Your FULL COLOR guide to building cool Web pages

HTML, XHTML & CSS FOR DUMMIES®

6th Edition

Ed Tittel
Jeff Noble

HTML, XHTML & CSS FOR DUMMIES®

6th Edition

by Ed Tittel and Jeff Noble

WILEY

Wiley Publishing, Inc.

HTML, XHTML & CSS FOR DUMMIES®

6th Edition

by Ed Tittel and Jeff Noble

WILEY

Wiley Publishing, Inc.

HTML, XHTML & CSS For Dummies®, 6th Edition

Published by
Wiley Publishing, Inc.
111 River Street
Hoboken, NJ 07030-5774

`www.wiley.com`

Library of Congress Control Number: 2008924086

ISBN: 978-0-470-23847-9

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

WILEY

About the Authors

Ed Tittel is a full-time independent writer, trainer, and consultant who works out of his home near beautiful Austin, Texas. Ed has been writing for the trade press since 1986 and has worked on more than 140 books. In addition to this title, Ed has worked on more than 35 books for Wiley, including *Windows Server 2008 For Dummies, XML For Dummies,* and *Networking with NetWare For Dummies*.

Ed is a Contributing Editor at Tomshardware.com, writes for half-a-dozen different TechTarget.com Web sites, including WhatIs.com, SearchNetworking.com, and SearchWindows.com, and also writes occasionally for other Web sites and magazines. When he's not busy doing all that work stuff, Ed likes to travel, shoot pool, spend time with his family (especially taking walks with young Gregory), and turn the tables on his Mom, who now makes her home with the rest of the Texas Tittels.

You can contact Ed Tittel by e-mail at `etittel@yahoo.com`.

Jeff Noble runs a small Web design and multimedia company called Conquest Media (`www.conquestmedia.com`) in Austin, Texas. Jeff has been working on, in, and around the Web for nearly 10 years, and he specializes in designing and creating unique, easy to use, functional Web sites. When he's away from his computer, Jeff is often far from the madding crowd, choosing instead to hike and camp in wild places as far away from a wall socket as he can get.

Jeff is available for Web site design, implementation, and consulting work. You can contact him by e-mail at `jeff@conquestmedia.com`.

Authors' Acknowledgments

Now that we've made it into the twelfth go-round for *HTML For Dummies,* we must once again thank our many readers for keeping this book alive. We'd also like to thank them and the Wiley editors for providing the feedback that drives the continuing improvement of this book. Please don't stop now — tell us what you want to do, and what you like and don't like about this book. Especially, please tell us what you liked and didn't like about this, our first full-color edition of our book.

Let me also thank the many people who've also worked on this book over the years, including James Michael Stewart, Natanya Anderson, Dori Smith, Tom Negrino, Mary Burmeister, Rich Wagner, Brock Kyle, Chelsea Valentine, and Kim Lindros. Of course, for this edition, I'm especially indebted to my co-author and friend, Jeff Noble, for infusing insight and enthusiasm into this book. I am eternally grateful to you for your ideas, your hard work, and your experience in reaching an audience of budding Web experts.

Next, I'd like to thank the Wiley team for their efforts on this title. At Wiley, I must thank Bob Woerner and Paul Levesque for their outstanding efforts, and Barry Childs-Helton and Sue Jenkins for their editorial efforts in design, layout, contents, and coverage. A special shout out should go to the friendly folks in Composition Services for their artful page layouts, especially when it came to keeping all the color-coded code straight.

I'd like to thank my lovely wife, Dina Kutueva-Tittel, and periodically pugnacious 4-year-old son, Gregory, for putting up with the usual rhythm of making books happen. I know I'm not always as easy to live with as I should be, but hopefully, I'll get to keep working on that. Also, I'd like to thank my parents, Al and Ceil, for all the great things they did for me, and for hanging in there well into their ninth decades on this planet. I hope you're both still around to see the thirteenth edition come to print as well! Finally, profound thanks to you again, Mom, for cultivating and encouraging my love of words, writing, and banter.

Ed Tittel

First and foremost, I'd like to thank my mom, Sheryl, for always believing in me. I'd like to thank my older brother Chris, even though he once shot me with his BB gun when we were kids. (I'd also like to mention that one year for Christmas he gave me a used Skid Row tape made to look semi-new covered with Saran Wrap. But I've moved past that.) Thanks again, bro. To my "adopted" family, The Elizondos, I appreciate everything you have done for me over the years and I promise to never sing Christmas carols in the house. *"Shut up Jeff!"*

I'd certainly like to thank Ed Tittel for giving a semi-literate guy such as myself this opportunity: you're a good friend, a great writer, and definitely know a thing or two about local Austin eateries.

I want to give special thanks to the following friends and colleagues: Slade Deliberto for teaching me how to design Web sites, Matt "Softball" Douglass for convincing me CSS is a lot better than old school HTML tags, Justin Haworth for bailing me out of many confusing Flash scripting problems, Peter "3D Pete" Vogel for his mentoring and assistance creating the Conquest Media logo, Jason "GodLikeMouse" Graves for teaching me almost everything I know about JavaScript, XML and CSS layout, and Russell Wilson for his usability lessons and expert advice. Thanks to the whole gang at Conquest Media, too!

Jeff Noble

Publisher's Acknowledgments

We're proud of this book; please send us your comments through our Dummies Online Registration Form located at www.dummies.com.

Some of the people who helped bring this book to market include the following:

Acquisitions, Editorial, and Media Development

Project Editor: Paul Levesque

Acquisitions Editor: Bob Woerner

Copy Editor: Barry Childs-Helton

Technical Editor: Sue Jenkins

Editorial Manager: Leah Cameron

Media Development Supervisor: Richard Graves

Editorial Assistant: Amanda Foxworth

Cartoons: Rich Tennant, www.the5thwave.com

Production

Project Coordinator: Katie Key

Layout and Graphics: Kathie Rickard, Erin Zeltner

Proofreaders: Betty Kish, Dwight Ramsey, Toni Settle

Indexer: Ty Koontz

Publishing and Editorial for Technology Dummies

Richard Swadley, Vice President and Executive Group Publisher

Andy Cummings, Vice President and Publisher

Mary Bednarek, Executive Acquisitions Director

Mary C. Corder, Editorial Director

Publishing for Consumer Dummies

Diane Graves Steele, Vice President and Publisher

Joyce Pepple, Acquisitions Director

Composition Services

Gerry Fahey, Vice President of Production Services

Debbie Stailey, Director of Composition Services

Contents at a Glance

Introduction 1

Part I: Getting to Know (X)HTML and CSS 7

Chapter 1: The Least You Need to Know about HTML, CSS, and the Web................9

Chapter 2: Creating and Viewing a Web Page 29

Chapter 3: Proper Planning Prevents Poor Page Performance................................41

Part II: Formatting Web Pages with (X)HTML 57

Chapter 4: Creating (X)HTML Document Structure 59

Chapter 5: Text and Lists..69

Chapter 6: Linking to Online Resources 91

Chapter 7: Finding and Using Images 103

Part III: Taking Precise Control Over Web Pages and Styles... 119

Chapter 8: Introducing Cascading Style Sheets 121

Chapter 9: Using Cascading Style Sheets..137

Chapter 10: Getting Creative with Colors and Fonts...157

Chapter 11: Using Tables to Jazz Up Your Pages 175

Part IV: Integrating Scripts with (X)HTML 211

Chapter 12: Scripting Web Pages...213

Chapter 13: The Nuts and Bolts of JavaScript...223

Chapter 14: Working with Forms 247

Chapter 15: Fun with Client-Side Scripts...271

Part V: (X)HTML Projects.. 289

Chapter 16: The About Me Page 291

Chapter 17: The eBay Auction Page 299

Chapter 18: A Company Site...307

Chapter 19: A Product Catalog...319

Part VI: The Part of Tens 329
Chapter 20: Ten HTML Dos and Don’ts 331
Chapter 21: Ten Ways to Exterminate Web Bugs 339
Chapter 22: Ten Cool HTML Tools and Technologies 347
Chapter 23: Ten Tip-Top Online HTML References 357
Index 367

Table of Contents

Introduction 1

About This Book 1
How to Use This Book 2
Three Presumptuous Assumptions 3
How This Book Is Organized 3
  Part I: Getting to Know (X)HTML and CSS 4
  Part II: Formatting Web Pages with (X)HTML 4
  Part III: Taking Precise Control Over Web Pages and Styles 4
  Part IV: Integrating Scripts with (X)HTML 5
  Part V: (X)HTML Projects 5
  Part VI: The Part of Tens 5
Icons Used in This Book 6
Where to Go from Here 6

Part I: Getting to Know (X)HTML and CSS 7

Chapter 1: The Least You Need to Know about HTML, CSS, and the Web 9

Web Pages in Their Natural Habitat 9
  Hypertext 10
  Markup 11
  Browsers 13
  Web servers 13
Anatomy of a URL 14
  HTML and XHTML: What's the difference? 16
  Syntax and rules 17
  Elements 18
  Attributes 20
  Entities 21
Parts Is Parts: What Web Pages Are Made Of 23
  Organizing HTML text 24
  Images in HTML documents 25
  Links and navigation tools 26
Listing 1-1: Meet an Author! 26

Chapter 2: Creating and Viewing a Web Page .29
Before You Get Started 29
Creating a Page from Scratch 30
Step 1: Planning a simple design 31
Step 2: Writing some HTML 32
Step 3: Saving your page 34
Step 4: Viewing your page 36
Editing an Existing Web Page 37
Posting Your Page Online 39

Chapter 3: Proper Planning Prevents Poor Page Performance 41
Planning Your Site 42
Mapping your site 44
Building solid navigation 47
Planning outside links 49
Hosting Your Web Site 51
Hosting your own Web site 52
Using a hosting provider 53
Obtaining your own domain 53
Moving files to your Web server 54

Part II: Formatting Web Pages with (X)HTML 57

Chapter 4: Creating (X)HTML Document Structure 59
Establishing a Document Structure 59
Labeling Your (X)HTML Document 60
Adding an HTML DOCTYPE declaration 60
Adding an XHTML DOCTYPE declaration 60
The <html> element 61
Adding the XHTML namespace 61
Adding a Document Header 62
Giving your page a title 62
Defining metadata 63
Automatically redirecting users to another page 65
Creating the (X)HTML Document Body 67

Chapter 5: Text and Lists 69
Formatting Text 69
Paragraphs 70
Headings 72
Controlling Text Blocks 74
Block quotes 74
Preformatted text 75
Line breaks 77
Horizontal rules 78

Organizing Information 81
Numbered lists 81
Bulleted lists 84
Definition lists 87
Nesting lists 89

Chapter 6: Linking to Online Resources 91

Basic Links 91
Link options 92
Common mistakes 95
Customizing Links 96
New windows 96
Locations in Web pages 98
Non-HTML resources 100

Chapter 7: Finding and Using Images 103

The Role of Images in a Web Page 103
Creating Web-Friendly Images 104
Adding an Image to a Web Page 106
Image location 106
Using the <img /> element 106
Adding alternative text 108
Specifying image size 110
Setting an image border 112
Controlling image alignment 113
Setting image spacing 114
Images That Link 115
Triggering links 115
Building image maps 117

Part III: Taking Precise Control Over Web Pages and Styles 119

Chapter 8: Introducing Cascading Style Sheets 121

Advantages of Style Sheets 122
What CSS can do for a Web page 123
What you can do with CSS 124
CSS Structure and Syntax 126
Selectors and declarations 128
Working with style classes 129
Inheriting styles 131
Using Different Kinds of Style Sheets 133
Internal style sheets 133
External style sheets 134
Understanding the Cascade 136

Chapter 9: Using Cascading Style Sheets 137

Managing Layout, Positioning, and Appearance 138
Developing specific styles 138
Externalizing style sheets 146
Multimedia...148
Visual media styles 148
Paged media styles 153

Chapter 10: Getting Creative with Colors and Fonts 157

Color Values 158
Color names...158
Color numbers...159
Color Definitions 160
Text 161
Links 161
Backgrounds..163
Fonts..164
Font family 164
Sizing 166
Positioning...168
Text treatments...170
The catchall font property...174

Chapter 11: Using Tables to Jazz Up Your Pages 175

What Tables Can Do for You...176
Table Basics 178
Sketching Your Table..179
Developing layout ideas...179
Drafting the table 180
Constructing Basic Tables...181
Components 181
Layout...182
Adding borders 186
Adjusting height and width 189
Padding and spacing 192
Shifting alignment 195
Adding Spans...198
Column spans..198
Row spans..200
Populating Table Cells 201
Testing Your Table..205
Table-Making Tips 205
Following the standards...205
Sanitizing markup 206
Nesting tables within tables 206
Avoiding dense tables 208
Adding color to table cells...208
Other Table Markup of Interest 209

Part IV: Integrating Scripts with (X)HTML 211

Chapter 12: Scripting Web Pages 213

What JavaScript Can Do for Your Pages 214
- Arrange content dynamically 215
- Work with browser windows 218
- Solicit and verify user input 219
- But wait . . . there's more! 221

Chapter 13: The Nuts and Bolts of JavaScript 223

Including Scripts in Web Pages 224
Using the Same Script on Multiple Pages 225
Exploring the JavaScript Language 227
- Basic syntax rules 228
- Variables and data types 229
- Operating on expressions 231
- Working with statements 234
- Loops 236
- Functions 239
- Arrays 241
- Objects 243

Events and Event Handling 243
Document Object Model (DOM) 245
Other JavaScript Items of Interest 246
References and Resources 246

Chapter 14: Working with Forms 247

Uses for Forms 247
- Searches 248
- Data collection 249

Creating Forms 251
- Structure 251
- Input tags 252
- Validation 264

Processing Data 264
- Using CGI scripts and other programs 265
- Sending data by e-mail 266

Designing User-Friendly Forms 267
Other Noteworthy Forms-Related Markup 269

Chapter 15: Fun with Client-Side Scripts 271

Adding Rollovers to Your Pages 272
- Image rollovers with JavaScript 272
- Text rollovers with CSS 275

Displaying Dynamic Content on Your Page 278
- HTML and JavaScript 278
- JavaScript and DOM 280

Displaying Pop-up Windows.....281
Working with Cookies.....284
Using the XHTML Object Element.....287

Part V: (X)HTML Projects..... 289

Chapter 16: The About Me Page.....291

Overview and Design Considerations.....291
    Audience analysis.....292
    Component elements.....292
Page Markup.....292
    Your home page.....292
    Looking good.....295

Chapter 17: The eBay Auction Page.....299

Designing Your Auction Page.....300
Presentation Issues to Consider.....303
Using a Template for Presenting Your Auction Item.....303

Chapter 18: A Company Site.....307

Issues to Consider When Designing Your Site.....307
Basic Elements of a Company Web Site.....308
    The home page.....309
    The Web Design page.....312
    The Contact Us page.....314
    The style sheet.....316
Give Your Visitors What They Need.....318

Chapter 19: A Product Catalog.....319

Dissecting a Product Catalog.....319
Choosing a Shopping Cart.....322
    PayPal.....323
    Other e-commerce solutions.....323
    Incorporating a PayPal shopping cart.....324
Page Markup.....326

Part VI: The Part of Tens..... 329

Chapter 20: Ten HTML Dos and Don'ts.....331

Concentrate on Content.....331
    Never lose sight of your content.....332
    Structure your documents and your site.....332

Go Easy on the Graphics, Bells, Whistles, and Hungry Dinosaurs........333
Make the most from the least........333
Build attractive pages........333
Create Well-Formulated HTML and Test........334
Keep track of those tags........334
Avoid browser dependencies........335
Navigating your wild and woolly Web........336
Keep It Interesting After It's Built!........336
Think evolution, not revolution........337
Beating the two-dimensional-text trap........337
Overcoming inertia takes vigilance........338

Chapter 21: Ten Ways to Exterminate Web Bugs........339

Avoid Dead Ends and Spelling Faux Pas........339
Make a list and check it — twice........340
Master text mechanics........340
Keep Your Perishables Fresh!........341
Lack of live links — a loathsome legacy........341
When old links must linger........342
Make your content mirror your world........342
Check Your Site, and Then Check It Again!........343
Look for trouble in all the right places........343
Cover all the bases with peer reviews........344
Use the best tools of the testing trade........344
Schedule site reviews........344
Let User Feedback Feed Your Site........345
Foster feedback........345
If you give to them, they'll give to you!........346

Chapter 22: Ten Cool HTML Tools and Technologies........347

HTML Editors........348
Helper editors........348
WYSIWYG editors........350
Graphics Tools........351
Photoshop Elements: The amateur champ........352
Professional contenders........352
Link Checkers........353
HTML Link Validator: The champ........354
Contenders........354
HTML Validators........355
W3C validator........355
Built-in validators........355
FTP Clients........356
Swiss Army Knives........356

Chapter 23: Ten Tip-Top Online HTML References 357

Nothing But the Specs, Please! 357
The HTML and XHTML DTDs 358
Character Codes Come In Many Flavors 360
Deprecated (X)HTML Elements and Attributes 361
Magnificent HTML Resource Sites 364

Index 367

Introduction

Welcome to the wild, wacky, and wonderful possibilities of the World Wide Web, more simply called *the Web.* In this book, we reveal the mysteries of the markup languages that are the lifeblood of the Web — the Hypertext Markup Language (HTML) and its successor, XHTML, along with the Cascading Style Sheet (CSS) language widely used to make the other stuff look good. Because HTML and XHTML (we use *(X)HTML* in this book to refer to both versions at once) and CSS may be used to build Web pages, learning how to use them brings you into the fold of Web authors and content developers.

If you've tried to build your own Web pages but found it too forbidding, now you can relax. If you can dial a telephone or find your keys in the morning, you too can become an (X)HTML author. No kidding!

This book keeps the technobabble to a minimum and sticks with plain English whenever possible. Besides plain talk about hypertext, (X)HTML, and the Web, we include lots of examples, plus tag-by-tag instructions to help you build your very own Web pages with minimum muss and fuss. We also provide more examples about what to do with your Web pages after they're created so you can share them with the world. We also explain the differences between HTML 4 and XHTML, so you can decide whether you want to stick with the best-known and longest-lived Web markup language (HTML) or its later and greater successor (XHTML).

We also have a companion Web site for this book that contains (X) HTML and CSS examples from the chapters in usable form — plus pointers to interesting widgets that you can use to embellish your own documents and astound your friends. Visit `www.edtittel.com/html4d6e` and start browsing from there.

About This Book

Think of this book as a friendly, approachable guide to taking up the tools of (X)HTML and CSS, and building readable, attractive pages for the Web. These things aren't hard to learn, but they pack a lot of details. You must handle at least some of these details as you build your own Web pages. Topics you find in this book include

- Designing and building Web pages
- Uploading and publishing Web pages for the world to see
- Testing and debugging your Web pages

You can build Web pages without years of arduous training, advanced aesthetic capabilities, or ritual ablutions in ice-cold streams. If you can tell somebody how to drive across town to your house, you can build a useful Web document. The purpose of this book isn't to turn you into a rocket scientist (or, for that matter, a rocket scientist into a Web site). The purpose is to show you the design and technical elements you need for a good-looking, readable Web page and to give you the confidence to do it!

How to Use This Book

This book tells you how to use (X)HTML and CSS to get your Web pages up and running on the World Wide Web. We tell you what's involved in designing and building effective Web documents that can bring your ideas and information to the whole online world — if that's what you want to do — and maybe have some high-tech fun communicating them.

All (X)HTML and CSS code appears in monospaced type like this:

```
<head><title>What's in a Title?</title></head>...
```

When you type (X)HTML tags, CSS, or other related information, be sure to copy the information exactly as you see it between the angle brackets (< and >), including the angle brackets themselves, because that's part of the magic that makes (X)HTML and CSS work. Other than that, you find out how to marshal and manage the content that makes your pages special, and we tell you exactly what you need to do to mix the elements of (X)HTML and CSS with your own work.

The margins of a book don't give us the same room as the vast reaches of cyberspace. Therefore, some long lines of (X)HTML and CSS markup, or designations for Web sites (called *URLs,* for *Uniform Resource Locators*), may wrap to the next line. Remember that your computer shows such wrapped lines as a *single line of (X)HTML or CSS,* or as a single URL — so if you type that hunk of code, keep it as one line. Don't insert a hard return if you see one of these wrapped lines. We clue you in that the (X)HTML or CSS markup is supposed to be *all one line* by breaking the line at a slash or other appropriate character (to imply "but wait, there's more!") and by slightly indenting the overage, as in the following silly example:

```
http://www.infocadabra.transylvania.com/nexus/plexus/lexus/
       praxis/okay/this/is/a/make-believe/URL/but/some/real/
       ones/are/SERIOUSLY/long.html
```

HTML doesn't care whether you type tag text in uppercase, lowercase, or both (except for character entities, also known as character codes). XHTML and CSS, however, want tag text only in lowercase to be perfectly correct. Thus, to make your own work look like ours as much as possible, enter all (X) HTML and CSS tag text, and all other code, *in lowercase only*. (If you have a prior edition of the book, this reverses our earlier instructions. The keepers of the eternal and ever-magnanimous standard of HTML, the World Wide Web Consortium (W3C), have restated the rules, so we follow their lead. We don't make the rules, but we *do* know how to play the game!)

You'll also find that our code listings are color coded, where we assign specific colors to various types of markup. This is explained in Chapter 1 in a sidebar titled "Markup Color Coding." (You might notice that all the illustrations have nice, pretty colors, too!)

Three Presumptuous Assumptions

They say that making assumptions makes a fool out of the person who makes them and the person who is subject to those assumptions (and just who are *they,* anyway? We *assume* we know, but . . . never mind).

You don't need to be a master logician or a wizard in the arcane arts of programming, nor do you need a Ph.D. in computer science. You don't even need a detailed sense of what's going on in the innards of your computer to deal with the material in this book.

Even so, practicality demands that we make a few assumptions about you, gentle reader: You can turn your computer on and off; you know how to use a mouse and a keyboard; and you want to build your own Web pages for fun, profit, or your job. We also assume that you already have a working connection to the Internet and a Web browser.

If you can write a sentence and know the difference between a heading and a paragraph, you can build and publish your own documents on the Web. The rest consists of details — and we help you with those!

How This Book Is Organized

This book contains six major parts, arranged like Russian *Matrioshka* (nesting dolls). Parts contain at least three chapters, and each chapter contains several modular sections. That way you can use this book to

- Jump around.
- Find topics or keywords in the Index or in the Table of Contents.
- Read the whole book from cover to cover.

Part I: Getting to Know (X)HTML and CSS

This part sets the stage and includes an overview of and introduction to the Web and the software that people use to mine its treasures. This section also explains how the Web works, including the (X)HTML and CSS that this book covers, and the server-side software and services that deliver the goods to end users (when we aren't preoccupied with the innards of our systems).

(X)HTML documents, also called *Web pages,* are the fundamental units of information organization and delivery on the Web. Here you also discover what HTML is about, how hypertext can enrich ordinary text, and what CSS does to modify and manage how that text looks on display. Next you take a walk on the Web side and build your very first (X)HTML document.

Part II: Formatting Web Pages with (X)HTML

HTML mixes ordinary text with special strings of characters called *markup,* used to instruct browsers how to display (X)HTML documents. In this part of the book, you find out about markup in general and (X)HTML in particular. We start with a fascinating discussion of (X)HTML document organization and structure. (Well . . . *we* think it's fascinating, and hope you do, too.) Next we explain how text can be organized into blocks and lists. Then we tackle how the hyperlinks that put the *H* into (X)HTML work. After that, we discuss how you can find and use graphical images in your Web pages and make some fancy formatting maneuvers to spruce up those pages.

Throughout this part of the book, we include discussion of (X)HTML markup elements *(tags)* and how they work. By the time you finish Part II, expect to have a good overall idea of what HTML is and how you can use it.

Part III: Taking Precise Control Over Web Pages and Styles

Part III starts with a discussion of Cascading Style Sheets (CSS) — another form of markup language that lets (X)HTML deal purely with content while it deals with how Web pages look when they're displayed in a Web browser or as rendered on other devices (PDAs, mobile phones, and special so-called assistive devices for print-handicapped users). After exploring CSS syntax and structures and discovering how to use them, you find out how to manipulate the color and typefaces of text, backgrounds, and more on your Web pages. You also learn about more complex collections of markup — specifically tables — as you explore and observe their capabilities in detail. We give you lots of examples to help you design and build commercial-grade (X)HTML documents. You can get started working with related (X)HTML tag syntax and structures that you need to know so you can build complex Web pages.

Part IV: Integrating Scripts with (X)HTML

(X)HTML isn't good at snazzing up text and graphics when they're on display (that's where CSS excels). And (X)HTML really can't *do* much by itself. Web designers often build interactive, dynamic Web pages by using scripting tools to add interactivity to an (X)HTML framework.

In this part of the book, you find out about scripting languages that enable Web pages to interact with users and that also provide ways to respond to user input or actions and to grab and massage data along the way. You get introduced to general scripting languages, and we jump directly into the most popular of such languages — JavaScript. You can discover the basic elements of this scripting language and how to add interactivity to Web pages. You can also explore typical uses for scripting that you can extend and add to your own Web site. We go on to explore how to create and extract data from Web-based data input forms and how to create and use scripts that react to a user's actions while she visits your Web pages.

Throughout this part of the book, examples, advice, and details show you how these scripting components can enhance and improve your Web site's capabilities — and your users' experiences when visiting your pages.

Part V: (X)HTML Projects

This part tackles typical complex Web pages. You can use these as models for similar capabilities in your own Web pages. These projects include personal and company pages, an eBay auction page, and even a product catalog page with its own shopping cart!

Part VI: The Part of Tens

We sum up and distill the very essence of the mystic secrets of (X)HTML. Here you can read further about cool Web tools, get a second chance to review top dos and don'ts for HTML markup, and review how to catch and kill potential bugs and errors in your pages before anybody else sees them. You also get a collection of killer online resources you can use to further your own ongoing education in HTML, XHTML, and CSS over time.

Icons Used in This Book

TECHNICAL STUFF

This icon signals technical details that are informative and interesting but aren't absolutely critical to writing HTML.

TIP

This icon flags useful information that makes HTML markup or other important stuff even less complicated than you feared it might be.

REMEMBER

This icon points out information you shouldn't pass by — don't overlook these gentle reminders (the life, sanity, or page you save could be your own).

WARNING!

Be cautious when you see this icon. It warns you of things you shouldn't do; consequences can be severe if you ignore the accompanying bit of wisdom.

ON THE WEB

Text marked with this icon contains information about something that can be found on this book's companion Web site. You can find all the code examples in this book, for starters. Simply visit our Web site for this book at `www.edtittel.com/html4d6e`, and look for pointers to examples, templates, and more. We also use this icon to point out some great and useful Web resources.

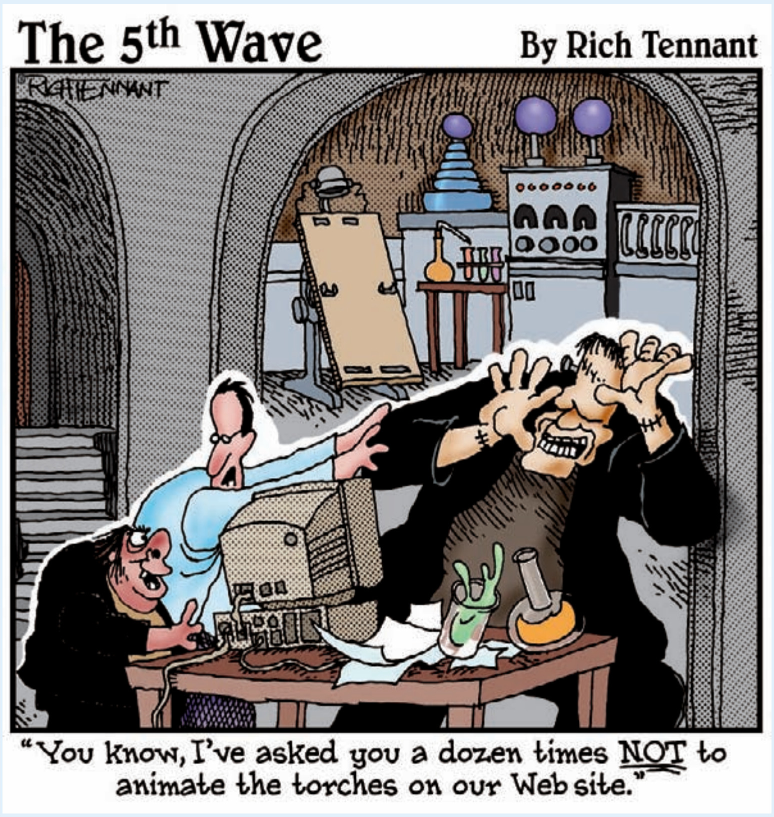
TRICKS OF THE TRADE

The information highlighted with this icon gives best practices — advice that we wish we'd had when we first started out! These techniques can save you time and money on migraine medication.

Where to Go from Here

This is where you pick a direction and hit the road! Where you start out doesn't matter. Don't worry. You can handle it. Who cares whether anybody else thinks you're just goofing around? We know you're getting ready to have the time of your life. Enjoy!

Part I
Getting to Know (X)HTML and CSS



"You know, I've asked you a dozen times NOT to animate the torches on our Web site."

In this part . . .

In this part of the book, we explore and explain basic HTML document links and structures. We also explain the key role that Web browsers play in delivering all this stuff to people's desktops. We even explain where the *(X)* comes from — namely, a reworking of the original description of HTML markup using XML syntax to create XHTML — and go on to help you understand what makes XHTML different (and possibly better, according to some) than plain old HTML. We also take a look at general Web-page anatomy, at the various pieces and parts that make up a Web page, and at how CSS helps to manage their presentation, placement, and even color when they appear on somebody's display.

Next, we take you through the exercise of creating and viewing a simple Web page so you can understand what's involved in doing this for yourself. We also explain what's involved in making changes to an existing Web page and how to post your changes (or a new page) online.

This part concludes with a rousing exhortation to figure out what you're doing before making too much markup happen. Just as a well-built house starts with a set of blueprints and architectural drawings, so should a Web page (and site) start with a plan or a map, with some idea of where your pages will reside in cyberspace and how hordes of users can find their way to them.

Chapter 1

The Least You Need to Know about HTML, CSS, and the Web

In This Chapter

- Creating HTML in text files
- Serving and browsing Web pages
- Understanding links and URLs
- Understanding basic HTML syntax
- Understanding basic CSS

Welcome to the wonderful world of the Web, (X)HTML, and CSS. With just a little knowledge, some practice, and something to say, you can either build your own little piece of cyberspace or expand on work you've already done.

This book is your down-and-dirty guide to putting together your first Web page, sprucing up an existing Web page, or creating complex and exciting pages that integrate intricate designs, multimedia, and scripting.

The best way to start working with HTML is to jump right in, so that's what this chapter does: It brings you up to speed on the basics of how (X)HTML and CSS work behind the scenes of Web pages, introducing you to their underlying building blocks. When you're done with this chapter, you'll know how (X)HTML and CSS work so you can start creating Web pages right now.

Web Pages in Their Natural Habitat

Web pages can accommodate many kinds of content, such as *text, graphics, forms, audio and video files,* and *interactive games.*

Browse the Web for just a little while and you see a buffet of information and content displayed in many ways. Every Web site is different, but most have one thing in common: Hypertext Markup Language (HTML). You'll also run into XHTML and Cascading Style Sheets (CSS) pretty regularly too.

Whatever information a Web page contains, every Web page is created in HTML (or some reasonable facsimile). HTML is the mortar that holds a Web page together; the graphics, content, and other information are the bricks; CSS tells Web pages how they should look when on display.

TECHNICAL STUFF

HTML files that produce Web pages are just text documents, as are XHTML and CSS files. That's why the Web works as well as it does. Text is a universal language for computers. Any text file you create on a Windows computer — including any HTML, XHTML, or CSS file — works equally well on a Mac or any other operating system.

But Web pages aren't *merely* text documents. They're made with special, attention-deprived, sugar-loaded text called *HTML, XHTML,* or *CSS.* Each uses its own specific set of instructions that you include (along with your content) inside text files that specify how a page should look and behave.

Stick with us to discover all the details you need to know about (X)HTML and CSS!

TIP

When we say (X)HTML, we're really talking about HTML and XHTML together. Although they're not identical, they're enough like each other for this kind of reference to make sense.

Hypertext

Special instructions in HTML permit lines of text to point (that is, *link*) to something else in cyberspace. Such pointers are called *hyperlinks.* Hyperlinks are the glue that holds the World Wide Web together. In your Web browser, hyperlinks usually appear in blue and are underlined. When you click one, it takes you somewhere else.

REMEMBER

Hypertext or not, a Web page is a text file, which means you can create and edit a Web page in any application that creates plain text (such as Notepad or TextEdit). Some software tools offer fancy options and applications (covered in Chapter 22) to help you create Web pages, but they generate the same text files that you create with plain-text editors. We're of the opinion, though, that those just getting started with HTML are best served by a simple text editor. Just break out Notepad on the PC (or TextEdit on the Mac) and you're ready to go.

WARNING!

Steer clear of word processors like WordPad or MS Word for creating HTML because they introduce all kinds of extra code on Web pages that you may neither want nor need.

The World Wide Web comes by its name honestly. It's quite literally a web of online pages hosted on Web servers around the world, connected in trillions of ways by hyperlinks that tie one page to another. Without such links, the Web would be just a bunch of standalone pages.

Much of the Web's value comes from its ability to link to pages and other resources (such as images, downloadable files, and media presentations) on either the same Web site or at another site. For example, FirstGov (`www.firstgov.gov`) is a *gateway* Web site — its sole function is to provide access to other Web sites. If you aren't sure which government agency handles first-time loans for homebuyers, or if want to know how to arrange a tour of the Capitol, visit the site shown in Figure 1-1 to find out.

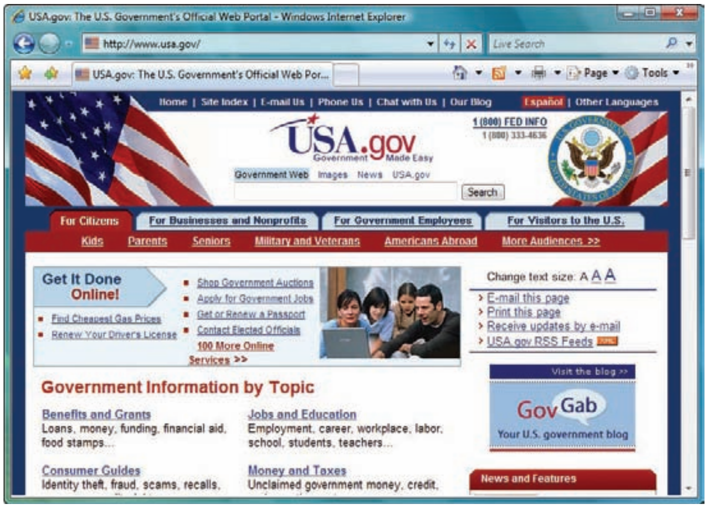

USA.gov: The U.S. Government's Official Web Portal - Windows Internet Explorer
http://www.usa.gov/
Live Search
USA.gov: The U.S. Government's Official Web Por...
Page
Tools
Home | Site Index | E-mail Us | Phone Us | Chat with Us | Our Blog
Español | Other Languages
USA.gov Government Made Easy
1 (800) FED INFO
1 (800) 333-4636
Government Web Images News USA.gov
Search
For Citizens
For Businesses and Nonprofits
For Government Employees
For Visitors to the U.S.
Kids Parents Seniors Military and Veterans Americans Abroad More Audiences >>
Get It Done Online!
Find Cheapest Gas Prices
Renew Your Driver's License
Shop Government Auctions
Apply for Government Jobs
Get or Renew a Passport
Contact Elected Officials
100 More Online Services >>
Change text size: A A A
E-mail this page
Print this page
Receive updates by e-mail
USA.gov RSS Feeds
Visit the blog >>
Gov Gab
Your U.S. government blog
Government Information by Topic
Benefits and Grants
Loans, money, funding, financial aid, food stamps...
Jobs and Education
Employment, career, workplace, labor, school, students, teachers...
Consumer Guides
Identity theft, fraud, scams, recalls,
Money and Taxes
Unclaimed government money, credit,
News and Features

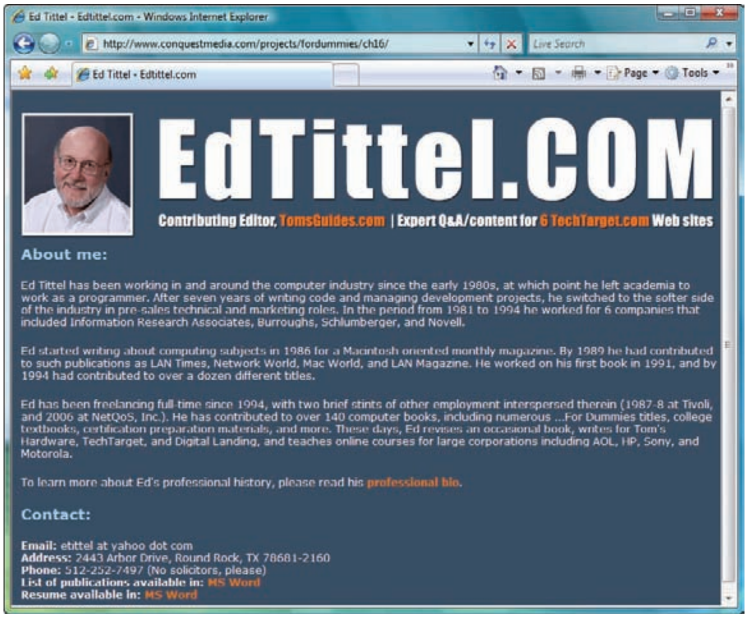
Figure 1-1: USA.gov uses hyperlinks to help visitors find government information.

Markup

Web browsers were created specifically for the purpose of reading HTML instructions (known as *markup*) and displaying the resulting Web page.

Markup lives in a text file (with your content) to give orders to a browser.

For example, look at the page shown in Figure 1-2. You can see how the page is made up and how it is formatted by examining its underlying HTML.


Ed Tittel - Edtittel.com - Windows Internet Explorer
http://www.conquestmedia.com/projects/fordummies/ch16/
Live Search
Ed Tittel - Edtittel.com
Page
Tools

EdTittel.COM

Contributing Editor, TomsGuides.com | Expert Q&A/content for 6 TechTarget.com Web sites

About me:

Ed Tittel has been working in and around the computer industry since the early 1980s, at which point he left academia to work as a programmer. After seven years of writing code and managing development projects, he switched to the softer side of the industry in pre-sales technical and marketing roles. In the period from 1981 to 1994 he worked for 6 companies that included Information Research Associates, Burroughs, Schlumberger, and Novell.

Ed started writing about computing subjects in 1986 for a Macintosh oriented monthly magazine. By 1989 he had contributed to such publications as LAN Times, Network World, Mac World, and LAN Magazine. He worked on his first book in 1991, and by 1994 had contributed to over a dozen different titles.

Ed has been freelancing full-time since 1994, with two brief stints of other employment interspersed therein (1987-8 at Tivoli, and 2006 at NetQoS, Inc.). He has contributed to over 140 computer books, including numerous ...For Dummies titles, college textbooks, certification preparation materials, and more. These days, Ed revises an occasional book, writes for Tom's Hardware, TechTarget, and Digital Landing, and teaches online courses for large corporations including AOL, HP, Sony, and Motorola.

To learn more about Ed's professional history, please read his professional bio.

Contact:

Email: etittel at yahoo dot com
Address: 2443 Arbor Drive, Round Rock, TX 78681-2160
Phone: 512-252-7497 (No solicitors, please)
List of publications available in: MS Word
Resume available in: MS Word

Figure 1-2: This Web page incorporates multiple parts and numerous bits of HTML and CSS markup.

This page includes an image, a heading that describes the page, several paragraphs of text about one of your authors, and an address block with links to a résumé and a list of publications.

However, different components of the page use different formatting:

- The heading at the top of the page is larger than text in the paragraphs.
- Blocks of text are separated by more blank space than between contiguous lines of text within blocks.
- Some text is in white, some orange, and some light blue.

The browser knows to display these components of the page in specific ways thanks to the *HTML markup,* shown in Listing 1-1. (You'll see Listing 1-1 in all its glory at the end of the chapter.)

Any text enclosed between less-than and greater-than signs (< >) is an HTML *tag* (often called the *markup*). For example, a `p` within brackets (`<p>`…`</p>` tags) identifies the text in paragraphs. The markup between the `<style>` and `</style>` tags at the head of the file uses CSS to define the look and feel for various HTML elements used on this page. That's really all there is to it. You embed the markup in a text file, along with text for readers to view, to tell the browser how to display your Web page.

Tags and the content between (and within) the tags are collectively called elements. Angle brackets `< >` enclose HTML and XHTML markup, curly braces `{ }` enclose CSS markup.

Browsers

The user's piece in the Web puzzle is a Web browser. Web browsers read instructions written in HTML and use those instructions to display a Web page's content on your screen.

You should always write your HTML with the idea that people will view the content using a Web browser. Just remember that there's more than one kind of browser out there, and each one comes in several versions.

Usually, Web browsers request and display Web pages available via the Internet from a Web server. You can also display HTML pages you've saved on your own computer before making them available on a Web server on the Internet. When you're developing your own HTML pages, you view these pages (called *local* pages) in your browser. You can use local pages to get a good idea of what people see after the page goes live on the Internet.

Each Web browser interprets HTML in its own way. The same HTML may not look exactly the same from one browser to the next. When you work with basic HTML, variations will be minor, but as you integrate other elements (such as scripting and multimedia), rendering markup gets hairy.

Chapter 2 shows how to use a Web browser to view a local copy of your first Web page.

Some people use text-only Web browsers, such as Lynx, because either

- They're visually impaired and can't use a graphical display.
- They like a lean, fast Web browser that displays only text.

Web servers

Your HTML pages aren't much good if you can't share them with the world. Web servers make that possible. A *Web server* is a computer that

- Connects to the Internet
- Runs Web-server software
- Responds to requests from Web browsers for Web pages

A bevy of browsers

The Web world is full of browsers of many shapes and sizes — or rather versions and feature sets. Two of the more popular browsers are Microsoft Internet Explorer and Mozilla Firefox. Other browsers, such as Apple Safari and Opera, are also widely used. As an HTML developer, you must think beyond your own browser experience and preferences. Every user has his or her own browser preferences and settings.

Each browser renders HTML a bit differently. Every browser handles JavaScript, multimedia, style sheets, and other HTML add-ins differently too. Throw different operating systems into the mix, and things get really fun.

Usually the differences between browsers are minor. But sometimes a combination of HTML, text, and media brings a specific browser to its knees. When you work with HTML, test your pages on as many different browsers as you can. Install at least three different browsers on your own system for testing. We recommend the latest versions of Internet Explorer, Firefox, and Opera.

Yahoo! has a fairly complete list of browsers at

```
http://dir.yahoo.com/Computers_and_Internet/Software/Internet/World_Wide_Web/Browsers
```

Almost any computer can be a Web server, including your home computer. But Web servers generally are computers dedicated to the task. You don't need to be an Internet or computer guru to publish your Web pages, but you must find a Web server to serve your pages:

- ✓ If you're building pages for a company Web site, your IT department may have a Web server. (Ask your IT guru for the information.)
- ✓ If you're starting a new site, you need a host for your pages.

TIP

Finding an inexpensive host is easy. Chapter 3 shows how to determine your hosting needs and find the perfect provider.

Anatomy of a URL

The Web is made up of billions of resources, each of them linkable. A resource's exact location is the key to linking to it. Without an exact address (a *Uniform Resource Locator,* or *URL*), you can't use the Address bar in a Web browser to visit a Web page directly.

REMEMBER

URLs are the standard addressing system for Web resources. Each resource (Web page, site, or individual file) has a unique URL. URLs work a lot like your postal address. Figure 1-3 identifies the components of a URL.

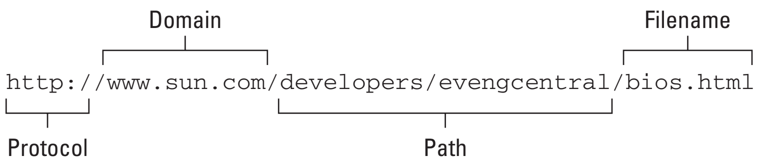


Figure 1-3: The components of a URL help it define the exact location of a file on the Web.

Each URL component helps define the location of a Web page or resource:

- **Protocol:** Specifies the protocol the browser follows to request the file.

  The Web page protocol is `http://` (the usual start to most URLs).
- **Domain:** Points to the general Web site (such as `www.sun.com`) where the file resides. A domain may host a few files (like a personal Web site) or millions of files (like a corporate site, such as `www.sun.com`).
- **Path:** Names the sequence of folders through which you must navigate to get to a specific file.

  For example, to get to a file in the `evangcentral` folder that resides in the `developers` folder, you use the `/developers/evangcentral/` path.
- **Filename:** Specifies which file in a directory path the browser accesses.

Although the URL shown in Figure 1-3 is no longer publicly accessible, it points to the Sun domain and offers a path that leads to a specific file named `bios.html`:

```
http://www.sun.com/developers/evangcentral/bios.html
```

TIP

Chapter 6 provides the complete details on how you use HTML and URLs to add hyperlinks to your Web pages, and Chapter 3 shows how to obtain a URL for your own Web site after you're ready move it to a Web server.

(X)HTML's Component Parts

The following section removes the mystery from the *X.* This section shows

- The differences between HTML and XHTML
- How HTML is written (its *syntax*)
- Rules that govern its use

- Names for important pieces and parts of HTML (and XHTML) markup
- How to make the best, most correct use of (X)HTML capabilities

HTML and XHTML: What's the difference?

HTML is *Hypertext Markup Language,* a notation developed in the late 1980s and early 1990s for describing Web pages. HTML is now enshrined in numerous standard descriptions (*specifications*) from the World Wide Web Consortium (W3C). The last HTML specification was finalized in 1999.

When you put an *X* in front of *HTML* to get *XHTML,* you get a new, improved version of HTML based on the *eXtensible Markup Language (XML).* XML is designed to work and behave well with computers, software, and the Internet.

The original formulation of HTML has some irregularities that can cause heartburn for software that reads HTML documents. XHTML, on the other hand, uses an extremely regular and predictable syntax that's easier for software to handle. XHTML will replace HTML someday, but HTML keeps on ticking. This book covers both varieties and shows you the steps to put the X in front of your own HTML documents and turn them into XHTML.

Introducing Internet protocols

Interactions between browsers and servers are made possible by a set of computer-communication instructions: Hypertext Transfer Protocol (HTTP). This protocol defines how browsers should request Web pages and how Web servers should respond to those requests.

HTTP isn't the only protocol at work on the Internet. The Simple Mail Transfer Protocol (SMTP) and Post Office Protocol (POP) make e-mail exchange possible, and the File Transfer Protocol (FTP) allows you to upload, download, move, copy, and delete files and folders across the Internet. The good news is that Web browsers and servers do all the HTTP work for you, so you only have to put your pages on a server or type a Web address into a browser.

To see how HTTP works, check out David Gourley and Brian Totty's chapter on HTTP messages, available through Google book search with "understanding http transactions" as the search string. Start your search at `http://google.books.com`, then scroll down until you see the link to "HTTP: The Definitive Guide – Page 80."

- Most HTML and XHTML markup are identical.
- In a few cases, HTML and XHTML markup look a little different.
- In a few cases, HTML and XHTML markup must be used differently.

This book shows how to create code that works in both HTML and XHTML.

Syntax and rules

HTML is a straightforward language for describing Web page contents. XHTML is even less demanding. Their components are easy to use — when you know how to use a little bit of (X)HTML. Both HTML and XHTML markup have three types of components:

- **Elements:** Identify different parts of an HTML page by using tags
- **Attributes:** Information about an instance of an element
- **Entities:** Non-ASCII text characters, such as copyright symbols (©) and accented letters (É). Entities originate from the Standard Generic Markup Language, or SGML.

Every bit of HTML and/or XHTML markup that describes a Web page's content includes some combination of elements, attributes, and entities.

Markup color coding

As we present HTML, XHTML, and CSS information in our code samples, we use color coding to help you distinguish what's what by way of markup. Here is a color key that you should keep in mind as you read all of our code listings.

```
Purple: indicates the DOCTYPE declaration used in (X)HTML documents. This is
          actually a totally different markup language known as the Standard
          Generalized Markup Language, or SGML. It is used to identify what specific
          set of rules that (X)HTML documents follow in their construction and
          content. It also applies to codes for character entities, which take the
          form &pos; or &123;.
Light Green: indicates ordinary garden variety XHTML and HTML markup
Dark Green: indicates XML markup
Orange: indicates Cascading Style Sheet, or CSS, markup
Blue: indicates JavaScript
```

We only colorize markup in code listings, because it affects readability too much when code appears in body copy. In that case, we simply use a different, monospaced font — as you'll see in the discussions of the `<html>`, `<head>`, and `<title>` elements in our first paragraph that discusses HTML markup here.

This chapter covers the basic form and syntax for elements, attributes, and entities. Parts II and III of the book detail how elements and attributes:

- Describe kinds of text (such as paragraphs or tables)
- Create an effect on the page (such as changing a font style)
- Add images and links to a page

Elements

Elements are the building blocks of (X)HTML. You use them to describe every piece of text on your page. Elements are made up of tags and the content within those tags. There are two main types of elements:

- Elements with content made up of a tag pair and whatever content sits between the opening and closing tag in the pair
- Elements that insert something into the page, using a single tag

Tag pairs

Elements that describe content use a *tag pair* to mark the beginning and the end of the element. Start and end tag pairs look like this:

```
<tag>...</tag>
```

Content — such as paragraphs, headings, tables, and lists — always uses a tag pair:

- The start tag (`<tag>`) tells the browser, “The element begins here.”
- The end tag (`</tag>`) tells the browser, “The element ends here.”

The actual content is what occurs between the start tag and end tag. For example, the Ed Tittel page in Listing 1-1 uses the paragraph element (`<p>`) to surround the text of a paragraph (we omit CSS inline markup for clarity):

```
<p>Ed started writing about computing subjects in 1986 for a
 Macintosh oriented monthly magazine. By 1989 he had contributed to such
publications as LAN Times, Network World, Mac World, and LAN Magazine. He worked
on his first book in 1991, and by 1994 had contributed to over a dozen different
titles.</p>
```

Single tags

Elements that insert something into the page are called *empty elements* (because they enclose no content) and use just a single tag, like this:

```
<tag />
```

Images and line breaks insert something into the HTML file, so they use one tag.

One key difference between XHTML and HTML is that, in XHTML, all empty elements must end with a slash before the closing greater-than symbol. This is because XHTML is based on XML, and the XML rule is that you close empty elements with a slash, like this:

```
<tag/>
```

However, to make this kind of markup readable inside older browsers, you must insert a space before the closing slash, like this:

```
<tag />
```

This space allows older browsers to ignore the closing slash (since they don't know about XHTML). Newer browsers that understand XHTML ignore the space and interpret the tag exactly, which is `<tag/>` (as per the XML rules).

HTML doesn't require a slash with empty elements, but this markup is deprecated (that is, identified as obsolete even though it still occurs in some markup). An HTML empty element looks like this:

```
<tag />
```

Listing 1-1 uses the image element (`<img />`) to include an image on the page:

```
<img src="images/header.gif" alt="header graphic" width="794" height="160" />
```

The `<img />` element references an image. When the browser displays the page, it replaces the `<img />` element with the file that it points to (it uses an attribute to do the pointing, which is shown in the next section). Following the XHTML rule introduced earlier, what appears in HTML as `<img>` appears in XHTML as `<img />` (and this applies to all single tag elements).

You can't make up HTML or XHTML elements. Elements that are legal in (X) HTML are a very specific set — if you use elements that aren't part of the (X) HTML set, every browser ignores them. The elements you can use are defined in the HTML 4.01 or XHTML 1.0 specifications. (The specs for HTML 4.01 can be found at `www.w3.org/TR/html4`, while the specs for XHTML 1.0 can be found at `www.w3.org/TR/xhtml1/`.)

Nesting

Many page structures combine nested elements. Think of your nested elements as *suitcases* that fit neatly inside one another.

For example, a bulleted list uses two kinds of elements:

- The `<ul>` element specifies that the list is unordered (bulleted).
- The `<li>` elements mark each item in the list.

When you combine elements by using this method, be sure you close the inside element completely before you close the outside element:

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
</ul>
```

Attributes

Attributes allow variety in how an element describes content or works. Attributes let you use elements differently depending on the circumstances. For example, the `<img />` element uses the `src` attribute to specify the location of the image you want to include on your page:

```
<img src="images/header.gif" alt="header graphic" width="794" height="160" />
```

In this bit of HTML, the `<img />` element itself is a general flag to the browser that you want to include an image; the `src` attribute provides the specifics on the image you want to include — `header.gif` in this instance. Other attributes (such as `width` and `height`) provide information about how to display the image, while the `alt` attribute provides a text alternative to the image that a text-only browser can display (or a text-to-speech reader can say, for the visually impaired).

TIP

Chapter 7 describes the `<img />` element and its attributes in detail.

You include attributes within the start tag of the element you want them with — after the element name but before the ending sign, like this:

```
<tag attribute="value" attribute="value">
```

REMEMBER

XML syntax rules decree that attribute values must always appear in quotation marks, but you can include the attributes and their values in any order within the start tag or within a single tag.

Every (X)HTML element has a collection of attributes that can be used with it, and you can't mix and match attributes and elements. Some attributes can take any text as a value because the value could be anything, like the location of an image or a page you want to link to. Others have a specific list of values the attribute can take, such as your options for aligning text in a table cell.

The HTML 4.01 and XHTML 1.0 specifications define exactly which attributes you can use with any given element and which values (if explicitly defined) each attribute can take.

TIP

Each chapter in Parts II and III covers which attributes you can use with each (X)HTML element. Also, see our online content for complete lists of deprecated (X)HTML tags and attributes.

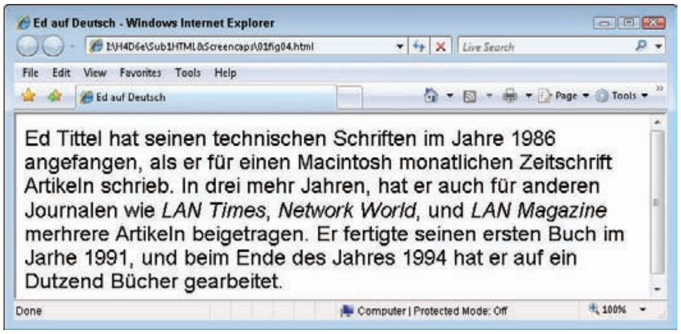
Entities

Text makes the Web possible, but it has limitations. *Entities* are special characters that you can display on your Web page.

Non-ASCII characters

Basic American Standard Code for Information Interchange (ASCII) text defines a fairly small number of characters. It doesn't include some special characters, such as *trademark symbols, fractions,* and *accented characters.*

For example, if we translate a paragraph of text for the page in Figure 1-4 into German, the result includes three *u* characters with umlauts *(ü).*


Ed auf Deutsch - Windows Internet Explorer
I:\H4D6e\Sub1HTML&Screencaps\01fig04.html
Live Search
File Edit View Favorites Tools Help
Ed auf Deutsch
Page
Tools
Done
Computer | Protected Mode: Off
100%

Ed Tittel hat seinen technischen Schriften im Jahre 1986 angefangen, als er für einen Macintosh monatlichen Zeitschrift Artikeln schrieb. In drei mehr Jahren, hat er auch für anderen Journalen wie *LAN Times*, *Network World*, und *LAN Magazine* merhrere Artikeln beigetragen. Er fertigte seinen ersten Buch im Jarhe 1991, und beim Ende des Jahres 1994 hat er auf ein Dutzend Bücher gearbeitet.

Figure 1-4: ASCII text can't represent all text characters, so HTML entities do instead.

ASCII text doesn't include an umlauted *u,* so HTML uses *entities* to represent such characters. The browser replaces the entity with the character it references. Each entity begins with an ampersand (&) and ends with a semicolon (;); entities come originally from SGML, so we color-code them in purple to reflect their origins. The following markup shows the entities in bold:

```
<html>
<head>
<style>
  body {
    font-family: sans-serif;
    font-size: large;
    }
</style>
<title>Ed auf Deutsch</title>
</head>
<body>
<p>Ed Tittel hat seinen technischen Schriften im Jahre 1986 angefangen, als er
f&uuml;r einen Macintosh monatlichen Zeitschrift Artikeln schrieb. In drei mehr
Jahren, hat er auch f&uuml;r anderen Journalen wie <cite>LAN Times</cite>,
<cite>Network World</cite>, und <cite>LAN Magazine</cite> merhrere Artikeln
beigetragen. Er fertigte seinen ersten Buch im Jarhe 1991, und beim Ende des
Jahres 1994 hat er auf ein Dutzend B&uuml;cher gearbeitet.</p>
</body>
</html>
```

The entity that represents the umlauted *u* is `&uuml;`.

(X)HTML character codes

The encodings for the ISO-Latin-1 character set are supplied by default, and related entities (a pointer to a complete table appears in Chapter 23) can be invoked and used without special contortions. But using the other encodings mentioned in Table 1-1 requires inclusion of special markup to tell the browser it must be ready to interpret Unicode character codes. (Unicode is an international standard — ISO standard 10646, in fact — that embraces enough character codes to handle most unique alphabets, plus plenty of other symbols and nonalphabetic characters as well.) This special markup takes the form `<meta http-equiv="Content-Type" content="text/html; charset=UTF 8">`; when the value for `charset` is changed to `UTF-8`, you can reference the common Unicode code charts that appear in Chapter 23 of this book.

Tag characters

HTML-savvy software assumes that some HTML characters, such as the greater-than and less-than signs, are meant to be hidden and not displayed on your finished Web page. If you actually want to show a greater-than or less-than sign on your page, you're going to have to make your wishes clear to the browser. The following entities let you display characters that normally are part of the hidden HTML markup:

- **less-than sign (<):** `&lt;`
- **greater-than sign (>):** `&gt;`
- **ampersand (&):** `&amp;`

REMEMBER

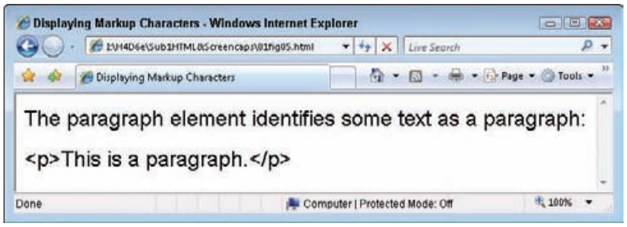
The < and > signs are used in markup, but these symbols are instructions to the browser and won't show up on the page. If you need these symbols on the Web page, include the entities for them in your markup, like this:

```
<p>The paragraph element identifies some text as a paragraph:</p>
<p>&lt;p&gt;This is a paragraph.&lt;/p&gt;</p>
```

In the preceding markup, the first line uses *tags* to describe a paragraph, and the second line shows how *entities* describe the < and > symbols.

Figure 1-5 shows these entities as characters in a browser window.

Figure 1-5: Entities let <, >, or & symbols appear in a browser window.


Displaying Markup Characters - Windows Internet Explorer
Live Search
Displaying Markup Characters
Page
Tools
Done
Computer | Protected Mode: Off
100%

The paragraph element identifies some text as a paragraph:

<p>This is a paragraph.</p>

Parts Is Parts: What Web Pages Are Made Of

Comments include text in (X)HTML files that isn't displayed in the final page. Each comment is identified with two special sequences of markup characters:

- Begin each comment with the string `<!--`
- End each comment with the string `-->`

In the following code, comments explain how each markup element functions and where it fits into the HTML markup hierarchy.

Elements are organized into a structure:

- Some elements can occur only inside other elements.
- Some elements are required for a well-structured (X)HTML document.

```
<html>  <!-- This tag should always occur at or near the beginning of any
             well-formed HTML document -->
<head>  <!-- The head element supplies information to label the whole HTML
             document -->
<title>Welcome to Ed Tittel.com</title> <!-- The text in the title element
             appears in the title bar of the browser window when the page
             is viewed -->
</head> <!-- closes the head element -->
```

```
<body>  <!-- The content that appears on any Web page appears or is
             invoked from inside the body element -->
        <!-- Skip a bunch of copy here . . . -->
<h1>Contact:</h1> <!óinsert level 1 head to set off the header text -->
<!-- The next four lines of text are explicitly broken at their ends using
     the break element <br />, and each line begins with identifying text
     in bold set off between <b> and </b> tags. -->
<p><b>Email:</b> etittel at yahoo dot com<br />
<b>Address:</b> 2443 Arbor Drive, Round Rock, TX 78681-2160<br />
<b>Phone:</b> 512-252-7497 (No solicitors, please)<br />
<b>List of publications available in:</b> <a href="docs/v_et.doc">MS
             Word</a><br />
<b>Resume available in:</b> <a href="docs/Resu-et13.doc">MS
             Word</a>
</p>     <!-- End of contact info paragraph -->
</body>  <!-- End of the body section -->
</html>  <!-- End of the HTML document -->
```

The preceding document is broken into a head and a body. Within each section, certain kinds of elements appear. Many combinations are possible — and that's what you see throughout this book!

Organizing HTML text

Beyond the division into head and body sections, text can be organized in plenty of ways in HTML documents.

Document heads

Inside the head section, you can define all kinds of labels and information besides a title, primarily to describe the document that follows, such as the character sets used, meta data about the current document, scripts to be invoked, and style information. The body section is where real content lives and most (X)HTML elements appear.

Document headings

Headings (denoted using elements h1 through h6) are different from the HTML document head. Individual headings structure the text that follows them, whereas the head identifies or describes the whole document.

TECHNICAL STUFF

In the Ed Tittel page example, the h1 element sets off the Contact block at the bottom of the page.

Paragraphs and more

When you want running text on a Web page, the paragraph element, p (which includes the <p> and </p> tags), breaks text into paragraphs. You can also

- Force line breaks by using the break element `<br />`
- Create horizontal rules (lines) by using the `<hr />` element

HTML also includes all kinds of ways to emphasize or identify text inside paragraphs; Parts II and III of this book show a few of them.

Lists

HTML permits easy definition of unordered or bulleted lists. Various mechanisms to create other kinds of lists, including numbered lists, are also available. Lists can be nested within lists to create as many levels of hierarchy as your list might need (perhaps when outlining a complex subject or modeling a table of contents with several heading levels you want to represent). Chapter 5 covers creating lists in more detail.

Tables

In addition to providing a variety of listing mechanisms, HTML also includes markup for defining tables. (Chapter 11 has more on tables.) Structure is part of how markup works, so within the definition of a table, you can

- Distinguish between column heads and table data
- Manage how rows and columns are laid out

Cascading Style Sheet markup

CSS markup can occur in separate style-sheet documents, in a block of text in the head of an HTML document, or appended in the style attribute within individual HTML elements — and even in some combination of all three such forms! What CSS does is provide much more detailed control over font selection, use of color for text and backgrounds, positioning of text and other elements on the page, and (as the old Ronco ad intones) “much, much more.”

You’ll delve into CSS in detail in Part III of this book, but we cover bits and pieces of CSS throughout the book as appropriate for the subject matter at hand. You can build a Web site without using CSS, and using CSS makes more work, but it’s the right tool for precise control over look and layout!

Images in HTML documents

Adding an image to any HTML document is easy. Careful and well-planned use of images adds a lot to Web pages. Chapter 7 shows how to grab images from files. Chapter 9 shows how to use complex markup to position and flow text around graphics. You also discover how to select and use interesting and compelling images to add both allure and information to your Web pages.

Links and navigation tools

A Web page's structure should help visitors find their way around collections of Web pages, look for (and hopefully, find) items of interest, and get where they most want to go quickly and easily. Links provide the mechanism to bring people into your Web pages, so Chapter 6 shows how to

- Reference external items or resources
- Jump from one page to the next
- Jump around inside a page
- Add structure and organization to your pages

  The importance of structure and organization goes up as the amount of information that you want to present to your visitors goes up.

Navigation tools, (which establish standard mechanisms and tools for moving around inside a Web site) provide ways to create and present your Web page (and site) structure to visitors as well as mechanisms for users to grab and use organized menus of choices

When you add everything up, your result should be a well-organized set of information and images that's easy to understand, use, and navigate.

Listing 1-1: Meet an Author!

Listing 1-1 is reproduced in its entirety here, color-coded to distinguish the various types of markup it uses. Lest you think this is mere vanity on Ed's part, we also hasten to point out that this is the basis for the "About Me" page described in Chapter 16 of this book, which we hope only makes it the more interesting, rather than the reverse!

Listing 1-1: Ed Tittel's "About Me" Web page

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Ed Tittel - Edtittel.com</title>
<style type="text/css">

body {
  background-image: url(images/background_page.gif);
  }

.white_text {
  color: #FFFFFF;
  font-family: Verdana, Arial, Helvetica, sans-serif;
  font-size: 12px;
  }

.bold_text {
  font-weight: bold;
  }

h1 {
  font-family: Verdana, Arial, Helvetica, sans-serif;
  font-weight: bold;
  font-size: 17px;
  color:#96CDFF;
  }

a:link {
  font-weight : bold;
  text-decoration : none;
  color: #FF7A00;
  background: transparent;
  }
a:visited {
  font-weight : bold;
  text-decoration : none;
  color: #91a3b4;
  background: transparent;
  }

a:hover {
  color: #FA0000;
  background: transparent;
  text-decoration : underline;
  }

a:active {
  color: #494949;
  background: transparent;
  font-weight : bold;
  text-decoration : underline;
  }

</style>
</head>

<body>
<table width="794" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td>
<!-- Top graphic of Ed and title -->
    <img src="images/header.gif" alt="header graphic" width="794"
     height="160"></td>
  </tr>
  <tr>
```

(continued)

Listing 1-1 *(continued)*

```
<!-- Text about Ed -->
    <td class="white_text"><h1>About me: </h1>
    <p class="white_text">
    Ed Tittel has been working in and around the computer industry since the
    early 1980s, at which point he left academia to work as a programmer. After
    seven years of writing code and managing development projects, he switched
    to the softer side of the industry in pre-sales technical and marketing
    roles. In the period from 1981 to 1994 he worked for 6 companies that
    included Information Research Associates, Burroughs, Schlumberger, and
    Novell.</p>
    <p class="white_text">Ed started writing about computing subjects in 1986
    for a Macintosh oriented monthly magazine. By 1989 he had contributed to
    such publications as LAN Times, Network World, Mac World, and LAN Magazine.
    He worked on his first book in 1991, and by 1994 had contributed to over a
    dozen different titles.</p>
    <p class="white_text">Ed has been freelancing full-time since 1994, with two
    brief stints of other employment interspersed therein (1987-8 at Tivoli,
    <br /> and 2006 at NetQoS, Inc.). He has contributed to over 140 computer
    books, including numerous ...For Dummies titles, college textbooks,
    certification preparation materials, and more. These days, Ed revises an
    occasional book, writes for Tom's Hardware, TechTarget, and Digital Landing,
    and teaches online courses for large corporations including AOL, HP, Sony,
    and Motorola.</p>
    <p class="white_text">To learn more about Ed's professional history, please
    read his <a href="bio.htm">professional bio</a>.</p>
  <h1>Contact:</h1>
    <p class="white_text"><span class="bold_text">Email:</span> etittel at yahoo
    dot com<br />
    <span class="bold_text">Address:</span> 2443 Arbor Drive, Round Rock, TX
    78681-2160<br />
    <span class="bold_text">Phone:</span> 512-252-7497 (No solicitors,
    please)<br />
    <span class="bold_text">List of publications available in:</span>
    <a href="docs/v_et.doc" target="_blank">MS Word</a><br />
    <span class="bold_text">Resume available in:</span>
    <a href="docs/Resu-et13.doc" target="_blank">MS Word</a>
    </p>
    </td>
  </tr>
</table>
</body>
</html>
```

That’s a huge amount of HTML to pore over at the very beginning of this book. Please take our word for it, though: If you read enough of this book’s contents, all of it will make perfect sense!

Chapter 2

Creating and Viewing a Web Page

In This Chapter

- Planning your Web page
- Writing some HTML
- Saving your page
- Viewing your page offline and online

Creating your very own Web page may seem daunting, but it's definitely fun, and our experience tells us that the best way to get started is to jump right in with both feet. You might splash around a bit at first, but you can keep your head above water without too much thrashing.

This chapter walks you through four basic steps to create a Web page. We don't stop and explain every nuance of the markup you use — we save that for other chapters. Instead, we want to make you comfortable working with markup, and content to create and view a suitably simple Web page.

Before You Get Started

Creating HTML documents differs from creating word-processor documents using an application like Microsoft Word because you end up having to use two applications:

- You create the Web pages in your text or HTML editor.
- You view the results in your Web browser.

Even though many HTML editors, such as Dreamweaver and HTML-Kit, provide a browser preview, it's still important to preview your Web pages inside actual Web browsers (such as Internet Explorer, Firefox, or Safari) so you can see them as your end users do. It might feel a bit unwieldy to edit inside one application and then switch to another to look at your work, but you'll be switching from text editor to browser and back like a pro in (almost) no time.

To get started on your first Web page, you need two types of software:

- **A text editor, such as Notepad, TextPad, or SimpleText**

  We discuss these tools in more detail in Chapter 22, but here's the thumbnail sketch. Notepad is the native text editor in Windows. TextPad is a shareware text editor available from `www.textpad.com`. SimpleText is the native text editor in the Macintosh operating system.
- **A Web browser**

We're going to recommend that you use a plain text editor for your first Web page and here's why:

- An advanced HTML editor, such as HotDog Professional or Dreamweaver, often *hides* your HTML from you. For your first page, you want to see your HTML in all of its (limited) glory.

  You can make a smooth transition to a more advanced editor after you become familiar with (X)HTML and CSS markup, syntax, and document structure.
- Word processors decked out with all the bells and whistles (such as Microsoft Word, in other words) usually insert lots of extra file information behind the scenes (for example, formatting instructions to display or print files). You can't see or change that extra information while you're editing, but what's worse, it interferes with your (X)HTML.

Creating a Page from Scratch

Using HTML to create a Web page from scratch involves four straightforward steps:

1. **Plan your page design.**
2. **Combine HTML and text in a text editor to make that design a reality.**
3. **Save your page.**
4. **View your page in a Web browser.**

So break out your text editor and Web browser — and roll up your sleeves.

Step 1: Planning a simple design

We've discovered that a few minutes spent planning your general approach to a page at the outset of work makes the page-creation process faster and easier.

You don't have to create a complicated diagram or elaborate graphical display in this step. Just jot down some ideas for *what you want on the page* and *how you want it arranged.*

TIP

You don't even have to be at your desk to plan a simple design. Take a notepad and pencil outside and design in the sun, or scribble on a napkin while you're having lunch. Remember, this is supposed to be fun.

The example in this chapter is our take on the traditional "Hello World" exercise used in just about every existing programming language: The first thing you learn when tackling a new programming language is how to display the phrase `Hello World` on-screen. In our example, we create a short letter to the world instead, so the page is a bit more substantial and gives you more text to work with. Figure 2-1 shows our basic design for this page.

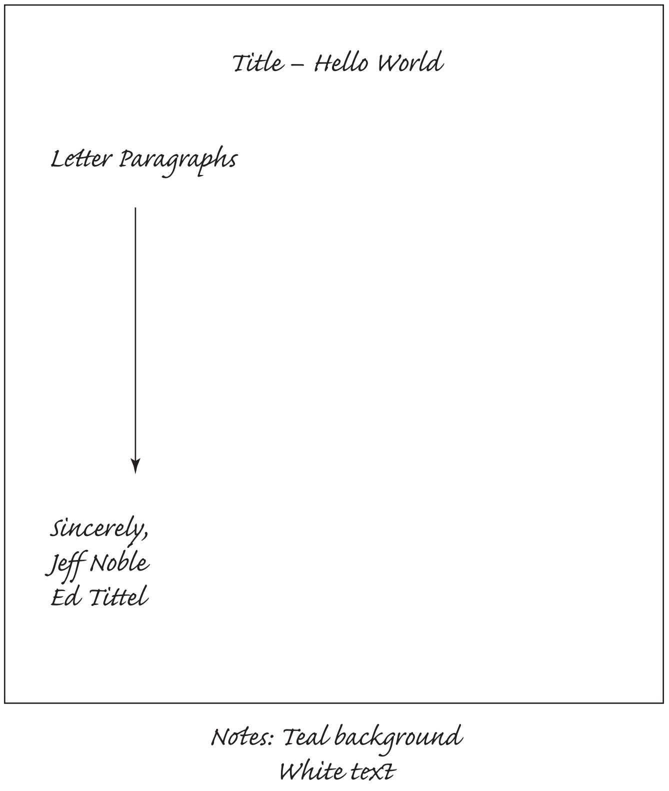


Figure 2-1: Taking a few minutes to sketch your page design makes writing HTML easier.

The basic design for the page includes four basic components:

- A serviceable title: "Hello World!"
- A few paragraphs explaining how HTML can help you communicate with the whole world
- A closing of "Sincerely"
- A signature

TIP

Jot down some notes about the color scheme you want to use on the page. For our example page we use a teal background and white text, and its title should be "HTML Makes the Web Go Round."

When you know what kind of information you want on the page, you can move on to Step 2 — writing the markup.

Step 2: Writing some HTML

You have a couple of different options when you're ready to create your HTML. In the end, you'll probably use some combination of these:

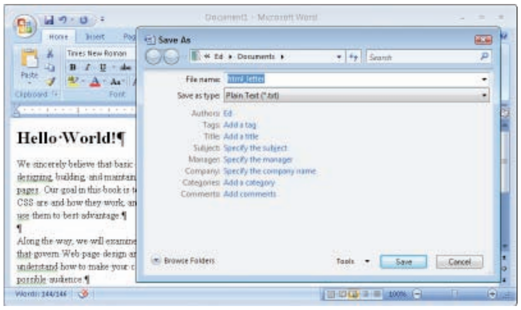
- If you already have some text that you just want to describe with HTML, save that text as a plain-text file and add HTML markup around it.
- Start creating markup and add the content as you go.

TIP

Our example in this chapter starts with some text in Word document format. We saved the content as a text file, opened the text file in our text editor, and added markup around the text.

To save a Word file as a text document, choose File➪Save As. In the dialog box that appears, choose Text Only (*.txt) from the Save As Type drop-down list.

Figure 2-2 shows how our draft letter appears in Microsoft Word before we convert it to text for our page.



Hello World!¶

We sincerely believe that basic
designing, building, and maintain
pages. Our goal in this book is t
CSS are and how they work, an
use them to best advantage.¶
¶
Along the way, we will examine
that govern Web page design a
understand how to make your c
possible audience.¶

File name: html letter
Save as type: Plain Text (*.txt)
Authors: Ed
Tags: Add a tag
Title: Add a title
Subject: Specify the subject
Manager: Specify the manager
Company: Specify the company name
Categories: Add a category
Comments: Add comments

Figure 2-2: The letter that is the text for our page, in word-processing form.

Listing 2-1 shows you what you must add to the prose from Microsoft Word to turn it into a fully functional HTML file.

Listing 2-1: The Complete HTML Page for the 'Hello World!' Letter

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

  <head>
    <title>HTML Makes the Web Go Round</title>
  </head>

  <body type="text/css"
        style="color: white;
               background-color: teal;
               font-size: 1.2;
               font-family: sans-serif">

  <h1>Hello World!</h1>

  <p>We sincerely believe that basic HTML knowledge is essential to
     designing, building, and maintaining readable and workable Web
     pages. Our goal in this book is to explain what HTML, XHTML, and
     CSS are and how they work, and then to show you exactly how to
     use them to best advantage.
  </p>

  <p>Along the way, we will examine the principles and best practices
     that govern Web page design and construction, and help you
     understand how to make your content accessible to the broadest
     possible audience.
  </p>

  <p>By the time you work your way through this book's contents, you
     should feel comfortable with creating and managing your own Web
     site. You should also understand what it takes to identify your
     audience, communicate with that audience, and keep your content
     fresh and interesting to keep them coming back for more.
  </p>

  <p>Sincerely,<br />
     Jeff Noble and Ed Tittel, your humble authors
  </p>

  </body>
</html>
```

The HTML markup includes a collection of markup elements and attributes that describe the letter's contents:

- The `<html>` element defines the document as an HTML document.
- The `<head>` element creates a header section for the document.
- The `<title>` element defines a document title that is displayed in the browser's title bar.

  The `<title>` element is *inside* the `<head>` element.
- The `<body>` element holds the text that appears in the browser window.

  The markup that follows the `style=" "` attribute inside the `<body>` element is CSS, otherwise known as the *C*ascading *S*tyle *S*heet markup language. It says we want white text on a teal background, where the text is larger than usual, and in a sans-serif font. (You'll find out all about styles and attributes in Chapters 8 and 9.)
- The `<h1>` element marks the `Hello World` text as a first-level heading.
- The `<p>` elements identify each paragraph of the document.
- The `<br />` element adds a manual line break after `Sincerely`.

Don't worry about the ins and outs of how the HTML elements work. They are covered in detail in Chapters 4 and 5. Also, please note that a Web page can include graphics, scripts, and other elements that we deliberately avoid in this contrived and simple example to keep things . . . well . . . simple! We will cover all these extras in profuse detail later in the book, though.

After you create a complete HTML page (or the first chunk of it that you want to review), you must save it before you can see your work in a browser.

Step 3: Saving your page

You use a text editor to create HTML documents and a Web browser to view them, but before you can let your browser loose on your HTML page, you must save that page. When you're just building a page, you should save a copy of it to your local hard drive and view it locally with your browser.

Choosing a location and name for your file

When you save your file to your hard drive, keep the following in mind:

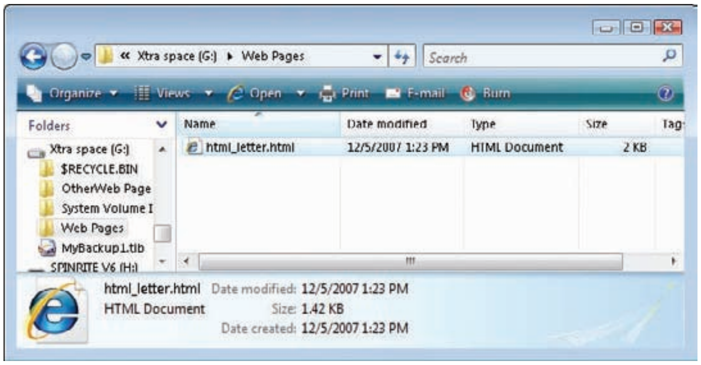
- You need to be able to find it again.

  Create a folder on your hard drive especially for your Web pages. Call it `Web Pages` or `HTML` (or any other name that makes sense to you), and be sure to put it somewhere easy to find.

- The name should make sense to you so you can identify file contents without actually opening the file.
- The name should work well in a Web browser.

  Don’t use spaces in the name. Some operating systems — most notably Unix and Linux (the most popular Web-hosting operating systems around) — don’t tolerate spaces in filenames; use an underscore (_) or hyphen (-) instead. It’s also a good idea to avoid other punctuation characters in filenames, and in general, to keep them as short as you can.

In our example, we saved our file in a folder called `Web Pages` and named it (drum roll, please) `html_letter.html`, as shown in Figure 2-3.



Name	Date modified	Type	Size	Tag
html_letter.html	12/5/2007 1:23 PM	HTML Document	2 KB	

html_letter.html Date modified: 12/5/2007 1:23 PM
HTML Document Size: 1.42 KB
Date created: 12/5/2007 1:23 PM

Figure 2-3: Use a handy location and a logical filename for HTML pages.

.htm or .html

You can actually choose from one of two suffixes for your pages: `.html` or `.htm`. (Our example filename, `html_letter.html`, uses the `.html` suffix.)

The shorter `.htm` is a relic from the “8.3” DOS days when filenames could only include eight characters plus a three-character suffix that described the file’s type. Today, operating systems can support long filenames and suffixes that are longer than three letters, so we suggest you stick with `.html`.

Web servers and Web browsers handle both `.htm` and `.html` equally well.

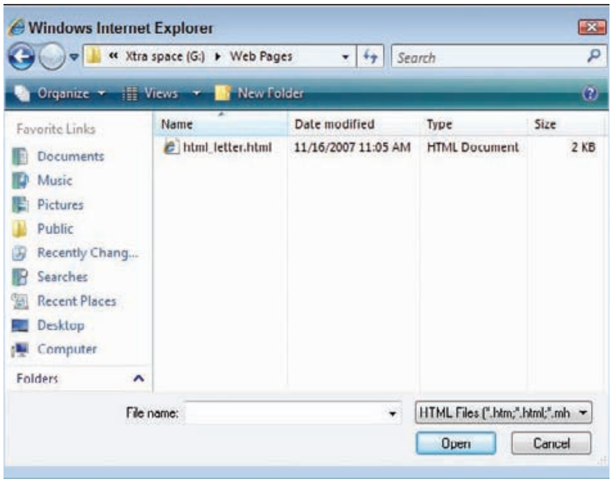
Stick with one filename option. `.html` and `.htm` files are treated the same by browsers and servers, but they’re actually different suffixes, so they create different filenames. (The name `html_letter.html` is different from `html_letter.htm`.) This difference matters a lot when you create hyperlinks (covered in Chapter 6).

Step 4: Viewing your page

After you save a copy of your page, you're ready to view it in a Web browser. Follow these steps to view your Web page in Internet Explorer. (Steps may be different if you're using a different browser.)

1. **If you haven't opened your browser, do that now.**
2. **Choose File➪Open.**
3. **In the Open dialog box that appears, click the Browse button.**
4. **In the new dialog that appears, navigate your file system until you find your HTML file, and then select it so it appears in the File name area.**

   Figure 2-4 shows a highlighted HTML file, ready to be opened.



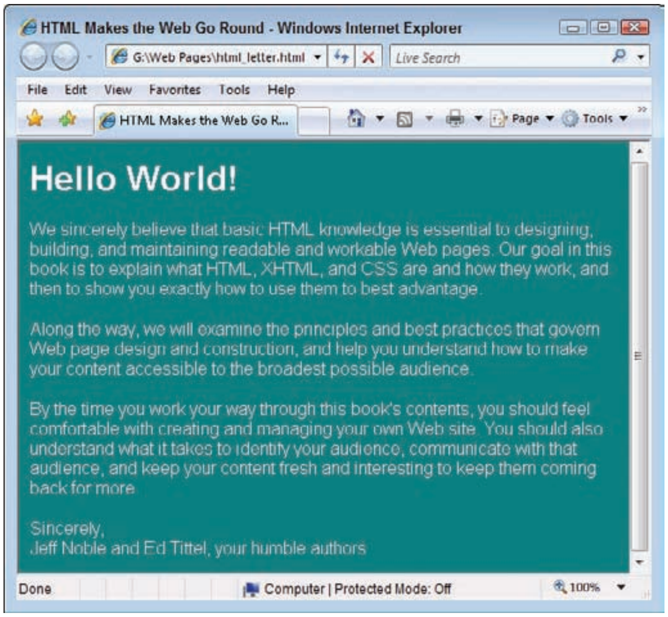
Name	Date modified	Type	Size
html_letter.html	11/16/2007 11:05 AM	HTML Document	2 KB

Figure 2-4: Use Internet Explorer to navigate to your Web pages.

5. **Click the Open button,**

   You are brought back to the Open dialog box. (***Note:*** Newer versions of IE will warn you they must open a new browser window for your local file, for security reasons, if you're already connected to the Internet; this is perfectly OK.)
6. **Click OK.**

   The page appears in your Web browser in all its glory, as shown in Figure 2-5.


HTML Makes the Web Go Round - Windows Internet Explorer
G:\Web Pages\html_letter.html
Live Search
File Edit View Favorites Tools Help
HTML Makes the Web Go R...
Page Tools
Done
Computer | Protected Mode: Off
100%

Hello World!

We sincerely believe that basic HTML knowledge is essential to designing, building, and maintaining readable and workable Web pages. Our goal in this book is to explain what HTML, XHTML, and CSS are and how they work, and then to show you exactly how to use them to best advantage.

Along the way, we will examine the principles and best practices that govern Web page design and construction, and help you understand how to make your content accessible to the broadest possible audience.

By the time you work your way through this book's contents, you should feel comfortable with creating and managing your own Web site. You should also understand what it takes to identify your audience, communicate with that audience, and keep your content fresh and interesting to keep them coming back for more.

Sincerely,
Jeff Noble and Ed Tittel, your humble authors

Figure 2-5: Viewing a local file in your Web browser.

REMEMBER

You aren't actually viewing this file on the Web just yet; you're just viewing a copy of it saved on your local hard drive. So don't give anyone the URL for this file yet — but do feel free to edit the HTML source file and view any changes you make.

TIP

An even faster way to view a Web page locally in a browser is to drag and drop the HTML file into an open browser window. You can do this from Windows Explorer or any other program that gives you file-level access.

Editing an Existing Web Page

Chances are you'll want to change one thing (at least) about your page after you view it in a Web browser for the first time. After all, you can't really see how the page is going to look when you're creating the markup, and you might decide that a first-level heading is too big or that you really *want* purple text on a green background (horrible idea, actually).

To make changes to the Web page you've created in a text editor and are viewing in a browser, repeat these steps until you're happy with the final appearance of your page:

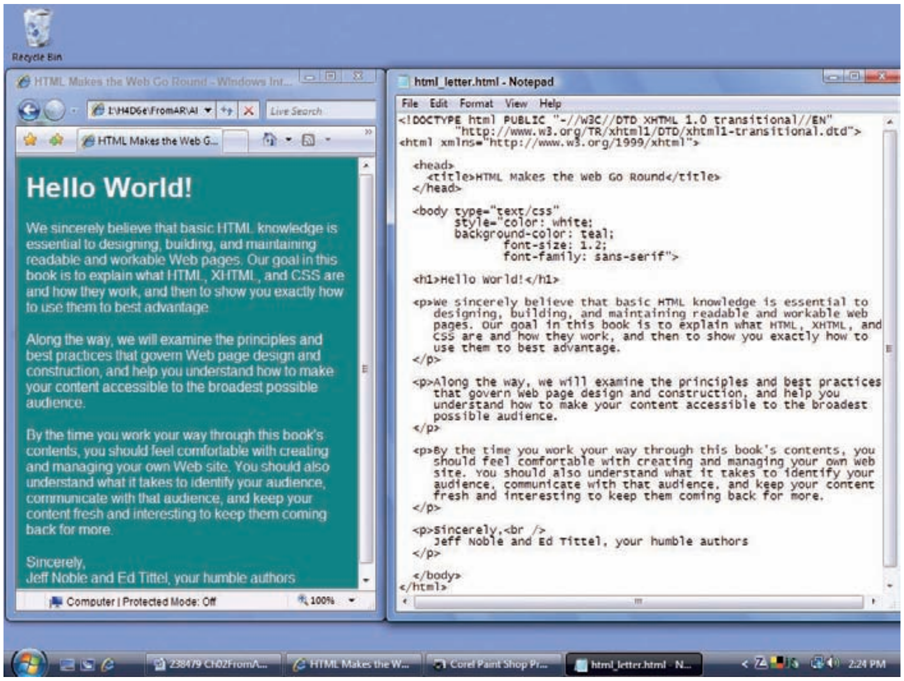
1. **Leave the browser window with the HTML page display open, and go back to the text editor.**
2. **If the HTML page isn't open in the text editor, open it.**

   You should have the same file open in both the browser and the text editor, as shown in Figure 2-6.
3. **Make your changes to the HTML and its content in the text editor.**
4. **Save the changes.**

   This is an important step. If you don't save your changes, you won't see them in the Web browser.
5. **Move back to the Web browser and click the Refresh button.**

If you keep the HTML file open in both the text editor and the browser while you work, checking changes is a breeze. You can quickly save a change in the editor, flip to the browser and refresh, flip back to the editor to make more changes, flip back to the browser and refresh, and so on.

In our example letter, we decided — after our initial draft of the HTML page — that we should add a date to the letter. Figure 2-7 shows the change we made to the HTML to add the date, and the resulting display in the Web browser.



Hello World!

We sincerely believe that basic HTML knowledge is essential to designing, building, and maintaining readable and workable Web pages. Our goal in this book is to explain what HTML, XHTML, and CSS are and how they work, and then to show you exactly how to use them to best advantage.

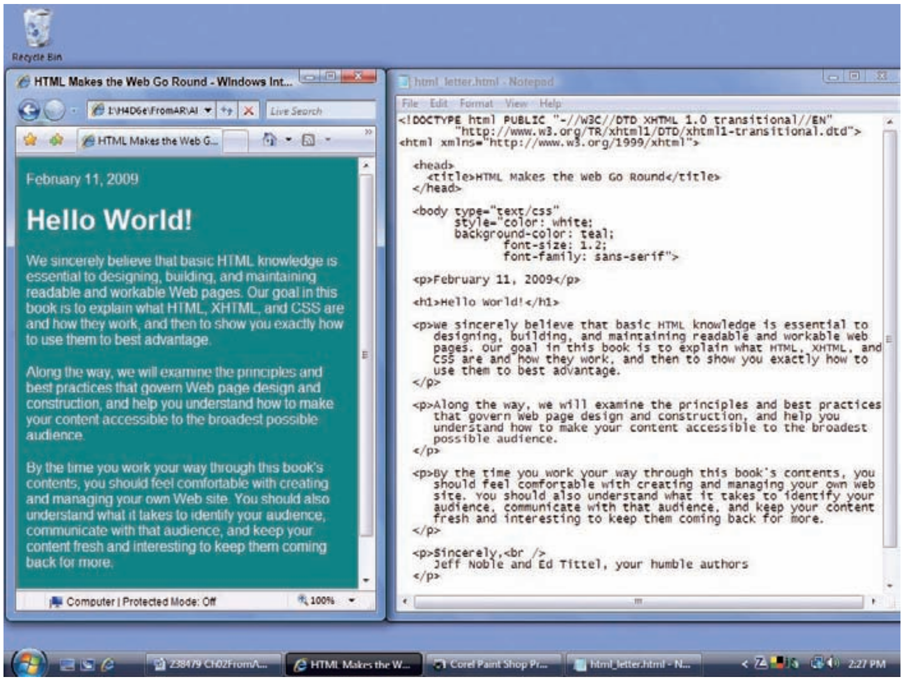
Along the way, we will examine the principles and best practices that govern Web page design and construction, and help you understand how to make your content accessible to the broadest possible audience.

By the time you work your way through this book's contents, you should feel comfortable with creating and managing your own Web site. You should also understand what it takes to identify your audience, communicate with that audience, and keep your content fresh and interesting to keep them coming back for more.

Sincerely,
Jeff Noble and Ed Tittel, your humble authors

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

  <head>
    <title>HTML Makes the Web Go Round</title>
  </head>

  <body type="text/css"
        style="color: white;
        background-color: teal;
               font-size: 1.2;
               font-family: sans-serif">

  <h1>Hello World!</h1>

  <p>We sincerely believe that basic HTML knowledge is essential to
     designing, building, and maintaining readable and workable web
     pages. Our goal in this book is to explain what HTML, XHTML, and
     CSS are and how they work, and then to show you exactly how to
     use them to best advantage.
  </p>

  <p>Along the way, we will examine the principles and best practices
     that govern web page design and construction, and help you
     understand how to make your content accessible to the broadest
     possible audience.
  </p>

  <p>By the time you work your way through this book's contents, you
     should feel comfortable with creating and managing your own web
     site. You should also understand what it takes to identify your
     audience, communicate with that audience, and keep your content
     fresh and interesting to keep them coming back for more.
  </p>

  <p>Sincerely,<br />
     Jeff Noble and Ed Tittel, your humble authors
  </p>

  </body>
</html>
```

Figure 2-6: Viewing an HTML file in your text editor and Web browser at the same time.



February 11, 2009

Hello World!

We sincerely believe that basic HTML knowledge is essential to designing, building, and maintaining readable and workable Web pages. Our goal in this book is to explain what HTML, XHTML, and CSS are and how they work, and then to show you exactly how to use them to best advantage.

Along the way, we will examine the principles and best practices that govern Web page design and construction, and help you understand how to make your content accessible to the broadest possible audience.

By the time you work your way through this book's contents, you should feel comfortable with creating and managing your own Web site. You should also understand what it takes to identify your audience, communicate with that audience, and keep your content fresh and interesting to keep them coming back for more.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

  <head>
    <title>HTML Makes the Web Go Round</title>
  </head>

  <body type="text/css"
        style="color: white;
        background-color: teal;
                font-size: 1.2;
                font-family: sans-serif">

  <p>February 11, 2009</p>

  <h1>Hello World!</h1>

  <p>We sincerely believe that basic HTML knowledge is essential to
     designing, building, and maintaining readable and workable web
     pages. Our goal in this book is to explain what HTML, XHTML, and
     CSS are and how they work, and then to show you exactly how to
     use them to best advantage.
  </p>

  <p>Along the way, we will examine the principles and best practices
     that govern web page design and construction, and help you
     understand how to make your content accessible to the broadest
     possible audience.
  </p>

  <p>By the time you work your way through this book's contents, you
     should feel comfortable with creating and managing your own web
     site. You should also understand what it takes to identify your
     audience, communicate with that audience, and keep your content
     fresh and interesting to keep them coming back for more.
  </p>

  <p>Sincerely,<br />
     Jeff Noble and Ed Tittel, your humble authors
  </p>
```

Figure 2-7: A change in the HTML is displayed in a browser after a quick save and refresh.

WARNING!

This approach to editing an HTML page applies only to pages saved on your local hard drive. If you want to edit a page that you've already stored on a Web server, you have to save a copy of the page to your hard drive, edit it, verify your changes, and then upload the file again to the server, as discussed in the following section.

Posting Your Page Online

After you're happy with your Web page, it's time to put it online. Chapter 3 includes a detailed discussion of what you need to do to put your page online, but to sum it up in a few quick steps:

1. **Find a Web hosting provider to hold your Web pages.**

   Your Web host might be a company Web server or space that you pay an Internet service provider (ISP) for. If you don't have a host yet, double-check with the ISP you use for Internet access — find out whether you get some Web-server space along with your access. Regardless of where you find space, get details from the provider on where to move your site's files and what your URL will be.

2. **Use an FTP client or a Web browser to make a connection to your Web server.**

   Use the username and password, as specified in the information from your hosting provider, to open an FTP session on the Web server.

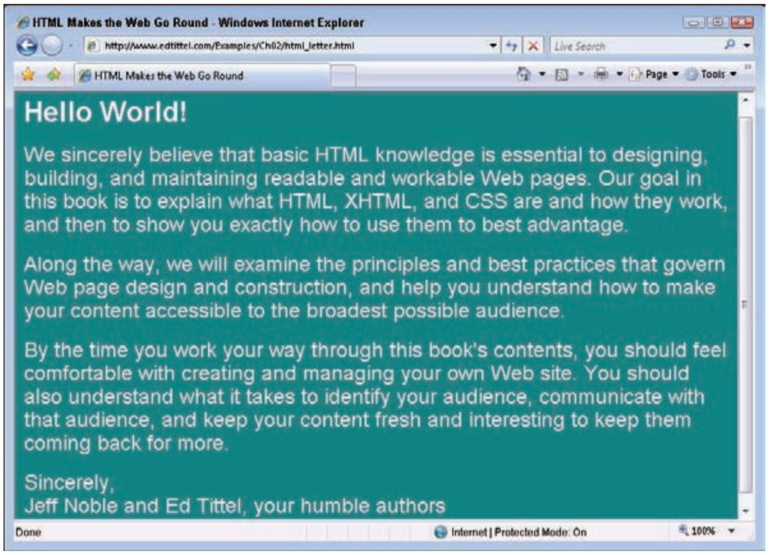
3. **Copy the HTML file from your hard drive to the Web server.**
4. **Use your Web browser to view the file via the Internet.**

For example, to host our letter online at `www.edtittel.com/examples/ch02`, we used Internet Explorer to access the site and provided the appropriate name and password, which you get from your ISP. A collection of folders and files appeared.

We copied the file to the server with a simple drag-and-drop operation from Windows Explorer to Internet Explorer.

The URL for this page is `http://www.edtittel.com/examples/ch02/html_letter.html`, and the page is now served from the Web browser instead of from a local file system, as shown in Figure 2-8.

Chapter 3 has details on how to serve your Web pages to the world.


HTML Makes the Web Go Round - Windows Internet Explorer
http://www.edtittel.com/Examples/Ch02/html_letter.html
Live Search
HTML Makes the Web Go Round
Page
Tools
Done
Internet | Protected Mode: On
100%

Hello World!

We sincerely believe that basic HTML knowledge is essential to designing, building, and maintaining readable and workable Web pages. Our goal in this book is to explain what HTML, XHTML, and CSS are and how they work, and then to show you exactly how to use them to best advantage.

Along the way, we will examine the principles and best practices that govern Web page design and construction, and help you understand how to make your content accessible to the broadest possible audience.

By the time you work your way through this book's contents, you should feel comfortable with creating and managing your own Web site. You should also understand what it takes to identify your audience, communicate with that audience, and keep your content fresh and interesting to keep them coming back for more.

Sincerely,
Jeff Noble and Ed Tittel, your humble authors

Figure 2-8: A file on a Web server is available to anyone with an Internet connection.

Chapter 3

Proper Planning Prevents Poor Page Performance

In This Chapter

- Planning your Web page
- Defining your Web site hierarchy
- Creating user-friendly navigation
- Hosting your site
- Uploading and editing your Web site

The overall design of your site is its *user interface* (UI). When you design a good UI, you give users tools to move through your site with minimum fuss. This chapter outlines standard Web-site design principles for your (X)HTML and CSS. These principles can ensure a usable and effective UI.

The UI is the mechanism that gives a user access to the information on your Web site. Each UI is unique, but they're all made from the same components (*text, graphics,* and *media files*), and they're all held together with (X)HTML.

WARNING!

Visitors probably won't return to your site if

- It's hard to navigate
- It's cluttered with flashing text and clashing colors
- It doesn't help people find what they're looking for

You've created a solid UI if

- Your site's navigation is intuitive.
- Images and media accent your design without overpowering it.
- You do all you can to help people find the information they want.

This chapter walks you through simple steps to design a Web site and your basic Web page. (Other chapters explain every nuance of the markup.)

Planning Your Site

An important first step in creating an effective UI for a site has nothing to do with markup, but has everything to do with planning. Before your site grows too large (or before you even build your site if you haven't yet started), scope out your site's exact purpose and goals. When you know your site's scope and goals, you can better create an interface to embody them.

Before designing your site, ask yourself these questions:

- Why are you creating this site?
- What do you want to convey to users?
- Who is your target audience? For example
    - What's the average age of your users?
    - How well does your audience work with the Internet?
- How many pages do you need in your site?
- What type of hierarchy will you use to organize your pages? For example, you can create your site so users go through it linearly, or you can allow them to jump around from topic to topic.

If you can answer these questions, you can better understand your site's goals and needs. For example, an online store might have these goals:

- Let visitors browse an online catalog and put items in a shopping cart.
- Provide visitors a way to purchase the items in their cart online.
- Help users make smart purchasing decisions.
- Ease merchandise returns and exchanges.
- Solicit feedback from users about products they want to see in the catalog or ways to make the site better.

Stating clear goals will help you get a better sense of what you must do on your Web site to fulfill these goals. To do the things an online store does, for example, your site is going to need the following:

- An online catalog, complete with shopping cart
- Buying guides or other information that can help users make better purchasing decisions

- A help-and-feedback section, perhaps with message forums to let users and experts interact
- A set of tools to expedite returns and exchanges

When you establish goals for your site, you can identify those elements best suited for inclusion, such as

- A navigation system that identifies the major areas of the site, to help users
  - Quickly identify what part they're in
  - Move from one part of the site to others without getting lost
- A set of standard design elements, such as buttons, page-title styles, and color specifications, to keep users oriented as they move from page to page in the same site
- A standard display for catalog items, including product-related information, such as product images and descriptions, prices, and availability data
- Well-designed forms to help users find products in the catalog, purchase items to put in their shopping carts, request a refund or help returning an item, and submit comments to the site
- Long text pages that offer extensive information on purchasing options, product returns, and other helpful information — but are still easy to read and to navigate

Your site's goals should dictate your site's

- **UI elements**

  When you add to an existing site, identify UI elements that
  - Meet the goals of the new section of the site
  - Complement the overall site UI design
- **Design**
- **Organization**

TRICKS OF THE TRADE

Design matters

This chapter recommends good design principles, but it's up to you to choose color schemes and the overall look and feel. What looks great to one person may be ugly to someone else.

If you're building a site for your business, that site can provide a first impression for potential customers or clients. The site should reflect your business style. If you run an architecture firm, for example, strong lines and a clean look may be the best way to present your company image. If you run a flower shop, your site may be a bit more organic and decorated (okay, *flowery*) to remind visitors what to expect if they walk into your store.

If you're new to Web design or graphics and you need a site that marks your business presence on the Web, consider getting help from a Web-design professional. Use the images, layouts, and navigational aids they create to build and manage the site yourself. Once established, a distinctive and consistent look and feel for your site is easy to maintain.

Regardless of who designs your site, take the time to get a critique from peers, friends, family members, and anyone else who is willing to be honest about how good (and even how bad) it looks. A negative-but-constructive critique from someone who knows and respects you beats a "Gee, that's ugly" from someone whose business you are trying to acquire.

Mapping your site

It's easier to get where you're going if you know how to get there. Mapping your Web site can be a vital step in planning — and later running — that site. This process involves two creative phases:

- **Creating a visual guide on paper or electronically that you can use to guide the development of your site**
- **Creating a visual guide on your Web site to help visitors find their way around once it's built**

Both have a place in good UI design, so each gets its own section.

Using a map for site development

When you use a site map during the development of a Web site — even a Web site that includes only a few pages — you can identify

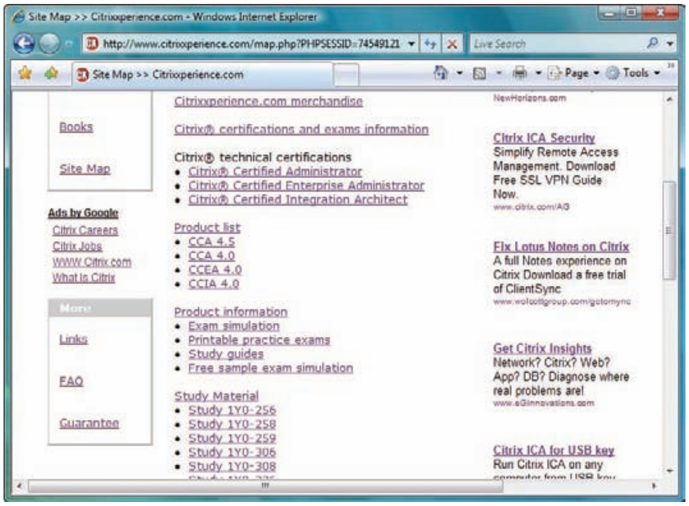
- Pages that you need to build
- How pages relate to each other
- Navigation elements that you need

TIP

As a bonus, a map provides you with a *checklist* of pages.

For example, Figure 3-1 shows part of the site map for the Citrixxperience Web site (`www.citrixxperience.com/map.php`).

This map shows that the site has several main sections. Three of those sections — product list, product information, and study materials — are each further divided into subsections. Each subsection page offers links that are pertinent to that particular subsection.


Site Map >> Citrixxperience.com - Windows Internet Explorer
http://www.citrixxperience.com/map.php?PHPSESSID=74549121
Live Search
Site Map >> Citrixxperience.com
Page
Tools

Books

Site Map

Ads by Google
Citrix Careers
Citrix Jobs
WWW Citrix com
What Is Citrix

More

Links

FAQ

Guarantee

Citrixxperience.com merchandise

Citrix® certifications and exams information

Citrix® technical certifications
- Citrix® Certified Administrator
- Citrix® Certified Enterprise Administrator
- Citrix® Certified Integration Architect

Product list
- CCA 4.5
- CCA 4.0
- CCEA 4.0
- CCIA 4.0

Product information
- Exam simulation
- Printable practice exams
- Study guides
- Free sample exam simulation

Study Material
- Study 1Y0-256
- Study 1Y0-258
- Study 1Y0-259
- Study 1Y0-306
- Study 1Y0-308

NewHorizons.com

Citrix ICA Security
Simplify Remote Access Management. Download Free SSL VPN Guide Now.
www.citrix.com/AG

Fix Lotus Notes on Citrix
A full Notes experience on Citrix Download a free trial of ClientSync
www.wolcottgroup.com/gotomync

Get Citrix Insights
Network? Citrix? Web? App? DB? Diagnose where real problems are!
www.eGInnovations.com

Citrix ICA for USB key
Run Citrix ICA on any

Figure 3-1: The site map for the Citrixxperience Web site.

WARNING!

Don't create *under construction* sections that don't include much of anything except the hint that something will appear someday. Users are disappointed if your site merely hints at information it doesn't really offer. Instead, consider using a small section of your home page to highlight "coming soon" items so visitors know new information will be available later on, but don't integrate anything that's not yet accessible into your navigation bar or buttons.

Using a map as a visual guide for your users

A *site map* can be a supplemental navigational tool that gives users a different way to find what they're looking for. A site map lays out all contents of your site so visitors can see all their options at once.

REMEMBER

People have many approaches to finding information. Give visitors as many options as you can (realistically) to help them navigate within your site:

If you build your site one piece at a time . . .

If you plan to build your Web site a page or section at a time, you can create a map of the final site and then decide which pages make the most sense to build first. When you have a good working idea of how your site will expand, you can plan for further expansion during each stage. For example, suppose you create a site map for your company's Web site, and the site needs a FAQ. If that section isn't quite finished when the site launches, disaster need not ensue — provided someone planned ahead to accommodate new sections and built that capability into the site. Just leave out links to (and mentions of) the FAQ when you launch the site.

When the FAQ section is ready, you

- Add the section to the site.
- Add a link to the main navigation elements.

If you know the resources are coming, you can create a navigation scheme that easily accommodates the FAQ when it's ready to go. Without a site map and a complete plan for the site, however, integrating new sections can suck up lots of time and effort.

- Some people like to be led.
- Some people like to rummage around.
- Some people like to see every possible option and choose one.

WARNING!

Site maps grow as your site grows. If your site is large and complex, your map may take several screens to display. When you surf the Web, massive sites such as `Microsoft.com`, `HP.com`, and `Amazon.com` don't offer site maps because maps of their sites would be huge and unwieldy. But smaller Web sites (such as `Symantec.com`) use site maps effectively.

TRICKS OF THE TRADE

You need to decide whether a site map is a good navigation tool for your site. Here are some points to ponder as you make this decision:

- A site map may be *unnecessary* if you have only a few pages.
- A site map may be the *best* choice if
  - Your site has several sections.
  - You can't think of other ways to access your content.

Many experts believe that sitemap pages are always good. They're especially good for visitors who surf the Web using assistive devices (screen readers, Braille printers, and so forth). They're also handy for navigating a site that lacks footer links or that uses graphics instead of HTML markup as its primary navigation technique. It also helps users who have turned their browser's JavaScript function off find their way around (sites that use rollover images for navigation become unusable in that case). As an added bonus, sitemaps also help search engines map out all the pages on a site.

Building solid navigation

The navigation you use on your site can make it or break it. If visitors can't find what they're looking for on your site, they'll probably leave and never come back. The type of navigation you use on your site depends on

- **How many pages are on your site**

  If you have only a few pages, your navigation might be a simple list of links on the home page to help users jump to each of the other pages.

- **How you organize your pages**

  If your site has many pages organized into different sections, your home page might link only to those sections (not to each page).

For example, the `Dummies.com` site houses a large collection of pages organized as a variety of sections; it would be impractical to link to all the pages in any navigation scheme. Also, the site includes articles on a wide variety of topics, as well as book information. The site could be organized into books and articles, but visitors are more likely to look for information on a specific subject, so the site is organized by topic. The home page, shown in Figure 3-2, prominently displays these different topic areas on the left.

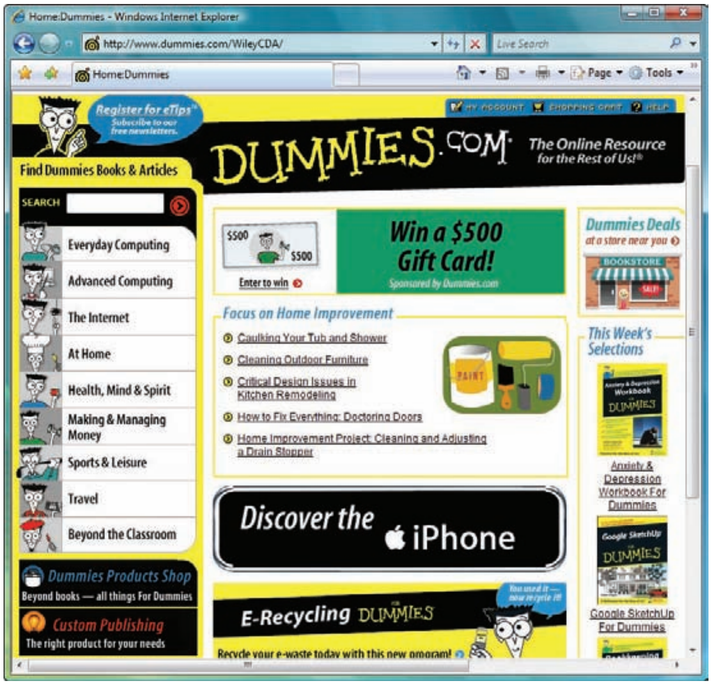
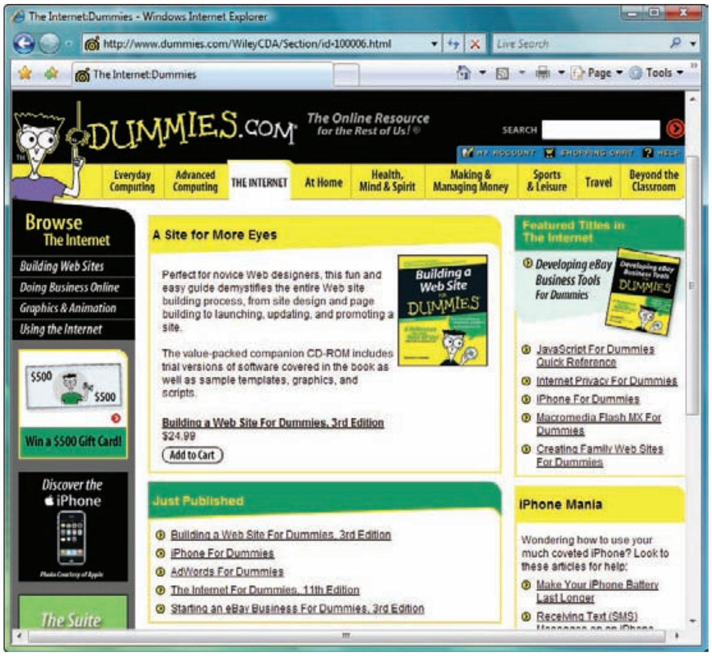

Home:Dummies - Windows Internet Explorer
http://www.dummies.com/WileyCDA/
Live Search
Home:Dummies
Page
Tools
Register for eTips
Subscribe to our free newsletters.
MY ACCOUNT
SHOPPING CART
HELP
DUMMIES.COM
The Online Resource for the Rest of Us!
Find Dummies Books & Articles
SEARCH
Everyday Computing
Advanced Computing
The Internet
At Home
Health, Mind & Spirit
Making & Managing Money
Sports & Leisure
Travel
Beyond the Classroom
Dummies Products Shop
Beyond books — all things For Dummies
Custom Publishing
The right product for your needs
$500
Enter to win
Win a $500 Gift Card!
Sponsored by Dummies.com
Focus on Home Improvement
Caulking Your Tub and Shower
Cleaning Outdoor Furniture
Critical Design Issues in Kitchen Remodeling
How to Fix Everything: Doctoring Doors
Home Improvement Project: Cleaning and Adjusting a Drain Stopper
PAINT
Discover the iPhone
E-Recycling DUMMIES
You used it — now recycle it!
Recycle your e-waste today with this new program!
Dummies Deals
at a store near you
BOOKSTORE
SALE
This Week's Selections
Anxiety & Depression Workbook For Dummies
Google SketchUp For Dummies

Figure 3-2: The Dummies.com site is organized by topic.

When you click one of these topic areas, the remaining topic areas stay available in a navigation bar across the top of the page (as shown in Figure 3-3). You don't have to return to the home page to jump from topic to topic.

Figure 3-3 shows that each topic has its own sub-navigation area (at left, echoing the layout of the home page) that lists subtopics within the topic. The links are different, but the general navigation scheme is consistent throughout the site. That tells visitors what to expect as they move around the site.

The topmost navigation area of each page includes a regular collection of links that appears on every page of the site to help visitors quickly access important areas from anywhere: a search box, account information, a shopping cart, and help. At the bottom, every page has the same set of links to information on the *For Dummies* Web site, a form to register for eTips, a sign up for RSS feeds, book registration, a contact link, the site copyright statement, and the site privacy policy. (Sorry, you'll have to look at that for yourself; I couldn't fit the bottom part of the page into Figure 3-3.) Like the shopping cart and help links, these links have to be on every page, but they need not be displayed prominently. Adding them to a consistent site footer keeps them accessible to visitors without obscuring key content for any given topic or subtopic.


The Internet:Dummies - Windows Internet Explorer

http://www.dummies.com/WileyCDA/Section/id-100006.html

The Internet:Dummies

DUMMIES.COM The Online Resource for the Rest of Us!

SEARCH

MY ACCOUNT SHOPPING CART HELP

Everyday Computing | Advanced Computing | THE INTERNET | At Home | Health, Mind & Spirit | Making & Managing Money | Sports & Leisure | Travel | Beyond the Classroom

Browse The Internet

- Building Web Sites
- Doing Business Online
- Graphics & Animation
- Using the Internet

Win a $500 Gift Card!

Discover the iPhone

The Suite

A Site for More Eyes

Perfect for novice Web designers, this fun and easy guide demystifies the entire Web site building process, from site design and page building to launching, updating, and promoting a site.

The value-packed companion CD-ROM includes trial versions of software covered in the book as well as sample templates, graphics, and scripts.

Building a Web Site For Dummies, 3rd Edition
$24.99
Add to Cart

Just Published

- Building a Web Site For Dummies, 3rd Edition
- iPhone For Dummies
- AdWords For Dummies
- The Internet For Dummies, 11th Edition
- Starting an eBay Business For Dummies, 3rd Edition

Featured Titles in The Internet

- Developing eBay Business Tools For Dummies
- JavaScript For Dummies Quick Reference
- Internet Privacy For Dummies
- iPhone For Dummies
- Macromedia Flash MX For Dummies
- Creating Family Web Sites For Dummies

iPhone Mania

Wondering how to use your much coveted iPhone? Look to these articles for help:

- Make Your iPhone Battery Last Longer
- Receiving Text (SMS) Messages on an iPhone

Figure 3-3: The main topic areas on this site are accessible from the top navigation bar.

If you create a map to aid site development, it can also help you choose the navigational tools to create for your site. Consider each page on the map in turn; list the links that each page has to include. Normally a pattern emerges that can help you identify the main navigation elements your site needs (such as links to all main topic areas and copyright information, as on the *For Dummies* site), as well as sub-navigation tools (such as links to subtopics on the topic pages).

After you know what tools you need, you can begin to design a visual scheme for your UI. Do you want to use buttons across the top, buttons down the side, or both? Do you need a footer that links to copyright or privacy information? If you have sections within sections within sections, how can you best help people navigate through them? Answering questions like these is the route to a solid navigation system that helps users find their way around your site — letting them focus on what they came for, and not on how to get there.

Whatever navigation scheme you devise, always give your visitors a way to get back to your home page from wherever they are on the site. Your site's home page is the gateway to the rest of the site. If visitors get lost or want to start again, make sure they can get back to Square One with no trouble.

After you design a site navigation scheme and put together a few pages, ask someone who isn't familiar with your site to use it. To help them along, give them a list of three or four tasks you'd like them to complete — pages to visit or a form to fill out, for example. If your test visitor gets lost or has lots of questions about how to navigate, you should rework your scheme. Your reviewer might also have suggestions on ways to make navigation features clearer and easier to use. You might know your site and its content *too well* to find navigation issues that a first-time user will discover immediately.

Planning outside links

The Web wouldn't be the Web without hyperlinks — after all, hyperlinks connect your site to the rest of the Web, and turn a collection of pages into a cohesive site. But overusing or misusing links can detract from your site — and even cost you some business.

Choose off-site links wisely

Internal linking is almost a walk in the park compared to external linking — after all, when you link to pages on your own site, the pages those links point to are *under your control.* You know what's on them today and what will be on them tomorrow, and even whether they will exist tomorrow. When you link to resources on someone else's site, however, all bets are off:

- You don't maintain those pages.
- You can't modify their content.
- You certainly won't know when they will disappear.

  Neither will your visitors — until they slam into a `404 File or directory not found` message (the usual sign of a *broken link* that now goes nowhere). The text in 404 messages varies depending on the server that hosts the Web site with the broken link.

Links to other sites are more useful when they're stable and have less chance of breaking. We recommend these guidelines:

- **Link to a *section* of a site, not to a specific page.**

  Pages come and go, but the general organization usually stays the same.
- **Link to corporate Web sites.**

  Corporate sites have more staying power than sites maintained by an individual.
- **Don't link directly to media files such as PDFs and images.**

  If you want to link to resources on another Web site, link to the Web page that links to the resources instead of the actual media files. Sites often update resources or give them new names. The page that links to the resource, however, is almost always certain to be updated to reflect new names. Therefore the resource page is a safer linking bet.

Linking to other sites *implies your support or endorsement of those sites.* When visitors follow links from your site to other sites, they assume you approve of that new site. That makes a couple of guidelines necessary:

- **If you don't want to be associated with content on another site, don't link to the site.**

  The only way to find out whether you approve of a seemingly relevant site is to visit it and check it out *before* you link.
- **Periodically review your links.** Be sure that
  - The sites' owners are the same.
  - The content is appropriate.

  When domain names expire, new owners may take them over and post new content that's either

  - Completely irrelevant
  - Damaging to your image, as with pornography

Craft useful link text

The text you associate with links is as important as the links you use on your site. That text gives users a hint about where the link takes them so they can decide whether to go along for the ride. For example, `Visit Dummies.com to read more about this book` is more helpful than `Read more about this book`.

The first example tells visitors that they're going to leave the current site to visit `Dummies.com` and read more about a book there. The second just tells them they're going to read more about the book — and they may be surprised to find themselves flung off one site and onto another.

Generally, when you create link text, let users know the following:

- **Whether they're leaving your site or not**
- **What kind of information the page they're linking to contains**
- **How the linked site relates to the current content or page**

The goal of your link text should be to inform users and build their trust. If your link text doesn't give them solid clues about what to expect from your links, they just won't trust your links — and won't follow them.

TRICKS OF THE TRADE

Avoid the use of *click here* in any link you create. If your link text is well-crafted, you don't need the extra words to prompt the user to click a link. Link text should speak for itself: let it *invite* a click, rather than *demanding* one.

Hosting Your Web Site

The first (and most important) step in putting your pages online is finding someplace on the Web to put them on display — a host. In general, you have two choices for hosting your pages:

- Host them yourself.
- Pay someone else to host them.

TIP

The word *host* is used in the Web industry to mean a Web server set up to hold Web pages (and related files) so they can be accessed by the rest of the world. This chapter uses *host* as both

- **Noun:** The host is the physical machine that holds the Web pages
- **Verb:** Hosting is the act of serving up the Web pages

You have to decide whether to host your own pages or to pay someone else to host them for you. This chapter describes both approaches — and gives you the skinny on each one. You can decide which option is best for you.

You aren't stuck with a hosting decision for life. If you find hosting your own pages overwhelming, you can move your files to a service provider (or vice versa, if the provider's service is underwhelming). To decide which hosting option is best for you, consider your needs for the next year, but plan to review your needs in no more than six months.

Hosting your own Web site

This section illustrates an average-size site (up to about 100 pages) that doesn't include more than a couple of multimedia files and doesn't have any special security or electronic commerce (e-commerce) applications.

If you need to run a complex site, such as a large corporate site or an online store, you need more expertise, equipment, and software than this section outlines. The following resources can help:

- Books such as *E-Commerce For Dummies* and *Webmastering For Dummies, 2nd Edition* (both from Wiley Publishing) can get you started setting up e-commerce and other complex sites.
- Consult a Web professional who has practical experience building and maintaining complex Web sites.

You can set up your own Web server and host your Web pages yourself. To do this, you need:

- **A computer designated as your Web server:** Web servers are often *dedicated* to this task, leaving word-processing and other activities to a different computer.
- **Web-server software:** Common Web-server software packages include Apache and Microsoft Internet Information Server (IIS), called Internet Information Services in Windows 2000 and later.

  In the Web world, the term *Web server* refers to both

  - A dedicated computer (the actual hardware)
  - Web-server software

  You can't use one without the other.
- **A dedicated Internet connection:** Your Web server isn't useful or reliable if it connects to the Internet only when you fire up a dialup link.

If hosting a Web site yourself sounds complicated and expensive, you're right. Not only do you have to pay for the equipment and an Internet connection, but you also have to know how to set up and administer a Web server and keep all its pieces working 24/7. Whenever possible, consider a hosting provider first.

Using a hosting provider

A *hosting provider* manages all the technical aspects of Web hosting, from hardware to software to Internet connections. You just manage your Web pages. Back when the Web was young, hosting-provider options were scarce, and what *was* available was expensive. Times have changed — and needs have grown — so reasonably priced hosting providers are abundant nowadays.

If you pay someone else to host your pages, two choices cover all costs:

- **Nothing:** Some services actually host your pages for free. That's it; you pay zip, zero, nada to put pages on the Web. What's the catch? You have to "pay" in other ways, usually by letting advertising appear on your site.
- **Something:** Most Web-hosting services, however, do charge a fee, from a few dollars to triple digits per month. The trick to making the most of your hosting budget is to find just the right service to meet your needs.

TIP

Read more about inexpensive Web hosting options at:

```
http://www-thehostingchart.com
```

Obtaining your own domain

A *domain name* is the high-level address for any given Web site. Examples of domain names are `microsoft.com`, `apple.com`, `w3c.org`, and `dummies.com`.

You might want your own domain name (hence your own domain) to reflect your business name (or even your personality). If you don't get a domain name of your own, your pages will be part of someone else's domain name — usually your hosting provider's. For example, a hypothetical personal Web site hosted without a domain name at `io.com` would use this URL

```
http://www.io.com/~edtittel
```

With a domain name of `edtittel.com`, the same Web site would be hosted at

```
http://www.edtittel.com
```

One's easier to remember than the other. Is that a good enough reason to have your own domain? Maybe . . . maybe not. The bottom line is that businesses or other entities that want to maintain a constant Web presence should probably invest in a domain name; hobbyists or enthusiasts don't need one.

TIP

Any good hosting provider can give you detailed instructions on how to register a domain name in the provider's system or attach your domain name to your Web site on its computers. If you're changing over from one hosting provider to another, your new provider should help you transfer your domain. Most providers either give you this information up front or have online help to walk you through the process. If it isn't immediately clear how to set up your domain, ask for help. If you don't get it, change providers.

Moving files to your Web server

After you secure a Web site host or decide to put up your own Web server, you need a way to move the Web pages you create on your local computer to the Web server. This isn't a one-time activity, either. As you maintain your Web site, you need to move files you've built on your local computer to the Web server to refresh your site.

How you move files to your Web server depends entirely on how your Web server is set up. Normally, you have a couple of transfer options:

- **The File Transfer Protocol (FTP)**
- **A Web interface, provided by your hosting provider, for moving and managing files**

Via FTP

Of these two options, FTP is almost always a possibility. FTP is a standard way of transferring files on the Internet, and any hosting provider should offer FTP access to your Web server. When you set up your site with your provider, it usually gives you written documentation (either on paper or on the Web) to tell you exactly how to transfer files to your Web server. Included in that information will be an FTP URL that usually takes the form `ftp://ftp.`*`domain`*`.com`.

You can use an FTP client such as SmartFTP (`www.smartftp.com`), WS_FTP (`www.ipswitch.com/Products/WS_FTP/`) or CuteFTP (`www.globalscape.com/cuteftp/`) to open a connection to this URL (Macintosh users will probably prefer Fetch at `www.fetchsoftworks.com` or Cyberduck at `cyberduck.ch`). Your provider will give you a username and password to use to access your Web-server directory on the FTP site. Then you can move files to your Web site using the client interface. It's really that easy. If you want to grab a copy of a file from your Web site and modify it, you can do that in three steps:

1. **Use the FTP client's interface to download a copy.**
2. **Make your modification.**
3. **Use the FTP client's interface to upload the file.**

TIP

Each FTP client's interface is different, but they're all pretty straightforward. Chapter 22 includes more information on finding a good FTP client; so when you find one, spend a few minutes reading its documentation.

TIP

You might not need FTP client software to move files to your Web server:

- Many browsers, such as current versions of Internet Explorer and Firefox, include basic, built-in FTP support. You can upload or download files, but you may be unable to create or delete directories.
- Many Web utilities, such as Dreamweaver, include file-management capabilities.

Via your hosting provider's Web site

To enhance usability and reduce technical support calls, many Web hosting providers offer Web pages to help you upload and manage your Web-site files without using a separate FTP utility or even the FTP tools inside (X)HTML editors. Most of these tools let you manage your site in various ways, such as

- Uploading and downloading files
- Creating and deleting directories
- Moving files around
- Deleting files

If you work with a hosting provider, find out whether it has a set of Web-based tools for managing your site.

Keep these thoughts in mind while you decide on a provider:

- Read the provider's documentation before you start to transfer your files. Every provider's interface is different.
- Most providers who offer Web interfaces won't stop you from managing your site with FTP.

TIP

Use FTP if the provider's interface is cumbersome or if you prefer FTP.

UI design resources

We recommend these Web sites and books on site and interface design if you want to create great UIs:

- For a crash course on Web design basics, read the "Basics" and "Design Process" sections in "Your Complete Guide to Web Design" at

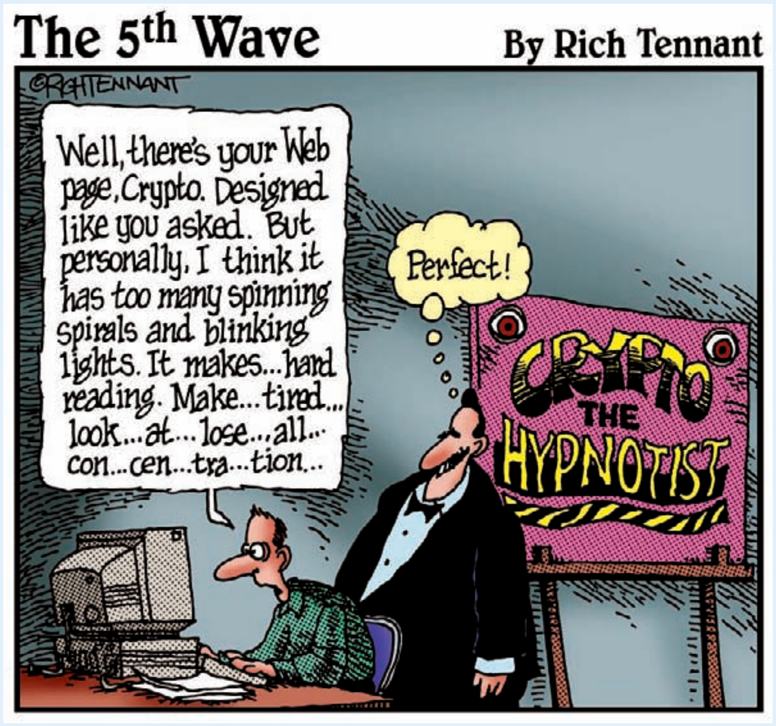
  `http://www.webdesignfromscratch.com/`

  Webmonkey's "Site Redesign Tutorial" offers an interesting perspective on what it takes to rework a site's design. Read it at

  `http://hotwired.lycos.com/webmonkey/design/site_building/tutorials/tutorial4.html`

- Jakob Nielsen is committed to creating accessible Web content, which means that all content is available to all visitors, including those with various handicaps that might prevent them from following visual or audible cues for navigation. His Web site, `http://useit.com`, is chock-full of resources and articles on creating accessible sites.
- Hey, negative examples are useful too. Web Pages That Suck guides you to good design by evaluating bad design. Be sure your site doesn't look like any of those featured at `www.webpagesthatsuck.com`.
- *Web Design For Dummies,* by Lisa Lopuck (Wiley), is another step in the direction of a sophisticated Web site with a knockout look.
- *Web Usability For Dummies,* by Richard Mander and Bud Smith (Wiley), can help you fine-tune your site to make it amazingly easy to use, which is a great help in keeping your visitors coming back for more.

Part II
Formatting Web Pages with (X)HTML


The 5th Wave
By Rich Tennant
©RICHTENNANT
Well, there's your Web page, Crypto. Designed like you asked. But personally, I think it has too many spinning spirals and blinking lights. It makes...hard reading. Make...tired... look...at...lose...all... con...cen...tra...tion...
Perfect!
CRYPTO THE HYPNOTIST

In this part . . .

In this part of the book, we describe the markup and document structures that make Web pages workable and attractive. To begin with, we examine gross HTML document structure, including document headers and bodies, and how to put the right pieces together. After that, we talk about organizing text in blocks and lists. Next, we explain how linking works in (X)HTML and how it provides the glue that ties the entire World Wide Web together. To wrap things up, we also explain how to add graphics to your pages. Thus, we cover the basic building blocks for well-constructed, properly proportioned Web pages — and not by coincidence, either

Chapter 4

Creating (X)HTML Document Structure

In This Chapter

- Creating a basic (X)HTML document structure
- Defining the (X)HTML document header
- Creating a full-bodied (X)HTML document

The framework for a simple (X)HTML document consists of a head and body. The head provides information about the document to the browser, and the body contains information that appears in the browser window. The first step toward creating any (X)HTML document is defining its framework.

This chapter covers the major elements needed to set up the basic structure of an (X)HTML document — including its head and body. We also show you how to tell the browser which version of HTML or XHTML you're using. Although version information isn't necessary for users, browsers use it to make sure they display document content correctly for your users.

Establishing a Document Structure

Although no two (X)HTML pages are alike — each employs a unique combination of content and elements to define a page — every properly constructed (X)HTML page follows the same basic document structure:

- A statement that identifies the document as an (X)HTML document
- A document header
- A document body

Each time you create an (X)HTML document, you start with these three elements; then you fill in the rest of your content and markup to create an individual page.

TIP

Although a basic document structure is a requirement for every (X)HTML document, creating it over and over again gets a little monotonous. Most (X)HTML-editing tools set up basic document structure automatically whenever you open a new document.

Labeling Your (X)HTML Document

At the top of your (X)HTML document sits the *Document Type Declaration,* or `DOCTYPE` *declaration.* This line of code specifies which version of HTML or XHTML you're using, and in turn lets browsers know how to interpret the document. We use the XHTML 1.0 specification in this chapter because it's widely used, and what most browsers and editing tools expect to see.

Adding an HTML DOCTYPE declaration

If you choose to create an HTML 4.01 document instead of an XHTML document, you can pick from three possible `DOCTYPE` declarations:

- **HTML 4.01 Transitional:** This is the most inclusive version of HTML 4.01, and it incorporates all HTML structural elements, as well as all presentation elements:

  ```
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
           "http://www.w3.org/TR/html4/loose.dtd">
  ```

- **HTML 4.01 Strict:** This streamlined version of HTML excludes all presentation-related elements in favor of style sheets as a mechanism for driving display:

  ```
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
  ```

- **HTML 4.01 Frameset:** This version begins with HTML 4.01 Transitional and adds all the elements that make frames possible:

  ```
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
           "http://www.w3.org/TR/html4/frameset.dtd">
  ```

Adding an XHTML DOCTYPE declaration

To create an XHTML document, use one of the following `DOCTYPE` declarations:

- **XHTML 1.0 Transitional:**

  ```
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
            "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
  ```

- **XHTML 1.0 Strict:**

  ```
  <!DOCTYPE html "-//W3C//DTD XHTML 1.0 Strict//EN"
            "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
  ```

- **XHTML 1.0 Frameset:**

  ```
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
            "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
  ```

REMEMBER

The XHTML DTD descriptions are similar to the HTML DTD descriptions defined in Chapter 1.

The <html> element

After you specify which version of (X)HTML the document follows, add an `<html>` element to contain all other (X)HTML elements in your page:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>




</html>
```

Adding the XHTML namespace

A *namespace* is a collection of names used by the elements and attributes in an XML document. XHTML uses a special collection of names; therefore it needs a namespace that looks like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">



</html>
```

TIP

Don't get bogged down by the meaning of namespaces. If you work with other XML vocabularies, you need to know about namespaces. For simple XHTML documents, you just need to remember to include the XHTML namespace. The preceding code snippet shows you exactly how to do so!

Adding a Document Header

The *head* of an (X)HTML document is one of two main components in a document. (The *body* of the document is the other main component.) The head, or *header,* provides basic information *about* the document, including its title and metadata (or information about information), such as keywords, author information, and a description. If you wish to use a style sheet with your page, you also include information about that style sheet in the header.

TIP

Chapter 8 provides a complete overview of creating Cascading Style Sheets (CSS) and shows you how to include them in (X)HTML documents.

The `<head>` element, which defines the page header, immediately follows the `<html>` opening tag:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
         "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>

  </head>
</html>
```

Giving your page a title

Every (X)HTML page needs a descriptive title to tell visitors what the page is about. This text appears in the title bar at the very top of the browser window, as shown in Figure 4-1. A page title should be concise yet informative. (For example, *My home page* isn't as informative as *Jeff's Web Design Services.*)

Define a page title by using the `<title>` element inside the `<head>` element:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
         "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Jeff's Web Design Services</title>
  </head>

</html>
```

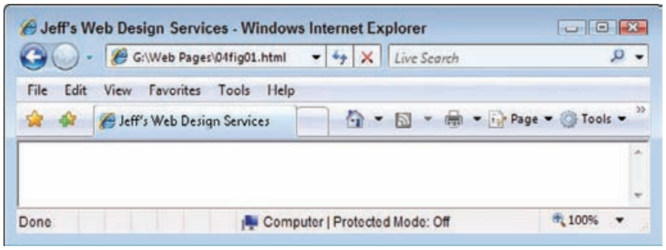

Jeff's Web Design Services - Windows Internet Explorer
G:\Web Pages\04fig01.html
Live Search
File Edit View Favorites Tools Help
Jeff's Web Design Services
Page
Tools
Done
Computer | Protected Mode: Off
100%

Figure 4-1: (X)HTML page titles appear in a Web browser's window title bar.

Search engines use `<title>` contents when they list Web pages in response to a query. The page title may be the first thing a Web surfer reads about your page, especially if she finds it using a search engine. In fact, a search engine will probably list your page title with many others on a search results page, which gives you only one chance to grab the Web surfer's attention and convince her to choose your page. A well-crafted title can do just that.

The title is also used for Bookmarks and in a browser's History; therefore keep your titles short and sweet.

Defining metadata

The term *metadata* refers to data about data. In the context of the Web, that means data that describes your Web page. Metadata for a page may include

- Keywords
- A description of your page
- Information about the page author
- The software application you used to create the page

Elements and attributes

You define each piece of metadata for your (X)HTML page with

- The `<meta />` element
- The `name` and `content` attributes

For example, the following elements create a list of keywords and a description for a consulting-service page:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
         "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
  <title>Jeff's Web Design Services</title>
  <meta name="keywords"
        content="Web consulting, page design, site construction" />
  <meta name="description"
        content="Synopsis of Jeff's skills and services" />
  </head>
</html>
```

Custom names

The (X)HTML specification doesn't

- Predefine the kinds of metadata you can include in your page
- Specify how to name different pieces of metadata, such as keywords and descriptions

So (for example) instead of using `keywords` and `description` as names for keyword and description metadata, you can just as easily use `kwrd` and `desc`, as in the following markup:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
         "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Jeff's Web Design Services</title>
    <meta name="kwrd"
          content=" Web consulting, page design, site construction " />
    <meta name="desc" content="Synopsis of Jeff's skills and services" />
  </head>
</html>
```

TECHNICAL STUFF

If you can use just any old values for the `<meta>` element's `name` and `content` attributes, how do systems know what to do with your metadata? The answer is — they don't. Each search engine works differently. Although keywords and description are commonly used metadata names, many search engines may not recognize or use other metadata elements that you include.

TIP

Many developers use metadata to either

- Leave messages for others who may look at the source code of the page
- Prepare for future browsers and search engines that use the metadata

TIP

Although keywords and page descriptions are optional, search engines use them to collect information about your Web site. Be sure to include detailed and concise information in your `<meta />` tag if you want your Web site discovered by search-engine robots.

Automatically redirecting users to another page

You can use metadata in your header to send messages to Web browsers about how they should display (or otherwise handle) your Web page. Web builders commonly use the `<meta />` element this way to redirect page visitors from one page to another automatically. For example, if you've ever come across a page that says `This page has moved. Please wait 10 seconds to be automatically sent to the new location.` (or something similar), you've seen this trick at work.

To use the `<meta />` element to send messages to the browser, here are the general steps you need to follow:

1. **Use the `http-equiv` attribute in place of the `name` attribute.**
2. **Choose from a predefined list of values that represents instructions for the browser.**

   These values are based on instructions that you can send to a browser in the HTTP header, but changing an HTTP header for a document is harder than embedding the instructions into the Web page itself.

To instruct a browser to redirect users from one page to another, here's what you need to do in particular:

1. **Use the `<meta />` element with `http-equiv="refresh"`.**
2. **Adjust the value of `content` to specify how many seconds before the refresh happens and what URL you want to jump to.**

For example, the `<meta />` element line in the following markup creates a refresh that jumps to `www.w3.org` after 15 seconds:

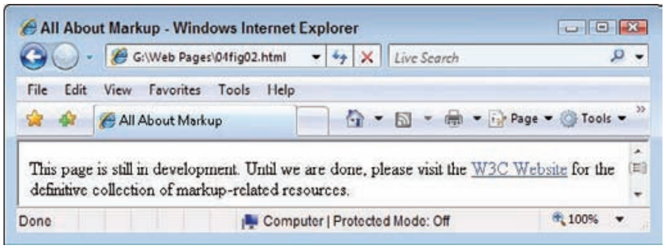
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>All About Markup</title>
    <meta http-equiv="refresh" content="15; url= http://www.w3.org/" />
  </head>

  <body>
    <p>This page is still in development. Until we are done, please visit
       the <a href="http://www.w3.org">W3C Website</a> for the definitive
       collection of markup-related resources.
    </p>

    <p>Please wait 10 seconds to be automatically redirected to the W3C.</p>
  </body>
</html>
```

WARNING!

Older Web browsers may not know what to do with `<meta />` elements that use the `http-equiv` element to create a redirector page. Be sure to include some text and a link on your page to enable a visitor to link manually to your redirector page if your `<meta />` element fails to do its job. (Linking is discussed in Chapter 2, and uses the anchor (`<a>`) element.)

If a user's browser doesn't know what to do with your redirect, the user simply clicks a link on the page to go to the new page, as in Figure 4-2.


All About Markup - Windows Internet Explorer
G:\Web Pages\04fig02.html
Live Search
File Edit View Favorites Tools Help
All About Markup
Page Tools

This page is still in development. Until we are done, please visit the W3C Website for the definitive collection of markup-related resources.

Done
Computer | Protected Mode: Off
100%

Figure 4-2: When you use a <meta /> element to create a page redirector, include a link in case the redirector fails.

You can use the `http-equiv` attribute with the `<meta />` element for a variety of purposes, such as setting an expiration date for a page and specifying a character set (the language) for the page to use. To find out what your `http-equiv` options are (and how to use them), check out the Dictionary of HTML META Tags at the following URL:

```
http://vancouver-webpages.com/META/metatags.detail.html
```

Creating the (X)HTML Document Body

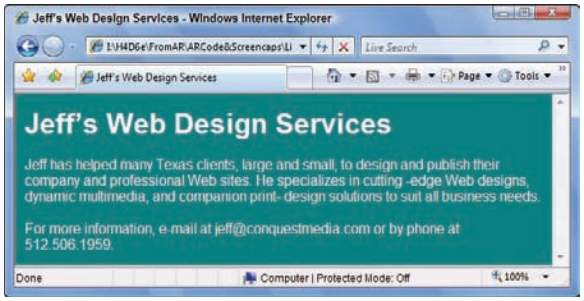
After you set up your page header, create a title, and define some metadata, you're ready to create the (X)HTML markup and content that will show up in a browser window. The <body> element holds the content of your document.

If you want to see something in your browser window, put it in the <body> element, like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
         "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Jeff's Web Design Services</title>
    <meta name="kwrd"
          content=" Web consulting, page design, site construction " />
    <meta name="desc" content="Synopsis of Jeff's skills and services" />
  </head>


<body type="text/css"
        style="color: white;
        background-color: teal;
               font-size: 1.2;
               font-family: sans-serif">    <h1>Jeff's Web Design Services</h1>
    <p>Jeff has helped many Texas clients, large and small, to design and
       publish their company and professional Web sites. He specializes in
       cutting-edge Web designs, dynamic multimedia, and companion print-
       design solutions to suit all business needs.</p>

    <p>For more information, e-mail at jeff@conquestmedia.com or
       by phone at 512.506.1959.</p>
  </body>
</html>
```

Figure 4-3 shows how a browser displays this complete (X)HTML page:

- The content of the <title> element is in the window's title bar.
- The <meta /> elements don't affect the page appearance at all.
- Only the paragraph text contained in the heading (<h1>) and <p> elements (in the <body> element) actually appears in the browser window.


Jeff's Web Design Services - Windows Internet Explorer
Live Search
Jeff's Web Design Services
Page
Tools
Done
Computer | Protected Mode: Off
100%

Jeff's Web Design Services

Jeff has helped many Texas clients, large and small, to design and publish their company and professional Web sites. He specializes in cutting-edge Web designs, dynamic multimedia, and companion print- design solutions to suit all business needs.

For more information, e-mail at jeff@conquestmedia.com or by phone at 512.506.1959.

Figure 4-3: Only content in the body. element appers in the browser window

Chapter 5

Text and Lists

In This Chapter

- ▶ Working with basic blocks of text
- ▶ Manipulating text blocks
- ▶ Creating bulleted, numbered, and definition lists

HTML documents consist of text, images, multimedia files, links, and other bits of content that you meld together into a page by using markup elements and attributes. You use blocks of text to create such things as headings, paragraphs, and lists. The first step in creating a solid HTML document is laying a firm foundation that establishes the document's structure.

Formatting Text

Here's a plus-ultra-technical definition of a *block of text:* some chunk of content that wraps from one line to another inside an HTML element.

In fact, your HTML page is a giant collection of blocks of text:

- Every bit of content on your page must be part of some block element.
- Every block element sits within the `<body>` element on your page.

HTML recognizes several kinds of text blocks that you can use in your document, including (but not limited to)

- Paragraphs
- Headings
- Block quotes
- Lists
- Tables
- Forms

Inline elements versus text blocks

The difference between inline elements and a block of text is important. HTML elements in this chapter describe blocks of text. An *inline element* is a word or string of words *inside* a block element (for example, text emphasis elements such as <em> or <strong>). Inline elements must be nested within a block element; otherwise, your HTML document isn't syntactically correct.

Inline elements, such as linking and formatting elements, are designed to link from (or change the appearance of) a few words or lines of content found inside those blocks.

Paragraphs

Paragraphs appear more often in Web pages than any other kind of text block.

HTML browsers don't recognize hard returns that you enter when you create your page inside an editor. You must use a <p> element to tell the browser to separate all contained text up to the closing </p> as a paragraph.

Formatting

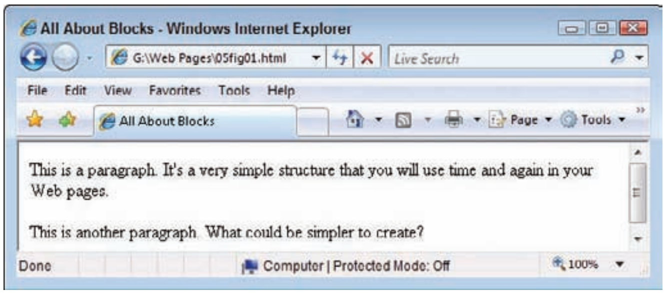
To create a paragraph, follow these steps:

1. **Add <p> in the body of the document.**
2. **Type the content of the paragraph.**
3. **Add </p> to close that paragraph.**

Here's what it looks like:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>All About Blocks</title>
  </head>

  <body>
    <p>This is a paragraph. It's a very simple structure that you will use
       time and again in your Web pages.</p>
    <p>This is another paragraph. What could be simpler to create?</p>
  </body>
</html>
```

This HTML page includes two paragraphs, each marked with a separate `<p>` element. Most Web browsers add a line break and a full line of white space after every paragraph on your page, as shown in Figure 5-1.


All About Blocks - Windows Internet Explorer
G:\Web Pages\05fig01.html
Live Search
File Edit View Favorites Tools Help
All About Blocks
Page Tools
Done
Computer | Protected Mode: Off
100%

This is a paragraph. It's a very simple structure that you will use time and again in your Web pages.

This is another paragraph. What could be simpler to create?

Figure 5-1: Web browsers delineate paragraphs with line breaks.

WARNING!

Sloppy HTML coders don't use the closing `</p>` tag when they create paragraphs. Although some browsers permit this dubious practice without yelling, omitting the closing tag

- Isn't correct syntax
- Causes problems with style sheets
- Can cause a page to appear inconsistently from one browser to another

You can control paragraph formatting (color, style, size, and alignment) by using Cascading Style Sheets (CSS), which we cover in Chapters 8 and 9.

Alignment

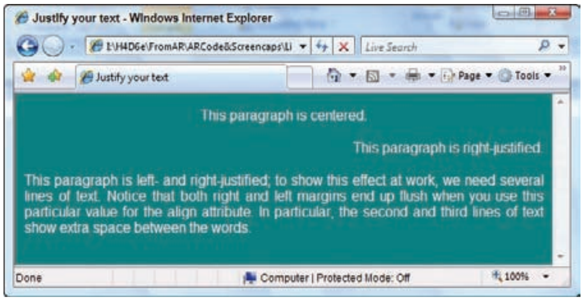
By default, the paragraph aligns to the left. You can use the `align` attribute with a value of `left`, `center`, `right`, or `justify` to control its alignment explicitly.

```
<p align="center">This paragraph is centered.</p>
<p align="right">This paragraph is right-justified.</p>
<p align="justify">This paragraph is left- and right-justified; to show
    this effect at work, we need several lines of text. Notice that
    both right and left margins end up flush when you use this particular
    value for the align attribute. In particular, the second and third
    lines of text show extra space between the words.</p>
```

Figure 5-2 shows how a Web browser aligns each paragraph according to the value of the `align` attribute.

WARNING!

The `align` attribute has been deprecated (made obsolete) in favor of using CSS (see Chapter 8).


Justify your text - Windows Internet Explorer
I:\H4D6e\FromAR\ARCode&Screencaps\Li
Live Search
Justify your text
Page
Tools
Done
Computer | Protected Mode: Off
100%

This paragraph is centered.

This paragraph is right-justified.

This paragraph is left- and right-justified; to show this effect at work, we need several lines of text. Notice that both right and left margins end up flush when you use this particular value for the align attribute. In particular, the second and third lines of text show extra space between the words.

Figure 5-2: Use the `align` attribute with a paragraph to specify horizontal alignment.

Headings

Headings break a document into sections. This book uses headings and subheadings to divide every chapter into sections, and you can do the same with your Web page. Headings

- Create an organizational structure
- Break up the visual appearance of the page
- Give visual clues about how pieces of content are grouped

HTML includes six elements for up to six different heading levels in your documents:

- `<h1>` is the most prominent heading (Heading 1)
- `<h6>` is the least prominent heading (Heading 6)

TIP

Follow order from highest to lowest as you use HTML heading levels. That is, don't use a second-level heading until you've used a first-level heading, don't use a third-level heading until you've used a second, and so on. If you want to change how headings look, Chapter 8 and Chapter 9 show you how to use style sheets for that purpose.

Formatting

To create a heading, follow these steps:

1. **Add `<hn>` in the body of your document.**
2. **Type the content for the heading.**
3. **Add `</hn>`.**

Browser displays

Every browser has a different way of displaying heading levels, as you see in the next two sections.

Graphical browsers

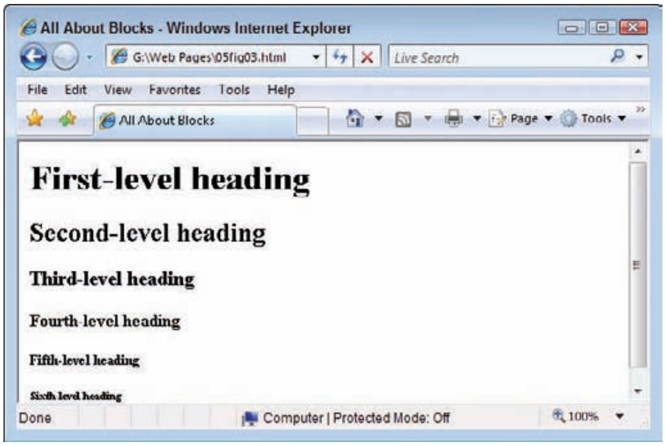
Most graphical browsers use a distinctive size and typeface for headings:

- First-level headings (`<h1>`) are the largest (usually two or three font sizes larger than the default text size for paragraphs).
- All headings use boldface type by default, whereas paragraph text uses plain (non-bold) type by default.
- Sixth-level headings (`<h6>`) are the smallest and may be two or three font sizes *smaller* than the default paragraph text.

The following snippet of HTML markup shows all six headings at work:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>All About Blocks</title>
  </head>

  <body>
    <h1>First-level heading</h1>
    <h2>Second-level heading</h2>
    <h3>Third-level heading</h3>
    <h4>Fourth-level heading</h4>
    <h5>Fifth-level heading</h5>
    <h6>Sixth-level heading</h6>
  </body>
</html>
```

Figure 5-3 shows this HTML page as rendered in a browser.


All About Blocks - Windows Internet Explorer
G:\Web Pages\05fig03.html
Live Search
File Edit View Favorites Tools Help
All About Blocks
Page Tools
Done
Computer | Protected Mode: Off
100%

First-level heading

Second-level heading

Third-level heading

Fourth-level heading

Fifth-level heading

Sixth-level heading

Figure 5-3: Web browsers display headings in decreasing size from level one to level six.

Use CSS to control the display of headings, including color, size, spacing, and alignment.

By default, most browsers use Times Roman fonts for headings. The font size decreases as heading level increases. (Default sizes for first- through sixth-level headings are, respectively, 24, 18, 14, 12, 10, and 8.) You can override any of this formatting by using CSS.

Text browsers

Text-only browsers use heading conventions different from those of graphical browsers because text-only browsers use a single character size and font to display all content.

Controlling Text Blocks

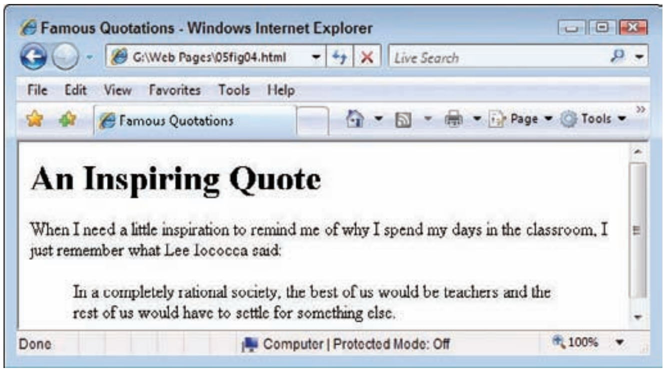
Blocks of text build the foundation for your page. You can break those blocks into smaller pieces to better guide readers through your content.

Block quotes

A *block quote* is a long quotation or excerpt from a printed source that you set apart on your page. Use the `<blockquote>` element to identify block quotes:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>Famous Quotations</title>
  </head>

  <body>
    <h1>An Inspiring Quote</h1>
    <p>When I need a little inspiration to remind me of why I spend my days
       in the classroom, I just remember what Lee Iococca said:</p>
    <blockquote>
      In a completely rational society, the best of us would be teachers
      and the rest of us would have to settle for something else.
    </blockquote>
  </body>
</html>
```

Most Web browsers display block-quote content with a slight left indent, as shown in Figure 5-4.


Famous Quotations - Windows Internet Explorer
G:\Web Pages\05fig04.html
Live Search
File Edit View Favorites Tools Help
Famous Quotations
Page Tools
Done
Computer | Protected Mode: Off
100%

An Inspiring Quote

When I need a little inspiration to remind me of why I spend my days in the classroom, I just remember what Lee Iococca said:

In a completely rational society, the best of us would be teachers and the rest of us would have to settle for something else.

Figure 5-4: Web browsers typically indent a block quote to separate it from paragraphs.

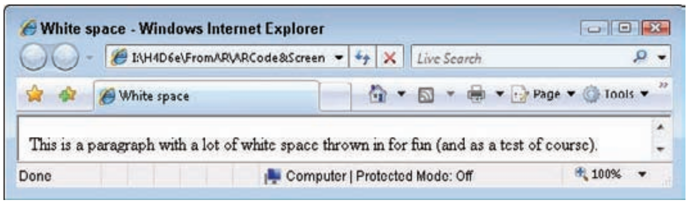
Preformatted text

Ordinarily, HTML ignores white space inside documents. A browser won't display a block element's

- Hard returns
- Line breaks
- Large white spaces

The following markup includes several hard returns, line breaks, and a lot of space characters. Figure 5-5 shows that the Web browser ignores all of this.

```
<p>This is a paragraph

   with a lot of white space

        thrown in for fun (and as a test of course).</p>
```

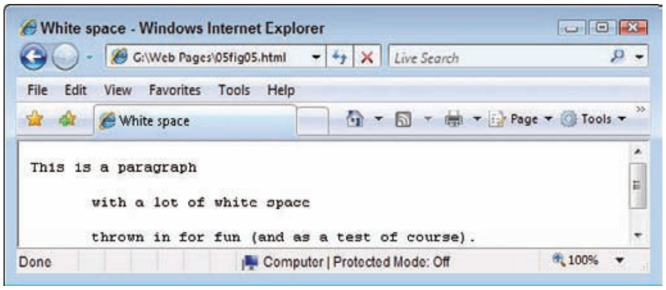

White space - Windows Internet Explorer
I:\H4D6e\FromAR\ARCode&Screen
Live Search
White space
Page Tools
Done
Computer | Protected Mode: Off
100%

This is a paragraph with a lot of white space thrown in for fun (and as a test of course).

Figure 5-5: Web browsers routinely ignore white space.

The preformatted text element (`<pre>`) instructs browsers to keep all white space intact as it displays your content (like the following sample). Use the `<pre>` element in place of the `<p>` element to make the browser apply all your white space, as shown in Figure 5-6.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>White space</title>
  </head>

  <body>
    <pre>This is a paragraph

       with a lot of white space

       thrown in for fun (and as a test of course).
    </pre>
  </body>
</html>
```


White space - Windows Internet Explorer
G:\Web Pages\05fig05.html
Live Search
File Edit View Favorites Tools Help
White space
Page Tools
Done
Computer | Protected Mode: Off
100%

```
This is a paragraph

      with a lot of white space

      thrown in for fun (and as a test of course).
```

Figure 5-6: Use preformatted text to force browsers to recognize white space.

TIP

You may want the browser to display white spaces in an HTML page where proper spacing is important, such as for

- Code samples
- Text tables

TIP

You can nest `<pre>` elements inside `<blockquote>` elements to carefully control how lines of quoted text appear on the page. Or better still, forget about these tags and use CSS to position text blocks inside `<div>` elements.

Line breaks

By default, browsers usually *wrap* text that appears in block elements, such as paragraphs, headings, and block quotes. If a line reaches the end of the browser window, the next word automatically starts on a new line. Use a *line break* to force an end to any line of text (denoted by the `<br />` element).

Function

The `<br />` element is the HTML equivalent of the manual line break that you enter after paragraphs and other blocks of text when you're using a word-processing program. When a browser sees a `<br />`, it ends the line there and starts the next line.

TECHNICAL STUFF

The difference between a line break and a paragraph is that a line break doesn't use any special formatting that you can apply at the end or beginning of a paragraph, such as

- Extra vertical space
- First-line indenting

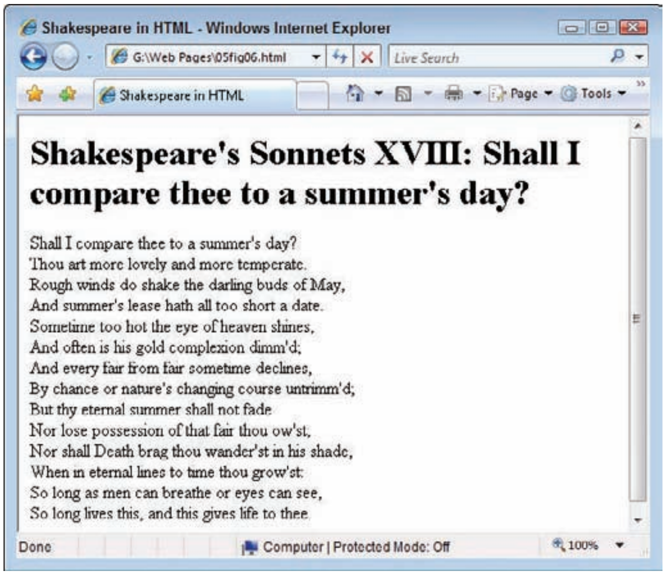
Formatting

The following markup formats lines of text in a poem using line breaks. The entire poem is *a single paragraph,* where `<br />` marks the end of each line:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title> Shakespeare in HTML</title>
  </head>

  <body>
  <h1>Shakespeare's Sonnets XVIII: Shall I compare thee to a summer's day? </h1>
    <p>
      Shall I compare thee to a summer's day? <br />
      Thou art more lovely and more temperate. <br />
      Rough winds do shake the darling buds of May, <br />
      And summer's lease hath all too short a date. <br />
      Sometime too hot the eye of heaven shines, <br />
      And often is his gold complexion dimm'd; <br />
      And every fair from fair sometime declines, <br />
      By chance or nature's changing course untrimm'd; <br />
      But thy eternal summer shall not fade <br />
```

```
        Nor lose possession of that fair thou ow'st; <br />
        Nor shall Death brag thou wander'st in his shade, <br />
        When in eternal lines to time thou grow'st: <br />
        So long as men can breathe or eyes can see, <br />
        So long lives this, and this gives life to thee. <br />
      </p>
    </body>
  </html>
```

Figure 5-7 shows how a browser handles each line break. Here, the poem isn't left-indented because a `<p>` element replaces the `<blockquote>` element.


Shakespeare in HTML - Windows Internet Explorer
G:\Web Pages\05fig06.html
Live Search
Shakespeare in HTML
Page
Tools
Done
Computer | Protected Mode: Off
100%

Shakespeare's Sonnets XVIII: Shall I compare thee to a summer's day?

Shall I compare thee to a summer's day?
Thou art more lovely and more temperate.
Rough winds do shake the darling buds of May,
And summer's lease hath all too short a date.
Sometime too hot the eye of heaven shines,
And often is his gold complexion dimm'd;
And every fair from fair sometime declines,
By chance or nature's changing course untrimm'd;
But thy eternal summer shall not fade
Nor lose possession of that fair thou ow'st,
Nor shall Death brag thou wander'st in his shade,
When in eternal lines to time thou grow'st:
So long as men can breathe or eyes can see,
So long lives this, and this gives life to thee.

Figure 5-7: Using the <br /> element to specify where lines in block elements should break.

Horizontal rules

The horizontal rule element (`<hr />`) helps you include solid straight lines *(rules)* on your page.

TIP

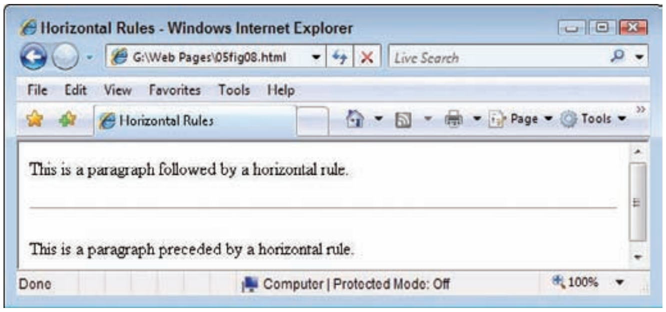
The browser creates the rule based on the `<hr />` element, so users don't wait for a graphic to download. A horizontal rule is a good option to

- Break a page into logical sections.
- Separate headers and footers from the rest of the page.

Formatting

When you include an `<hr />` element on your page, as in the following XHTML, the browser replaces it with a line, as shown in Figure 5-8.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>Horizontal Rules</title>
  </head>

  <body>
    <p>This is a paragraph followed by a horizontal rule.</p>

    <hr />

    </p>This is a paragraph preceded by a horizontal rule.</p>
  </body>
</html>
```


Horizontal Rules - Windows Internet Explorer
G:\Web Pages\05fig08.html
Live Search
File Edit View Favorites Tools Help
Horizontal Rules
Page Tools
Done
Computer | Protected Mode: Off
100%

This is a paragraph followed by a horizontal rule.

This is a paragraph preceded by a horizontal rule.

Figure 5-8: Use the <hr /> element to add horizontal lines to your page.

TIP

A horizontal rule always sits on a line by itself; you can't add the `<hr />` element in the middle of a paragraph (or other block element) and expect the rule to appear in the middle of the block.

Attributes

Four different attributes control the appearance of each horizontal rule:

- `width`: Specifies line width either in *pixels* or by *percentage of display area width* (which we call "the page" in discussion that follows).

  For example, a rule can be 50 pixels wide or take 75 percent of the page.
- `size`: Specifies the height of the line in pixels. The default is 1 pixel.

TECHNICAL STUFF

- `align`: Specifies the horizontal alignment of the rule as either `left` (the default), `center`, or `right`.

  If you don't define a width for your rule, it takes the entire width of the page. The alignment won't make any difference.

- `noshade`: Specifies a solid line with no shading.

  By default, most browsers display hard rules with a shade.

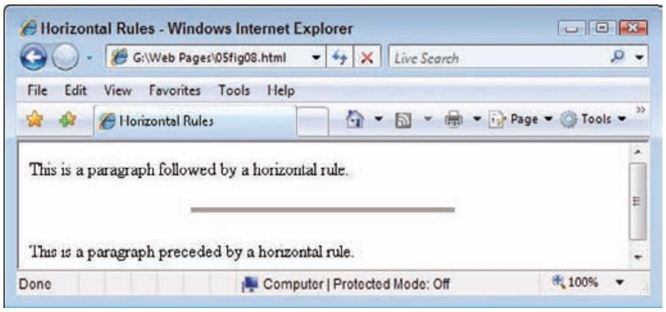
TIP

These formatting attributes are deprecated in favor of CSS.

This bit of HTML creates a horizontal rule that takes up 45 percent of the page, is 4 pixels high, aligned to the center, and has shading turned off:

```
<p>This is a paragraph followed by a horizontal rule.</p>

<hr width="45%" size="4" align="center" noshade="noshade" />

<p>This is a paragraph preceded by a horizontal rule.</p>
```

Figure 5-9 shows how adding these attributes alters how the rule is displayed.


Horizontal Rules - Windows Internet Explorer
G:\Web Pages\05fig08.html
Live Search
File Edit View Favorites Tools Help
Horizontal Rules
Page Tools
Done
Computer | Protected Mode: Off
100%

This is a paragraph followed by a horizontal rule.

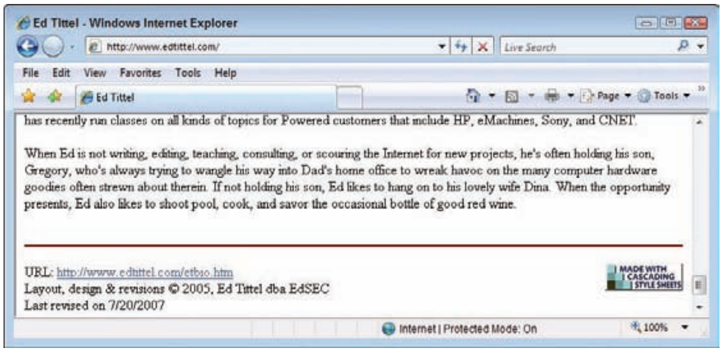
This is a paragraph preceded by a horizontal rule.

Figure 5-9: Use the <hr /> attributes to better control how a browser displays the rule.

Figure 5-10 shows how you can use horizontal rules in the real world to highlight important content. The EdTittel.com home page uses a colored hard rule to separate the footer from the rest of the page.

TIP

CSS gives you much more control over the placement of horizontal rules; you can even fancy them up with more advanced color and shading options.


Ed Tittel - Windows Internet Explorer
http://www.edtittel.com/
Live Search
File Edit View Favorites Tools Help
Ed Tittel
Page
Tools
MADE WITH CASCADING STYLE SHEETS
Internet | Protected Mode: On
100%

has recently run classes on all kinds of topics for Powered customers that include HP, eMachines, Sony, and CNET.

When Ed is not writing, editing, teaching, consulting, or scouring the Internet for new projects, he's often holding his son, Gregory, who's always trying to wangle his way into Dad's home office to wreak havoc on the many computer hardware goodies often strewn about therein. If not holding his son, Ed likes to hang on to his lovely wife Dina. When the opportunity presents, Ed also likes to shoot pool, cook, and savor the occasional bottle of good red wine.

URL: http://www.edtittel.com/etbio.htm
Layout, design & revisions © 2005, Ed Tittel dba EdSEC
Last revised on 7/20/2007

Figure 5-10: The EdTittel.com home page uses a colored rule to separate page content from page-footer information.

Organizing Information

Lists are powerful tools for arranging similar elements together, and they give visitors to your site an easy way to zoom in on groups of information. Just about anything fits in a list, from sets of instructions to collections of links.

Lists use a combination of elements — at least two components:

- A markup element that says "Hey browser! The following items go in a list."
- Markup elements that say "Hey browser! This is an item in the list."

HTML supports three different types of lists:

- Numbered lists
- Bulleted lists
- Definition lists

Numbered lists

A *numbered list* consists of at least two items, each prefaced by a number. Usually, a person numbers a list when the order of items is important.

You use two kinds of elements for a numbered list:

- The ordered list element (`<ol>`) specifies that this is a numbered list.
- List item elements (`<li>`) mark each item in the list.

Formatting

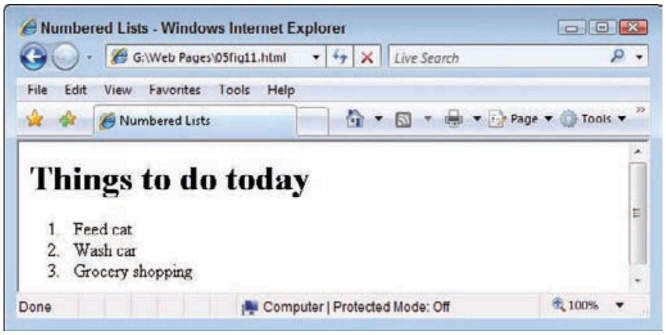
A numbered list with three items requires elements and content in the following order:

1. `<ol>`
2. `<li>`
3. Content for the first list item
4. `</li>`
5. `<li>`
6. Content for the second list item
7. `</li>`
8. `<li>`
9. Content for the third list item
10. `</li>`
11. `</ol>`

The following markup defines a three-item numbered list:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>Numbered Lists</title>
  </head>

  <body>
    <h1>Things to do today</h1>
    <ol>
      <li>Feed cat</li>
      <li>Wash car</li>
      <li>Grocery shopping</li>
    </ol>
  </body>
</html>
```

Figure 5-11 shows how a browser renders this markup. You don't actually have to specify a number for each item in the list; the browser identifies the list items from the markup and adds the numbers.


Numbered Lists - Windows Internet Explorer
G:\Web Pages\05fig11.html
Live Search
File Edit View Favorites Tools Help
Numbered Lists
Page Tools
Done
Computer | Protected Mode: Off
100%

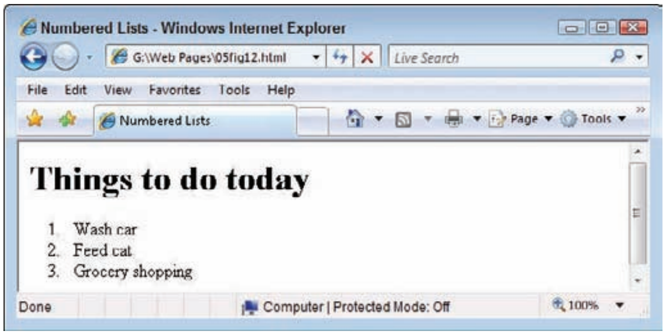
Things to do today

1. Feed cat
2. Wash car
3. Grocery shopping

Figure 5-11: Use the <ol> and <li> tags to create a numbered list.

If you swap the first two items in the list, they're still numbered in order when the page appears, as shown in Figure 5-12.

```
<ol>
  <li>Wash car</li>
  <li>Feed cat</li>
  <li>Grocery shopping</li>
</ol>
```


Numbered Lists - Windows Internet Explorer
G:\Web Pages\05fig12.html
Live Search
File Edit View Favorites Tools Help
Numbered Lists
Page Tools
Done
Computer | Protected Mode: Off
100%

Things to do today

1. Wash car
2. Feed cat
3. Grocery shopping

Figure 5-12: Web browsers set numbers for your list according to order of appearance.

Numbering

Two different `<ol>` attributes control the appearance of a numbered list:

- `start`: Specifies the first number in the list.
  - The default starting number is 1.
  - You can specify any number as the start number for the new list.

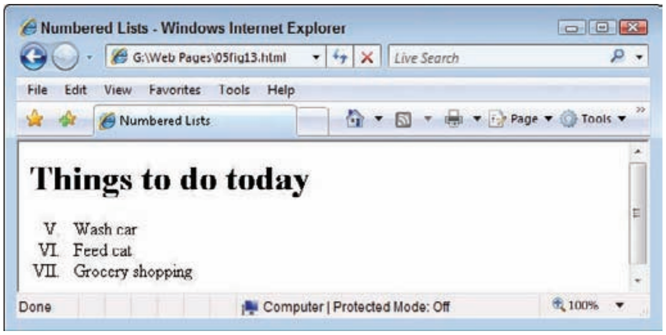
    TIP Specify a start number when you resume a list after an unnumbered paragraph or some other block element.

- `type`: Specifies the numbering style from the list. You can choose from five predefined numbering styles:
  - `1`: Decimal numbers.
  - `a`: Lowercase letters.
  - `A`: Uppercase letters.
  - `i`: Lowercase Roman numerals.
  - `I`: Uppercase Roman numerals.

The following markup uses ordered elements and attributes to build a list that uses uppercase Roman numerals that begin at 5 (V in Roman numerals):

```
<ol start="5" type="I">
  <li>Wash car</li>
  <li>Feed cat</li>
  <li>Grocery shopping</li>
</ol>
```

Figure 5-13 shows how attributes affect the list's appearance in a browser.


Numbered Lists - Windows Internet Explorer
G:\Web Pages\05fig13.html
Live Search
File Edit View Favorites Tools Help
Numbered Lists
Page Tools
Done
Computer | Protected Mode: Off
100%

Things to do today

V. Wash car
VI. Feed cat
VII. Grocery shopping

Figure 5-13: The start and type attributes guide the appearance of a numbered list in a browser.

TIP

You have more control over your lists if you use CSS to define formatting. That's why the `start` and `type` attributes for list markup are *deprecated* (that is, abandoned as outmoded in the current version).

Bulleted lists

A *bulleted list* consists of one or more items each prefaced by a *bullet* (often a big dot; this book uses *check marks* as bullets).

You use this type of list if the items' order of presentation isn't necessary for understanding the information presented.

Formatting

A bulleted list requires the following:

- The unordered list element (`<ul>`) specifies a bulleted list.
- A list item element (`<li>`) marks each item in the list.
- The closing tag for the unordered list element (`</ul>`) indicates that the list has come to its end.

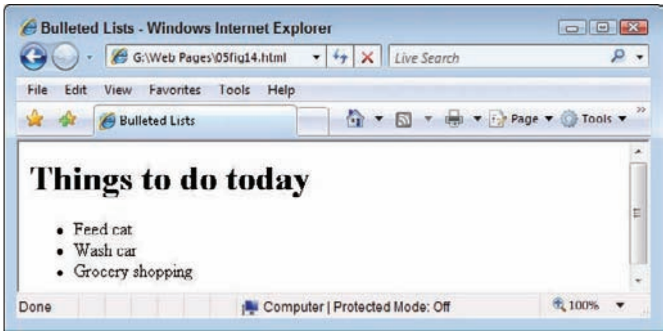
An *unordered list* (another name for bulleted list) with three items requires elements and content in the following order:

1. `<ul>`
2. `<li>`
3. Content for the first list item
4. `</li>`
5. `<li>`
6. Content for the second list item
7. `</li>`
8. `<li>`
9. Content for the third list item
10. `</li>`
11. `</ul>`

The following markup formats a three-item list as a bulleted list:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>Bulleted Lists</title>
  </head>

  <body>
    <h1>Things to do today</h1>
    <ul>
      <li>Feed cat</li>
      <li>Wash car</li>
      <li>Grocery shopping</li>
    </ul>
  </body>
</html>
```

Figure 5-14 shows how a browser renders this with bullets.


Bulleted Lists - Windows Internet Explorer
G:\Web Pages\05fig14.html
Live Search
File Edit View Favorites Tools Help
Bulleted Lists
Page Tools
Done
Computer | Protected Mode: Off
100%

Things to do today

- Feed cat
- Wash car
- Grocery shopping

Figure 5-14: An unordered list uses bullets instead of numbers to mark items.

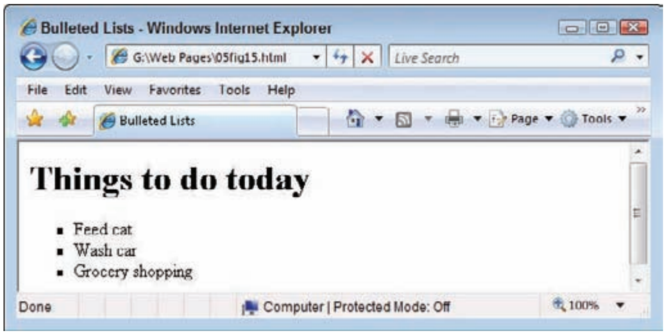
Styles

You can use the `type` attribute (deprecated) with the `<ul>` element to specify what kind of bullet you want the list to use.

- `disc`: Solid circle bullets (the default)
- `square`: Solid square bullets
- `circle`: Hollow circle bullets

The addition of the `type` attribute to the bulleted-list markup just given changes bullets from discs to squares, as shown in Figure 5-15. Here's what that markup looks like:

```
<ul type="square">
  <li>Feed cat</li>
  <li>Wash car</li>
  <li>Grocery shopping</li>
</ul>
```


Bulleted Lists - Windows Internet Explorer
G:\Web Pages\05fig15.html
Live Search
File Edit View Favorites Tools Help
Bulleted Lists
Page Tools
Done
Computer | Protected Mode: Off
100%

Things to do today

▪ Feed cat
▪ Wash car
▪ Grocery shopping

Figure 5-15: Use the type attribute to change the bullet style for an unordered list.

TIP

Use CSS if you want more control over the formatting of your lists, including the ability to use your own graphics as bullet symbols.

Definition lists

Definition lists group terms and definitions into a single list and require three different elements to complete the list:

- `<dl>`: Holds the list definitions.
- `<dt>`: Defines a term in the list.
- `<dd>`: Defines a definition for a term.

You can have as many terms (defined by `<dt>`) in a list as you need. Each term can have one or more definitions (defined by `<dd>`).

To create a definition list with two items requires elements and content in the following order:

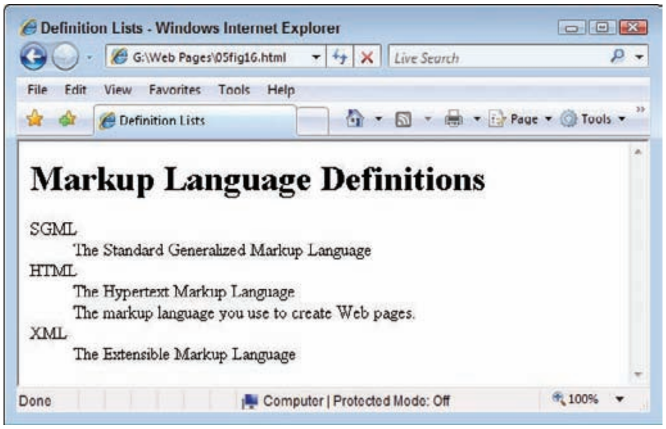
1. `<dl>`
2. `<dt>`
3. First term name
4. `</dt>`
5. `<dd>`
6. Content for the definition of the first item
7. `</dd>`
8. `<dt>`
9. Second term name
10. `</dt>`
11. `<dd>`
12. Content for the definition of the second item
13. `</dd>`
14. `</dl>`

The following definition list includes three terms, one of which has two definitions:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
```

```
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
        <title>Definition Lists</title>
      </head>

      <body>
        <h1>Markup Language Definitions</h1>
        <dl>
          <dt>SGML</dt>
            <dd>The Standard Generalized Markup Language</dd>
          <dt>HTML</dt>
             <dd>The Hypertext Markup Language</dd>
             <dd>The markup language you use to create Web pages.</dd>
          <dt>XML</dt>
            <dd>The Extensible Markup Language</dd>
        </dl>
      </body>
    </html>
```

Figure 5-16 shows how a browser displays this HTML.


Definition Lists - Windows Internet Explorer
G:\Web Pages\05fig16.html
Live Search
File Edit View Favorites Tools Help
Definition Lists
Page Tools
Done
Computer | Protected Mode: Off
100%

Markup Language Definitions

SGML
    The Standard Generalized Markup Language
HTML
    The Hypertext Markup Language
    The markup language you use to create Web pages.
XML
    The Extensible Markup Language

Figure 5-16: Definition lists group terms and their related definitions into one list.

TIP

If you think the items in a list are too close together, you can take one of two actions:

- Put two `<br/>` elements before each `</li>` or `</dd>` element to add more white space.
- Use CSS styles to carefully control all aspects of list appearance, as shown in Chapter 8.

Note that definition lists often display differently inside different browsers, and aren't always handled the same by search engines or text-to-speech translators. Alas, this means definition lists may not be the best choice of formatting for lists you create (even lists of definitions). See the excellent coverage of this topic at `www.maxdesign.com/au/presentation/definition` for a more detailed discussion.

Nesting lists

You can create subcategories by *nesting* lists within lists. Some common uses for nested lists include

- Site maps and other navigation tools
- Tables of contents for online books and papers
- Outlines

You can combine any of the three kinds of lists to create *nested lists,* such as a multilevel table of contents or an outline that mixes numbered headings with bulleted list items as the lowest outline level.

The following example starts with a numbered list that defines a list of things to do for the day, and uses three bulleted lists to break down those items further, into specific tasks:

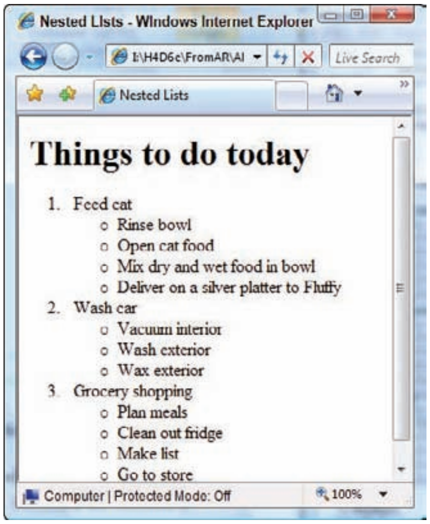
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>Nested Lists</title>
  </head>

  <body>
    <h1>Things to do today</h1>
    <ol>
      <li>Feed cat</li>
        <ul>
          <li>Rinse bowl</li>
          <li>Open cat food</li>
          <li>Mix dry and wet food in bowl</li>
          <li>Deliver on a silver platter to Fluffy</li>
        </ul>
      <li>Wash car</li>
        <ul>
          <li>Vacuum interior</li>
          <li>Wash exterior</li>
          <li>Wax exterior</li>
        </ul>
      <li>Grocery shopping</li>
        <ul>
          <li>Plan meals</li>
```

```
                <li>Clean out fridge</li>
                <li>Make list</li>
                <li>Go to store</li>
              </ul>
           </ol>
        </body>
      </html>
```

All nested lists follow the same markup pattern:

- Each list item in the top-level ordered list is followed by a complete second-level list.
- The second-level lists sit inside the top-level list, not in the list items.

Figure 5-17 shows how a browser reflects this nesting in its display.


Nested Lists - Windows Internet Explorer
E:\H4D6c\FromAR\AI
Live Search
Nested Lists
Computer | Protected Mode: Off
100%

Things to do today

1. Feed cat
   - Rinse bowl
   - Open cat food
   - Mix dry and wet food in bowl
   - Deliver on a silver platter to Fluffy
2. Wash car
   - Vacuum interior
   - Wash exterior
   - Wax exterior
3. Grocery shopping
   - Plan meals
   - Clean out fridge
   - Make list
   - Go to store

Figure 5-17: Nested lists combine lists for a multilevel organization of information.

WARNING!

As you build nested lists, watch opening and closing tags carefully. Close first what you opened last is an important axiom. If you don't open and close tags properly, lists might not use consistent indents or numbers, or text might be indented incorrectly because a list somewhere was never properly closed.

Chapter 6

Linking to Online Resources

In This Chapter

- Creating simple links
- Opening linked pages in new windows
- Setting up links to locations within a Web page
- Creating links to things other than Web pages

Hyperlinks, or simply *links,* connect (X)HTML pages and other resources on the Web. When you include a link on your page, you enable visitors to travel from your page to another Web site, another page on your site, or even another location on the same page. Without links, a page stands alone, disconnected from the rest of the Web. With links, that page becomes part of an almost boundless collection of information.

Basic Links

To create a link, you need

- **The Web address** (called a Uniform Resource Locator, or URL) for the Web site or file that's your link target. This usually starts with `http://`.
- **Some text** in your Web page to label or describe the link.

  Try to ensure that the text you use says something useful about the resource being linked.
- **An anchor element (`<a>`) with `href` attribute** to bring it all together.

  The element to create links is called an *anchor element* because you use it to anchor a URL to some text on your page. When users view your page in a browser, they can click the text to activate the link and visit the page whose URL you specified in that link. You insert the full URL in the `href` attribute. This tells the link where to go.

TIP

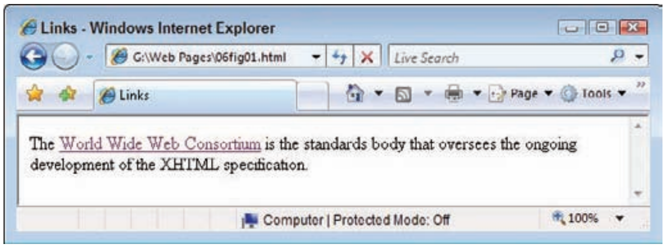
You can think of the structure of a basic link as a cheeseburger (or your preferred vegan substitute). The URL is the cheese, the patty is the link text, and the anchor tags are the buns. Tasty, yes?

For example, if you have a Web page that describes HTML standards, you may want to refer Web surfers to the World Wide Web Consortium (W3C) — the organization that governs all things related to (X)HTML standards. A basic link to the W3C's Web site, `www.w3.org`, looks like this:

```
<p>The <a href="http://www.w3.org">World Wide Web Consortium</a> is the
   standards body that oversees the ongoing development of the XHTML
   specification.</p>
```

You specify the link URL (`http://www.w3.org`) in the anchor element's `href` attribute. The text (`World Wide Web Consortium`) between the anchor element's open and close tags (`<a>` and `</a>`) labels or describes the link.

Figure 6-1 shows how a browser displays this bit of markup.


Links - Windows Internet Explorer
G:\Web Pages\06fig01.html
Live Search
Links
Page
Tools
Computer | Protected Mode: Off
100%

The World Wide Web Consortium is the standards body that oversees the ongoing development of the XHTML specification.

Figure 6-1: A paragraph with a link to the W3C.

TIP

You can also anchor URLs to images so users can click an image to activate a link. (For more about creating images that link, see Chapter 7.)

For a detailed discussion of the ins and outs of URLs, see Chapter 1.

Link options

You can link to a variety of online resources:

- Other (X)HTML pages (either on your Web site or on another Web site)
- Different locations on the same (X)HTML page
- Resources that aren't even (X)HTML pages at all, such as e-mail addresses, pictures, and text files

Anchor elements aren't block elements

Anchor elements are *inline elements* — they apply to a few words or characters within a block of text (the text that you want to use as a link) instead of defining formatting for blocks of text. The anchor element typically sits inside a paragraph (`<p>`) or other block element, such as a paragraph or list item. When you create a link, you should always create it within a block element, such as a paragraph, list item, heading, or even a table cell. Turn to Chapter 5 for more information on block elements.

Although many Web browsers display anchors just fine even if you don't nest them in block elements, some browsers don't handle this breach of (X)HTML syntax very well — these, for example:

- Text-only browsers for Palm devices and mobile phones
- Text-to-speech readers for the visually impaired

Text-based browsers rely on block elements to properly divide the sections of your page. Without a block element, these browsers might display your links in the wrong places.

TIP

Link locations, captions, and destinations have a big impact on how site visitors perceive links. Chapter 3 covers best practices for using links in your site design.

The kind of link you create is determined by where you link.

Absolute links

An *absolute link* uses a complete URL to connect browsers to a Web page or online resource.

Links that use a complete URL to point to a resource are called *absolute* because they provide a complete, stand-alone path to another Web resource. When you link to a page on someone else's Web site, the Web browser needs every bit of information in the URL to find the page. The browser starts with the domain in the URL and works its way through the path to a specific file.

REMEMBER

When you link to files on someone else's site, you must always use absolute URLs in the `href` attribute of the anchor element. Here's an example:

```
http://www.website.com/directory/page.html
```

Relative links

A *relative link* uses a kind of shorthand to specify the URL for the resource where you're pointing.

Use the following guidelines with relative links in your (X)HTML pages:

- You create relative links between resources in the same domain.
- Because both resources are in the same domain, you can omit domain information from the URL.

  A *relative* URL uses the location of the resource you're linking from to identify the location of the resource you're linking to (for example, `page.html`).

A relative link is similar to telling someone that he or she needs to go to the Eastside Mall. If the person already knows where the Eastside Mall is, he or she doesn't need additional directions. Web browsers behave the same way.

If you use relative links on your site, your links still work if you change either

- Servers
- Domain names

Simple links

You can take advantage of relative URLs when you create a link between pages on the same Web site. If you want to make a link from `http://www.mysite.com/home.html` to `http://www.mysite.com/about.html`, you can use this simplified, relative URL in an anchor element on `home.html`:

```
<p>Learn more <a href="about.html">about</a> our company.</p>
```

When a browser sees a link without a domain name, the browser assumes the link is *relative* — and uses the domain and path from the linking page to find the linked page.

Site links

As your site grows more complex and you organize your files into various folders, you can still use relative links. But you must provide additional information in the URL to help the browser find files that don't reside in the same directory as the file from which you're linking.

Use `../` (two periods and a slash) before the filename to indicate that the browser should move up one level in the directory structure.

The markup for this process looks like this:

```
<a href="../docs/home.html>Documentation home</a>
```

The notation in this anchor element instructs the browser to:

1. **Move up one folder from the folder the linking document is stored in.**
2. **Find a folder called `docs`.**
3. **Find a file called** `home.html`**.**

When you create a relative link, the location of the file *to* which you link is always relative to the file *from* which you link. As you create a relative URL, trace the path a browser takes if it starts on the page you're linking from to get to the page to which you're linking. That path defines the URL you need.

Common mistakes

Every Web resource — whether it's a site, page, or image — has a unique URL. Even one incorrect letter in your URL can lead to a *broken link*. Broken links lead to an error page (often the HTTP error `404 File or directory not found`).

URLs are so finicky that a simple typo breaks a link.

If a URL doesn't work, try these tactics:

- **Check the capitalization.** Some Web servers (Linux and UNIX most notably) are *case-sensitive* (meaning they distinguish between capital and lowercase letters). These servers treat the filenames `Bios.html` and `bios.html` as different files on the Web server. That means any browser looking for a particular URL *must* use uppercase and lowercase letters when necessary. Be sure the capitalization in the link matches the capitalization of the URL.

  To avoid problems with files on your Web site, follow a standard naming convention. Often, using only lowercase letters can simplify your life.

- **Check the extension.** `Bios.htm` and `Bios.html` are two different files. If your link's URL uses one extension and the actual filename uses another, your link won't work.

  To avoid problems with extensions on your Web site, pick either `.html` or `.htm` *and stick to that extension.*

- **Check the filename.** `bio.html` and `bios.html` are two different files.

- **Cut and paste.** Avoid retyping a URL if you can copy it. The best and most foolproof way to create a URL that works is as follows:

  1. **Load a page in your browser.**
  2. **Copy the URL from the browser's address or link text box.**
  3. **Paste the URL into your (X)HTML markup.**

The importance of http:// in (X)HTML links

Browsers make surfing the Web as easy as possible. If you type **www.sun.com**, **sun.com**, or often even just **sun**, in your browser's address window, the browser obligingly brings up `http://www.sun.com`. Although this technique works when you type URLs into your browser window, it doesn't work when you're writing markup.

The URLs that you use in your HTML markup must be fully formed. Browsers won't interpret URLs that don't include the page protocol. If you forget the `http://`, your link may not work!

TIP

The cut and paste method for grabbing URLs presumes you're grabbing them from a Web site somewhere. If you open a local file on your PC in a browser, you'll see something that looks like this: `file:\\\I:\H4D6e\html_letter.html`. The `file:\\\` is an Internet Explorer convention used to identify the document as a file in your local file system, `I:\` is a drive letter, the `H4D6e\` is a folder or directory on that drive, and the rightmost text element (`html_letter.html` in this case) is the name of the HTML file you've opened. You can't use URLs like this on a Web site, so please — don't try to!

Customizing Links

You can customize links to

- Open linked documents in new windows
- Link to specific locations *within* a Web page of your own
- Link to items other than (X)HTML pages, such as
    - Portable Document Format (PDF) files
    - Compressed files
    - Word-processing documents

New windows

The Web works because you can link pages on your Web site to pages on other people's Web sites by using a simple anchor element. But when you link to someone else's site, you send users away from your own site.

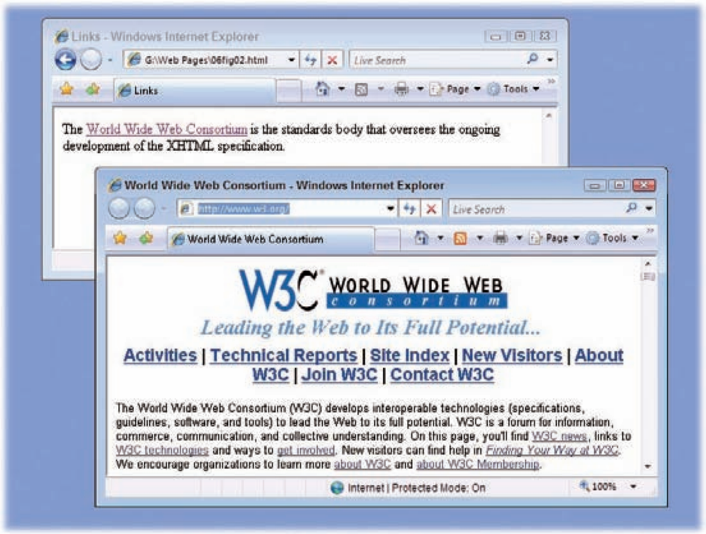
To keep users on your site, HTML can open the linked page in a new window. The simple addition of the `target` attribute to an anchor element opens that link in a new browser window instead of opening it in the current window:

```
<p>The <a href="http://www.w3.org" target="_blank">World Wide Web Consortium</a>
is the standards body that oversees the ongoing development of the XHTML
specification.</p>
```

When you give a `target` attribute a `_blank` value, this tells the browser to

1. **Keep the linking page open in the current window.**
2. **Open the linked page in a new window.**

The result of the `target="_blank"` attribute is shown in Figure 6-2.


Links - Windows Internet Explorer
G:\Web Pages\06fig02.html
Live Search
Links
Page
Tools
World Wide Web Consortium - Windows Internet Explorer
http://www.w3.org/
Live Search
World Wide Web Consortium
Page
Tools
W3C WORLD WIDE WEB consortium
Leading the Web to Its Full Potential...
Activities | Technical Reports | Site Index | New Visitors | About W3C | Join W3C | Contact W3C
Internet | Protected Mode: On
100%

The World Wide Web Consortium is the standards body that oversees the ongoing development of the XHTML specification.

The World Wide Web Consortium (W3C) develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential. W3C is a forum for information, commerce, communication, and collective understanding. On this page, you'll find W3C news, links to W3C technologies and ways to get involved. New visitors can find help in *Finding Your Way at W3C*. We encourage organizations to learn more about W3C and about W3C Membership.

Figure 6-2: Use the target attribute to open a new window for a linked file.

WARNING!

Pop-up windows irritate some users.

You can use JavaScript to control the size, location, and appearance of pop-up windows, as well as put buttons on them to help users close them quickly. Chapter 12 covers pop-up windows in more detail — including JavaScript details.

Locations in Web pages

Locations within Web pages can be marked for direct access by links on

- The same page
- The same Web site
- Other Web sites

Keep these considerations in mind when adding links to Web pages:

- Several short pages may present information more conveniently for readers than one long page with internal links.

  Links within large pages work nicely for quick access to directories, tables of contents, and glossaries.
- *Intradocument* linking works best on your own Web site, where you can create and control the markup.

  When you link to spots on someone else's Web site, you're at its manager's mercy. That person controls linkable spots. Your links will break if the site designer removes or renames any spot to which you link.

Naming link locations

To identify and create a location within a page for direct access from other links, use an empty anchor element with the name attribute, like this:

```
<a name="top"></a>
```

The anchor element that marks the spot doesn't affect the appearance of any surrounding content. You can mark spots wherever you need them without worrying about how your pages look (or change) as a result.

Linking to named locations

As we mention earlier, you can mark locations for direct access by links

- Within the same page
- Within the same Web site
- On other Web sites

Within the same page

Links can help users navigate a single Web page. Intradocument hyperlinks include such familiar features as

- Back to Top links
- Tables of contents

An intradocument hyperlink, also known as a named document link, uses a URL like this:

```
<a href="#top">Back to top</a>
```

The pound sign (#) indicates that you're pointing to a spot on the same page, not on another page.

Listing 6-1 shows how two anchor elements combine to link to a spot on the same page. (Documents that use intradocument links are usually longer. This document is short so you can easily see how to use the `top` anchor element.)

Listing 6-1: Intradocument Hyperlinks

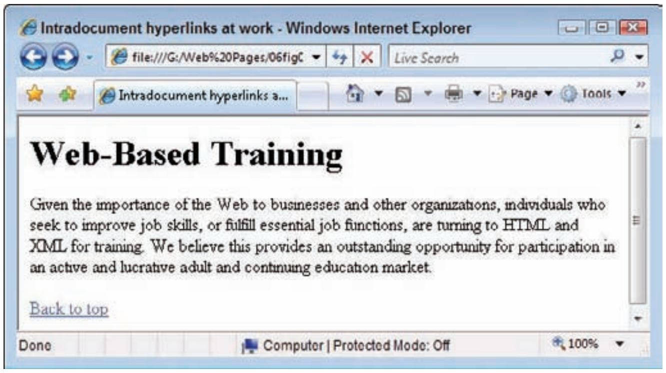
```
<html>
  <head>
    <title>Intradocument hyperlinks at work</title>
  </head>

  <body>
    <h1><a name="top"></a>Web-Based Training</h1>

    <p>Given the importance of the Web to businesses and
       other organizations, individuals who seek to improve
       job skills, or fulfill essential job functions, are
       turning to HTML and XML for training. We believe
       this provides an outstanding opportunity for
       participation in an active and lucrative adult and
       continuing education market.</p>

    <p><a href="#top">Back to top</a></p>

  </body>
</html>
```

Figure 6-3 shows how this HTML markup appears in a Web browser. If the user clicks the Back to Top link, the browser jumps back to the `top` spot — marked by `<a name="top"></a>`.

Within the same Web site

You can combine intradocument and interdocument links to send visitors to a spot on a different Web page on your site. Thus, to link to a spot named `descriptions` on a page named `home.html` on your site, use this markup:

```
<p>Review the <a href="home.html#descriptions">document descriptions</a>
   to find the documentation for your particular product.</p>
```


Intradocument hyperlinks at work - Windows Internet Explorer
file:///G:/Web%20Pages/06figC
Live Search
Intradocument hyperlinks a...
Page
Tools

Web-Based Training

Given the importance of the Web to businesses and other organizations, individuals who seek to improve job skills, or fulfill essential job functions, are turning to HTML and XML for training. We believe this provides an outstanding opportunity for participation in an active and lucrative adult and continuing education market.

Back to top

Done | Computer | Protected Mode: Off | 100%

Figure 6-3: Use anchor elements to mark and link spots on a page.

On other Web sites

If you know that a page on another site has spots marked, you can use an absolute URL to point to a particular spot on that page, like this:

```
<p>Find out how to
<a href="http://www.yourcompany.com/training/online.htm#register">
register</a> for upcoming training courses led by our instructors.</p>
```

WARNING!

Be sure to check all links regularly to catch and fix the broken ones.

The Open Directory Project provides a laundry list of free and commercial tools you can use to make finding and fixing broken links easier:

```
http://dmoz.org/Computers/Software/Internet/Site_Management/Link_Management/
```

Non-HTML resources

Links can connect to virtually any kind of file, such as

- Word-processing documents
- Spreadsheets
- PDFs
- Compressed files
- Multimedia

A great use for non-HTML links is for software and PDF download pages.

File downloads

Non-Web files must nevertheless be accessed via the Internet, so they possess unique URLs, just like HTML pages. Any file on a Web server (regardless of its type) can be linked using a URL.

For instance, if you want your users to download a PDF file named `doc.pdf` and a `.zip` archive called `software.zip` from a Web page, you use this HTML:

```
<h1>Download the new version of our software</h1>
<p><a href="software.zip">Software</a></p>
<p><a href="doc.pdf">Documentation</a></p>
```

You can't know how any user's browser will respond to a click on a link that leads to a non-Web file. The browser may

- Prompt the user to save the file
- Display the file without downloading it (this is common for PDFs)
- Display an error message (if the browser can't handle or doesn't recognize the type of file involved)

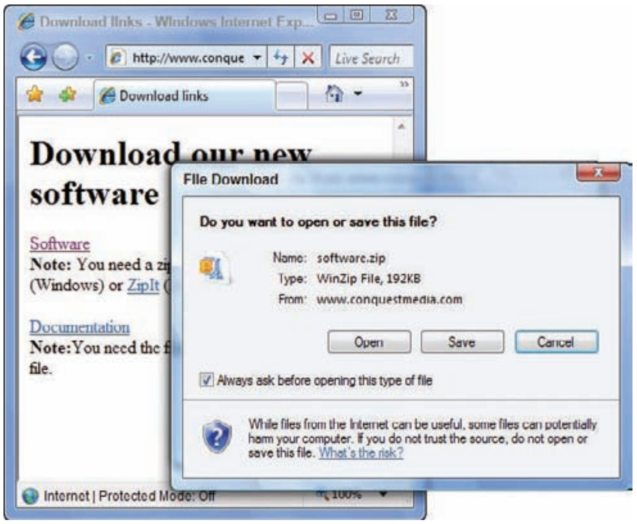
To help users download files successfully, you should provide your users with

- As much information as possible about the file formats in use.
- Any special tools they need to work with the files.
    - To work with the contents of a Zip file, the users need a compression utility, such as WinZip or ZipIt, if their operating systems do not natively support Zip files.
    - To view a PDF file, users need the free Adobe Acrobat Reader.

You can make download markup more user-friendly by adding supporting text and links, like this:

```
<h1>Download our new software</h1>
<p> <a href="software.zip">Software</a> <br />
   <b>Note:</b>
    You need a zip utility such as
<a href="http://www.winzip.com">WinZip</a> (Windows) or
<a href="http://www.maczipit.com">ZipIt</a> (Macintosh)
   to open this file.</p>
<p><a href="doc.pdf">Documentation</a> <br />
   <b>Note:</b>You need the free
<a href="http://www.adobe.com/products/
acrobat/readstep2.html">Adobe Reader</a>
   to view this file.</p>
```

Figure 6-4 shows how a browser renders this HTML — and the dialog box it displays when you click the software link.


Download links - Windows Internet Exp...
http://www.conque
Live Search
Download links
Open
Save
Cancel
Internet | Protected Mode: Off
100%

Download our new software

Software
Note: You need a zi
(Windows) or ZipIt (

Documentation
Note:You need the f
file.

File Download

Do you want to open or save this file?

Name: software.zip
Type: WinZip File, 192KB
From: www.conquestmedia.com

☑ Always ask before opening this type of file

While files from the Internet can be useful, some files can potentially harm your computer. If you do not trust the source, do not open or save this file. What's the risk?

Figure 6-4: This browser prompts you to save or view the Zip file.

E-mail addresses

A link to an e-mail address can automatically open a new e-mail addressed to exactly the right person.

This is a great way to help users send you e-mail with comments and requests.

An e-mail link uses the standard anchor element and `href` attribute. The value of the `href` attribute is the receiving e-mail address prefaced with `mailto:`

```
<p>Send us your
  <a href="mailto:comments@mysite.com">comments</a>.</p>
```

The user's browser configuration controls how the browser handles an e-mail link. Most browsers follow these two basic steps automatically:

1. Open a new message window in the default e-mail program.
2. Insert the address from the `href` attribute into the To field of the message.

Unfortunately, Web page `mailto:` links are a prime source of e-mail addresses for spammers. Creating a form to receive feedback is often a better idea; better still, use JavaScript encryption on the e-mail address (for more info, see Steven Chapman's great article "Hiding Your Email Address" at `http://javascript.about.com/library/blemail1.htm`). We generally tend to provide our e-mail addresses in the form: `jeff at conquestmedia dot com`, knowing that people are smart enough to substitute `@` for `at` and `.` for `dot`, but that address-harvesters usually aren't that canny.

Chapter 7

Finding and Using Images

In This Chapter

- Determining the right format for your images
- Adding images to Web pages
- Creating images and image maps that trigger links

Web-page designers use images to deliver important information, direct site navigation, and contribute to the overall look and feel of a Web page. But you have to use images properly or you risk reducing their effectiveness.

When used well, images are a key element of page design. When used poorly, they can make a page unreadable or inaccessible.

This chapter is a crash course in using images on Web pages. You find out which image formats are Web-friendly and how to use (X)HTML elements to add images to your Web pages. You also discover how to attach links to your images and how to create image maps for your Web page.

The Role of Images in a Web Page

Images in Web sites may be logos, clickable navigation aids, or display content; they may also make a page look prettier, or serve to unify or illustrate a page's theme. A perfect example of the many different ways images can enhance and contribute to Web pages is the White House home page at `www.whitehouse.gov`, shown in Figure 7-1, where you see the Presidential seal and photos used to good effect.

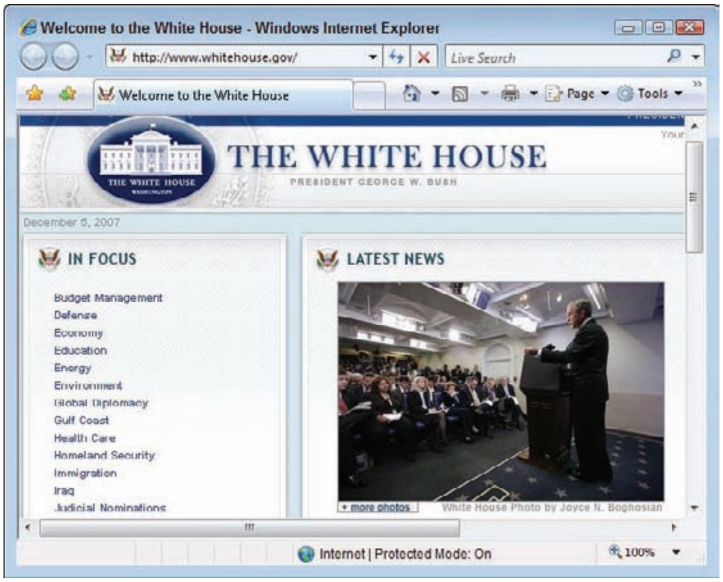


Figure 7-1: The White House Web page uses images in a variety of ways.

Creating Web-Friendly Images

You can create and save graphics in many ways, but only a few formats are actually appropriate for images that you intend to use on the Web. As you create Web-friendly images, you must account for file formats and sizes.

Often, graphics file formats are specific to operating systems or software applications. But you can't predict a visitor's computer and software (other than a Web browser). So you need images that anyone can view with any browser. This means you need to use *cross-platform* file formats that users can view with any version of Microsoft Windows, the Mac OS, or Linux.

Optimizing images

As you build graphics for your Web page, maintain a healthy balance between file quality and size. Webmonkey has two good tutorials on trimming image file sizes and optimizing entire sites for fast download. For tips and tricks that can help you build pages that download quickly, review these handy resources:

- Optimizing Images

  `http://www.yourhtmlsource.com/optimisation/image optimisation.html`

- Optimizing Web Graphics

  `http://www.websiteoptimization.com/speed/12`

Only these three compressed formats are suitable for general use on the Web:

- **Graphics Interchange Format (GIF):** Images saved as GIFs often are smaller than those saved in other file formats. GIF supports up to 256 colors only, so if you try to save an image created with millions of colors as a GIF, you lose image quality. GIF is the best format for less-complex, nonphotographic images, such as line art and clip art.
- **Joint Photographic Experts Group (JPEG):** The JPEG file format supports 24-bit color (millions of colors) and complex images, such as photographs. JPEG is cross-platform and application-independent. A good image-editing tool can help you tweak the compression so you can strike an optimum balance between image quality and image-file size.
- **Portable Network Graphics (PNG):** PNG is the latest cross-platform and application-independent image file format. It was developed to bring together the best of GIF and JPEG. PNG has the same compression as GIF but supports 24-bit color (and even 32-bit color) like that of JPEG.

Any good graphics-editing tool, such as those mentioned in Chapter 22, lets you save images in any of these formats. Experiment with them to see how converting a graphic from one format to another changes its appearance and file size, then choose whichever format produces the best results.

Table 7-1 shows guidelines for choosing a file format for images by type.

Table 7-1 Choosing the Right File Format

File Format	*Best Used For*	*Watch Out*
GIF	Line art and other images with few colors and less detail.	Don't use this format if you have a complex image or photo.
JPEG	Photos and other images with millions of colors and lots of detail.	Don't use with line art. This format compromises too much quality when you compress the file.
PNG	Photos and other images with millions of colors and lots of detail.	Don't use with line art. PNG offers the best balance between quality and file size.

TIP

For a complete overview of graphics formats, visit

- Builder.com's "Examine Graphic Channels and Space"

  `http://builder.cnet.com/webbuilding/0-3883-8-4892140-1.html`
- Webmonkey's "Web Graphics Overview"

  `http://www.webmonkey.com/01/28/index1a.html`

Adding an Image to a Web Page

When an image is ready for the Web, you need to use the correct markup to add it to your page. But you need to know where to store your image as well.

Image location

You can store images for your Web site in several places. Image storage works best if it uses *relative* URLs (stored somewhere on the Web site with your other (X)HTML files). You can store images in the same root file as your (X)HTML files, which gets confusing if you have a lot of files, or you can create a `graphics` or `images` directory in the root file of your Web site.

Relative links connect resources from the same Web site. You use absolute links between resources on two different Web sites. Turn to Chapter 6 for a complete discussion of the differences between relative and absolute links.

Three compelling reasons to store images on your own site are

- **Control:** When the images are stored on your site, you have complete control over them. You know that your images aren't going to disappear or change, and you can work to optimize them.
- **Speed:** If you link to images on someone else's site, you never know when that site may go down or be unbelievably slow. Linking to images on someone else's site also causes the other site owner to pay for the bandwidth required to display it on your site.
- **Copyright:** If you link to images on another Web site to display them on your site, you may violate copyright law. (In this case, obtain permission from the copyright holder to store and display the images on your site.)

Using the <img /> element

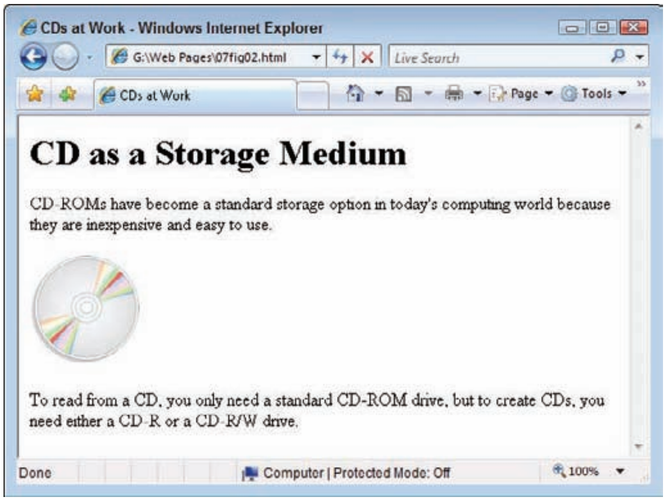
The image (`<img />`) element is an *empty element* (sometimes called a *singleton tag*) that you place on the page where you want your image to go.

An empty element has only one tag, with neither a distinct opening nor closing tag.

The following markup places an image named `07fg02-cd.jpg`, which is saved in the same directory as the (X)HTML file, between two paragraphs:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
   <title>CDs at Work</title>
</head>
  <body>
  <h1>CD as a Storage Medium</h1>
  <p>CD-ROMs have become a standard storage option in today's computing world
     because they are inexpensive and easy to use.</p>
  <img src="07fg02-cd.jpg" />
  <p>To read from a CD, you only need a standard CD-ROM drive, but to create
     CDs, you need either a CD-R or a CD-R/W drive.</p>
  </body>
</html>
```

A Web browser replaces the `<img />` element with the image file provided as the value for the `src` attribute, as shown in Figure 7-2.


CDs at Work - Windows Internet Explorer
G:\Web Pages\07fig02.html
Live Search
CDs at Work
Page
Tools
Done
Computer | Protected Mode: Off
100%

CD as a Storage Medium

CD-ROMs have become a standard storage option in today's computing world because they are inexpensive and easy to use.

To read from a CD, you only need a standard CD-ROM drive, but to create CDs, you need either a CD-R or a CD-R/W drive.

Figure 7-2: Use the <img /> element to place graphics in a Web page.

The `src` attribute is like the `href` attribute that you use with an anchor (`<a>`) element. The `src` attribute specifies the location for the image you want to display on your page. The preceding example points to an image file in the same folder as the HTML file referencing it.

Adding alternative text

Alternative text describes an image so those who can't see the images for some reason can access that text to learn more about the image. Adding alternative text (often referred to by HTMLers as "alt text") is a good practice because it accounts for

- Visually impaired users who may not be able to see images and must rely on alternative text for a text-to-speech reader to read to them.
- Users who access the Web site from a phone browser with limited graphics capabilities.
- Users with slow modem connections who don't display images.

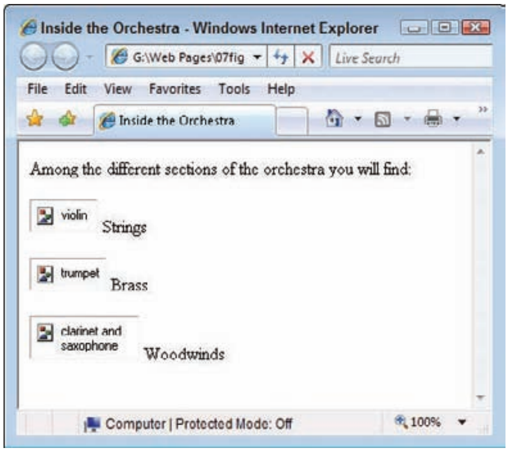
TECHNICAL STUFF

Some search engines and cataloguing tools use alternative text to index images.

Most of your users will see your images, but be prepared for those who won't. The (X)HTML specifications require that you provide alternative text to describe each image on a Web page. Use the `alt` attribute with the `<img />` element to add this information to your markup, like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
  <title>Inside the Orchestra</title>
</head>

<body>
  <p>Among the different sections of the orchestra you will find:</p>
  <p><img src="07fg03-violin.jpg" alt="violin " /> Strings</p>
  <p><img src="07fg03-trumpet.jpg" alt="trumpet" /> Brass</p>
  <p><img src="07fg03-woodwinds.jpg" alt="clarinet and saxophone" />
      Woodwinds</p>
</body>
</html>
```

When browsers don't display an image (or can't, as in text-only browsers such as Lynx), they display the alternative text instead, as shown in Figure 7-3. (We turned pictures off in IE to produce the screenshot.)


Inside the Orchestra - Windows Internet Explorer
G:\Web Pages\07fig
Live Search
File Edit View Favorites Tools Help
Inside the Orchestra
violin
trumpet
clarinet and saxophone
Computer | Protected Mode: Off
100%

Among the different sections of the orchestra you will find:

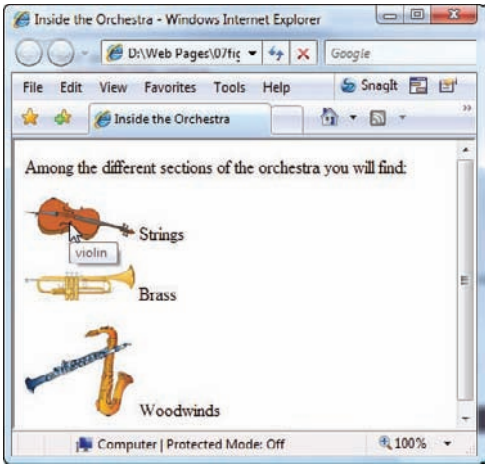
Strings

Brass

Woodwinds

Figure 7-3: When a browser doesn't show an image, it shows alternative text.

When browsers show an image, some browsers — including Internet Explorer, Netscape, and Opera — show alternative text as pop-up tooltips when you hover your mouse pointer over an image for a few seconds, as shown in Figure 7-4. Firefox, however, does not.


Inside the Orchestra - Windows Internet Explorer
D:\Web Pages\07fig
Google
File Edit View Favorites Tools Help
SnagIt
Inside the Orchestra
violin
Computer | Protected Mode: Off
100%

Among the different sections of the orchestra you will find:

Strings

Brass

Woodwinds

Figure 7-4: A browser may display alternative text as a pop-up tip.

This means you can use alternative text either to describe the image to those who can't see it or to provide useful (or amusing) information about the image.

TIP

The W3C's Web Accessibility Initiative (WAI) includes helpful tips for creating useful and usable alternatives to visual content at

```
www.w3.org/TR/WCAG10-TECHS/#gl-provide-equivalents
```

Specifying image size

You can use the height and width attributes with the <img /> element to let the browser know just how tall and wide an image is (in pixels):

```
<p><img src="07fg03-trumpet.jpg" width="50" height="70" alt="trumpet"
               />Brass</p>
```

Most browsers download the HTML and text associated with a page before they download the page graphics. Instead of making users wait for the whole page to download, browsers typically display the text first and then fill in graphics as they become available. If you tell the browser how big a graphic is, the browser can reserve a spot for it in the page display. This smooths the process of adding graphics to the Web page.

TIP

You can check the width and height of an image in pixels in any image-editing program, or in the image viewers built into Windows and the Mac OS. (You might be able simply to view the properties of the image in either Windows or the Mac OS to see its height and width.)

Another good use of the height and width attributes is to create colored lines on a page by using just a small colored square. For example, this markup adds a 10x10-pixel blue box to a Web page:

```
<img src="07fg05-blue-box.gif" alt="blue box" height="10" width="10" />
```

Use the <img /> element height and width attributes to set image height and width. Thus we use these values to create 10x10 pixel blue box in a browser window (shown in Figure 7-5), even though the original image is 100x100 pixels in size. In general it's safe to reduce image dimensions using these attributes, though you'll always want to check the results carefully during testing (and with any kind of aspect sensitive image, you'll want to maintain its aspect ratio by dividing the original dimensions by some common value, as we did in going from 100x100 to 10x10, by dividing by 10).

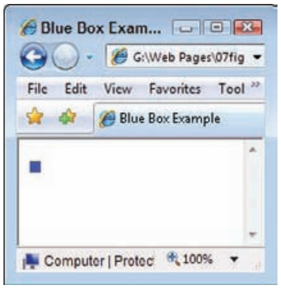


Figure 7-5: A small box.

However, a change to the values for `height` and `width` in the markup turns this small blue box into a line 20 pixels high and 200 pixels wide:

```
<img src="07fg05-blue-box.gif" alt="blue box" height="20" width="200" />
```

The browser expands the image to fit the height and width specifics in the markup, as shown in Figure 7-6.

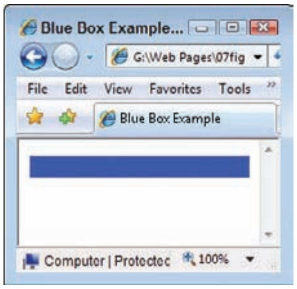


Figure 7-6: A small box becomes a short line.

Using this technique, you can turn a single image like the blue box (only 1K in size) into a variety of lines — and even boxes:

- This can ensure that all dividers and other border elements on your page use the same color — they're all based on the same graphic.
- If you decide you want to change all your blue lines to green, you just change the image. Every line you've created changes colors.

When you specify a height and width for an image that are different from the image's actual height and width, you rely on the browser to scale the image display. This works great for single-color images such as the blue box, but it doesn't work well for images with multiple colors or images that display actual pictures. The browser doesn't size images well, and you wind up with a distorted picture. Figure 7-7 shows how badly a browser handles enlarging a trumpet image when the markup doubles the image height and width (note the jaggies on the trumpet bell, for example):

```
<p><img src="07fg03-trumpet.jpg" width="200" height="64" alt="trumpet" />
               Brass</p>
```

If you need several sizes for the same image — as for a logo or navigation button — use a large image as the master for that graphic, and make smaller versions; doing so gives you better control over the final look and feel of each image.

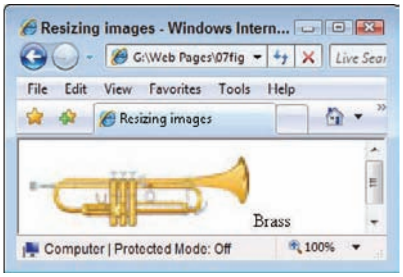

Resizing images - Windows Intern...
G:\Web Pages\07fig
Live Sear
File Edit View Favorites Tools Help
Resizing images
Brass
Computer | Protected Mode: Off
100%

Figure 7-7: Don't use a browser to resize complex images.

Setting an image border

By default, every image has a border of 1 — which doesn't show up in most browsers until you turn that image into a link (as shown in the "Images That Link" section later). You can use the `border` attribute with the `<img />` element to better control the border the browser displays around your image. This markup sets the border for the clarinet image to 10 pixels:

```
<img src="07fg03-woodwinds.jpg" width="100" height="83" alt="clarinet and
              saxophone" border="10" />
```

The browser uses this border on all four sides of the image, as in Figure 7-8.

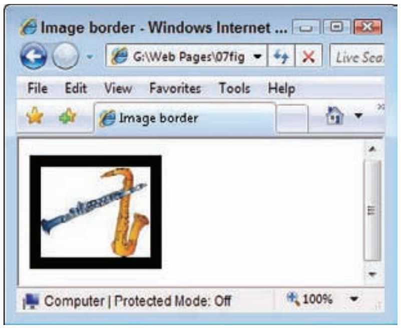


Image border - Windows Internet ...
G:\Web Pages\07fig
Live Sea
File Edit View Favorites Tools Help
Image border
Computer | Protected Mode: Off
100%

Figure 7-8: Use the border attribute to create a border around your image.

In Figure 7-8, the border is black and applies to all four sides of the image. If you want to control the color of the border or make the border appear differently on each side of the image, you have two options:

- Build the border into the image in an image-editing tool.
- Use Cascading Style Sheets (CSS), which we cover in Chapters 8 and 9.

If you use an image-editing tool to create your border, use its features to create a patterned border or apply a unique effect. However, the extra information in the image may make it bigger. Carefully balance your image size and its appearance so it doesn't take too long to download.

If you use CSS to apply a border, your image won't get any bigger, but your border may not show up in older browsers that don't support CSS well. The choice you make depends on how crucial the border is to your image (if it's very important, embed it in the image) and what browser you think your visitors use (newer browsers have better support for style sheets).

Controlling image alignment

The `align` attribute works with the `<img />` element to control how your image appears relative to the text around it. The possible values for this attribute are

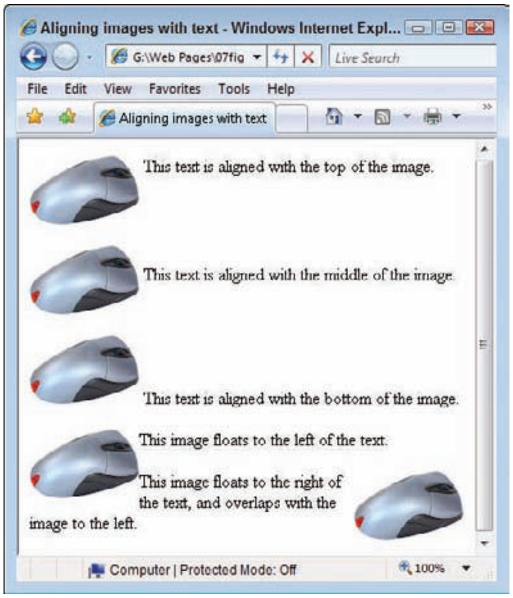
- `top`: Aligns the text around the image with the top of the image.
- `middle`: Aligns the text around the image with the middle of the image.
- `bottom`: Aligns the text around the image with the bottom of the image.
- `left`: The image sits on the left, and text floats to the right of the image.
- `right`: The image sits on the right, and text floats to the left of the image.

By default, most browsers align images to the left and float all text to the right. The following markup shows how five different `<img />` elements use the `align` attribute to change how text floats around an image — in this case, the `07fg09-mouse.jpg` image:

```
<p> <img src="07fg09-mouse.jpg" alt="mouse with top-aligned text"
         height="63" width="100" align="top" />
     This text is aligned with the top of the image.
</p>

<p> <img src="07fg09-mouse.jpg" alt="mouse with middle-aligned text"
         height="63" width="100" align="middle" />
    This text is aligned with the middle of the image.
</p>

<p> <img src="07fg09-mouse.jpg" alt="mouse with bottom-aligned text"
         height="63" width="100" align="bottom" />
    This text is aligned with the bottom of the image.
</p>

<p> <img src="07fg09-mouse.jpg" alt="mouse with left-aligned text"
         height="63" width="100" align="left" />
     This image floats to the left of the text.
</p>
```

```
<p> <img src="07fg09-mouse.jpg" alt="mouse with right-aligned text"
        height="63" width="100" align="right" />
     This image floats to the right of the text, and overlaps with
     the image to the left.
</p>
```

Figure 7-9 shows how a browser interprets different alignment attributes.


Aligning images with text - Windows Internet Expl...
G:\Web Pages\07fig
Live Search
File Edit View Favorites Tools Help
Aligning images with text
Computer | Protected Mode: Off
100%

This text is aligned with the top of the image.

This text is aligned with the middle of the image.

This text is aligned with the bottom of the image.

This image floats to the left of the text.

This image floats to the right of the text, and overlaps with the image to the left.

Figure 7-9: You can vary image alignment to control image placement on the page.

TIP

The `<img />` attributes may not give you all the control you want of image alignment. Chapter 11 shows how tables and images can be used together to fine-tune image alignments, while Chapter 9 shows how you can use CSS properties to better control how images sit on the page.

Setting image spacing

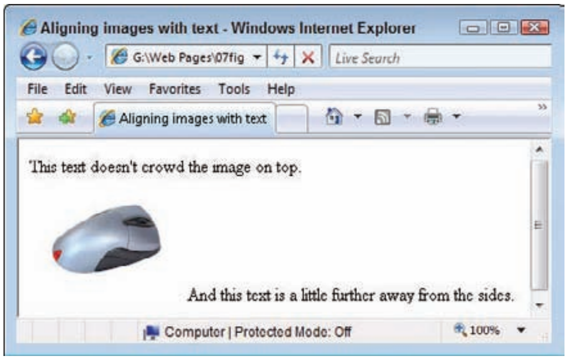
Most browsers leave about one pixel of white space between images and text or other images next to them. You can give your images breathing room with

- The `vspace` *(vertical space)* attribute for top and bottom
- The `hspace` *(horizontal space)* attribute for left and right

The following HTML gives the mouse graphic 20 pixels of white space on either side and 25 pixels on the top and bottom:

```
<p>
  This text doesn't crowd the image on top.<br />
  <img src="07fg09-mouse.jpg"
       height="63" width="100" hspace="20" vspace="25"
       alt="mouse on a white background" />
  And this text is a little further away from the sides. </p>
```

Figure 7-10 shows how a browser adds space around the image.


Aligning images with text - Windows Internet Explorer
G:\Web Pages\07fig
Live Search
File Edit View Favorites Tools Help
Aligning images with text
Computer | Protected Mode: Off
100%

This text doesn't crowd the image on top.

And this text is a little further away from the sides.

Figure 7-10: The `hspace` and `vspace` attributes control the white space around an image.

The default value for `hspace` and `vspace` is `1`. If you want images so close together that their sides touch (like for a set of navigation buttons), set the value for these attributes to `0` to eliminate that extra 1 pixel of space.

You can use CSS to position images on a page. You can position images with accuracy and with control over placement, spacing, white space, and how text flows around the graphic. Chapter 9 has the details on positioning items.

Images That Link

Web pages often use images for navigation. They're prettier than plain-text links, and you can add both form and function on your page with one element.

Triggering links

To create an image that triggers a link, you substitute an `<img />` element in place of text to which you would anchor your link. This markup links text:

```
<p><a href="http://www.w3.org">Visit the W3C</a></p>
```

This markup replaces the text `Visit the W3C` with an appropriate icon:

```
<p><a href="http://www.w3.org"><img src="w3.jpg"
     alt="Visit the W3C Web Site" height="48" width="315" border="1" /></a>

</p>
```

The preceding markup creates a linked image to `http://www.w3.org`. In the preceding example, the alternative text now reads `Visit the W3C Web Site` so users who can't see the image know where the link goes. When a user moves the mouse pointer over the image, the cursor changes from an arrow into a pointing hand (or any icon the browser uses for a link).

We include a border around this image as a visual cue to let users know it also serves as a link; it appears as a light blue outline (as shown in Figure 7-11) until the link is followed. After that, the blue outline turns purple to let users know this link has been visited.

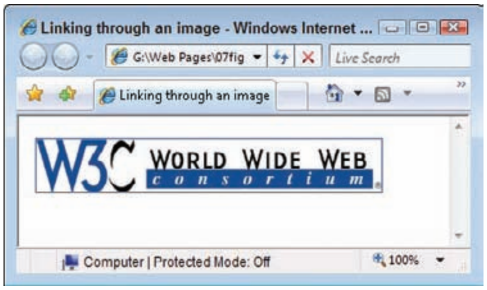

Linking through an image - Windows Internet ...
G:\Web Pages\07fig
Live Search
Linking through an image
W3C WORLD WIDE WEB consortium
Computer | Protected Mode: Off
100%

Figure 7-11: Combine image and anchor elements to create a linked image.

A quick click of the image launches the W3C Web site. It's as simple as that.

TRICKS OF THE TRADE

As shown earlier in the chapter, you should set the border of any image you use in a link to `0` if you want to keep the browser from surrounding your image with a blue line. Without the line, however, users need other visual (or alternative text) clues so they know an image is a link. Be sure images that serve as links scream to the user (tastefully, of course) "I'm a link!" In all cases, if the automatic outline is eliminated, you should build an outline into the graphic itself or add a caption that indicates the image serves as a link.

Building image maps

When you use an `<img />` element with an anchor element to create a linking image, you can attach only one link to that image. To create a larger image that connects links to different regions on the page, you need an *image map.*

To create an image map, you need two things:

- An image with distinct areas obvious to users. (For example, an image of a park might show a playground, a picnic area, and a pond area.)
- Markup to map the different regions on the map to different URLs.

Elements and attributes

Use the `<img />` element to add the map image into your page, just as you would any other image. In addition, include the `usemap` attribute to let the browser know that there's image map information to go with that image. The value of the `usemap` attribute is the name of your map.

You use two elements and a collection of attributes to define the image map:

- `<map>` holds the map information. The `<map>` element uses the `name` attribute to identify the map. The value of `name` should match the value of `usemap` in the `<img />` element that goes with the map.
- `<area />` links specific parts of the map to URLs. The `<area />` element takes these attributes to define the specifics for each section of the map:
  - `shape`: Specifies the shape of the region (a clickable hot spot that makes the image map work). You can choose from `rect` (rectangle), `circle`, and `poly` (a triangle or polygon).
  - `coords`: Define the region's coordinates. A *rectangle's* coordinates include the *left, right, top,* and *bottom points.* A *circle's* coordinates include the *x and y coordinates* for the center of the circle as well as the circle's *radius.* A polygon's coordinates are a collection of *x and y coordinates* for every vertex in the polygon. (You can use an image map editor like Mapedit from `www.boutell.com/mapedit`, or a graphics editor such as PaintShop Pro from `www.corel.com`, to determine image coordinates; Mapedit also records them for you.)
  - `href`: Specifies the URL to which the region links (can be absolute or relative).
  - `alt`: Provides alternative text for the image region.

Markup

This defines a three-region map called `NavMap` linked to the `navigation.gif`:

```
<img src="07fg12-navmap.gif" width="302" height="30" usemap="#NavMap" border="0"
               />
<map name="NavMap" />
  <area shape="rect" coords="0,0,99,30" href="home.html" alt="Home" />
  <area shape="rect" coords="102,0,202,30" href="about.html" alt="About" />
  <area shape="rect" coords="202,0,301,30" href="products.html"
        alt="Products" />
</map>
```

Figure 7-12 shows how a browser displays this markup.

TIP

When the mouse sits over a region in the map, the cursor turns into a pointing hand (just as it changes over any other hyperlink). So take advantage of the alternative text to include useful information about the link.

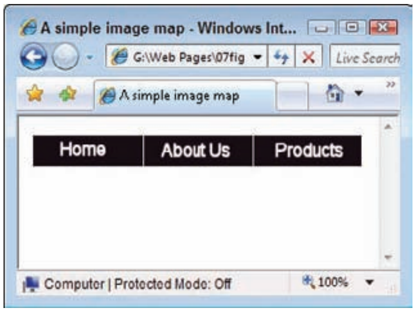


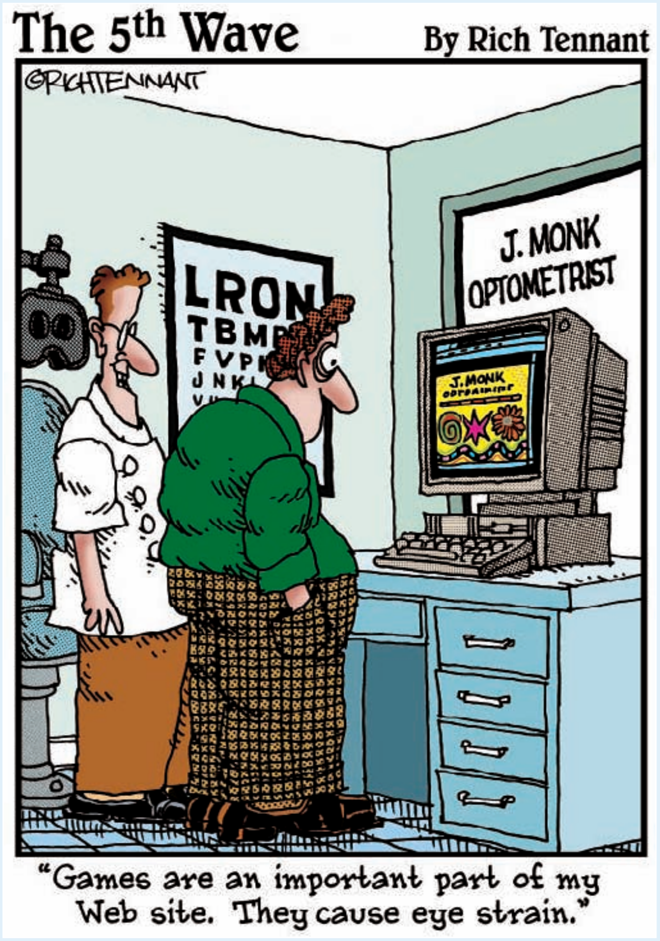
Figure 7-12: Image maps turn different areas of an image into linking regions.

TRICKS OF THE TRADE

A common use for image maps is to turn maps of places (states, countries, cities, and such) into linkable maps. About.com's image map tutorial at `http://webdesign.about.com/od/imagemaps/a/aabg051899a.htm` provides more details on building image maps by hand. HTML Goodies has a great collection of image map tutorials and information at `www.htmlgoodies.com/tutorials/image_maps/index.php`.

Creating image maps by hand can be tricky. Use an image editor to identify each point in your map, and *then* create the proper markup for it. Most (X) HTML tools include utilities to help you make image maps. If you take advantage of such a tool, you can create image maps quickly and with few errors. Find out more about (X)HTML tools in Chapter 22.

Part III
Taking Precise Control Over Web Pages and Styles



"Games are an important part of my Web site. They cause eye strain."

In this part . . .

In this part of the book, we introduce and describe Cascading Style Sheets (CSS), a powerful markup language that is often used to supplement (X)HTML and to manage the way it looks inside a Web browser. (X)HTML can reference CSS by including either an external style sheet or inline CSS markup within an (X)HTML document.

Here you start out by familiarizing yourself with the many and various capabilities of CSS, get a look at different kinds of style sheets, and get acquainted with the rules for handling multiple style sheets when they're applied to a single Web page (that's where the *Cascading* in CSS comes from). Of course, you also find out how to build and use CSS for things like creating visual layouts, positioning individual items, and handling fonts. Because CSS also provides controls over color and modifying the way text appears on the page, you find out how to deal with these capabilities as well.

Tables are an important way to organize and represent data in (X)HTML. No surprise that this part of the book shows basic table setup, structure, and syntax, and also explains how you can use CSS to manage table appearance.

Chapter 8

Introducing Cascading Style Sheets

In This Chapter

- ▶ Understanding CSS
- ▶ Creating style rules
- ▶ Linking style rules to Web pages
- ▶ Introducing CSS properties
- ▶ Understanding inheritance and the style cascade

The goal of *Cascading Style Sheets* (CSS) is to separate a Web page's style from its structure and to make it easier to maintain Web pages you've created. The structural elements of a page, such as headings (`<h1>` through `<h6>`) and body text, don't affect the look of those elements. By applying styles to those elements, you can *specify the element's layout on the page* and *add design attributes* (such as fonts, colors, and text indentation).

Style sheets give you precise control over how structural elements appear on a Web page. What's even better is that you can create one style sheet for an entire Web site to keep the layout and look of your content consistent from page to page. And here's the icing on this cake: Style sheets are easy to build and even easier to integrate into Web pages. In fact, you can add style markup to individual (X)HTML elements (called *inline style*), create sequences of style instructions in the head of an (X)HTML document (called an *internal style sheet*), or refer to a separate stand-alone style sheet using some kind of link or other reference (called an *external style sheet*) inside your (X)HTML document. In short, there are lots of ways to add style to a Web page!

REMEMBER

As more Web sites transition to XHTML, the goal of the markup powers-that-be is to eventually *deprecate* (make obsolete) all formatting markup, such as the `<font>` element, from HTML's collection of elements. Someday, all presentation will belong to CSS.

When you want tight control over the display of your Web pages, style sheets are the way to go:

- Generally, style sheets give you more flexibility than markup can.
- Future HTML and XHTML elements will no longer include display-oriented tags.

TIP

Most modern browsers handle CSS well. However, some older browsers — such as Internet Explorer 4.0 and Netscape Navigator — have trouble displaying CSS correctly. Earlier browsers can't display CSS at all. If you know that many of your site's users still use one or more of these obsolete browsers, test your pages in these browsers; make sure they can read your pages.

Advantages of Style Sheets

HTML's formatting capabilities are limited, to say the least. When you design a page layout in HTML, you're limited to tables, font controls, and a few inline styles, such as bold and italic. *Style sheets* supply lots of tools to format Web pages with precise controls. With style sheets you can

- **Carefully control every aspect of page display:** Specify the amount of space between lines, character spacing, page margins, image placement, and more. You can also specify positioning of elements on your pages.
- **Apply changes globally:** You can guarantee consistent design across an entire Web site by applying the same style sheet to every Web page.

  TIP

  You can modify the look and feel of an entire site by changing just one document (the style sheet) instead of the markup on every page. Need to change the look of a heading? Redefine that heading's style attributes in the style sheet and save the sheet. The heading's look changes throughout your site. You can imagine one page after rapidly adopting the new look in a "cascade" of changes (hence the name) but that's just a metaphor; the cascade is instantaneous.
- **Instruct browsers to control appearance:** Provide Web browsers with more information about how you want your pages to appear than you can communicate using HTML.
- **Create dynamic pages:** Use JavaScript or another scripting language along with style sheets to create text and other content that moves, appears, or hides in response to user actions.

What CSS can do for a Web page

The gist of how style sheets work is as follows:

1. You define rules in a style sheet that specify how you want content described by a set of markup to appear.

   For example, you could specify that every first-level heading (`<h1>`) be displayed in purple Garamond 24-point type with a yellow background (not that you *would,* but you could).
2. You link style rules and markup.
3. The browser does the rest.

The current specification, CSS2.1, can

- Specify font type, size, color, and effects
- Set background colors and images
- Control many aspects of text layout, including alignment and spacing
- Set margins and borders
- Control list display
- Define table layout and display
- Automatically generate content for such standard page elements as counters and footers
- Control cursor display
- Define aural style sheets for text-to-speech browsers

TECHNICAL STUFF

CSS3: Next-generation style sheets

The next generation of CSS — CSS3 — is a collection of *modules* that address different aspects of Web-page formatting, such as fonts, background colors, lists, and text colors. The first of these modules became standards (officially called *Candidate Recommendations*) in mid-2004. But the majority of CSS3 modules aren't expected to become Candidate Recommendations until mid-2008 or later, and few browsers implement CSS3 features. In short, you don't need to worry about CSS3 — yet.

The W3C has devoted an entire section of its Web site to this topic at `www.w3.org/style/css`. You can find general CSS information there, as well as keep up with the status of CSS3. The site links to good CSS references and tutorials, and includes information on software packages that can make your style-sheet endeavors easier.

Property measurement values

Many HTML properties use measurement values. We tell you which measurement values go with which properties throughout this book. Standard property measurements dictate the size of a property in two ways.

Absolute value measurements can dictate a specific length or height with one of these values:

- **inches,** such as `.5in`
- **centimeters,** such as `3cm`
- **millimeters,** such as `4mm`
- **picas,** such as `1pc`

  There are about six picas in an inch.
- **points,** such as `16pt`

  There are 12 points in a pica.
- **pixels,** such as `13px` (these match up to individual dots on your computer display).

Relative value measurements base length or height on a *parent element* value in the document:

- **p%:** A percentage of the current `font-size` value, such as `150%`.

  For example, you can define a font size of `80%` for all paragraphs. If your document body is defined with a 15-point font, the font size of the paragraphs is 12 points (80 percent of 15).
- **ex:** A value that is relative to the x-height of the current font. An x-height is the equivalent of the height of the lowercase character of a font, such as `1.5ex`.
- **em:** A value that is relative to the current font size, such as `2em`.

  Both `1em` and `100%` equal the current size.

Be careful when using these values; some properties support only some measurement values — length values, say, but not relative values. Don't let this jargon scare you. Just define the size in a value you're familiar with.

What you can do with CSS

You have a healthy collection of properties to work with as you write your style rules. You can control just about every aspect of a page's display — from borders to font sizes and everything in-between:

- **Background properties** control the background colors associated with blocks of text and with images. You can also use these properties to attach background colors to your page or to individual elements.
- **Border properties** control borders associated with the page, lists, tables, images, and block elements (such as paragraphs). You can specify border width, color, style, and distance from the element's content.

- **Classification properties** control how elements such as images flow on the page relative to other elements. You can use these properties to integrate images and tables with the text on your page.
- **List properties** control how lists appear on your page, such as
  - Managing list markers
  - Using images in place of bullets
- **Margin properties** control the margins of the page and margins around block elements, tables, and images. These properties extend ultimate control over the white space on your page.
- **Padding properties** control the amount of white space around any block element on the page. When you use these with margin and border properties, you can create some complex layouts.
- **Positioning properties** control where elements sit on the page; you can use them to put elements in specific places on the page.
- **Size properties** control how much space (in height and width) your elements (both text and images) take up on your page. They're especially handy for limiting the size of text boxes and images.
- **Table properties** control the layout of tables. You can use them to control cell spacing and other table-layout specifics.
- **Text properties** control how text appears on the page. You can set such properties as font size, font family, height, text color, letter and line spacing, alignment, and white space. These properties give you more control over your text with style sheets than the `font` HTML element can.

Entire books and Web sites are devoted to the fine details of using each and every property in these categories. We suggest one of these references:

- *CSS Web Design For Dummies* by Richard Mansfield, published by Wiley Publishing.
- Westciv's CSS2 reference on the Web:

  `www.westciv.com/style_master/academy/css_tutorial/index.html`

Although CSS syntax is straightforward, combining CSS styles with markup to fine-tune a page layout can get a little complicated. But to become a CSS guru, you just need to

- Know the details of how the different properties work.
- Experiment with how browsers handle CSS.

  Practice shows how to convey your message on the Web using CSS.

CSS Structure and Syntax

A style sheet is made of *style rules.* Each style rule has two parts:

- **Selector:** Specifies the markup element to which style rules apply.
- **Declaration:** Specifies how the content described by the markup looks.

You use a set of punctuation marks and special characters to define a style rule. The syntax for a style rule always follows this pattern:

```
selector {declaration;}
```

A declaration breaks down further into two items:

- **Properties** are aspects of how the computer displays text and graphics (for example, font size or background color).
- **Values** are the data that specify how you want text and images to look on your page (for example, a 24-point font size or a yellow background).

You separate the property from the value in a declaration with a colon (and each declaration ends with a semi-colon):

```
selector {property: value;}
```

For example, these three style rules set the colors for first-, second-, and third-level headings:

```
h1 {color: teal;}
h2 {color: maroon;}
h3 {color: black;}
```

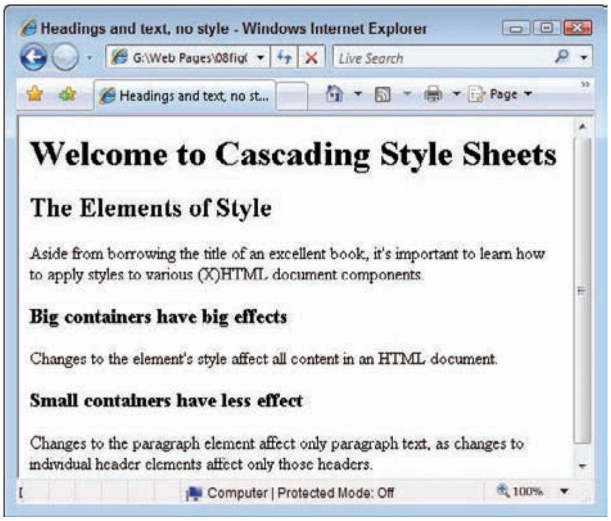
The CSS specification lists exactly which properties you can work with in your style rules and the different values they can take. Most are pretty self-explanatory (`color` and `border`, for example). See “What you can do with CSS,” earlier in this chapter, for a quick rundown of properties included in the CSS2 specification.

Style sheets override a browser’s internal display rules; your formatting specifications affect *the final appearance of the page in the user’s browser.* This means you can better control how your content looks and create a more consistent and appropriate experience for visitors.

For example, the following style rules specify the font sizes (in pixels) for first-, second-, and third-level headings:

```
h1 {font-size: 16px;}
h2 {font-size: 12px;}
h3 {font-size: 10px;}
```

Figure 8-1 shows a simple HTML page with all three heading levels (plus some body text) without the style sheet applied. The browser uses its default settings to display the headings in different font sizes.


Headings and text, no style - Windows Internet Explorer
G:\Web Pages\08figt
Live Search
Headings and text, no st...
Page
Computer | Protected Mode: Off
100%

Welcome to Cascading Style Sheets

The Elements of Style

Aside from borrowing the title of an excellent book, it's important to learn how to apply styles to various (X)HTML document components.

Big containers have big effects

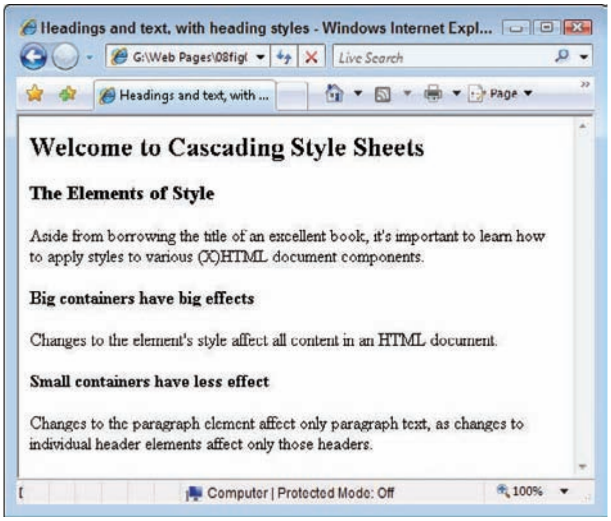
Changes to the element's style affect all content in an HTML document.

Small containers have less effect

Changes to the paragraph element affect only paragraph text, as changes to individual header elements affect only those headers.

Figure 8-1: An HTML page without style specifications.

Figure 8-2 shows the same Web page with a style sheet applied. The headings are significantly smaller than in the preceding figure. That's because the style sheet rules override the browser's defaults.


Headings and text, with heading styles - Windows Internet Expl...
G:\Web Pages\08figt
Live Search
Headings and text, with ...
Page
Computer | Protected Mode: Off
100%

Welcome to Cascading Style Sheets

The Elements of Style

Aside from borrowing the title of an excellent book, it's important to learn how to apply styles to various (X)HTML document components.

Big containers have big effects

Changes to the element's style affect all content in an HTML document.

Small containers have less effect

Changes to the paragraph element affect only paragraph text, as changes to individual header elements affect only those headers.

Figure 8-2: An HTML page with style specifications in effect.

Users can change their preferences so their browsers ignore your style sheets. (Most users will use your sheets.) Test your Web page with style sheets turned off to be sure it looks good (or acceptable) without your style sheets. (For detailed instructions on disabling or altering style sheets, see Jim Hatcher's discussion entitled "Reading Web Pages without CSS" at `www.jimthatcher.com/webcourseb.htm`; instructions vary depending on your Web browser, but you can use accessibility plug-ins to manage or disable style sheets as well.)

Selectors and declarations

You probably want a style rule to affect the display of more than one property for any given selector. You can create several style rules for a single selector, each with one declaration, like this:

```
h1 {color: teal;}
h1 {font-family: Arial;}
h1 {font-size: 36px;}
```

However, such a large collection of style rules becomes hard to manage. CSS lets you combine several declarations in a *single* style rule that affects the display characteristics of a single selector, like this:

```
h1 {color: teal;
    font-family: Arial;
    font-size: 36px;}
```

All the declarations for the `h1` selector are within the same set of brackets (`{}`) and are separated by semicolons (`;`). You can put as many declarations as you want in a style rule; just end each declaration with a semicolon.

The semicolon at the end of the last declaration is optional. Some people include it to be consistent and end every declaration with a semicolon, but it's not necessary. We use it both ways throughout this book.

From a purely technical standpoint, white space is irrelevant in style sheets (just as it is in HTML), but you should use a consistent spacing scheme to make it easy to read and edit your style sheets. One exception to this white-space rule occurs when you declare multiple font names in the font-family declaration. See the "Font family" sidebar for more information.

You can make the same set of declarations apply to a collection of selectors, too: You just separate the selectors with *commas.* The following style rule applies the declarations for text color, font family, and font size to the `h1`, `h2`, and `h3` selectors:

```
h1, h2, h3 {color: teal;
            font-family: Arial;
            font-size: 36px;}
```

Font family

When assigning values to the `font-family` property, you can provide a list of comma-separated font names. These names must match fonts available to the user's Web browser. If a font name — such as "Times New Roman" — includes spaces, it must be enclosed in quotation marks.

```
h1 {font-family: Verdana; "Times New
        Roman", serif;}
```

The browser seeks to use Verdana first, and if that's not available, it looks for Times New Roman next, and then uses a generic serif font as its last option. Chapter 10 covers fonts in CSS.

REMEMBER

The sample style rules in this section show that style-sheet syntax relies heavily on punctuation. When a style rule doesn't work exactly as you anticipate, make sure your syntax doesn't use a semicolon where you need a colon, and doesn't use a parenthesis where you need a bracket. Watch out for commas and semicolons, too!

TIP

The W3C's validation service can help you find problems in your style sheets:

```
http://jigsaw.w3.org/css-validator/
```

Working with style classes

Sometimes you need style rules that apply only to specific instances of an HTML markup element. For example, if you want a style rule that applies only to paragraphs that hold copyright information, you need a way to tell the browser that the rule has a limited scope.

To target a style rule more closely, combine the `class` attribute with a markup element. The following examples show HTML for two kinds of paragraphs:

- A regular paragraph (without a `class` attribute)

  ```
  <p>This is a regular paragraph.</p>
  ```

- A `class` attribute with the value of `copyright`

  ```
  <p class="copyright">This is a paragraph of class copyright.</p>
  ```

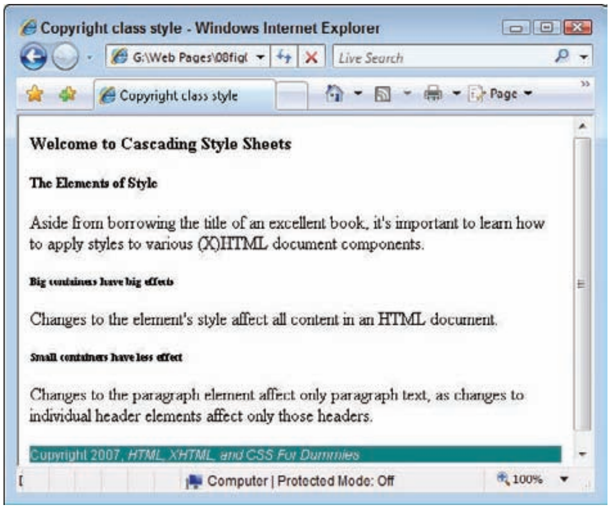
To create a style rule that applies only to the copyright paragraph, follow the paragraph selector in the style rule with

- A period (`.`)
- The value of the `class` attribute, such as `copyright`

The resulting rule looks like this:

```
p.copyright {font-family: Arial;
             font-size: 12px;
             color: white;
             background: teal;}
```

This style rule specifies that all paragraphs of class `copyright` display white text on a teal background in 12-pixel Arial font. Figure 8-3 shows how a browser applies this style rule only to the paragraph with the `copyright` attribute.


Copyright class style - Windows Internet Explorer
G:\Web Pages\08figl
Live Search
Copyright class style
Page
Computer | Protected Mode: Off
100%

Welcome to Cascading Style Sheets

The Elements of Style

Aside from borrowing the title of an excellent book, it's important to learn how to apply styles to various (X)HTML document components.

Big containers have big effects

Changes to the element's style affect all content in an HTML document.

Small containers have less effect

Changes to the paragraph element affect only paragraph text, as changes to individual header elements affect only those headers.

Copyright 2007, HTML, XHTML, and CSS For Dummies

Figure 8-3: Classes can target your style rules more precisely.

You can also create style-rule classes that aren't associated with any element, like the following example:

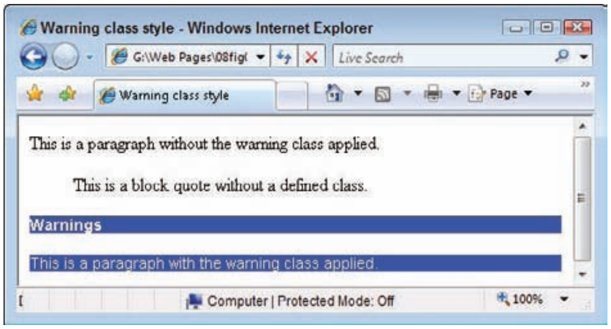
```
.warning {font-family: Arial;
          font-size: 14px;
          background: blue;
          color: white;}
```

You can use this style class with any element by adding `class="warning"` to that element. Figure 8-4 shows how a browser applies the warning style to the paragraph and heading, but not the block quote, in this HTML:

```
<p>This is a paragraph without the warning class applied.</p>
<blockquote>This is a block quote without a defined class.</blockquote>
<h1 class="warning">Warnings</h1>
<p class="warning">This is a paragraph with the warning class applied.</p>
```

You can also use the span element to selectively apply custom styles to inline content (or to create arbitrary content containers that extend from the opening <span> tag to its closing </span> counterpart):

```
<p>This is a paragraph without the <span class="warning">warning class</span>
 applied only to the words "warning class."</p>
```



Warning class style - Windows Internet Explorer

This is a paragraph without the warning class applied.

This is a block quote without a defined class.

Warnings

This is a paragraph with the warning class applied.

Figure 8-4: You can use classes to create style rules that work with any element.

Inheriting styles

One of the basic concepts in HTML (and markup in general) is nesting tags:

- Your entire HTML document is nested within <html> and </html> tags.
- Everything a browser displays in a window is nested within <body> and </body> tags. (That's just the beginning, really.)

The CSS specification recognizes that you often nest one element inside another and wants to be sure that styles associated with the parent element find their way to the child element. This mechanism is called *inheritance*.

TIP

Pay attention to inheritance!

When you build complex style sheets that guide the appearance of every aspect of a page, keep inheritance in mind. For instance, if you set margins for the page in a body style rule, all margins you set for every other element on the page are based on margins set for the body. If you know how your style rules work together, you can use inheritance to *minimize style rule repetition* and *create a cohesive display for your page.*

This chapter covers basic CSS syntax, but you can fine-tune your style rules with advanced techniques. A complete overview of CSS syntax rules is available in the "CSS Structure and Rules" tutorial by the Web Design Group at www.htmlhelp.com/reference/css/structure.html.

When you assign a style to an element, the same style is applied to all the elements nested inside the element. For example, a style rule for the `body` element that sets the page background, text color, font size, font family, and margins looks like this:

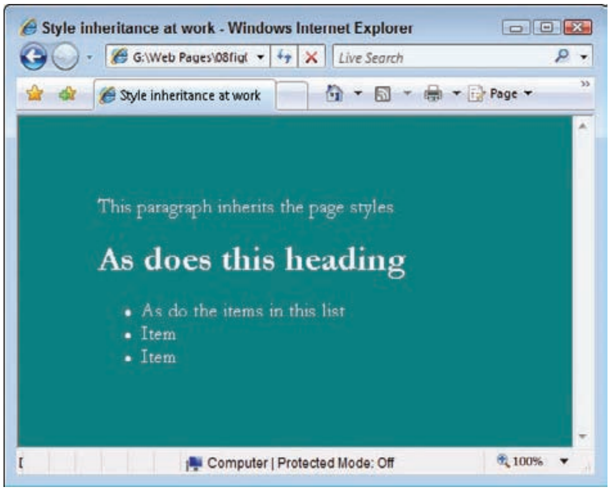
```
body {background: teal;
      color: white;
      font-size: 18px;
      font-family: Garamond;
      margin-left: 72px;
      margin-right: 72px;
      margin-top: 72px;}
```

TIP

If you want to set style rules for the entire document, set them in the `body` element. Changing the font for the entire page, for example, is much easier to do that way; it beats changing every single element one at a time.

When you link the following HTML to the preceding style rule, which applies only to the `body` element, that formatting is inherited by all subordinate elements (as shown in Figure 8-5):

```
<body>
  <p>This paragraph inherits the page styles.</p>
  <h1>As does this heading</h1>
  <ul>
      <li>As do the items in this list</li>
      <li>Item</li>
      <li>Item</li>
   </ul>
</body>
```


Style inheritance at work - Windows Internet Explorer
G:\Web Pages\08figt
Live Search
Style inheritance at work
Page
Computer | Protected Mode: Off
100%

This paragraph inherits the page styles

As does this heading

- As do the items in this list
- Item
- Item

Figure 8-5: Inheritance means style rules apply to nested elements.

Using Different Kinds of Style Sheets

When you finish creating your style rules, you're ready to connect them to your HTML page with one of these options:

- **Insert style information into your document.** You can either
  - Use the `<style>` element to build a style sheet into a Web page.

    This is an *internal style sheet.*
  - Use the `style` attribute to add style information directly to a tag.

    This is an *inline style.*
- Use an *external style sheet.* You can either
  - Use the `<link>` tag to *link* your Web page to an external style sheet.
  - Use the CSS `@import` statement to *import* an external style sheet into the Web page.

Internal style sheets

An internal style sheet lives within your HTML page. Just add style rules to the `<style>` element in the document header. You can include as many (or as few) style rules as you want in an internal style sheet. (See Listing 8-1.)

Listing 8-1: Adding an Internal Style Sheet to an HTML Document

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Internal Style Sheet Example</title>
  <style type="text/css">
    body {background: black;
          color: white;
          font-size: 16px;
          font-family: Garamond;
          margin-left: 72px;
          margin-right: 72px;
          margin-top: 72px;}

    h1, h2, h3 {color: teal;
                font-family: Arial;
                font-size: 36px;}

    p.copyright {font-family: Arial;
```

(continued)

Listing 8-1 *(continued)*

```
                  font-size: 12px;
                  font-color: white;
                  background: black;}

      .warning {font-family: Arial;
                font-size: 16px;
                font-color: red;}
    </style>
  </head>
  <body>

  <!-- Document content goes here -->

  </body>
  </html>
```

TIP

The benefit of an internal style sheet is convenience: Your style rules are on the same page as your markup so you can tweak both quickly. But if you want the same style rules to control the appearance of more than one HTML page, move those styles from individual Web pages to an external style sheet.

External style sheets

An external style sheet holds all your style rules in a separate text document you can reference from any HTML file on your site. You must maintain a separate style sheet file, but an external style sheet offers benefits for overall site maintenance. If your site's pages use the same style sheet, you can change any formatting characteristic on all pages with a change to the style sheet.

REMEMBER

We recommend using external style sheets on your sites.

Linking

To reference an external style sheet, use the `link` element in the Web page header, like this:

```
<html>
<head>
  <title>External Style Sheet Example</title>
  <link rel="stylesheet" type="text/css" href="styles.css" />
<head>
<body>

<!-- Document content goes here -->

</body>
</html>
```

The `href` attribute in the `<link>` element can take either

- A relative link (a style sheet on your own site)
- An absolute link (a style sheet that doesn't reside on your own site)

  Usually, you shouldn't use a style sheet that doesn't reside on your Web site — you want control of your site's look and feel.

  TIP

  If you want to quickly add style to your Web page (or experiment to see how browsers handle different styles), use an absolute URL to point to one of the W3C's Core style sheets. Read more about them at

  ```
  www.w3.org/StyleSheets/Core/
  ```

TIP

Chapter 6 covers the difference between relative and absolute links.

Importing

The `@import` statement instructs the browser to load an external style sheet and use its styles. You use it within the `<style>` element but before any of the individual style rules, like so:

```
<style>
  @import "http://www.somesite.edu/stylesheet.css";
</style>
```

TIP

Style rules in an imported style sheet take precedence over any rules that come before the `@import` statement. So if you have multiple external style sheets referenced with more than one `@import` statement on your page, rules apply from the later style sheets (the ones farther down on the page).

Use inline styles carefully

You can attach individual style rules, called an *inline style,* to individual elements in an HTML document. An inline style rule attached to an element looks like this:

```
<p style="color: green;">Green text.</p>
```

Adding style rules to an element isn't really the best approach. We generally recommend that you choose *either* internal *or* (preferably) external style sheets for your rules instead of attaching the rules to individual elements in your document. Here are a few reasons:

- Your style rules get mixed up in the page and are hard to find.
- You must place the entire rule in the value of the `style` attribute, which makes complex rules hard to write and edit.
- You lose all the benefits that come with grouping selectors and reusing style rules in external style sheets.

Understanding the Cascade

Multiple style sheets can affect page elements and build upon one another. It's like inheriting styles within a Web page. This is the *cascading* part of CSS.

Here's a real-world example: a Web site for a university's English department. The English department is part of the School of Humanities, which is just one school in the university. Each of these entities — the university, the school, and the English department — has its own style sheet.

1. The university's style sheet provides style rules for all pages in the site.
2. The school's style sheet links to the university's style sheet (using an `@import` statement), and adds more style rules specific to the look the school wants for its own site.
3. The English department's style sheet links to the school's style sheet.

   So the department's pages both *have their own style rules* and *inherit the style rules from both the school and the university's style sheet.*

But what if multiple style sheets define rules for the same element? What if, for example, all three style sheets specify a rule for the `h1` element? In that case, the nearest rule to the page or element you're working on wins out:

- If an `h1` rule exists on the department's style sheet, it takes precedence over the school and university `h1` styles.
- If an individual page within the department applies a style rule to `h1` in a `<style>` tag, that rule applies.

Chapter 9

Using Cascading Style Sheets

In This Chapter

▶ Getting a handle on using CSS

▶ Positioning objects on a page

▶ Creating font rules

▶ Creating style sheets for print

▶ Understanding aural style sheets

Understanding the structure and syntax of CSS is easy. Learning about the properties that CSS can apply to (X)HTML documents takes a little more time and effort. However, where the learning curve really gets interesting is when it comes to learning how to use CSS to take a plain or ordinary Web page and "kick it up a notch." This chapter deals with how to put CSS to work, rather than focusing on its structure and inner workings.

If you need a refresher of CSS style rules and properties, read Chapter 8 (a high-level overview of CSS and how it works). Then you can return to this chapter and put CSS into action.

Now it's time to make a page and give it some style!

To use CSS efficiently, follow these general guidelines:

- When you test how a page looks, use internal styles so you can tweak to your heart's delight. (This chapter shows internal style sheets, but Chapter 8 covers internal and external style sheets in greater detail.)
- When your test page looks just right, move those internal styles to an external sheet, and then apply them throughout your site.

Managing Layout, Positioning, and Appearance

You can use CSS to lay out your pages so that images and blocks of text

- Appear exactly where you want them to.
- Fit exactly within the amount of space you want them to occupy.

After you create styles within a document, you can create an external style sheet to apply the same styles to any page.

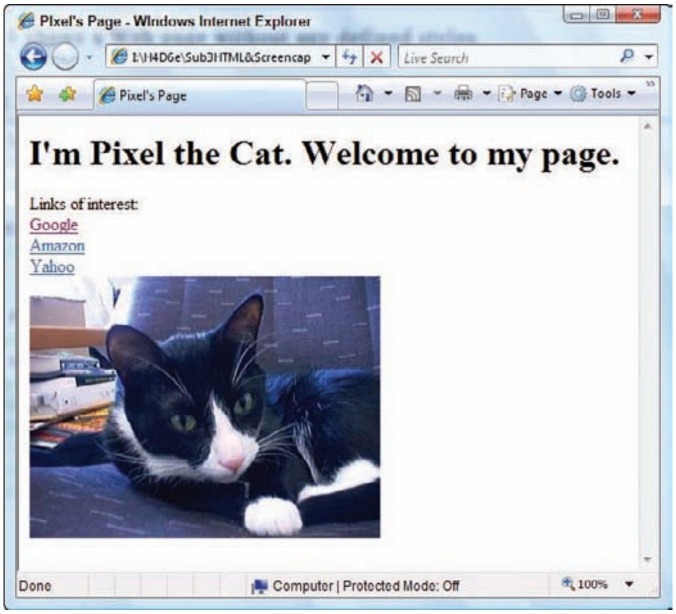
Developing specific styles

Listing 9-1 shows a Web page without any defined styles (see Figure 9-1).

Listing 9-1: A Fairly Dull Page

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Pixel's Page</title>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
</head>
 <body>
  <h1>I'm Pixel the Cat. Welcome to my page.</h1>
  <div class="navbar">
    Links of interest:<br />
      <a href="http://www.google.com/">Google</a><br />
      <a href="http://www.amazon.com/">Amazon</a><br />
      <a href="http://www.yahoo.com/">Yahoo</a>
  </div>
  <img src="/images/pixel1.jpg" alt="The Cat" width="320" height="240"
   id="theCat" />
 </body>
</html>
```

The cat looks great, but the page certainly doesn’t show off his possibilities. The addition of some styles improves this page immensely. Here’s how!


Pixel's Page - Windows Internet Explorer
E:\H4D6e\Sub3HTML&Screencap
Live Search
Pixel's Page
Page
Tools
Done
Computer | Protected Mode: Off
100%

I'm Pixel the Cat. Welcome to my page.

Links of interest:
Google
Amazon
Yahoo

Figure 9-1: This style-free page doesn't maximize this cat's possibilities.

Visual layouts

Instead of the links appearing above the image, as they are in Figure 9-1, we want them on the left, a typical location for navigation tools. The following markup floats the text for the search site links to the left of the image:

```
<style type="text/css">
  .navbar {
    background-color: #CCC;
    border-bottom: #999;
    border-left: #999;
    border-width: 0 0 thin thin;
    border-style: none none groove groove;
    display: block;
    float: left;
    margin: 0 0 0 10px;
    padding: 0 10px 0 10px;
    width: 100px;
  }
</style>
```

In the preceding rules, we

- Added a `<style>` element
- Defined the `navbar` class inside the `<style>` element
- Used the `navbar` class to instruct the content to float to the left of images, which causes them to appear in the same part of the page to the left, rather than above the graphic.

This rule says that anything on the page that belongs to the `navbar` class (as shown in Figure 9-2) should display with

- A light-gray background
- A thin grooved-line border at bottom and left, in a darker gray
- No top or right border
- A block that floats to the left (so everything else on the page moves right, as with the image in Figure 9-2)
- A left margin of 10 pixels
- Padding at top and bottom of 10 pixels each
- A navbar area 100 total pixels wide

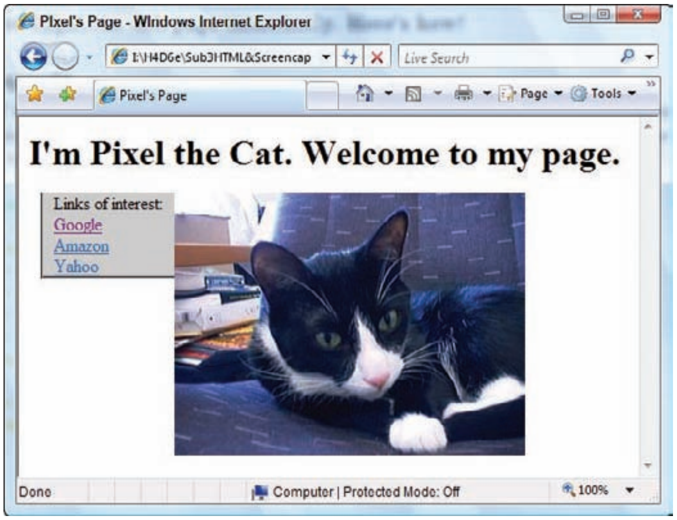

Pixel's Page - Windows Internet Explorer
Live Search
Pixel's Page
Page
Tools
I'm Pixel the Cat. Welcome to my page.
Links of interest:
Google
Amazon
Yahoo
Done
Computer | Protected Mode: Off
100%

Figure 9-2: The navigation bar now looks more like standard left-hand navigation.

TIP

Note that several properties in the declaration, called shorthand properties, take multiple values, such as `margin` and `padding`. Shorthand properties collect values from multiple related CSS properties (such as `margin-height`, `margin-width`, and so forth). See our online materials for a complete list.

Those values correspond to settings for the top, right, bottom, and left edges of the navbar's box. `margin` creates an empty zone around the box, and `padding` defines the space between the edges or borders of the box and the content inside the box. Here are the rules that explain how to associate values with properties that deal with margins, borders, padding, and so forth:

- If all the sides have the same value, a single value works.
- If *any* side is different from the others, *every* side needs a separate value. It's okay to set some or all of these values to zero as you see fit; this can often help to ensure that pages display consistently across a wider range of browsers (and browser versions).

TECHNICAL STUFF

To remember what's what, think of the edges of an element box in clockwise order, starting with the top edge: `top`, `right`, `bottom`, and then `left`.

Positioning

CSS provides several ways to specify exactly where an element should appear on a page. These controls use various kinds of positioning based on the relationships between an element's box and its parent element's box to help page designers put page elements where they want them to go. The kinds of properties involved are discussed in the following sections.

Location

You can control the horizontal and vertical location of an image. But when you use absolute positioning with any page element, you instruct that element exactly where it must sit, relative to the upper left corner of the browser window. Thus, instead of letting it be drawn automatically to the right of the navigation bar, you can place an image down and to the left, as in Figure 9-3:

```
#theCat {position: absolute; top: 100px; left: 100px;}
```

TIP

You might be wondering why the `navbar` rule starts with a period, and the `theCat` rule starts with a pound symbol (also known as a hash mark or octothorpe). That's because the period applies to a `class` attribute, but the pound symbol applies to an `id` attribute. You can apply either a `class` or an `id` to specific elements; the difference between these two is that a `class` can be used repeatedly, but an `id` can appear only once on a page. You can't have anything else on the page that uses `theCat` as its `id`. The difference, quite simply, is that a `class` lets you refer to some entire kind of element with a single reference, but an `id` can address only a single instance of an element.

Overlapping

Two objects *can* be assigned to the same position in a Web page. When that happens, the browser must decide the display order and which objects to show and which ones to hide.

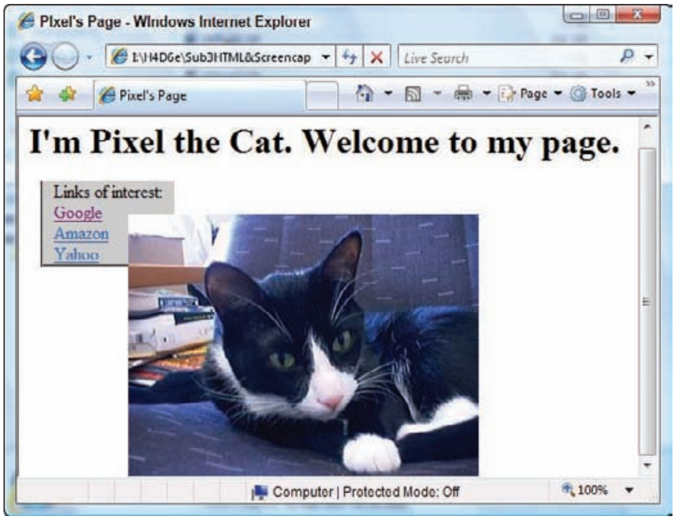

Pixel's Page - Windows Internet Explorer
I'm Pixel the Cat. Welcome to my page.
Links of interest:
Google
Amazon
Yahoo
Computer | Protected Mode: Off
100%

Figure 9-3: The image is more striking in this location.

The `z-index`, added to any rule, tells CSS how you want any object stacked over and under other objects that occupy the same space on the page:

- Lower numbers move down the stack.
- Higher numbers move up the stack.
- The default value for `z-index` is `auto`, which means it's the same as for its parent element.

Giving `theCat` a `z-index` value of `-1` automatically puts it behind everything else on the page (as shown in Figure 9-4) for which the `z-index` isn't set.

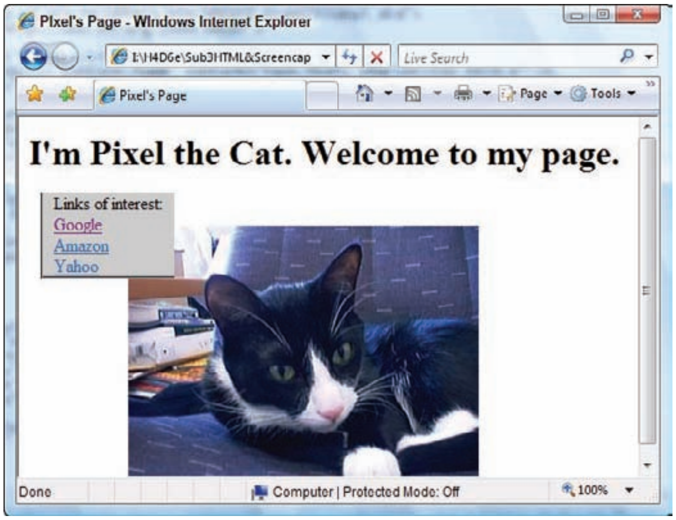
Fonts

You can make a page more interesting by replacing old boring default fonts. Start by specifying a generic body font as well as setting some other default rules, such as background color and text color.

Body text

Here's an example that sets the style for text within the `body` tag:

```
body {font-family: verdana, geneva, arial, helvetica, sans-serif;
      font-size: 12px; line-height: 16px; background-color: white;
      color: teal;}
```


Pixel's Page - Windows Internet Explorer
Live Search
Pixel's Page
Page
Tools
Done
Computer | Protected Mode: Off
100%

I'm Pixel the Cat. Welcome to my page.

Links of interest:
Google
Amazon
Yahoo

Figure 9-4: The cat's peeking out from behind the navigation bar.

Because the `body` element holds all content for any Web page, this affects everything on the page. The preceding rule instructs the browser to show all text that appears within the `body` element as follows:

- The text is rendered using one of the fonts listed. We placed Verdana at the head of the list because it is the preferred choice, and browsers check for available fonts in the order listed. (Note that a generic font, in this case `sans-serif`, almost always appears last in such lists because the browser can almost always supply such a font itself.)

  You can list more than one font. The browser uses the first font in your list that's available in the browser. For example, the browser looks for fonts from our list in this order:

  1. Verdana
  2. Geneva
  3. Arial
  4. Helvetica
  5. The browser's default sans-serif font
- 12-pixel font size
- 16-pixel line height

  The lines are spaced as though the fonts are 16 pixels high, so there's more vertical space between lines.

Figure 9-5 shows that

- All changes apply to the entire page, including the navigation bar.
- The `font-family` changes in the `h1` heading, but neither the `font-size` nor `line-height` changes for that heading.

  Because headers have specific defaults for `font-size` and `line-height`, another rule is needed to modify them.

TECHNICAL STUFF

In shooting Figure 9-5, the HTML used for our screen capture includes an additional tweak for IE. That's because a bug in Internet Explorer for Windows that doesn't occur in other browsers causes heading (`h1`) text to get truncated at the top. (Try the source (X)HTML for Figure 9-5 in IE to see what we mean; we had to add CSS markup that set `line-height: 105%;` for `h1` to create this display.) Unfortunately, CSS rendering can be unpredictable enough that you must test style rules in various browsers to see how they look — and then tweak accordingly.

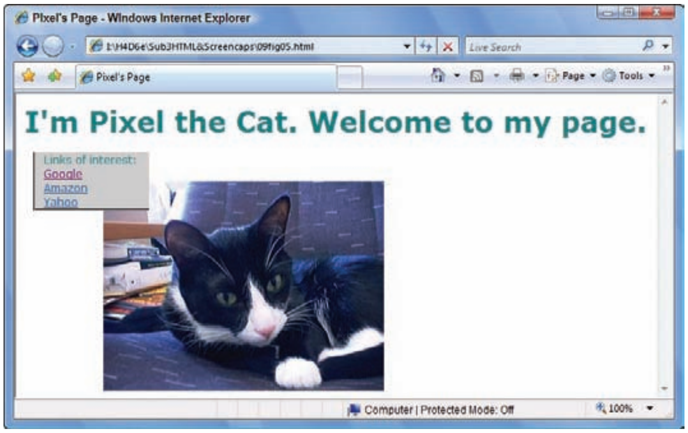

Pixel's Page - Windows Internet Explorer
Live Search
Pixel's Page
Page
Tools
I'm Pixel the Cat. Welcome to my page.
Links of interest:
Google
Amazon
Yahoo
Computer | Protected Mode: Off
100%

Figure 9-5: The fonts are nicer, but they could still use a little more work.

Headings

If we explicitly assign style properties to the `h1` element, display results are more predictable. Here's a sample set of styles:

```
h1 {font-family: "trebuchet ms", verdana, geneva, arial, helvetica, sans-serif;
    font-size: 24px; line-height: 26px;}
```

Figure 9-6 shows a first-level heading using the font family and type size that we want: 24-pixel Trebuchet MS, with a 26-pixel line height. If we didn't have the Trebuchet MS font on our system, the heading would appear in Verdana.

REMEMBER

When a font name includes spaces (like `trebuchet ms` or `times new roman`), the full name must be within quotation marks. (See Chapter 8 for more information.)

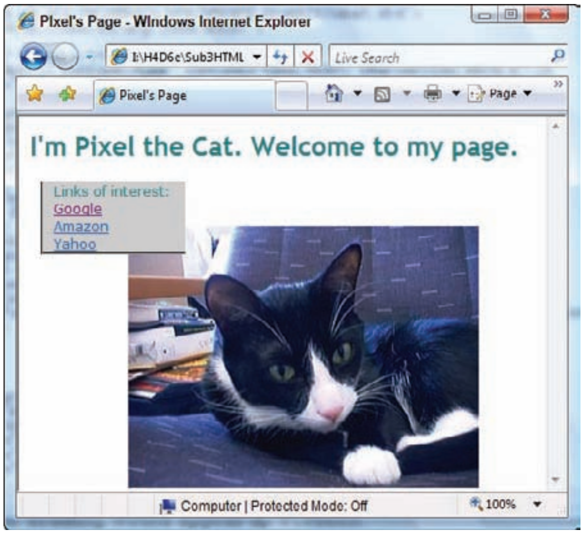

Pixel's Page - Windows Internet Explorer
I:\H4D6e\Sub3HTML
Live Search
Pixel's Page
Page
I'm Pixel the Cat. Welcome to my page.
Links of interest:
Google
Amazon
Yahoo
Computer | Protected Mode: Off
100%

Figure 9-6: Declaring a rule for h1 makes it appear just how we like it.

Hyperlinks

We think that having the hyperlinks underlined — which is normal — makes the menu look a little cluttered. Luckily, we can turn underlines off with CSS, but we still want the hyperlinks to look like hyperlinks, so we tell CSS to

- Make links bold.
- Make underlines appear when the cursor is over a link.
- Show links in certain colors.

The following style rules define how a browser should display hyperlinks:

```
a {text-decoration: none; font-weight: bold}
a:link {color: blue}
a:visited {color: #93C}
a:hover {text-decoration: underline}
```

What's going on here? Starting from the top, we're setting two rules for the `<a>` tag that apply to all links on the page:

- **The `text-decoration` declaration sets its value to `none`.**

  This gets rid of the underlining for all the links.
- **The `font-weight` declaration has a value of `bold`.**

  This makes all the links on the page appear in bold.

The remaining rules in the preceding code are *pseudo class selectors.* Their most common usage is to modify how links appear in their different states. (For more information on pseudo classes, see Chapter 10.) Here we change the color when a link has been visited, and we turn underlining on when the mouse pointer hovers over link text — doing so identifies hyperlinks when the cursor is in clicking range. Figure 9-7 shows how the page appears when the previous style rules are applied.

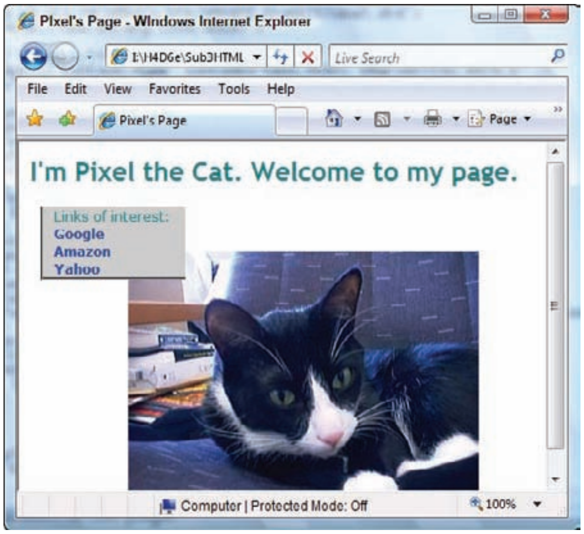

Pixel's Page - Windows Internet Explorer
E:\H4D6e\Sub3HTML
Live Search
File Edit View Favorites Tools Help
Pixel's Page
Page
I'm Pixel the Cat. Welcome to my page.
Links of interest:
Google
Amazon
Yahoo
Computer | Protected Mode: Off
100%

Figure 9-7: The final version of our page.

Externalizing style sheets

When the final page is the way you want it, you're ready to cut and paste your tested, approved, internal style sheet into an external style sheet.

- Every page of the site can use the whole style sheet with the addition of only one line of code to each page.
- Changes can be made site-wide with one change in the external style sheet.

To create an external style sheet from a well-tested, internal style sheet, follow these steps:

1. **Copy all text that sits between the `<style>` and `</style>` tags.**
2. **Paste that text into its own new document.**

   This text should include only CSS markup, without any HTML tags or markup.
3. **Add a `.css` suffix to the document's name (for example, `myStyles.css`) when saving.**

   The suffix shows at a glance that it's a CSS file.

So you've got your external style sheet. Time now to link your HTML file to said external style sheet. You have two options available to you:

- **Use the `<link>` tag.**

  All CSS-capable browsers understand the `link` tag.
- **Use the `<style>` tag with the `@import` keyword.**

  Only newer browsers understand the `<style>` and `@import` combination.

See Chapter 8 for more on these two methods.

Style sheets for old and new browsers

To include rules that both old and new browsers can handle, you can create *two* style sheets for a site:

- A basic style sheet that contains only the simplest of styles
- A complex style sheet that uses the capabilities of the most powerful new browsers

The following code uses two style sheets:

```
<link href="simpleStyles.css" rel="stylesheet"  type="text/css" />
<style type="text/css">
  @import "complexStyles.css";
</style>
```

Here's how that works:

- A `<link>` tag brings in `simpleStyles.css`, a basic style sheet for old browsers.
- The `<style>` tag and `@import` keyword combination brings in `complexStyles.css`, a complex style sheet for new browsers, which overrides the styles in `simpleStyles.css`.

Both old and new browsers get exactly the rules they can handle.

Multimedia

You can specify how you want your Web pages to look or behave on different *media types* depending on the medium.

Table 9-1 lists all the media types and their uses.

Table 9-1 Recognized Media Types

Media Type	*Description*
`all`	Suitable for all devices
`aural`	For speech synthesizers
`braille`	For Braille tactile-feedback devices
`embossed`	For paged Braille printers
`handheld`	For handheld devices (such as those with a small screen, monochrome monitor, and limited bandwidth)
`print`	For paged, opaque material and for documents viewed onscreen but in Print Preview mode
`projection`	For projected presentations such as projectors or transparencies
`screen`	For color computer screens
`tty`	For media that use a fixed-pitch character grid such as teletypes, terminals, or portable devices with limited display capabilities
`tv`	For television-type devices (such as those with low resolution, color capability, limited-scrollability screens, and some sound available)

CSS can make changes to customize how the same pages

- Render on a computer screen
- Print on paper

  A nifty color background might make your page a mess when it's printed on a black-and-white laser printer, but proper use of print-media styles can keep this sort of thing from happening!
- Sound when read out loud

Visual media styles

Table 9-2 lists the CSS properties that you're most likely to use in a typical Web page. Our online content for this book includes brief descriptions of the most commonly used CSS properties and (X)HTML tags and attributes.

Table 9-2 Visual Media Styles

Property	*Values*	*Default Value*	*Description*
`background-color`	Any color, by name or hex code	`transparent`	Background color of the page
`background-image`	URL	`none`	URL of an image to display in a page background
`color`	Any color, by name or hex code	Up to you!	Color of the foreground text

`font-family`	Any named font
	`cursive`
	`fantasy`
	`monospace`
	`sans-serif`
	`serif`
		Up to you!	Font to display

`font-size`	Number + unit
	`xx-small`
	`x-small`
	`small`
	`medium`
	`large`
	`x-large`
	`xx-large`
		`medium`	Size of the font to be displayed

`font-weight`	`normal`
	`bold`
	`bolder`
	`lighter`
		`normal`	Weight (how bold or light) the font should appear

`line-height`	`normal` number + unit	`normal`	Vertical spacing between lines of text

`text-align`	`left`
	`right`
	`center`
	`justify`
		Up to you + normal text direction	Which way the text on the page should be aligned

continued

Table 9-2 *(continued)*

Property	*Values*	*Default Value*	*Description*
`list-style-image`	URL	`none`	URL of an image to display as the bullets for a list
`list-style-position`	`inside` `outside`	`outside`	Wrapping the list text inside or outside of bullet points
`list-style-type`	`disc` `circle` `square` `decimal` `decimal-leading-zero` `lower-alpha` `upper-alpha` `none`	`disc`	Bullet type on lists
`display`	`block` `inline` `none`	`inline`	Format of a defined section of the page
`top`	Percentage number + unit `auto`	`auto`	For absolutely positioned objects, the offset from the top edge of the positioning context
`right`	Percentage number + unit `auto`	`auto`	For absolutely positioned objects, the offset from the right edge of the positioning context
`bottom`	Percentage number + unit `auto`	`auto`	For absolutely positioned objects, the offset from the bottom edge of the positioning context
`left`	Percentage number + unit `auto`	`auto`	For absolutely positioned objects, the offset from the left edge of the positioning context

Property	*Values*	*Default Value*	*Description*

`position`	`static`
	`absolute`
	`relative`
	`fixed`
		`static`	Method by which an element box is laid out, relative to positioning context

`visibility`	`collapse`
	`visible`
	`hidden`
	`inherit`
		`inherit`	Indicates whether an object will be displayed on the page

`z-index`	`Number` `auto`	`auto`	Stacking order of an object

`border-style`	`none`
	`dotted`
	`dashed`
	`solid`
	`double`
	`groove`
	`ridge`
	`inset`
	`outset`
		Not defined	The displayed style of an object's borders Can be broken out into `border-top-style, border-right-style, border-bottom-style,` and `border-left-style`

`border-width`	`thin`
	`medium`
	`thick`
	Number
		Not defined	Width of the border around an object Can be broken out into `border-top-width, border-right-width, border-bottom-width,` and `border-left-width`

`border-color`	Any color, by name or hex code `transparent`	Not defined	Color of an object's border Can be broken out into `border-top-color, border-right-color, border-bottom-color,` and `border-left-color`

continued

Table 9-2 *(continued)*

Property	*Values*	*Default Value*	*Description*

`float`	`left`
	`right`
	`none`
		`none`	Specifies whether the object should be floated to one side of the document

`height`	Percentage number + unit `auto`	`auto`	Displayed height of an object
`width`	Percentage number + unit `auto`	`auto`	Displayed width of an object
`margin`	Percentage number + unit `auto`	Not defined	Displayed margins of an object Can be broken out into `margin-top`, `margin-right`, `margin-bottom`, and `margin-left`
`padding`	Percentage number + unit `auto`	Not defined	Displayed blank space around an object Can be broken out into `padding-top`, `padding-right`, `padding-bottom`, `and padding-left`

`cursor`	`auto`
	`crosshair`
	`default`
	`pointer`
	`move`
	`text`
	`help`
		`auto`	Cursor appearance in the browser window

TIP

Some browsers don't support all CSS properties. If you're using CSS features, test your pages with the browsers that you expect your visitors will use.

If you want to take an extremely thorough guide to CSS everywhere you go, put it on your iPod! Westciv's free podGuide is a folder of small text files. Download the zipped file and follow the instructions on how to install it, and you have complete documentation with you at all times. (You also win the title of "World's Biggest CSS Geek.") The podGuide is at

```
www.westciv.com/news/podguide.html
```

Paged media styles

CSS can customize how a page looks when it's printed. We recommend these guidelines:

- Replace sans-serif fonts with serif fonts.

  Serif fonts are easier to read in print than sans-serif fonts.
- Insert advertisements that
  - Make sense when they aren't animated
  - Are useful without clicking

Aural (speech-sound) styles

Aural browsers and styles aren't just for the visually impaired. They're also useful for Web users who

- Have reading problems
- Need information while driving

The following example recommends voices to be played using male and female characters to make it clear which characters are speaking:

```
<style>
    @media aural {
       p.stanley {voice-family: male;}
       p.stella {voice-family: female;}
    }
</style>
```

Usually you don't have to worry much about adding aural styles to your page. The default readers should work just fine if

- Your page is mostly text.
- You don't have a strong opinion about how it sounds, so that any clearly male or female voice will do.

That said, you can find a complete listing of all aural style properties on this book's companion Web site.

In general, paged media styles help ensure that text looks as good when it's printed as it does in a Web browser. Paged media styles also help you hide irrelevant content when pages are printed (banners, ads, and so forth), thus reducing wasted paper and user frustration. See Table 9-3 for an explanation of paged media properties in CSS that you can use to help your users make the most when printing Web pages.

Table 9-3 Paged Media Styles

Property	*Values*	*Default Value*	*Description*
`orphans`	Number	`2`	The minimum number of lines in a paragraph that must be left at the bottom of a page
`page-break-after`	`auto` `always` `avoid` `left` `right`	`auto`	The page-breaking behavior after an element
`page-break-before`	`auto` `always` `avoid` `left` `right`	`auto`	The page-breaking behavior before an element
`page-break-inside`	`auto` `avoid`	`auto`	The page-breaking behavior inside an element
`widows`	Number	`2`	The minimum number of lines in a paragraph that must be left at the top of a page

The example in Listing 9-2 uses these options for paged media styles:

- Make the output black text on a white background.
- Replace sans-serif fonts with serif fonts.

Listing 9-2: Adding a Print Style Sheet

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>This is my page</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
<style>
    body {background-color: black; color: white; font-family: sans-serif;}

    @media print {
     body {background-color: white; color: black; font-family: serif}
   }
</style>
</head>
<body>
    This page will look very different when sent to the printer.
</body>
</html>
```

If you're now wondering why none of the properties in Table 9-3 were set, but other properties were, it's because (in this example) their defaults worked fine. And just because those page properties can be set doesn't mean that you can't set other properties also — it isn't an either/or.

Chapter 10

Getting Creative with Colors and Fonts

In This Chapter

- Using CSS to define text formatting
- Working with page colors and backgrounds
- Changing font display
- Adding text treatments

Before style sheets came along, you had to rely on HTML markup to control backgrounds, colors, fonts, and text sizes on Web pages. With style sheets on the scene, however, designers could now separate style information from content — meaning they could use Cascading Style Sheets (CSS) to control font, color, and other style information.

Why bother? Simple. When you use CSS, you get the following:

- Better control when updating or editing formatting information.
- No more HTML documents cluttered with `<font>` tags.
- More options for formatting text, such as defining line height, font weight, and text alignment, and converting text to *uppercase* (capital letters) or lowercase characters.

TIP

(X)HTML still includes a few formatting elements, such as `<tt>`, `<i>`, `<big>`, `<b>`, and `<small>`; however, the remaining formatting elements, such as `<font>`, are *deprecated*. That means they're no longer recommended for use (although they still work, and most browsers recognize them). We don't think you should use them anymore, but that decision is yours to make.

Color Values

(X)HTML defines color values in two ways:

- By *name* (you choose from a limited list)
- By *number* (harder to remember, but you have many more options)

Color names

The HTML specification includes 16 color names that you can use to define colors in your pages. Table 10-1 shows these colors below (the numbers that start with the hash mark # are in *hexadecimal* notation, a mix of the letters A–F (for 10 through 15) and the more typical 0–9 we all know and love from decimal numbers.

Table 10-1: Named color values in (X)HTML

Color name	*#RGB code*	*color*
Black	#000000	⬛
Silver	#C0C0C0	⬜
Gray	#808080	⬛
White	#FFFFF	⬜
Maroon	#800000	🟫
Red	#FF0000	🟥
Purple	#800080	🟪
Fuchsia	#FF00FF	🟪
Green	#008000	🟩
Lime	#00FF00	🟩
Olive	#808000	🟫
Yellow	#FFFF00	🟨
Navy	#000080	🟦
Blue	#0000FF	🟦
Teal	#008080	🟦
Aqua	#00FFFF	🟦

You can safely use color names in your CSS markup and be confident that browsers will recognize them and use the correct colors in your Web pages. You can also compare the colors you see on the screen to those you see on this printed page to see how print and digital displays can sometimes differ. (In some cases, it may be the color balance on your screen that's off; in others, the color the printer tried to match on the page may not be precisely correct — it's not as easy as you might think!)

TIP

Visit `www.htmlhelp.com/reference/html40/values.html#color` to see how your browser displays these colors. If you can, view this page on two or three different computers to see how a different browser, operating system, graphics card, and monitor can subtly change the display.

This CSS style declaration says all text within `<p>` tags should be *blue:*

```
p {color: blue;}
```

If you're looking for burnt umber, chartreuse, or salmon, you're out of luck. A box of 64 crayons this list is not. You can, however, also find hex codes for Web-safe colors, along with color swatches, on the Cheat Sheet. These colors, though unnamed, are *Web-safe* because they reproduce pretty reliably on most color computer display devices and printers.

Color numbers

Color numbers allow you to use any color (even salmon) on your Web page.

Hexadecimal color codes

TECHNICAL STUFF

Hexadecimal notation uses six characters — a combination of numbers and letters — to define color. If you know a color's hexadecimal code (often called its *hex code* for short), you have all you need to use that color in your HTML page.

TIP

When you use hexadecimal code to define a color, you should always precede it with a pound sign (#). Otherwise, it may not display properly in some Web browsers.

This CSS style declaration makes all text contained by `<p>` tags *blue:*

```
p {color: #0000FF;}
```

Finding any color's hex code

You can't just wave your magic wand and come up with the hex code for any color. But that doesn't mean that you can't find out through less magical means. Color converters follow a precise formula that changes a color's standard RGB notation into hexadecimal notation. Because you have better things to do with your time than compute hex codes, you have several options for finding out the code for your color of choice, including Web-safe colors on this book's Cheat Sheet. None of these make you use a calculator:

- **On the Web:** Some good sources for hexadecimal color charts are

  ```
  www.colorschemer.com/online.html
  www.webmonkey.com/reference/color_
          codes
  ```

  You simply find a color you like and type the hex code listed next to it into your HTML.
- **Using image-editing software:** Many image-editing applications, such as Adobe Photoshop or Jasc Paint Shop Pro, display the hexadecimal notation for any color. Even Microsoft Word's color picker shows you hex codes for colors in an image. If you have an image you like that you want to use as a color source for your Web page, open the image in your favorite editor and find out what the colors' hex codes are.

RGB values

You can use two decimal RGB values to define color. These value types aren't as common as hexadecimal values, but they're just as effective:

- `rgb(r,g,b)`: The `r`, `g`, and `b` are integers between 0 and 255 that represent the red, green, and blue of the color.
- `rgb(r%,g%,b%)`: The `r%`, `g%`, and `b%` represent the percentage of red, green, and blue of the color.

TECHNICAL STUFF

Every color can be defined as a mixture of red, green, and blue (RGB). You can use either an RGB value or the equivalent hex code to describe a color's RGB value to a Web browser.

Color Definitions

You can define individual colors for any text on the Web page, as well as define a background color for the entire Web page or some portion thereof.

CSS uses the following properties to define color:

- color defines the font color and is also used to define colors for links in their various states (*link*, *active*, *focus*, *visited*, and *hover*; all of these states are described later in this chapter).
- background or background-color defines the background color for the entire page or defines the background for a particular element (for example, a background color for all first-level headings, similar to the idea of highlighting something in a Word document).

Text

You can change the color of text on your Web page with three steps:

1. **Determine the selector.**

   For example, will the color apply to all first-level headings, to all paragraphs, or to a specific paragraph?

2. **Use the color property.**
3. **Identify the color name or hexadecimal value.**

The basic syntax for the style declaration is:

```
selector {color: value;}
```

Here is a collection of style declarations that use the color property:

```
body {color: olive; font-family: Verdana, sans-serif;
      background-color: #FFFFFF; font-size: 85%;}
hr {text-align: center;}
.navbar {font-size: 75%; text-align: center;}
h1 {color: #808000;}
p.chapternav {text-align: center;}
.footer {font-size: 80%;}
```

In the preceding CSS rules, the color for all text on the page is defined by using the body selector. The color is applied to all text in the body of the document unless otherwise defined. For example, the first-level heading is defined as forest green by using hexadecimal notation.

Links

Pseudo classes allow you to define style rules based on information outside the document tree.

The most common CSS use of pseudo classes is to define a style rule for a given element in the *document tree* — a technical term that just means that the browser builds a hierarchical representation of all elements in a document, much like a family tree, where every element has a parent and may contain a child. For example, `:link` is a pseudo class that defines style rules for any link that hasn't yet been visited.

There are five common pseudo classes that you can use with hyperlinks:

- `:link` defines formatting for links that haven't been visited.
- `:visited` defines formatting for links that have been visited.
- `:focus` defines formatting for links that are selected by the keyboard (for example, by using the Tab key) and are about to be activated by using the Enter key.
- `:hover` defines formatting for links when the mouse cursor hovers over them.
- `:active` defines formatting for links when they are selected (clicked by the mouse, or activated by using the Enter key).

The pseudo class name is preceded by a colon (`:`).

Pseudo classes can be used with

- Elements (such as the `<a>` element that defines hyperlinks)
- Classes
- IDs

For example, to define the style rules for visited and unvisited links, use the following syntax:

- This sets the color of any hyperlink pointing to an unvisited URL to red by using its hexadecimal value:

  ```
  a:link {color: #FF0000;}
  ```

- This sets any hyperlink that points to a visited URL to appear in the named color `green`:

  ```
  a:visited {color: green;}
  ```

- This designates unvisited links with a class of `internal` to appear in (named color) `yellow`: (See Chapter 8 for a discussion of CSS classes.)

  ```
  a.internal:link {color: yellow;}
  ```

Links can occupy multiple states at one time. For example, a link can be visited and hovered over at the same time. Always define link style rules in the following order: `:link`, `:visited`, `:visible`, `:focus`, `:hover`, `:active`.

CSS applies *last rule seen* to display your page. In this case, if you put the pseudo class selectors in the wrong order, your results may not be what you want. For example, if `visited` follows `hover`, and the two have overlapping rules, then hover effects apply only to links that haven't yet been visited.

The following CSS rules render the document with olive as the color for links that haven't been visited and yellow as the color of visited links:

```
body {color: #808000; font-family: Verdana, sans-serif; font-size: 85%;}
a:link {color: olive;}
a:visited {color: yellow;}
```

Some browsers don't support pseudo classes with elements such as `input` or `select` (these are forms elements). Current browsers support their use with the `a` element. Test your results if you want to use pseudo classes with an element other than `<a>`.

The CSS specification defines `:link` and `:visited` as mutually exclusive, and it is up to the browser application to determine when to change the state (visited versus unvisited) for any given link. For example, a browser might determine that a link is unvisited if you clear your history data.

Backgrounds

To change the background color for your Web page, or a section of that page, follow these steps:

1. **Determine the selector.** For example, will the color apply to the entire background, or will it apply only to a specific section?
2. **Use the `background-color` or `background` property.**
3. **Identify the color name or hexadecimal value.**

The basic syntax for the style declaration is:

```
selector {background-color: value;}
```

In the following collection of style declarations, the first style declaration uses the `background-color` property and sets it to light green by using hexadecimal notation:

```
body {color: #808000; font-family: Verdana, sans-serif;
      background-color: #EAF3DA; font-size: 85%;}
```

You can apply a background color to a block of text — for example, a paragraph — just like you define a background color for the entire page.

TIP

You use `background` as a shorthand property for all individual background properties or `background-color` to set just the color, like this:

```
selector {background: value value value;}
```

See Chapter 8 or "The Shorthand Property" section of Webmonkey's "Mulder's Stylesheets Tutorial" for more information (here's the link):

```
http://www.webmonkey.com/98/15/index3a_page6.html?tw=authoring
```

Fonts

You can define individual font properties for different HTML elements with

- Individual CSS properties, such as `font-family`, `line-height`, and `font-size`
- A group of font properties in the catchall shorthand `font` property

Font family

To define the font face (a named and often copyrighted set of specific character designs, such as Times-Roman, Bodoni, or Helvetica) by using the CSS `font-family` property:

1. **Identify the selector for the style declaration.**

   For example, making `p` the selector defines a font family for all `<p>` tags.

2. **Add the property name `font-family`.**

   TECHNICAL STUFF

   Not all font families are supported by every browser. CSS allows you to specify multiple font families in case a browser doesn't support the font family you prefer. You can list multiple font family names, separated by commas. The browser uses the first name in the list that is available on the computer on which it's running.

3. **Define a `value` for the property (the name of the font family).**

   REMEMBER

   Use single or double quotation marks around any font family names that include spaces.

To format all first-level headings to use the Verdana font, use a style declaration like this:

```
h1 {font-family: Verdana, Helvetica, sans-serif;}
```

TIP

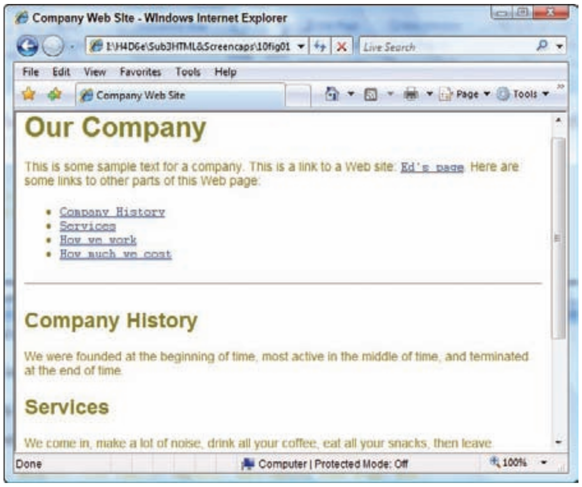
In the preceding declaration, two more font families appear in case someone's browser doesn't support the Verdana font family.

We recommend including these font families in your style declarations:

- At least one of these *common* font families:
  - Arial: Sample SAMPLE
  - Helvetica: Sample SAMPLE
  - Times New Roman: Sample SAMPLE
  - Verdana: Sample SAMPLE
- At least one of these *generic* font families:
  - serif: Sample SAMPLE
  - sans-serif: Sample SAMPLE
  - cursive: Sample SAMPLE
  - fantasy: Sample SAMPLE
  - monospace: `Sample SAMPLE`

Different elements may be formatted using different font families. These rules define a different font family for hyperlinks (see Figure 10-1):

```
body {color: #808000; font-family: Arial, sans-serif; font-size: 85%;}
hr {text-align: center;}
a {font-family: Courier, "Courier New", monospace;}
```


Company Web Site - Windows Internet Explorer
E:\H4D6e\Sub3HTML&Screencaps\10fig01
Live Search
File
Edit
View
Favorites
Tools
Help
Company Web Site
Page
Tools

Our Company

This is some sample text for a company. This is a link to a Web site: Ed's page. Here are some links to other parts of this Web page:

- Company History
- Services
- How we work
- How much we cost

Company History

We were founded at the beginning of time, most active in the middle of time, and terminated at the end of time.

Services

We come in, make a lot of noise, drink all your coffee, eat all your snacks, then leave.

Done | Computer | Protected Mode: Off | 100%

Figure 10-1: The font family for hyperlinks differs from the font family for the rest of the text.

Sizing

The following properties allow you to control the dimensions of your text.

Font size

The style declaration to specify the size of text is

```
selector {font-size: value;}
```

The `value` of the declaration can be

- One of the standard font-property measurement values (listed in Chapter 8)
- One of these user-defined *keywords:*

  `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, or `xx-large`

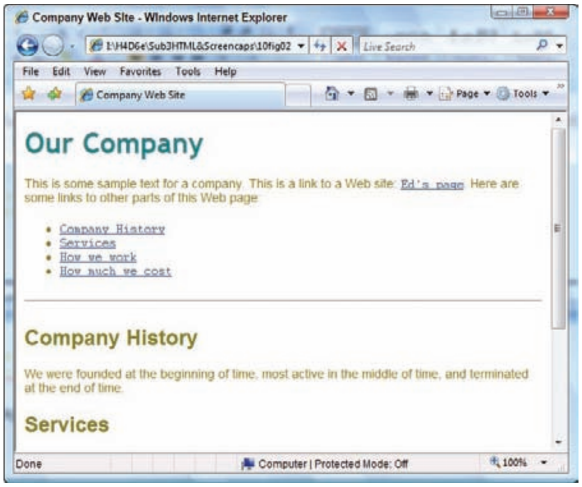
  The value of each keyword is determined by the *browser,* not by the style rule.

The rules listed in upcoming subsections define

- A relative font value for all text
- An absolute value for the font size for all first-level headings

```
body {color: #808000; font-family: Arial, sans-serif; font-size: 85%;}
h1 {font-family: "trebuchet ms", verdana, geneva, arial, helvetica,
sans-serif; font-size: 24pt; line-height: 28pt; color: teal;}
```

The result appears in Figure 10-2.


Company Web Site - Windows Internet Explorer
I:\H4D6e\Sub3HTML&Screencaps\10fig02
Live Search
File Edit View Favorites Tools Help
Company Web Site
Page Tools
Done
Computer | Protected Mode: Off
100%

Our Company

This is some sample text for a company. This is a link to a Web site: Ed's page Here are some links to other parts of this Web page

- Company History
- Services
- How we work
- How much we cost

Company History

We were founded at the beginning of time, most active in the middle of time, and terminated at the end of time.

Services

Figure 10-2: First-level headings are 24 points tall; font size for other text is relative.

Line height

The *line height* of a paragraph is the amount of space between each line within the paragraph.

TIP

Line height is like line spacing in a word processor.

To alter the amount of space between lines of a paragraph, use the `line-height` property:

```
selector {line-height: value;}
```

The value of the `line-height` property can be either

- One of the standard font property measurement values (listed in Chapter 8)
- A number that multiplies the element's font size, such as `1.5`

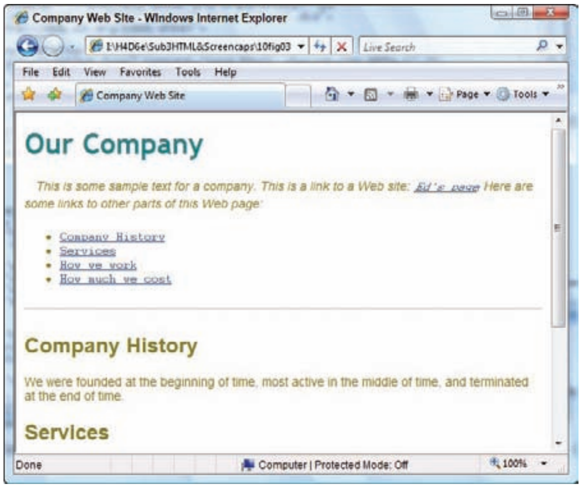
We assign a `quotation` class to the first paragraph throughout this chapter so you can see the changes. This allows us to apply these styles to the first paragraph by using

```
<p class="quotation">
```

in the HTML document.

The following rules style the first paragraph in italics, indent that paragraph, and increase the line height to increase readability (see Figure 10-3):

```
body {color: #808000; font-family: Verdana, sans-serif; font-size: 85%;}
.quotation {font-style: italic; text-indent: 10pt; line-height: 150%;}
```


Company Web Site - Windows Internet Explorer
I:\H4D6e\Sub3HTML&Screencaps\10fig03
Live Search
File Edit View Favorites Tools Help
Company Web Site
Page Tools
Done
Computer | Protected Mode: Off
100%

Our Company

This is some sample text for a company. This is a link to a Web site: Ed's page Here are some links to other parts of this Web page:

- Company History
- Services
- How we work
- How much we cost

Company History

We were founded at the beginning of time, most active in the middle of time, and terminated at the end of time.

Services

Figure 10-3: Any element that belongs to the `quotation` class gets the same formatting.

Character spacing

You can *increase* or *reduce* the amount of spacing between letters or words by using these properties:

- `word-spacing`: The style declaration for `word-spacing` is

  ```
  selector {word-spacing: value;}
  ```

  Designers call the space between words *tracking*.
- `letter-spacing`: The style declaration for `letter-spacing` is

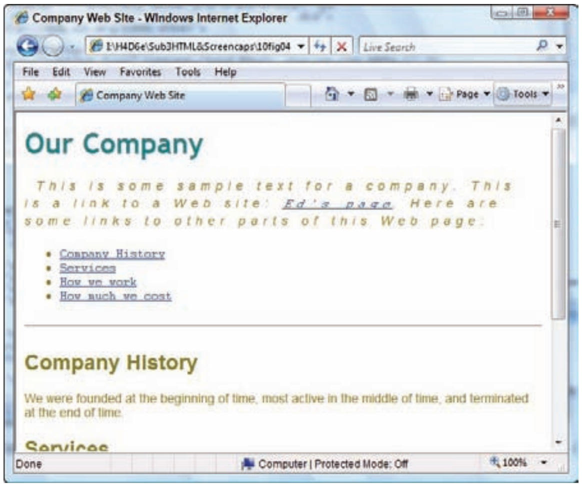
  ```
  selector {letter-spacing: value;}
  ```

  Designers call the space between letters *kerning*.

The value of either spacing property must be a *length* defined by a standard font property measurement value (listed in Chapter 8).

The following code increases the letter spacing (kerning) of the first paragraph (see Figure 10-4):

```
body {color: #808000; font-family: Verdana, sans-serif; font-size: 85%;}
.quotation {font-style: italic; text-indent: 10pt; letter-spacing: 6px;}
```


Company Web Site - Windows Internet Explorer
I:\H4D6e\Sub3HTML&Screencaps\10fig04
Live Search
File Edit View Favorites Tools Help
Company Web Site
Page
Tools
Done
Computer | Protected Mode: Off
100%

Our Company

This is some sample text for a company. This is a link to a Web site: Ed's page. Here are some links to other parts of this Web page:

- Company History
- Services
- How we work
- How much we cost

Company History

We were founded at the beginning of time, most active in the middle of time, and terminated at the end of time.

Services

Figure 10-4: Kerning can be larger or smaller than the font's normal spacing.

Positioning

Alignment properties allow you to control how the edges of text blocks line up against one another (otherwise known as "edge alignment").

Alignment

Alignment determines whether the left and right sides of a text block are

- **Flush:** Starting or ending together
- **Ragged:** Starting or ending at different points

Syntax

Alignment is defined with the `text-align` property. The style declaration to align text is as follows:

```
selector {text-align: value;}
```

The value of the `text-align` property must be one of the following keywords:

- `left` **aligns the text to the left.** The right side of the text block is *ragged.*
- `right` **aligns the text to the right.** The left side of the text block is *ragged.*
- `center` **centers the text in the middle of the window.** Both sides of the text block are *ragged.*
- `justify` **aligns the text for both the left and right side.** The spacing within the text in each line is adjusted so both sides of the text block are *flush.*

  WARNING! Justifying text affects letter or word spacing in the paragraph. Test the results before displaying your Web pages to the world.

Markup

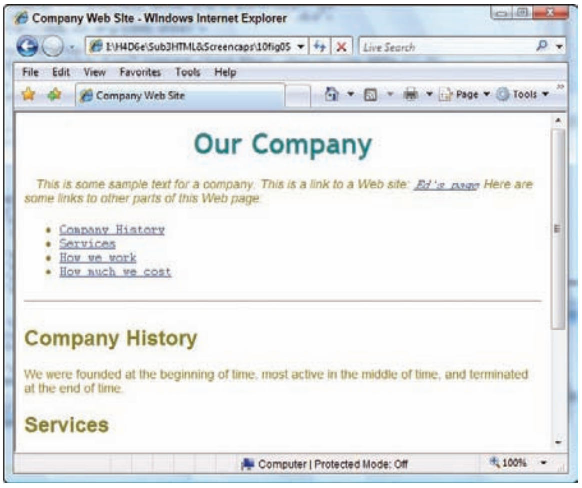
The following example defines the alignment for the first-level heading and the first paragraph (see Figure 10-5):

```
body {color: #808000; font-family: Verdana, sans-serif; font-size: 85%;}
h1 {color: teal; font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;
    font-weight: 800; font-size: 24pt; line-height: 30 pt; text-align: center}
.quotation {font-style: italic; text-indent: 10pt; text-align: left;}
```

Indent

You can define the amount of space that should precede the first line of a paragraph by using the `text-indent` property.

TECHNICAL STUFF This doesn't indent the whole paragraph. That requires CSS box properties, such as `margin-left` and `margin-right` (see Chapter 9).


Company Web Site - Windows Internet Explorer
I:\H4D6e\Sub3HTML&Screencaps\10fig05
Live Search
File Edit View Favorites Tools Help
Company Web Site
Page Tools

Our Company

This is some sample text for a company. This is a link to a Web site: Ed's page Here are some links to other parts of this Web page

- Company History
- Services
- How we work
- How much we cost

Company History

We were founded at the beginning of time, most active in the middle of time, and terminated at the end of time.

Services

Computer | Protected Mode: Off
100%

Figure 10-5: The first-level heading is centered; the first paragraph is aligned to the left.

Syntax

The style declaration used to indent text is

```
selector {text-indent: value;}
```

Here `value` must be one of the standard length-property measurement values (listed in Chapter 8).

Markup

As seen in this chapter, the `quotation` class has a `text-indent` of 10 points.

```
body {color: #808000; font-family: Verdana, sans-serif; font-size: 85%;}
.quotation {font-style: italic; text-indent: 10pt;}
```

Text treatments

CSS allows you to decorate your text by using boldface, italics, underline, overline, or line-through, and even allows your text to blink (when that's supported by browsers).

Bold

Using a boldface font is one of the more common text embellishments a designer uses. To apply boldface in HTML, use the <b> tag. However, CSS provides you with more control over the font weight of the bolded text.

Syntax

This style declaration uses the `font-weight` property:

```
selector {font-weight: value;}
```

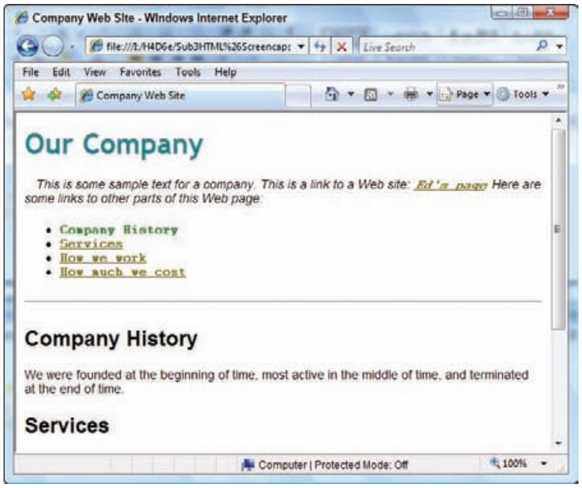
The value of the `font-weight` property may be one of the following:

- `bold`: Renders the text in an average bold weight
- `bolder`: Relative value that renders a font weight bolder than the current weight (possibly assigned by a parent element)
- `lighter`: Relative value that renders a font weight lighter than the current weight (possibly assigned by a parent element)
- `normal`: Removes any bold formatting
- One of these integer values: `100` (lightest), `200`, `300`, `400` (normal), `500`, `600`, `700` (standard bold), `800`, `900` (darkest)

Markup

The following example bolds hyperlinks (see Figure 10-6), and turns the underline off and changes the color to green once a link is visited (we did this to the Company History item to show you what it looks like):

```
body {color: black; font-family: Arial, Verdana, sans-serif; font-size: 85%;}
a {font-weight: bold;}
a:link {color: olive; text-decoration: underline;}
a:visited {color: green; text-decoration: none;}
```


Company Web Site - Windows Internet Explorer
file:///I:/H4D6e/Sub3HTML%26Screencap:
Live Search
File Edit View Favorites Tools Help
Company Web Site
Page Tools
Computer | Protected Mode: Off
100%

Our Company

This is some sample text for a company. This is a link to a Web site: Ed's page Here are some links to other parts of this Web page.

- Company History
- Services
- How we work
- How much we cost

Company History

We were founded at the beginning of time, most active in the middle of time, and terminated at the end of time.

Services

Figure 10-6: All hyperlinks are bolded.

Italic

Italics are commonly used to set off quotations or to emphasize text. To apply italics in HTML, use the `<i>` tag. However, CSS provides you with more control over the font style of text through the `font-style` property.

Syntax

This style declaration uses the `font-style` property:

```
selector {font-style: value;}
```

The value of the `font-style` property may be one of the following:

- `italic`: Renders the text in *italics* (a special font that usually *slants*).
- `oblique`: Renders the text as *oblique* (a slanted version of the normal font).
- `normal`: Removes any italic or oblique formatting.

Markup

The following example assigns an italic font style to the first-level heading:

```
body {color: #808000; font-family: Verdana, sans-serif; font-size: 85%;}
h1 {color: teal; font-family: "MS Trebuchet", Arial, Helvetica, sans-serif;
              text-transform: uppercase;
    font-style: italic; font-weight: 800; font-size: 24pt; line-height: 30pt;
              text-align: center;}
```

Capitalization

You use the `text-transform` property to set capitalization in your document.

Syntax

This style declaration uses the `text-transform` property:

```
selector {text-transform: value;}
```

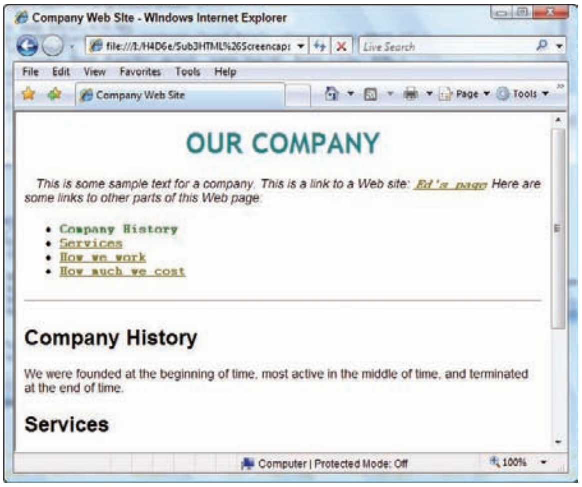
The value of the `text-transform` property may be one of the following:

- `capitalize`: Capitalizes the first character in every word.
- `uppercase`: Renders all letters of the text of the specified element in uppercase.
- `lowercase`: Renders all letters of the text of the specified element in lowercase.
- `none`: Keeps the value of the inherited element.

Markup

The following example renders the first-level heading in uppercase (shown in Figure 10-7):

```
body {color: #808000; font-family: Verdana, sans-serif; font-size: 85%;}
h1 {color: #808000; font-family: Arial, Helvetica, sans-serif;
    font-weight: 800; font-size: 24pt;
    text-align: center; text-transform: uppercase;}
```


Company Web Site - Windows Internet Explorer
file:///I:/H4D6e/Sub3HTML%26Screencaps
Live Search
File Edit View Favorites Tools Help
Company Web Site
Page
Tools
Computer | Protected Mode: Off
100%

OUR COMPANY

This is some sample text for a company. This is a link to a Web site: Ed's page Here are some links to other parts of this Web page.

- Company History
- Services
- How we work
- How much we cost

Company History

We were founded at the beginning of time, most active in the middle of time, and terminated at the end of time.

Services

Figure 10-7: The first-level heading is rendered in all uppercase.

The text-decoration property

The `text-decoration` property allows for text formatting that's a tad crazier. It isn't used often.

Syntax

This style declaration uses the `text-decoration` property:

```
selector {text-decoration: value;}
```

The value of the `text-decoration` property may be one of the following:

- `underline`: Underlines text.
- `overline`: Renders the text with a line over it.
- `line-through`: Renders the text with a line through it.
- `blink`: Blinks the text on the screen.

  WARNING! Are you *sure* you want blinking text?

  - `blink` isn't supported by all browsers.
  - `blink` can be dreadfully annoying and distracting.
- `none`: Removes any text decoration.

Markup

The following example changes the link when the mouse hovers over it. In this case, it turns off any underlining for a link:

```
body {color: #808000; font-family: Verdana, sans-serif; font-size: 85%;}
a:link {color: olive; text-decoration: underline;}
a:visited {color: olive; text-decoration: underline;}
a:hover {color: olive; text-decoration: none;}
```

The catchall font property

Many font properties can be summarized in one style declaration by using the shorthand `font` property. When it's used, only one style rule is needed to define a combination of font properties:

```
selector {font: value value value;}
```

The value of the `font` property is a list of any values that correspond to the various font properties:

- The following values must be defined in the following order, though they need not be consecutive:
  - `font-size` (required)
  - `line-height` (optional)
  - `font-family` (required)
- The `font-family` value list must end with a semicolon.
- Use commas to separate multiple font family names.
- The following values are optional and may occur in any order within the declaration. Individual values are separated by spaces:
  - `font-style`
  - `font-variant`
  - `font-weight`

For example, you can use the following style declaration to create a specific style for a first-level heading:

```
h1 {font: italic bold 150% Arial, Helvetica, sans-serif;}
```

Chapter 11

Using Tables to Jazz Up Your Pages

In This Chapter

- Understanding table benefits
- Mapping a table design
- Building simple tables
- Stretching items across multiple cells
- Adding images, hyperlinks, or text to table cells
- Table tips and techniques

Historically, tables contain and lay out data in a grid to make analysis easier. In (X)HTML, however, they serve an entirely different purpose — to control Web-page layout. Most Web pages contain at least one table — some even nest tables within tables. (X)HTML tables can present everything from text to images on your pages efficiently and attractively.

Also, CSS provides ample positioning power to give designers flexibility and precision when working with tables — a killer combination!

This chapter provides step-by-step instructions for building and using (X)HTML tables and then using CSS to control their presentation. Use our best tried-and-true tips and techniques to speed up and simplify your efforts.

What Tables Can Do for You

Putter around the Internet for a while and you'll soon discover that lots of sites — such as Amazon.com, eBay, Yahoo!, and Google, to name a few — use tables to display their content, even if you can't see them in an obvious way. In fact, such invisible tables dominate the Web. Two ideas drive all this usage:

- Using tables to arrange items on your Web page.
- Turning borders off so users can't see these tables.

REMEMBER

By nature, Web pages start out linear (that is, they proceed from start to end in a straight line of paragraphs and other text). Tables allow you to step out of linear mode and put text and images in more interesting places on a page.

You can use tables in a couple of ways:

- **Traditional (ho-hum) method:** You can define table or individual cell widths by using absolute numbers. This type of table doesn't resize when users resize their browser windows.

  Some designers prefer to use tables for traditional purposes — to present data — a straightforward approach that's easily tackled.
- **Presentation-focused (wow) method:** You can define table and cell width by using *percentages.* This table resizes itself when users resize their browser windows.

  Many designers perform creative, complex tricks with their tables.

Use (X)HTML and CSS in tables

(X)HTML tables can require some finesse to make them do just what you need. Only three percent of Internet surfers use browsers that don't support CSS. This means tables make a safe basis for page design. We recommend using

- (X)HTML tables to lay the basic foundation for your page
- CSS to provide additional table formatting

If you know your target users use updated browsers, you can use CSS for all your positioning needs. For example, if you're designing an intranet Web site for a group of computer programmers, you can require that its viewers use only a newer browser and eliminate table elements completely, if you like.

Although this chapter covers all aspects of HTML tables, it focuses on layout tips and techniques.

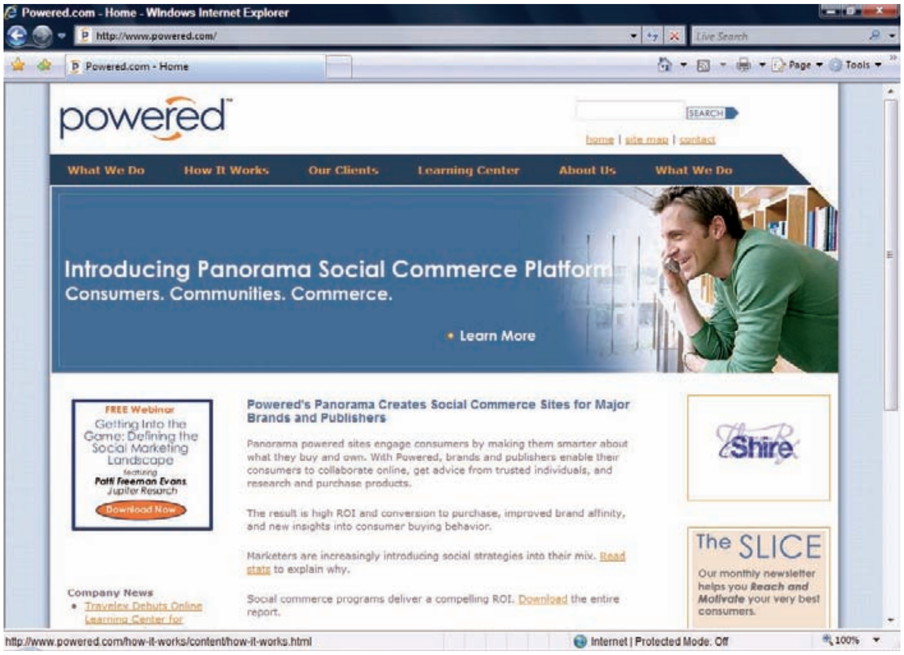
When you use tables for layout, they can result in a couple of outcomes:

- Tables can produce complex layout structures, as shown in Figure 11-1. (Some other examples of complex tables are viewable at `www.amazon.com` and `www.yahoo.com`.)

  After you open these Web pages in your Web browser, look at each page's HTML source code (try View➪Source from your menu bar). Observe how complex the markup is, and mark ye well when the markup looks haphazardly arranged (alas, if only they'd asked us . . .).

- Some Web-page design models keep the interface simple with the less-is-more approach — and therefore they're easy to use. Figure 11-2 shows the simple approach.

  `www.google.com` uses a simple table to arrange navigation.



Powered.com - Home - Windows Internet Explorer
http://www.powered.com/
Live Search
Powered.com - Home
Page
Tools
powered
SEARCH
home | site map | contact
What We Do
How It Works
Our Clients
Learning Center
About Us
What We Do
Introducing Panorama Social Commerce Platform
Consumers. Communities. Commerce.
Learn More
FREE Webinar
Getting Into the Game: Defining the Social Marketing Landscape
featuring
Patti Freeman Evans
Jupiter Research
Download Now
Shire
The SLICE
Company News
Travelex Debuts Online Learning Center for
http://www.powered.com/how-it-works/content/how-it-works.html
Internet | Protected Mode: Off
100%

Powered's Panorama Creates Social Commerce Sites for Major Brands and Publishers

Panorama powered sites engage consumers by making them smarter about what they buy and own. With Powered, brands and publishers enable their consumers to collaborate online, get advice from trusted individuals, and research and purchase products.

The result is high ROI and conversion to purchase, improved brand affinity, and new insights into consumer buying behavior.

Marketers are increasingly introducing social strategies into their mix. Read stats to explain why.

Social commerce programs deliver a compelling ROI. Download the entire report.

Our monthly newsletter helps you *Reach and Motivate* your very best consumers.

Figure 11-1: This Web page uses various tables for layout.

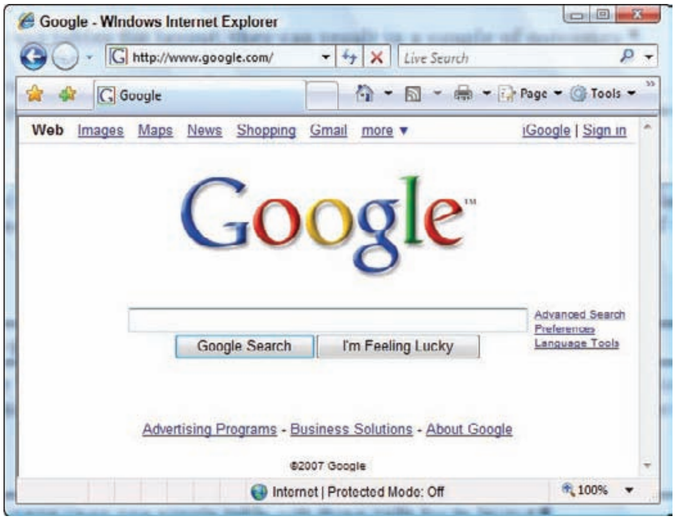

Google - Windows Internet Explorer
http://www.google.com/
Live Search
Google
Page
Tools
Web Images Maps News Shopping Gmail more ▾
iGoogle | Sign in
Google
Advanced Search
Preferences
Language Tools
Google Search
I'm Feeling Lucky
Advertising Programs - Business Solutions - About Google
©2007 Google
Internet | Protected Mode: Off
100%

Figure 11-2: This Web page uses one simple table with three cells for its layout.

Table Basics

All the complexity of HTML tables builds from three basic elements:

- **Borders:** Every basic table must always have *exactly* four borders that make up a rectangle.
- **Cells:** These are individual areas (spaces) inside the borders of a table.
- **Cell span:** Within that four-walled structure, you can delete or add cell walls (as shown with the cells on the right side of the table in Figure 11-3). When you delete cell walls, you make a cell *span* multiple rows or columns — and that's exactly what makes tables flexible tools for layout.

TIP

Cell spanning and cell width work differently:

- When you *span* cells, you change cell space by combining, or merging, cells. This step removes cell walls.
- When you *increase the width* of a cell, you only change the space within that cell.

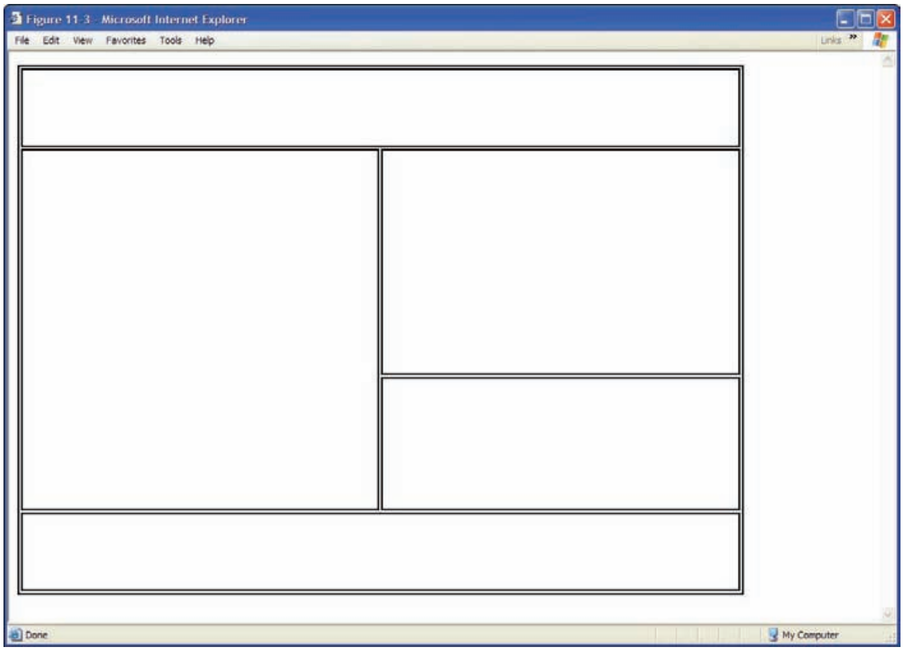


Figure 11-3 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links
Done
My Computer

Figure 11-3: You can use an HTML table to lay out Web pages.

Sketching Your Table

Tables can become complex. You need to plan them carefully. Mapping to the nearest pixel can grow tedious and can take several attempts, but it's an essential step in designing a well laid-out page.

Developing layout ideas

Start with a general idea and slowly plan your layout until it becomes more solid and specific. Follow these basic steps:

1. **Grab (believe it or not) a sheet of paper and a pencil so you can sketch out your ideas.**

   Start with a *general* idea of where you want everything to go on your page.

2. **Evaluate what to include in your Web page and decide on the overall layout.**

   This way, you can begin to determine

   - How many columns and rows you need
   - The width of the table and cells
   - Whether to make any cells span rows or columns

   The following design choices are yours to make:

   - Whether the table will be centered, left-aligned, or right-aligned (relative to the browser window within which it appears).
   - Whether you want to include hyperlinks — and where you might want to include them.

     For example, many Web sites, such as the one in Figure 11-1, include a logo image that provides a hyperlink to the site's home page, so no matter what page you're on, you can always get back to the front door.

3. **Figure out the pixel dimensions of images you want to use.** Make sure that the table fills a browser window nicely without forcing the user to scroll left and right to see everything.

   Decide between using text or images for navigation, as follows:

   - If you want more font control over your navigation, consider using images for your navigational items.

     The font is embedded in the image; therefore the user's browser settings can't override the font you choose.

   - If you don't need additional font controls, use textual navigation.
   - Decide where the main logo should go and what size it should be.

     In Figure 11-2, the logo is the main focal point. Its dimensions are 276 pixels wide by 110 pixels tall.

   Concentrate on managing the width of the table. Let the contents of your table determine the cell height. Height is less important because users are familiar with scrolling up and down Web pages. However, they may get frustrated by scrolling left and right to read content.

Drafting the table

When you know how big and how numerous your design elements are, you can sketch a rough table on paper.

If you opt for a simple approach, each main element (logo, hyperlink image, and navigation) gets its own cell. In Figure 11-4, that means three cells. If you have only a few cells, you'll probably have to span some so their contents fills the width of your page.

- A complex design may need several rows.
- A simple, clean design (such as the one in Figure 11-4) may require only two rows.

  Figure 11-4 shows the final sketch for your table:

  - The first row *spans* both columns.
  - The second row contains two *separate* columns.

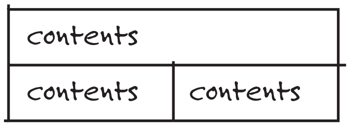


Figure 11-4: Start by sketching the table dimensions, even before opening a text editor.

The author of our sample Web site uses images in place of text for the navigational elements; however, for usability reasons, try using text in place of images when possible. Even so, if you want complete control of the font(s) in which your text appears, you may have to use images — and create an *image* of the text written in your chosen font.

Constructing Basic Tables

When you have a sketch that gives a solid indication of page and table layout, you can open your HTML editor and create a skeleton for your table.

Components

The markup building blocks for your table's framework are the three basic components for any table:

- **Table:** <table>
- **Table row:** <tr>

  <tr> is always enclosed within <table>.
- **Table (data) cell:** <td>

  <td> is always enclosed within <tr>.

With these three elements, you can build a simple table. Think of their relationship and composition like this: Each table contains one or more table *rows*, and each table row contains one or more table *cells*. Usually, a table will contain at least two rows, often more, and at least two cells — nearly always more — why else would you need or want to use a table?

REMEMBER

The <table>, <tr>, and <td> opening and closing tags are required. If you forget to include any, your table won't display correctly in most browsers.

Layout

Tables come in many forms and at varying levels of complexity. A simple two-dimensional data table that's part of a Web page is easier to design and implement than a more complex table layout that contains an entire Web page. As you read through the following sections, you will see and appreciate this distinction clearly.

Creating a simple table

The <table> tag and its markup typically appear between the <body> tags in your document. However, you can also use them within most block elements and within the <td> and <th> tags to nest tables. (See the "Nesting tables within tables" section later in this chapter.) Use the following markup to create a simple table with two rows and two columns (four data cells) — replacing cell 1, cell 2, and so on with your text:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Tables</title>
</head>
<body>
 <table>
  <tr>
    <td> cell 1 </td>
    <td> cell 2 </td>
  </tr>
  <tr>
    <td> cell 3 </td>
    <td> cell 4 </td>
    </tr>
  </table>
</body>
</html>
```

The preceding example creates a table with two rows based on the sketch in Figure 11-4. The first table row encloses cells 1 and 2; the second table row encloses cells 3 and 4.

Table rows always run horizontally, and the contents of each cell — in this case, `cell 1`, `cell 2`, and so on — are contained within their own `<td>` element. Don't forget that you must close your table tags, or your table will not display correctly.

Creating a table-based Web page

To create the shell of your table-based Web page (for example, one based on the sketch from the preceding section, Figure 11-4), follow these steps:

1. **Start with the `<table>` element:**

   ```
   <table>
     ...
   </table>
   ```

   The `<table>` element can have a number of optional attributes (for example, `border="1"` or `bgcolor="black"`) — for now, however, keep it simple.

2. **Decide how many *rows* you want the table to have:**

   The following markup creates a table with two rows:

   ```
   <table>
     <tr>...</tr>
     <tr>...</tr>
   </table>
   ```

3. **Create *cells* in each row with the table data cell (`<td>`) element.**

   Each `<td>` element creates a cell, so the number of `<td>` elements in a row is the number of columns.

   The sketch in Figure 11-4 shows a two-column table with *three* cells: the first row contains one cell, and the second row contains two cells. The markup for this arrangement looks like this:

   ```
   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
         "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
   <html xmlns="http://www.w3.org/1999/xhtml">
   <head>
       <title>Tables</title>
   </head>
   <body>
   ```

```
<table>
  <tr>
    <td> contents </td>
  </tr>
  <tr>
    <td> contents </td>
    <td> contents </td>
  </tr>
</table>
</body>
</html>
```

Here's where tables can get a bit tricky. A simple table with an even number of rows and columns (say two rows and two columns) is a piece of cake — but as you get handier at designing your own pages, you'll discover that your needs aren't likely to produce such symmetrical tables very often. If your cell will span more than one row or column (such as the first cell in the preceding example), you have to add an attribute that tells the browser which cell does the spanning.

The number in the attribute corresponds to the number of columns or rows you want the cell to span, which means if you're creating a table like the one in our example, you have to add the `colspan="2"` attribute to the first `<td>` element. (The first cell in the table spans two columns.)

See the section, "Adding Spans," later in this chapter for more information. But for now, assume that you're creating a table like ours. The markup looks like this:

```
<table>
  <tr>
    <td colspan="2"> contents </td>
  </tr>
  <tr>
    <td> contents </td>
    <td> contents </td>
  </tr>
</table>
```

Congratulations — you're done with your first table. Well, sort of. To effectively use tables for layout, you need to know how to control several display issues, such as borders, table widths, and the handling of white space within your table. (For example, without borders, you can't really tell the table is there — it won't show up in your browser. This isn't a bad thing or a good thing per se, but something that you can change if you want your borders to show up in browsers.) Keep reading for more information on completing your table and integrating it into your page.

Other table elements

Although tables were invented to contain and display data, they may also be used to control Web-page layout, though many professionals now prefer to use `<div>` tags with CSS and classes or IDs to really tweak and customize layouts. This chapter focuses on the table elements that you can use to control layout, if you like. If you want to create a traditional table, you can use the following table elements:

- `<th>`: The table header element displays text in boldface with a default center alignment.

  You can use the `<th>` element within any row of a table, but you most often find and use it in the first row at the top — or head — of a table. Except for their position and egotism, they act just like table data (`<td>`) tags and should be treated as such.

- `<caption>`: This is the table caption element. It is designed to exist anywhere inside the `<table> . . . </table>` tags but not inside table rows or cells (because then they wouldn't be captioning anything). This element can occur only once per table.

  Similar to table cells, captions accommodate any HTML elements that can appear in the body of a document (in other words, inline elements), but only those. By default, captions are horizontally centered with the table, and their lines wrap to fit within the table's width. The `<caption>` element accepts the `align` attribute.

- `<tbody>`: You can group table rows into a table body section with the table body (`<tbody>`) element.

  A newer element in the HTML 4 specification, `<tbody>` allows table bodies to scroll independently of the table head (`<thead>`) and table foot (`<tfoot>`). The table body should contain rows of table data. The `<tbody>` element must contain at least one table row (`<tr>`).

- `<thead>`: You can group table rows into a table head section by using the table head (`<thead>`) element. The table head contains information about the table's columns.

  The `<thead>` element must contain at least one table row.

- `<tfoot>`: Much like the `<thead>` element, you can group table rows into a table footer section by using the table footer (`<tfoot>`) element. The table footer contains information about the table's columns and must contain at least one table row.

  Include your footer information before the first instance of the `<tbody>` element so that the browser renders that information before taking a stab at all the content data cells.

- `<colgroup>`: This element creates an explicit column group. You specify the number of columns by using the `span` attribute or by using the `<col>` element, which we define shortly.

  You use the `span` attribute to specify a uniform width for a group of columns.

- `<col>`: The `<col>` element is an empty element. You use the `<col>` element to further define column structure. The `<col>` element shouldn't be used to group columns — that's the `<colgroup>` element's job. You use the `<col>` element after you define a column group and set a uniform width; it specifies a uniform width for a subset of columns.

Adding borders

A table border defines the outer edge of the table.

REMEMBER

When the table is used to arrange elements on a page, you don't want a visible border. There are two ways you can turn a table border on or off:

- Set the `border` attribute within the `<table>` element. The value of the `border` attribute must be an integer that defines the border thickness in pixels.

  To turn the border off, set the `border` attribute equal to `0`: `<table border="0">`
- Define a border using the CSS border properties.

  You may define the border style, width, or color by using CSS. (See the later section, "Using CSS border properties.")

Using the (X)HTML border attribute

For an (X)HTML table, *border* refers to both

- Outside borders
- Individual cell borders

You use the `border` attribute to turn all these table borders on *or* off.

To turn on the table (and cell) border, add the `border` attribute to the `<table>` start tag, as shown in the following bold markup:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Tables</title>
</head>
<body>
 <table border="1">
  <tr>
    <td colspan="2"> contents </td>
  </tr>
  <tr>
    <td> contents </td>
    <td> contents </td>
  </tr>
 </table>
</body>
</html>
```

The value of the border attribute defines the thickness of the border in pixels. For example, border="5" produces a 5-pixel border. If you leave this attribute off, most browsers don't display a border. If you don't want your border visible, however, then we suggest that you add border="0" to turn off the border for sure.

Where clear delineation between cell contents is desirable (as with price charts, real data tables, and other collections of text or numerical data), borders help visitors break what they're seeing into separate bits of information. But when a table is used to organize a Web page that all hangs together nicely, turning borders off can help to reinforce this cohesiveness.

WARNING!

By default, most browsers use an invisible 2-pixel border on tables. When you design your table, you should do one of the following:

- Allow for those invisible 2-pixel borders in your design.
- Configure your own borders.
- Eliminate the border by setting the border attribute to equal 0 (border="0").

TIP

Turn on the table border when you're first creating and tweaking your table. Sometimes it's difficult to see just what is going on without a border. After you've finished tweaking your table, you can turn off the border.

Using CSS border properties

Unlike the (X)HTML border attribute, CSS lets you define border styles for any or all border edges. For example, you can define a dotted gray border for the left side of the table and leave the rest of the table's border invisible.

Style

As you might expect, the border-style property allows you to define the style (such as dotted or solid) of the border.

The style declaration used to add a border style is as follows:

```
selector {border-style: value;}
```

The value for the border-style property must be one of the predefined keywords:

- dotted
- dashed
- solid
- double

- `groove`
- `ridge`
- `inset`
- `outset`

To create a solid border (for example), use the following style declaration:

```
table {border-style: solid;}
```

Width

Similar to using the (X)HTML `border` attribute, you can define the border width in pixels. However, CSS provides you with additional width value data types to choose from. Here's the style declaration used to add a border width:

```
selector {border-width: value;}
```

The value for the `border-width` property can be

- **A predefined keyword:** `thin`, `medium`, or `thick`.
- **An absolute or relative length:** See the Chapter 8 sidebar, "Property measurement values," for more information. The values described in that sidebar are relevant to HTML as well as to CSS.

To set the width of a border to 1pixel, use the following style declaration:

```
table {border-width: 1px;}
```

Color

The style declaration used to define a border color is:

```
selector {border-color: value;}
```

The value for the `border-color` property must be defined using a predefined color name or a hexadecimal value:

- **Color name:** `aqua`, `black`, `blue`, `fuchsia`, `gray`, `green`, `lime`, `maroon`, `navy`, `olive`, `purple`, `red`, `silver`, `teal`, `white`, or `yellow`.
- **Hexadecimal value:** See Chapter 10.

To set the color of a border to black, use the following style declaration:

```
table {border-color: black;}
```

Using the catchall border property

Similar to defining font properties, you can use the shorthand border property to define multiple style rules at once:

```
table {border: 1px solid gray;}
```

There are five catchall border properties that you can use for a table or a box:

- border: Defines formatting for all four sides.
- border-left: Defines formatting for the left side.
- border-right: Defines formatting for the right side.
- border-top: Defines formatting for the top.
- border-bottom: Defines formatting for the bottom.

TIP

The border properties aren't only for use with tables, they're part of the CSS *box model* (which identifies the area on the page to which declarations apply, including edge, margin, border, and padding values for their top, right, bottom, and left edges). They can define borders for almost any (X)HTML element, as long as it isn't an inline element.

Adjusting height and width

Most browsers determine the width of the table cells by judging the content of the cells (images and/or text).

WARNING!

The browser provides as much space as possible to contain the content. However, there are limits for both images and text:

- Side-by-side images must fit in the width of the browser window.

  For example, if you have an image that is 200 pixels wide, the cell expands to accommodate the image. However, if you have several cells in a row, each with images over 400 pixels wide, the cells only expand as far as the browser window allows.

- Text may expand and distort the layout.

  If a cell contains a lot of text, the cell expands as far as it can until the first line break or the end of the text. That might make for a very unattractive table.

Where tables are used to help control layout, controlling the width of cells and the table is important. You have two ways to control width:

- Use the (X)HTML `width` attribute within the `<table>` or `<td>` element.
- Assign a width value to a `<table>` or `<td>` element using the CSS `width` property.

The (X)HTML width attribute

If you don't set table and cell width, the user's browser determines the width of every cell according to the width of its contents — no more, no less.

For example, suppose you want to put a logo in the first cell and navigational items in the cell to its left. If you don't assign the width to the first cell (containing the logo), the navigational items are placed right beside the logo, with no or almost no space between the two. To avoid that cramped look, you can use the `width` attribute to strategically define an exact number of pixels between the logo and navigational items.

If you're using tables for layout purposes, we recommend that you set the width for the table and cells.

Syntax

Defining width is easy when you use the `width` attribute. For example, you can set the width of your table at 630 pixels like this:

```
<table border="1" width="630">
…
</table>
```

The value of the `width` attribute can be defined in either

- Pixels (a positive integer, such as `630`)

  This is an absolute value.
- Percentage of the display area's width (a positive integer followed by a percent sign, such as `95%`)

  This is a relative value that allows your table to be resized depending on the size of the browser window.

These values can also set the width of individual cells.

Markup

To add widths to the table built earlier in this chapter (and to set width for its individual cells), add the following markup shown in bold text:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Tables</title>
</head>
<body>
 <table border="1" width="630">
  <tr>
    <td colspan="2" width="630"> contents </td>
  </tr>
  <tr>
    <td width="400"> contents </td>
    <td width="230"> contents </td>
  </tr>
 </table>
</body>
</html>
```

Figures 11-5 and 11-6 show the difference between a site that doesn't define table and cell widths and one that uses the `width` attribute.

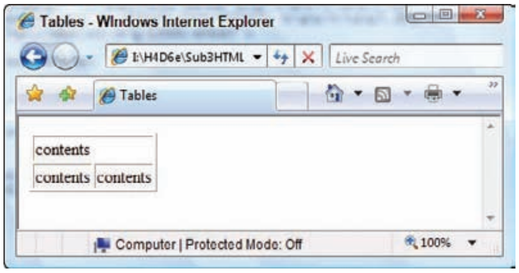

Tables - Windows Internet Explorer
I:\H4D6e\Sub3HTML
Live Search
Tables
contents
contents contents
Computer | Protected Mode: Off
100%

Figure 11-5: This image doesn't define width properties.

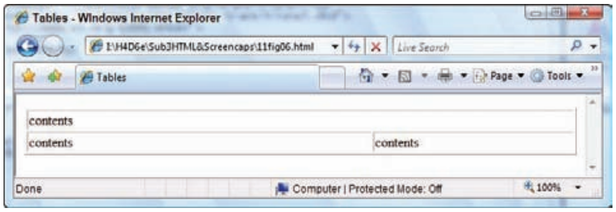

Tables - Windows Internet Explorer
I:\H4D6e\Sub3HTML&Screencaps\11fig06.html
Live Search
Tables
Page
Tools
contents
contents
contents
Done
Computer | Protected Mode: Off
100%

Figure 11-6: This image defines width properties.

WARNING!

If you set the pixel width smaller than the content's pixel size, the browser ignores the `width` attribute and defaults to display all cell contents. So please: check all your dimensions!

The CSS width property

The style declaration used to define width is:

```
selector {width: value;}
```

The value for the `width` property must be either

- `auto`

  This keyword allows the browser to determine the necessary width.
- An absolute or relative length:

  See the Chapter 8 sidebar "Property measurement values" for more information.

To set the width of the table displayed in Figure 11-6, use the following style declarations:

```
table {width: 630px;}
td.cellone {width: 630px;}
td.celltwo {width: 630px;}

td.cellthree {width: 630px;}
```

Padding and spacing

Determining the white space between cells is essential for proper layout. Keeping in mind the sketch from Figure 11-4, you have to determine — to the pixel — how space will be used in your table.

(X)HTML attributes

Two attributes can help you define white space by putting some space between cells: `cellpadding` and `cellspacing`. These attributes use two different techniques to put some space between cells:

- `cellspacing` adds space *between* cells (the border width is adjusted).
- `cellpadding` adds space *inside* a cell (within the cell walls).

The value for either attribute is defined in *pixels.* For example, `cellpadding="5"` adds 5 pixels' worth of padding to each cell.

To define either attribute, add it to the `<table>` start tag, as follows:

```
<table cellpadding="5" cellspacing="5">
```

When using tables for layout, without visible borders, it doesn't matter much which attribute you use. However, if you add color to your tables — or use a border for any reason — you can see a considerable difference. That's because `cellpadding` increases the space within the border, and `cellspacing` increases the width of the border itself, as shown clearly in Figures 11-7 and 11-8.

WARNING!

The default value for `cellpadding` is 1; the default for `cellspacing` is 2. If you don't define `cellpadding` and `cellspacing`, your users' browsers assume the defaults. Accounting for those pixels in your sketch is a good idea, unless you set those values explicitly to zero.

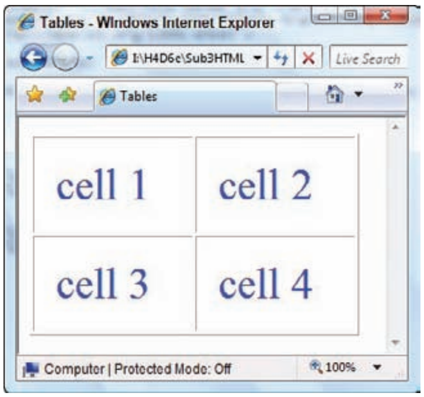

Tables - Windows Internet Explorer
I:\H4D6e\Sub3HTML
Live Search
Tables
cell 1
cell 2
cell 3
cell 4
Computer | Protected Mode: Off
100%

Figure 11-7: The cell-padding attribute increases the space within each cell (here it's set to 20).

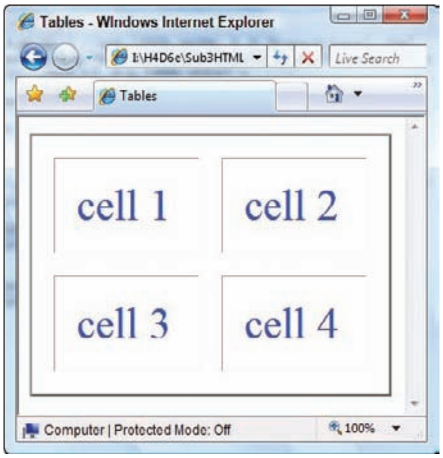

Tables - Windows Internet Explorer
I:\H4D6e\Sub3HTML
Live Search
Tables
cell 1
cell 2
cell 3
cell 4
Computer | Protected Mode: Off
100%

Figure 11-8: The cell-spacing attribute increases the width of the border (here it's also set to 20).

Working with `cellpadding` and `cellspacing` to get your table layout just right can be a bit of a headache. Sometimes you need to create empty cells to help control layout. Although this trick is a bit of a workaround, many designers use it. You just

1. **Create a cell.**
2. **Fill the cell with either one of these:**
   - `<br />`
   - A spacer image (a transparent `.gif` that is 1×1 pixel) with which you can manipulate the width

CSS

You can use CSS to control cell padding and spacing between cells.

Within cells

To control the padding within cells, you use the `padding` property, like so:

```
selector {padding: value;}
```

The value for the `padding` property must be defined by an absolute or relative length, or percentage.

To set the padding of a table cell, use the following style declaration:

```
td {padding: 10px;}
```

The `padding` property can be used with most (X)HTML elements. For example, if you created a footer and assigned it a class name, you can define padding for the element using the following style rule:

```
.footer { padding: 5px;}
```

Between cells

You can control the spacing between your cells using the `border-spacing` property:

```
selector {border-spacing: value;}
```

The value for the `border-spacing` property must be defined by an absolute or relative length, or percentage:

To set the padding of a table cell, use the following style declaration:

```
td {padding: 10px;}
```

REMEMBER

The `border-spacing` property can be used only in conjunction with the `<td>` element.

Shifting alignment

If you use tables to define your layout, you need to control their placement in the browser window. You can do this by using (X)HTML or CSS.

You use attributes that are part of the HTML standard to align your tables (horizontally) and your table contents (horizontally and vertically).

Aligning tables is similar to aligning images.

Horizontal alignment

You can horizontally align cell contents using the `align` attribute in various table elements.

- To align your table horizontally, use the `align` attribute with the `<table>` element.

  The `align` attribute, when used with the `<table>` element, has the following possible values: `left`, `right`, or `center` of the document.
- You can use the `align` attribute with the `<td>` (cell) or `<tr>` (row) elements to align text within the cell or row.

  The values that can be used with the `align` attribute in the `<td>` or `<tr>` elements are

  - `align="right"`: Aligns the table or cell contents against the right side.
  - `align="left"`: Aligns the table or cell contents against the left side. (This is the default setting.)
  - `align="center"`: Centers the table or cell contents. When applied to the `<table>`element, it centers the table, when applied to table cells it centers their contents.
  - `align="justify"`: Justifies cell contents in the middle (not widely supported).
  - `align="char"`: Aligns cell contents around a specific character (not widely supported).

The following example aligns a table in the center of the page, with centered text in each cell (see Figure 11-9):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Tables</title>
</head>
<body>
 <table border="2" width="430" align="center" cellpadding="20">
  <tr>
    <td width="630" colspan="2" align="center"> contents </td>
  </tr>
  <tr>
    <td width="230" align="center"> contents </td>
    <td width="200" align="center"> contents </td>
  </tr>
 </table>
</body>
</html>
```

Vertical alignment

You can vertically align cell contents by using the `valign` attribute. It can only be used with the `<tr>` (cell) and `<td>` (row) elements.

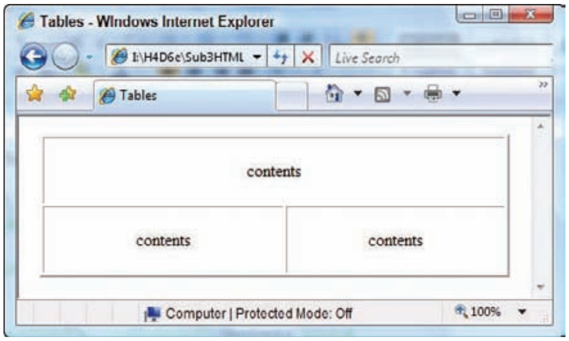

Tables - Windows Internet Explorer
I:\H4D6e\Sub3HTML
Live Search
Tables
contents
contents
contents
Computer | Protected Mode: Off
100%

Figure 11-9: A simple table centered.

The possible values are

- `valign="top"`: Vertically aligns cell contents to the top of the cell.
- `valign="bottom"`: Vertically aligns cell contents to the bottom of the cell.
- `valign="middle"`: Vertically centers the cell contents. (This is the default.)
- `valign="baseline"`: Defines a baseline for all other cells in the same row, so alignment is the same for all cells.

You can also use the align and valign attributes with the following table elements: <col>, <colgroup>, <tbody>, <tfoot>, <th>, and <thead>.

If you set alignment for a row (<tr>) and then set alignment for a cell within that row (<td>), the setting for the cell overrides the setting for the row.

You might as well get used to hearing that most X(HTML) formatting attributes are deprecated in favor of CSS. However, even though the align attribute is deprecated for most (X)HTML elements, it is still OK when used with table elements.

You can't use the valign attribute with the <table> tag.

Using CSS to define alignment

There are two CSS properties you can use to control table alignment using CSS: text-align and vertical-align. They function just like the preceding align and valign attributes.

To use the text-align property, assign it one of the following values:

- right: Aligns the table or cell contents against the right side.
- left: Aligns the table or cell contents against the left side. (This is the default.)
- center: Centers the table or cell contents.
- justify: Justifies cell contents in the middle.

To use the vertical-align property, assign it one of the following values:

- top: Vertically aligns cell contents to the top of the cell.
- bottom: Vertically aligns cell contents to the bottom of the cell.
- middle: Vertically centers the cell contents. (This is the default.)
- baseline: Defines a baseline for all other cells in the same row, so alignment is the same for all cells.

You can control the alignment of an entire row by assigning alignment properties to the <tr> element.

You can't center a table by using the text-align property — it's only for text alignment. Currently, you have a few options for centering the entire table. None of them is ideal, but they all work:

- Use the deprecated `<center>` tags around the table (not advised).
- Use the deprecated `align` attribute within the table: `<table align="center">`. (Not all browsers handle this the same way, so check this markup in all of them!)
- Enclose the table in a `<div>` element and use the `text-align` property to center its contents: `div.mytable {text-align: center;}`. (Recommended.)

The `<div>` element is discussed further in Chapters 8 and 9.

Adding Spans

Spanning is one of the reasons tables may be useful when arranging elements in your Web page.

Spanning enables you to stretch items across multiple cells; you essentially tear down a cell wall. Whether you need to span rows or columns, you can use the concept of spanning to wrangle your table into almost any arrangement.

WARNING!

Spanning columns and rows takes careful planning. That planning should occur during the sketching phase (as we describe earlier in this chapter, in the section "Sketching Your Table").

To span cells, you add one of these attributes to the `<td>` (that is, cell) element:

- `colspan` extends a cell *horizontally* (across multiple *columns*).
- `rowspan` extends a cell *vertically* (across multiple *rows*).

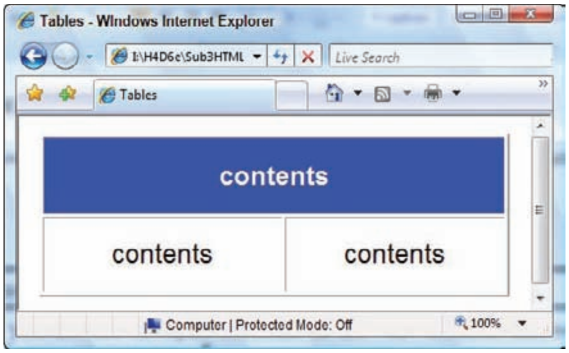
TECHNICAL STUFF

Spanning cells works using only (X)HTML attributes; CSS doesn't provide equivalent functionality. That said, the `<div>` element lets you do just about anything without even using a table that you can do with spanning inside a table (and explains why it's become the preferred approach for professional page designers).

Column spans

To span columns, use the `colspan` attribute in the `<td>` element and set its value equal to the number of cells you wish to span. Here we set the spanning column background to blue, and make text white and extra bold so it matches the appearance of plain black text on a white background in the two cells below.

Figure 11-10 illustrates a cell that spans two columns.


Tables - Windows Internet Explorer
I:\H4D6e\Sub3HTML
Live Search
Tables
Computer | Protected Mode: Off
100%

contents

contents	contents

Figure 11-10: The cell spans two columns.

In this example, a single blue cell in the first row spans the white cells in the two columns of the next row. You use the `colspan` attribute set to `2`, as shown in the following markup, because the cell in the first row spans two columns:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Tables</title>
</head>
<body>
 <table border="2" width="430" align="center" cellpadding="20"
    type="text/css" style="font-family: Arial, sans-serif; font-size: 18pt;">
  <tr>
    <td width="430" colspan="2" align="center"
     type="text/css"
     style="background-color: blue; color: white; font-weight: bolder;">
     contents </td>
  </tr>
  <tr>
    <td width="230" align="center"> contents </td>
    <td width="200" align="center"> contents </td>
  </tr>
 </table>
</body>
</html>
```

TIP

Note in the preceding example, and in the next one, we use incline CSS code to keep the style information together with the table markup. In production markup, you'd want to put your CSS markup into an external style-sheet file.

After you add a `colspan` attribute

- Verify that you have the appropriate number of `<td>` cells in the first row. For example, if you define a cell to span two columns, you should have one less `<td>` in that row. If you use `colspan="3"`, there should be two fewer `<td>` cells in that row.
- Make sure that the other rows have the appropriate number of `<td>` cells. For example, if you define a cell to span two columns, the other rows in that table should have two `<td>` cells to fill out the two columns.

Row spans

You use the `rowspan` attribute with the `<td>` tag. Figure 11-11 illustrates a cell that spans two rows.

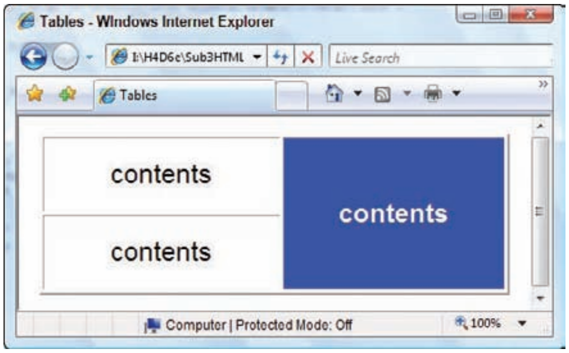

Tables - Windows Internet Explorer
I:\H4D6e\Sub3HTML
Live Search
Tables
contents
contents
contents
Computer | Protected Mode: Off
100%

Figure 11-11: The last cell containing navigational items spans two rows.

To span rows, you use the `rowspan` attribute in the `<td>` element and set the value equal to the number of cells you want to span.

REMEMBER

Sketch your table first so you know which cells should span which columns and rows. The example design we use throughout most of this chapter uses the `colspan` attribute with the first cell. However, the design could have been just as simple if we used a `rowspan` with the last cell that contains the navigational items. Either way, the table is efficiently laid out.

The modified table comes from the following markup (note the bold `rowspan`):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Tables</title>
</head>
<body>
 <table border="2" width="430" align="center" cellpadding="20"
    type="text/css" style="font-family: Arial, sans-serif; font-size: 18pt;">
  <tr>
    <td width="230" align="center"> contents </td>
    <td width="200" align="center" rowspan="2"
     type="text/css"
     style="background-color: blue; color: white; font-weight: bolder;">
     contents </td>
  </tr>
  <tr>
    <td width="230" align="center">
     contents </td>
  </tr>
 </table>
</body>
</html>
```

Populating Table Cells

After you sketch your table and define table properties (such as width, cell padding and spacing, and cell spanning), you're ready to populate the table cells with images, hyperlinks, text, and almost any other (X)HTML element. This is a simple process: You add images, hyperlinks, and text to the `<td>` element in much the same way you add them to the `<body>` element.

The following markup shows a populated table, with data added in bold:

The following markup shows a populated table, with data added in bold:

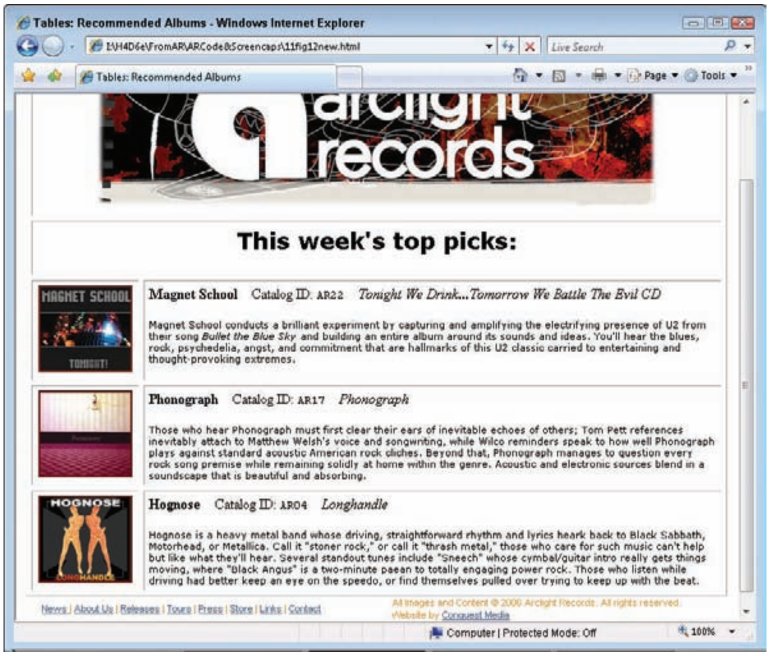
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Tables: Recommended Albums</title>
    <meta http-equiv="Content-type" content="text/html;charset=UTF-8" />
    <style type="text/css">
      h1 {
        font-size: 200%;
        font-family: Verdana, Tahoma, Arial, sans-serif;
        }
      p {
        font-size: 90%;
        font-family: Verdana, Tahoma, Arial, sans-serif;
        }
      #footer {
        font-size: 70%;
        font-family: Arial, sans-serif;
        color: orange;
        }
    </style>
```

```
</head>
<body>
 <table border="1" width="800" align="center" cellpadding="5" cellspacing="5">
  <tr>
    <td colspan="2" valign="bottom" align="center">
      <a href="http://www.arclightrecords.com">
        <img src="arclight-header.jpg" width="640" height="203"
          alt="arclight header art" border="0" />
      </a>
    </td>
  </tr>
  <tr>
    <td colspan="2" valign="center" align="center">
      <h1>This week's top picks:</h1>
      </a>
    </td>
</tr>
  <tr>
    <td valign="center" align="center" width="111">
      <img src="releases-ar22.gif" width="107" height="99"
       alt="magnet school cover" border="0" />
    </td>
    <td valign="top" align="left" width="689">
    <strong>Magnet School</strong>&nbsp;&nbsp;&nbsp;
    Catalog ID: <tt>AR22</tt>&nbsp;&nbsp;&nbsp;
    <em>Tonight We Drink...Tomorrow We Battle The Evil CD</em><br />
    <p>Magnet School conducts a brilliant experiment by capturing and amplifying
              the
    electrifying presence of U2 from their song <cite>Bullet the Blue Sky</cite>
    and building an entire album around its sounds and ideas. You'll hear the
              blues,
    rock, psychedelia, angst, and commitment that are hallmarks of this U2
              classic
    carried to entertaining and thought-provoking extremes.</p>
    </td>
  </tr>
  <tr>
    <td valign="center" align="center" width="111">
      <img src="releases-ar17.gif" width="107" height="99"
       alt="Phonograph Phonograph cover" border="0" />
    </td>
    <td valign="top" align="left" width="680">
    <strong>Phonograph</strong>&nbsp;&nbsp;&nbsp;
    Catalog ID: <tt>AR17</tt>&nbsp;&nbsp;&nbsp;
    <em>Phonograph</em><br />
    <p>Those who hear Phonograph must first clear their ears of inevitable
              echoes of
    others; Tom Petty references inevitably attach to Matthew Welsh's voice and
    songwriting, while Wilco reminders speak to how well Phonograph plays
              against
    standard acoustic American rock cliches. Beyond that, Phonograph manages to
    question every rock song premise while remaining solidly at home within the
    genre. Acoustic and electronic sources blend in a soundscape that is beautiful
```

```
      and absorbing.</p>
      </td>
    </tr>
    <tr>
      <td valign="center" align="center" width="111">
        <img src="releases-ar04.gif" width="107" height="99"
         alt="Hognose Longhandle cover" border="0" />
      </td>
      <td valign="top" align="left" width="689">
      <strong>Hognose</strong>&nbsp;&nbsp;&nbsp;
      Catalog ID: <tt>AR04</tt>&nbsp;&nbsp;&nbsp;
      <em>Longhandle</em><br />
      <p>Hognose is a heavy metal band whose driving, straightforward rhythm and
                 lyrics
      hark back to Black Sabbath, Motorhead, or Metallica. Call it "stoner rock," or
      call it "thrash metal," those who care for such music can't help but like what
      they'll hear. Several standout tunes include "Sneech" whose cymbal/guitar
                 intro
      really gets things moving, where "Black Angus" is a two-minute paean to
                 totally
      engaging power rock. Those who listen while driving had better keep an eye on
      the speedo, or find themselves pulled over trying to keep up with the
                 beat.</p>
      </td>
    </tr>
   </table>
  <table width="800" border="0" align="center" cellpadding="0" cellspacing="0"
                 id="footer">
    <tr>
      <td width="18">&nbsp;</td>
      <td width="382"><a href="index.htm">News </a>| <a href="aboutus.htm">About
                 Us</a> |
      <a href="releases.htm">Releases</a> | <a href="tours.htm">Tours</a> |
      <a href="press.htm">Press</a> | <a href="store.htm">Store</a> |
      <a href="links.htm">Links</a> | <a href="contactus.htm">Contact</a> </td>
      <td width="18">&nbsp;</td>
      <td width="382">All Images and Content &copy; 2006 Arclight Records. All
                 rights reserved.
      <br />
      Website by <a href="http://www.conquestmedia.com">Conquest Media</a></td>
    </tr>
  </table>
  </body>
  </html>
```

There are numerous interesting things to observe about this Web page. We sized it for 800 pixels in width (a pretty standard page size nowadays that will accommodate all but the oldest computer displays), and used the `colspan` attribute to enable the header image and the title to center themselves across the entire page. We also carefully divided the album cover and text display areas using a common thumbnail size for the artwork (107 × 99 pixels as it turned out); with 2 extra pixels on each side that left 689 pixels over for the text area, which we used to accommodate the artist name, catalog number and title on a single line of text, followed by up to 7 lines of text to describe the album and its music. It's depicted in Figure 11-12, where we leave borders turned on so you can see them (for production use, you'd turn some or all of them off).

Notice also the use of the non-breaking-space character entity (`&nbsp;`): In the album listings we use it to force a little white space between artist name, catalog number and album title on the first line in each listing; in the footer area at the bottom of the page, we use it to force an 18-pixel margin between the footer menu on the left and the copyright notice on the right. We also make use of the `h1` style to choose text for the page headline, paragraph (`p`) style to select text for the album copy, and use the `footer` ID to change things up a little for the page footer. All of this should give you, our readers, plenty of ideas for using tables and styles to help manage layouts where that makes sense.


Tables: Recommended Albums - Windows Internet Explorer
I:\H4D6e\FromAR\ARCode&Screencaps\11fig12new.html
Live Search
Tables: Recommended Albums
Page
Tools
arclight records
MAGNET SCHOOL
TONIGHT!
HOGNOSE
LONGHANDLE
Computer | Protected Mode: Off
100%

This week's top picks:

Magnet School Catalog ID: AR22 *Tonight We Drink...Tomorrow We Battle The Evil CD*

Magnet School conducts a brilliant experiment by capturing and amplifying the electrifying presence of U2 from their song *Bullet the Blue Sky* and building an entire album around its sounds and ideas. You'll hear the blues, rock, psychedelia, angst, and commitment that are hallmarks of this U2 classic carried to entertaining and thought-provoking extremes.

Phonograph Catalog ID: AR17 *Phonograph*

Those who hear Phonograph must first clear their ears of inevitable echoes of others; Tom Pett references inevitably attach to Matthew Welsh's voice and songwriting, while Wilco reminders speak to how well Phonograph plays against standard acoustic American rock cliches. Beyond that, Phonograph manages to question every rock song premise while remaining solidly at home within the genre. Acoustic and electronic sources blend in a soundscape that is beautiful and absorbing.

Hognose Catalog ID: AR04 *Longhandle*

Hognose is a heavy metal band whose driving, straightforward rhythm and lyrics heark back to Black Sabbath, Motorhead, or Metallica. Call it "stoner rock," or call it "thrash metal," those who care for such music can't help but like what they'll hear. Several standout tunes include "Sneech" whose cymbal/guitar intro really gets things moving, where "Black Angus" is a two-minute paean to totally engaging power rock. Those who listen while driving had better keep an eye on the speedo, or find themselves pulled over trying to keep up with the beat.

News | About Us | Releases | Tours | Press | Store | Links | Contact

All images and Content © 2006 Arclight Records. All rights reserved.
Website by Conquest Media

Figure 11-12: The Arclight Records Recommended Titles page uses tables to organize album listings and thumbnails.

Testing Your Table

Testing is the final step before your table goes live. You must test your tables in all the popular browsers — including Internet Explorer, Firefox, Safari, and Opera. If you don't, your users might have to squint at your pages, or they might see your tables as one big mess.

TIP

As you're creating your table, keep a browser window open at the same time. Each time you change the width of a cell or add an item to a cell, save the document and view it in the browser window. That way, when you test your table, you probably won't have too much tweaking to do.

REMEMBER

Always test your site using any browser that your users might employ. For example, if your table is aligned with `align="center"` but in an old version of Internet Explorer the table remains flush with the left side, you might want to surround the table with `div` tags and include the `align="center"` attribute inside the opening `<div>`. However, you won't have too many problems with tables if you stick to the standard.

Table-Making Tips

We've spent years of building, maintaining, and troubleshooting tables, and in that time we've discovered some neat tricks. The following tips are a head start toward creating effective tables.

Following the standards

The first — and (we think) most important — tip is to keep with the established standards. The folks involved with the Web Standards Project have campaigned for full standard support in browsers and HTML authoring applications since 1998. Their hard work should make your life easier.

A long time ago, if you built an HTML table, you'd be forced to create different versions of your Web page (each version containing browser-specific elements and attributes) just to define some basic table properties. As you might imagine, creating and maintaining different versions of the same Web page drove development costs sky-high. To get around those costs, many developers carefully crafted their tables with specific markup that worked in Internet Explorer and Netscape — but what about Opera? Well, happily those are problems of the past. The newest versions of Internet Explorer, Firefox, Safari, and Opera *all* support HTML, as well as CSS and XHTML. To find out more about Web standards, visit `www.webstandards.org`.

Sanitizing markup

Efficiently written markup is easier to troubleshoot and maintain. To that end, many designers use white space to separate elements. For example, the following markup doesn't use much white space and is hard to read:

```
<table border="1" width="630">
<tr><td width="630" colspan="2"> contents </td></tr>
<tr><td width="400"> contents </td>
<td width="230"> contents </td></tr></table>
```

Check out this cleaner version:

```
<table border="1" width="630">
  <tr>
    <td width="630" colspan="2"> contents </td>
  </tr>
  <tr>
    <td width="400"> contents </td>
    <td width="230"> contents </td>
  </tr>
</table>
```

WARNING! The white space we include in our markup is *between* elements — not *within* elements. If, for example, you add white space between the `<td>` and `</td>` tags, it affects how a cell's content is displayed, which isn't generally something you want to do.

Nesting tables within tables

Many designers are forced to nest tables within tables to achieve a desired effect. This is both *legal* and *common.*

WARNING! A few nested tables won't affect your users too badly. But nesting many tables within tables can lengthen download time.

To nest a table, simply add the `<table>` element within a `<td>` element as follows:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Nesting Tables</title>
</head>
<body>
 <table border="1" cellpadding="20" align="center">
```

```
      <tr align="center">
        <td> contents </td>
        <td> contents </td>
      </tr>
      <tr>
        <td>
           <table border="1">
           <tr>
           <td> contents  </td>
           <td> contents  </td>
           </tr>
           <tr>
           <td> contents  </td>
           <td> contents  </td>
           </tr>
           </table>
        </td>
        <td> contents </td>
      </tr>
     </table>
    </body>
    </html>
```

This markup produces the tables shown in Figure 11-13.

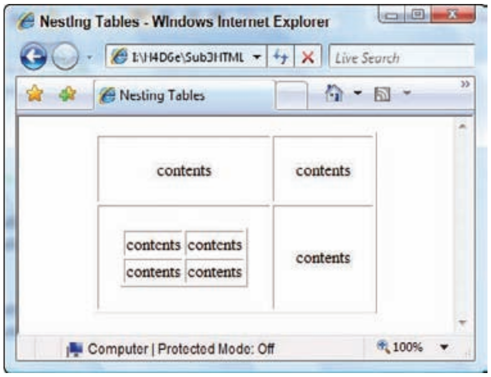


Figure 11-13: Nested tables.

When using nested tables . . .

- Check cell widths — the width of the third cell should match the width of the nested table.
- Create *and test* the table you intend to nest — separately, before you add it to your primary table.

Avoiding dense tables

We recommend creativity, but be careful — don't pack a screen full of dense, impenetrable information — especially numbers. A long, unbroken list of numbers quickly drives away all but the truly masochistic — pretty much negating the purpose of the table to begin with. Put those numbers into an attractive table (better yet, *several* tables interspersed with a few well-chosen images). Watch your page's attractiveness and readability soar; hear visitors sigh with relief.

Individual table cells can be surprisingly roomy. You can position graphics in them precisely. If you're moved to put graphics in a table, be sure to

- Select images that are similar in size and looks.
- Measure those images to determine their heights and widths in pixels. (Shareware programs such as Paint Shop Pro and GraphicConverter do this automatically.)
- Use HTML markup to position these images within their table cells.

A short-and-sweet table keeps the graphics in check and guarantees that the text always sits nicely to its right.

Two more handy graphics-placement tips produce a consistent, coherent image layout:

- Size your rows and columns of cells that contain images to accommodate the *largest* graphic.
- *Center* all graphics in each cell (both vertically and horizontally).

Adding color to table cells

You can use either CSS or (X)HTML to change the background color of a cell or table. Before CSS was around, designers used the `bgcolor` attribute to change the background of table cells in much the same way it affects the background of an entire HTML document. Simply add the `bgcolor` attribute to any table cell to change its background color:

```
<td bgcolor="teal">...</td>
```

However, now you have a bit more flexibility to use CSS to add some color:

```
td {background-color: red;}
```

We cover the `background-color` property in Chapter 10.

TIP

The `bgcolor` attribute may be used with any of the table elements. However, the `bgcolor` of a cell overrides any `bgcolor` defined for a row or table. Note that `bgcolor` is also deprecated, and that most Web experts use CSS markup instead, using the aforementioned `background-color` property as well as the `color` property to set text or foreground color.

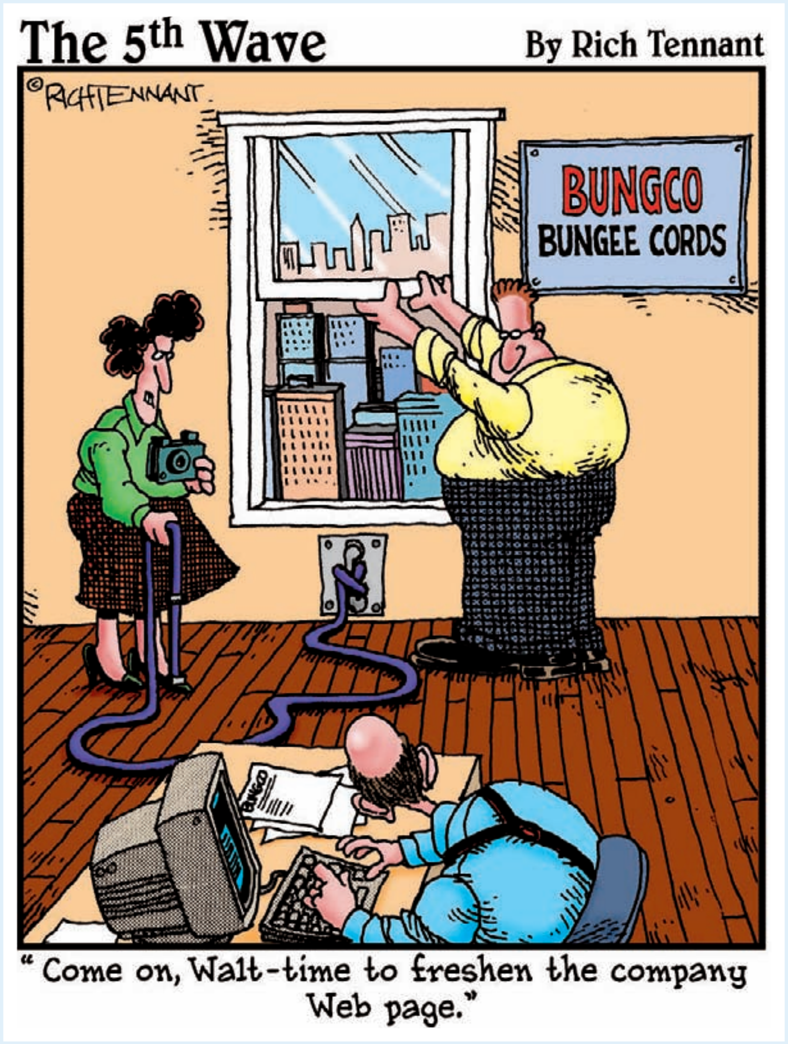
Other Table Markup of Interest

Table 11-1 lists other table-related (X)HTML attributes that you might find in HTML files.

Table 11-1 Additional Table-related (X)HTML Attributes

Name	*Function/Value Equals*	*Value Types*	*Related Element(s)*
`abbr`	Abbreviates table header cell name	Text	`<td> <th>`
`axis`	Sets a comma-separated list of related table headers	CDATA	`<td> <th>`
`char`	Defines alignment character for table elements	ISO 10646 char	`<col /> <colgroup> <tbody> <td> <tfoot> <th> <thead> <tr>`
`charoff`	Defines offset when alignment `char` is used	Length (p/%)	`<col /> <colgroup> <tbody> <td> <tfoot> <th> <thead> <tr>`
`frame`	Identifies visible components in a table structure	`{"above"|"below"|"border"|"box"| "hsides"|"lhs"|"rhs"|"void"|"vsides"}`	`<table>`
`rules`	Governs the display of rule bars in a table	`{"all"|"cols"|"groups"|"none"|"rows"}`	`<table>`
`scope`	Describes scope for table-header cells	`{"col"|"colgroup"|"row"|"rowgroup"}`	`<td><th>`
`summary`	Describes a table's purposes for rendering as speech	Text	`<table>`
`span`	Sets the number of table columns to which `col` attributes apply	Number	`<col />`

Part IV
Integrating Scripts with (X)HTML



"Come on, Walt-time to freshen the company Web page."

In this part . . .

In this part of the book, we introduce and describe the types of scripting languages that work on Web pages, and we dig into JavaScript — by far the most popular of all Web-scripting languages in use. Scripting languages help turn static, unchanging Web pages into active, dynamic documents that can solicit and respond to user input. You start by learning basic JavaScript elements, data types, and values, and progress to topics that include rearranging Web-page content on the fly, performing calculations and displaying their results, requesting and checking user input, and a whole bunch more.

Next, you dig more deeply into JavaScript so you can understand it — and use it — in your Web pages. You also learn how to incorporate JavaScript into Web pages and how it handles and changes Web page contents on the fly. You also learn about checking your work and using cookies (those interesting but elusive data packages that adhere to Web users as they flit about online).

The last two chapters in this part show you ways to put JavaScript to work in your Web pages. You explore how to define and use a navigation bar, which presents users with dynamic menus of options and information to make it easier for them to move around your Web site. You find out how to use JavaScript to create and use various data-entry forms in your Web pages to solicit, check, and respond to user input. You also pick up the basic concepts and techniques for creating dynamic HTML (sometimes called DHTML) and using client-side JavaScripts and prefabricated code to perform basic tasks, such as displaying date and time information, counting site visitors, or tabulating current statistics.

Chapter 12

Scripting Web Pages

In This Chapter

- Understanding what JavaScript is
- Exploring what JavaScript can do for your Web pages
- Dissecting three sample scripts

When used in conjunction with your HTML markup, scripts — small programs that you add to your Web page — help your Web pages respond to user actions. Scripts create the interactive and dynamic effects you see on the Web, such as images that automatically change when visitors move mouse pointers over them, additional browser windows that pop up when a page loads, and animated navigation bars.

Because scripts are mini-programs, they're often written in a programming language called JavaScript. If you are unfamiliar with the term, JavaScript may sound like a Hollywood screenplay doused with coffee. However, it is actually a scripting language built right into all popular Web browsers.

Fortunately, because of the Nobel-Prize-worthy invention of "copy and paste," you don't need to be a technoguru to add scripting to your Web sites. The Web has many sites that feature canned JavaScript scripts that you can freely copy and then paste right into your Web page. (Chapter 13 lists several of the best JavaScript sites.)

In this chapter, you explore how scripting works inside your Web page by dissecting three sample scripts written in JavaScript. Chapter 13 continues this discussion by diving deeper into the JavaScript language itself.

TIP

Many good Web-page editors (such as Adobe Dreamweaver and GoLive) have built-in tools to help you create scripts — even if you don't know anything about programming.

JavaScript is not Java

In the late 1990s, the originators of the JavaScript scripting language wanted to ride the coattails of the massive popularity of the Java programming language, so they gave it a catchy name — JavaScript. However, when they made this decision, they also introduced a lot of confusion given the similarity of the two names. To clarify, the full-featured Java programming language *isn't* a scripting language on the Web. Java is a descendent of the C and C++ programming languages. Programmers can create Java applications that can run on Windows, Macintosh, Linux, and other computer platforms:

- On the client side, Java is used to create *applets* (small programs that download over the Net and run inside Web browsers). Because Java is designed to be cross-platform, these applets should run identically on any Java-enabled browser.
- On the server side, Java is used to create many Web-based applications.

What JavaScript Can Do for Your Pages

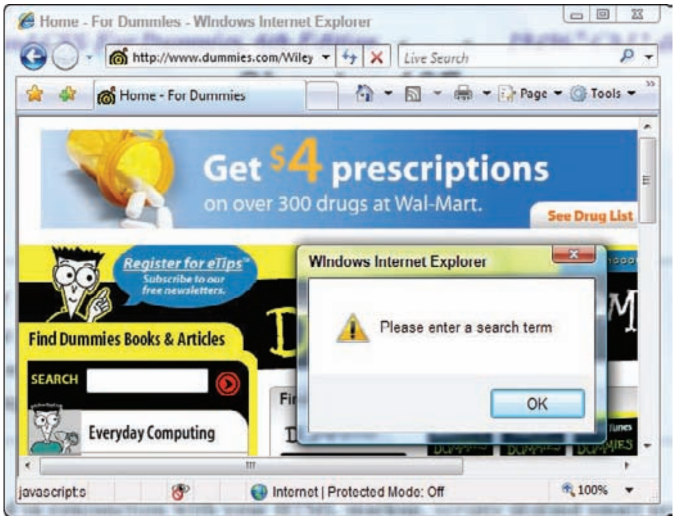
Adding scripts to your Web site is much like those reality-TV makeover shows that transform a house or a person's appearance into something completely new and wonderful. So too with JavaScript. You can transform a plain and dull Web page into an interactive and dynamic Web extravaganza to bring joy to your visitors for years to come. (Okay, maybe we're exaggerating just a tad, but you get the point.)

For example, if you visit Dummies.com (`www.dummies.com`) and click the red button next to the search box without entering a term to search on, the browser displays a nice warning box that reminds you to enter a search term before you actually search, as shown in Figure 12-1.

A short script verifies whether you've entered a search term before the engine runs the query:

- If you enter a search term, you don't see the warning.
- If you don't enter a search term, the script built into the page prompts the warning dialog box to appear.

This bit of scripting makes the page *dynamic,* which means it adds programmed functionality to your Web pages — and allows them to respond to what users do on the page (for example, filling out a form or moving the mouse pointer over an image). When you add scripts to your page, the page interacts with users and changes its display or its behavior in response to what users do.


Home - For Dummies - Windows Internet Explorer
http://www.dummies.com/Wiley
Live Search
Home - For Dummies
Page
Tools
Get $4 prescriptions
on over 300 drugs at Wal-Mart.
See Drug List
Register for eTips
Subscribe to our free newsletters.
Find Dummies Books & Articles
SEARCH
Everyday Computing
Windows Internet Explorer
OK
javascript:
Internet | Protected Mode: Off
100%

Please enter a search term

Figure 12-1: A script pops up a dialog box telling you what you did wrong.

The page URL doesn't change and another browser window doesn't open when you try to search on nothing. The page responds to what you do without sending a request back to the Web server for a new page. This is why the page is considered *dynamic*.

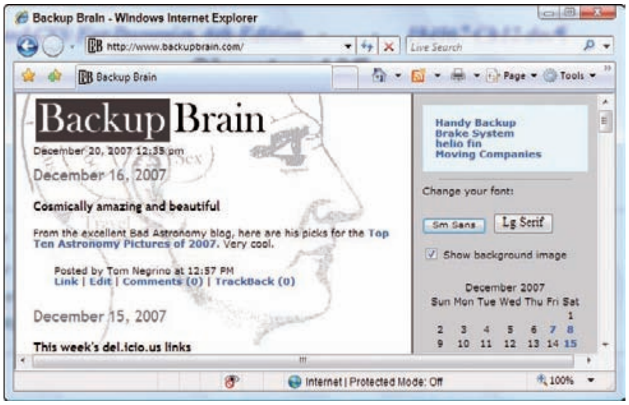
If you tried this trick without using a script (that is, without dynamic functionality), the browser would send the empty search string back to the Web server. Then the server would return a warning page reminding the user to enter a search term. All the work would be done on the Web server instead of in the Web browser. This would be slower (because the request must first go to the server, and then the server must transmit the warning page back to your browser) — which would feel much less fluid to the user. It's much better to just click a button on the page and have an alert pop up instantly to help the user.

In the following sections, we showcase three common ways in which JavaScript can be used in your Web pages.

Don't worry about the details of the JavaScript code in the following examples. Just focus on how JavaScript scripts can be pasted into your Web page and work alongside your HTML markup.

Arrange content dynamically

JavaScript can be used with CSS (covered in Chapters 8 and 9) to change the look of a page's content in response to a user action. Here's an example: Two authors share a Weblog, Backup Brain (`www.backupbrain.com`). One of the authors prefers small, sans-serif type, and the other one finds it easier to read larger, serif type. So the Weblog has buttons that change the look of the site to match each person's preference. Of course, the site's visitors can use the buttons to switch the look of the type, too, and the site remembers the visitor's choice for future visits by setting a *cookie* (a small preference file written to the user's computer). Figure 12-2 shows the two looks for the page.


Backup Brain - Windows Internet Explorer
http://www.backupbrain.com/
Live Search
Backup Brain
Page
Tools
Internet | Protected Mode: Off
100%

Backup Brain

December 20, 2007 12:35 pm

December 16, 2007

Cosmically amazing and beautiful

From the excellent Bad Astronomy blog, here are his picks for the Top Ten Astronomy Pictures of 2007. Very cool.

Posted by Tom Negrino at 12:57 PM
Link | Edit | Comments (0) | TrackBack (0)

December 15, 2007

This week's del.icio.us links

Handy Backup
Brake System
helio fin
Moving Companies

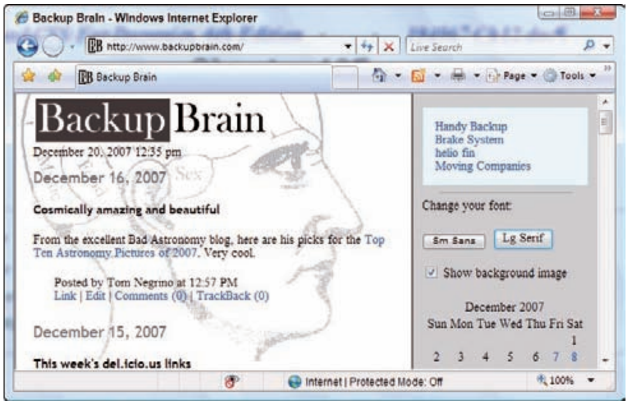
Change your font:

Sm Sans | Lg Serif

✓ Show background image

December 2007

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15


Backup Brain - Windows Internet Explorer
http://www.backupbrain.com/
Live Search
Backup Brain
Page
Tools
Internet | Protected Mode: Off
100%

Backup Brain

December 20, 2007 12:35 pm

December 16, 2007

Cosmically amazing and beautiful

From the excellent Bad Astronomy blog, here are his picks for the Top Ten Astronomy Pictures of 2007. Very cool.

Posted by Tom Negrino at 12:57 PM
Link | Edit | Comments (0) | TrackBack (0)

December 15, 2007

This week's del.icio.us links

Handy Backup
Brake System
helio fin
Moving Companies

Change your font:

Sm Sans | Lg Serif

✓ Show background image

December 2007

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8

Figure 12-2: Clicking the "Change your font" buttons changes how the text displays.

JavaScript and CSS create this effect by switching between two style sheets:

- The sans-serif style sheet, `sansStyle.css`
- The serif style sheet, `serifStyle.css`

Listing 12-1 shows the source code for an example page that contains this switching mechanism.

- When a user clicks the Sm Sans button on the page, a script runs (`styleSwitcher.js`, referenced in the `<head>` element) and switches the active style sheet to `sansStyle.css`. (Chapter 13 covers `.js` files.)
- When the user clicks the Lg Serif button, the same script switches to the `serifStyle.css` style sheet.

Listing 12-1: Style Switching

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Style Changer</title>
    <link href="simpleStyle.css" rel="stylesheet" rev="stylesheet" />
    <link href="sansStyle.css" rel="stylesheet" rev="stylesheet"
        title="default" />
    <link href="serifStyle.css" rel="alternate stylesheet"
        rev="alternate stylesheet" title="serif" />
    <style type="text/css" media="all">@import url("complexStyle.css");</style>
    <script src="styleSwitcher.js" language="javascript1.5"
        type="text/javascript"></script>
</head>
<body>
<div class="navBar">
<br />Change your font:
<form action="none">
    <input type="button" class="typeBtn" value="Sm Sans"
       onclick="setActiveStylesheet('default')" />
    <input type="button" class="typeBtn2" value="Lg Serif"
       onclick="setActiveStylesheet('serif')" />
</form>
</div>

<div class="content" id="headContent">
<p>Replace this paragraph with your own content.</p>
</div>
</body>
</html>
```

You can see the example page for yourself at

```
www.javascriptworld.com/chap11-3.html
```

This example relies on several different files (HTML, CSS, and JavaScript). You can download all 2.6 MB of these files if you'd like, from

```
www.javascriptworld.com/JavaScript6eScripts.zip
```

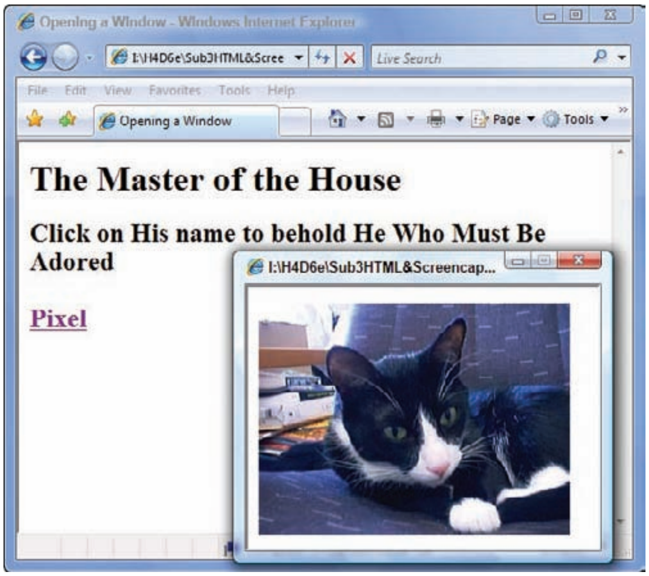
Work with browser windows

JavaScript can tell your browser to open and close windows.

WARNING!

You've probably seen an annoying version of this trick: advertising pop-up windows that appear when you try to leave a site. (Let's not go there.) But this technology can be used for good as well as evil. For example, you can preview a set of big image files with small thumbnail versions. Clicking a thumbnail image can perform such actions as

- Opening a window with a larger version of the image.
- Opening a page with a *text link* that opens a window with an illustration of that text, as shown in Figure 12-3.


Opening a Window - Windows Internet Explorer
I:\H4D6e\Sub3HTML&Scree
Live Search
File Edit View Favorites Tools Help
Opening a Window
Page
Tools
I:\H4D6e\Sub3HTML&Screencap...

The Master of the House

Click on His name to behold He Who Must Be Adored

Pixel

Figure 12-3: When you click the link, a pop-up window appears with a picture in it.

The code required to do this sort of pop-up window is fairly straightforward, as Listing 12-2 shows.

Listing 12-2: Pop-up Windows

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Opening a Window</title>
    <script language="Javascript" type="text/javascript">
    function newWindow() {
        catWindow = window.open("images/pixel2.jpg", "catWin",
          "width=330,height=250")
    }
    </script>
</head>
<body bgcolor="#FFFFFF">
    <h1>The Master of the House</h1>
    <h2>Click on His name to behold He Who Must Be Adored<br /><br />
    <a href="javascript:newWindow()">Pixel</a></h2>
</body>
</html>
```

Pop-up windows can backfire on you if you use them too much. Many Web sites use pop-up windows to deliver ads, so users are becoming desensitized (or hostile) to them, and simply ignore them (or install software that prevents them). Before you add a pop-up window to your site, be sure it's absolutely necessary.

Chapter 13 has more details on creating pop-up windows with JavaScript.

Solicit and verify user input

A common use for JavaScript is to verify that users have filled out all the required fields in a form before the browser actually submits the form to the form-processing program on the Web server. Listing 12-3 places a form-checking function, `checkSubmit`, in the `<script>` element of the HTML page and references it in the `onsubmit` attribute of the `<form>` element.

Listing 12-3: Form Validation

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Linking scripts to HTML pages</title>
  <script type="text/javascript" language="javascript">
    function checkSubmit ( thisForm ) {
      if ( thisForm.FirstName.value == '' ) {
            alert('Please enter your First Name.');
            return false;
      }

      if ( thisForm.LastName.value == '' ) {
            alert('Please enter your Last Name.');
            return false;
      }

      return true;
```

(continued)

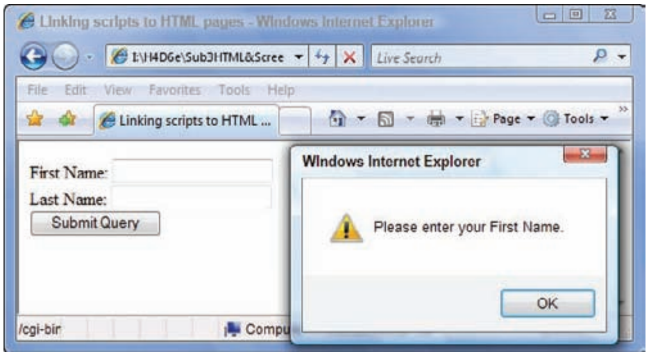
Listing 12-3 *(continued)*

```
      }
  </script>
  </head>

  <body>
    <form method="POST" action="/cgi-bin/form_processor.cgi"
          onsubmit="return checkSubmit(this);">
    <p>
      First Name: <input type="text" name="FirstName" /><br />
      Last Name: <input type="text" name="LastName" /><br />
      <input type="submit" />
    </p>
    </form>
  </body>
  </html>
```

This script performs one of two operations if either form field isn't filled in when the user clicks the Submit button:

- It instructs the browser to display a warning to let the user know he or she forgot to fill in a field.
- It returns a value of `false` to the browser, which prevents the browser from actually submitting the form to the form-processing application.

If the fields are filled in correctly, the browser displays no alerts and returns a value of `true`, which tells the browser the form is ready for the Web server. Figure 12-4 shows how the browser displays an alert if the first name field is empty.


Linking scripts to HTML pages - Windows Internet Explorer
I:\H4D6e\Sub3HTML&Scree
Live Search
File Edit View Favorites Tools Help
Linking scripts to HTML ...
Page Tools
First Name:
Last Name:
Submit Query
/cgi-bin
Compu
Windows Internet Explorer
OK

Please enter your First Name.

Figure 12-4: A good use of JavaScript is to validate form data.

TIP

Although this example only verifies whether users filled out the form fields, you can create more advanced scripts that check for specific data formats (such as @ signs in e-mail addresses and only numbers in phone number fields).

TRICKS OF THE TRADE

When you create forms that include required fields, we recommend that you always include JavaScript field validation to catch missing data before the script finds its way back to the server. Visitors get frustrated when they take the time to fill out a form only to be told to click the Back button to provide missing information. When you use JavaScript, the script catches any missing information before the form page disappears so users can quickly make changes and try to submit again.

But wait . . . there's more!

You can do much more with JavaScript. The following list highlights several common uses of the scripting language:

- Detect whether a user has a browser plug-in installed that handles multimedia content
- Build slide shows of images
- Automatically redirect the user to a different Web page
- Add conditional logic to your page, so that if the user performs a certain action, other actions are triggered
- Create, position, and scroll new browser windows
- Create navigation bars and change the menus on those bars dynamically
- Automatically put the current date and time on your page
- Combine JavaScript and CSS to animate page elements

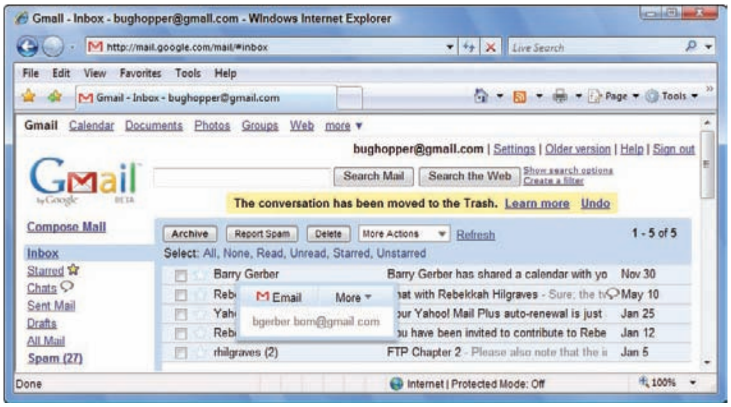
TECHNICAL STUFF

Server-side scripting

JavaScript is a scripting language that runs inside the browser, but there are other scripting languages that run on the server side — such as Perl, ASP (Active Server Pages), PHP, Python, .NET, ColdFusion, and others. Programs written in these languages reside on the server and are called by the Web page, usually in response to a form filled out by the user. People who write these Web pages may include small snippets of code that pass bits of information from the HTML page to the program on the server. When called, the program runs and then returns a result of some sort to the user.

Amazon.com is a familiar e-commerce Web application that runs mostly on the server side, using server scripts. Therefore Web pages displayed by the browser when you visit Amazon are the result of processing server-side scripts — all of which takes place before the page ever gets to your browser.

An innovative use of JavaScript occurs in Gmail, the free Web-based e-mail service from Google, which you can find at `www.gmail.com`. Gmail uses JavaScript to load an entire e-mail user interface into the user's browser, which makes Gmail much more responsive to user actions than most other Web-based mail programs. Gmail uses JavaScript to keep to an absolute minimum the number of times the page has to fetch additional information from the servers. By doing much of the processing in the user's browser, the Gmail Web application feels more like an e-mail program that runs on your computer. Figure 12-5 shows the JavaScript-powered Gmail interface. It's a great example of the power of JavaScript.


Gmail - Inbox - bughopper@gmail.com - Windows Internet Explorer
http://mail.google.com/mail/#inbox
Live Search
File Edit View Favorites Tools Help
Gmail - Inbox - bughopper@gmail.com
Page Tools
Gmail Calendar Documents Photos Groups Web more
bughopper@gmail.com | Settings | Older version | Help | Sign out
Search Mail
Search the Web
Show search options
Create a filter
Compose Mail
Inbox
Starred
Chats
Sent Mail
Drafts
All Mail
Spam (27)
Archive
Report Spam
Delete
More Actions
Refresh
1 - 5 of 5
Select: All, None, Read, Unread, Starred, Unstarred
Email
More
Done
Internet | Protected Mode: Off
100%

The conversation has been moved to the Trash. Learn more Undo

Barry Gerber	Barry Gerber has shared a calendar with yo	Nov 30
Reb	hat with Rebekkah Hilgraves - Sure; the tv	May 10
Yah	our Yahoo! Mail Plus auto-renewal is just	Jan 25
Reb	ou have been invited to contribute to Rebe	Jan 12
rhilgraves (2)	FTP Chapter 2 - Please also note that the i	Jan 5

bgerber bom@gmail com

Figure 12-5: The Gmail interface is powered by JavaScript.

Chapter 13

The Nuts and Bolts of JavaScript

In This Chapter

- Putting scripts in your pages
- Using external script pages
- Delving into the JavaScript language
- Checking out more JavaScript Resources

A lot of good "canned JavaScript" is available for free on the Web; you know what we mean — scripts written by someone else that you simply copy and paste into your HTML page. But as good as canned scripts can be, copy-and-paste goes only so far. Sooner or later, you're going to encounter unique needs that can't be fulfilled with a free script.

Canned JavaScript is much like canned Spam (the meat product, not the e-mail affliction): Great for convenience, but you probably don't want to make it an exclusive diet. Instead, knowing how to script — or at least how to tweak a prewritten script — is as important as knowing how to fix some good ol' fashioned home cooking.

In this chapter, you "open the can" of the JavaScript language and have a look at what's inside. (Don't worry; you won't encounter any meaty pink substances along the way.) You discover how to plug scripts into your pages, how to bundle your scripts into external JavaScript files to save time and effort, and how the nuts and bolts of the JavaScript language work. Finally, at the end of this chapter, we point you to good sources of additional information about JavaScript. These will come in handy as your scripting needs advance.

Including Scripts in Web Pages

Because a JavaScript script is a totally separate animal from HTML markup, you have to contain this JavaScript beast inside an HTML container tag, `<script>` and `</script>`. You can put a script in one of two places on an HTML page:

- Within the `<head>` and `</head>` tags (this is called a *header script*)
- Within the `<body>` and `</body>` tags (this is called a *body script*)

Header scripts contain code that you either want processed before the page loads or else you want them available to be called by other scripts in your Web page. Body scripts are executed when the `<body>` tag is processed. Typically, body scripts are used to generate HTML content for the page.

Listing 13-1 shows a header script. This simple script pops up a welcoming message box when the user loads the page.

Listing 13-1: Header Script

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>My JavaScript page</title>
     <script language="Javascript" type="text/javascript">
       alert("Welcome to my JavaScript page!")
     </script>
</head>
<body bgcolor="#FFFFFF">
   <h2>This script pops up a message box for the user.</h2>
</body>
</html>
```

The preceding `<script>` tag has two attributes:

- `language="Javascript"` tells the browser which scripting language the document uses.
- `type="text/javascript"` tells the browser that the script is plain text in JavaScript.

The script itself, `alert("Welcome to my JavaScript page!")`, is straightforward. The `alert()` method displays a message box that pops up on top of the browser window and shows a customized message to the Web page visitor. You specify the message you want displayed by enclosing the text within quotation marks and putting the text string inside the `alert()` method's parentheses, as shown in Listing 13-1. (***Note:*** (Curly quotes and single quotes won't work.) Make sure you close the script with the `</script>` tag, and your script is ready to go.

Using the Same Script on Multiple Pages

If you have a single Web page that uses a JavaScript script, it's handy to be able to contain all the scripting code inside a single `<script>` tag. However, suppose you have a boatload of pages, each of which needs to call the same script. You can always copy and paste the script into each page, but there are two downsides to that approach:

- You have to add the script to each page and make sure it's set up correctly and working.
- Any time you tweak the script, you're forced to update *each and every HTML page that uses it.* If you have two pages, that's no big deal. But if you have more than three, it can lead to a maintenance migraine.

Fortunately, this latter headache can be avoided — even without ibuprofen! Instead, you can use an external JavaScript file, also called a `.js` file (pronounced "dot jay ess"). A `.js` file is an ordinary text file that stores your JavaScript scripts. You can store one or more of your JavaScript scripts in a single `.js` file and access them from multiple HTML pages. (It is much like the `.css` files in which you store external style sheets, except that a `.js` file stores external JavaScript code.)

To use the same script on multiple pages, you should

1. **Put the script in an external JavaScript file.**

   If you have the script already inside your HTML page, remove all the JavaScript code inside the `<script>` tag and paste it into a separate file.

2. **Reference the file in any HTML page when you need the script.**

   Define a `<script>` tag in the head section of your Web page, but don't add any code inside it. Instead, use the `src` (for *source*) attribute in the `<script>` tag to call the external `.js` file.

Listing 13-2 shows the reference to the external file.

Listing 13-2: External Script Reference

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>My JavaScript page</title>
<script src="external.js" language="JavaScript" type="text/javascript">
</script>
</head>
<!-- (the rest of your HTML page goes here) -->
```

You don't need to include anything else between the opening and closing script tags beyond what's shown. In Listing 13-2, the name of the source file, `external.js`, is placed between double quotes. You can reference this file, `external.js`, with either a relative or absolute link, so you can refer to external JavaScript files in other directories on your server, or even on other servers (if you have access to those servers).

Adding the `src` attribute to the `<script>` tag tells the browser to look for that external file in the specified path. The resulting Web pages look and act like the scripts are in the header or body of the page's script tags, though the script is in the external `.js` file.

With this technique, you need to change a JavaScript only once on your site in the external file, not in each individual page on the site. All pages that reference the external file automatically receive the updated code. It's a big time-saver when updating your site.

If you use a script on only one page, it's often easier simply to put the script on the page in a body or header script.

If you have multiple external `.js` files, you can use any or all of them on any HTML page. Just include multiple `<script>` references on the page. It's perfectly okay for a page to include multiple scripts — and to both refer to external `.js` files and to include its own scripts inside `<script>` tags.

When you have multiple `<script>` tags defined in your Web page, the browser processes them in the order in which they are declared. If, for some reason, you have an external `.js` file that conflicts with a script inside a `<script>` tag, the last one defined wins.

There is nothing magical about the inside of the `.js` file itself — it is pure JavaScript code. No HTML tags are allowed. Listing 13-3 shows an example of a script in an external `.js` file. This script implements button rollovers for a Web page. When the user moves the mouse pointer over a button image, the image changes to highlight the choice.

Listing 13-3: An External JavaScript File

```
homeOff = new Image
productsOff = new Image
contactOff = new Image
pressOff = new Image

homeOver = new Image
productsOver = new Image
contactOver = new Image
pressOver = new Image

homeOff.src = "images/home_off.jpg"
productsOff.src = "images/products_off.jpg"
contactOff.src = "images/contact_off.jpg"
pressOff.src = "images/press_off.jpg"

homeOver.src = "images/home_over.jpg"
productsOver.src = "images/products_over.jpg"
contactOver.src = "images/contact_over.jpg"
pressOver.src = "images/press_over.jpg"

function imgOver(thisImg) {
    document[thisImg].src = "images/" + thisImg + "_over.jpg"
}

function imgOut(thisImg) {
    document[thisImg].src = "images/" + thisImg + "_off.jpg"
}
```

Note that the contents of the `.js` file shown in Listing 13-3 could be pasted directly between `<script>` and `</script>` tags inside an HTML or XHTML document and function identically.

Exploring the JavaScript Language

If you travel to a country whose people don't speak your language, you usually buy a pocket guide to help you make your way through the country. You don't necessarily need to know all the particulars and idiosyncrasies of the language, but you do need to know the essentials — phrases like "Where's the bathroom?" or "Where can I get an espresso?" — in order to survive.

In the same way, if you want to work with JavaScript, you don't need to become a hotshot scripting guru. You do, however, need to know enough about scripting to do the programming equivalent of ordering a meal or finding a bathroom.

Like any other programming language, JavaScript is made up of several components, including

- Basic syntax rules
- Commands, values, and variables
- Operators and expressions
- Statements
- Loops
- Functions
- Arrays
- Object orientation

The following sections explore each of these.

Basic syntax rules

Every language has its own set of rules to make it possible to communicate. English, for example, uses periods to end sentences, quotation marks to denote quotes, and exclamation points to indicate something exciting is happening! JavaScript is no exception.

Here are some of the basic syntax rules that you should understand as you begin to discover what the scripting language is all about.

Statements

Just as an English or French document is composed of sentences, JavaScript scripts are composed of one or more *statements*. For example, the script in Listing 13-1 has a single statement, whereas the script in Listing 13-3 has more than 20. Statements can end simply by putting the next statement on the following line. You can also optionally end a statement with a semicolon.

Capitalization

JavaScript is a case-sensitive language. The text you type in a script must not only be spelled correctly but must also be in the correct case. For example, the `alert()` method we use earlier in this chapter is in the correct syntax for that method. If we use `Alert()` or `ALERT()`, the script won't work.

White space

JavaScript ignores spaces and tabs (usually called *white space*) between statements, but it's a good idea to use space to make your code more readable. For example, the following two code examples function in the same way, but the first is much easier to read than the second.

- The following code separates and organizes statements with spaces and line breaks, so it's easy to read and understand:

  ```
  if (document.images) {
      arrowRed = new Image
      arrowBlue = new Image

      arrowRed.src = "images/redArrow.gif"
      arrowBlue.src = "images/blueArrow.gif"
  }
  ```
- The following code separates statements with semicolons and doesn't use spaces and line breaks for organization, so it's harder to read:

  ```
  if (document.images) {arrowRed = new Image; arrowBlue = new Image;
  arrowRed.src ="images/redArrow.gif"; arrowBlue.src = "images/
  blueArrow.gif"}
  ```

Comments

Comments are text within your script that's ignored by the browser when the script runs. Comments are invaluable help to

- **Other people who are trying to figure out your code.**
- **You.** Months after you write the script, comments can make the code much easier for you to change.

Single-line comments

You can add comments to your JavaScript by adding two slashes to a comment that fits all on one line, like this:

```
//The code that runs below displays a snazzy pop-up window
```

Multiple-line comments

If your comment is lengthy and you need to span more than one line, you can either start each line with two slashes or else enclose your comments with `/*` and `*/` marks.

```
/* The code that runs below displays a really nifty, snazzy,
   jazzy, wicked-cool pop-up window.
   Last modified: June 10, 2007    */
```

Variables and data types

In JavaScript, you can execute various commands that are built in to the language itself, such as `alert()`, shown in Listing 13-1. However, you often use commands to act on pieces of information, known as *values*. For example, `alert()` displays a string value that is contained within its parentheses. A value can be either a literal value (such as a number or a string of alphanumeric characters) or a variable. Each value is categorized by its type.

Variables

A *variable* is a placeholder for a value. For example, the variable `favPerson` contains the string value *Gilligan.* In JavaScript, you can write this as `favPerson = "Gilligan"`.

The equals sign is read as "is set to." In other words, the variable `favPerson` now contains (is set to) the value "Gilligan." (The equals sign is an *assignment operator,* which is explained later in this chapter.) When assigning a value to a variable, keep in mind the following rules:

- The variable name is always on the left side of the equals sign.
- The variable value is always on the right side.

Here are examples of variables and the value that each contains:

```
x = 5
first_Time = false
formZipcode = "92683"
```

In the preceding example, `x` contains the numeric value of `5`. However, the `formZipcode` variable contains a text string, not a number, because that value (like all string values) is enclosed in double quotes.

TIP

If you need to perform mathematical operations on a variable, assign a number value to it, not a quoted string.

The actual act of creating a variable and assigning it a value is called *declaring* the variable. So, to declare the variable `pi` to be equal to `3.14`, you write this:

```
pi = 3.14
```

When you declare a variable, remember that

- **JavaScript is case-sensitive.**

  `myname`, `MyName`, and `myName` are treated as three separate variables because each has a different capitalization.
- **Variable names can use only letters, numbers, and underscores.**

  They can't contain spaces or other punctuation.
- **Variable names can't start with a number.**

- **Variable names can't be the same as a reserved word.**

  Reserved words are special keywords, such as `if` or `with`, that are used by JavaScript for its core functionality. Make sure you avoid naming a variable the same as one of these words. A complete list of reserved words is available at `www.javascripter.net/faq/reserved.htm`.

Data types

When you work with a literal value or variable, JavaScript categorizes it as a particular data type. Table 13-1 shows the common types of values.

Table 13-1 Data Types

Type	Description	Example
Number	Any numeric value	`42`
String	Text characters inside quote marks	`"My name is Inigo Montoya"`
Object	A JavaScript object, which can be defined by the language or else created on your own	`window`
Function	Value returned by a function	`myFunction()`
Boolean	A logically true or false value	`true`
Null	Empty; has no value	`null`

Operating on expressions

As the preceding sections discuss, a literal value (such as `5` and `"Lightbulb"`) or a variable can represent a value of a particular type. However, in JavaScript, a complete statement, called an *expression,* can also return a value. For example, consider the following two expressions:

```
2+1+2                               // Evaluates to a value of 5
"A" + "three" + "hour" + "tour"     // Evaluates to "Athreehourtour"
```

As you can see from these two examples, JavaScript often uses symbols as you evaluate, manipulate, and work with expressions. These symbols are called *operators*. In the examples shown above, the + (plus) symbol may be used to add numeric values or to combine two or more strings together into a single one (an operation known as *concatenation*).

JavaScript has several different types of operators, including assignment, arithmetic, counting, and comparison types.

Assignment operators

Assignment operators put values into variables. For example, `x = 8` assigns the value of `8` to the variable `x`. Table 13-2 shows the assignment operators — although, as you can see, they really combine assignment and arithmetic functionality.

Table 13-2 Assignment Operators

Operator	*Assignment*	*Description*
`=`	`x = y`	Sets `x` to the value of `y`
`+=`	`x += y`	Same as `x=x + y`
`-=`	`x -= y`	Same as `x=x - y`
`*=`	`x *= y`	Same as `x=x * y`
`/=`	`x /= y`	Same as `x=x / y`

Arithmetic operators

When you feel like crunching numbers, use arithmetic operators. You'll quickly recognize these symbols from your high-school math class. Expressions with the most common operators are listed in Table 13-3.

Table 13-3 Arithmetic Operators

Operator	*Example*	*Description*
`+`	`x + y` (numeric)	Adds `x` and `y` together
`-`	`x - y`	Subtracts `y` from `x`
`*`	`x * y`	Multiplies `x` and `y` together
`/`	`x / y`	Divides `x` by `y`
`-`	`-x`	Reverses the sign of `x`

Counting operators

JavaScript provides operators that are especially designed for counting either up or down while a process runs. The same operator can

- Retrieve a variable
- Count up or count down

Table 13-4 shows the counting operators.

Table 13-4 Counting Operators

Operator	*Description*
`++x`	Increases `y` by 1 (same as `x=x+1`) before an assignment
`x++`	Increases `y` by 1 after an assignment
`--x`	Decreases `y` by 1 (same as `x=x-1`) before an assignment
`x--`	Decreases `y` by 1 after an assignment

Changing before an assignment

When you place the `++` or `--` operators *before* the variable, the value of the variable changes *before* you use the variable. For example, if `x` is `8`, then `y=++x` changes the variables in this order:

1. Set `x` to `9`.
2. Set `y` to `9`.

Changing after an assignment

When you place the `++` or `--` operators *after* the variable, the value of the variable changes *after* you use the variable. For example, if `x` is 8, then `y=x++` changes the variables in this order:

1. Set `y` to `8`.
2. Set `x` to `9`.

Comparison operators

Comparison operators tell you whether expressions on both sides of the operator are the same or different. The result of a comparison operation is either `true` or `false`. Table 13-5 shows the comparison operators.

Table 13-5 Comparison Operators

Operator	*Example*	*Description*
`==`	`x == y`	Returns `true` if `x` and `y` are equal
`!=`	`x != y`	Returns `true` if `x` and `y` are not equal
`>`	`x > y`	Returns `true` if `x` is greater than`y`
`>=`	`x >= y`	Returns `true` if `x` is greater than or equal to `y`
`<`	`x < y`	Returns `true` if `x` is less than `y`
`<=`	`x <= y`	Returns `true` if `x` is less than or equal to `y`
`||`	`x || y`	Returns `true` if either `x` or `y` is true
`&&`	`x && y`	Returns `true` if both `x` and `y` are true
`!`	`!x`	Returns `true` if `x` is false

Working with statements

As discussed in the “Basic syntax rules” section, JavaScript statements are the basic units of a script. Two common types are expression statements and conditional statements.

Expression statement

An *expression statement* returns a value. For example, consider the following statement:

```
fullName = firstName + " " + lastName
```

The result of this expression is that the variable `fullName` is assigned the concatenated value of

- The value of the variable `firstName`
- A space
- The value of the variable `lastName`

The plus signs indicate that the result is *concatenated* — put together to form a string.

Conditional statement

A conditional statement can check your data and decide what to do. It has three steps:

REMEMBER

1. *Test* a value.

   The result of the test is always either `true` or `false`.
2. *Select* an action according to the result of the test.
3. *Perform* the selected action.

The most common conditional statements are the `if` and `if/else` statements. Consider the following `if` statement:

```
if ( x = "Boxen" ) {
     y = 1
     alert( "You are a smarty! The correct answer is Boxen. Well done!")
}
```

The `if` statement tests the expression inside the parentheses and determines whether or not `x = "Boxen"`. If the test evaluates to `true`, then the statements inside the curly braces are executed. If the test evaluates to `false`, then these lines are bypassed.

The `if/else` statement can also be used to specify code to be processed when the `if` test evaluates to `false`:

```
if ( x = "Boxen" ) {
     y = 1
     alert( "You are a smarty! The correct answer is Boxen. Well done!")
}
else {
     y = 0
     alert( "You are totally wrong! The correct answer is Boxen. Bad!")
}
```

A second example helps illustrate the steps involved in an `if/else` conditional statement:

```
if (confirm("Are you sure you want to do that?")) {
    alert("You said yes")
}
else {
    alert("You said no")
}
```

Here's how the statement works:

1. The `if` portion of the statement displays a dialog box that asks the user to confirm a choice, using the `confirm()` method.

   This is the "Are you sure you want to do that?" message you see on the left in Figure 13-1.

2. The `confirm()` method returns either `true` or `false`, depending on the user's response.

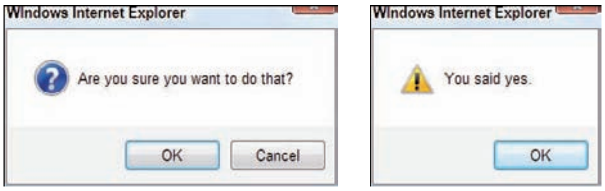
   • If the user clicks the OK button in the dialog box, the `confirm()` method returns `true`.

   • If the user clicks the Cancel button, the `confirm()` method returns `false`.

3. The code then performs an action based on the value that the `con-firm()` method returns.

   • If the method returns `true`, an alert appears with the message, "You said yes," as shown on the right in Figure 13-1.

   • If the method returns `false`, an alert appears with the message, "You said no."


Windows Internet Explorer

Are you sure you want to do that?

Windows Internet Explorer

You said yes.

Figure 13-1: Confirming a user action.

Loops

When you need to repeat an action in a JavaScript script, you use a *loop*. For example, a script that uses a loop can

- Make sure every character in a Zip code field is a number.
- Check every item in a list for a specific value

for loop

The `for` loop repeats steps a specific number of times.

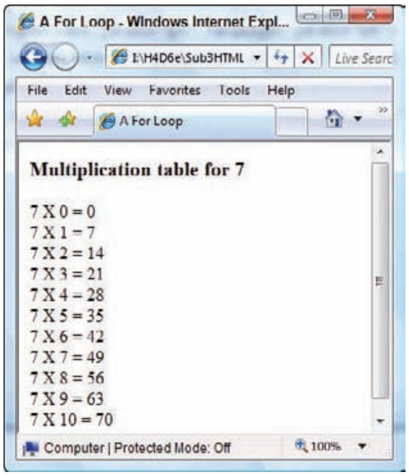
TIP

If you don't know how many times you need to repeat some steps, use a `while` loop instead of a `for` loop.

The `for` loop in Listing 13-4 calculates a multiplication table. Figure 13-2 shows the result in the browser.

Listing 13-4: A for Loop

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>A For Loop</title>
</head>
<body>
<script type="text/javascript" language="javascript">

document.write("<h3>Multiplication table for 7</h3>")

for (loopCount = 0; loopCount <= 10; loopCount++)
{
    document.write("7 X ",loopCount," = ", 7 * loopCount,"<br />");
}
</script>
</body>
</html>
```


A For Loop - Windows Internet Expl...
I:\H4D6e\Sub3HTML
Live Search
File Edit View Favorites Tools Help
A For Loop
Computer | Protected Mode: Off
100%

Multiplication table for 7

```
7 X 0 = 0
7 X 1 = 7
7 X 2 = 14
7 X 3 = 21
7 X 4 = 28
7 X 5 = 35
7 X 6 = 42
7 X 7 = 49
7 X 8 = 56
7 X 9 = 63
7 X 10 = 70
```

Figure 13-2: This script's for loop calculates and displays the multiplication result.

A `for` loop has three steps:

1. The *initialization step* sets the beginning value of the loop variable.

   In Listing 13-4, `loopCount = 0` is the initialization step.

2. The *limiting step* tells the loop when to stop looping.

   In Listing 13-4, `loopCount <= 10` is the limiting step. The loop repeats as long as the value of `loopCount` is less than or equal to 10.

3. The *increment step* tells the loop to increase the variable `loopCount` by a specific amount after the `for` block (the set of statements contained inside the curly braces) is executed.

   In Listing 13-4, `loopCount++` is the increment step. It increases the value of `loopCount` by 1 each time through the loop.

while loop

A `while` loop repeats steps until you get a certain kind of result (such as finding a name in a list).

TIP

If you know exactly how many times you need to repeat steps, use a `for` loop instead of a `while` loop.

Listing 13-5 shows the construction of the `while` loop.

Listing 13-5: A while Loop

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>A While Loop</title>
</head>
<body>
<script type="text/javascript" language="javascript">
con = confirm("Do you want to continue?")
while(con == false)
{
    document.write("Continuing to wait<br />");
    con = confirm("Do you want to continue?")
}
</script>
</body>
</html>
```

A `while` loop works as follows:

1. The `while` statement evaluates the expression inside its parentheses.

   In Listing 13-5, `con == false` is evaluated. The value of `con` is dependent on whether the user clicks the yes or no button in a confirmation message box.

2. As long as the expression evaluates to `true`, the code contained inside the curly braces (the `while` block) is repeated.

   In Listing 13-5, notice that the `confirm` statement is triggered again at the end of the `while` block to determine whether the loop should continue.

Functions

A *function* is a grouped set of JavaScript statements that

- Is identified by a name
- Is sectioned off from the rest of the script
- Performs a specific task
- Must be called by other parts of the script to execute

Functions are useful when you want to organize your code into separate units or when you use a bit of code more than once in a script. For example, a user may enter information into a form. You can use a function to save that information, perform a calculation on it, and allow other parts of the script to call the function to retrieve the result of the calculation.

A function consists of

- The *function declaration*, which contains the keyword function, a unique function name, and parentheses. Optionally, you can pass values into the function by adding arguments inside the parentheses.
- The *function block,* which is a set of one or more statements surrounded by curly braces.

The basic structure of a function looks like this:

```
function name_of_function(argument) {
    // One or more statements
}
```

Here is an example of a function:

```
function alertMessage() {
    alert("Please enter a value in this field.")
}
```

When your page loads into the browser and your script is processed by the browser, the function code doesn't run automatically. Instead, it has to be explicitly called in your script. Therefore, to trigger the `alertMessage()` function, you need to call it by name:

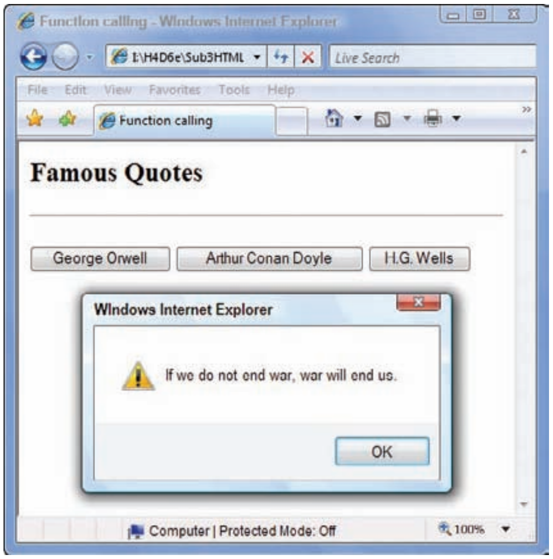
```
alertMessage()
```

Listing 13-6 shows a script with a function that is used to display a variety of alerts, depending on which button the user presses.

Listing 13-6: Calling a Function

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Function calling</title>
    <script language="javascript" type="text/javascript">
             function saySomething(message) {
               alert(message)
             }
    </script>
</head>
<body>
<h2>Famous Quotes</h2>
<hr />
<form action="#">
    <input type="button" value="George Orwell" onclick=
      "saySomething('To write or even speak English is not a science but
       an art.')" />
    <input type="button" value="Arthur Conan Doyle" onclick=
       "saySomething('We cannot command our love, but we can our actions.')" />
    <input type="button" value="H.G. Wells" onclick=
       "saySomething('If we do not end war, war will end us.')" />
</form>
</body>
</html>
```

The result of this script is shown in Figure 13-3.


Function calling - Windows Internet Explorer
I:\H4D6e\Sub3HTML
Live Search
File Edit View Favorites Tools Help
Function calling
Famous Quotes
George Orwell
Arthur Conan Doyle
H.G. Wells
Windows Internet Explorer
OK
Computer | Protected Mode: Off
100%

If we do not end war, war will end us.

Figure 13-3: An alert pops up as a result of clicking the H.G. Wells button.

In the script, when the user clicks one of the buttons on the page, the `say-Something` function is called and is passed the information in quotes, which the function stores in the variable `message`. The function then displays the alert, with the value of `message`, which is the quotation it was passed.

Arrays

An *array* is a collection of values. Arrays are useful because you can use them to manipulate and sort groups of things.

The location of information in an array is based on a numbered position called the *index*. Index numbering always starts at 0 and goes up. JavaScript has a special object — the `Array` object — just to handle arrays.

Creating arrays

To create an instance of an array, you must use the `new` operator along with the `Array` object, like this:

```
x = new Array()
```

TIP

You can fill in the array when you create the `Array` object, separating the array elements with commas, like so:

```
theDays = new Array("Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
                    "Saturday", "Sunday")
```

Accessing arrays

After the array is created, you can write to it and read from it by using the `[]` operator. By placing a position number in this operator, you can access the data stored at that index number.

For example, the value of `theDays[2]` in the preceding example is `Wednesday` (array positions always begin with `0`, so `Monday` is `0`). Please remember this, because many programming languages use 1 not 0 as their first array index value, and many new JavaScript programmers mistakenly think it works the same way.

Reading elements

To read an element from an array, create a variable and assign it a value from the array, like this:

```
thisDay = theDays[6]
```

The value of `thisDay` is now `Sunday`.

Writing elements

To write a value to the array, follow these steps:

1. Identify the index of the value you want to change.
2. Assign a new value to the array element, like this:

   ```
   theDays[0] = "Mon"
   ```

Looping

Every array has a `length` property, which is very useful for discovering how many elements the array contains, and is often used to loop through the array elements, as in this example:

```
planets = new Array ("Mercury", "Venus", "Earth", "Mars")
for (i = 0; i < planets.length; i++)
alert (planets[i]);
```

This causes the browser to display a series of four alert boxes, each containing one of the names of the `planets` array. The value of `planets.length` is `3` (since numbering starts at 0), and the script steps through each element of the array until the value of the counting variable `i` is greater than 3, at which time the script ends.

Objects

Most JavaScript scripts are designed to "give life" to objects that exist inside your browser. A rollover brings an image link to life. A validated e-mail address field is smart about what kind of e-mail address it will accept. A document displays new text on the fly, basing what it shows on a response from the Web page visitor.

Within JavaScript, you work with a variety of objects — such as the browser window, a button, a form field, an image, or even the document itself. Because JavaScript's primary calling is to work with objects, the scripting language is called an *object-based language*.

Think, for a moment, of an object that exists in the real world, such as a car or an MP3 player. Each of these objects has characteristics that describe the object, such as color, weight, and height. Many objects also have a behavior that can be triggered. A car can be started; an MP3 player can be played.

These real-world analogies can be applied to JavaScript. Objects you work with have descriptive qualities (called *properties*) and behaviors (called *methods*). For example, a `document` object represents the HTML page in your browser. It has properties, such as `linkColor`, `title`, and `location`, as well as methods, such as `open()`, `clear()`, and `write()`. (JavaScript methods always have parentheses following their names.)

JavaScript uses periods (or dots) to access an object's properties or methods:

```
object.property
object.method()
```

For example, to get the `title` of the `document` and assign it to a variable, you write this:

```
mytitle = document.title
```

To call the `clear` method of the document, you write this:

```
document.clear()
```

Events and Event Handling

Events are actions that either the browser executes or the user performs while visiting your page. Loading a Web page, moving the mouse over an image, closing a window, and submitting a form are all examples of events.

JavaScript deals with events by using commands called *event handlers.* Any action by the user on the page triggers an event handler in your script. Table 13-6 is a list of JavaScript's event handlers.

Table 13-6 Event Handlers

Event Handler	*Description*
`onabort`	User cancels a page load.
`onblur`	An element loses focus (and is no longer available through the event handler) because the user focuses on a different element.
`onchange`	User changes the contents of a form element or selects a different check box, radio button, or menu item.
`onclick`	User clicks an element with the mouse.
`ondblclick`	User double-clicks an element with the mouse.
`onerror`	Browser encounters an error in the scripts or other instructions on the page.
`onfocus`	An element becomes the focus of the user's attention — as does (for example) a form field when you start typing in it.
`onkeydown`	User presses and holds a key on the keyboard.
`onkeypress`	User presses and immediately releases a key on the keyboard.
`onkeyup`	User releases a depressed key.
`onload`	Browser loads an HTML page.
`onmouse-down`	User moves the mouse pointer over an element, presses the mouse button down, and holds it down.
`onmouse-move`	User moves the mouse pointer anywhere on the page.
`onmouseout`	User moves the mouse pointer off an element.
`onmou-seover`	User moves the mouse pointer over an element.
`onmouseup`	User releases a held mouse button.
`onreset`	User clicks a form's Reset button.
`onresize`	User resizes the browser window.
`onselect`	User selects a check box, radio button, or menu item from a form.
`onsubmit`	User clicks a form's Submit button.
`onunload`	Browser stops displaying one Web page because it's about to load another.

Not all objects support every event handler. For example, the `onload` handler is supported by only the `window` and `image` objects. The `onsubmit` event handler is supported by only the `form` object.

A common way to deal with event handlers is to use them as an attribute of an HTML element. This is called an *inline* event handler. Here is an example of the `onsubmit` inline event handler being used as an attribute of a `<form>` tag:

```
<form onsubmit="submitIt(this)" action="submitForm.cgi">
```

This example calls the `submitIt` function when the user clicks the form's Submit button. You can also embed JavaScript commands in the HTML, like this:

```
<input type="button" value="Click Me!"
   name="button1" onclick="alert("That tickles!");" />
```

A third way to use event handlers is to express them in JavaScript code, like this:

```
document.button1.onclick = function () { alert("That tickles!")}
```

Chapter 15 offers examples of event handlers.

Document Object Model (DOM)

JavaScript gives you the tools to manipulate the objects in a Web page. The *Document Object Model (DOM)* is the specification for how all those objects are represented. The DOM is a Web standard, defined by the World Wide Web Consortium — the W3C, for short. (More information than you can imagine about the DOM specification is available at `www.w3.org/DOM`.)

The DOM allows JavaScript to programmatically access and manipulate the contents of a document. The DOM defines

- Each object on a Web page
- Attributes associated with those objects
- Methods that you can use to manipulate those objects

By using the DOM, JavaScript can *dynamically* update the content, structure, and style of Web pages. This means that you can use JavaScript to produce effects in your Web pages, such as

- Rewriting your document on the fly
- Changing styles and style sheets
- Page layout

Other JavaScript Items of Interest

Table 13-7 lists other script- and forms-related markup attributes that you might find in (X)HTML files.

Table 13-7 Other Script- and Forms-related (X)HTML Attributes

Name	*Function/Value Equals*	*Value Types*	*Related Element(s)*
`declare`	Declares document object without invoking it	`"declare"`	`<script>`
`defer`	Allows user agent to defer script execution	`"defer"`	`<script>`

References and Resources

This part of the book presents the basics of the JavaScript language and how to add and adapt scripts that you find on the Web to your own HTML pages. But the JavaScript language is more powerful than that.

If you want to start writing your own code, you need more information. The best place to get your questions answered is online. Many resources on the Web can help you use JavaScript. Visit the Web site associated with this book and click the Chapter 13 link for a detailed list of Web sites and books that can help you create and use JavaScript.

Chapter 14

Working with Forms

In This Chapter

- Using forms in your Web pages
- Creating forms
- Working with form data
- Designing easy-to-use forms

Most of the HTML you write helps you display content and information for your users. Sometimes, however, you want a Web page to gather information from users instead of giving static information to them. HTML form markup tags give you a healthy collection of elements and attributes for creating forms to collect information from visitors to your site.

This chapter covers the many different uses for forms. It also shows you how to use form markup tags to create just the right form for soliciting information from your users, reviews your options for working with the data you receive, and gives you some tips for creating easy-to-use forms that really help your users provide the information you're looking for.

Uses for Forms

The Web contains millions of forms, but every form is driven by the same set of markup tags. Web forms can be short or long, simple or complex, and they have myriad uses. But they all fall into one of two broad categories:

- **Search forms** that let users search a site or the entire Web
- **Data-collection forms** that provide information for such uses as online shopping, technical support, site preferences, and personalization

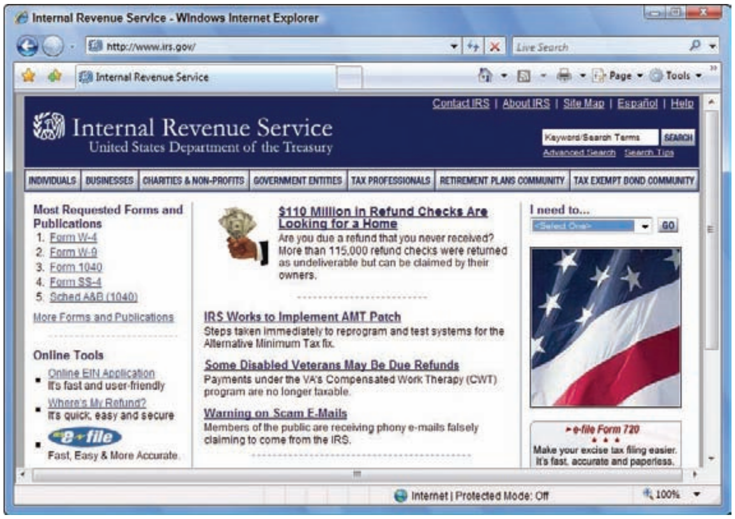
Before you create any form markup, you need to determine what kind of data your visitors will search for on your site and/or what kind of data you need to collect from visitors. Your data drives the form elements you use — and how you put them together on a page.

Searches

Search forms help you give visitors information.

The following search forms are from the friendly folks at the Internal Revenue Service (IRS). The difference between these search forms is the data the IRS site needs from you for its search:

- The IRS home page (shown in Figure 14-1) is a simple search form that uses two different single-field forms to help visitors search for general information and tax forms. This type of form can produce dozens of relevant responses. Visitors can both
  - Choose the best option.
  - Look at more than one option.
- A more complicated search form, such as the Refund Status page (as shown in Figure 14-2), produces only *one* specific response. It searches IRS records for the status of *your* refund. This page demands detailed information because the IRS doesn't want you to see anyone else's refund; therefore it both
  - Finds the data that visitors actually need.
  - Hides data that visitors shouldn't see.



Most Requested Forms and Publications

1. Form W-4
2. Form W-9
3. Form 1040
4. Form SS-4
5. Sched A&B (1040)

More Forms and Publications

Online Tools

- Online EIN Application
  It's fast and user-friendly
- Where's My Refund?
  It's quick, easy and secure
- Fast, Easy & More Accurate

$110 Million in Refund Checks Are Looking for a Home
Are you due a refund that you never received? More than 115,000 refund checks were returned as undeliverable but can be claimed by their owners.

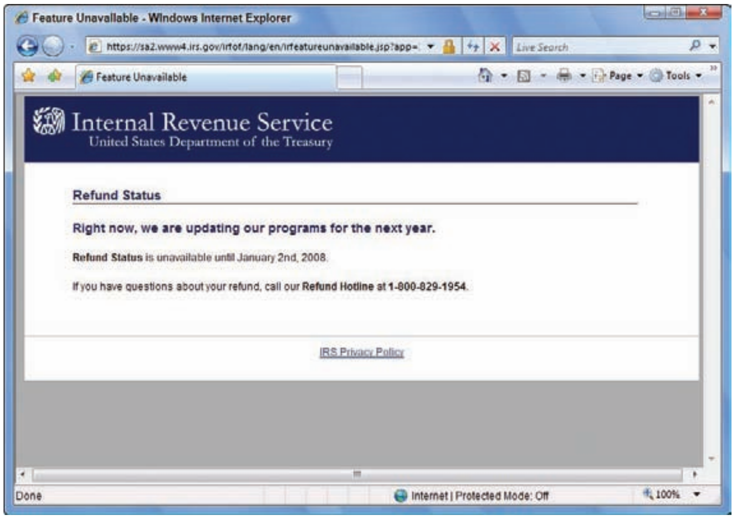
IRS Works to Implement AMT Patch
Steps taken immediately to reprogram and test systems for the Alternative Minimum Tax fix.

Some Disabled Veterans May Be Due Refunds
Payments under the VA's Compensated Work Therapy (CWT) program are no longer taxable.

Warning on Scam E-Mails
Members of the public are receiving phony e-mails falsely claiming to come from the IRS.

e-file Form 720
Make your excise tax filing easier. It's fast, accurate and paperless.

Figure 14-1: The IRS home page uses two short search forms.


Feature Unavailable - Windows Internet Explorer
https://sa2.www4.irs.gov/irfof/lang/en/irfeatureunavailable.jsp?app=
Live Search
Feature Unavailable
Page
Tools
Internal Revenue Service
United States Department of the Treasury
IRS Privacy Policy
Done
Internet | Protected Mode: Off
100%

Refund Status

Right now, we are updating our programs for the next year.

Refund Status is unavailable until January 2nd, 2008.

If you have questions about your refund, call our **Refund Hotline at 1-800-829-1954**.

Figure 14-2: The refund-status search form is a little more complex.

REMEMBER

Searches come in all shapes and sizes, so the search forms that drive those searches should come in all shapes and sizes, too. A short keyword search might do the trick, or you might need a more sophisticated search method.

Data collection

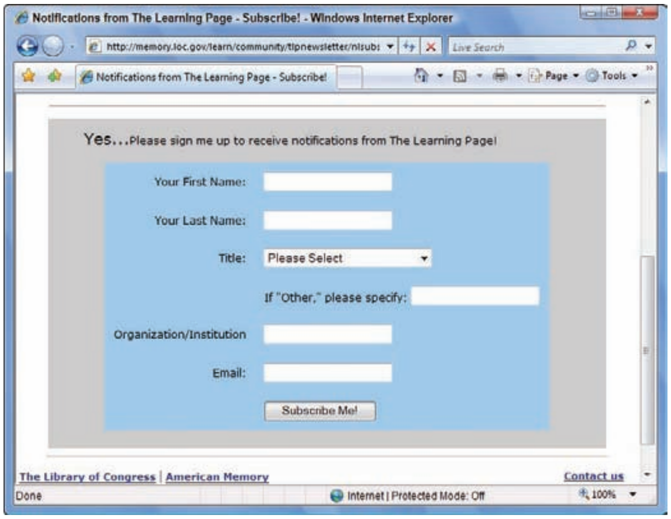
Data-collection forms receive information you want to process or save. When you create a form that collects information, the information you need is what drives the structure and complexity of the form:

- If you need just a little information, the form may be short and (relatively) sweet.

  The Library of Congress (LC) uses a form to collect information from teachers to subscribe to a free electronic newsletter, as shown in Figure 14-3. The LC doesn't need much information to set up the subscription, so the form is short and simple.

- If you need a lot of information, your form may be several pages long.

  RateGenius uses long and detailed forms to gather the information it needs to help customers get the best possible loan rate. The page in Figure 14-4 is just the first of several that a visitor must fill out to provide all the necessary information.



Notifications from The Learning Page - Subscribe! - Windows Internet Explorer

http://memory.loc.gov/learn/community/tipnewsletter/nlsub

Yes…Please sign me up to receive notifications from The Learning Page!

Your First Name:

Your Last Name:

Title: Please Select

If "Other," please specify:

Organization/Institution

Email:

Subscribe Me!

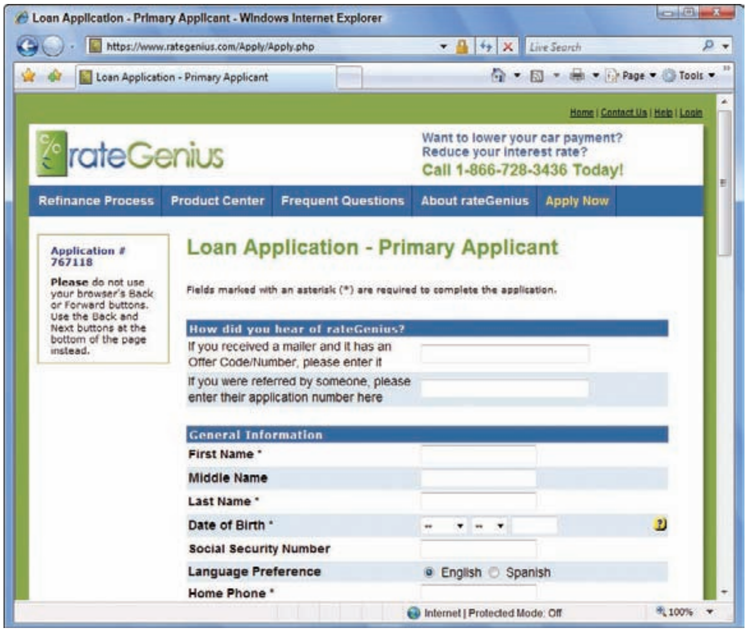
The Library of Congress | American Memory

Contact us

Done

Internet | Protected Mode: Off

Figure 14-3: A free subscription form collects basic information.



Loan Application - Primary Applicant - Windows Internet Explorer

https://www.rategenius.com/Apply/Apply.php

Home | Contact Us | Help | Login

rateGenius

Want to lower your car payment?
Reduce your interest rate?
Call 1-866-728-3436 Today!

Refinance Process | Product Center | Frequent Questions | About rateGenius | Apply Now

Application # 767118

Please do not use your browser's Back or Forward buttons. Use the Back and Next buttons at the bottom of the page instead.

Loan Application - Primary Applicant

Fields marked with an asterisk (*) are required to complete the application.

How did you hear of rateGenius?

If you received a mailer and it has an Offer Code/Number, please enter it

If you were referred by someone, please enter their application number here

General Information

First Name *

Middle Name

Last Name *

Date of Birth *

Social Security Number

Language Preference [✓ English] Spanish

Home Phone *

Internet | Protected Mode: Off

Figure 14-4: An online car-loan site uses many detailed forms to collect necessary data.

Creating Forms

HTML form markup tags and attributes can

- **Define the overall form structure.**

  Every form has the same basic structure.
- **Tell the Web browser how to handle the form data.**
- **Create input objects (such as text fields and drop-down lists).**

  Which input elements you use depends on the data you're collecting.

Structure

All of the input elements associated with a single form are

- Contained within a `<form>` tag
- Processed by the same form handler

  A *form handler* is a program on the Web server (or a simple `mailto` URL) that manages the data a user sends to you through the form. A Web browser can only *gather* information through forms; it doesn't know what to do with the information once it has it. You must provide some other mechanism to actually *do* something useful with the data you collect. (This chapter covers form handlers in detail later.)

Attributes

You always use these two key attributes with the `<form>` tag:

- `action`: The URL of the form handler.
- `method`: How you want the form data to be sent to the form handler.

  Your form handler dictates which of these values to use for `method`:

  - `get` sends the form data to the form handler on the URL.
  - `post` sends the form data in the Hypertext Transfer Protocol (HTTP) header.

  Webmonkey offers a good overview of the difference between `get` and `post` in its "Good Forms" article:

  ```
  http://www.webmonkey.com/99/30/index4a_page3.html
  ```

Markup

The markup in Listing 14-1 creates a form that uses the `post` method to send user-entered information to a form handler (`guestbook.cgi`) to be processed on the Web server.

Listing 14-1: A Simple Form Processed by a Form Handler

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Forms</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
    <form action="cgi-bin/guestbook.cgi" method="post">

    <!-- form input elements go here -->

    </form>
</body>
</html>
```

REMEMBER

The value of the `action` attribute is a URL, so you can use absolute or relative URLs to point to a form handler on your server.

Input tags

The tags you use to solicit input from your site visitors make up the bulk of any form. HTML supports a variety of different input options — from text fields to radio buttons and from files to images.

Every input control associates some value with a name:

- When you create the control, you give it a name.
- The control sends back a value based on what the user does in the form.

For example, if you create a text field that collects a user's first name, you might name the field `firstname`. When the user types his or her first name in the field and submits the form, the value associated with `firstname` is whatever name the user typed in the field.

The whole point of a form is to gather values associated with input controls, so the way you set the name and value for each control is important. The following sections explain how you should work with names and values for each of the input controls.

Input fields

You can use a variety of input fields in your forms.

For input elements that require a user to select an option (such as a check box or radio button) rather than typing something into a field, you define both the name and the value. When the user selects a box or a button and then clicks the Submit button, the form returns the name and value assigned to the element.

Text

Text fields are single-line fields that users can type information into.

- To define the input field as a text field, use the `<input />` element with the `type` attribute set to `text`.

  ```
  <input type="text" />
  ```

- You use the `name` attribute to give the input field a name.

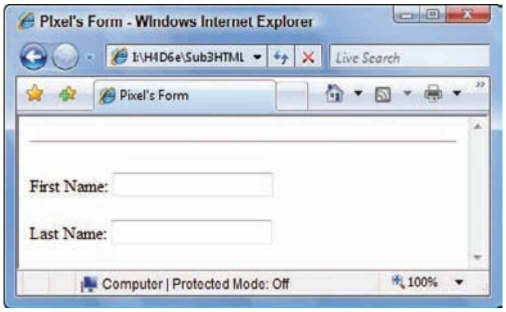
  ```
  <input type="text" name="firstname" />
  ```

- The user supplies the value when he or she types in the field.

This markup creates two text input fields — one for a first name and one for a last name:

```
<form action="cgi-bin/guestbook.cgi" action="post">
  <p>First Name: <input type="text" name="firstname" /></p>
  <p>Last Name: <input type="text" name="lastname" /></p>
</form>
```

In addition to the `<input />` elements, the preceding markup includes paragraph (`<p>`) elements and some text to label each of the fields. By themselves, most form elements don't give the user many clues about the type of information you want them to enter. You also must use HTML block and inline elements to format the appearance of your form. Figure 14-5 shows how a browser displays this HTML.


Pixel's Form - Windows Internet Explorer
I:\H4D6e\Sub3HTML
Live Search
Pixel's Form
Computer | Protected Mode: Off
100%

First Name:

Last Name:

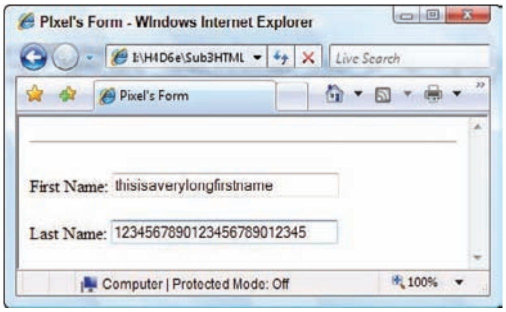
Figure 14-5: Text-entry fields in a form.

You can control the size of a text field with these attributes:

- `size`: The length (in characters) of the text field
- `maxlength`: The maximum number of characters the user can type into the field

The following markup creates a form that sets both fields to a `size` of 30 and a `maxlength` of 25. Each field will be about 30 characters long; even so, a user can type only 25 characters into each field, as shown in Figure 14-6.

```
<form action="cgi-bin/guestbook.cgi" action="post">
<p>First Name: <input type="text" name="firstname" size="30"
                maxlength="25" /></p>
<p>Last Name: <input type="text" name="lastname" size="30"
               maxlength="25" /></p>
</form>
```


Pixel's Form - Windows Internet Explorer
E:\H4D6e\Sub3HTML
Live Search
Pixel's Form
Computer | Protected Mode: Off
100%

First Name: thisisaverylongfirstname

Last Name: 1234567890123456789012345

Figure 14-6: You can specify the length and maximum number of characters for a text field.

Passwords

A *password field* is a text field that doesn't display what the user types. Someone looking over the user's shoulder sees each keystroke represented on the screen by a *placeholder character,* such as an asterisk or bullet.

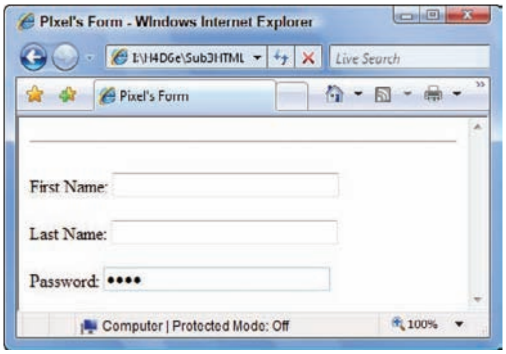
You create a password field by using the `<input />` element with the `type` attribute set to `password`, as follows:

```
<form action="cgi-bin/guestbook.cgi" action="post">
<p>First Name: <input type="text" name="firstname" size="30"
                maxlength="25" /></p>
<p>Last Name: <input type="text" name="lastname" size="30" maxlength="25" /></p>
<p>Password: <input type="password" name="psswd" size="30" maxlength="25" /></p>
</form>
```

TIP

Password fields are programmed like text fields.

Figure 14-7 shows how a browser replaces what you type with bullets. (Some browsers may replace the text with asterisks or some other character. It depends on the browser's default settings.)


Pixel's Form - Windows Internet Explorer
E:\H4D6e\Sub3HTML
Live Search
Pixel's Form
Computer | Protected Mode: Off
100%

First Name:

Last Name:

Password: ••••

Figure 14-7: Password fields mask the text a user enters.

Check boxes and radio buttons

If only a few possible values are available to the user, you can give him or her a collection of options to choose from:

- **Check boxes:** Choose *more* than one option.
- **Radio buttons:** Choose only *one* option.

TIP

If many choices are available, use a drop-down list instead of radio buttons or check boxes.

To create radio buttons and check boxes, you

- Use the `<input />` element with the `type` attribute set to `radio` or `checkbox`.
    - If the attribute value is `radio`, a round radio button appears.
    - If it's `checkbox`, a check box appears.

  Radio buttons differ from check boxes in an important way: Users can select a single radio button from a set of options but can select any number of check boxes (including none, one, or more than one).
- Create each option with these attributes:
    - The `name` attribute to give the option a name.
    - The `value` attribute to specify what value is returned if the user selects the option.

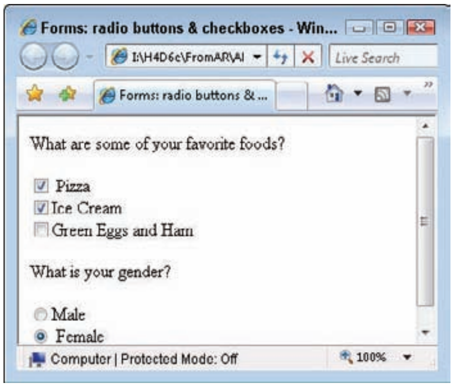
TIP

You can use the checked attribute (with a value of checked) to specify that an option should be already selected when the browser displays the form. This is a good way to specify a default selection in a list.

This markup shows how to format check-box and radio-button options:

```
<form action="cgi-bin/guestbook.cgi" action="post">
<p>What are some of your favorite foods?</p>
<p><input type="checkbox" name="food" value="pizza" checked="checked" />
    Pizza<br />
   <input type="checkbox" name="food" value="icecream" />Ice Cream<br />
   <input type="checkbox" name="food" value="eggsham" />Green Eggs and Ham<br />
</p>

<p>What is your gender?</p>
<p><input type="radio" name="gender" value="male" />Male<br />
   <input type="radio" name="gender" value="female" checked="checked" />
    Female</p>
</form>
```

In the preceding code, each set of options uses the same name for each input control but gives a different value to each option. You give each item in a set of options the same name to let the browser know they're part of a set. Figure 14-8 shows how a browser displays this markup, where we've also checked the box for "ice cream" and left the default check next to "pizza" as-is. If you want to, in fact, you can check as many boxes as you by default in the page markup, simply by included checked="checked" in each <input … /> element you choose to check in advance.


Forms: radio buttons & checkboxes - Win...
I:\H4D6c\FromAR\AI
Live Search
Forms: radio buttons & ...
Computer | Protected Mode: Off
100%

What are some of your favorite foods?

☑ Pizza
☑ Ice Cream
☐ Green Eggs and Ham

What is your gender?

○ Male
● Female

Figure 14-8: Check boxes and radio buttons.

Hidden fields

A *hidden field* gives you a way to collect name and value information that the user can't see along with the rest of the form data. Hidden fields are useful for keeping track of information associated with the form (such as its version or name).

TECHNICAL STUFF

If your Internet service provider (ISP) provides a generic application for a guest book or feedback form, you might have to put your name and e-mail address in the form's hidden fields so the data goes specifically to you.

To create a hidden field, you

- Use the `<input />` element with its `type` attribute set to `hidden`.
- Supply the name and value pair you want to send to the form handler.

Here's an example of markup for a hidden field:

```
<form action="cgi-bin/guestbook.cgi" action="post">
<input type="hidden" name="e-mail" value="me@mysite.com" />
<p>First Name: <input type="text" name="firstname" size="30"
               maxlength="25" /></p>
<p>Last Name: <input type="text" name="lastname" size="30" maxlength="25" /></p>
<p>Password: <input type="password" name="psswd" size="30" maxlength="25" /></p>
</form>
```

TRICKS OF THE TRADE

As a general rule, using your e-mail address in a hidden field is just asking for your address to be picked up by spammers. If your ISP says that this is how you should do your feedback form, ask them if they have any suggestions for how you can minimize the damage. Surfers to your page can't see your e-mail address, but spammers' spiders can read the underlying tags. At a minimum, you would hope that your ISP supports one of the many JavaScript encryption tools available to obscure e-mail addresses from harvesters.

File uploads

A form can receive documents and other files, such as images, from users. When the user submits the form, the browser grabs a copy of the file and sends it with the other form data. To create this *file-upload field,*

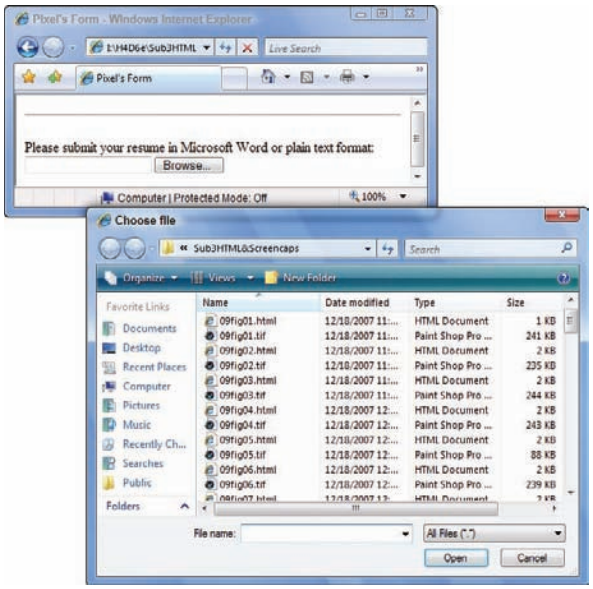
- Use the `<input />` element with the `type` attribute set to `file`.

  The file itself is the form field value.
- Use the `name` attribute to give the control a name.

Here's an example of markup for a file-upload field:

```
<form action="cgi-bin/guestbook.cgi" action="post">
<p>Please submit your resume in Microsoft Word or plain text format:<br />
   <input type="file" name="resume" />
</p>
</form>
```

Browsers render a file-upload field with a browse button that allows a user to surf his or her local hard drive and select a file to send to you, as in Figure 14-9.



Please submit your resume in Microsoft Word or plain text format:
Browse...

Name	Date modified	Type	Size
09fig01.html	12/18/2007 11:...	HTML Document	1 KB
09fig01.tif	12/18/2007 11:...	Paint Shop Pro ...	241 KB
09fig02.html	12/18/2007 11:...	HTML Document	2 KB
09fig02.tif	12/18/2007 11:...	Paint Shop Pro ...	235 KB
09fig03.html	12/18/2007 11:...	HTML Document	2 KB
09fig03.tif	12/18/2007 11:...	Paint Shop Pro ...	244 KB
09fig04.html	12/18/2007 12:...	HTML Document	2 KB
09fig04.tif	12/18/2007 12:...	Paint Shop Pro ...	243 KB
09fig05.html	12/18/2007 12:...	HTML Document	2 KB
09fig05.tif	12/18/2007 12:...	Paint Shop Pro ...	88 KB
09fig06.html	12/18/2007 12:...	HTML Document	2 KB
09fig06.tif	12/18/2007 12:...	Paint Shop Pro ...	239 KB
09fig07.html	12/18/2007 12:...	HTML Document	2 KB

Figure 14-9: A file-upload field.

WARNING!

When you accept users' files through a form, you can receive files that are either huge or are infected by viruses. Consult with whoever is programming your form handler to discuss options for protecting the system where files are saved. Several barriers can help minimize your risks, including

- Virus-scanning software
- Restrictions on file size
- Restrictions on file type

Drop-down lists

Drop-down lists are a great way to give users lots of options in a little screen space. You use two different tags to create a drop-down list:

- `<select>` holds the list.

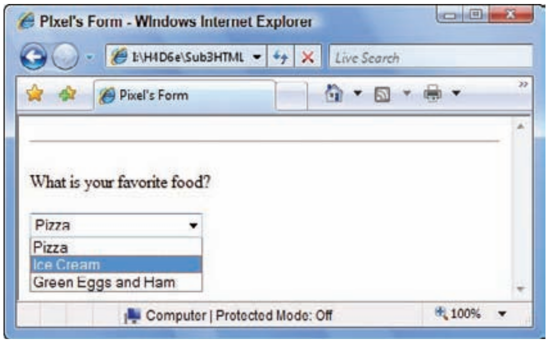
  Use a `name` attribute with the `<select>` element to name the entire list.
- A collection of `<option>` elements identifies the list options.

  The `value` attribute assigns a unique value for each `<option>` element.

Here's an example of markup for a drop-down list:

```
<form action="cgi-bin/guestbook.cgi" action="post">
<p>What is your favorite food?</p>
<select name="food">
  <option value="pizza">Pizza</option>
  <option value="icecream">Ice Cream</option>
  <option value="eggsham">Green Eggs and Ham</option>
</select>
</form>
```

The browser turns this markup into a drop-down list with three items, as shown in Figure 14-10.


Pixel's Form - Windows Internet Explorer
E:\H4D6e\Sub3HTML
Live Search
Pixel's Form
Computer | Protected Mode: Off
100%

What is your favorite food?

Pizza
Pizza
Ice Cream
Green Eggs and Ham

Figure 14-10: A drop-down list.

TIP

You can enable users to select more than one item from a drop-down list by changing the default settings of your list:

TECHNICAL STUFF

- If you want your user to be able to choose more than one option (by holding down the Alt [Windows] or ⌘ [Mac] key while clicking options in the list), add the `multiple` attribute to the `<select>` tag. The value of `multiple` is `multiple`.

  Because of XHTML rules, standalone attributes cannot stand alone; therefore, the value is the same as the name of the attribute.

- By default, the browser displays only one option until the user clicks the drop-down menu's arrow to display the rest of the list. Use the `size` attribute with the `<select>` tag to specify how many options to show.

  If you specify fewer than the total number of options, the browser includes a scroll bar with the drop-down list.

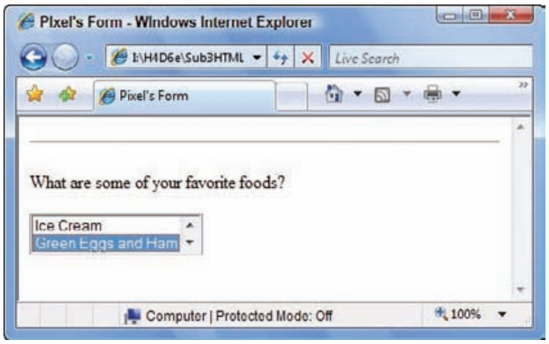
You can specify that one of the options in the drop-down list be already selected when the browser loads the page, just as you can specify a check box or radio button to be checked. Simply add the `selected` attribute to have a value of `selected` for the `<option>` tag you want as the default.

The following markup

- Allows the user to choose more than one option from the list
- Displays two options
- Selects the third option in the list by default

```
<form action="cgi-bin/guestbook.cgi" action="post">
<p>What are some of your favorite foods?</p>
<select name="food" size="2" multiple="multiple">
 <option value="pizza">Pizza</option>
 <option value="icecream">Ice Cream</option>
 <option value="eggsham" selected="selected">Green Eggs and Ham</option>
</select>
</form>
```

Figure 14-11 shows how adding these attributes modifies the appearance of the list in a browser.


Pixel's Form - Windows Internet Explorer
E:\H4D6e\Sub3HTML
Live Search
Pixel's Form
Computer | Protected Mode: Off
100%

What are some of your favorite foods?

Ice Cream
Green Eggs and Ham

Figure 14-11: A drop-down list with modifications.

Multi-line text boxes

If a single-line text field isn't enough room for responses, create a *text box* instead of a text field:

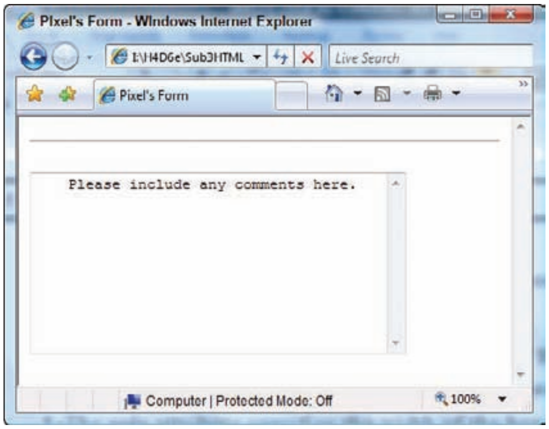
- The `<textarea>` element defines the box and its parameters.
- The `rows` attribute specifies the height of the box in rows based on the font in the text box.
- The `cols` attribute specifies the width of the box in columns based on the font in the text box.

The text that the user types into the box provides the value, so you need only give the box a name with the `name` attribute:

```
<form action="cgi-bin/guestbook.cgi" action="post">
  <textarea rows="10" cols="40" name="comments">
    Please include any comments here.
  </textarea>
</form>
```

TIP

Any text you include between the `<textarea>` and `</textarea>` tags appears in the text box in the browser, as shown in Figure 14-12. The user then enters information in the text box and overrides your text.


Pixel's Form - Windows Internet Explorer
I:\H4D6e\Sub3HTML
Live Search
Pixel's Form
Computer | Protected Mode: Off
100%

Please include any comments here.

Figure 14-12: A text box.

Submit and reset

Submit and Reset buttons help the user tell the browser what to do with the form. You can create buttons to either submit or reset your form, using the `<input />` element with the following `type` and `value` attributes:

- **Submit**

  Visitors have to tell a browser when they're done with a form and want to send the contents. You create a button to *submit* the form to you by using this markup:

  ```
  <input type="submit" value="Submit" />
  ```

  You don't use the `name` attribute for the Submit and Reset buttons. You use the `value` attribute instead to specify how the browser labels the buttons for display.

- **Reset**

  Visitors need to clear the form if they want to start all over again or decide not to fill it out. You create a button to reset, or *clear,* the form by using the following markup:

  ```
  <input type="reset" value="Clear" />
  ```

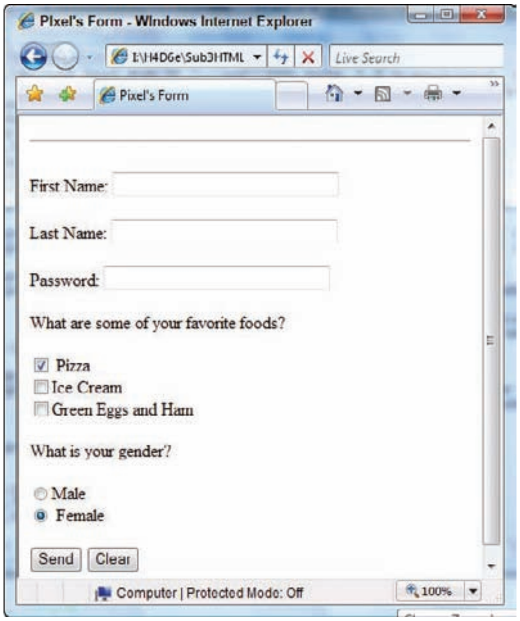
TIP

You can set the value to anything you want to appear on the button. In our example, we set ours to `Clear`. (You can use something that's more appropriate to you if you'd like.)

Here's an example of markup to create Submit and Reset buttons named Send and Clear, respectively:

```
<form action="cgi-bin/guestbook.cgi" action="post">
<p>First Name: <input type="text" name="firstname" size="30"
                maxlength="25" /></p>
<p>Last Name: <input type="text" name="lastname" size="30" maxlength="25" /></p>
<p>Password: <input type="password" name="psswd" size="30" maxlength="25" /></p>

<p>What are some of your favorite foods?</p>
<p><input type="checkbox" name="food" value="pizza" checked="checked" />
    Pizza<br />
   <input type="checkbox" name="food" value="icecream" />Ice Cream<br />
   <input type="checkbox" name="food" value="eggsham" />Green Eggs and Ham<br />
</p>

<p>What is your gender?</p>
<p><input type="radio" name="gender" value="male" />Male<br />
   <input type="radio" name="gender" value="female" checked="checked" />
    Female</p>

<p>
   <input type="submit" value="Send" />
   <input type="reset" value="Clear" />
</p>
</form>
```

Figure 14-13 shows how a browser renders these buttons in a form.


Pixel's Form - Windows Internet Explorer
E:\H4D6e\Sub3HTML
Live Search
Pixel's Form
Computer | Protected Mode: Off
100%

First Name:

Last Name:

Password:

What are some of your favorite foods?

☑ Pizza
☐ Ice Cream
☐ Green Eggs and Ham

What is your gender?

○ Male
● Female

Send Clear

Figure 14-13: Submit and reset buttons labeled as Send and Clear.

Customizing

If you don't like the Submit and Reset buttons that a browser creates, you can substitute your own graphical buttons by using

- The `<input />` element with a `type` of `image`.
- An `src` attribute that specifies the image's location.
- A `value` that defines the result of the field:
  - For an image that submits the form, set `value` to `submit`.
  - For an image that clears the form, set `value` to `reset`.

TIP

Use the `alt` attribute to provide alternative text for browsers that don't show images (or for users who can't see them).

The following markup creates customized Submit and Reset buttons:

```
<p><input type="image" value="submit" src="submit_button.gif" alt="Submit" />
   <input type="image" value="reset" src="reset_button.gif" alt="Clear" />
</p>
```

Validation

No matter how brilliant your site's visitors may be, there's always a chance that they'll enter data you aren't expecting. JavaScript to the rescue!

Form validation is the term for checking the data the user enters before it's put into your database. Check the data with both *JavaScript* and Common Gateway Interface *(CGI)* scripts on your server.

JavaScript

You can validate entries in JavaScript before data goes to the server. This means that visitors don't wait for your server to check the data — they're told quickly (before they click Submit, if you want) if there's a problem.

TIP

If you want to use JavaScript in your forms and on your Web site, you can learn more about it in Chapters 12 and 13 of this book, or online at:

- `www.w3schools.com/js/default.asp`
- `www.quirksmode.org/js/forms.html`
- `http://www.webmonkey.com/programming/javascript/`

CGI

You need to validate your form data on the server side because users can surf with JavaScript turned off. (They'll have a slower validation process.) Find out more about CGI in the next section and at

- `www.4guysfromrolla.com/webtech/LearnMore/Validation.asp`
- `www.cgi101.com/book`

Processing Data

Getting form data is really only half the form battle. You create form elements to get data from users, but then you have to do something with that data. Of course, your form and your data are unique every time, so no single, generic form handler can manage the data for every form. Before you can find (or write) a program that handles your form data, you must know what you want to do with it. For example . . .

- If you just want to receive comments from a Web form by e-mail, you might need only a simple `mailto:` URL.
- If a form gathers information from users to display in a guest book, you
  - Add the data to a text file or a small database that holds the entries.
  - Create a Web page that displays the guest-book entries.
- If you need a shopping cart, you need programs and a database that can handle inventory, customer-order information, shipping data, and cost calculations.

TIP

Your Web-hosting provider — whether it's an internal IT group or an ISP to which you pay a monthly fee — has the final say in what kind of applications you can use on your Web site to handle form data. If you want to use forms on your site, be sure that your hosting provider supports the applications you need to run on the server to process form input data (which will normally use the `post` or `get` methods we discuss earlier in this chapter). Chapter 3 includes more information on finding the right ISP to host your pages.

Using CGI scripts and other programs

Typically, form data is processed in some way or another by a Common Gateway Interface (CGI) script written in some programming language such as Perl, Java, AppleScript, or one of many other languages that run on Web servers. These scripts make the data from your form useful by

- Putting it into a database
- Creating customized HTML based on it
- Writing it to a *flat file* — computer-geek speak for a plain, unadorned text file, or one that uses commas or tab characters on individual lines of text to separate field values, aka CSV for "comma-separated values" or TSV for "tab-separated values.

TIP

If you aren't familiar with CGI scripts and how they work, the "CGI Scripts for Fun and Profit" article on Webmonkey provides an excellent overview:

```
http://www.webmonkey.com/99/26/index4a.html
```

You don't have to be a programmer to make the most of forms. Many ISPs support (and provide) scripts for processing common forms such as guest books, comment forms, and even shopping carts. Your ISP may give you

- All the information you need to get the program up and running
- HTML to include in your pages

WARNING!

You can tweak the markup that manages how the form appears in the canned HTML you get from an ISP, but don't change the form itself — especially the form tag names and values. The Web-server program uses these to make the entire process work.

Several online script repositories provide free scripts that you can download and use along with your forms. Many of these also come with some generic HTML you can dress up and tweak to fit your Web site. You simply drop the program that processes the form into the folder on your site that holds programs (usually called `cgi-bin`), add the HTML to your page, and you're good to go. Some choice places on the Web to find scripts you can download and put to work immediately are

- **Matt's Script archive:** `www.scriptarchive.com/nms.html`
- **The CGI Resource Index:** `http://cgi.resourceindex.com`
- **ScriptSearch:** `www.scriptsearch.com`

TECHNICAL STUFF

If you want to use programs that aren't provided by your ISP on your Web site, you need complete access to your site's cgi-bin folder. Every ISP's setup is different, so read your documentation to find

- Whether your ISP allows you to use CGI scripts in your Web pages
- Languages the ISP supports (Perl is a safe bet, but it's safer to be sure.)

Sending data by e-mail

You can opt to receive your form data from e-mail instead of using a script or other utility to process a form's data. You get just a collection of name-and-value pairs tucked into a text file sent to your e-mail address, but that isn't necessarily a bad thing. You can include a short contact form on your Web site that asks people to send you feedback (a feature that always looks professional); then you can simply include, in the action URL, the e-mail address you want the data sent to:

```
<form action="mailto:me@mysite.com" action="post">
```

Many spam companies get e-mail addresses by trolling Web sites looking for `mailto` URLs. Consider setting up a special e-mail account just for comments — that way, your regular e-mail address won't get pulled onto spam mailing lists. On the other hand, you can also use JavaScript based e-mail address encryption tools that will garble and disguise the contents of such addresses — as long as they can be un-encrypted on the receiving end, that is!

Designing User-Friendly Forms

Designing *useful* forms is a different undertaking from designing *easy-to-use* forms. Your form may gather the data that you need, but if it's hard for visitors to use, they may abandon it before they're done.

As you use the markup elements from this chapter, along with the other elements that drive page layout, keep the following guidelines in mind:

- **Provide textual cues for all your forms.** Be clear about
  - Information you want
  - Format you need

  For example, tell users such inputting details as whether
  - Dates must be entered as `mm/dd/yy` (or as `mm/dd/yyyy`).
  - The number of characters a field can take is limited.

    Characters can be limited with the `maxlength` attribute.
- **Use field width and character limits to provide visual clues.** For example, if users should enter a phone number as *xxx-xxx-xxxx,* consider creating three text fields — one for each part of the phone number.
- **Group similar fields together.** A logical grouping of fields makes filling out a form easier. It's confusing if you ask for the visitor's first name, then birthday, then last name.
- **Break long forms into easy-to-manage sections.** Forms in short chunks are less intimidating and more likely to be completed.

  Major online retailers (such as Amazon.com) use this method to get the detail they need for orders without making the process too painful.

TIP

- **Mark required fields clearly.** If some parts of your form *can't* be left blank when users submit the form, mark those fields clearly.

  You can identify required fields by

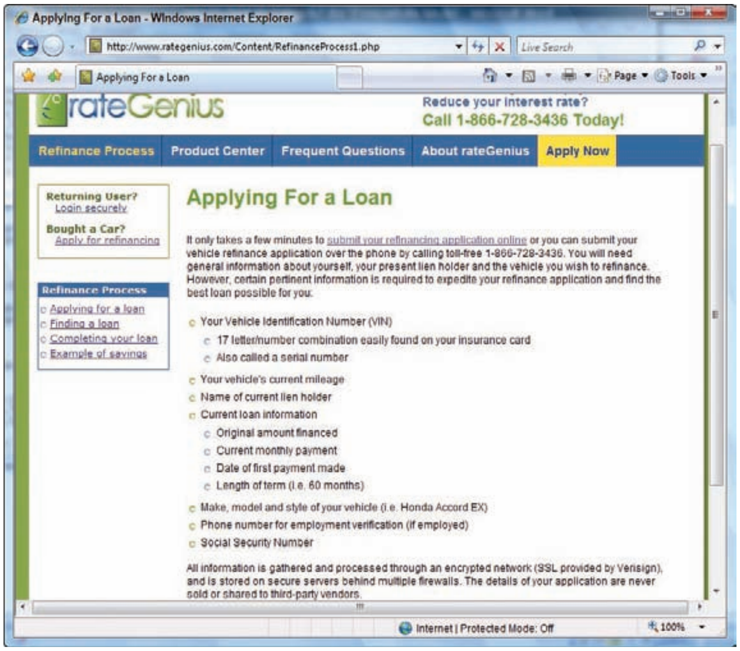
  - Making them bold
  - Using a different color
  - Placing an asterisk beside them

- **Tell users what kind of information they need for the form.** If users need any information in their hands before they fill out your form, a *form gateway* page can detail everything users should have before they start filling out the form.

  The RateGenius Apply For a Loan page (shown in Figure 14-14) lays out clearly for visitors about to fill out a long form exactly what information to prepare before starting.

TIP

The series of forms RateGenius uses to gather information for car loans and loan refinancing are excellent examples of long forms that collect a variety of different kinds of data by using all the available form markup elements. Visit `www.rategenius.com` to review its form techniques.



Applying For a Loan

It only takes a few minutes to submit your refinancing application online or you can submit your vehicle refinance application over the phone by calling toll-free 1-866-728-3436. You will need general information about yourself, your present lien holder and the vehicle you wish to refinance. However, certain pertinent information is required to expedite your refinance application and find the best loan possible for you:

- Your Vehicle Identification Number (VIN)
  - 17 letter/number combination easily found on your insurance card
  - Also called a serial number
- Your vehicle's current mileage
- Name of current lien holder
- Current loan information
  - Original amount financed
  - Current monthly payment
  - Date of first payment made
  - Length of term (i.e. 60 months)
- Make, model and style of your vehicle (i.e. Honda Accord EX)
- Phone number for employment verification (if employed)
- Social Security Number

All information is gathered and processed through an encrypted network (SSL provided by Verisign), and is stored on secure servers behind multiple firewalls. The details of your application are never sold or shared to third-party vendors.

Figure 14-14: A form-gateway page helps users prepare to fill out a long form.

Other Noteworthy Forms-Related Markup

Table 14-1 lists other forms-related (X)HTML markup attributes that you might find in HTML files.

Table 14-1 Other Forms-Related (X)HTML Attributes

Name	*Function/Value Equals*	*Value Types*	*Related Element(s)*
`Accept`	Lists acceptable MIME types for file upload	CS Media types	`<form>` `<input />`
`accept-charset`	Lists character encodings	SS Encodings	`<form>`
`Checked`	Preselects option for select lists	`"checked"`	`<input />`
`Disabled`	Disables form	`"disabled"`	`<button>` `<input>` `<opt-group>` `<option>` `<select>` `<tex-tarea>`
`enctype`	Specifies encoding method for form-input data	Media type	`<form>`
`for`	Points to ID reference from other attributes	Idref	`<label>`
`label`	Identifies a group of options in a form	Text	`<opt-group>`
`label`	Specifies an option name in a form	Text	`<option>`
`method`	HTTP method to use when submitting a form	`{"get"|"put"}`	`<form>`
`multiple`	Permits selection of multiple options in a form	`"multiple"`	`<select>`

Name	*Function/Value Equals*	*Value Types*	*Related Element(s)*
`name`	Names a specific form control	CDATA	`<button>` `<tex-tarea>`
`name`	Names a specific form-input field	CDATA	`<select>`
`name`	Names a form for script access	CDATA	`<form>`
`readonly`	Blocks editing of text fields within a form	`"readonly"`	`<input />` `<textarea`
`size`	Specifies number of lines of text to display for a drop-down menu	Number	`<select>`
`tabindex`	Defines tabbing order for form fields	Number	`<a><area />` `<button>` `<input />` `<object>` `<select>` `<tex-tarea>`
`type`	Defines button function in a form	`{"button"|"reset"|"submit"}`	`<button>`
`type`	Specifies type of input required for form-input field	`{"button"|"checkbox"|"file"|"hidden"|"image"|"password"|"radio"|"reset"|"submit"|"text"}`	`<input />`
`value`	Supplies a value to send to the server when clicked	CDATA	`<button>`
`value`	Associates values with radio buttons and check boxes	CDATA	`<input />`

Chapter 15

Fun with Client-Side Scripts

In This Chapter

- Defining DHTML
- Dealing with image and text rollovers
- Adding dynamic content
- Showing pop-up windows
- Using Web cookies

If you're the outdoor type, you can get an adrenaline rush by climbing a mountain, mountain biking, or perhaps inventing a new sport, such as parafishing or sewer snorkeling. If you are reading this book, chances are you're sitting in front of your computer trying to create a Web site. If so, we have another idea for the ultimate Web adrenaline rush: Dynamic HTML!

Dynamic HTML, also known as *DHTML,* is techie talk for a useful and powerful set of technologies. It's the combination of HTML, Cascading Style Sheets (CSS), the Document Object Model (the DOM), and JavaScript. If you use these four technologies together, you're creating DHTML.

DHTML is like a printed document in which the DOM acts as the nouns, JavaScript as the verbs, CSS as the adjectives, and HTML as the paper itself. The individual parts are useful, but it's in combination that they become truly powerful. If you can put them all together, you can speak DHTML.

In this chapter, we explore how to use DHTML and its component technologies to bring active content to your Web pages. Specifically, we explore how to create rollovers, add dynamic content to your page, display pop-up windows, and tap in to the power of cookies.

Adding Rollovers to Your Pages

If you are new to HTML, a rollover probably sounds like a pet trick. But, in actuality, a rollover is perhaps the most common use of DHTML on the Web. It's an instruction that brings your Web page to life when a mouse pointer hovers over an image or text.

Image rollovers with JavaScript

If all JavaScript scripts went to school, the image rollover would certainly be the BMOC, the Big Muckety-muck on Campus. It's definitely the most popular use for JavaScript. Without image rollovers, your image buttons look dull and drab; visitors to your site might even assume that your buttons aren't actually live links if those buttons don't change in some fashion when a cursor moves over them. With image rollovers, however, your pages let loose a dash of adrenaline with each mouse-pointer hover.

TIP

We used the terrific (and free!) button generator at `www.tomaweb.com/buttons.asp` to make these and other buttons you'll see in this chapter — and others to follow. You might want to try it, too!

Consider the two states of the image rollover you see in Figure 15-1:

- The left side of Figure 15-1 shows a button in its inactive *(off)* state.
- The right side of Figure 15-1 shows the same button when the cursor is moved over it. That's the active *(on)* state.

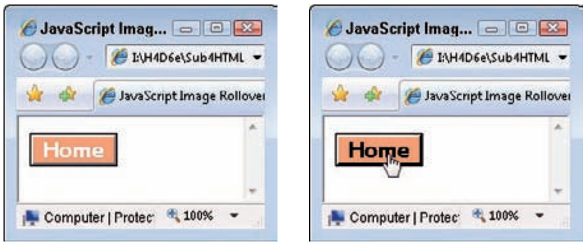


Figure 15-1: A simple button by itself (left) and with the mouse pointer hovering (right).

Listing 15-1 shows the code for a JavaScript image rollover.

Listing 15-1: JavaScript Image Rollover

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>JavaScript Image Rollover</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <script type="text/javascript" language="javascript">
        function chgImg(imgField,imgState) {
            document[imgField].src = "images/" + imgField + imgState + ".gif"
        }
    </script>
</head>
<body>
    <a href="index.html" onmouseover="chgImg('homeButton','On')"
        onmouseout="chgImg('homeButton','Off')">
        <img src="images/homeButtonOff.gif" width="65" height="15"
          border="0" alt="Go Home" name="homeButton" /></a>
</body>
</html>
```

The `images` subdirectory holds two separate image files with identical dimensions:

- `homeButtonOff.gif` appears when the button is in the *off* state (when the page loads).
- `homeButtonOn.gif` appears when the button is in the *on* state (when the cursor is over the button).

If you want to add JavaScript rollovers to your existing Web page, follow these steps:

1. **Decide on an attribute name for the rollover button.**

   You want to give each button a unique name. For example, if you have a button for an About page, you might call it `aboutMeButton`. Call a button linked to your Home page `homeButton`.

2. **Create your button images in your favorite image-editing application.**

   You need two identical-size images *for each button*.

3. **Name the `On` and `Off` button image files with the attribute.**

   For example, the `aboutMeButton` button needs two image files:

   ```
   aboutMeButtonOn.gif
   aboutMeButtonOff.gif
   ```

4. **Put button image files into an images subdirectory in the directory for the page that contains the rollovers (or in the same directory, as in our preceding example).**

5. **Add the JavaScript code in Listing 15-1 to your page.**

   That's everything between (and including) the `<script>` and `</script>` tags. It goes inside the `<head>` tags at the top.

6. **Add the *off* versions of each image to your page.**
7. **Add the `name` attribute to each `<img>` tag on your page.**
8. **Surround each `<img>` tag with an `<a href> </a>` tag.**
9. **Add these event handlers to each `<a href>` tag:**

   Add the following attributes to use `homeButton`:

   ```
   onmouseover="chgImg('homeButton','On')"
   onmouseout="chgImg('homeButton','Off')"
   ```

   Next, add these attributes to use `aboutMeButton`:

   ```
   onmouseover="chgImg('aboutMeButton','On')"
   onmouseout="chgImg('aboutMeButton','Off')"
   ```

With this image-rollover script, note the following behavior:

- For dialup visitors to your Web site, this rollover script takes a moment to download the active image file the first time the visitor hovers the mouse pointer over the image.

  TIP

  However, you can *preload* the active states of your images; that is, tell JavaScript to load all the *on* versions of your buttons when the page initially loads. This technique enables your page to instantaneously swap images when they're rolled over. Listing 15-2 shows the added code to preload the images.

  WARNING!

  Recent versions of Microsoft Internet Explorer (5.*x* and newer) contain a bug that can make preloading ineffective. Under some circumstances, the browser ignores any images already downloaded into your local cache — and instead requests the image anew each time a visitor moves the mouse pointer over the rollover image. If you encounter this problem, consider a text rollover instead (discussed in the "Text rollovers with CSS" section later in this chapter).

- This script depends on giving buttons specific names; if you want more flexibility, your script must handle it.
- This script causes trouble with certain ancient browsers, particularly Netscape versions 1 and 2 and Internet Explorer versions 1, 2, and 3.

Listing 15-2: Enhanced JavaScript Image Rollover with Preloader

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
    <title>JavaScript Image Rollover</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <script type="text/javascript" language="javascript">

        //Preloads images
        if (document.images) {
          homeButtonOn = new Image
          homeButtonOff = new Image
          homeButtonOn.src = "homeButtonOn.gif"
          homeButtonOff.src = "homeButtonOff.gif"
        }

        function chgImg(imgField,imgState) {
            document[imgField].src = imgField + imgState + ".gif"
        }
    </script>
</head>
<body>
    <a href="index.html" onmouseover="chgImg('homeButton','On')"
          onmouseout="chgImg('homeButton','Off')">
          <img src="homeButtonOff.gif" width="80" height="30"
             border="0" alt="Go Home" name="homeButton" /></a>
</body>
</html>
```

Text rollovers with CSS

For years, the only option available for creating a rollover was to create button images, then "activate" them with JavaScript (as discussed in the preceding "Image rollovers with JavaScript" section). However, now that CSS has gained acceptance in newer browser versions, there's an alternative way to create rollovers without using images at all.

Text rollovers have advantages and disadvantages when compared to JavaScript image rollovers:

- **Good news:** Text is faster and more meaningful to search engines, and it's always easier to just add text to a page than it is to create two images and then add them to a page, as with an image rollover. Plus, you don't need to worry about preloading images.
- **Bad news:** Although you can control the text font, style, and border, you can't do all the nifty visual tricks that you can with images, such as anti-aliasing. In addition, this method works only in reasonably current browsers. (If your target viewing audience uses a browser released in this century, you should be fine.)

Figure 15-2 shows a plain-Jane Web page with two rollover text links: *Home* and *About Me*. Moving the cursor over one of the images, as shown in Figure 15-3, causes the rolled-over version of the text to display. Listing 15-3 displays the HTML and CSS required for this rollover effect.

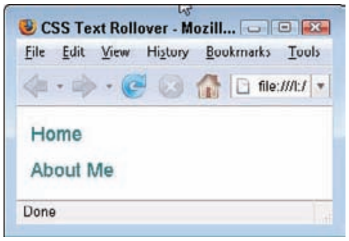

CSS Text Rollover - Mozill...
File Edit View History Bookmarks Tools
file:///I:/
Home
About Me
Done

Figure 15-2: A page with text rollovers handled with CSS.

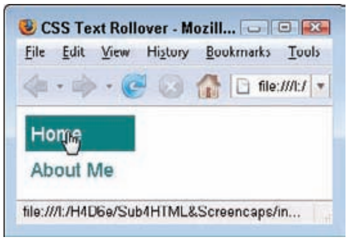

CSS Text Rollover - Mozill...
File Edit View History Bookmarks Tools
file:///I:/
Home
About Me
file:///I:/H4D6e/Sub4HTML&Screencaps/in...

Figure 15-3: Moving the cursor over the link text changes the text and background colors.

The page can still show up on-screen whether you've visited the linked page or not. Figure 15-4 shows how the page appears after you've been to this site's home page. And although that image is grayed out, it's still a link, so rolling over it still produces the same effect as in Figure 15-3.

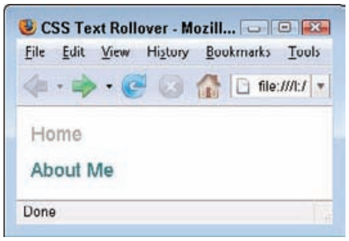

CSS Text Rollover - Mozill...
File Edit View History Bookmarks Tools
file:///I:/
Home
About Me
Done

Figure 15-4: After you've been to a page, the link text color shows that the page was visited.

Listing 15-3: A Text Rollover with CSS

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>CSS Text Rollover</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <style type="text/css">
        h4 {font: 18px geneva, sans-serif; margin: 0; color: #000;
             background: #FFF;}
        a {text-decoration: none;}
        div#navbar {width: 100px;}
        div#navbar a {display:block; margin: 0; padding: 0.3em;}
        div#navbar a:link {color: #008080; background-color: transparent;}
        div#navbar a:visited {color: #C0C0C0; background-color: transparent;}
        div#navbar a:hover {background: #FFFFFF; color: #008080;}
    </style>
</head>
<body>
<div id="navbar">
    <h4><a href="index.html">Home</a></h4>
    <h4><a href="aboutMe.html">About Me</a></h4>
</div>
</body>
</html>
```

TECHNICAL STUFF

In this example, we change the text from teal-on-white to white-on-teal when the cursor hovers over the link; that way it's easy for you to see what's going on in these screenshots. You may want your site to use a more colorful approach (or perhaps a different color scheme). It goes gray after being visited.

Adding this type of navigation to your site couldn't be simpler:

1. **Within the** `<head>` **tags, add the preceding code (from Listing 15-3) inside and including the** `<style>` **and** `</style>` **tags.**
2. **Add links inside individual** `<h4>` **tags.**
3. **Make sure that the entire menu is inside a** `<div>` **tag with an** `id` **attribute of** `navbar`**.**

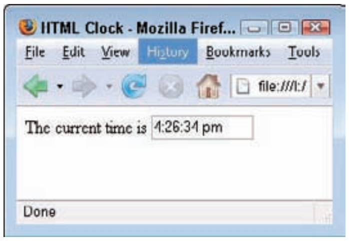
If you add the CSS to your site via a link to a site-wide external style sheet (see Chapters 8 and 9 for more information on style sheets), you can add, change, or delete menu-bar links on your site at any time, without having to touch a single line of CSS or JavaScript. You simply add or modify your `<a href>` tags. Slick, huh?

Displaying Dynamic Content on Your Page

Web pages can take advantage of JavaScript to change by themselves without requiring user input or updates from a Web server. To demonstrate how JavaScript does this, we create a simple clock that automatically updates itself every second. We first show you how to do this using JavaScript and HTML, and then how to do it using JavaScript and the DOM.

HTML and JavaScript

You can create a JavaScript-enabled clock by using JavaScript and an ordinary HTML `<input>` tag. Listing 15-4 displays the code that you need to make this happen, and Figure 15-5 displays the results on-screen.


HTML Clock - Mozilla Firef...
File Edit View History Bookmarks Tools
file:///I:/
Done

The current time is 4:26:34 pm

Figure 15-5: This page displays the current time, updated every second, inside a text field.

Listing 15-4: A Simple HTML and JavaScript Clock

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>HTML Clock</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <script type="text/javascript" language="javascript">
        window.onload = theClock

        function theClock() {
            now = new Date;

            theTime = ((now.getHours() > 0 && now.getHours()
                      < 13)) ? now.getHours() : (now.getHours() == 0)
                      ? 12 : now.getHours()-12;
            theTime += (now.getMinutes() > 9) ? ":" + now.getMinutes() : ":0"
                    + now.getMinutes();
            theTime += (now.getSeconds() > 9) ? ":" + now.getSeconds() : ":0"
                    + now.getSeconds();
            theTime += (now.getHours() < 12) ? " am" : " pm";

            document.myForm.myClock.value = theTime;
            setTimeout("theClock()",1000);
        }
    </script>
</head>
<body>
<form action="#" name="myForm">
The current time is
<input type="text" name="myClock" readonly="readonly" size="11" />
</form>
</body>
</html>
```

In Listing 15-4, the clock is updated inside a form text field so that JavaScript can write out to the page without having to reload the entire page every second. Wherever the text field is on your page, that's where the time appears. The clock shows the time set on the user's computer, not the time on the Web server that's serving the pages.

To add this clock to your page, just follow these steps:

1. **Copy everything from the beginning** `<script>` **tag to the ending** `</script>` **tag in Listing 15-4.**

   The complete code listings for this book are available at the Web site associated with this book.

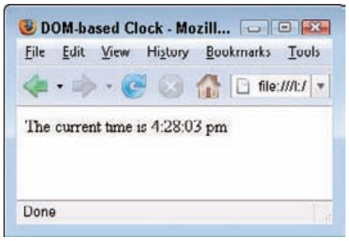
2. **Paste the code into the** `<head>` **section of your page.**

3. **Add the** `<form>` **and** `<input>` **tags (including the** `name` **attribute on each) on your page where you want your clock to appear.**

The very first thing that JavaScript does when the Web page loads is set the window's `onload` event handler to trigger the `theClock` function. This is no big deal — unless you want to run another script when the page loads. However, this script is written in such a way that it never comes back to run anything else, since the clock is constantly updating itself.

JavaScript and DOM

When you add both JavaScript and some DOM manipulation to your page, you can update the text of the page itself, as shown in Listing 15-5. Figure 15-6 shows a clock that updates every second, but the clock text looks just like the other text on the line.


DOM-based Clock - Mozill...
File Edit View History Bookmarks Tools
file:///I:/
Done

The current time is 4:28:03 pm

Figure 15-6: This page displays the current time, updated every second, as simple text.

Listing 15-5: A Slightly More Complex JavaScript and DOM Clock

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>DOM-based Clock</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <script type="text/javascript" language="javascript1.5">
      window.onload = theClock

      function theClock() {
          now = new Date;

          theTime = ((now.getHours() > 0 && now.getHours() < 13)) ?
            now.getHours() : (now.getHours() == 0) ? 12 : now.getHours()-12;
          theTime += (now.getMinutes() > 9) ? ":" + now.getMinutes() : ":0"
            + now.getMinutes();
          theTime += (now.getSeconds() > 9) ? ":" + now.getSeconds() : ":0"
            + now.getSeconds();
          theTime += (now.getHours() < 12) ? " am" : " pm";

          clockSpan = document.getElementById("myClock");
          clockSpan.replaceChild(document.createTextNode(theTime),
             clockSpan.firstChild);

          setTimeout("theClock()",1000);
      }
    </script>
</head>
<body>
The current time is <span id="myClock">?</span>
</body>
</html>
```

The script in Listing 15-5 is virtually identical to Listing 15-4, except for a couple of different lines of JavaScript. Using the DOM, the script can grab that text and replace it with new text every second.

- **The good news:** You can style that text with CSS and make it appear just like everything else on the page. The look is far superior to what you get if you put the dynamic text inside an `<input>` tag.
- **The bad news:** Older browsers don't support the `<span>` tag, so if your visitors use legacy versions of Netscape or Microsoft browsers, consider using the HTML and JavaScript version instead.

Other examples in this book show the initial `<script>` tag with the `language` attribute set to `javascript`. This particular script specifies `javascript1.5`, which tells the browser to ignore everything that's going on if you aren't using a modern browser. If you come into this page with an older browser, you won't get an error, but you won't get the dynamic effects, either.

To add the DOM-enabled scripted clock to your page, follow these steps:

1. **Add everything between the beginning and ending** `<script>` **tags to the** `<head>` **section of your page.**
2. **Add a** `<span>` **tag with an** `id` **attribute of** `myClock` **anywhere on your page.**

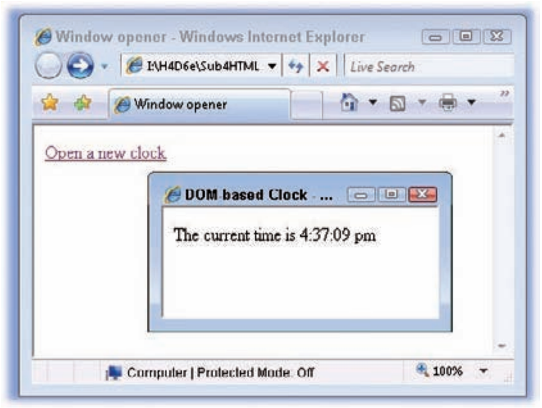
   The clock appears!

Are you getting errors when you try to add the DOM-powered clock to your page? Some browsers have a problem with either nothing or a space in the `<span>` tag. Solution: As with the example shown in Listing 15-5, put something (anything) inside the `<span>` tag for when it's initially loaded. In this case, there's a question mark, but it won't ever be visible to Web page visitors.

Displaying Pop-up Windows

Pop-up windows can be useful, but they're also one of the most abused tools on the Web. Having a way to provide some extra information to site visitors without making them leave your page is useful. Unfortunately, so many unethical people have given pop-ups (in particular, advertising pop-ups) a bad name that many Web surfers install pop-up blockers. Consequently, if you add pop-up windows to your site, make sure that they aren't the only way visitors can access vital information.

Figure 15-7 shows a simple pop-up window containing the clock from Listing 15-5. This little window is a nice, floating, constantly updated clock that can stay up even after you've left the calling page. Listing 15-6 shows how to create this pop-up window, which is a new browser window with no address bar, menu bar, scroll bars, status bar, or toolbars, as shown in Figure 15-7.



The current time is 4:37:09 pm

Figure 15-7: Clicking the link opens a new browser window.

Listing 15-6: Opening a New Browser Window

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Window opener</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <script type="text/javascript" language="javascript">
        function OpenWindow (newPage) {
            window.open(newPage,"newWin","width=200,height=50,resizable=yes");
        }
    </script>
</head>
<body>
<a href="domClock.html"
   onclick="OpenWindow(this.href);return false">Open a new clock</a>
</body>
</html>
```

The “Open a new clock” link, when clicked, calls a tiny JavaScript function that opens a new window that’s 200 pixels wide, 50 pixels high, and resizable. You also need to rename the code from Listing 15-5 to `domClock.html` and save it into the same directory where Listing 15-6 lives.

Follow these steps to add this new window to your own site:

1. **Add everything from the beginning to the ending `<script>` tags to the `<head>` of your page.**
2. **In the body, figure out where you want the link to be.**
3. **Add the `onclick` event handler attribute to the `<a href>` tag around the text or image.**

You can have multiple links on the same page that each open a new window, and they can all have identical `onclick` handlers and call the same JavaScript function.

The script is coded so that all the different bars — address bars, menu bars, mini-bars, you name it — are turned off. You can change the code so the window sizes are different (or various fields either are or aren’t displayed) by varying the contents of the last field in the `window.open` function. Table 15-1 shows the valid entries for this parameter: Just put them all, separated by commas (but not spaces), into a single string, and you get exactly the results you want.

The default for every JavaScript Window parameter is `no`, so there’s no difference between setting an entry to `no` and just leaving it off entirely.

Table 15-1 JavaScript’s Window Parameters

Name	*Values (Default in Italics)*	*Description and Value*
`location`	`yes`, *`no`*	Should the new window display the location bar (also known as the address bar)?
`menubar`	`yes`, *`no`*	Should the new window display the menu bars? (Applies only to Windows and Unix.)
`resiz-able`	`yes`, *`no`*	Should the user be allowed to resize the new window?
`scroll-bars`	`yes`, *`no`*	Should the user be allowed to scroll the new window?
`status`	`yes`, *`no`*	Should the new window display the status bar?
`toolbar`	`yes`, *`no`*	Should the new window display the toolbar?

(continued)

Table 15-1 *(continued)*

Name	*Values (Default in Italics)*	*Description and Value*
`height`	Numeric	The height of the new window in pixels
`width`	Numeric	The width of the new window in pixels
`top`	Numeric	The top position of the new window in pixels, relative to the top edge of the browser window
`left`	Numeric	The left position of the new window in pixels, relative to the left edge of the browser window

Working with Cookies

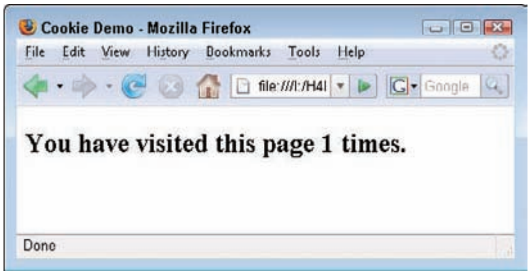
Every time we start talking about cookies, we are tempted to grab a glass of milk and get ready for dipping. But then we remind ourselves that Web cookies, as useful as they can be, actually taste pretty bland. (We imagine they'd taste far more like chicken than the famous Toll House recipe.) Although they may not be tasty, you might find cookies to be helpful as you create your Web site.

A cookie lets you store information on visitors' computers that you can revisit later. Cookies offer a powerful way to maintain "state" within your Web pages.

The code in Listing 15-7 reads and writes two cookies when a visitor loads the page:

- `pageHit` contains a count of the number of times the visitor has loaded the page.
- `pageVisit` contains the last date and time the visitor visited.

Figure 15-8 shows how the page appears on the initial visit, and Figure 15-9 shows how it looks on subsequent visits.

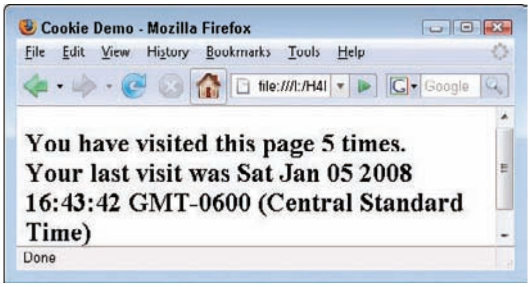


Cookie Demo - Mozilla Firefox

You have visited this page 1 times.

Done

Figure 15-8: This cookie knows you've never been to this page before.



Cookie Demo - Mozilla Firefox

You have visited this page 5 times.
Your last visit was Sat Jan 05 2008 16:43:42 GMT-0600 (Central Standard Time)

Done

Figure 15-9: These cookies know not only that you've been here before, but when.

Listing 15-7: Cookie-Handling Script

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Cookie Demo</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <script type="text/javascript" language="javascript">
    now = new Date
    expireDate = new Date
    expireDate.setMonth(expireDate.getMonth()+6)

    hitCt = parseInt(cookieVal("pageHit"))
    hitCt++
    lastVisit = cookieVal("pageVisit")
    if (lastVisit == 0) {
       lastVisit = ""
    }

    document.cookie = "pageHit="+hitCt+";expires=" + expireDate.toGMTString()
    document.cookie = "pageVisit="+now+";expires=" + expireDate.toGMTString()
```

(continued)

Listing 15-7 *(continued)*

```
    function cookieVal(cookieName) {
        thisCookie = document.cookie.split("; ")
         for (i=0; i<thisCookie.length; i++) {
              if (cookieName == thisCookie[i].split("=")[0]) {
                  return thisCookie[i].split("=")[1]
             }
        }
        return 0
    }
    </script>
</head>
<body>
<h2>
    <script type="text/javascript" language="javascript">
    document.write("You have visited this page " + hitCt + " times.")
    if (lastVisit != "") {
        document.write("<br />Your last visit was " + lastVisit)
    }
    </script>
</h2>
</body>
</html>
```

Unlike preceding examples, Listing 15-7 has a `<script>` section in both the head and the body:

- Cookies are read and written in the header script when the page loads.
- The body script dynamically writes out the contents of the page itself.

Follow these steps to add the cookie-handling script to your page:

1. **Copy both** `<script>` **sections and put them into the appropriate parts of your page.**
2. **Change the** `<body>` **section to contain the text that you want the page to display.**

   The lines inside the `document.write()` statements write the text out to the document on the fly.

TRICKS OF THE TRADE

A cookie has an expiration date, after which it's no longer available. This example creates cookies that expire in six months. If you want your cookies to live longer (or not so long), adjust the JavaScript code near the top that sets a value for `expireDate`. Thus, the following example increases the current expiration date by six months:

```
expireDate.setMonth(expireDate.getMonth()+6)
```

Using the XHTML Object Element

You can use the (X)HTML `<object>` element to embed content inside a Web page. It provides a general mechanism to embed content of all kinds, from another text file to numerous types of active content, such as programs written in languages other than JavaScript and multimedia (Flash animations, for example). This is advanced stuff for Web-page builders, so we list only object-related (X)HTML attributes in Table 15-2 and then conclude with pointers to details on using this element.

If you want to use programming languages (such as Perl, Python, Java, and so forth) or various types of multimedia in your Web pages, you should cozy up to the `<object>` element.

Table 15-2 Object-related (X)HTML Attributes

Name	*Function/Value Equals*	*Value Types*	*Related Element(s)*
`archive`	Identifies location (URI) for archive file	URI	`<object>`
`classid`	Identifies object implementation URI	URI	`<object>`
`codebase`	Identifies base URI for `classid`, `data`, and `archive`	URI	`<object>`
`codetype`	Identifies content type for code	Media type	`<object>`
`data`	Identifies object data by location	URI	`<object>`
`standby`	Specifies message that appears on-screen while object is loading	Text	`<object>`
`type`	Identifies content type for object data	Media type	`<object>`

The following resources address the (X)HTML `<object>` tag nicely:

- W3Schools offers (X)HTML tag information online; `<object>` coverage includes links to a complete tag list at `www.w3schools.com/tags/tag_object.asp`.
- Juicy Studio offers a detailed discussion of the `<object>` element at www `http://juicystudio.com/article/object-paranoia.php`.
- In the HTML 4 Recommendation, the W3C includes “Objects, Images, and Applets” at `www.w3.org/TR/REC-html40/struct/objects.html`.

Part V
(X)HTML Projects

The 5th Wave
By Rich Tennant
© RICHTENNANT
Cleopatra
Carpet Cleaners
"So far our Web presence has been pretty good. We've gotten some orders, a few inquiries, and nine guys who want to date our logo."

In this part . . .

In this part of the book, you can explore, understand, and see some typical Web page projects, including all the markup and underlying scripts, graphics, and other materials that go into their makeup. They're ready-to-use examples that you can edit or customize for your own needs and circumstances; these projects are designed to function as templates of a sort that you can adapt and use as your own Web pages.

Here, you find typical implementations of a personal Web page, an eBay auction page, a basic company Web site, and a product catalog page that incorporates an honest-to-gosh shopping cart application. This is where everything from Parts II, III, and IV is put to work in useful, snazzy Web pages you can tailor for your own purposes.

Chapter 16

The About Me Page

In This Chapter

- Deciding what your page needs to contain
- Analyzing your audience
- Building the page
- Using templates

It's time to build your very own page, one that tells the world who you are and what you're like.

You get only one chance to make a good first impression, and your About Me Page is how people all over the world can get to know who you are. So put your best foot forward and make your home page reflect exactly the kind of image you want to project.

Overview and Design Considerations

Every Web site should start with a plan of its goals and intended audience, and a simple home page is no exception. Think about

- Who will visit the site
- What you want them to get out of it
- How often you want to update the site
- Content to include

Audience analysis

Your site will be your public face to the world, so analyze who you think will be visiting — and what you want them to get out of a visit. For instance, a site that amuses and entertains your friends might be exactly what you *don't* want a prospective employer to see!

WARNING!

Even if you don't give people the URL of your site, they're still likely to find it through search engines. *Googling*—searching the Web, particularly via Google.com, for information about—prospective employees and dates is more common than not. Do you really want your parents and siblings, or your co-workers, to read all the details about what you did last weekend?

Component elements

At its simplest, any Web page consists of nothing but a single HTML text file. If you're a great designer, that's sufficient. Chances are, though, that you'll want to add an image or two to give the site some visual interest. The two examples of home pages in this chapter each use three files: the home page and two images.

Page Markup

After you have a template for a Web page (see the "*HTML, XHTML and CSS For Dummies*" sidebar), just fill in the blanks. Include your content in new *headings* (`<h2>` tags in the following examples) and new *paragraphs* (with their corresponding `<p>` tags).

Your home page

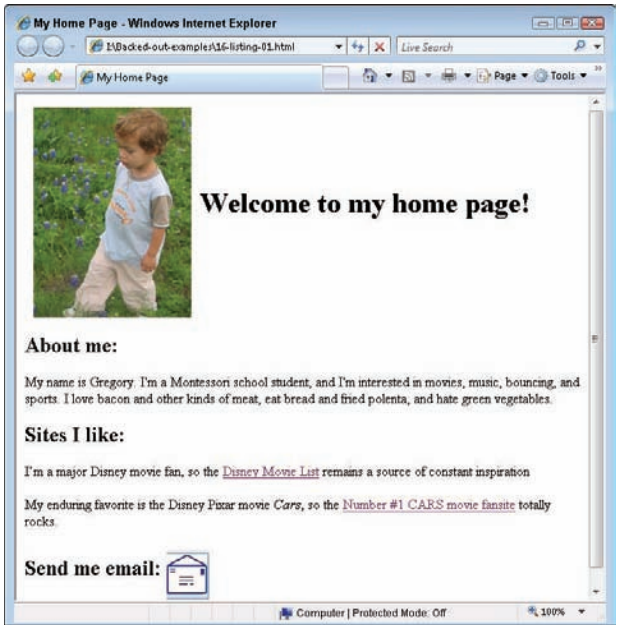
Listing 16-1 is the HTML code for a typical home page. Figure 16-1 shows how it looks when displayed in a browser.

Listing 16-1: A Home Page

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>My Home Page</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
```

```
    <h1><img src="gregory-bbnt.jpg" alt="Gregory in the bluebonnets" width="180"
         height="240" align="middle" hspace="10" />Welcome to my home page!</h1>
    <h2>About me:</h2>
    <p>My name is Gregory. I'm a Montessori school student, and I'm interested
       in movies, music, bouncing, and sports. I love bacon and other kinds of
       meat, eat bread and fried polenta, and hate green vegetables.</p>
    <h2>Sites I like:</h2>
    <p>I'm a major Disney movie fan, so the
    <a href="http://www.disneymovieslist.com/">
    Disney Movie List</a> remains a source of constant inspiration</p>
    <p>My enduring favorite is the Disney Pixar movie <i>Cars</i>, so the
    <a href="http://cars.starsfansite.com/">Number #1 CARS movie fansite</a>
    totally rocks.</p>
    <h2>Send me email: <a href="mailto:gregory@edtittel.com">
        <img src="email.jpg" alt="email image" width="48" height="66"
        align="middle" border="0" /></a></h2>
</body></html>
```

This page is about as simple as it gets: There's no style information, no JavaScript, only two images, and not a lot of text. But it's enough to give you an idea of what kind of person put up the site and what he's like.


My Home Page - Windows Internet Explorer
I:\Backed-out-examples\16-listing-01.html
Live Search
My Home Page
Page
Tools
Computer | Protected Mode: Off
100%

Welcome to my home page!

About me:

My name is Gregory. I'm a Montessori school student, and I'm interested in movies, music, bouncing, and sports. I love bacon and other kinds of meat, eat bread and fried polenta, and hate green vegetables.

Sites I like:

I'm a major Disney movie fan, so the Disney Movie List remains a source of constant inspiration

My enduring favorite is the Disney Pixar movie *Cars*, so the Number #1 CARS movie fansite totally rocks.

Send me email:

Figure 16-1: A simple home page.

This page contains two small graphics:

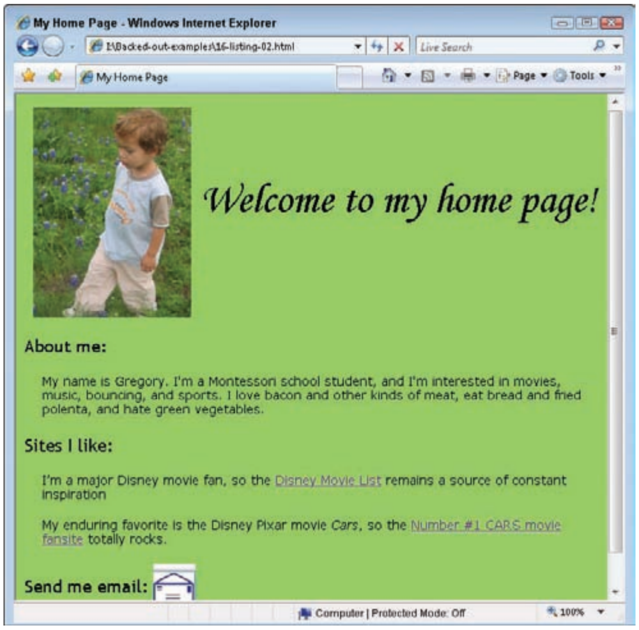
- `gregory-bbnt.jpg` is a favorite photo that shows the site subject in a field of Texas bluebonnets.
- `email.jpg` is a button that visitors can click. When a visitor clicks it, his or her e-mail client pops open in a window with a preaddressed e-mail.

It isn't hard to go from a super-simple site to a site that's considerably more attractive without getting horribly complex. Listing 16-2 and Figure 16-2 show a site with a lot more style and only a little more complexity.

Listing 16-2: Another Home Page

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>My Home Page</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <style type="text/css">
        body {color: #000; background-color: #9C6;}
        h1 {font: 48px "monotype corsiva", fantasy;}
        h2 {margin-top: 20px; font: 20px "trebuchet ms", verdana,
            arial, helvetica, geneva, sans-serif;}
        p {margin-left: 20px; font: 14px/16px verdana, geneva, arial,
           helvetica, sans-serif}
    </style>
</head>
<body>
    <h1><img src="gregory-bbnt.jpg" alt="Gregory amidst bluebonnets" width="180"
               height="240"
         align="middle" hspace="10" />Welcome to my home page!</h1>
    <h2>About me:</h2>
    <p>My name is Gregory. I'm a Montessori school student, and I'm interested
       in movies, music, bouncing, and sports. I love bacon and other kinds of
       meat, eat bread and fried polenta, and hate green vegetables.</p>
    <h2>Sites I like:</h2>
    <p>I'm a major Disney movie fan, so the
    <a href="http://www.disneymovieslist.com/">
    Disney Movie List</a> remains a source of constant inspiration</p>
    <p>My enduring favorite is the Disney Pixar movie <i>Cars</i>, so the
    <a href="http://cars.starsfansite.com/">Number #1 CARS movie fansite</a>
    totally rocks.</p>
    <h2>Send me email: <a href="mailto:gregory@edtittel.com">
        <img src="email.jpg" alt="email image" width="48" height="66"
        align="middle" border="0" /></a></h2>
</body>
</html>
```

Text and tags within the `<body>` element are about the same as inside the first example, but the result is different because of the style rules in the `<head>`.


My Home Page - Windows Internet Explorer
I:\Backed-out-examples\16-listing-02.html
Live Search
My Home Page
Page
Tools
Computer | Protected Mode: Off
100%

Welcome to my home page!

About me:

My name is Gregory. I'm a Montessori school student, and I'm interested in movies, music, bouncing, and sports. I love bacon and other kinds of meat, eat bread and fried polenta, and hate green vegetables.

Sites I like:

I'm a major Disney movie fan, so the Disney Movie List remains a source of constant inspiration

My enduring favorite is the Disney Pixar movie *Cars*, so the Number #1 CARS movie fansite totally rocks.

Send me email:

Figure 16-2: Our less-simple-and-more-stylish home page.

The style rules set a background color for the page and specify the fonts to be used. Although the two pages share identical content, the latter page gives a stronger impression of its maker's personality.

Looking good

Adding cool fonts and bright colors to your page is a good way to add visual interest — but it makes your site look tacky if it's overdone.

Follow these tips for a colorful, professional-looking page:

- **Pick a graphic and use its colors elsewhere on the page.**

  The green at the back of the photo harmonizes nicely with the green background:

  - The background color in the photo blends with the background color for the page (if you use transparent gifs and pick the same background color, it will blend in seamlessly).
  - The color of the background also provides the color for the e-mail icon (we built it at `www.tomaweb.com`, using their free button gif builder).

- **Check your page on other computers to make sure your colors really look the way you want them to look.**

  Colors often appear differently on different monitors, and not everyone's monitor is set up correctly.

- **Be selective when choosing fonts and font colors.**

  - A font on your computer might not be on other computers.

    Provide alternate fonts as a backup in your style rules.
  - Don't use too many different fonts on one page or it'll end up looking like a ransom note.
  - Use font colors that contrast with your background so people can read what you've written.

Listing 16-3 is a bare-bones template with comments that tell you where to add your own content. Start with this, and where you end up is limited only by your imagination and creativity.

Listing 16-3: A Home Page Template

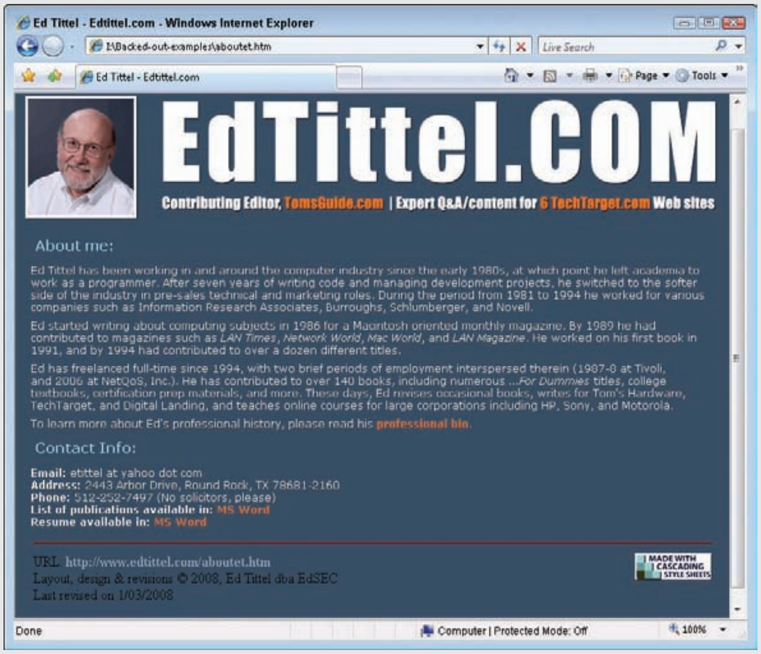
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>My Home Page</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <style type="text/css">
        body {color: #000000; background-color: #FFFFFF;}

        /* Add additional style rules here */

    </style>
</head>
<body>
    <h1>Welcome to my home page!</h1>

    <h2>About me:</h2>
    <!-- Add something here about you and your interests -->

    <h2>Sites I like:</h2>
    <!-- Add links here to sites that you like to visit -->

    <h2><a href="mailto:yourName@example.com">Send me email!</a></h2>
</body>
</html>
```

Using *HTML, XHTML and CSS For Dummies* page templates

Part V of this book (Chapters 16–19) contains page templates designed specifically so you can easily copy and modify them to suit your tastes. You can either

- Type them in yourself
- Download them from the Web site associated with this book.

If you're looking for more templates, you can find great ones for copying, pasting, and adding your content in most WYSIWYG HTML editors. Chapter 22 covers these editors. You can also find a more advanced About Me page that Ed Tittel built for his recently reworked Web site, that includes a custom graphic header, a more professional look and content, with links to a résumé, detailed professional bio, and a list of publications — entirely suitable for a professional first impression (a very best foot forward)! Visit it online at `www.edtittel.com` (where you can use View Source in your browser to see the HTML used therein). A different version of that page is depicted below, which we constructed specifically for this book. (Want to peek at the page's source code? Again, You'll find it at the Web site associated with this book.


Ed Tittel - Edtittel.com - Windows Internet Explorer
I:\Backed-out-examples\aboutet.htm
Live Search
Ed Tittel - Edtittel.com
Page
Tools
EdTittel.COM
Contributing Editor, TomsGuide.com | Expert Q&A/content for 6 TechTarget.com Web sites
MADE WITH CASCADING STYLE SHEETS
Done
Computer | Protected Mode: Off
100%

About me:

Ed Tittel has been working in and around the computer industry since the early 1980s, at which point he left academia to work as a programmer. After seven years of writing code and managing development projects, he switched to the softer side of the industry in pre-sales technical and marketing roles. During the period from 1981 to 1994 he worked for various companies such as Information Research Associates, Burroughs, Schlumberger, and Novell.

Ed started writing about computing subjects in 1986 for a Macintosh oriented monthly magazine. By 1989 he had contributed to magazines such as *LAN Times*, *Network World*, *Mac World*, and *LAN Magazine*. He worked on his first book in 1991, and by 1994 had contributed to over a dozen different titles.

Ed has freelanced full-time since 1994, with two brief periods of employment interspersed therein (1987-8 at Tivoli, and 2006 at NetQoS, Inc.). He has contributed to over 140 books, including numerous ...*For Dummies* titles, college textbooks, certification prep materials, and more. These days, Ed revises occasional books, writes for Tom's Hardware, TechTarget, and Digital Landing, and teaches online courses for large corporations including HP, Sony, and Motorola.

To learn more about Ed's professional history, please read his **professional bio**.

Contact Info:

Email: ebttel at yahoo dot com
Address: 2443 Arbor Drive, Round Rock, TX 78681-2160
Phone: 512-252-7497 (No solicitors, please)
List of publications available in: **MS Word**
Resume available in: **MS Word**

URL: http://www.edtittel.com/aboutet.htm
Layout, design & revisions © 2008, Ed Tittel dba EdSEC
Last revised on 1/03/2008

Chapter 17

The eBay Auction Page

In This Chapter

- Designing an online auction page
- Better ways to sell your auction items
- Template markup

Whether you're trying to buy or sell a car, a rare CD, or a bologna sandwich that bears an uncanny likeness to Calvin Coolidge, eBay is the 21st century's answer to a street market. In fact, with more than 100 million active participants, eBay is sometimes called "the world's garage sale." However, if these staggering numbers leave you starry-eyed over anticipated profits, be careful: With millions of items for sale, it's easy for your "Coolidge-lookalike bologna sandwich" to get lost in such a huge and teeming crowd.

Given this reality, the more attractively your auction item is presented and the better the description is written, the greater the chances your item will sell — and at a higher price.

In this chapter, we show you how to use HTML and CSS effectively to make your eBay auction look wicked cool. You'll discover how to

- Highlight parts of your description.
- Add pictures.

Although eBay is the biggest auction site online, you can use HTML in item descriptions on most other online auction sites as well, including Yahoo! Auctions, uBid, and others.

Designing Your Auction Page

Online auction sites let you include a few specific elements in your auction item page, such as

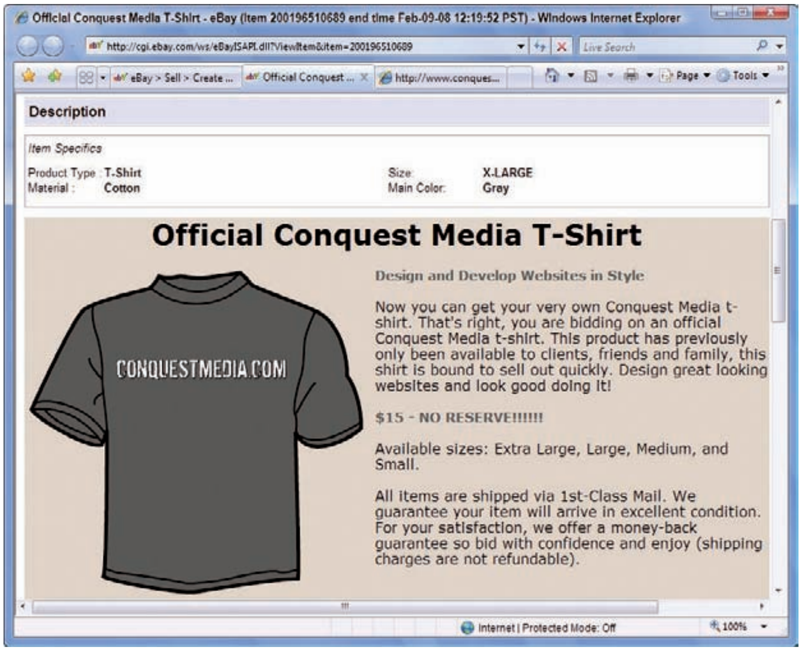
TIP

- **Title** (and sometimes a subtitle)
- **Description**

  This chapter focuses on the item description because that's where HTML markup can enhance the look of the description and add pictures.
- **Pictures**

Figure 17-1 shows an example of an auction description from eBay that uses HTML to add style and an embedded picture.

Auction sites typically allow you to use a series of online forms to list items for sale. If you want to use HTML and CSS in your item description, you include that markup in the text field that the auction site gives you for the description, as shown in Figure 17-2.


Official Conquest Media T-Shirt - eBay (Item 200196510689 end time Feb-09-08 12:19:52 PST) - Windows Internet Explorer
http://cgi.ebay.com/ws/eBayISAPI.dll?ViewItem&item=200196510689
Live Search
eBay > Sell > Create ...
Official Conquest ...
http://www.conques...
Page
Tools
CONQUESTMEDIA.COM
Internet | Protected Mode: Off
100%

Description

Item Specifics

Product Type :	T-Shirt	Size:	X-LARGE
Material :	Cotton	Main Color:	Gray

Official Conquest Media T-Shirt

Design and Develop Websites in Style

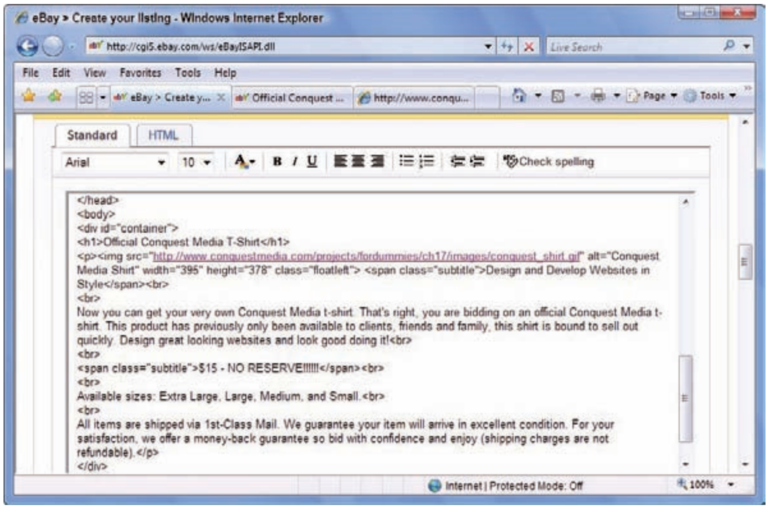
Now you can get your very own Conquest Media t-shirt. That's right, you are bidding on an official Conquest Media t-shirt. This product has previously only been available to clients, friends and family, this shirt is bound to sell out quickly. Design great looking websites and look good doing it!

$15 - NO RESERVE!!!!!!

Available sizes: Extra Large, Large, Medium, and Small.

All items are shipped via 1st-Class Mail. We guarantee your item will arrive in excellent condition. For your satisfaction, we offer a money-back guarantee so bid with confidence and enjoy (shipping charges are not refundable).

Figure 17-1: This auction description uses HTML to style the text and add a left-justified picture.


eBay > Create your listing - Windows Internet Explorer
http://cgi5.ebay.com/ws/eBayISAPI.dll
Live Search
File Edit View Favorites Tools Help
eBay > Create y...
Official Conquest ...
http://www.conqu...
Page
Tools
Standard
HTML
Arial
10
Check spelling
Internet | Protected Mode: Off
100%

```
</head>
<body>
<div id="container">
<h1>Official Conquest Media T-Shirt</h1>
<p><img src="http://www.conquestmedia.com/projects/fordummies/ch17/images/conquest_shirt.gif" alt="Conquest
Media Shirt" width="395" height="378" class="floatleft"> <span class="subtitle">Design and Develop Websites in
Style</span><br>
<br>
Now you can get your very own Conquest Media t-shirt. That's right, you are bidding on an official Conquest Media t-
shirt. This product has previously only been available to clients, friends and family, this shirt is bound to sell out
quickly. Design great looking websites and look good doing it!<br>
<br>
<span class="subtitle">$15 - NO RESERVE!!!!!!</span><br>
<br>
Available sizes: Extra Large, Large, Medium, and Small.<br>
<br>
All items are shipped via 1st-Class Mail. We guarantee your item will arrive in excellent condition. For your
satisfaction, we offer a money-back guarantee so bid with confidence and enjoy (shipping charges are not
refundable).</p>
</div>
```

Figure 17-2: Entering HTML in the online description form field at eBay.

eBay allows you to use HTML for style only in the item description; you can't use it in the title or subtitle lines.

Because the auction site itself creates much of what appears on an online auction page, you don't have to create the entire page of HTML markup. You just create markup for the item-description part of the page. That means

- You don't need to include the `<html>`, `<title>`, `<head>`, or `<body>` tags (remember, your markup will be part of a larger, complete HTML document).
- You can't include any scripts in the description.

When you create your auction-item page, be aware that all browsers are not created equal. A couple of limitations apply:

- If you use Microsoft Internet Explorer 6 or newer as your browser, eBay gives you an HTML editor that allows you to style text directly — and then turns it into HTML markup, as shown in Figure 17-2 (notice the HTML tab above the text-entry box; we're using the Windows Vista IE version 7.0.6000).
- Other browsers, including Mozilla Firefox and Safari on the Macintosh, now also support tabs for Standard (plain text) or HTML (markup) which you can use the same way.

When designing your item's title and description, don't immediately rush to add fancy HTML formatting. Instead, your first task is to create a description that effectively presents your product. Before you worry about the HTML markup, write a compelling title and description. Consider the following tips when you write your text:

- **Write a concise, descriptive title.** A good title includes words that clearly and specifically identify what you are selling. eBay's search engine uses these titles to help people find your item, and you won't sell what buyers can't find.
- **Look at completed listings to see successful descriptions.** Use those ideas to stimulate your own.

  REMEMBER However, don't plagiarize other people's descriptions (or rip off their pictures). That way lies trouble.
- **Spell words in the title and description correctly.** Misspelled words won't be found by visitors searching for your item.

  In fact, we've bought equipment worth thousands of dollars for a fraction of its market value, largely because it got no other bids — all because the sellers misspelled the items' names in the auction titles.
- **Be sure that you're listing the item in the proper auction category.** If you list it in the wrong category, your item will get lost, buried, obscured, masked . . . well, you get the idea.
- **Resist the temptation to use large fonts and lots of styles.** Buyers want to see your item description and photographs as quickly as possible. Keep your text and images direct, visually uncluttered, and to the point.
- **Use good photographs or illustrations.** Items with pictures or graphics sell much better:
  - The photo of your item (more than one is usually better) should be sharp, with the item's important features clearly visible.
  - Make sure that your image files are a reasonable size; buyers hate large photos that take a long time to load or require scrolling (we recommend reducing photos to no larger than 640×480 pixels).

    TIP Use a photo-editing program to reduce the pictures that your digital camera or scanner produces to a smaller size and save them as lower-resolution JPEG files. You want a size that loads quickly and can be viewed without scrolling, as discussed in Chapter 7.
- **Avoid animation and music like the plague.** Serious bidders click off of your item in a flash if they find either of these annoyances in your item description.

Presentation Issues to Consider

When you create your listings, remember that a variety of users will view your page using different browsers and operating systems. With that in mind, peruse the following helpful tips for creating your listing:

- **Design your page so it works with as many browsers as possible.** Any Web browser may view your listing. (For example, you can't assume that your buyers have a browser capable of properly rendering CSS.)
- **Use an appropriate font size.** The font size that you use should be large enough to be legible at a variety of screen resolutions. Standard font sizes such as a 10- to 12-point font are good examples. Some buyers won't bother to read your item description if it is in a tiny font size. At the same time, don't make the font size too large. Large fonts can make your auction item page look amateurish.
- **Don't use huge type that requires users to scroll the page a lot.** For example, four headings—all in a 48-point font—is way too big.
- **Use backgrounds that don't distract your users from the text and images on the Web site.**

  Avoid colored or patterned backgrounds because

  - People who are colorblind might have problems reading them.
  - Colored backgrounds can make your page hard to read when printed on a monochrome printer. (Many users print auctions for inventory records.)
  - They can make your page look amateurish.

Using a Template for Presenting Your Auction Item

In this section, we provide a handy HTML template that enables you to display pictures of your item alongside its description:

- **A left column contains two pictures of the auction item.**

  The example assumes that you're hosting the image files on a Web server that you control. (Though eBay will now let you host one image per auction for free, you must pay for additional images; if you don't mind spending money on this, feel free to upload them using the eBay auction setup form instead.) You should prepare the image files and upload them to your server before you begin using the template.
- **A right column contains text describing the item.**

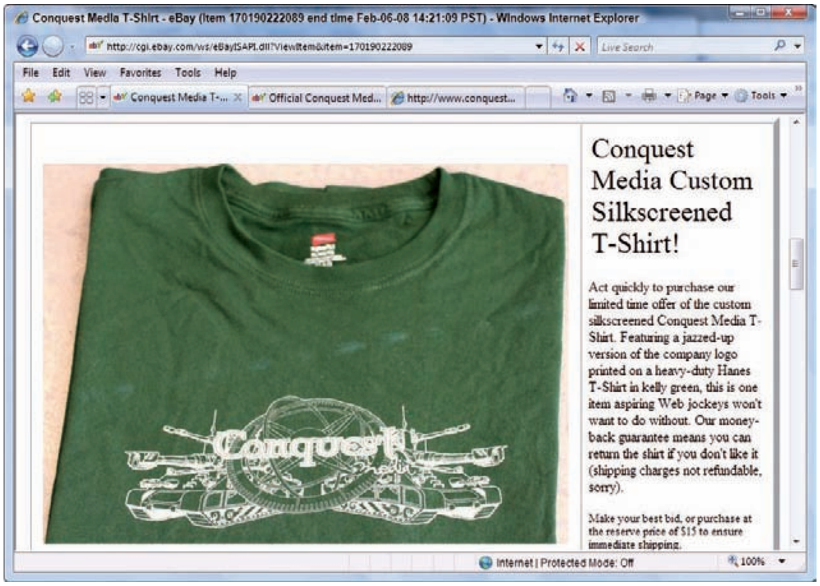
Listing 17-1 shows the HTML markup for the auction item description. You can type it in any text editor, replacing the parts set off by the HTML comment tags with the appropriate information (as indicated in the comment-tag text).

Listing 17-1: Auction Item HTML Template

```
<!-- Begin Description Table -->
<!-- Picture column -->

<table align="center" cellpadding="8" border="7"
       cellspacing="0" bgcolor="#FFFFFF">
<tr>
<td valign="top" align="Left" width="1%"><br /><br />

<!-- First picture goes below; replace URL with the location of your picture -->

<img border="0" align="top" hspace="5"
     src="http://www.example.com/images/image1.jpg"
     alt="Alternative image text" />

<br /><br />

<!-- Next picture goes below; replace URL with the location of your picture -->

<img border="0" align="top" hspace="5"
     src="http://www.example.com/images/image1.jpg"
     alt="Alternative image text" /></td>

<!-- Text column -->

<td valign="top" align="Left">

<!-- This table-within-a-table for the headline makes your description
     look better -->

<table border="0" >
  <tr><td align="Left" ><font face="Times New Roman" color="#000000" size="6">
      Your Exciting Item Title Goes Here!</font>
  </td></tr>
</table><br />

<p><font face="Times New Roman" color="#000000" size="3">

<!-- Begin Description -->

Replace this text with the description of your auction item.

<!-- End Description -->

</font></p>

<p><font face="Times New Roman" color="#000000" size="2">
```

```
<!-- Enter your payment terms and details here. -->

  </font></p>

<p><font face="Times New Roman" color="#000000" size="2">

<!-- Enter your shipping terms and details here. -->

  </font></p>

  </td>
  </tr>
</table>

  <!-- End Description Table -->
```

In Figure 17-3, you can see the on-screen results of the preceding auction item-description template. (We sold many copies of this item successfully on eBay.)


Conquest Media T-Shirt - eBay (Item 170190222089 end time Feb-06-08 14:21:09 PST) - Windows Internet Explorer
http://cgi.ebay.com/ws/eBayISAPI.dll?ViewItem&item=170190222089
Live Search
File Edit View Favorites Tools Help
Conquest Media T-...
Official Conquest Med...
http://www.conquest...
Page
Tools
Internet | Protected Mode: Off
100%

Conquest Media Custom Silkscreened T-Shirt!

Act quickly to purchase our limited time offer of the custom silkscreened Conquest Media T-Shirt. Featuring a jazzed-up version of the company logo printed on a heavy-duty Hanes T-Shirt in kelly green, this is one item aspiring Web jockeys won't want to do without. Our money-back guarantee means you can return the shirt if you don't like it (shipping charges not refundable, sorry).

Make your best bid, or purchase at the reserve price of $15 to ensure immediate shipping.

Figure 17-3: The template as it appeared on eBay.

Many auction sites, including eBay, host pictures for your item — often for free. For example, eBay hosts one picture for free, but you must pay for extra pictures. You might also consider using sites such as Picasa (`http://picasa.google.com`) that offer free image-hosting services that include Web access for online auction sellers.

Chapter 18

A Company Site

In This Chapter

- ▶ Deciding what your site needs to contain
- ▶ Including the basic parts of a Web site
- ▶ Separating content from presentation
- ▶ Building the site

Companies large and small differ on their office dress policies — from being required to wear three-piece suits in the office to being allowed to work in a SpongeBob T-shirt and torn cutoffs. However, all companies, despite their differences in formality and workwear, want to present themselves effectively to the outside world. As such, they want their Web sites to reek of confidence, capability, and professionalism. No one feels good about forking over hard-earned money to a company with a cheesy and tacky Web site (unless that company sells cheese and tacks).

In this chapter, you explore the basics of creating a company Web site — and look at the typical elements you want to utilize as you design your own company's site.

Issues to Consider When Designing Your Site

When you start to plan your company's Web site, the most important task is to consider the kind of people who are going to visit — potential or existing customers, clients, or partners. After you determine a list of the types of visitors, brainstorm about what they will want from your Web site.

Working with the concept of *personas* (okay, that's *personae* in Latin), in which you envision a few of the site's visitors and what they each want to get from the site, can be valuable. As you lay out the site, think about how each of these imaginary people interacts with your design. Will they find what they're looking for?

If you're designing a site for a company that has many departments, you'll soon discover that each department may have a different vision of what the company Web site should be. For instance, marketing wants the front page of the site to be a gigantic Flash animation showing all the company's products — whereas management wants every page on the site to look exactly like a corporate brochure and to look the same in every browser known to humankind.

Your job, as a Web designer and developer, is not only to design the site but also to educate people around you about what is possible and feasible — while staying within your budget.

Basic Elements of a Company Web Site

As you consider creating a company Web site, consider the following among basic elements that you may want to include. As a purely hypothetical but hopefully illustrative example, this sample company site includes six key files:

- **The initial Web page, `index.html`,** is the site's home page. It contains the basic marketing message about the company and its services; despite its name, it's entitled "About Us." For most company Web sites, the home page presents key facts about the company and identifies its business focus, target markets, and seeks to attract visitors to explore further.

  A site's home page can have any of a variety of file names, such as `index.html`, `default.html`, and `home.html`. You want to check with your Webmaster or your Web-hosting provider to determine the exact filename you should use. However, in general, the filename `index.html` will almost always work.

- **The products/services pages** are named `multimedia.html`, `print.html` and `web_design.html`. Each contains summary information about company projects and services, related to multimedia and print projects, as well as custom Web-design services. When creating filenames, keep them short, simple, and intelligible so you and others can at least guess what they hold.

- **The contact us page, `contactus.html`,** contains an e-mail link to the company to enable visitors to communicate with the principals via e-mail. Let's also observe again that it's essential to respond rapidly, professionally, and thoroughly to such inquiries. Many "Contact Us" pages also include a mail address and one or more phone numbers for the company; some even include (or point to) maps and directions to reach company HQ or other locations.

- **A press page,** `press.html`, contains
  - Links to the press releases generated by the company
  - Information that marketing thinks members of the press might want, such as news coverage, analyst reports, product reviews, and even image libraries (if applicable)

  This page isn't discussed in the rest of the chapter, but you can easily modify the basic HTML template discussed for the other site pages to create this unique page on your own.
- **An image,** `logo.gif`, is displayed on the site's home page to give visitors an initial impression of the company and its creative ethos. This could be any image, from a company logo to pictures of employees in action. On our sample site, we repeat this same image at the head of each page on the site. This is a common technique, and helps to establish a sense of continuity and branding to help hold your site together.
- **A style sheet,** `style.css`, contains the formatting instructions for each page of the site.

  Every page links to this style sheet by using the `<link>` tag. A change in this file changes the appearance of every page on the site.

The home page

Listing 18-1 shows the home-page markup for Conquest Media, our not-so-fictitious company (it represents author Jeff Noble's growing business venture). Figure 18-1 shows how it looks when displayed in a browser.

Listing 18-1: Our Company's Home Page (index.html)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
  <title>Conquest Media</title>
  <link href="style.css" rel="stylesheet" type="text/css">
</head>
<body>
<!-- Main text area for body copy-->
  <div align="center"><img src="images/logo.gif" /></div>
<!-- Page container that centers contents  -->
  <div id="container">
<!-- Navigation  -->
  <div id="nav">
```

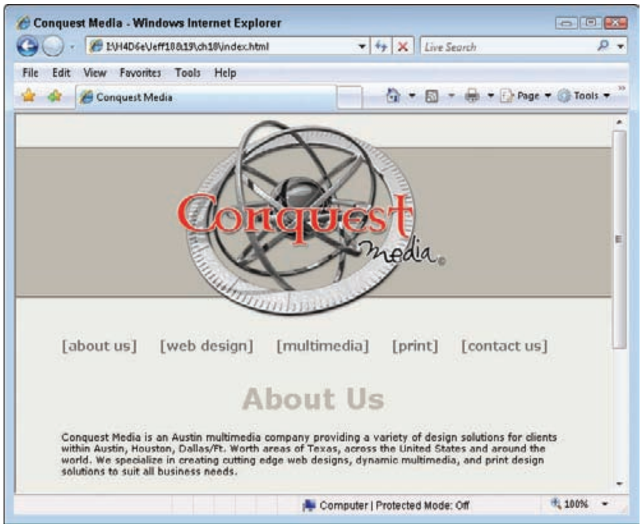
(continued)

Listing 18-1 *(continued)*

```
  <ul>
    <li><a href="index.html">[about us]</a></li>
    <li><a href="web_design.html">[web design]</a></li>
    <li><a href="multimedia.html">[multimedia]</a></li>
    <li><a href="print.html">[print]</a></li>
    <li><a href="contactus.html">[contact us] </a></li>
  </ul>
  </div> <!-- close nav division -->
  <div id="container">
  <h1>About Us</h1>
  <p>Conquest Media is an Austin multimedia company providing a
     variety of design solutions for clients within Austin,
     Houston, Dallas/Ft. Worth areas of Texas, across the United
     States and around the world. We specialize in creating
     cutting edge web designs, dynamic multimedia, and print
     design solutions to suit all business needs.</p>
  <p>We understand that every client of Conquest Media has
     different requests and specifications that we will gladly meet.
     It&rsquo;s our guarantee to work with you on a one-to-one basis
     to complete your project.</p>
  <p>Our talented team of 3D artists, web designers, multimedia
     specialists, and database programmers has the experience, talent,
     technical knowledge and skills to deliver impressive and effective
     results. Simply put, there are a lot of design studios out there,
     and anyone can say they are a designer, but we have the portfolio
     to back it up. </p>
  <p>We invite you to review our portfolio of web design, multimedia,
     and print designs to see just exactly what we do here, why we love
     it, and why people love our work.</p>
  <div id="footer">All images and content &copy; Conquest Media 2008.
     All Rights Reserved.</div>
<!-- end #container --></div>
</body>
</html>
```

As you look at the markup in Listing 18-1, you can see that it doesn't contain any information about colors, fonts, or how the page itself should be displayed. All that information is *in the style sheet,* which allows the most flexible approach to updating the site in the future.

The navigation used in the home page and for the other site pages is based on a simple text bar (a collection of text links, each enclosed in square brackets `[ ]`). These links are simple, easy to code, and can be *spidered* (automatically searched for keywords) by search engines such as Google. They work as well in older browsers as they do in newer ones.


Conquest Media - Windows Internet Explorer
I:\H4D6e\Jeff18&19\ch18\index.html
Live Search
File Edit View Favorites Tools Help
Conquest Media
Page Tools
Conquest media
[about us] [web design] [multimedia] [print] [contact us]
Computer | Protected Mode: Off
100%

About Us

Conquest Media is an Austin multimedia company providing a variety of design solutions for clients within Austin, Houston, Dallas/Ft. Worth areas of Texas, across the United States and around the world. We specialize in creating cutting edge web designs, dynamic multimedia, and print design solutions to suit all business needs.

Figure 18-1: Our company's home page.

If you want to use this template for your home page, just

- Change the contents of the `<title>` and `<h1>` tags, and customize the navigation to meet your needs (you can use the text rollovers for navigation described in Chapter 15 instead, if you prefer).
- Add the company's description where text that follows the `<h1>` element occurs, after deleting the Conquest Media description.

It's all greeked to me

If you would like to mock up a page for which you don't yet have actual content, we recommend using greeked text. *Greeked text* is placeholder text that typically starts with the Latin phrase "Lorem Ipsum" and gets repetitively nonsensical from there. It's easy to tell apart from real text.

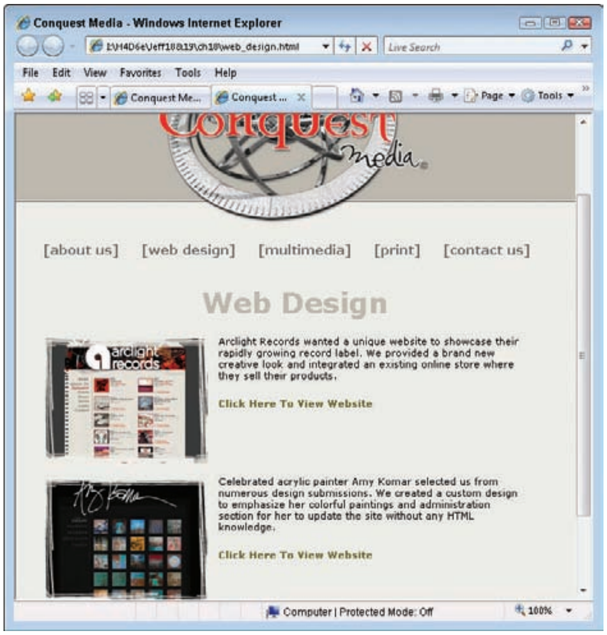
If you want to add greeked text to your page, check out `www.lipsum.com`, which will let you choose such options as the number of words, paragraphs, and bytes of greeked text you need. Then you can copy and paste the place holder text into your page.

The Web Design page

Listing 18-2 and Figure 18-2 show the company's Web Design page and demonstrate how the overall look is the same, yet slightly different, for an interior site page.

Listing 18-2: Our Company's Web Design Page (web_design.html)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Conquest Media</title>
  <link href="style.css" rel="stylesheet" type="text/css">
</head>
<body>
  <div id="header"><img src="images/logo.gif" /></div>
<!-- Page container that centers contents  -->
  <div id="container">
  <div id="nav">
  <ul>
    <li><a href="index.html">[about us]</a></li>
    <li><a href="web_design.html">[web design]</a></li>
    <li><a href="multimedia.html">[multimedia]</a></li>
    <li><a href="print.html">[print]</a></li>
    <li><a href="contactus.html">[contact us]</a></li>
  </ul>
  </div> <!-- end navigation area -->
<h1>Web Design </h1>
  <div id="item">
  <div id="leftColumn">
    <img src="images/portfolio_web-1.gif"  alt="Arclight Records"
     width="190" height="146">
  </div> <!-- end leftColumn -->
  <div id="rightColumn">
  <p>Arclight Records wanted a unique website to showcase their
     rapidly growing record label. We provided a brand new creative
     look and integrated an existing online store where they sell
     their products.</p>
  <p><a href="http://www.arclightrecords.com">
     Click Here To View Website</a></p>
  </div><!-- end rightColumn -->
  </div> <!-- end Arclight item -->
  <div id="item">
  <div id="leftColumn">
  <img src="images/portfolio_web-2.gif" alt="Amy Komar"
   width="190" height="146">
  </div> <!-- end leftColumn -->
  <div id="rightColumn">
  <p>Celebrated acrylic painter Amy Komar selected us from numerous
     design submissions. We created a custom design to emphasize her
     colorful paintings and administration section for her to update
     the site without any HTML knowledge.</p>
  <p><a href="http://www.amykomar.com">
     Click Here To View Website</a></p>
  </div> <!-- end rightColumn -->
  </div> <!-- end Amy Komaritem -->
  <div id="footer">All images and content &copy; Conquest Media 2008.
     All Rights Reserved.
  </div> <!-- end footer -->
  </div> <!-- end container -->
</body>
</html>
```


Conquest Media - Windows Internet Explorer
E:\H4D6e\Jeff18&19\ch18\web_design.html
Live Search
File Edit View Favorites Tools Help
Conquest Me...
Conquest ...
Page
Tools
Conquest media
arclight records
Computer | Protected Mode: Off
100%

[about us] [web design] [multimedia] [print] [contact us]

Web Design

Arclight Records wanted a unique website to showcase their rapidly growing record label. We provided a brand new creative look and integrated an existing online store where they sell their products.

Click Here To View Website

Celebrated acrylic painter Amy Komar selected us from numerous design submissions. We created a custom design to emphasize her colorful paintings and administration section for her to update the site without any HTML knowledge.

Click Here To View Website

Figure 18-2: Our company's Web Design page.

To use the template shown in Listing 18-2, follow these steps:

1. **Customize the title, heading, footer, and navigation bar for your page.**
2. **Add descriptive text within the `<p>` tag to describe your products.**
3. **If it makes sense to do so, create a bulleted or numbered list (`<ul>` or `<ol>`), and describe each product specifically within individual `<li>` tags instead of using paragraphs of text.**

TIP

You can add links to subpages from within the individual product descriptions. If you do this, use this page as a template for the individual product pages, but make sure to create a Products link in the navigation bar. That way, site visitors can retrace their steps back to where they came from without clicking the Back button.

The Contact Us page

This simple page allows visitors to the site to send their feedback directly to the company. It provides an e-mail link, as shown in Listing 18-3 and Figure 18-3.

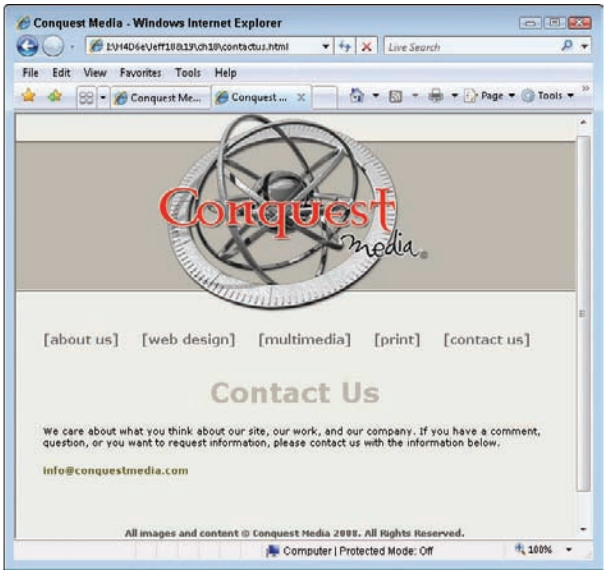
Listing 18-3: Contact Our Company (contact.html)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Conquest Media</title>
<link href="style.css" rel="stylesheet" type="text/css">
</head>
<body>
<!-- Main text area for body copy-->
<div id="header"><img src="images/logo.gif" /></div>
<!-- Page container that centers contents  -->
<div align="center">
<!-- Navigation  -->
<div id="nav">
<ul>
<li><a href="index.html">[about us]</a></li>
<li><a href="web_design.html">[web design]</a></li>
<li><a href="multimedia.html">[multimedia]</a></li>
<li><a href="print.html">[print]</a></li>
<li><a href="contactus.html">[contact us]</a></li>
</ul>
</div>
<div id="container">
<h1>Contact Us</h1>
  <p>We care about what you think about our site, our work, and our
     company. If you have a comment, question, or you want to request
     information, please contact us with the information below.</p>
<p><a href="mailto:info@conquestmedia.com">
info@conquestmedia.com</a></p>
<div id="footer">All images and content &copy; Conquest Media 2008.
     All Rights Reserved.</div>
<!-- end #container --></div>
<!-- end centered page> </div>
</body>
</html>
```

TIP

You could make the contact page considerably smarter by creating a form to solicit a name and an associated e-mail address, along with other information of potential interest. You might also want to add a little JavaScript that verifies whether valid information is entered into each of the fields. You want to make sure that

- A name was entered.
- Some text was entered in the message `<textarea>` field.
- Something resembling an e-mail address was given in the e-mail address field. (But there's no way to verify that the e-mail address belongs to the person filling out the form.)


Conquest Media - Windows Internet Explorer
E:\H4D6e\eff18&19\ch18\contactus.html
Live Search
File Edit View Favorites Tools Help
Conquest Me...
Conquest ...
Page
Tools
Conquest media
Computer | Protected Mode: Off
100%

[about us] [web design] [multimedia] [print] [contact us]

Contact Us

We care about what you think about our site, our work, and our company. If you have a comment, question, or you want to request information, please contact us with the information below.

info@conquestmedia.com

All images and content © Conquest Media 2008. All Rights Reserved.

Figure 18-3: Contacting our company.

TRICKS OF THE TRADE

Embedding an e-mail address into the HTML markup like this is generally a bad idea. It works, but spammers usually find the e-mail address in the HTML file and abuse it. In the long run, it's smarter to use a Common Gateway Interface (CGI) program that hands out this address on the server side, possibly after requiring requesters to jump through some hoops to prove they're human (as when they have to read some altered, wavy text on-screen and type it in to prove they can read). See Chapter 14 for more information on forms handling, or contact your Webmaster or Web host for assistance.

The style sheet

Listing 18-4 is a very basic version of a style sheet. All the HTML pages in this Web site reference this style sheet. The great advantage of using a style sheet to define your formatting instructions is that you can update this one file to give the whole site an entirely new look.

Listing 18-4: The Site-Wide Style Sheet (style.css)

```
body {
  font: 100% Verdana, Arial, Helvetica, sans-serif;
  background-image:url(images/background_page.gif);
  margin: 0;
  /* it's good practice to zero the margin and padding of the body
     element to account for differing browser defaults */
  padding: 0;
  text-align: center;
  /* this centers the container in IE 5* browsers. The text is then
     set to the left aligned default in the #container selector */
  color: #000000;
  font: 11px verdana, arial, sans-serif;
  font-weight: normal;
  }  /*end body styles */

h1{
  text-align: center;
  color: #beb9ae;
  } /* end h1 styles */

.portfolio_text{
  font-size: 13px;
  font-weight: bold;
  color: #393939;
  } /* end .portfolio text styles */

#header{
  width: 100%;
  height: 255px;
  text-align: center;
  background-image: url(images/background_top.gif);
  min-width: 565px;
  } /* end header styles */

#container {
  text-align: left;
  width: 575px;
  margin: 0 auto;
  } /* end container styles */
```

```
#nav {
  font-size: 15px;
  width: 100%;
  height: 50px;
  float: left;
  } /* end navigation bar styles */

#nav ul {
  font-size: 15px;
  margin: 0px;
  padding: 0px;
  white-space: nowrap;
  } /* end navigation bar unordered list styles */

#nav li {
  font-size: 15px;
  color: #9cb166;
  list-style-type: none;
  display: inline;
  padding: 10px;
  } /* end navigation bar list item styles */

a {
  font-weight: bold;
  text-decoration: none;
  color: #666633;
  background: transparent;
  } /* end anchor element styles */

a:visited {
  font-weight: bold;
  text-decoration: none;
  color: #666666;
  background: transparent;
  } /*end visited hyperlink display styles */

a:hover {
  color: #FA0000;
  background: transparent;
  text-decoration: underline;
  } /* end hover over hyperlink display styles */

a:active {
  color: #494949;
  background: transparent;
  font-weight: bold;
  text-decoration: underline;
  } /* end active hyperlink display styles */
```

(continued)

Listing 18-4 *(continued)*

```
#item {
  text-align: center;
  width: 100%;
  height: 160px;
  } /* end display item styles */

#leftColumn {
  float: left;
  width:190px;
  text-align: left;
  } /* end left column layout styles (site thumbnails) */

#rightColumn {
  text-align: left;
  float: left;
  margin-left: 10px;
  width: 350px;
  } /* end right column layout styles (descriptive text) */

#footer {
  font-size: 10px;
  font-weight: bold;
  color: #52544f;
  text-align: center;
  padding:40px;
  clear: both;
  } /*end page footer styles */

li {
  margin-bottom: 10px
  } /*end list item styles */
```

This style sheet sets up headers, footers, and navigation, and gives its users consistent styles to apply to list-items and body text. It's an integral part of the look of our pages, as removing the `stylesheet` reference in any single page then viewing it "naked" will quickly illustrate.

Give Your Visitors What They Need

As you put your company pages together, give your visitors what they need and make it easy for them to find, and they'll keep coming back for more. How can you find out what they need? Easy! Talk to those people in sales, support, marketing, and other departments, to find out what kinds of information, documents, software, and other goodies they give to customers all the time, and you've identified the things that are most important for you to highlight — and make available — on your Web pages.

Chapter 19

A Product Catalog

In This Chapter

- Designing a product catalog
- Choosing a shopping cart
- Adding the PayPal shopping cart to your pages
- Example HTML templates

In days gone by, a product catalog was a big production, a huge investment . . . and hard copy only. Not only did printing costs add up, but sending it to every George, Jerry, and Kramer meant only big companies could afford this maneuver. Unless your name was Sears & Roebuck, J.C. Penney, or Eddie Bauer, you had no way to reach a broad audience with your catalog.

The Internet, of course, has changed all this. Now, whether you're part of a big or a small company, you can produce and maintain a professional-looking catalog on your Web site. And, without a significant investment, you can even sell directly to your customers from an online store.

This chapter covers the basics of creating a product catalog and selling your goods on the Web.

Dissecting a Product Catalog

An online product catalog usually includes these components:

- **A navigation interface** to help users move easily through the catalog.

  The navigation interface is normally a menu system. Navigation interfaces are discussed briefly in Chapter 3.

- **At least one category page,** with several items listed in each category.

  Choosing a category from the menu system brings the user to a category page, which identifies individual items with

  - Thumbnail images
  - A brief description

  The image and title of an item are linked to that item's detail page. The user clicks the link for detailed information about an item.

  The site may allow the user to purchase items from the category page. (Some sites require purchases from the item's detail page.)

- **A detail page for each item in the catalog,** which usually displays

  - At least one large image of the item.
  - A detailed description of the item.
  - A button that adds the item to the site's *shopping cart,* if the site allows purchases. (This chapter covers shopping carts later.)

Design basics

Whether you *sell* directly to online buyers or just *show* your retail store's inventory, keep these design principles in mind:

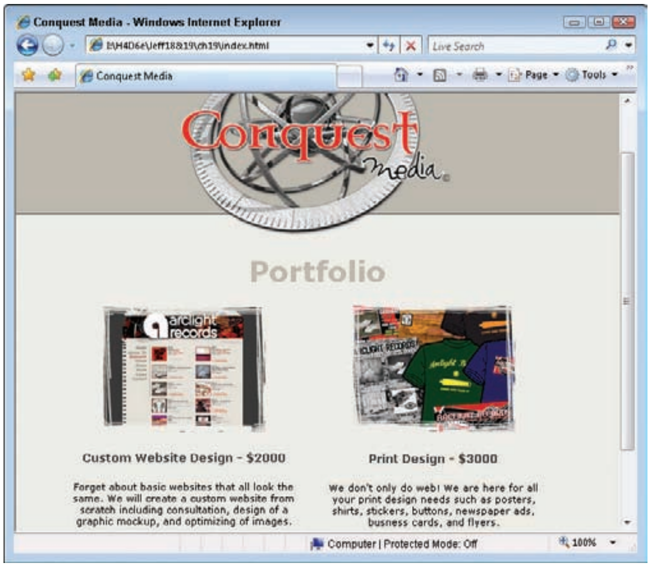
- **Keep your catalog clean.** Your online store should encourage users to *browse.* Users need to be able to see many items quickly, but you should also make it easy for users to get more detail on items that interest them.
- **Make your site design colorful, interesting, and fun.** (Keep the file size of the graphics small — no more than 50 KB per photo — so pages download quickly.)
- **Make it easy to get around.** Site navigation should be easy, logical, and obvious. If site visitors find interesting items quickly, they buy. Otherwise they lose interest and find what they want elsewhere.
- **Provide detail.** Visitors can't see or touch the item, so printed detail is a must. There are no space limitations on the Web (unlike printed catalogs), so you can include lots of details about items. (Info on shipping charges, returns, and contact should be easy to find, too.)
- **Make buying easy if you're selling.** Streamline the buying process as much as possible. An online purchase shouldn't take more than three screens. You'll make more sales and gain repeat customers.

  Too many online stores use shopping-cart software that seems positively *user-hostile.* Buyers must fill out page after page of selection and confirmation screens to complete a sale. Streamline this process to encourage visitors to become regular customers.

This chapter's example of a product catalog uses the following resources:

- Two templates for the product catalog:
  - A category page with small images of items within that category
  - A detail page for one example item
- The navigational menu system

Figure 19-1 shows the single category page for Conquest Media. Site visitors click the thumbnail picture of an item to jump to the detail page.


Conquest Media - Windows Internet Explorer
I:\H4D6e\Jeff18819\ch19\index.html
Live Search
Conquest Media
Page
Tools
Conquest media
Portfolio
Computer | Protected Mode: Off
100%

Custom Website Design - $2000

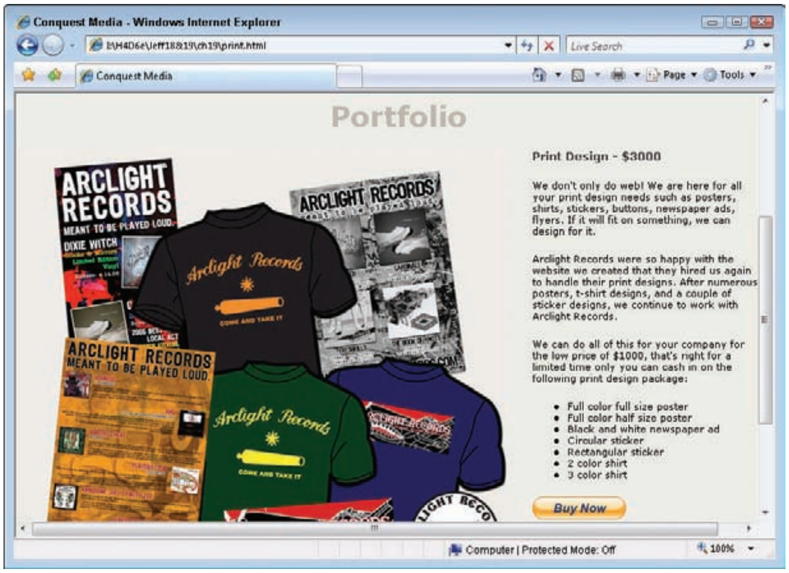
Forget about basic websites that all look the same. We will create a custom website from scratch including consultation, design of a graphic mockup, and optimizing of images.

Print Design - $3000

We don't only do web! We are here for all your print design needs such as posters, shirts, stickers, buttons, newspaper ads, busness cards, and flyers.

Figure 19-1: A category page from the online catalog.

After a visitor clicks an item in the category page, the item-detail page appears, as shown in Figure 19-2. This page contains the all-important Buy Now button, which allows the visitor to purchase the item.


Conquest Media - Windows Internet Explorer
Conquest Media
Page
Tools
Portfolio
Buy Now
Computer | Protected Mode: Off
100%

Print Design - $3000

We don't only do web! We are here for all your print design needs such as posters, shirts, stickers, buttons, newspaper ads, flyers. If it will fit on something, we can design for it.

Arclight Records were so happy with the website we created that they hired us again to handle their print designs. After numerous posters, t-shirt designs, and a couple of sticker designs, we continue to work with Arclight Records.

We can do all of this for your company for the low price of $1000, that's right for a limited time only you can cash in on the following print design package:

- Full color full size poster
- Full color half size poster
- Black and white newspaper ad
- Circular sticker
- Rectangular sticker
- 2 color shirt
- 3 color shirt

Figure 19-2: An item-detail page.

Choosing a Shopping Cart

If you want people to purchase from your site, you need a *shopping cart*. The cart allows buyers to purchase items and pay for them (usually with a credit card or a bank account transaction).

The shopping-cart software (which runs on a Web server) leads the buyer through the following steps of buying a product online:

1. The buyer selects an item and adds it to the shopping cart.
2. If the buyer wants to shop for other items, he or she can continue shopping and place other items in the shopping cart.
3. When ready to purchase, the buyer chooses to move to the *checkout* process.

   At checkout, the shopping cart software

   - Totals the purchases
   - Adds shipping costs (if necessary)
   - Leads the buyer through the payment process of entering such details as a credit card number and shipping address

TIP

In concept, a shopping cart for an online store is fairly simple. But in execution, it can get complex. This chapter surveys only the basics of e-commerce. If you are going to dive into it fully, we recommend these books:

- *Starting an Online Business For Dummies,* 5th Edition, by Greg Holden (Wiley)
- *MySQL/PHP Database Applications* by Brad Bulger, Jay Greenspan, and David Wall (Wiley)

PayPal

In this chapter, we use the shopping cart from a well-known e-commerce site, PayPal. Owned by eBay, PayPal's shopping cart is free for you to use on your Web site. Your customers can purchase multiple items with a single payment, and you can accept credit-card and bank-account payments. (PayPal charges you a transaction fee when you receive payment.)

PayPal offers a button generator that takes information about the name and price of an item you have for sale and creates HTML markup for an Add to Cart button that you can then insert directly into your product-catalog page.

TECHNICAL STUFF

This button generator and the PayPal shopping cart require a PayPal Premier or Merchant account. (PayPal Personal accounts don't accept debit- or credit-card payments; they only send or receive bank transfers.)

Other e-commerce solutions

PayPal is one of the easiest shopping carts to implement on your site, but many others are available.

WARNING!

The following technologies require more of a serious business and financial commitment to setting up your online presence.

Hosting e-commerce services

Hosted e-commerce services let you build an online storefront on your site but let the service provider deal with the technical aspects of your store and your transaction processing.

A good example of the online storefront service is Yahoo! Merchant Solutions (`http://smallbusiness.yahoo.com/ecommerce`). You can create a storefront on a Yahoo! server that features your own domain name, a product catalog, site-building tools (you can even avoid using raw HTML if you prefer), a secure shopping cart, e-mail order confirmations, integration with UPS for shipping, and order statistics tools. An online store on Yahoo! is fairly easy to set up and operate, especially if you're more merchant than Web developer. Prices start at about $40 a month.

Do-it-yourself software

If you are really a technical guru (or aren't faint of heart), you can install shopping-cart software on your own Web server and configure it manually.

TECHNICAL STUFF

If you choose this option, you need the technical know-how, a Web server, and a constant Internet connection for hosting your e-commerce Web site.

One do-it-yourself shopping cart software package is Zen Cart (`www.zen-cart.com`). It's a free, open-source shopping cart written in PHP. Stores created with Zen Cart are highly customizable with many useful features — for example, customers can review your products and you can customize tax and shipping rates for everywhere you sell your items.

WARNING!

If you use Zen Cart, expect to spend at least a few days setting up your store before you're ready for business. You must know how to upload and install Zen Cart software on your server, how to rename files and set UNIX permissions, and how to create a MySQL database. Then you must create or modify page templates for your store and supply numerous server-side parameters.

In general, we recommend you stick with PayPal or a hosted e-commerce solution to avoid the complexity of trying to do it all yourself. Don't say we didn't warn you!

Incorporating a PayPal shopping cart

Creating HTML markup for the shopping cart is easy: Use the PayPal button generator, and then copy and paste the resulting markup into your Web page.

REMEMBER

To use a PayPal shopping cart, you must be a PayPal Premier or Merchant account holder. After you establish an account, you can create your own Add to Cart and View Cart (or Buy Now) buttons by performing the instructions shown in the following sections.

Add to Cart button

Follow these steps to insert an Add to Cart button on your page:

TECHNICAL STUFF

1. **In your Web browser, go to the PayPal site:** `https://www.paypal.com`.

   This site is secure, so all transactions are encrypted between the site and your browser.

2. **Log in to your Premier or Merchant account.**

   Your account overview appears, with details only you need to know about.

3. **Click the Merchant Tools tab.**

4. **On the Merchant Tools page, click the PayPal Shopping Cart link.**

5. **On the PayPal Shopping Cart page, fill out the information about the item you want to sell.**

   You must enter the item name, the price, and the currency you accept. An item number (used in reports that PayPal provides for you after the sale) and the default country for the buyer's payment form are optional.

6. **In the Select an Add to Cart Button section, click to select the button style shown.**

   If you don't like the style that appears in response, click the Choose a Different Button link to pick a different button style.

   TIP

   You can *create* a button image and use it with the PayPal shopping cart:

   a. Create the button graphic in an image-editing program.

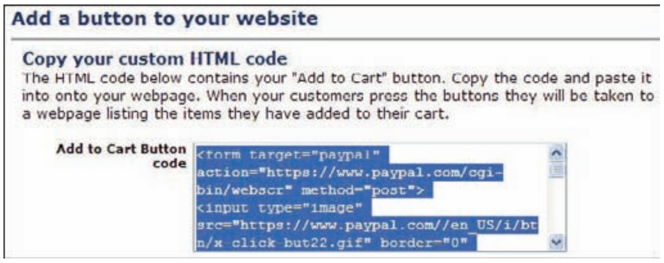
   b. Upload the graphic to a Web server.

   c. Select the Yes, I Would Like to Use My Own Image radio button on the PayPal Shopping Cart page.

   d. Fill in the URL for the graphic on your Web server.

7. **Click the Create Button Now button at the bottom of the page.**

8. **On the Add a Button to Your Website page, select *all* the text in the Add to Cart Button Code field, as shown in Figure 19-3, and then choose Edit➪Copy in your browser.**



Add a button to your website

Copy your custom HTML code

The HTML code below contains your "Add to Cart" button. Copy the code and paste it into onto your webpage. When your customers press the buttons they will be taken to a webpage listing the items they have added to their cart.

Add to Cart Button code

```
<form target="paypal"
action="https://www.paypal.com/cgi-
bin/webscr" method="post">
<input type="image"
src="https://www.paypal.com//en_US/i/bt
n/x-click-but22.gif" border="0"
```

Figure 19-3: Copying the Add to Cart button code.

9. **Switch to your HTML page editor and paste the cart code where you want the button to appear.**

10. **Save and preview the HTML page you just modified in your Web browser to see the button on the page.**

View Cart button

If you add a View Cart button to your page, follow these steps:

1. **Go to the PayPal Add a Button to Your Website page.**

   This HTML markup was generated at the same time as the Add a Button HTML markup.

2. **Select *all* the text in the View Cart Button Code field.**
3. **Copy and paste the code into your HTML page.**

Page Markup

Listing 19-1 includes the markup for the category page.

Listing 19-1: Category Page Template

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Conquest Media</title>
<link href="style.css" rel="stylesheet" type="text/css">
<!-- See sample Web site at www.edtittel.com/html4d6e/ch19 for stylesheet -->
</head>
<body>
<div id="header"><img src="images/logo.gif" /></div>
<div id="container">
  <div id="mainContent">
    <h1>Portfolio</h1>
      <div id="leftColumn"><img src="images/portfolio_web-1.gif"
         alt="Website Design" width="190" height="146" />
      <p class="portfolio_text">Custom Website Design - $2000</p>
      <p>Forget about basic websites that all look the same. We will create a
         custom website from scratch including consultation, design of a
         graphic mockup, and optimizing of images.</p>
      </div> <!-- end left column markup -->
      <div id="rightColumn">
      <p><a href="print.html"><img src="images/porfolio_print-1.gif"
         alt="Print" width="190" height="147" border="0"></a>
      <p class="portfolio_text">Print Design - $3000</p>
      <p>We don't only do web! We are here for all your print design needs such
         as posters, shirts, stickers, buttons, newspaper ads, busness cards,
         and flyers.</p>
      </div> <!-- end right column markup -->
      <div id="footer">All images and content &copy; Conquest Media 2008.
         All Rights Reserved.</div>
    <!-- end #mainContent --></div>
  <!-- end #container --></div>
</body>
</html>
```

Listing 19-2 includes the markup for the detail-page template.

Listing 19-2: Detail-Page Template

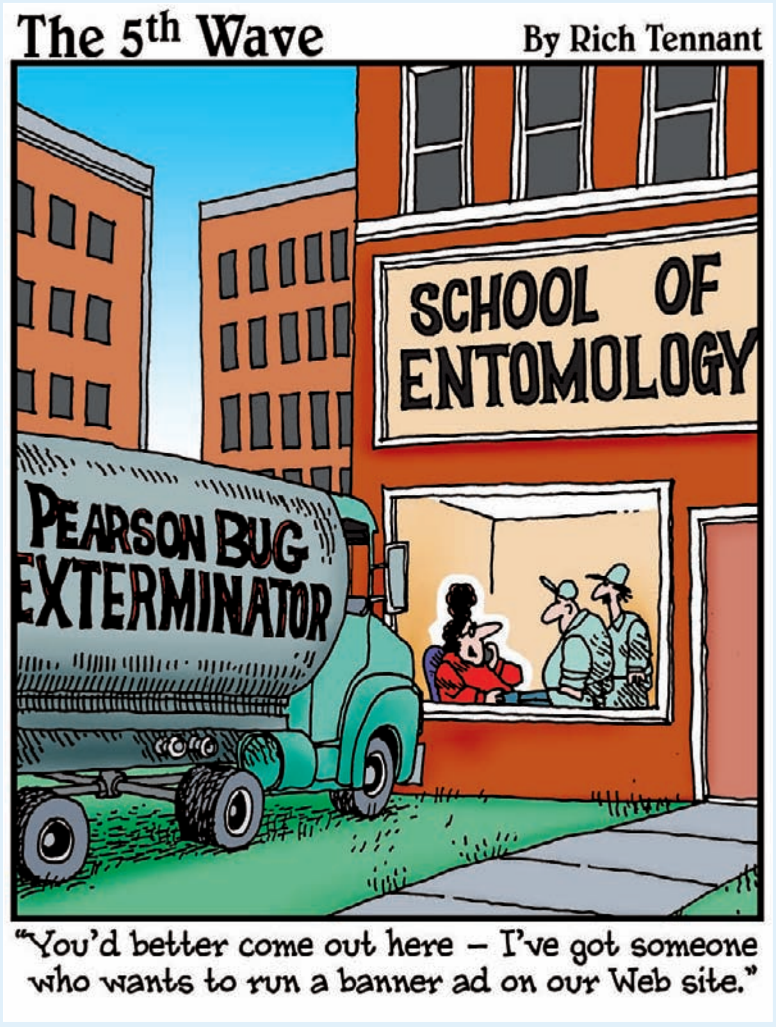
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Conquest Media</title>
  <link href="style.css" rel="stylesheet" type="text/css">
<!-- See sample Web site at www.edtittel.com/html14d6e/ch19 for stylesheet -->
</head>
<body>
  <div id="header"><img src="images/logo.gif" /></div>
  <div id="containerPrint">
    <div id="mainContent">
    <h1>Portfolio</h1>
    <div id="leftColumnPrint">
    <img src="images/porfolio_print-1.jpg" alt="Website Design"
     width="565" height="500" />
    </div> <!-- end leftColumnPrint markup -->
    <div id="rightColumnPrint">
    <p class="portfolio_text">Print Design - $3000</p>
    <p>We don't only do web! We are here for all your print design needs
       such as posters, shirts, stickers, buttons, newspaper ads, flyers.
       If it will fit on something, we can design for it. </p>
    <p>Arclight Records were so happy with the website we created that they
       hired us again to handle their print designs. After numerous posters,
       t-shirt designs, and a couple of sticker designs, we continue to work
       with Arclight Records.</p>
    <p>We can do all of this for your company for the low price of $1000,
       that's right for a limited time only you can cash in on the following
       print design package:</p>
    <!-- list of items-->
    <ul class="disc">
      <li>Full-color full-size poster </li>
      <li>Full-color half-size poster</li>
      <li>Black-and-white newspaper ad</li>
      <li>Circular sticker</li>
      <li>Rectangular sticker</li>
      <li>2-color shirt</li>
      <li>3-color shirt</li>
    </ul>
<!-- Paypal Button-->
```

(continued)

Listing 19-2 *(continued)*

```
     <form action="https://www.paypal.com/cgi-bin/webscr" method="post">
     <input type="hidden" name="cmd" value="_s-xclick" />
     <input type="image"
      src="https://www.paypal.com/en_US/i/btn/btn_buynow_LG.gif" border="0"
      name="submit"
      alt="Make payments with PayPal - it's fast, free and secure!" />
     <img alt="" border="0" src="https://www.paypal.com/en_US/i/scr/pixel.gif"
      width="1" height="1" />
     <input type="hidden" name="encrypted" value="-----BEGIN PKCS7-----
        your PayPal security code here (you'll grab this when you cut'n'paste)
        -----END PKCS7-----" />
       </form>
     </div> <!-- end rightColumnPrint markup -->
   <div id="footer">All images and content &copy; Conquest Media 2008.
      All Rights Reserved.</div>
   </div> <!-- end #mainContent -->
 </div> <!-- end #container -->
</body>
 </html>
```

Part VI
The Part of Tens



"You'd better come out here – I've got someone who wants to run a banner ad on our Web site."

In this part . . .

Here we point you at some undeniably cool HTML tools, cover top dos and don’ts for HTML markup, and help you catch potential bugs and errors in your Web pages. We also bring some killer HTML, XHTML, and CSS specification and resource sites to your attention. And with our tongues planted firmly in our cheeks (that makes it kind of hard to talk, you know), we recap some of the most important advice and information in this book. Enjoy!

Chapter 20

Ten HTML Dos and Don'ts

In This Chapter

- Concentrating on content
- Going easy on the graphics, bells, whistles, and hungry dinosaurs
- Creating well-formulated HTML and then testing, testing, testing
- Keeping it interesting after the building is ovah!

By themselves, HTML and XHTML are neither particularly complex nor overwhelmingly difficult. As some high-tech wags (including a few rocket scientists) have put it, *HTML ain't rocket science!* Nevertheless, important do's and don'ts can make or break the Web pages you build with HTML, XHTML, and CSS. Consider these humble admonishments as guidelines for making the most of your markup without losing touch with your users (or watching your page blow up on its launch pad).

If some points we make throughout this book seem to crop up here, too — especially regarding proper and improper use of (X)HTML — it's no accident. Heed ye well the prescriptions and avoid ye the maledictions. But hey, they're your pages. You can do what you want. Your users will decide the ultimate outcome. (We'd *never* say, "We told you so." Nope. Not us!)

Concentrate on Content

Any Web site lives or dies by its content. That a site is meaningful, that it delivers information directly, easily, and efficiently, and that a user can reasonably expect to find something new and interesting there with each new visit — all are pluses. But all those things (and more) rest on solid, useful content that gives visitors a reason to come (and return) to your site.

Never lose sight of your content

So we return to the crucial question of payload: page content. Why? Well, as Darrell Royal (legendary football coach of the University of Texas Longhorns in the '60s and '70s) is rumored to have said to his players, "Dance with who brung ya." In normal English (as opposed to Texan), this means that you should stick with the people who've supported you all along, and give your loyalty to those who've given it to you.

We're not sure what this means for football, but for Web pages it means keeping faith with your users and keeping content paramount. If you don't have strong, solid, informative content, users quickly get that empty feeling that hits when Web pages are content-free. Then they'll be off to richer hunting grounds on the Web, looking for content wherever it can be found.

TIP

To satisfy user hunger, put your most important content on your site's major pages. Save the frills and supplementary materials for secondary pages. The short statement of this principle for any kind of markup is: "Tags are important, but what's between the tags — the content — is what really counts." Chapter 3 covers making your content the best it can be.

Structure your documents and your site

For users, a clear road map of your content is as important for a single home page as it is for an online encyclopedia. When longer or more complex documents grow into a full-fledged Web site, a road map becomes more important still. This map ideally takes the form of (you guessed it) a flow chart of page organization and links. If you like pictures with a purpose, the chart could appear in graphic form in an explicitly labeled site map.

We're strong advocates of top-down page design: Don't start writing content or placing tags until you understand what you want to say and how you want to organize your material. Start building your (X)HTML document or documents using paper and pencil (or whatever modeling tool you prefer). Sketch out relationships within the content and among your pages. Know what and where you're building before rolling out the heavy equipment.

REMEMBER

Good content flows from good organization. It helps you stay on track during page design, testing, delivery, and maintenance. Organization helps users find their way through your site. Need we say more? Well, yes: Don't forget that *organization changes over time.* Revisit and critique your organization and structure on a regular basis — and don't be afraid to change either one to keep up with changes in content or focus.

Go Easy on the Graphics, Bells, Whistles, and Hungry Dinosaurs

Markup, scripting, and style sheets make much possible. But not all possibilities deserve implementation, — nor can Web sites live by snazzy graphics, special effects, and blinking marquees alone. Let your design and content drive the markup, the graphics, and interaction, and your site will do its job without overdazzling (or confusing) visitors.

Make the most from the least

More is not always better, especially when it comes to Web pages. Try to design and build your pages using minimal ornaments and simple layouts. Don't overload pages with graphics or cram in as many levels of headings as you can fit. Instead, do everything you can to make sure your content is easy to read and follow. To keep distractions and departures to a minimum, and make sure any hyperlinks you include add real value to your site.

TIP

Gratuitous links to useless information are nobody's friend; if you're tempted to link to a Webcam that shows a dripping faucet, resist, resist, resist!

REMEMBER

Structure and images exist to *highlight* content. The more bells, whistles, and dinosaur yowls dominate a page, the more they distract visitors from your content. Use structure and graphics sparingly, wisely, and carefully. Anything more poses obstacles to content delivery. Go easy on animations, links, and layout tags, or risk having your message (even your page) devoured by a hungry T. Rex.

Build attractive pages

When users visit Web pages with a consistent framework that focuses on content, they're likely to feel welcome. The important thing is to *supplement* content with graphics and links — don't overwhelm users with a surfeit of pictures and links. Making Web pages pretty and easy to navigate only adds to a site's basic appeal and makes your cybercampers even happier.

TIP

If you need inspiration, cruise the Web and look for layouts and graphics that work for you. If you take the time to analyze what you like, you can work from other people's design principles without having to steal details from their layouts or looks (which isn't a good idea anyway).

As you design Web documents, start with a basic, standard page layout. Pick a small, interesting set of graphical symbols or icons and adopt a consistent navigation style. Use graphics sparingly (yes, you've heard this before); make them as small as possible — limit size, number of colors, shading, and so on, while retaining eye appeal. After you build simple, consistent navigation tools, label them clearly and use them everywhere. Your pages can be both appealing and informative if you invest enough time and effort.

Create Well-Formulated HTML and Test

If you start with solid markup and good content — and then plough through what you've built to make sure everything works the way as it should (and communicates what it ought) — you're on your way to a great Web site. But once construction is over, testing begins. And only when testing produces positive results should you open your virtual doors to the public.

Keep track of those tags

Although you're building documents, it's easy to forget to use closing tags, even when they're required (for example, the `</a>` that closes the opening anchor tag `<a>`). When you're testing Web pages, some browsers can compensate for such errors, leaving you with a false sense of security.

The Web is no place to depend on the kindness of strangers. Scrutinize your tags to head off possible problems from browsers that might not be quite so understanding (or lax, as the case may be).

As for the claims that some vendors of HTML authoring tools make ("You don't have to know any HTML!"), all we can say is, *"Uh-huh, suuurre. . ."* HTML itself is a big part of what makes Web pages work; if you understand it, you can troubleshoot with minimal fuss. Also, only you can ensure that your pages' inner workings are correct and complete for your documents, whether you build them yourself or some program builds them for you.

We could go on and on about this, but we'll exercise some mercy and confine our remarks to the most pertinent items:

- **Keep track of tags yourself while you write or edit HTML by hand.** If you open a tag — be it an anchor, a text area, or whatever — create the closing tag for it right then and there, even if you have content to add. Most HTML editors do this for you.

- **Use a syntax checker to validate your work during the testing process.** Syntax checkers are automatic tools that find missing tags or errors. Use these whether you build pages by hand or with software. The W3C's (free) HTML Validator lives at `http://validator.w3.org`.
- **Obtain and use as many browsers as you can when testing pages.** This not only alerts you to missing tags, but it can also reveal potential design flaws or browser dependencies (covered in the "Avoid browser dependencies" section later in this chapter). This exercise also emphasizes the need for alternate text. That's why we also check our pages with Lynx (a character-only browser).
- **Always follow HTML document syntax and layout rules.** Just because most browsers don't require elements such as `<html>`, `<head>`, and `<body>` doesn't mean you can omit them. It means browsers don't give a hoot whether you use them or not. But browsers per se are not your audience. Your users (and future browsers) may indeed care.

REMEMBER

Although HTML isn't exactly a programming language, it still makes sense to treat it like one. Following formats and syntax helps you avoid trouble, and careful testing and rechecking of your work ensures a high degree of quality, compliance with standards, and a relatively trouble-free Web site.

Avoid browser dependencies

When you're building Web pages, the temptation to view the Web only in terms of your favorite browser is hard to avoid. That's why you must always recall that users view the Web in general (and your pages in particular) from many perspectives — through many different browsers.

During the design and writing phases, you'll probably hop between HTML and a browser's-eye view of your work. At that point, you should switch from one browser to another and test your pages among several browsers (including at least one text-only browser like Lynx). This helps balance how you visualize your pages and also helps keep you focused on content. (Using a text-only browser is also a great way to ensure that visually impaired visitors can still relate to your site.)

TIP

You can use public Telnet servers with Lynx (a character-mode browser) installed for free that don't require software installation. Otherwise, visit `http://brainstormsandraves.com/articles/browsers/lynx` for a good discussion of using Lynx when testing Web pages (you'll also find pointers to Lynx downloads for Windows, DOS, Mac OS, and other platforms there). There's even a free Firefox plugin for Lynx previews inside a pop-up window available at `https://addons.mozilla.org/en-US/firefox/addon/1944`.

During testing and maintenance, you must browse your pages from many points of view. Work from multiple platforms; try both graphical and character-mode browsers on each page. Testing takes time but repays that investment with pages that are easy for everyone to read and follow. It also helps viewers who come at your materials from platforms other than your own, and helps your pages achieve true independence from any single viewpoint. Why limit your options?

TIP

If several pages on your site use the same basic (X)HTML, create one template for those pages. Test that template with as many browsers as you can. When you're sure the template is browser-independent, use it to create other pages. This helps every page look good, regardless of which browser visitors use, and moves you closer to real HTML enlightenment.

Navigating your wild and woolly Web

Users who view the splendor of your site don't want to be told *you can't get there from here.* Aids to navigation are vital amenities on a quality Web site. A *navigation bar* requires a consistent placement and use of controls to help users get from A to B. Judicious use of links and careful observation of what constitutes a complete screen (or screenful) of text, helps users minimize (or even avoid) scrolling. Text anchors make it easy to move to previous and next screens, as well as to the top, index, and bottom in any document. Just that easy, just that simple — or so it appears to the user.

TIP

We believe in the *low scroll* rule: Users should have to scroll *no more than one* screenful in either direction from a point of focus or entry to find a navigation aid that lets them jump (not scroll) to their next point of interest.

We don't believe that navigation bars are mandatory — nor that names for controls should always be the same. But we do believe that the more control you give users over their reading, the better they like it. The longer a document gets, the more important such controls become; they work best if they occur about every 30 lines in longer documents (or in a separate, always-visible frame if you use HTML frames).

Keep It Interesting After It's Built!

The tendency to sit on one's fundament, if not rest on one's laurels, after launching a Web site is nearly irresistible. It's okay to sit down, but it isn't okay to leave things alone for too long or to let them go stale for lack of attention and refreshment. If you stay interested in what's on your site after it's ready for prime time, your content probably won't go past its freshness date. Do what you can (and what you must) to stay on top of things, and you'll stay engaged — as should your site visitors!

Think evolution, not revolution

Over time, Web pages change and grow. Keep a fresh eye on your work and keep recruiting fresh eyes from the ranks of those who haven't seen your work before to avoid what we call "organic acceptance."

This concept is best explained by the analogy of your face in the mirror: You see it every day; you know it intimately, so you aren't as sensitive as someone else to how your face changes over time. Then you see yourself on video, or in a photograph, or through the eyes of an old friend. At that point, changes obvious to the world reveal themselves to you as you exclaim, "I've gone completely gray!" or "My spare tire could mount on a semi!"

Changes to Web pages are usually evolutionary, not revolutionary. They proceed in small daily steps; big leaps are rare. Nevertheless, you must stay sensitive to the underlying infrastructure and readability of your content as pages evolve. Maybe the lack of on-screen links to each section of your Product Catalog didn't matter when you had only three products — but now that you offer 25, they're a must. You've heard that form follows function; in Web terms, the structure of your site needs to follow changes in its content. If you regularly evaluate your site's effectiveness at communicating, you know when it's time to make changes, large or small.

This is why user feedback is crucial. If you don't get feedback through forms or other means, aggressively solicit some from your users. If you're not sure how you're doing, consider: If you don't ask for feedback, how can you tell?

Beating the two-dimensional-text trap

Because of centuries of printed material and the linear nature of books, our mindsets may need adjustment. The nonlinear potentials of hypermedia give new meaning to the term *document*, especially on the Web. It can be tempting to pack pages full of capabilities until they resemble a Pony Express dynamite shipment that gallops off in many directions at once. Be safe: Judge hypermedia by whether it

- Adds interest
- Expands on your content
- Makes a serious — and relevant — impact on users

Within these constraints, such material can vastly improve any user's experience of your site.

Stepping intelligently outside old-fashioned linear thinking can improve your users' experience of your site and make your information more accessible to its audience. That's why we encourage careful use of document indexes, cross-references, links to related documents, and other tools to help users navigate around your site. Keep thinking about the impact of links as you look at other people's Web materials; it's the quickest way to shake free of the linear-text trap. (The printing press was high-tech for its day, but that *was* nearly 600 years ago!) If you're looking for a model for Web site behavior, don't think about your new trifold four-color brochure, however eye-popping it may be; think about how your customer-service people talk with new customers on the phone. *("What can I do to help you today?")*

Overcoming inertia takes vigilance

When you deal with your Web materials post-publication, it's only human to goof off after finishing a big job. Maintenance isn't as heroic or inspiring as creation, yet it represents most of the activity required to keep any document alive and well. Sites that aren't maintained often become ghost sites; users stop visiting when developers stop working on them. Never fear — a little work and attention to detail keeps pages fresh. If you start with something valuable and keep adding value, a site's value appreciates over time — just like any other artistic masterpiece. Start with something valuable and leave it alone, and it soon becomes stale and loses value.

TIP

Consider your site from the viewpoint of a master aircraft mechanic: Correct maintenance is a real, vital, and an on-going accomplishment, without which you risk a crash. A Web site, as a vehicle for important information, deserves regular attention; maintaining a Web site requires discipline and respect. See `www.disobey.com/ghostsites/index.shtml` for a humorous look at ghost sites.

REMEMBER

Keeping up with change translates into creating (and adhering to) a regular maintenance schedule. Make it somebody's job to spend time on a site regularly; check to make sure the job's getting done. If people get tagged to handle regular site updates, changes, and improvements, they flog other participants to give them tasks when scheduled site maintenance rolls around. Pretty soon, everybody's involved in keeping information fresh — just as they should be. This keeps your visitors coming back for more!

Chapter 21

Ten Ways to Exterminate Web Bugs

In This Chapter

- Avoiding gaffes in markup and spelling
- Keeping links hot and fresh
- Gathering beta-testers to check, double-check, and triple-check your site
- Applying user feedback to your site

After you put the finishing touches on a set of pages (but before you go public on the Web for all the world to see), it's time to put them through their paces. Testing remains the best way to control site quality and effectiveness.

Thorough testing *must* include content review, analysis of (X)HTML and CSS syntax and semantics, link checks, and various sanity checks to make doubly sure that what gets built is what you really want. Read this chapter for some gems of testing wisdom (learned from a lifetime of Web adventures) as we seek to rid your Web pages of bugs, errors, and lurking infelicities. Out! Out! Darned Spot!

Avoid Dead Ends and Spelling Faux Pas

A sense of urgency that things must work well and look good on a Web site never fails to motivate you to keep your site humming along. That said, if you work from a visual diagram of how your site is (or should be) organized, you'll be well-equipped to check structure, organization, and navigation. Likewise, if you put your pages through their paces regularly (or at least each time they change) with a spell checker, you'll be able to avoid unwanted *tpyos*.

Make a list and check it — twice

Your design should include a road map (often called a *site map*) that tells you what's where in every individual (X)HTML document and stylesheet in your site — and clues you in on the relationships among its pages. If you're really smart, you'll keep this map up-to-date as you move from design to implementation. (In our experience, things always change when you go down this path.) If you're merely as smart as the rest of us, don't berate yourself — update that map *now!* Be sure to include all intra- and interdocument links.

A site map provides the foundation for a test plan. Yep, that's right — effective testing isn't random. Use your map to

- Investigate and check every page and every link systematically.
- Make sure everything works as you think it should — and that what you built has some relationship (however surprising) to your design.
- Define the list of things to check as you go through the testing process.
- Check everything (at least) twice. (Red suit and reindeer harness optional.)

Master text mechanics

By the time any collection of Web pages comes together, you're looking at thousands of words, if not more. Yet many Web pages get published without a spell check, which is why we suggest — no, *demand* — that you include a spell check as a step when testing and checking your materials. (Okay, we don't have a gun to your head, but you *know* it's for your own good.) Many (X)HTML tools, such as Expression Web, HomeSite, and Dreamweaver, include built-in spell checkers, and that's the first spell-check method you should use. These (X)HTML tools also know how to ignore markup and just check your text.

Even if you use (X)HTML tools only occasionally, and hack out most of your markup by hand, do a spell check before posting your documents to the Web. (For a handy illustration of why this step matters, keep a log of spelling and grammatical errors you find during your Web travels. Be sure to include a note on how those gaffes reflect on the people who created the pages involved. Get the message?)

You can use your favorite word processor to spell check your pages. Before you check them, add (X)HTML and CSS markup to your custom dictionary, and pretty soon the spell checker runs more smoothly — getting stuck only on URLs and other strange strings that occur from time to time in Web documents.

If you'd prefer a different approach, try any of the many (X)HTML-based spell-checking services now available on the Web. We like the free Lite Edition of the CSE HTML Validator (`www.htmlvalidator.com/`).

If CSE HTML Validator Lite's spell checker doesn't float your boat, visit a search engine, such as `www.yahoo.com` or `www.google.com`, and use **web page spell check** as a search string. Doing so lets you produce a list of spell-checking tools made for Web pages.

One way or another, persist until you root out all typos and misspellings. Your users may not thank you for your impeccable use of language — but if they don't trip over errors while exploring your work, they'll think more highly of your pages (and their creator), even if they don't know why. Don't forget to put your eyeballs on the copy, though, too: no spell checker in the world will recognize "It's time two go too the store" as badly-mangled text, though you should catch that right away! Better yet, hire a professional editor or proofreader to help out during testing.

Keep Your Perishables Fresh!

New content and active connections to current, relevant resources are the hallmarks of a well-tended Web site. You can't achieve these goals without regular (sometimes constant) effort, so plan for ongoing activity. The rewards can be huge — starting with a genuine sense of user excitement at what new marvels and treasures reveal themselves on their next visit to your site. Such anticipation is impossible to imitate (without doing what you'll have to do to keep things fresh in the first place).

Lack of live links — a loathsome legacy

We performed an unscientific, random-sample test to double-check our own suspicions; users told us that positive impressions of a particular site are proportional to the number of working links they find there. The moral of this survey: *Always check your links.* This is as true after you publish your pages as it is before they're made public. Nothing irritates users more than a link that produces the dreaded `404 File not found` error instead of the good stuff they seek! Remember, too, that link checks are as indispensable to page maintenance as they are to testing.

If you're long on 21st-century street smarts, hire a robot to do this job for you: They work long hours (with no coffee breaks), don't charge much, and check every last link in your site (and beyond, if you let them). The best thing about robots is that you can schedule them to work at regular intervals: They always show up on time, always do a good job, and never complain (though we haven't found one that brings homemade cookies or remembers birthdays). All you must do is search online for phrases like link checker. You'll find lots to choose from!

We're fond of a robot named MOMspider, created by Roy Fielding of the W3C. Visit the MOMspider site at `http://ftp.ics.uci.edu/pub/websoft/MOMspider`. This spider takes some work to use, but you can set it to check only local links, and it does a bang-up job of catching stale links before users do. (Some HTML software, such as HomeSite, includes a built-in link checker to check your links both before and after you post your pages.)

TIP

If a URL points to one page that simply points to another (a pointer), you can't leave that link alone. Sure, it works, but for how long? And how annoying! So if your link-checking expedition shows a pointer that points to a pointer (yikes), do yourself (and your users) a favor by updating the URL to point *directly* to the real location. You save users time, reduce Internet traffic, and earn good cyberkarma.

When old links must linger

If you must leave a URL active even after it has become passé to give your users time to bookmark your new location, instruct browsers to jump straight from old page to new by including the following HTML command in the old doc's `<head>`:

```
<meta http-equiv="refresh" content="0"; url="newurlhere" />
```

This nifty line of code tells a browser that it should refresh the page. The delay before switching to the new page is specified by the value of the `content` attribute, and the destination URL is determined by the value of the `url` attribute. If you build such a page, also include a plain-vanilla link in its `<body>` section, so users with older browsers can follow that link manually, instead of automatically. You might also want to add text that tells visitors to update their bookmarks with the new URL. Getting there may not be half the fun, but it's the whole objective.

Make your content mirror your world

When it comes to content, the best way to keep things fresh is to keep up with the world in which your site resides. As things change, disappear, or pop up in that world, similar events should occur on your Web site. Since something new is always happening, and old ways or beliefs are always fading away — even in studies of ancient cultures or beliefs — if you report on what's new and muse on what's fading from view, you'll provide constant reasons for your visitors to keep coming back for more. What's more, if you can accurately and honestly reflect (and reflect upon) what's happening in your world of interest, you'll grab loyalty and respect as well as continued patronage.

Check Your Site, and Then Check It Again!

There's an ongoing need for quality control in any kind of public content, but that need is particularly acute on the Web, where the whole world can stop by (and where success often follows the numbers of those who drop in *and return*). You must check your work while you're building the site and continue to check your work over time. This practice forces you to revisit your material with new and shifting perspectives, and to evaluate what's new and what's changed in the world around you. That's why testing and checking are never really over; they just come and go — preferably, on a regular schedule!

Look for trouble in all the right places

You and a limited group of hand-picked users should thoroughly test your site before you share it with the rest of the world — and more than once. This process is called *beta-testing,* and it's a bona fide, five-star *must* for a well-built Web site, especially if it's for business use. When the time comes to beta-test your site, bring in as rowdy and refractory a crowd as you can find. If you have picky customers (or colleagues who are pushy, opinionated, or argumentative), be comforted knowing that you have found a higher calling for them: Such people make ideal beta-testers — if you can get them to cooperate.

TIP

Don't wait till the very last minute to test your Web site. Sometimes the glitches found during the beta-test phase can take weeks to fix. Take heed: Test early and test often, and you'll thank us in the long run!

Beta-testers use your pages in ways you never imagined possible. They interpret your content to mean things you never intended in a million years. They drive you crazy and crawl all over your cherished beliefs and principles. And they do all this before your users do! Trust us, that's a blessing — even if it's in disguise.

These colleagues also find gotchas, big and small, that you never knew existed. They catch typos that word processors couldn't. They tell you things you left out and things that you should have omitted. They give you a fresh perspective on your Web pages, and they help you see them from extreme points of view.

REMEMBER

The results of all this suffering, believe it or not, are positive. Your pages will be clearer, more direct, and more correct than they would have been had you tested them by yourself. (If you don't believe us, of course, you *could* try skipping this step. And when real users start banging on your site, forgive us if we don't watch.)

Cover all the bases with peer reviews

If you're a user with a simple home page or a collection of facts and figures about your private obsession, this tip may not apply to you. Feel free to read it anyway — it just might come in handy down the road.

If your pages express views and content that represent an organization, chances are, oh, *about 100 percent* that you should subject your pages to peer-and-management review before publishing them to the world. In fact, we recommend that you build reviews into each step along the way as you build your site — starting by getting knowledgeable feedback on such basic aspects as the overall design, writing copy for each page, and the final assembly of your pages into a functioning site. These reviews help you avoid potential stumbling blocks, such as unintentional off-color humor or unintended political statements. If you have any doubts about copyright matters, references, logo usage, or other important details, get the legal department involved. (If you don't have one, you may want to consider a little consulting help for this purpose.)

Building a sign-off process into reviews so you can prove that responsible parties reviewed and approved your materials is a good idea. We hope you don't have to be that formal about publishing your Web pages, but it's far, far better to be safe than sorry. (This process is best called *covering the bases*, or perhaps it's really covering something else? You decide.)

Use the best tools of the testing trade

When you grind through your completed Web pages, checking your links and your HTML, remember that automated help is available. If you visit the W3C HTML Validator at `http://validator.w3.org`, you'll be well on your way to finding computerized assistance to make your HTML pure as air, clean as the driven snow, and standards-compliant as, ah, *really well-written HTML.* (Do we know how to mix a metaphor, or what?)

Likewise, investigating link checkers covered earlier in the chapter is smart; use them regularly to check links on your pages. These faithful servants tell you if something isn't current, and tell you where to find links that need fixing.

Schedule site reviews

Every time you change or update your Web site, you should test its functionality, run a spell check, perform a beta test, and otherwise jump through important hoops to put your best foot forward online. But sometimes you'll make just a small change — a new phone number or address, a single product listing, a change of name or title to reflect a promotion — and you won't go through the whole formal testing process for "just one little thing."

That's perfectly understandable — but one thing inevitably leads to another, and so on. Plus, if you solicit feedback, chances are good that you'll get something back that points out a problem you'd never noticed or considered before. It's essential to schedule periodic Web site reviews, even if you've made no big changes or updates since the last review. Information grows stale, things change, and tiny errors have a way of creeping in as one small change succeeds another.

Just as you take your car in for an oil change or replace your air-conditioning filter, plan to check your Web site regularly. Most big organizations we talk to do this every three months or so; others do it more often. Even when you think you have no bugs to catch, errors to fix, or outdated information to refresh, you'll often be surprised by what a review turns up. Make this part of your routine, and your surprises will be less painful — and require less work to remedy!

Let User Feedback Feed Your Site

Who better to tell you what works and what doesn't than those who use (and hopefully, even depend on) your site? Who better to say what's not needed and what's missing? But if you want user feedback to foster site growth and evolution, you must not only ask for it, you have to encourage it to flow freely and honestly in your direction, then act on that feedback to keep those wellsprings working.

Foster feedback

Even after you publish your site, testing never ends. (Are you having flashbacks to high school or college yet? We sure are.) You may not think of user feedback as a form (or consequence) of testing, but it represents the best reality check your Web pages are ever likely to get, which is why doing everything you can — including offering prizes or other tangibles — to get users to fill out HTML forms on your Web site is a good idea.

This reality check is also why reading *all* feedback you get is a must. Go out and solicit as much feedback as you can handle. (Don't worry; you'll soon have more.) But carefully consider all feedback that you read — and implement the ideas that actually bid fair to improve your Web offerings. Oh, and it's a really good idea to respond to feedback with personal e-mail, to make sure your users know you're reading what they're saying. If you don't have time to do that, make some!

Even the most finicky and picky of users can be an incredible asset: Who better to pick over your newest pages and to point out the small, subtle errors or flaws they so revel in finding? Your pages will have contributed mightily to the advance of society by actually finding a legitimate use for a universal delight in nitpicking. And your users can develop a real stake in boosting your site's success, too. Working with users gets them more involved, and helps guide the content of your Web pages (if not the rest of your professional or obsessional life). Who could ask for more? Put it this way: You may yet find out, and it could be very helpful.

If you give to them, they'll give to you!

Sometimes, simply asking for feedback or providing surveys for users to fill out doesn't produce the results you want — either in quality or in volume. Remember the old days when you'd occasionally get a dollar bill in the mail to encourage you to fill out a form? It's hard to deliver cold, hard cash via the Internet, but a little creativity on your part should make it easy for you to offer your users something of value in exchange for their time and input. It could be an extra month on a subscription, discounts on products or services, or some kind of freebie by mail. (Maybe you can finally unload those stuffed Gila monsters you bought for that trade show last year. . . .)

But there's another way you can give back to your users that might not even cost you too much. An offer to send participants the results of your survey, or to otherwise share what you learn, may be all the incentive participants need to take the time to tell you what they think, or to answer your questions. Just remember that you're asking your users to give of their time and energy, so it's only polite to offer something in return.

Chapter 22

Ten Cool HTML Tools and Technologies

In This Chapter

- Identifying your HTML toolbox needs
- Discovering your favorite HTML editor
- Adding a graphics application to your toolbox
- Authoring systems for the Web
- Understanding essential utilities for Web publishing

HTML documents are made of plain text, which means you can build one using a no-frills text editor like Notepad. Once upon a time, that was all Web authors used. But as the Web has evolved, so have the tools used to create Web pages. Nowadays, Web authoring is complex enough that a simple text editor can't cut unless

- ✓ You don't care (much) about graphics and HTML validation.
- ✓ You're on a quick in-and-out mission to make small changes to an existing HTML document.

As you gain more experience with HTML, you'll build your own HTML toolbox. This chapter should help you stock that toolbox. In fact, some of these tools may already be on your system, quietly waiting to help you create amazing Web pages.

TIP

When you go shopping for items for your HTML toolbox, look for good buys. Students and educators often qualify for big discounts on major-brand software, if they use a search engine to look for "educational software discount." But careful shopping can save anybody money on just about any software purchase. Try comparison-shopping at sites such as CNET Shopper (`shopper.cnet.com`) or PC Magazine (`http://pcmag.shopping.com`).

HTML Editors

This book explains how to create and maintain (X)HTML pages with nothing more complicated than a pocketknife and a ball of string. But a good, capable HTML editor can turn the chore of creating complicated (X)HTML pages into a relatively easy task.

HTML editors come in two flavors. The flavor you need depends on the complexity of the Web page you are creating or editing.

- **Helper editors** have fewer capabilities.
- **WYSIWYG** (what you see is what you get) **editors** do everything but your laundry.

Helper editors

An HTML helper works the way it sounds. It helps you create HTML, but it doesn't do all the markup work for you.

In a helper, HTML is displayed "raw" — tags and all. You can reach right into the code and tweak it (provided you have *HTML, XHTML & CSS For Dummies*).

But good helpers save time and lighten your load. Functions like these make HTML development easier and more fun:

- Tags are a different color than content.
- The spell checker knows tags aren't misspelled words.

Use a helper editor when you're building complex tables or multilevel lists. The more complex your markup, the more help a helper editor can provide!

HomeSite: The champ

HomeSite is an HTML editor suitable for both beginners and professionals. It requires HTML knowledge to use, but it assists you at every step.

We like the HomeSite interface. You can

- Browse images directly in the editor.
- Customize the toolbars and menus for your personal needs.
- Create a browser view instantly by clicking a tab.
- Move and use context menus with a right-click of your mouse.

Text is easy to enhance and modify with features like these:

- Color-coded HTML
- Integrated spell checker
- Search-and-replace tools to update whole projects, folders, and files
- Internal HTML validation
- Extensive online help if you need to access documentation on HTML and other popular scripting languages

HomeSite helps you perform

- Project management
- Link verification
- File uploads to a remote Web server

TIP

HomeSite retails for $99, and works only on Windows, in case you think you might want to buy it. If you don't have HomeSite already, and don't want to fork out the cash, try one of the following challengers as your helper editor instead (note also that HomeSite comes bundled with Dreamweaver).

Contenders

There are many more good HTML helper editors than there are good WYSIWYG editors. Here's our slate of alternatives.

BBEdit/TextWrangler

BBEdit has ruled the Macintosh world for years. It comes in two versions:

- A free product formerly known as BBEdit Lite has been superseded by a newer, free text editor called TextWrangler.
- BBEdit ($200 retail)

  TIP

  If you don't need the powerful and specialized set of HTML editing, preview, and cleanup tools that come with BBEdit, use TextWrangler and save! (A detailed features comparison is available online at `www.barebones.com/products/bbedit/threeway.shtml`.)

If you use a Macintosh, check BBEdit out at `www.barebones.com`.

HTML-Kit

HTML-Kit is a compact Windows tool with

- Menu-driven support for both HTML and Cascading Style Sheets (CSS) markup
- A nice preview window for a browser's-eye view of your markup

If you want to download HTML-Kit, go to `www.chami.com/html-kit`. You can download a free version, or register your copy for $65 and obtain a bunch of extra tools — including a spiffy table designer, a log analyzer, and a nifty graphical HTML/XHTML/XML editor that lets you view and navigate all those documents through their syntactical structure.

WYSIWYG editors

A WYSIWYG editor creates markup for you as you create and lay out Web page content on your monitor (often by dragging and dropping visual elements, or working through GUI menus and options), shielding your delicate eyes from bare markup along the way. These tools are like word processors or page-layout programs; they do lots of work for you.

TRICKS OF THE TRADE

WYSIWYG editors make your work easier and save hours of endless coding — you have a life, right? — but you should only use WYSIWYG editors during the initial design stage. For example, you can use a WYSIWYG editor to create a complex table in under a minute during initial design work. Later, when the site is live, you would then use a helper to refine and tweak your HTML markup directly.

Dreamweaver: still the champ

Dreamweaver is the best WYSIWYG Web development tool for Macintosh and PC systems. Many (if not most) Web developers use Dreamweaver. Dreamweaver is an all-in-one product that supports

- Web-site creation
- Maintenance
- Content management

The current version is Adobe Dreamweaver CS3. It also belongs to a suite of products — Adobe Creative Suite 3, usually abbreviated CS3 — that work together to provide a full spectrum of Internet solutions. Adobe CS3 comes in many flavors — which include components such as InDesign, PhotoShop, Illustrator, Acrobat Professional, Dreamweaver, Fireworks, Contribute, After Effects Professional, Premiere Pro, Soundbooth, Encore, and even OnLocation. In fact, for a mere $2,500 or so, you can buy the Adobe Create Suite 3 Master Collection and get all of these things in a single (very expensive) box!

Dreamweaver features an easy-to-follow-and-learn GUI so you can style Web pages with CSS without even knowing what a style rule is! Many of the benefits of Dreamweaver stem from its sleek user interface and its respect for clean HTML. You can learn more about Dreamweaver by visiting the Adobe Web site at `www.adobe.com/products/dreamweaver`.

If you're too low on funds for a top-of-the-line WYSIWYG HTML editor like Dreamweaver CS3 (suggested retail price is about $400, but discounts of up to $200 are available), there are other possibilities. You can ponder the suggestions in the next section or go a-searching on the Web (the search string "WYSIWYG HTML editor" should do nicely) to find lots more still!

Contenders

WYSIWYG editors generate allegiances that can seem as pointless as the enmity between owners of Ford and Chevy trucks. All three of the following editors have fans, and all can both produce great Web pages.

- **Adobe GoLive** is a Web page editor that offers text and WYSIWYG editors, along with color coding, automatic code completion, HTML validation, nice site management chops, and bunches more. It lists for $400. Check it out at .

  ```
  www.adobe.com/products/golive
  ```

- **CoffeeCup HTML Editor 2007** is a Windows-based Web package that offers a code editor (text) and a visual editor (WYSIWYG), along with drag'n'drop scripting, support for pre-fab code elements called *snippets*, and a nifty image editor/mapper. It costs only $49. Check it out at

  ```
  www.coffeecup.com/html-editor/
  ```

- **HotDog Pro** is a compact, likeable HTML editor that operates in text and WYSIWYG modes. It, too, supports color coding, an HTML validator, offers lots of interesting image handling features, and even a slick multi-file find and replace/edit toolset. Take a bite at:

  ```
  www.sausage.com/hotdog-professional.html
  ```

Graphics Tools

Graphics applications are beasts. They can do marvelous things, but learning how to use them can be overwhelming at first.

If you aren't artistically inclined, consider paying someone else to do your graphics work. Graphics applications can be pricey and complicated. But you should have some kind of high-function (if not high-end) graphics program to tweak images should you need to. Our highest rating goes to Adobe Photoshop, but considering its cost and the average newbie HTML hacker's budget, we discuss a lower-cost alternative first in the following section.

Photoshop Elements: The amateur champ

At around $100 (with discounts as low as $80), Adobe Photoshop Elements 6.0 is an affordable PC- and Mac-based starter version of the full-blown Photoshop (the gold standard for graphics). You can do almost anything with Photoshop Elements that you might need for beginning and intermediate-level graphics editing.

This product is for you if you want to add images to your site but you don't want to work with graphics all the time, or use fancy special effects . To learn more about Photoshop Elements, visit `www.adobe.com`, select Products⇨All Products, and then select whichever version of Photoshop Elements (PC or Mac) you want to read more about from the drop-down menu.

TIP

If you're really on a tight budget, check out the $90, PC-only Paint Shop Pro Photo X2 at `www.corel.com` instead. It does nearly everything that Photoshop does and costs less than Photoshop Elements.

Professional contenders

If you work with photographs or other high-resolution, high-quality images or artwork, you may need one of these Web graphics tools.

Adobe Photoshop

If it weren't so darned expensive, we'd grant top honors to Photoshop. Alas, $650 is too high for many novices' budgets. Wondering whether to upgrade from Photoshop Elements? Adobe mentions these capabilities among its "Top reasons to upgrade":

- **Improved file browser:** Shows and tells you more about more kinds of graphics files and gives you more-powerful search tools.
- **Shadow/Highlight correction:** Powerful built-in tools add or manipulate shadows and highlights in images.
- **More-powerful color controls:** Color palettes and color-matching tools with detailed controls that Elements lacks.
- **Text on a path:** Full-blown Photoshop lets you define any kind of path graphically and then instructs your text to follow that path. This capability supports fancy layouts that Elements can't match.

If you need to use sophisticated visual effects, edits, or tweaks on high-resolution photorealistic images, full-blown Photoshop is your best bet. For basic Web sites, however, Photoshop Elements is more than up to the task — which is why it's the most popular graphics editing tool.

Like its little brother Photoshop Elements, full-blown Photoshop works with both Macintosh and PC operating systems. The current version is Adobe Photoshop CS3. It's included in all of Adobe's product suites.

Photoshop CS3 add-ons and plug-ins provide specialized functions — such as complex textures or special graphics effects. This extensibility is nice because graphics professionals who need such capabilities can buy them (most cost $100 and up, with $300 a pretty typical price) and add them without muss or fuss. But those who don't need them don't need to pay extra for the base-level software.

Adobe Fireworks

Fireworks is a graphics program designed specifically for Web use, so it offers lots of nice features and functions for that purpose. The current version is Adobe Fireworks CS3.

Fireworks is tightly integrated with other Adobe products and therefore is of potentially great interest if you're using (or considering) Dreamweaver. Simply put, this combination of Adobe products makes it very easy to add graphical spice to Web pages.

For more information about Fireworks and related Macromedia products, check out `www.adobe.com/products/fireworks`.

Link Checkers

A broken link on your site can be embarrassing. To spare your users the dreaded `404 Object Not Found` error message, use a link checker to make sure your links are

REMEMBER

- Correctly formatted before you publish
- Live on the Web after you publish

  Other Web sites may change or disappear after you publish your site. Regularly check your site's links to make sure they still work.

The worst broken link points to a page on your own site.

Many HTML editors and Web servers include built-in local link checkers, and they may even scour the Web to check external links.

HTML Link Validator: The champ

HTML Link Validator 4.47 is a professional-strength tool at an affordable price ($35). We recommend it because it handles many kinds of links and reports clearly and concisely on their condition.

You can find HTML Link Validator all over the Web. A good place to grab it is at `www.download.com` (search the program name to find it immediately).

Contenders

Both of the following programs are pretty good link checkers, They need the application of a little elbow grease to learn and to use, but the price is right: *free*.

W3C Link Checker

This is a utility created by volunteers for the World Wide Web Consortium. You can either

- Download it from `http://validator.w3.org/docs/checklink.html`.

  You have a couple of download options:

  - Grab a compiled version for your computer and operating system and run it as-is.
  - Grab the source code and tweak it for your needs and situation.
- Use the online version at `http://validator.w3.org/checklink`.

LinkScan/QuickCheck

LinkScan offers a real-time single-page link check, and a free evaluation software package that can handle sites with up to 500,000 documents. It creates an annotated, color-coded listing of each HTML or XHTML document it parses, and makes it easy to find broken or suspect links, missing image files, and so forth.

Check it out at `www.elsop.com/quick/`.

HTML Validators

Validation compares a document to a set of document rules — a Document Type Definition (DTD), an XML Schema, or whatever other rules explicitly describe its syntax and structure. Simply put, validation checks the actual markup and content against the rules that govern it and flags any deviations it finds.

Typically, a document author follows this process:

1. **Create an HTML document in an HTML editor.**

   Let's say this step results in a file called `mypage.htm`.
2. **Submit `mypage.htm` to an HTML or XHTML validation site for inspection and validation.**

   If any problems or syntax errors are detected, the validator reports such errors in an annotated version of the original HTML document.
3. **If the validator reports errors, the author corrects those errors and resubmits the document for validation.**

   REMEMBER

   Sometimes, breaking HTML rules is the only way for your page to look right in older Web browsers. But document rules exist for a reason: Nonstandard or incorrect HTML markup often produces odd or unpredictable results.

Browsers usually forgive markup errors. Most browsers identify HTML pages without an `<html>` element. But someday, markup languages may be so complex and precise that browsers won't be able to guess whether you're publishing in HTML or another markup language. Get the markup right from the beginning and save yourself a bunch of trouble later.

TIP

HTML validation is built into many HTML editors.

W3C validator

The W3C has a free, Web-based validation system available at `http://validator.w3.org`. It will provide copious output about what errors or inconsistencies it finds in your documents until you fix them all. It also includes an option for viewing annotated source code so you can see exactly where it's finding items it doesn't like. This is a great tool, that is well worth learning and using.

Built-in validators

Many tools in this chapter offer HTML validation. These include HTML-Kit, HomeSite, Dreamweaver, and BBEdit. Use 'em if you got 'em; get 'em if you don't!

FTP Clients

After you create your Web site on your computer, you have to share it with the world. So you need a tool to transfer your Web pages to your Web server. A very convenient way to accomplish this task is through FTP (*F*ile *T*ransfer *P*rotocol). FTP has been around since the early days of the Internet (way before the Web came along).

After you select a server host and you know how to access a Web server (your service provider should supply you with this information), you must upload your pages to that server. That means you need FTP.

All FTP programs are similar and easy to operate. We recommend these:

- **FileZilla** is a fast, capable, free Open Source FTP program with an intuitive drag'n'drop user interface. It's available online at `http://filezilla-project.org`.
- **Fetch** for the Mac is located at `http://fetchsoftworks.com`.

Swiss Army Knives

Collections of tools can help you manage and control your Web site. They're the Web version of a chunky red knife with a tool for every purpose. We call these *Swiss army knives.* (Metaphorically speaking, of course.)

- **HTML Toolbox,** from NetMechanic, is the sharpest tool in the shed. This puppy has most of the features we recommend in this chapter, including link and spell checking and validation.

  This convenient little package costs you about $60 per year (it's priced per URL, for sites up to 100 pages). If that's not too much to ask, check it out at

  `www.netmechanic.com/maintain.htm`
- **HTML-Kit** supports plug-ins to add functions such as link checks and spelling checks. Most of these plug-ins are free or inexpensive.

  `www.chami.com/html-kit/plugins`
- **Easy HTML Construction Kit** offers a collection of useful conversion, reformatting, and template management tools for a paltry $25. It's at

  `www.hermetic.ch/html.htm`

Chapter 23

Ten Tip-Top Online HTML References

In This Chapter

- Getting the specs for HTML, XHTML, and CSS
- Discovering HTML and XHTML DTD types
- Pointing out XHTML and HTML Character Codes
- Ferreting out favorite HTML resource sites

When the time comes to dig more deeply into markup details, you'll want to know where to turn for information and answers. In this Part of Ten, we point out the master documents that specify HTML, XHTML, and CSS down to the last jot and tittle — namely the W3C Recommendations that govern their contents and structure. We also provide information about the Document Type Definitions, or DTDs, to which such recommendation correspond, provide pointers to information about character codes for use as character entities, and conclude with our very favorite HTML, XHTML, and CSS online resources.

Nothing But the Specs, Please!

The formal documents that describe HTML and XHTML are on the W3C's Web site at `www.w3.org`. That said, markup languages usually include version numbers to identify them specifically and uniquely. The current version of HTML is 4.01. It dates back to December 1997; you can find the document at `www.w3.org/TR/html4`.

XHTML has gone through two major drafts, 1.0 and 1.1, since it first appeared in 2000. The 1.1 version is more advanced than 1.0, but most Web content developers (and, for that matter, software tools) still follow the 1.0 specification anyway. An XHTML 2.0 specification is in "Public Working Draft" status (its authors haven't finalized its content and structure).

TIP

When a W3C specification is finished, it's known as a *W3C Recommendation*.

You can find specifications for all three versions of XHTML as follows:

- XHTML 2.0 Working Draft (7/26/2006)

  `http://www.w3.org/TR/xhtml2/`
- XHTML 1.1 Module-based XHTML Recommendation (5/31/2001)

  `www.w3.org/TR/xhtml11/`
- XHTML 1.0 Recommendation (Second Edition, 8/1/2002)

  `http://www.w3.org/TR/xhtml1/`

The Cascading Style Sheets markup language also falls under the W3C's control. You can find a wealth of information on this subject at its Cascading Style Sheets Home page at `www.w3.org/Style/CSS`. As with XHTML, CSS comes in three versions, called levels 1, 2, and 3; again, as with XHTML, the third version remains a work in progress.

You can find specifications for all three levels of CSS as follows:

- CSS level 1 (1/11/1999)

  `http://www.w3.org/TR/CSS1`
- CSS level 2 revision 1 (aka CSS 2.1, 7/19/2007)

  `http://www.w3.org/TR/CSS21/`
- CSS level 3 (varies): look for Level 3 entries on the CSS Current Work page:

  `http://www.w3.org/Style/CSS/current-work`

The HTML and XHTML DTDs

The HTML and XHTML specifications use *Document Type Definitions* (DTDs) written in the Standard Generalized Markup Language (SGML) — the granddaddy of all markup — to define the details.

In its earlier versions, HTML used elements for formatting; over time, developers realized that

- Formatting needed its own language (known as Cascading Style Sheets, or CSS).
- HTML elements should describe only a page's structure, not its appearance or display characteristics.

This resulted in three flavors of HTML, which also apply to XHTML as well. This explains why each of the following list items employs the (X)HTML designation. In reality, of course, there are a total of six DTDs)to which we provide links in Table 22-1, which follows our list):

- **(X)HTML Transitional**: Uses HTML's elements to describe font faces and page colors. XHTML Transitional accounts for formatting elements in older versions of HTML. Formatting elements in XHTML Transitional are *deprecated* (considered obsolete) because the W3C would like developers to move away from them and to a combination of XHTML Strict and CSS. We use the XHTML Transitional DTD for the markup in this book.
- **(X)HTML Strict**: Doesn't include any elements that describe formatting. This version is designed to let CSS drive the page formatting. The CSS-with-XHTML Strict approach is an ambitious way to build Web pages, but in practice it has its pros and cons. CSS provides more control over your page formatting, but creating style sheets that work well in all browsers can be tricky. (Chapter 9 covers style sheets and the issues around using them in more detail.)
- **(X)HTML Frameset**: Includes *frames* — markup that allows you to display more than one Web page or resource at a time in the same browser window. Frames are still used in some Web sites but are less popular today than they were in the late 1990s. Our advice is to use them only if you *must* display information from multiple HTML documents at the same time in a single browser window.

All Web browsers support all elements in HTML Transitional (and in XHTML 1.0 Transitional, if proper tag formatting is used); you can choose to use elements from it or stick with (X)HTML Strict instead. If you use frames, then technically you have to work with (X)HTML Frameset, but all elements still work the same way. This book covers all (X)HTML tags in all versions (lumping them into one category called (X)HTML) because all real-world Web browsers support all three flavors.

REMEMBER

Any properly constructed HTML or XHTML document must reference a DTD in its first line of text. Simply put, that means you'll use one of the entries from the Markup column in Table 23-1 to start any of your coding efforts!

Table 23-1 Where the HTML and XHTML DTDs Are

Type	*Name*	*Markup*
HTML	Transitional	`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">`
HTML	Strict	`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">`
HTML	Frameset	`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">`
XHTML	Transitional	`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">`
XHTML	Strict	`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">`
XHTML	Frameset	`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frame-set.dtd">`

Character Codes Come In Many Flavors

When it comes to reproducing odd or unusual characters in HTML or XHTML documents, a strange coding technique is used to represent such so-called "character entities." These generally take the form `&amp;` (for ampersand, for instance) where the ampersand symbol (&) indicates the character entity is starting and everything from there up to the semi-colon (;) is a symbolic or numeric character code. It just so happens that `&amp;` and `&38;` both reference the same thing: the character code that represents an ampersand character on your computer display.

Table 23-2 includes pointers to a whole slew of different character codes you can use in XHTML documents without reservations, but must use carefully in HTML documents. (XML is a lot smarter about Unicode characters, which include nearly every printable glyph known to man.) For HTML, the ISO-Latin-1 character set (and its regionalized variants) are safe, but you should experiment carefully to see what you can and can't use from Unicode in your

HTML documents by testing with lots of different Web browsers to see what works and what doesn't.

Table 23-2 Online Pointers to (X)HTML Character Codes

Name	*URL*
Unicode Code Charts	`www.unicode.org/charts/`
ISO-Latin-1 character set	`www.htmlhelp.com/reference/charset/`
Greek characters	`www.unicode.org/charts/PDF/U0370.pdf`
Currency symbols	`www.unicode.org/charts/PDF/U20A0.pdf`
Miscellaneous symbols	`www.unicode.org/charts/PDF/U2600.pdf`
Arrow characters	`www.unicode.org/charts/PDF/U27F0.pdf` `www.unicode.org/charts/PDF/U2900.pdf`
Mathematical characters	Search **math** at `www.unicode.org/charts/` (there are six different, relevant code charts)
General punctuation	`www.unicode.org/charts/PDF/U2000.pdf`

Deprecated (X)HTML Elements and Attributes

In markup terminology, elements or attributes may be *deprecated*. This means they're still recognized but doomed to obsolescence. If you see (X)HTML markup you don't recognize or can't find elsewhere in this book, chances are good that it's deprecated. (Note: XHTML doesn't recognize deprecated items if you use the Strict DTD, but XHTML Transitional and Frameset DTDs do recognize them.)

You can find good information about deprecated (X)HTML online, too. Here are a few of our favorites:

- HTML 4 Deprecated Features (Web Design Group/htmlhelp.com):

  `http://htmlhelp.com/reference/html40/deprecated.html`
- Deprecated Tags in HTML 4.0 (HTML Goodies):

  `http://www.htmlgoodies.com/tutorials/html_401/html4-ref/article.php/3460291`
- Tags and Elements Deprecated in XHTML 1.0 (About.com):

  `http://webdesign.about.com/od/htmltags/a/bltags_deprctag.htm`
- Deprecated Elements in HTML (html-reference.com):

  `http://www.html-reference.com/depreciated.htm`

Table 23-3 lists deprecated (X)HTML elements, whereas Table 23-4 lists deprecated (X)HTML attributes (in alphabetical order, for easy reference).

Table 23-3 Deprecated (X)HTML Elements

Element	*Common Name*	*Empty?*	*Category*	*Description*
`applet`	Applet	No	Inclusion	Includes Java applet in (X)HTML document
`b`	Bold	No	Presentation	Turns text bold (use `<strong>` instead)
`base-font`	Base font	Yes	Presentation	Sets default font for text to which no style sheet or `font` element applies
`center`	Center text	No	Presentation	Centers enclosed text in current display area
`dir`	Directory list	No	List	Lists style for lists of short strings (such as file names)
`font`	Font info	Yes	Presentation	Sets size, font, and color for element content
`i`	Italic	No	Presentation	Turns text italic (use `<em>` instead)
`isin-dex`	Single-line input	Yes	Form-related	Prompts user for single line of input
`menu`	Menu list	No	List	Creates compact list format
`param`	Object parameters	Yes	Inclusion	Passes "command-line" input to Java applet

Element	*Common Name*	*Empty?*	*Category*	*Description*
`s`	Strikethrough	No	Presentation	Uses strikethrough font for element content
`strike`	Strikethrough	No	Presentation	Uses strikethrough font for element content
`u`	Underline	No	Presentation	Uses underline font for element content

Table 23-4 Deprecated (X)HTML Attributes

Name	*Where Deprecated*	*Description*
`align`	`<caption><img><table><hr><div><h1..6><p>`	Sets alignment at `top`, `bottom`, `left`, `right`
`alink`	`<body>`	Sets color for active document links
`back-ground`	`<body>`	Sets background picture for document body (URL is target)
`bgcolor`	`<body><table><tr><td><th>`	Sets background color for document body
`border`	`<img><object>`	Sets width of border around image
`clear`	`<br>`	Sets side of a line break on which floating objects may not be positioned
`color`	`<basefont><font>`	Sets color for `basefont` (default) or `font` element content
`compact`	`<ol><ul>`	Special compact formatting for list elements
`hspace`	`<img><object>`	Sets horizontal margin around an image or object

(continued)

Table 23-4 *(continued)*

Name	*Where Deprecated*	*Description*
`link`	`<body>`	Sets defualt color for document links
`noshade`	`<hr>`	Instructs browser to draw horizontal rules without 3-D shading
`nowrap`	`<td><th>`	Instructs browser not to perform word wrap
`size`	`<basefont><font><hr>`	Sets size for `<basefont>` or `<font>` from 1 to 7, `<hr>` in pixels
`start`	`<ol>`	Sets starting number for ordered list
`text`	`<body>`	Sets text (foreground) color for document body
`type`	`<li>`	Sets list style (`1`|`a`|`A`|`i`|`I` for ordered lists, `disc`|`circle`|`square` for unordered lists)
`value`	`<li>`	Sets the value for a list item, specified by number
`vlink`	`<body>`	Sets color for document links already visited
`vspace`	`<img><object>`	Sets vertical margin for an image
`width`	`<hr><pre><td><th>`	Sets width (percentage or pixels) for object sizing or spacing

Magnificent HTML Resource Sites

Though there's no shortage of good HTML resources online, some are so simply superlative that we have to call your attention to them here, as we list our personal Top 10 favorites. In no particular order, here they are:

- The World Wide Web Consortium is not just the fountain from whence HTML, XML, XHTML, CSS, and a whole lot more springs, it's also a great source of information, tutorials, and tools. Spend some time rooting around at `www.w3.org` and you won't be disappointed.

- The Web Design Group's motto is "…Making the Web accessible to all," but they've also got great references and tools, as well as help forums, FAQs galore, and some great design guides and color code info. Explore this site at `www.htmlhelp.com`.
- W3 Schools uses "the best things in life are free" as its tag line, and it's got lots of free and useful stuff to prove that point — including HTML and XML tutorials that cover the basics of markup, plus CSS, tons of XML applications, numerous scripting languages for client- and server-side scripts, and a whole lot more. See what there is to learn and do at `www.w3schools.com`.
- Once upon a time, Webmonkey was part of *Wired* magazine. It's always been a great resource for Web developers, but now it's on its own. Dig into this site for an extensive how-to library that covers lots of interesting topics in (for openers) Web authoring, design, multimedia, and e-business, as well as quick references for everything from JavaScript to special characters and colors codes. You can check it out at `www.webmonkey.com`.
- HTML Goodies bills itself as "the ultimate html resource." We're not sure what that really means, or how to tell if that claim is really true. But we are sure you can find a plethora of pleasing and useful HTML help, information, and (yes) goodies, so why not root around a while and see what you can find at `www.htmlgoodies.com`.
- Zvon's tag line is that it's "The Guide to the XML Galaxy" and indeed we'd be remiss if we didn't acknowledge its bias toward XML and corresponding lack of HTML coverage. Nevertheless, its CSS and DTD tutorials alone make it worth visiting and learning from, and should you ever take the XML plunge we guarantee you'll return time and time again to `www.zvon.org`.
- WebCom.com calls itself the "comprehensive resource for publishing on the World Wide Web." As such, it's mostly a clearinghouse for other good stuff by way of being a good hosting provider. But it's a pretty good clearinghouse at that, and most readers of this book will benefit from scoping out its Web Primers, HTML Guides (especially "WebCom's Own Guide to HTML"), and Publishing Guides, if not other information silos at `www.webcom.com/html`.
- The Web Developer's Virtual Library (WDVL) is a treasure trove of information, tutorials, tools coverage, and Web design and programming techniques out the ying-yang. This is a site to spend some serious time on — in fact, you may find yourself spending more time there than is good for you. Attack it with a specific agenda (unless you have more time than you know what to do with) at `http://wdvl.com`.

- WebReference.com offers both oodles and scads of information about HTML, XHTML, CSS, and everything involved in designing, creating, and maintaining quality Web sites. ***Hint:*** Use the Sitemap link to get a sense of what's there, or you might not really appreciate the many great items you can find there. Look it over at `www.webreference.com`.
- John December is a long time Web maven who's written and researched Web topics since the early 1990s. He has an HTML area at his Web site (`www.december.com/html`) that offers a useful introductory course to HTML, pointers to key specifications, color code and coding information, plus lots of markup examples and pointers to help budding Web designers and developers tune up their skills. It's definitely worth repeat visits.

Of course, now that we've given you ten of these top sites, we feel we've only managed to get you started. As you create your own set of favorites and find the tools you like best, you'll also establish your own set as well. Enjoy!

Index

Symbols

& (ampersand)
  beginning entities, 21
  displaying on Web page, 22–23
* (asterisk) for JavaScript comments, 229
: (colon) before CSS pseudo-classes, 162
, (comma) in CSS declarations, 128
{ } (curly braces) for CSS markup, 13
= (equals sign)
  reading and writing array elements, 242
  setting JavaScript variables, 230
! (exclamation mark) beginning comments (<!--), 23
> (greater-than sign)
  displaying on Web page, 22–23
  ending comments (-->), 23
  for HTML tags, 12
- (hyphen) instead of spaces in filenames, 35
`@import` statement, 135
< (less-than sign)
  beginning comments (<!--), 23
  displaying on Web page, 22–23
  for HTML tags, 12
`&nbsp` (non-breaking space) character, 204
. (period)
  accessing object properties or methods, 243
  after CSS selector, 129–130
  in relative link notation (../), 94–95
" (quotation marks) for attribute values, 20
; (semicolon)
  in CSS declarations, 128
  ending entities, 21
  separating JavaScript statements, 229
/ (slash)
  closing empty elements, 19
  in HTML versus XHTML, 19
  for JavaScript comments, 229
  in relative link notation (../), 94–95
[ ] (square brackets) as array operator, 242
_ (underscore) instead of spaces in filenames, 35

A

`<a>` element. *See also* links
  for absolute links, 93
  CSS styles for, 145–146, 161–163, 165
  `href` attribute, 91, 93, 94, 99, 100, 101, 102
  with image triggering links, 115–116
  for named locations, 98–100
  opening page in new window, 97
  overview, 91
  for relative links, 94–95
  `target` attribute, 97
`abbr` attribute for tables, 210
About Me page
  adding fonts and colors, 295–297
  audience analysis for, 292
  components, 292
  design considerations, 291–292
  Ed Tittel's page, 26–28, 298
  markup, 292–297
  more complex version, 294
  simple version, 292–293
About.com image map tutorial, 118
absolute links, 93
absolute property measurement values, 124
accessibility
  alternative text for images, 108–109, 116
  CSS aural styles for, 153
  site map benefits for, 46
  UI design resources, 56
  W3C's Web Accessibility Initiative (WAI), 109
accessing arrays, 242
`action` attribute (`<form>` element), 251
`active` pseudo-class (CSS), 162–163
Add to Cart button, 324–325
Adobe Fireworks, 353
Adobe GoLive WYSIWYG editor, 351
Adobe Photoshop, 352–353
Adobe Photoshop Elements, 352
`alert()` method (JavaScript), 224
`align` attribute
  for centering tables, 198
  `<hr>` element, 80
  `<img />` element, 113–114
  with other table elements, 197
  `<p>` element, 71–72
  `<table>` or `<tr>` or `<td>` element, 195–196

aligning
  horizontal rules, 80
  images, 113–114
  paragraphs, 71–72
  positioning elements with CSS, 141–142
  tables, 195–198
  text blocks using CSS, 168–169
alt attribute
  <area /> element, 117
  <img /> element, 108
alternative text (alt text) for images, 108–109, 116
American Standard Code for Information Interchange (ASCII), 21
ampersand (&)
  beginning entities, 21
  displaying on Web page, 22–23
anchor element. *See* <a> element
angle brackets. *See* greater-than sign (>); less-than sign (<)
animation, 302, 333
applets, 214
<area /> element, 117
arithmetic operators (JavaScript), 232
arrays (JavaScript), 241–242
ASCII (American Standard Code for Information Interchange), 21
assignment operators (JavaScript), 232
asterisk (*) for JavaScript comments, 229
attributes. *See also specific elements and attributes*
  defined, 17
  deprecated, 361, 363–364
  forms-related, 269–270
  JavaScript-related, 246
  measurement values, 124
  overview, 20–21
  table-related, 210
auction page. *See* eBay Auction page
aural styles (CSS), 153
axis attribute for tables, 210

• B •

<b> element, 170
background
  for auction sites, 303
  background or background-color property (CSS), 161, 163–164, 209
  for table cells, 209
BBEdit HTML editor, 349
beta-testing, 343
bgcolor element for table cells, 208–209
blinking text, 173–174
block quotes, 74–75
<blockquote> element, 74
blocks of text. *See* text blocks
blue color coding in this book, 17
<body> element. *See also* text
  cookie script in, 286
  CSS style for, 142–144
  described, 34
  overview, 67
  scripts in, 224
  setting CSS styles in, 132
body scripts, 224
boldface, applying to text, 170–171
border attribute
  <img /> element, 112
  <table> element, 186
border property (CSS), 189
border-bottom property (CSS), 189
border-color property (CSS), 188
border-left property (CSS), 189
border-right property (CSS), 189
borders
  for images, 112–113
  for tables, 178, 186–189
  using small image to create, 111
border-spacing property (CSS), 194–195
border-style property (CSS), 187–188
border-top property (CSS), 189
border-width property (CSS), 188
box model (CSS), 189
boxes, creating with small image, 110, 111
<br /> (break) element
  described, 25
  for empty table cells, 194
  formatting, 77–78
  “Hello World!” Letter example, 34
broken links
  avoiding, 341–342
  fixing, 95–96
  link checkers, 353–354
  to page on your own site, 354
browsers
  auction page considerations, 301, 303
  avoiding dependencies, 335–336
  block quotes display in, 75
  bulleted lists display in, 85–86
  checking editing in, 37–39
  color display by, 159
  CSS disabled by some, 128

CSS information for, 122
CSS overriding display rules of, 126–127
CSS style sheets for old and new, 147
definition lists display in, 88–89
differences among, 14
headings display in, 72–74
image rollover problems with, 274
image scaling by, 111–112
link display by, 92
list online, 14
numbered lists display in, 84
overview, 13
posting files to server via, 40, 55
previewing Web pages in, 29–30, 36–37
testing on multiple, 335–336
text rollover display in, 276
text-only, 13
Builder.com, 105
Bulger, Brad (*MySQL/PHP Database Applications*), 323
bulleted lists
browser display of, 85–86
bullet styles for, 86
CSS for, 87
defined, 84
elements for, 20
formatting, 85
nesting, 89–90
uses for, 84
button generator online, 272

• C •

calling (JavaScript)
functions, 240–241
methods, 243
canned JavaScript, limitations of, 223
`<caption>` element, 185
case
CSS sensitivity to, 3
HTML not sensitive to, 3
JavaScript sensitivity to, 228, 230
link URLs sensitivity to, 95
setting with CSS, 172–173
XHTML sensitivity to, 3
cell spans
cell width changes versus, 178
`colspan` attribute for, 184, 198–200
defined, 178
need for, 184
planning needed for, 198
`rowspan` attribute for, 198, 200–201
`cellpadding` attribute, 192–194
cells (table). *See also* cell spans
color for, 208–209
creating with `<td>` element, 183–184
defined, 178
empty, for controlling layout, 194
padding and spacing for, 192–195
populating, 201–204
width adjustments, 189–192
`cellspacing` attribute, 192–194
`<center>` element, deprecated, 198
centering tables, 197–198
CGI (Common Gateway Interface)
for e-mail address information, 315
for form validation, 264
further information, 264, 265
online sources for form handlers, 266
CGI Resource Index site, 266
Chapman, Steven (“Hiding Your Email Address”), 102
`char` attribute for tables, 210
character codes, 22, 360–361
character spacing using CSS, 168
`charoff` attribute for tables, 210
check boxes, 255–256
`checked` attribute (`<input />` element), 256
Citixxperience site map, 45
`class` attribute (CSS), 129–131, 167
client-side scripts
for cookies, 284–286
for displaying dynamic content, 278–281
for image rollovers, 272–275
`<object>` element for embedding content, 287–288
for pop-up windows, 281–284
for text rollovers, 275–277
clock examples
adding to page, 279, 281
CSS styles for, 281
HTML and JavaScript, 278–279
JavaScript and DOM, 280–281
pop-up window, 282–283
closing tags
keeping track of, 334
for nested lists, 90
for `<p>` element, 71
CoffeeCup HTML Editor 2007, 351
`<col>` element, 185, 197
`<colgroup>` element, 185, 197
colon (:) before CSS pseudo-classes, 162

`color` property (CSS)
  hexadecimal color codes with, 159
  for link colors, 162, 163
  overview, 161
  for text colors, 161
colors
  background color definitions (CSS), 163–164
  browser display of, 159
  for coding in this book, 17
  design considerations, 44
  hexadecimal codes for, 158, 159–160
  for links using CSS, 161–163
  named values in (X)HTML, 158–159
  RGB values for, 160
  site-wide style sheet for, 310, 316–318
  for table borders using CSS, 188, 189
  for table cells, 208–209
  for text using CSS, 161
  tips for using, 296
  Web-safe, 159
`cols` attribute (`<textarea>` element), 261
`colspan` attribute for tables, 184, 198–200
comma (,) in CSS declarations, 128
comments
  JavaScript, 229
  (X)HTML, 23
Common Gateway Interface. *See* CGI
company site
  basic elements, 308–309
  Contact Us page, 308, 314–315
  customizing templates for your use, 311, 313
  design considerations, 307–308
  giving visitors what they need, 318
  home page, 308, 309–311
  logo, 309
  press page, 309
  products/services pages, 308
  site-wide style sheet, 309
  style sheet, 316–318
  Web Design page, 312–314
comparison operators (JavaScript), 233–234
concatenation, 231, 234
conditional statements (JavaScript), 235–236
`confirm()` method (JavaScript), 236
Conquest Media example
  Contact Us page, 314–315
  customizing templates for your use, 311, 313
  home page, 309–311
  product catalog, 321–328
  site-wide style sheet, 316–318
  Web Design page, 312–314
Contact Us page, 308, 314–315
content
  arranging dynamically, 215–217
  dynamic, displaying, 278–281
  embedding with DHTML, 287–288
  importance of, 331
  keeping fresh and interesting, 336–337, 341, 342
  keeping paramount, 332
  structuring documents and site for, 332
`content` attribute (`<meta />` element), 63–64
cookies, 284–286
`coords` attribute (`<area />` element), 117
copyright issues for images, 106
counting operators (JavaScript), 232–233
creating a Web page
  applications needed, 29–30
  editing your page, 37–39
  planning a simple design, 31–32
  posting your page, 39–40
  saving your page, 34–35
  steps, 30
  viewing your page, 36–37
  word processors not useful for, 10, 30
  writing HTML, 32–33
CSS (Cascading Style Sheets). *See also* font specifications using (CSS); *specific properties*
  advantages of, 121–122, 157
  aural styles for accessibility, 153
  background color definitions, 161, 163–164
  benefits of, 25
  box model, 189
  browser display rules overridden by, 126–127
  for bulleted lists, 87
  cascading nature of, 122, 136
  `class` attribute for styles, 129–131
  for clock, 281
  color coding in this book, 17
  color definition properties, 161
  conventions in this book, 2–3
  CSS2.1 specification, 123
  CSS3 specification, 123
  declarations, 126, 128
  developing specific styles, 138–146
  in DHTML, 271
  documentation in iPod format, 153
  examples online, 1
  external style sheets, 134–135
  externalizing style sheets, 146–147
  further information, 125

goal of, 121
guidelines for using, 137
for headings, 74
for horizontal rules, 80
inheritance, 131–132
inline styles, 133, 135
internal style sheets, 133–134
knowledge required for, 125
link color definitions, 161–163
media types and their uses, 148
for old and new browsers, 147
overlapping objects, 141–142
paged media styles, 153–155
positioning elements using, 141–142
positioning text using, 168–170
precedence for style rules, 136, 163
print style sheet, 154–155
property measurement values, 124
pseudo-classes, 161–163
selectors, 126, 128
setting rules for entire document, 132
shorthand properties, 140–141
site-wide style sheet, 309, 316–318
specifications for, 358
steps for using, 123
style rules, 126
style switching with JavaScript, 215–217
syntax, 126
for table alignment, 197–198
for table borders, 187–189
for table cell padding and spacing, 194–195
table formatting considerations, 176
table `width` property, 192
testing pages for browsers disabling, 128
text color changes using, 161
for text rollovers, 275–278
uses for, 124–125
visual layouts using, 139–141
visual media styles, 148–153
for white space control, 76
W3C Core style sheets, 135
W3C validation service, 129

"CSS Structure and Rules" tutorial (Web Design Group), 131
CSS Web Design For Dummies (Mansfield), 125
curly braces ({ }) for CSS markup, 13
CuteFTP software, 55
Cyberduck FTP client, 55

• D •

dark green color coding in this book, 17
data types (JavaScript), 231
data-collection forms. *See also* forms
defined, 247
overview, 249–250
validating, 219–220, 264
`<dd>` element, 87, 88
December, John (Web maven with site), 366
declarations
CSS, 126, 128
object (JavaScript), 246
variable (JavaScript), 230–231
`declare` attribute for JavaScript objects, 246
`default.html` page, 308. *See also* home page
`defer` attribute (JavaScript), 246
definition lists
browser display of, 88–89
CSS for, 88
defined, 87
elements for, 87
formatting, 87–88
resources online, 89
designing. *See also* planning Web pages
About Me page, 291–292
building attractive pages, 333–334
color considerations, 44
company site, 307–308
following document syntax and layout rules, 335
greeked text for mock-ups, 311
keeping simple, 333
planning a simple design, 31–32
product catalog, 320
professional help for, 44
standard page layout for, 334
structuring your site, 332
top-down page design, 332
UI design resources, 56
user-friendly forms, 267–268
DHTML (Dynamic HTML), 271. *See also* client-side scripts
Dictionary of HTML META Tags online, 66
disabilities. *See* accessibility
`<div>` element
for centering tables, 198
for text rollovers, 277
for white space, 76
`<dl>` element, 87
`DOCTYPE` declaration, 60–61
document structure
basic for every (X)HTML page, 59–60
`<body>` element, 67
browser display of elements, 67–68
`<head>` element, 62–66
HTML `DOCTYPE` declaration, 60

document structure *(continued)*
  <html> element, 61
  <meta /> element, 63–66
  redirecting users to another page, 65–66
  XHTML DOCTYPE declaration, 60–61
  XHTML namespace, 61–62
document tree, 162
Document Type Definitions (DTDs), 358–360
DOM (Document Object Model)
  clock using, 280–281
  in DHTML, 271
  further information, 245
  overview, 245–246
domain name
  deciding on need for, 53–54
  registering with hosting provider, 54
  relative links not affected by changing, 94
  URL component, 15
download links for PDF files, 100–102
Dreamweaver WYSIWYG editor, 350–351
drop-down lists, 258–260
<dt> element, 87
DTDs (Document Type Definitions), 358–360
Dynamic HTML (DHTML), 271. *See also* client-side scripts
dynamic pages. *See also* JavaScript
  arranging content dynamically, 215–217
  cookies for, 284–286
  defined, 214
  displaying dynamic content, 278–281
  form validation, 219–220, 264
  pop-up windows, 96–97, 218–219, 281–284
  rollovers for, 272–277
  search term example, 214–215

• E •

Easy HTML Construction Kit, 356
eBay Auction page
  background for, 303
  browser considerations, 301, 303
  designing, 300–302
  importance of appearance, 299
  presentation issues, 303
  scripts not allowed for, 301
  tips for writing text for, 302
  using template for auction items, 303–305
E-Commerce For Dummies (Wiley publication), 52
editing Web pages, 37–39
elements. *See also specific elements*
  defined, 13, 17
  deprecated, 361–363
  empty, 18–19
  keeping track of tags, 334–335
  nesting, 20
  overview, 18–20
  single tags, 18–19
  tag pairs, 18–19
  well-structured (X)HTML requirements, 23–24
em property measurement values, 124
e-mail
  hiding addresses from spammers, 102, 315
  links, 102
  receiving form data by, 266–267
empty elements
  defined, 18–19
  <img /> as, 106
end tags, 18
entities
  defined, 17, 21
  non-ASCII characters, 21–22
  overview, 21–23
  tag characters, 22–23
  (X)HTML character codes, 22
equals sign (=)
  reading and writing array elements, 242
  setting JavaScript variables, 230
events (JavaScript)
  defined, 243
  handlers, 244–245, 274
  overview, 243–244
evolutionary changes to site, 337
ex property measurement values, 124
"Examine Graphic Channels and Space" (Builder.com), 105
exclamation mark (!) beginning comments (<!--), 23
expression statements (JavaScript), 234
expressions (JavaScript), 231. *See also* operators (JavaScript)
eXtensible Markup Language (XML), 16, 17
external style sheets (CSS)
  importing, 135
  linking to page, 134–135
  turning internal style sheets into, 146–147

• F •

FAQ page, planning for, 46
feedback from users, 345–346
Fetch FTP client, 55, 356
file formats for images, 104–105
File Transfer Protocol. *See* FTP

filenames
  in broken links, fixing, 95
  guidelines for, 34–35
  spaces in, avoiding, 35
  suffixes for HTML files, 35
  URL component, 15
file-upload fields, 257–258
FileZilla FTP client, 356
Fireworks (Adobe), 353
FirstGov Web site, 11
`focus` pseudo-class (CSS), 162–163
folders, creating for Web pages, 34
`font` property (CSS), 174
font specifications using (CSS)
  for body text, 142–144
  changing dynamically with JavaScript, 215–217
  character spacing, 168
  font families to include in declarations, 165
  `font` property for, 174
  `font-family` property, 129, 142, 144, 164–165
  for headings, 144–145, 164–165
  for hyperlinks, 145–146, 165
  line height, 167
  properties for, 164
  site-wide style sheet for, 310, 316–318
  size, 166
  text treatments, 170–174
  tips for choosing fonts, 296, 302
`font-size` property (CSS), 166
`font-style` property (CSS), 172
`font-weight` property (CSS), 170–171
`footer` ID with tables, 204
`for` loops (JavaScript), 237–238, 242
forcing page refresh, 342
`<form>` element, 251–252
form handlers
  CGI scripts for, 265–266
  defined, 251
  ISP support for, 266
  online sources of, 266
  prerequisites for creating, 264–265
  receiving data by e-mail, 266–267
forms. *See also* data-collection forms; input controls for forms
  breaking into sections, 267
  checking hosting support for, 265
  creating, 251–252
  cues for users, 267
  data as driver for, 247
  designing to be user-friendly, 267–268
  gateway page for, 268
  marking required fields, 268
  markup, 252
  search, 247, 248–249
  uses for, 247
  validating, 219–220, 264
  (X)HTML attributes related to, 269–270
`frame` attribute for tables, 210
Frameset `DOCTYPE` declaration, 60, 61
FTP (File Transfer Protocol)
  client software, 55, 356
  posting files to server via, 40, 54–55
functions (JavaScript), 239–241

• G •

gateway page for forms, 268
gateway sites, 11
ghost sites, 338
GIF (Graphics Interchange Format), 105
Gmail interface, 222
goals for site, 42–43
Gourley, David (HTTP expert), 16
graphics. *See* images
graphics tools. *See* image-editing tools
greater-than sign (>)
  displaying on Web page, 22–23
  ending comments (`-->`), 23
  for HTML tags, 12
greeked text, 311
green color coding in this book, 17
Greenspan, Jay (*MySQL/PHP Database Applications*), 323

• H •

`h` elements (`h1` through `h6`). *See* headings
handicaps. *See* accessibility
hard returns
  ignored in block elements, 75
  in preformatted text, 76
`<head>` element
  clock code in, 279, 281
  described, 34, 62
  forcing page refresh, 342
  header section overview, 24
  `<meta />` element in, 63–66
  metadata in, 63–65
  placement of, 62
  scripts in, 224
  text rollovers in, 277
  `<title>` element in, 34, 62–63
header scripts, 224–225
header section. *See* `<head>` element

headings
  browser display of, 72–74
  CSS styles for, 74, 144–145, 164–165
  elements for, 24, 72
  font sizes for, 74
  formatting, 72
  overview, 24
  with tables, 204
  uses for, 72
height attribute (<img /> element), 110–111
"Hello World!" Letter example
  code listing, 33
  editing the page, 37–39
  planning, 31–32
  posting the page, 37–39
  saving the page, 34–35
  viewing the page, 36–37
  writing HTML, 32–33
hexadecimal color codes, 158, 159–160
hidden fields, 257
"Hiding Your Email Address" (Chapman), 102
Holden, Greg (*Starting an Online Business For Dummies*), 323
home page
  company site, 308, 309–311
  customizing template for your page, 311
  providing access from all areas of site, 49
home.html page, 308. *See also* home page
HomeSite HTML editor, 348–349
h1 through h6 elements. *See* headings
horizontal rules
  attributes, 79–80
  CSS for, 80
  formatting, 79
  <hr> element for, 25, 78–80
  uses for, 78, 80–81
hosted e-commerce services, 323–324
hosting your site. *See also* Web servers
  checking support for forms, 265
  costs of, 53
  domain name for, 53–54
  finding a provider, 39
  host, defined, 51
  hosting provider for, 53
  yourself, 52–53
HotDog Pro WYSIWYG editor, 351
hover pseudo-class (CSS), 162–163
<hr /> element, 25, 79–80. *See also* horizontal rules
href attribute. *See also* href attribute (<a> element)
  <area /> element, 117
  <link> element, 135
href attribute (<a> element)
  for absolute links, 93
  for file downloads, 101
  for intradocument links, 99
  for mailto: links, 102
  for named locations, 99, 100
  overview, 91
  for relative links, 94
hspace attribute (<img /> element), 114–115
.htm or .html suffix, 35
HTML editors
  helpers, 348–350
  previewing capabilities of, 29–30
  text editors versus, 30
  WYSIWYG, 350–351
<html> element, 34, 61
HTML Goodies site, 118, 365
HTML (Hypertext Markup Language). *See also* XHTML; (X)HTML
  as basis for Web pages, 10
  clock using, 278–279
  color coding in this book, 17
  conventions in this book, 2–3
  deprecated elements and attributes, 121, 157, 361–364
  in DHTML, 271
  DOCTYPE declaration, 60
  DTDs, 358–360
  importance of understanding, 334
  measurement values, 124
  resources online, 364–366
  specifications for 4.01, 19
  validating, 344
  XHTML versus, 16–17, 19
HTML Link Validator, 354
HTML toolbox
  collections, 356
  finding good buys, 347
  FTP clients, 356
  graphics tools, 351–353
  HTML editors, 29–30, 348–351
  HTML validators, 355
  link checkers, 353–354
HTML Toolbox (NetMechanic), 356
HTML-Kit tools, 349–350, 355, 356
HTTP (Hypertext Transfer Protocol), 16
http:/, required for link URLs, 96
http-equiv attribute, 65–66
hyperlinks. *See* links
hypertext, 10–11
Hypertext Markup Language. *See* HTML
hyphen (-) instead of spaces in filenames, 35

• I •

icons in margins of this book, 6
if statements (JavaScript), 235
if/else statements (JavaScript), 235–236
image maps, 117–118
image rollovers
  adding to page, 273–274
  button generator for, 272
  code listings, 273, 274–275
  preloader enhancement for, 274–275
  states, 272, 273
  text rollovers versus, 275
image-editing tools
  creating borders using, 113
  finding hexadecimal color codes using, 160
  overview, 351–353
images. *See also* <img /> element
  alignment, 113–114
  alternative text for, 108–109, 116
  for auction sites, 300, 302
  borders for, 112–113
  copyright issues, 106
  file format information online, 105
  file formats for, 104–105
  graphics tools, 351–353
  image maps, 116–118
  as links, 115–118
  logo for company site, 309
  off-site, avoiding direct links to, 50
  optimization tutorials, 104
  in pop-up windows, JavaScript for, 218–219
  relative links for, 106
  role in Web pages, 103–104
  rollovers, 272–275
  scaling by browsers, 111–112
  site map for alternative accessibility, 46
  size specification for, 110–112
  spacers for empty table cells, 194
  spacing for, 114–115
  storage location for, 106
  turning small image into lines and boxes, 110, 111
  Web-friendly, creating, 104–105
  wise use of, 333
<img /> element
  align attribute, 113–114
  alt attribute, 108
  border attribute, 112
  described, 19
  as empty element, 106
  height attribute, 110–111
  hspace attribute, 114–115
  image maps using, 117–118
  for links in anchor tag, 115–116, 117
  markup example, 107
  src attribute, 20, 107
  triggering links, 115–116
  vspace attribute, 114–115
  width attribute, 110–111
@import statement, 135
importing external style sheets (CSS), 135
indenting paragraphs using CSS, 169–170
index.html page, 308. *See also* home page
inheritance (CSS), 131–132
inline elements, 71, 93. *See also specific elements*
inline event handlers (JavaScript), 245
inline styles (CSS), 133, 135
<input /> element
  for check boxes, 255–256
  checked attribute, 256
  for file-upload fields, 257
  for hidden fields, 257
  maxlength attribute, 254
  name attribute, 253, 255
  for password fields, 254
  for radio buttons, 255–256
  for Reset buttons, 262, 263
  size attribute, 254
  src attribute, 263
  for Submit buttons, 262, 263
  for text fields, 253–254
  type attribute, 253, 254, 257, 262, 263
  value attribute, 255, 262, 263
input controls for forms
  check boxes, 255–256
  drop-down lists, 258–260
  file-upload fields, 257–258
  hidden fields, 257
  multi-line text boxes, 261
  overview, 219–220
  password fields, 254–255
  radio buttons, 255–256
  Reset button, 261–263
  Submit button, 261–263
  text fields, 253–254
internal style sheets (CSS)
  adding to page, 133–134
  turning into external style sheets, 146–147
Internet connection for Web server, 52
Internet resources
  About Me page by Ed Tittel, 298
  browser color display test page, 159
  browsers list, 14
  button generator, 272
  CGI information, 264, 265

Internet resources *(continued)*
character codes, 361
Citixxperience site, 45
companion site for this book, 1
CSE HTML Validator, Lite Edition, 340
CSS documentation in iPod format, 153
CSS reference, 125
CSS specifications, 358
"CSS Structure and Rules" tutorial, 131
CSS style sheet validation service, 129
definition lists discussion, 89
deprecated elements and attributes, 361
Dictionary of HTML META Tags, 66
DOM specification, 245
DTDs, 360
For Dummies site, 47
FirstGov site, 11
form handling scripts, 266
FTP client software, 55, 356
ghost sites, 338
graphics formats information, 105
graphics tools, 352, 353
greeked text, 311
hexadecimal color codes, 160
"Hiding Your Email Address," 102
HTML 4.01 specifications, 19
HTML helper editors, 349, 350
HTML resource sites, 364–366
HTML utility collections, 356
HTML validators, 344, 355
image map tutorials, 118
image optimization tutorials, 104
JavaScript information, 264
link checkers, 354
Lynx, 335
MOMspider link checker, 342
`<object>` element information, 288
PayPal, 324
spell checking, 340–341
templates from this book, 298
TextPad text editor, 30
top online references, 357–366
UI design, 56
White House home page, 103–104
W3C Core style sheets, 135
W3C's Web Accessibility Initiative (WAI), 109
WYSIWYG editors, 350, 351
XHTML specifications, 19, 357–358
Yahoo! Merchant Solutions, 323
Zen Cart shopping cart, 324
intradocument links, 98–100. *See also* links
italics, applying to text, 172

• J •

Java versus JavaScript, 214
JavaScript
for arranging content dynamically, 215–217
arrays overview, 241–242
basic syntax rules, 228–229
body scripts, 224
canned, limitations of, 223
clock examples using, 278–281
color coding in this book, 17
components, 228
CSS with, 122
data types overview, 231
deferring execution of, 246
defined, 213
in DHTML, 271
disabling for older browsers, 281
Document Object Model (DOM), 245–246
events and event handling, 243–245
expression statements, 234
expressions, defined, 231
external file for, 225–227
for form validation, 219–220, 264
functions overview, 239–241
further information, 246, 264
header scripts, 224–225
help for users with function turned off, 46
for image rollovers, 272–275
including scripts in Web pages, 224–225
Java versus, 214
loops overview, 236–239
objects overview, 243
operators overview, 231–234
for pop-up windows, 218–219, 281–284
referencing an external file, 225–226
uses for, 214–222
variables overview, 229–231
window parameters, 283–284
(X)HTML markup related to, 246
JPEG (Joint Photographic Experts Group) format, 105
`.js` files, 225–227
Juicy Studio site, 288

• K •

keywords in `<meta />` element, 64–65

• L •

language attribute (<script> element), 224, 281
less-than sign (<)
  beginning comments (<!--), 23
  displaying on Web page, 22–23
  for HTML tags, 12
letter-spacing property (CSS), 168
<li> element
  for bulleted lists, 20, 85
  for definition lists, 88
  for numbered lists, 81–82
light green color coding in this book, 17
line breaks
  <br /> element for, 25, 77–78
  ignored in block elements, 75, 77
  paragraphs versus, 77
  in preformatted text, 76
line-height property (CSS), 167
lines
  horizontal rules, 25, 78–81
  using small image to create, 110, 111
line-through text, 173–174
link checkers, 353–354
<link> element for external style sheets (CSS), 134–135, 147
link pseudo-class (CSS), 162–163
links
  absolute, 93
  anchor element for, 91
  basic, 91–92
  broken, avoiding, 341–342
  broken, fixing, 94–96
  browser display of, 92
  "click here," avoiding, 51
  CSS styles for, 145–146, 161–163, 165
  cut and paste method for, 95–96
  defined, 10, 91
  to e-mail addresses, 102
  to external style sheets (CSS), 134–135, 147
  for file downloads, 101–102
  font family for (CSS), 165
  forcing page refresh, 342
  http:/ required for URLs, 96
  images as, 115–118
  intradocument, 98–100
  MOMspider link checker, 342
  to named locations, 98–100
  to non-HTML resources, 100–102
  off-site, considerations for, 49–51
  opening pages in new window, 96–97
  between pages of site, 94–95
  pseudo-classes for (CSS), 161–163
  redirecting users to another page, 66
  relative, 93–95
  resources that can be linked to, 92
  text for, 51
  tips for using, 337–338
  URL specification for, 92, 93, 94, 95–96
LinkScan/QuickCheck link checker, 354
lists
  bulleted, 20, 84–87
  definition, 87–89
  nesting, 89–90
  numbered, 81–84
  overview, 25
logo for company site, 309
loops (JavaScript)
  defined, 236
  for, 237–238, 242
  while, 238–239
Lopuck, Lisa (*Web Design For Dummies*), 56
lowercase. *See* case
low-scroll rule for navigation aids, 336
Lynx for testing, 335

• M •

mailto: links, 102
maintaining your site, 338
Mander, Richard (*Web Usability For Dummies*), 56
Mansfield, Richard (*CSS Web Design For Dummies*), 125
<map> element, 117
mapping your site. *See* site map
margin shorthand property (CSS), 140–141
markup. *See also specific kinds*
  color coding in this book, 16
  defined, 4, 11
  overview, 11–13
Matt's Script archive site, 266
maxlength attribute (<input /> element), 254
measurement values for properties (CSS), 124
media styles (CSS)
  paged media, 153–155
  visual media, 148–153
media types (CSS), 148

<meta /> element
  content attribute, 63–64
  http-equiv attribute, 65–66
  keywords and page descriptions in, 64–65
  name attribute, 63–64
  redirecting users to another page, 65–66
metadata
  defined, 63
  <meta /> element for, 63–64
  uses for, 64–65
method attribute (<form> element), 251
methods (JavaScript), calling, 243
Microsoft Word
  avoiding for HTML creation, 10, 30
  saving documents as text files, 32
MOMspider link checker, 342
monospaced type in this book, 2
moving files to Web server, 39–40, 54–56
multi-line text boxes, 261
multimedia
  paged media styles (CSS), 153–155
  visual media styles (CSS), 148–153
multiple attribute (<select> element), 259
music in auction pages, avoiding, 302
MySQL/PHP Database Applications (Bulger, Greenspan, and Wall), 323

• N •

name attribute
  <input /> element, 253, 255
  <meta /> element, 63–64
  <select> element, 259
namespaces, 61–62
navbar class (CSS), 139–141
navigation tools. *See also* UI (user interface)
  defined, 26
  For Dummies site example, 47–49
  to home page, 49
  images versus text, 180
  importance of, 336
  low-scroll rule for, 336
  navigation bar using CSS, 139–141
  off-site links, 49–51
  planning for expansion, 46
  for product catalog, 319
  simple text bar, 310
  site map, 45–46
  site map as aid to creating, 44, 49
  site size issues for, 47
  test visitor for, 49
  uses for, 43
  visual scheme for, 49
&nbsp (non-breaking space) character, 204
nesting elements
  CSS inheritance with, 131–132
  lists, 89–90
  overview, 20
  <pre> elements inside <blockquotes>, 76
  tables, 206–207
NetMechanic's HTML Toolbox, 356
new operator, creating arrays using, 242
Nielsen, Jakob (accessibility expert), 56
non-breaking space (&nbsp) character, 204
noshade attribute (<hr> element), 80
Notepad text editor, 30
numbered lists
  browser display of, 84
  defined, 81
  elements for, 81
  formatting, 82–83
  nesting, 89–90
  numbering, 83–84

• O •

<object> element, 287–288
objects (JavaScript), 243, 246
off-site links. *See also* links
  absolute links for, 93
  considerations for choosing, 49–50
  endorsement implied by, 50
  guidelines for, 50
  link text for, 51
  to named locations, 100
  to non-HTML resources, 100–102
  opening in new window, 96–97
  reviewing periodically, 50
<ol> element, 81, 83–84
On the Web icon, 6
operators (JavaScript)
  arithmetic, 232
  assignment, 232
  comparison, 233–234
  counting, 232–233
  defined, 231
<option> element for drop-down lists, 259, 260
orange color coding in this book, 17
ordered lists. *See* numbered lists
outside links. *See* off-site links
overlapping objects with CSS, 141–142
overlining text, 173–174

• P •

<p> element
  align attribute, 71–72
  closing tag for, 71
  creating paragraphs, 70–71
  described, 24
  “Hello World!” Letter example, 34
  with <input /> element, 253
  with tables, 204
padding for tables
  cellpadding attribute for, 192–193
  CSS styles for, 194–195
padding property (CSS)
  shorthand property for white space, 140–141
  for table cells, 194–195
paged media styles (CSS), 153–155
pageHit cookie, 284–286
pageVisit cookie, 284–286
Paint Shop Pro Photo X2, 352
paragraphs. *See also* <p> element
  alignment, 71–72
  formatting, 70–71
  indenting (CSS), 169–170
  line breaks versus, 77
  line breaks within, 77–78
  line height specification (CSS), 167
password fields, 254–255
paths (URL component), 15
PayPal
  adding shopping cart to site, 324–326
  shopping cart from, 323
  Web site, 324
PDF files
  linking to, 100–102
  off-site, avoiding direct links to, 50
peer reviews, 344
percentage property measurement values, 124
period (.)
  accessing object properties or methods, 243
  after CSS selector, 129–130
  in relative link notation (../), 94–95
Photoshop (Adobe), 352–353
Photoshop Elements (Adobe), 352
planning Web pages. *See also* designing; UI (user interface)
  building navigation tools, 47–49
  getting critiques, 44
  hosting considerations, 51–54
  identifying elements needed, 43
  importance of, 44
  mapping your site, 44–46
  moving files to your Web server, 39–40, 51–54
  off-site link considerations, 49–51
  overview, 31–32
  professional design help for, 44
  questions to ask, 42
  requirements determined by goals, 43
  setting goals for site, 42–43
  table layout, 179–180
  UI design resources, 56
  under construction sections, avoiding, 45
PNG (Portable Network Graphics), 105
populating table cells, 201–204
pop-up text for images, 109
pop-up windows
  adding to page, 283
  JavaScript for, 218–219, 281–284
  JavaScript window parameters, 283–284
  opening linked documents in, 96–97
  users’ dislike of, 97, 219, 281–282
post method, 252
posting files to Web server, 39–40, 54–56
<pre> element, 76
preformatted text, 75–76
preloaders for rollovers, 274–275
press page for company site, 309
print style sheet (CSS), 154–155
product catalog
  advantages of Internet for, 319
  category-page template, 326–327
  components, 319–320
  designing, 320
  detail-page template, 327–328
  further information, 323
  incorporating PayPal shopping cart, 324–326
  navigation interface, 319
  resources used by, 321
  shopping-cart software for, 322–324
properties (CSS). *See also specific properties*
  for color definitions, 161
  defined, 126
  measurement values, 124
  for paged media styles, 154
  shorthand, 140–141
  for table borders, 187–189
  for visual media styles, 149–152
protocols, 15, 16. *See also specific protocols*
pseudo-classes (CSS), 161–163
purple color coding in this book, 17

• Q •

quotation marks (“) for attribute values, 20
quotes, block, 74–75

• R •

radio buttons, 255–256
reading array elements, 242
redirecting users to another page, 65–66
relative links
  defined, 93
  guidelines for, 94
  for images, 106
  notation (../), 94–95
  simple, 94
  site links, 94–95
  unaffected by server or domain change, 94
relative property measurement values, 124
Remember icon, 6
Reset button for forms, 261–263
revolutionary changes, avoiding, 337
RGB color values, 160
rollovers
  defined, 272
  image, JavaScript for, 272–275
  text, CSS for, 275–277
rows attribute (<textarea> element), 261
rows (table), <tr> element for, 182–183, 195–196
rowspan attribute for tables, 198, 200–201
rules attribute for tables, 210

• S •

saving Word document as text file, 32
scope attribute for tables, 210
<script> element. *See also* client-side scripts
  language attribute, 224, 281
  processing order for multiple, 226
  src attribute, 225, 226
  type attribute, 224
scripting. *See also* CGI (Common Gateway Interface); client-side scripts; JavaScript
  online sources for scripts, 266
  server-side, 221
ScriptSearch site, 266
search engines
  metadata used by, 64–65
  site map benefits for, 46
  <title> element used by, 63
search forms, 247, 248–249. *See also* forms
<select> element for drop-down lists, 259–260
selected attribute (<option> element), 260
selectors (CSS), 126, 128
semicolon (;)
  in CSS declarations, 128
  ending entities, 21
  separating JavaScript statements, 229
servers. *See* Web servers
server-side scripting, 221
shape attribute (<area /> element), 117
shopping-cart software
  Add to Cart button, 324–325
  do-it-yourself, 324
  functions provided by, 322
  further information, 323
  hosted e-commerce services for, 323–324
  incorporating PayPal into site, 324–326
  PayPal for, 323
  View Cart button, 325
shorthand properties (CSS), 140–141
Simple Mail Transfer Protocol (SMTP), 16
SimpleText text editor, 30
site map
  benefits for users, 46
  planning for expansion, 46
  process of creating, 44
  under construction sections, avoiding, 45
  using for development, 44–45, 340
  as visual guide for users, 45–46
“Site Redesign Tutorial” (Webmonkey), 56
site reviews, 344–345
size
  for fonts, 74, 166
  for horizontal rules, 79
  image borders increasing, 113
  of site, navigation and, 47
  specifying for images, 110–112
  for text fields, 254
size attribute
  <hr> element, 79
  <input /> element, 254
slash (/)
  closing empty elements, 19
  in HTML versus XHTML, 19
  for JavaScript comments, 229
  in relative link notation (../), 94–95
SmartFTP software, 55
Smith, Bud (*Web Usability For Dummies*), 56
SMTP (Simple Mail Transfer Protocol), 16
software. *See also specific kinds*
  finding good buys, 347

FTP clients, 55, 356
graphics tools, 351–353
HTML editors, 29–30, 348–351
HTML utility collections, 356
HTML validators, 355
link checkers, 353–354
shopping-cart, 322–324
text editors, 30
for Web servers, 52
spacing. *See* aligning; white space
spammers, hiding e-mail addresses from, 102, 315
span attribute for tables, 210
<span> element, applying CSS styles using, 131
spell checking, 339, 340–341
square brackets ([ ]) as array operator, 242
src attribute
<img /> element, 20, 107
<input /> element, 263
<script> element, 225, 226
standards
HTML, 3, 16
for tables, 205
start attribute (<ol> element), 83
start tags, 18
Starting an Online Business For Dummies (Holden), 323
statements (JavaScript)
conditional, 235–236
defined, 228
expression, 234
Strict DOCTYPE declaration, 60, 61
strikethrough (line-through) text, 173–174
structure. *See* document structure
style attribute (CSS), 133, 135
<style> element
CSS style rules in, 133–134
turning internal to external style sheet, 147
for visual layouts, 139–141
style rules (CSS)
changing dynamically with JavaScript, 215–217
class attribute, 129–131
defined, 126
inheritance, 131–132
multiple per selector, 128
for old and new browsers, 147
for positioning elements, 141–142
precedence for, 136, 163
setting for entire document, 132
site-wide style sheet, 309, 316–318
<span> element with, 131
for visual layouts, 139–141
z-index with, 141
style sheets. *See* CSS (Cascading Style Sheets)
Submit button for forms, 261–263
summary attribute for tables, 210
syntax checkers, 335
syntax, defined, 15

• T •

<table> element
align attribute, 195–196
cellpadding attribute, 192–194
cellspacing attribute, 192–194
optional attributes, 183
required for tables, 182
for simple table, 182–183
width attribute, 190–191
width property (CSS), 192
tables
aligning, 195–198
basic structural elements of, 178–179
basic (X)HTML elements of, 181–182
borders for, 186–188
cell spans for, 178, 184, 198–201
centering, 197–198
color for cells, 208–209
creating a simple table, 182–183
creating a table-based Web page, 183–184
dense, avoiding, 208
drafting, 180–181
empty cells for controlling layout, 194
images versus text for navigation, 180
nesting, 206–207
other (X)HTML attributes, 209–210
other (X)HTML elements, 185
overview, 25
padding and spacing for, 192–195
planning layout of, 179–180
populating cells, 201–204
presentation-focused, 176
simple versus complex, 177–178
standards for, 205
testing, 205
tips for creating, 205–209
uses for, 175–178
white space for markup, 206
width adjustments, 189–192
(X)HTML and CSS for, 176
tags. *See* elements; *specific elements*
target attribute (<a> element), 97
<tbody> element, 185, 197

<td> element
  align attribute, 195–196
  border-spacing property (CSS), 194–195
  colspan attribute, 184, 198–200
  creating cells using, 183–184
  nesting <table> elements in, 206–207
  padding property (CSS), 194–195
  required for tables, 182
  rowspan attribute, 198, 200–201
  for simple table, 182–183
  valign attribute, 196
  width attribute, 190–191
  width property (CSS), 192
Technical Stuff icon, 6
templates
  for auction item presentation, 303–305
  category page, 327–328
  detail page, 327–328
  downloading, 298
  testing on multiple browsers, 336
  using ones in this book, 298
testing. *See also* validating
  beta-testing, 343
  Lynx for, 335
  multiple browsers for, 335–336
  navigation tools, 49
  pages for browsers disabling CSS, 128
  peer reviews for, 344
  site reviews for, 344–345
  tables, 205
  tips for, 335–336
  user feedback for, 345–346
text. *See also* font specifications using (CSS); text blocks
  for auction sites, tips, 302
  boldface for, 170–171
  color change using CSS, 161
  definition lists, 87–89
  for forms, 267
  greeked, 311
  horizontal rules, 25, 78–81
  HTML page as collection of blocks, 69
  for hyperlinks, 51
  hypermedia usage tips, 337–338
  as image alternative, 108–109, 116
  inline elements versus blocks, 70
  italics for, 172
  line breaks, 25, 77–78
  nesting lists, 89–90
  positioning using CSS, 168–170
  preformatted, 75–76
  rollovers, 275–277
  spell checking, 339, 340–341
  tips for choosing fonts, 296
text blocks. *See also* forms; tables; *specific kinds*
  aligning using CSS, 168–169
  bulleted lists, 20, 84–87, 89–90
  formatting ignored by browsers, 75
  headings, 24, 72–74, 144–145, 164–165, 204
  inline elements versus, 70
  numbered lists, 81–84, 89–90
  paragraphs, 24, 70–72, 77–78, 167, 169–170
  quotes, 74–75
  types of, 69
text editors, 30. *See also* HTML editors; *specific editors*
text fields, 253–254
text rollovers
  adding to page, 277
  browser display of, 276
  code listing, 277
  CSS for, 275–278
  image rollovers versus, 275
text-align property (CSS)
  for tables, 197
  for text, 169
<textarea> element, 261
text-decoration property (CSS), 173–174
text-indent property (CSS), 169–170
text-only browsers
  alternative text for images, 108–109, 116
  headings display in, 74
  need for, 13
TextPad text editor, 30
text-transform property (CSS), 172–173
TextWrangler HTML editor, 349
<tfoot> element, 185, 197
<th> element, 185, 197
<thead> element, 185, 197
Tip icon, 6
<title> element, 34, 62–63
Tittel, Ed (About Me page), 26–28
tools. *See* software
top-down page design, 332
Totty, Brian (HTTP expert), 16
<tr> element, 182–183, 195–196
Transitional DOCTYPE declaration, 60, 61
Tricks of the Trade icon, 6
type attribute. *See also* type attribute (<input /> element)
  <ol> element, 84
  <script> element, 224
  <ul> element, 85
type attribute (<input /> element)
  for check boxes, 255
  for file-upload fields, 257
  for hidden fields, 257

for password fields, 254
for radio buttons, 255
for Reset buttons, 262, 263
for Submit buttons, 262, 263
for text fields, 253
typos, avoiding, 339

• U •

UI (user interface). *See also* designing; navigation tools; planning Web pages
defined, 41
design resources, 56
Gmail's use of JavaScript for, 222
importance of, 41
site map, 44–46
visual scheme for, 49
`<ul>` element, 20, 85–86
under construction sections, avoiding, 45
underlining text, 173–174
underscore (_) instead of spaces in filenames, 35
Unicode character codes, 22, 361
unordered lists. *See* bulleted lists
uploading files to Web server, 39–40, 54–56
uppercase. *See* case
URLs (Uniform Resource Locators). *See also* links
components, 15
conventions in this book, 2
overview, 14–15
for posting files to server, 54
redirecting users to another page, 65–66
user feedback, 345–346
user interface. *See* UI

• V •

validating. *See also* testing
forms, 219–220, 264
HTML validators for, 344, 355
overview, 355
spell checking, 339, 340–341
style sheets, W3C service for, 129
`valign` attribute for table elements, 196, 197
`value` attribute
`<input />` element, 255, 262, 263
`<option>` element, 259
values. *See also* variables (JavaScript)
CSS, 126
JavaScript, 229, 230
variables (JavaScript)
declaring, 230–231
defined, 230
overview, 229–231
`vertical-align` property for tables (CSS), 197
View Cart button, 325
`visited` pseudo-class (CSS), 162–163
visual layouts using CSS, 139–141
visual media styles (CSS), 148–153
`vspace` attribute (`<img />` element), 114–115

• W •

Wall, David (*MySQL/PHP Database Applications*), 323
Warning! icon, 6
Web browsers. *See* browsers
Web Design For Dummies (Lopuck), 56
Web Design Group, 131, 365
Web Developer's Virtual Library (WDVL), 365
"Web Graphics Overview" (Webmonkey), 105
Web hosting. *See* hosting your site
Web Pages That Suck site, 56
Web resources. *See* Internet resources
Web servers. *See also* hosting your site
dedicated, 52
defined, 52
Internet connection for, 52
moving files to, 39–40, 51–54
overview, 13–14
relative links not affected by changing, 94
software for, 52
Web Usability For Dummies (Mander and Smith), 56
WebCom.com site, 365
Webmastering For Dummies (Wiley publication), 52
Webmonkey
hexadecimal color codes, 160
image optimization tutorials, 104
"Site Redesign Tutorial," 56
"Web Graphics Overview," 105
Web site, 365
WebReference.com, 366
Web-safe colors, 159
Westciv, CSS information from, 125, 153
`while` loops (JavaScript), 238–239
White House home page, 103–104

white space
- cellspacing attribute for tables, 192–193
- CSS and <div> elements for, 76
- CSS properties for tables, 194–195
- CSS shorthand properties for, 140–141
- ignored in block elements, 75
- image settings for, 114–115
- in JavaScript, 228–229
- line breaks for, 77
- &nbsp (non-breaking space) character for, 204
- in preformatted text, 76
- for table markup, 206
- uses for, 76

width attribute
- <hr> element, 79
- <img /> element, 110–111
- <table> or <td> element, 190–191

width property for tables (CSS), 192
word processors, avoiding for HTML creation, 10, 30
word-spacing property (CSS), 168
World Wide Web, 11
writing array elements, 242
WS_FTP software, 55
W3C Link Checker, 354
W3C (World Wide Web Consortium)
- Core style sheets, 135
- CSS3 specification, 123
- HTML standards kept by, 3, 16
- HTML Validator, 344, 355
- link to site, 92
- <object> element information, 288
- style sheet validation service, 129
- Web Accessibility Initiative (WAI), 109
- Web site, 364

W3Schools site, 288, 365
WYSIWYG editors, 350–351

• X •

XHTML. *See also* HTML (Hypertext Markup Language); (X)HTML
- color coding in this book, 17
- conventions in this book, 2–3
- DOCTYPE declaration, 60–61
- DTDs, 358–360
- HTML versus, 16–17, 19
- namespace, 61–62
- specifications for, 19, 357–358

(X)HTML. *See also* HTML (Hypertext Markup Language); XHTML
- color coding in this book, 17
- color names, 158–159
- color numbers, 159–160
- conventions in this book, 2–3
- deprecation of formatting markup, 121, 157, 197
- do's and don'ts, 331–338
- DTDs, 358–360
- examples online, 1
- HTML versus XHTML, 16–17, 19
- specifications, 19
- well-structured, elements required for, 23–24

XML (eXtensible Markup Language), 16, 17

• Y •

Yahoo!
- browsers list, 14
- Merchant Solutions, 323–324

"Your Complete Guide to Web Design" online, 56

• Z •

Zen Cart shopping cart, 324
Zvon's site, 365