, 508-509
text via JavaScript, 275
programming languages, strings, 72
“progressive enhancement” strategies for, 294
web design and, 435
prompts (dialog boxes), 417-418
properties
DOM objects, 299
document objects, 301
history objects, 303
location objects, 305-306
relationship properties, 309
JavaScript objects, 288
prototypes, 357-358
values, 353
Prototype JavaScript library, 453-454
prototypes (JavaScript objects), 357-358
pseudoclasses, 134-138
pt value (CSS height/width properties), 53
publicizing websites, 553-555
publishing web content
blog publication, 19
local publication, 18-19
pull-down pick lists (forms), 524-525
purchasing domain names, 6
px value (CSS height/width properties), 53
Python scripting language, 274
radio buttons (forms), 523-524
random numbers, generating, 360-363
Red Eye Removal, 154
relationship properties (DOM nodes), 309
relative addresses and web pages, 124-125
relative positioning, 213-215
display property (CSS), 53
positioning property, 214
relative-root addresses and web pages, 124
relative-root links and web pages, 124
reliability, web hosting providers, 6
remote scripting, 71. See also AJAX
republishing images, 163
reserved words, JavaScript syntax, 292
resizing
browser windows, 413-414
images, 153
resolution (graphics), 150
ridge value (CSS border-style properties), 54
rollover images, 394
rounding decimal numbers, 360
rows (tables), creating, 107
Ruby scripting language, 274

S
Safari, 9, 465
sans-serif font (text), 86
saving files
files with HTML tags, 27
.js files, 69
scaling images, 153
<script> tag (JavaScript), 67-69
detecting non-JavaScript browsers, 457
time display example, 71-72
Script.aculo.us JavaScript library, 455-459

scripting

AJAX, 479
  ajaxRequest function, 486
  ajaxResponse function, 486
  back end, 480, 493-494
  debugging applications, 491-496
  examples of, 481
  frameworks, 482
  front end, 480, 494-495
  JavaScript client, 480
  libraries, 482, 485-491
  limitations of, 482
  live search forms, 492-496
  quiz building example, 487-491
  requests, 480, 483-484, 493-495
  server-side scripts, 480, 493-495
  XML and, 481
  XMLHttpRequest, 483-484

ASP, 274
  client-side scripting, 274
  comments, adding, 293
  cross-browser scripting, 443
  debugging browsers, 444
  event handlers and JavaScript, 437
  feature sensing, 437, 443-444

Greasemonkey, 463-464
  API functions, 471
  browser support, 465
  creating user scripts, 468, 475
  debugging user scripts, 474-476
  describing user scripts, 469-470
  finding user scripts, 466
  installing, 464
  installing user scripts, 466
  managing user scripts, 466-467
  metadata and user scripts, 469-470
  site-specific user scripts, 472-473
  testing user scripts, 468-471
  text area macro user scripts, 475-476
  turning on/off, 468
  interpreted languages, 66
  JavaScript, 274
    accessibility, 439
    adding scripts to web pages, 73-74
    adding to web pages, 67-68
    advantages over HTML, 299
    AJAX, 71
    AJAX live search forms, 494-496
    AJAX quiz building example, 489-490
    AJAX requests, 480, 494-495
    arrays, 337-342
    best practices, 293-295, 433-439
    break statements, 381
    capabilities of, 66, 70
    case statements, 376
    changing images via user interaction, 281-283
    comments, 293, 438
    conditional expressions, 370-373
    conditional operators, 370
    continue statements, 382
    continuing loops, 382
    creating .js files, 69
    creating output, 73
    cross-browser scripting, 437, 443-444
    data types, 330-331
    Date objects, 72
    design patterns, 439
    determining order of script operation, 291
    development of, 66
    displaying random web content via, 276-280
do, while loops, 380
document.write statements, 68, 73
documenting code, 438
DOM, 280-281, 299-321, 409-418
documentation keyword, 372-375
error handling, 76-78
escaping loops, 381
window objects (DOM), 409
  events, 69
  expressions, 328
  external scripts, 69
  flow control, 369-384
  for loops, 377-378
  for, in loops, 382-384
  form events, 527-528
  frames and JavaScript objects, 425
  functions, 68, 288, 292, 347, 352-354, 361-363
  Gmail and, 446
  Greasemonkey, 463-476
  history of, 66
  HTML and, 274-276
  if statements, 369-375
  infinite loops, 380-381
  .js file extension, 69
  JSON, 481
  libraries (third-party), 453-459
  linking to .js files, 69
  logical operators, 371
  modifying scripts, 74-76
  navigating websites, 70
  non-JavaScript browsers, 445-449
  objects, 288-289, 292, 352-361, 364-366
operators, 328-330
parseFloat() function, 331
parseInt() function, 331
plus signs (+) in statements, 73
programming language versus scripting language, 66
"progressive enhancement" strategies, 294
reading browser information, 440-443
remote scripting, 71
saving .js files, 69
<script> tags, 67-72
scripting language versus programming language, 66
simplifying, 354-356
special effects, 70
statements, 72, 287
statements, conditional statements, 289
statements, for statements, 290
statements, function calls, 288
statements, loops, 290
statements, objects, 288-289, 292
strings, 332-340, 357
syntax, case-sensitivity, 292
syntax, functions, 292
syntax, objects, 292
syntax, reserved words, 292
syntax, spacing (whitespace), 292
syntax, variables, 292
testing scripts, 74
time display example, 71-78
toLowerCase() method, 334
toUpperCase() method, 334
Trixie, 465-468
"unobtrusive scripting", 433, 447-449
usability, 438-439
using, 68
validating forms, 70
variables, 72, 288, 292, 325-328
web standards and browser specificity, 437-438
while loops, 379-380
with keyword, 363-364
JSON, 481
JSP, 274
languages, 65
Perl, 274
PHP, 274, 514
Python, 274
remote scripting, 465
Ruby, 274
server-side scripts, 274, 480, 493-495
text editors, 74
Trixie, 465
installing user scripts, 466
managing user scripts, 467
turning on/off, 468
.txt file extension, 74
"unobtrusive scripting", 433, 447-449
user scripts
creating, 468, 475
debbuging, 474-476
describing, 469-470
finding, 466
Geaseemonkey, 463-476
installing, 466
managing, 466-467
metadata and, 469-470
scripting, 465
site-specific scripts, 472-473
testing, 468-471
text area macro user scripts, 475-476
Trixie, 465-467
VBScript, 274
scrolling lists (forms), 524-525
search engines, 445
heading tags (HTML), 36
SEO, 553, 562-563
spamming, 557
websites, listing with search engines, 555-562
searches
Google searches, 4
live search forms, creating via AJAX, 496
HTML forms, 492
JavaScript front end, 494-495
PHP back end, 493-494
secondary navigation, 236
security, user scripts, 465
selection lists (forms), 524-525
selectors (CSS), 57
semicolon (;), JavaScript statements, 72, 287
server-side scripting, 274, 480, 493-495
servers
browsers, basic server interaction, 3-5
document roots, 13-16
FTP client connections, 13
space, 6
"uptime", 6
web hosting providers, selecting, 6
shorthand conditional expressions (JavaScript), 372-373
siblings (DOM), 308
single-page websites, 538-539
site-specific user scripts, 472-473
sizing
borders (tables), 107
browser windows, 413-414
cells (tables), 111-113
elements (CSS box model), 210-212
images, 153
tables, 110-113
sizing

text
  font-size style rule (CSS), 86-87
  style sheets, 50
skeleton pages. See templates
<small> tags (HTML), 83
small text, 83
solid value (CSS border-style properties), 54-55
sorting arrays (JavaScript)
  numeric arrays, 340-342
  string arrays, 340
source editors, blogs and, 19
spacing (whitespace), JavaScript syntax, 292
spamming search engines, 557
spanning with tables, 115
special effects (JavaScript), 70
splitting string arrays (JavaScript), 339
Starbucks.com, website organization, 544
statements (JavaScript), 287
  conditional statements, 289
  for statements, 290
  function calls, 288
  loops, 290
  objects, 288
    built-in objects, 289
    custom objects, 289
    DOM objects, 289
    naming, 292
  plus signs (+) in, 73
  semicolons, 72
  time display example, 72
  variables, 288, 292
Stephenson, Sam, 453
streaming multimedia, 181
<strike> tags (HTML), 84
strike-through text, style sheets, 56
string arrays, sorting, 340
strings, 72

strings (JavaScript), 330
  case, converting, 334
  length of, calculating, 334
  string arrays, 338
  sorting, 340
  splitting, 339
string objects
  adding methods to, 357
  assigning values, 332-334
  combining values, 332-334
  creating, 332
substrings
  finding, 336-337
  getting single characters, 336
  using parts of strings, 335-336
<strong> tags (HTML), 83
strong text. See boldface text
style classes (CSS), 57-58
style IDs (CSS), 59
style properties (CSS), 57
style rules (CSS), 46, 50
  color style rule, fonts and, 86-87
  font weight style rule, 83
  font-family style rule, 86
  font-size style rule, 86-87
  list-style-type style rule, 99-101
  multiple style properties in, 58
  text-align style rule, 93-95
  viewing, 51
style sheets
  align property, 191, 203-204
  box model, 209, 212
    borders, 210
    content, 210
    lists and, 226-229
    lists and, 226
    margins, 210
    padding, 210
    sizing elements, 210-212
    clear property, text flow and, 220
color, specifying via style sheets, 146-148
creating, 47, 49-52
CSS tags, 50
CSS Zen Garden, 191-192
definition of, 45
DOCTYPE declarations, 212
external style sheets, 46-51
float property, 191, 204-207, 220
formatting properties, 53
  background-color property, 55
  border property, 54-55
  border-bottom property, 54-55
  border-color property, 54-55
  border-left property, 54-55
  border-right property, 54-55
  border-style property, 54-55
  border-top property, 54-55
  border-width property, 54
  color property, 56
  font property, 56
  font-family property, 55
  font-size property, 55
  font-style property, 56
  font-weight property, 56
  line-height property, 56
  padding property, 56
  text-align property, 55-56
  text-decoration property, 56
  text-indent property, 55
HTML documents, linking to, 50
image maps, creating, 231-235
inline styles, 60-61
internal style sheets, 46, 59-61
italic text, 56
layouts
  display property, 52-53
  fixed layouts, 254
  fixed/liquid hybrid layouts, 258-268
  height property, 53
liquid layouts, 253-257
width property, 53
line-through text, 56
links, styling, 134-138
list-style-image property, 226
list-style-position property, 226, 229-231
list-style-type property, 226
lists
horizontal navigation, 245-248
navigation lists, 236-248
vertical navigation, 236-244
margin property, 191-199
media-specific style sheets, 503-504
overflow property, text flow and, 220
padding property, 191, 199-202
positioning
absolute positioning, 213-217
overlapping elements, 213, 217-219
positioning property, 213
relative positioning, 213-215
z-index property, 217-219
print pages, designing style sheets for, 505-508
properties, hyphenating, 312
selectors, 57
strike-through text, 56
style classes, 57-58
style IDs, 59
style properties, 57
style rules, 46, 50
text
formatting color, 46
sizing, 50
underline text, 56
validating, 61
web browsers, CSS support, 51
z-index property, 217-219
<sub> tags (HTML), 83
subscript text, 83
substrings
finding, 336-337
parts of strings, using, 335-336
single characters, getting, 336
Subversion website version control, 550
<sup> tags (HTML), 83
superscript text, 83
support
CSS, web browser support for, 51
web hosting providers, selecting, 6
switch statements (JavaScript)
multiple conditions, using, 375-376
syntax of, 376
syntax, JavaScript, 332
T
<table> tags (HTML), 107
tables
aligning within, 113-115
borders
sizing, 107
spacing, 116
specifying color via CSS, 146-148
cells
creating, 107
sizing, 111-113
color in, 115
creating, 107-110
headings, creating, 108
images in, 116
page layout via, 116-117
rows, creating, 107
sizing, 110-113
spanning within, 115
uses for, 107
tags (CSS), 50
elements and, 46
<link /> tag, 50
selectors, 57
tags (HTML), 5, 28
<a> tags, 170
identifying locations within web pages, 126
linking to anchor locations, 126-129
naming, 127
attributes, 92-93, 123
<b> tags, 81-83
<big> tags, 83
<body> tag, 27, 31-33
<br /> tags, 32-33, 85
closing tags, 30
containers, 307
custom tags, creating, 58
<div> tags, 170
<em> tags, 83
empty tags, 30, 33
event handlers, 68
<font> tags, 81, 86
<head> tags, 27, 31-33, 68
heading tags, 34-36
horizontal rule tag, 33
<hr /> tag, 33
<html> tag, 27, 31-33
<i> tags, 81-83
<img> tags, 162-163
line breaks, 32-33
naming files with, 27
tags (HTML)

nested tags, 97
older HTML tags, formatting and, 81
<p> tag, 31-33
paragraphs, 32-33
<pre> tags, 84-85
pseudoclasses, 134-138
saving files with, 27
<script> tags, 67-72
<small> tags, 83
<strike> tags, 84
<strong> tags, 83
<sub> tags, 83
<table> tags, 107
<td> tags, 107
<th> tags, 108
<title> tags, 27, 31-33, 36
<tr> tags, 107

<tt> tags, 84-85
<u> tags, 84
xml : lang attribute, 31
xmlns attribute, 31
tags (XHTML)
<embed> tags, 183
<object> tags, 180-183
tasks, combining. See functions (JavaScript)
<td> tags (HTML), 107
templates, web page creation, 31
testing
AJAX quiz building example, 490-491
JavaScript scripts, Internet Explorer 6.0, 74
user scripts, 468, 470-471
web content, 19-20
websites, multiple web browsers, 8-9, 26
text
aligning
attributes, 92-93
block-level elements, 93-95
paragraphs, 93-95
tables, 113-115
text-align style rule (CSS), 93-95
alternate text, 163-164
ASCII text, 26, 34, 82
color, CSS and, 56, 146-148
documents, writing text within, 302
flowing text, 220
fonts
Arial font, 86
color style rule (CSS), 86-87
font property (CSS), 56
font weight style rule (CSS), 83
font-family property (CSS), 55
font-family style rule (CSS), 86
font-size property (CSS), 55
font-size style rule (CSS), 86-87
font-style property (CSS), 56
font-weight property (CSS), 56
sans-serif font, 86
Times Roman font, 86
typewriter font. See monospaced text
formatting
aligning text, 92-95
big text, 83
boldface text, 82-83
customizing fonts in HTML, 85-89
definition lists, 96, 226
foreign languages, 89-91
italic text, 82-83
monospaced text, 84-85
multitiered lists, 100-101
nested lists, 97-100, 226
ordered lists, 95-96, 226
outlines, 98-100
resumes, creating, 87-89
small text, 83
special characters, 89-91
subscript text, 83
superscript text, 83
unordered lists, 95-96, 226
web page creation, 33-35
forms, accepting text input in, 519
graphics and, 163-164
HTML, whitespace, 32
indenting, web page creation, 35
italic text, style sheets, 56
leading, line height property (CSS), 56
line breaks, web page creation, 32-33
line-through text, style sheets, 56
paragraphs, web page creation, 32-33
"plain" text, 26, 34, 82
printing, JavaScript and, 275
sizing, font-size style rule (CSS), 86-87
strike-through text, style sheets, 56
style sheets, 56
formatting color, 46
sizing, 50
text-align property (CSS), 55-56
text-align style rule (CSS), 93-95
text area macro user scripts, 475-476
text areas (forms), 526
text-decoration property (CSS), 56
text editors, 74
text fields (forms), 526
text-indent property (CSS), 55
underline text, style sheets, 56
web browsers, adjusting font size settings, 20
web pages
adding to web pages, 319-321
modifying text in, 317-318
TextEdit, creating HTML files, 27
<th> tags (HTML), 108
third-party JavaScript libraries
AJAX Frameworks, 456
dojo, 456
downloading, 457
effects, adding via, 457-459
effects, using, 457
HTML documents, adding libraries to, 457
jQuery, 454-455
Mochikit, 456
MooTools, 456
Prototype, 453-454
Script.aculo.us, 455-459
scripts, building, 458-459
Yahoo! UI Library, 456

tiled backgrounds, 159-160
time, displaying (JavaScript example), 71
Date objects, 72
timing loading of web content, 20
<time> tags, 71-72
toLowerCase() method (JavaScript), 334
tool tips, 164
toUpperCase() method (JavaScript), 334
<tr> tags (HTML), 107

text area macro user scripts, 475-476
Trixie, 465
turning on/off, 468
user scripts
installing, 466
managing, 467

truncating decimal numbers, 360
<tt> tags (HTML), 84-85
turning on/off
Greasemonkey, 468
Trixie, 468

.txt file extension, 74
typewriter font. See monospaced text

U
<u> tags (HTML), 84
underline text, style sheets, 56
"unobtrusive scripting", 433, 447-449
unordered lists, 95-96, 226
"uptime," servers and, 6
URLs, opening, 483
usability, JavaScript best practices, 438-439
USB drivers, 18
user input, accepting in forms, 519
user scripts
creating, 468, 475
debugging, 474-476
describing, 469-470
displaying (JavaScript example), 71
finding, 466
Greasemonkey, 463
API functions, 471
browser support, 465
creating user scripts, 468, 475
debugging user scripts, 474-476
describing user scripts, 469-470
finding user scripts, 466
installing, 464
logging, 461
managing user scripts, 466-467
metadata and user scripts, 469-470
site-specific user scripts, 472-473
testing user scripts, 468-471
text area macro user scripts, 475-476
Trixie, 465
installing user scripts, 466
managing user scripts, 467
UTC (Universal Time [Coordinated]), 71
UTC variable (JavaScript), 72

V
validating
forms, 70, 531-532
style sheets, 61
web content, 36-37

variables (JavaScript), 288
declaring, 326
global variables, 326
incrementing/decrementing, 327
local variables, 326
localtime variable, 72
naming, 292, 325-326
scope of, 326
time display example, 72
UTC variable, 72
values, assigning to variables, 327-328
VBScript scripting language, 274
version control, websites and, 548-550
vertical image alignment, 167-168
vertical navigation, 236-238
  multilevel vertical navigation, 240-244
  single-level vertical navigation, 239-241
vertical-align style property, 167
video, embedding, 182
viewing
  CSS style rules, 51
  HTML files, 29
  web pages, 29
visual editors, blogs, 19

W
W3C color standards, 143
W3C CSS Validator, 61
W3C event model, 436-437
W3C Validation Service, 37-38
web browsers
  140 cross-browser color names, 143-144
  cross-browser scripting
    debugging browsers, 444
    feature sensing, 437, 443-444
  CSS, support for, 51
  debugging, 444
  development of, 2
  dialog boxes, displaying, 417-418
  distributing, 18
  Firefox, 9, 464
  frames, 418-420
    adding individual frames, 423
    creating frameset documents, 421-423
    frames array, 426
    inline frames, 426-429
  JavaScript objects and, 425
  linking windows to frames, 423-424
  modifying borders, 424-425
  nested framesets, 425
  Google Chrome, 9, 465
  history, accessing, 2, 303-305
  HTML development, 2
  images, auto image loading, 20
  information, reading via JavaScript
    dishonest browsers, 442-443
    displaying information, 440-441
  Internet Explorer, 9
    event properties, 393
    Trixie, 465-468
  links, opening in new browser windows, 134
  lists, displaying in, 97
  margins and, 226-229
  non-Internet Explorer event properties, 393-394
  non-JavaScript browsers
    avoiding errors, 446-449
    detecting, 445-446
    JavaScript optionality, 446
    <noscript> tag, 445-446
    <script> tag, 457
  non-viewable window areas, 254
  Opera, 9, 465
  padding and, 226-228
  pop-up windows, 134
  popularity of, 26
  Safari, 9
  search engines, 445
  sensing. See feature sensing
  servers, basic browser server interaction, 3-5
  text, adjusting font size settings, 20
  websites
    comparing, 26
    testing, 8-9, 26
  windows
    creating, 410-411
    linking frames to windows, 423-424
    moving, 413-414
    opening/closing, 411-412
    resizing, 413-414
    timeouts, 414-416
web content
  absolute addresses, 124-125
  aligning via align property (CSS), 191, 203-204
  clear property (CSS), text flow and, 220
  color
    140 cross-browser color names, 143-144
    best practices, 141-143
    Colorblind Web Page Filter tool, 148
    hexadecimal color codes, 144-146
    using, 141-143
    W3C color standards, 143
  comparing, 26
  creating, 2-3
  ASCII text, 26, 34
  boilerplate code, 28-30
  comparing web content HTML codes, 37
  formatting text, 33-35
  HTML tags, 27-33
  indenting text, 35
  line breaks, 32-33
  organizing content via headings, 34-36
  overview of, 29
  paragraphs, 32-33
  “plain” text, 26, 34
  templates, 31
CSS box model, 209
  borders, 210
  content, 210
lists and, 226-229
margins, 210
padding, 210
sizing elements, 210-212
delivering, 3-5
directories, 123-124
float property (CSS), 191, 204, 207, 220
folders, 123-125
forms, 513
  accessing elements via JavaScript, 528
  check boxes, 521-523
  creating, 514-519
  displaying data in pop-up windows, 528-530
  hidden data, 520
  JavaScript events, 527-528
  naming form data, 519-520
  pull-down pick lists, 524-525
  radio buttons, 523-524
  scrolling lists, 524-525
  selection lists, 524-525
  sending data via email, 530-531
  submitting form data, 527
text areas, 526
text fields, 526
text input, 519
user input, 519
validating, 531-532
"graceful degradation", 434
graphics
  adjusting color, 154
  Adobe Photoshop, 148
  aligning graphics, 165-168
  animated graphics, 160-161
  background graphics, 171-172
  banners, 156
  buttons, 156
  choosing software, 149
  clip art, 149
  compression, 150
copyrights and, 149
  Creative Commons licenses, 149
cropping, 151-152
dithering, 158
file sizes, 150
Flickr, 161, 163
GIFs, 157-161
GIMP, 149
  GIMP adjusting image color, 154
  GIMP banners, 156
  GIMP buttons, 156
  GIMP cropping images, 151-152
  GIMP JPEG compression, 155
  GIMP resizing images, 153
  Google Images, 161
  Google Picasa, 149
glexing from web pages, 149
image maps, 173-178
PEGs, 155, 159-160
Picnik, 149
placing graphics on web pages, 181-183
PNGs, 158-159
Red Eye Removal, 154
republishing, 163
resizing, 153
resolution, 150
specifying height/width, 165
text descriptions, 163-164
tiled backgrounds, 159-160
transparency graphics, 158
turning graphics into links, 169-171
uses of, 150
images, changing images via user interaction, 281-283
links
  absolute links, 124
  anchor tags, 126-129
e-mail addresses, 132-133
images as, 134
linking between, 129-131
linking to external web content, 131-132
opening in new browser windows, 134
relative-root links, 124
styling via CSS, 134-138
lists
  borders, 227-228
color, 227-228
CSS box model and, 226-229
horizontal navigation, 245-248
navigation lists, 235-248
placing list item indicators, 229-231
vertical navigation, 236-244
loading, timing, 20
managing
  coding clarity, 548
  comments, 546-547
documenting code, 546-547
  indenting code, 548
  maintainable code, 546-548
  version control, 548-550
margins
  browsers and, 226-229
  margin property (CSS), 191-199
multimedia, integrating with, 178
  embedded multimedia files, 180-183
  links, 179-180
tips for using, 184-185
organizing, 123-124, 538
  larger websites, 543-546
  simple websites, 540-542
  single-page websites, 538-539
overflow property (CSS), text flow and, 220
padding
  browsers and, 226-228
  padding property (CSS), 191, 199-202
web content

Print Preview, viewing web content in, 508-509
print-friendly web pages, 499
criteria for print-friendliness, 500-503
designing style sheets for print pages, 505-508
media-specific style sheets, 503-504
reviewing content for print-friendliness, 500
viewing web pages in Print Preview, 508-509
“progressive enhancement”, 435
publishing
locally, 18-19
to blogs, 19
random web content, displaying via JavaScript, 276-280
relative addresses, 124-125
relative-root addresses, 124
search engines, listing web content with, 555-562
style sheets
creating, 47-52
definition of, 45
external style sheets, 46-51
formatting properties, 53-56
formatting text color, 46
inline styles, 60-61
internal style sheets, 46, 59-61
layout properties, 52-53
linking to HTML documents, 50
selectors, 57
sizing text, 50
style classes, 57-58
style IDs, 59
style properties, 57
style rules, 46, 50-51, 58
validating, 61
web browser support, 51
tables
aligning within, 113-115
cells, creating, 107
cells, sizing, 111-113
color in, 115
creating, 107-110
headings, creating, 108
images in, 116
page layout via, 116-117
rows, creating, 107
sizing, 110-113
sizing borders, 107
spacing borders, 116
spanning within, 115
uses for, 107
testing, 19-20
text, formatting, 82
adding to web pages, 319-321
aligning text, 92-95
big text, 83
boldface text, 82-83
customizing fonts in HTML, 85-89
definition lists, 96, 226
flowing text, 220
foreign languages, 89-91
italic text, 82-83
modifying, 317-318
monospaced text, 84-85
multitiered lists, 100-101
nested lists, 97-100, 226
older HTML tags, 81
ordered lists, 95-96, 226
outlines, 98-100
resumes, creating, 87-89
small text, 83
special characters, 89-91
subscript text, 83
superscript text, 83
unordered lists, 95-96, 226
transferring, FPT, 29
validating, 36-37
viewing, 5, 29
web hosting providers, selecting, 6-8, 26
website architectures, creating, 125
YouTube and, 184
web design
“graceful degradation”, 434
“progressive enhancement”, 435
web hosting provider DailyRazor, 7
web hosting providers
A Small Orange, 7
bandwidth, 6
customer service, 6
domain names, purchasing, 6
LunarPages, 7
pricing, 6
reliability, 6
selecting, 6-8, 26
server space, 6
web pages
absolute addresses, 124-125
aligning via align property (CSS), 191, 203-204
clear property (CSS), text flow and, 220
color
140 cross-browser color names, 143-144
best practices, 141-143
Colorblind Web Page Filter tool, 148
hexadecimal color codes, 144-146
using, 141-143
W3C color standards, 143
creating
ASCII text, 26, 34
boilerplate code, 28-30
comparing web page HTML codes, 37
formatting text, 33-35
HTML tags, 27-33
indenting text, 35
line breaks, 32-33
organizing content via headings, 34-36
overview of, 29
paragraphs, 26-34
“plain” text, 26, 34
templates, 31
CSS box model, 209
borders, 210
content, 210
lists and, 226-229
margins, 210
padding, 210
sizing elements, 210-212
directories, 123-124
float property (CSS), 191, 204-207, 220
folders, 123-125
forms, 513
accessing elements via JavaScript, 528
check boxes, 521-523
creating, 514-519
displaying data in pop-up windows, 528-530
hidden data, 520
JavaScript events, 527-528
naming form data, 519-520
pull-down pick lists, 524-525
radio buttons, 523-524
scrolling lists, 524-525
selection lists, 524-525
sending data via email, 530-531
submitting form data, 527
text areas, 526
text fields, 526
text input, 519
user input, 519
validating, 531-532
“graceful degradation”, 434

graphics
adjusting color, 154
Adobe Photoshop, 148
aligning graphics, 165-168
animated graphics, 160-161
background graphics, 171-172
banners, 156
buttons, 156
choosing software, 149
clip art, 149
compression, 150
copyrights and, 149
Creative Commons licenses, 149
cropping, 151-152
dithering, 158
file sizes, 150
Flickr, 161-163
GIFs, 157-161
GIMP, 149
GIMP adjusting image color, 154
GIMP banners, 156
GIMP buttons, 156
GIMP cropping images, 151-152
GIMP JPEG compression, 155
GIMP resizing images, 153
Google Images, 161
Google Picasa, 149
gathering from web pages, 149
image maps, 173-178
JPEG compression, 155
JPEGs, 159-160
Picnik, 149
placing graphics on web pages, 161-163
PNGs, 158-159
Red Eye Removal, 154
republishing, 163
resizing, 153
resolution, 150
specifying height/width, 165
text descriptions, 163-164
tiled backgrounds, 159-160
transparent graphics, 158
turning graphics into links, 169-171
uses of, 150
images, changing images via user interaction, 281-283
JavaScript, adding to web pages, 67-68
links
absolute links, 124
anchor tags, 126-129
e-mail addresses, 132-133
images as, 134
linking between web pages, 129-131
linking to external web pages, 131-132
opening in new browser windows, 134
relative-root links, 124
styling via CSS, 134-138
lists
borders, 227-228
color, 227-228
CSS box model and, 226-229
horizontal navigation, 245-248
navigation lists, 235-248
placing list item indicators, 229-231
vertical navigation, 236-244
loading, timing, 20
managing
coding clarity, 548
comments, 546-547
documenting code, 546-547
indenting code, 548
maintainable code, 546-548
version control, 548-550
margins
browsers and, 226-229
margin property (CSS), 191-199
multimedia, integrating with, 178
embedded multimedia files, 180-183
links, 179-180
tips for using, 184-185
organizing, 123-124, 538
larger websites, 543-546
simple websites, 540, 542
single-page websites, 538-539
overflow property (CSS), text flow and, 220
padding
browsers and, 226-228
padding property (CSS), 191, 199-202
Print Preview, viewing web pages in, 508-509
print-friendly web pages, 499
criteria for print-friendliness, 500-503
designing style sheets for print pages, 505-508
media-specific style sheets, 503-504
reviewing content for print-friendliness, 500
viewing web pages in Print Preview, 508-509
“progressive enhancement”, 435
relative addresses, 124-125
relative-root addresses, 124
scripts, adding to web pages (JavaScript), 73-74
search engines, listing web pages with, 555-556
style sheets
creating, 47-52
definition of, 45
external style sheets, 46-51
formatting properties, 53-56
formatting text color, 46
inline styles, 60-61
internal style sheets, 46, 59-61
layout properties, 52-53
linking to HTML documents, 50
selectors, 57
sizing text, 50
style classes, 57-58
style IDs, 59
style properties, 57
style rules, 46, 50-51, 58
validating, 61
web browser support, 51
tables
aligning within, 113-115
cells, creating, 107
cells, sizing, 111-113
color in, 115
creating, 107-110
headings, creating, 108
images in, 116
page layout via, 116-117
rows, creating, 107
sizing, 110-113
sizing borders, 107
spacing borders, 116
spanning within, 115
uses for, 107
text, formatting
adding to web pages, 319-321
aligning, 92-95
big text, 83
boldface text, 82-83
customizing fonts in HTML, 85-89
definition lists, 96, 226
flowing text, 220
foreign languages, 89-91
italic text, 82-83
modifying, 317-318
monospaced text, 84-85
multitiered lists, 100-101
nested lists, 97-100, 226
older HTML tags, 81
ordered lists, 95-96, 226
outlines, 98-100
resumes, creating, 87-89
small text, 83
special characters, 89-91
subscript text, 83
superscript text, 83
unordered lists, 95-96, 226
transferring FTP, 29
validating, 36-37
viewing, 29
web content, displaying random content via JavaScript, 276-280
website architectures, creating, 125
YouTube and, 184
websites
aligning via align property (CSS), 191, 203-204
architectures, creating, 125
bad website examples, 144
clear property (CSS), text flow and, 220
color
140 cross-browser color names, 143-144
best practices, 141-143
Colorblind Web Page Filter tool, 148
hexadecimal color codes, 144-146
using, 141-143
W3C color standards, 143
comparing, 26
connecting to, Classic FTP client, 12
CSS box model, 209
borders, 210
content, 210
lists and, 226-229
margins, 210
padding, 210
sizing elements, 210-212
dynamic websites
changing images via user interaction, 281-283
client-side scripting, 274
displaying random web content via JavaScript, 276-280
DOM, 280-281
<table>
<thead>
<tr>
<th>websites</th>
<th>597</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIFs, 157-161</td>
<td>CSS box model and, 226-229</td>
</tr>
<tr>
<td>GIMP, 149</td>
<td>horizontal navigation, 245-248</td>
</tr>
<tr>
<td>GIMP adjusting image color, 154</td>
<td>navigation lists, 235-248</td>
</tr>
<tr>
<td>GIMP banners, 156</td>
<td>placing list item indicators, 229-231</td>
</tr>
<tr>
<td>GIMP buttons, 156</td>
<td>vertical navigation, 236-244</td>
</tr>
<tr>
<td>GIMP cropping images, 151-152</td>
<td>managing</td>
</tr>
<tr>
<td>GIMP JPEG compression, 155</td>
<td>coding clarity, 548</td>
</tr>
<tr>
<td>GIMP resizing images, 153</td>
<td>comments, 546-547</td>
</tr>
<tr>
<td>Google Images, 161</td>
<td>documenting code, 546-547</td>
</tr>
<tr>
<td>Google Picasa, 149</td>
<td>indenting code, 548</td>
</tr>
<tr>
<td>grabbing from web pages, 149</td>
<td>maintainable code, 546-548</td>
</tr>
<tr>
<td>JPEGs, 155, 159-160</td>
<td>version control, 548-550</td>
</tr>
<tr>
<td>Picnik, 149</td>
<td>margins</td>
</tr>
<tr>
<td>placing graphics on web pages, 161-163</td>
<td>browsers and, 226-229</td>
</tr>
<tr>
<td>PNGs, 158-159</td>
<td>margin property (CSS), 191-199</td>
</tr>
<tr>
<td>Red Eye Removal, 154</td>
<td>multimedia, integrating with, 178-179</td>
</tr>
<tr>
<td>republishing, 163</td>
<td>embedded multimedia files, 180-183</td>
</tr>
<tr>
<td>resizing, 153</td>
<td>links, 179-180</td>
</tr>
<tr>
<td>resolution, 150</td>
<td>tips for using, 184-185</td>
</tr>
<tr>
<td>specifying height/width, 165</td>
<td>navigating JavaScript, 70</td>
</tr>
<tr>
<td>text descriptions, 163-164</td>
<td>organizing</td>
</tr>
<tr>
<td>tiled backgrounds, 159-160</td>
<td>Amazon.com, 543</td>
</tr>
<tr>
<td>transparent graphics, 158</td>
<td>BAWSI.org, 545</td>
</tr>
<tr>
<td>turning graphics into links, 169-171</td>
<td>ESPN.com, 540-542</td>
</tr>
<tr>
<td>uses of, 150</td>
<td>larger websites, 543-546</td>
</tr>
<tr>
<td>links</td>
<td>simple websites, 540-542</td>
</tr>
<tr>
<td>absolute links, 124</td>
<td>single-page websites, 538-539</td>
</tr>
<tr>
<td>anchor tags, 126-129</td>
<td>Starbucks.com, 544</td>
</tr>
<tr>
<td>email addresses, 132-133</td>
<td>overflow property (CSS), text flow and, 220</td>
</tr>
<tr>
<td>images as, 134</td>
<td>padding</td>
</tr>
<tr>
<td>linking between web pages, 129-131</td>
<td>browsers and, 226-228</td>
</tr>
<tr>
<td>linking to external web pages, 131-132</td>
<td>padding property (CSS), 191, 199-202</td>
</tr>
<tr>
<td>opening in new browser windows, 134</td>
<td>&quot;progressive enhancement&quot;, 435</td>
</tr>
<tr>
<td>relative-root links, 124</td>
<td>publicizing, 553-555</td>
</tr>
<tr>
<td>styling via CSS, 134-138</td>
<td>search engines, listing websites with, 555-562</td>
</tr>
<tr>
<td>borders, 227-228</td>
<td>SEO, 553, 562-563</td>
</tr>
<tr>
<td>color, 227-228</td>
<td>single-page websites, 538-539</td>
</tr>
</tbody>
</table>

Links:
- GIMP, 149
- Adobe Photoshop, 148
- clip art, 149
- Creative Commons licenses, 149
- cropping, 151-152
- dithering, 158
- file sizes, 150
- Flickr, 161-163
- Red Eye Removal, 154
- Picnik, 149
- grabbing from web pages, 149
- JPEGs, 155, 159-160
- PNGs, 158-159
- text descriptions, 163-164
- tiled backgrounds, 159-160
- transparent graphics, 158
- turning graphics into links, 169-171

Forms:
- forms, 513
- absolute links, 124
- anchor tags, 126-129
- email addresses, 132-133
- hosting page, 514-519
- displaying data in pop-up windows, 528-531
- hidden data, 520
- JavaScript events, 527-528
- naming form data, 519-520
- pull-down pick lists, 524-525
- radio buttons, 523-524
- scrolling lists, 524-525
- selection lists, 524-525
- submitting data via email, 530-531
- text areas, 526
- text fields, 526
- text input, 519
- user input, 519
- validating, 531-532

CSS:
- CSS box model and, 226-229
- horizontal navigation, 245-248
- navigation lists, 235-248
- placing list item indicators, 229-231
- vertical navigation, 236-244

Margins:
- browsers and, 226-229
- margin property (CSS), 191-199

Multimedia:
- embedding multimedia files, 180-183
- links, 179-180
- tips for using, 184-185

SEO:
- Amazon.com, 543
- BAWSI.org, 545
- ESPN.com, 540-542
- larger websites, 543-546
- simple websites, 540-542
- single-page websites, 538-539
- Starbucks.com, 544

Version Control:
- managing
- coding clarity, 548
- comments, 546-547
- documenting code, 546-547
- indenting code, 548
- maintainable code, 546-548
- version control, 548-550

Validation:
- "graceful degradation", 434

Organizing:
- Amazon.com, 543
- BAWSI.org, 545
- ESPN.com, 540-542
- larger websites, 543-546
- simple websites, 540-542
- single-page websites, 538-539
- Starbucks.com, 544

Overflow:
- overflow property (CSS), text flow and, 220

Padding:
- browsers and, 226-228
- padding property (CSS), 191, 199-202

"Progressive Enhancement":
- 435

Publicizing:
- publicizing, 553-555

Search Engines:
- Amazon.com, 543
- BAWSI.org, 545
- ESPN.com, 540-542
- larger websites, 543-546
- simple websites, 540-542
- single-page websites, 538-539
- Starbucks.com, 544
- Google Images, 161
- Google Picasa, 149
- grabbing from web pages, 149
- image maps, 173-178
- JPEGs, 155, 159-160
- Picnik, 149
- placing graphics on web pages, 161-163
- PNGs, 158-159
- Red Eye Removal, 154
- republishing, 163
- resizing, 153
- resolution, 150
- specifying height/width, 165
- text descriptions, 163-164
- tiled backgrounds, 159-160
- transparent graphics, 158
- turning graphics into links, 169-171
- uses of, 150

Text Areas:
- text areas, 526
- text fields, 526
- text input, 519
- user input, 519

Lists:
- borders, 227-228
- color, 227-228

Hyperlinks:
- absolute links, 124
- anchor tags, 126-129
- email addresses, 132-133
- images as, 134
- linking between web pages, 129-131
- linking to external web pages, 131-132
- opening in new browser windows, 134
- relative-root links, 124
- styling via CSS, 134-138

Images:
- GIFs, 157-161
- GIMP, 149
- GIMP adjusting image color, 154
- GIMP buttons, 156
- GIMP cropping images, 151-152
- GIMP JPEG compression, 155
- GIMP resizing images, 153
- Google Images, 161
- Google Picasa, 149
- grabbing from web pages, 149
- image maps, 173-178
- JPEGs, 155, 159-160
- Picnik, 149
- placing graphics on web pages, 161-163
- PNGs, 158-159
- Red Eye Removal, 154
- republishing, 163
- resizing, 153
- resolution, 150
- specifying height/width, 165
text descriptions, 163-164
tiled backgrounds, 159-160
transparent graphics, 158
turning graphics into links, 169-171
uses of, 150
websites

style sheets
creating, 47-52
definition of, 45
external style sheets, 46-51
formatting properties, 53-56
formatting text color, 46
inline styles, 60-61
internal style sheets, 46, 59-61
layout properties, 52-53
linking to HTML documents, 50
selectors, 57
sizing text, 50
style classes, 57-58
style IDs, 59
style properties, 57
style rules, 46, 50-51, 58
validating, 61
web browser support, 51
testing, 8-9, 26
text
adding to web pages, 319-321
flowing text, 220
modifying, 317-318
web content
absolute addresses, 124-125
directories, 123-124
folders, 123-125
organizing, 123-124
relative addresses, 124-125
relative-root addresses, 124
web pages
print-friendly web pages, 499-509
viewing in Print Preview, 508-509
YouTube and, 184
while loops (JavaScript), 379-380
whitespace (spacing)
HTML, 32
JavaScript syntax, 292
width
CSS box model, adjusting in, 210-212
fixed/liquid hybrid layouts, setting
minimum width in, 262-263
images, specifying width in, 165
width property (CSS), 53
window objects (DOM), 300
browser windows
creating, 410-411
moving, 413-414
opening/closing, 411-412
resizing, 413-414
timeouts, 414-416
dialog boxes, displaying, 417-418
properties of, 409-410
windows (browser)
closing, 412
creating, 410-411
frames, linking to windows, 423-424
moving, 413-414
non-viewable window areas, 254
opening/closing, 411-412
pop-up windows, displaying form
data in, 528-530
resizing, 413-414
timeouts, 414-416
with keyword (JavaScript), 363-364
.WMV video clips, embedding, 182
Word, creating HTML files, 27
WordPress Theme Gallery, layouts
and, 253
writing text within documents, 302
WWW (World Wide Web), HTML develop-
ment, 2
WYSIWYG (what-you-see-is-what-you-
get) editors, 27

Y - Z
Yahoo! Developer Network, JavaScript
design patterns, 439
Yahoo! Search, listing websites
with, 556
Yahoo! UI Library, 437, 456
YouTube, website integration, 184

X
XHTML
boilerplate code, 30
check boxes (forms), 522
defining, 3
development of, 40
function of, 51
goal of, 51
image maps, 231
inline frames, 426-429
tags
<embed> tags, 183
<object> tags, 180-183
xml : lang attribute (HTML tags), 31
xmlns attribute (HTML tags), 31

XML
AJAX and, 481, 488-489
boilerplate code, 30
development of, 39-40
xml : lang attribute (HTML tags), 31
XMLHttpRequest, 483
requests
awaiting responses, 484
creating, 483
interpreting response
data, 484
sending, 484
URLs, opening, 483
xmlns attribute (HTML tags), 31

Y - Z
Yahoo! Developer Network, JavaScript
design patterns, 439
Yahoo! Search, listing websites
with, 556
Yahoo! UI Library, 437, 456
YouTube, website integration, 184

z-index property (CSS), 217-219
Zen Garden (CSS), 191-192, 253