



Your Ideal Companion to
HTML For Dummies®, 3rd Edition

MORE

HTML

FOR

DUMMIES®

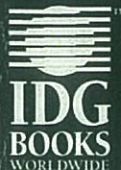
2ND EDITION

**NEW!
Revised &
Updated!**

**A Reference for
the Rest of Us!**

**by Ed Tittel
& Stephen N. James**

Coauthors of *HTML For Dummies*®, 3rd Edition



- **The Fun and Easy Way to Find Out MORE About Creating Web Pages With HTML (HyperText Markup Language)**
- **MORE of Ed and Stephen's Advice on HTML**
- **MORE on the Web Publication Process**



Your Ideal Companion to
HTML For Dummies®, 3rd Edition

MORE

HTML

FOR

DUMMIES®

2ND EDITION


**NEW!
Revised &
Updated!**

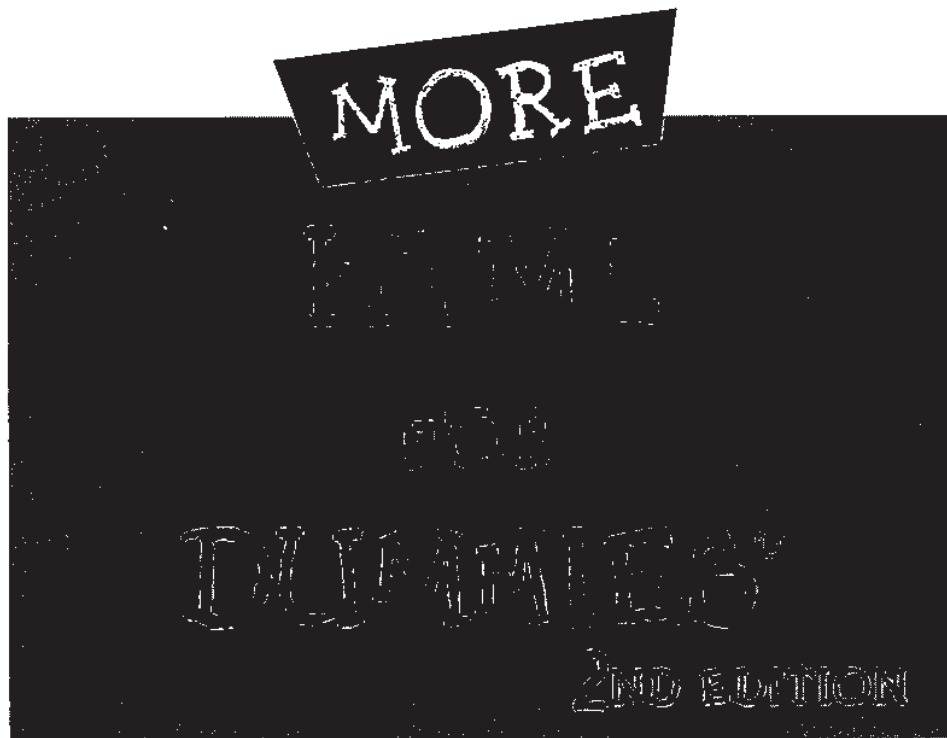
**A Reference for
the Rest of Us!**

**by Ed Tittel
& Stephen N. James**

Coauthors of *HTML For Dummies*®, 3rd Edition

- **The Fun and Easy Way™ to Find Out MORE About Creating Web Pages With HTML (HyperText Markup Language)**
- **MORE of Ed and Stephen's Advice on HTML**
- **MORE on the Web Publication Process From Design Through Maintenance**


**IDG
BOOKS
WORLDWIDE**



by Ed Tittel and Stephen Nelson James



IDG Books Worldwide, Inc.
An International Data Group Company

Foster City, CA ♦ Chicago, IL ♦ Indianapolis, IN ♦ Southlake, TX

MORE HTML For Dummies,® 2nd Edition

Published by
IDG Books Worldwide, Inc.
An International Data Group Company
919 E. Hillsdale Blvd.
Suite 400
Foster City, CA 94404
<http://www.idgbooks.com> (IDG Books Worldwide Web site)
<http://www.dummies.com> (Dummies Press Web site)

Copyright © 1997 IDG Books Worldwide, Inc. All rights reserved. No part of this book, including interior design, cover design, and icons, may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording, or otherwise) without the prior written permission of the publisher.

Library of Congress Catalog Card No.: 97-72408

ISBN: 0-7645-0233-6

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

2E/SW/QW/ZX/IN

Distributed in the United States by IDG Books Worldwide, Inc.

Distributed by Macmillan Canada for Canada; by Transworld Publishers Limited in the United Kingdom; by IDG Norge Books for Norway; by IDG Sweden Books for Sweden; by Woodslane Pty. Ltd. for Australia; by Woodslane Enterprises Ltd. for New Zealand; by Longman Singapore Publishers Ltd. for Singapore, Malaysia, Thailand, and Indonesia; by Simron Pty. Ltd. for South Africa; by Toppan Company Ltd. for Japan; by Distribuidora Cuspide for Argentina; by Livraria Cultura for Brazil; by Ediciencia S.A. for Ecuador; by Addison-Wesley Publishing Company for Korea; by Ediciones ZETA S.C.R. Ltda. for Peru; by WS Computer Publishing Corporation, Inc., for the Philippines; by Unalis Corporation for Taiwan; by Contemporanea de Ediciones for Venezuela; by Computer Book & Magazine Store for Puerto Rico; by Express Computer Distributors for the Caribbean and West Indies. Authorized Sales Agent: Anthony Rudkin Associates for the Middle East and North Africa.

For general information on IDG Books Worldwide's books in the U.S., please call our Consumer Customer Service department at 800-762-2974. For reseller information, including discounts and premium sales, please call our Reseller Customer Service department at 800-434-3422.

For information on where to purchase IDG Books Worldwide's books outside the U.S., please contact our International Sales department at 415-655-3200 or fax 415-655-3295.

For information on foreign language translations, please contact our Foreign & Subsidiary Rights department at 415-655-3021 or fax 415-655-3281.

For sales inquiries and special prices for bulk quantities, please contact our Sales department at 415-655-3200 or write to the address above.

For information on using IDG Books Worldwide's books in the classroom or for ordering examination copies, please contact our Educational Sales department at 800-434-2086 or fax 817-251-8174.

For press review copies, author interviews, or other publicity information, please contact our Public Relations department at 415-655-3000 or fax 415-655-3299.

For authorization to photocopy items for corporate, personal, or educational use, please contact Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, or fax 508-750-4470.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: AUTHOR AND PUBLISHER HAVE USED THEIR BEST EFFORTS IN PREPARING THIS BOOK. IDG BOOKS WORLDWIDE, INC., AND AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS BOOK AND SPECIFICALLY DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. THERE ARE NO WARRANTIES WHICH EXTEND BEYOND THE DESCRIPTIONS CONTAINED IN THIS PARAGRAPH. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES REPRESENTATIVES OR WRITTEN SALES MATERIALS. THE ACCURACY AND COMPLETENESS OF THE INFORMATION PROVIDED HEREIN AND THE OPINIONS STATED HEREIN ARE NOT GUARANTEED OR WARRANTED TO PRODUCE ANY PARTICULAR RESULTS, AND THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY INDIVIDUAL. NEITHER IDG BOOKS WORLDWIDE, INC., NOR AUTHOR SHALL BE LIABLE FOR ANY LOSS OF PROFIT OR ANY OTHER COMMERCIAL DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR OTHER DAMAGES. FULFILLMENT OF EACH COUPON OFFER IS THE RESPONSIBILITY OF THE OFFEROR.

Trademarks: All brand names and product names used in this book are trade names, service marks, trademarks, or registered trademarks of their respective owners. IDG Books Worldwide is not associated with any product or vendor mentioned in this book.



is a trademark under exclusive
license to IDG Books Worldwide, Inc.,
from International Data Group, Inc.

Table of Contents

Introduction	1
About This Book	1
How to Use This Book	2
A Few Assumptions	3
How This Book Is Organized	3
Part I: Advanced HTML Markup	4
Part II: Beyond HTML: Extending Your Web	4
Part III: Cool Web Applications	4
Part IV: Serving Up Your Web	5
Part V: Running a Successful Web Site	5
Part VI: Shortcuts and Tips Galore	5
Icons Used in This Book	6
Where to Go from Here	6
 Part I: Advanced HTML	 7
 Chapter 1: Understanding How HTML Happens	 9
The “Official” Channel: IETF	9
The “Unofficial” Channel: W3C	11
How a Proposal Becomes a Standard	11
Proposed standard	11
Draft standard	11
Standard	12
A Brief Review of HTML Standards to the Present	12
HTML 0.0	12
HTML 1.0	13
HTML 2.0	13
HTML+	13
HTML 3.0	14
HTML 3.2 (aka Wilbur)	14
Cougar	14
The never-ending HTML standards story	15
Pressure from the Marketplace	15
Netscape extensions	16
Internet Explorer extensions	16
Finding, Understanding, and Using HTML DTDs	17
The nature of an SGML document	17
About DTDs	18
What does a DTD contain?	18
Where the DTDs live	19

Deciding Which Flavor of HTML to Use	20
The advantages of using valid HTML.....	21
The disadvantages of using valid HTML	21
New HTML Worth Watching	21
Tables	22
Frames	22
Style sheets	23
Mathematics notation	23
Applets, scripts, forms, and more	23
Chapter 2: HTML Tables in Depth.....	25
Table Hindsight	25
Laying out tabular data for easy display	27
Multirow and multicolumn	27
Nesting	27
Tremendous Table Tags	28
The Awesome Attributes	28
<CAPTION> ... </CAPTION> Caption	29
<COL> ... </COL> Column (IE)	29
<COLGROUP> ... </COLGROUP> Column Group (IE)	29
<TABLE> ... </TABLE> Table	29
<TBODY> ... </TBODY> Table Body (IE)	30
<TD> ... </TD> Data Cell	30
<TFOOT> ... </TFOOT> Table Footer (IE)	31
<TH> ... </TH> Table Head	31
<THEAD> ... </THEAD> Table Header(IE)	31
<TR> ... </TR> Table Row	32
Attribute definitions	32
Some of the Good Stuff	35
Run for the borderline	35
Force the issue	36
A slick trick for the upper-left problem	36
Comment schmomment!	38
SPANning the gap	38
On Your Own	39
Chapter 3: HTML Frames in Full	41
Frame Hindsight	42
Fabulous Frame Tags	45
Introducing the attributes	45
<FRAME> Frame Definition	46
<FRAMESET> ... </FRAMESET> Frame Group	46
<NOFRAMES> ... </NOFRAMES> No Frames	46
<IFRAME> Inline Frame (IE)	46
Explaining frame attributes	47

<A>dditional Frame Markup	49
Complex Frame Creations	49
Some of the Good Stuff	53
Borderline, feels like I'm gonna lose my mind.	54
Request for comments!	55
Frame Scorn	55
Frame It Yourself	56
Chapter 4: Style Sheets in Detail	57
Understanding Style Sheets	58
Discovering the Syntax of Style Sheets	59
By the book: style rules	59
Properties	61
Color and background properties	61
Box properties	62
Classification properties	65
Font properties	65
Text properties	66
Units	67
Linking to External Style Sheets	68
Embedded and Inline Style Specifications	70
Using Tags to Define Style	71
Setting a Good Example: Sample Style Sheets	71
News.com	72
The Alertbox	76
The Richmond Review	79
Finding More Style Sheet Information	83
Chapter 5: Of Applets, Scripts, and Other Extensions	85
Caffeinating Your Pages: The <APPLET> and <PARAM> Tags	86
Applet attributes	86
Adding parameters to applets	88
About <EMBED>	88
HTML <EMBED> Markup	90
<EMBED>'s default attributes	90
<EMBED>'s other attributes	91
<NOEMBED>	91
General examples of <EMBED> tag markup	92
Specific examples of <EMBED> tag markup	92
An <EMBED> Replacement?	96
<OBJECT> . . . </OBJECT> Non-HTTP object	96
Getting Underneath <EMBED>	99

Chapter 6: Encouraging Interaction with HTML Forms 101

It's Just Not Enough: The Limitations of Forms	101
We Shall Overcome: Cougar Forms Markup	103
Rendering text areas read-only	103
Checking values without the server	104
Two buttons are not enough	104
Clicking in the general vicinity: hot labels	105
Getting there faster: keyboard shortcuts	106
Selective data acceptance	107
Disabling form controls	107
Forms and Scripting	108

Part II: Beyond HTML: Extending Your Web 109**Chapter 7: Extending and Improving the Web Experience 111**

If Your Browser Can't See It, Is It Really There?	111
Beyond HTML	112
Assisting the browser: helper applications	112
Hooking up with plugins	113
What's in Store for Your Web Pages?	114
The Hottest Plugins	115
The Perils of Incompatibility	116
Designing for Extended Capability	117
Serving the Audience	117
Trends in Web Extension Technology	118

Chapter 8: Shocking Your Web Site 119

A Shockwave Overview	119
A few words from Macromedia Director	120
Doing the Lingo limbo	121
Shockwave's bright future	121
What you need to use Shockwave	122
Installing and Using Afterburner	122
Running Afterburner on a Macintosh	123
Installing Afterburner on a Macintosh	123
Running Afterburner for Windows	124
Installing Afterburner on Windows	124
Implementing Afterburner	125
Configuring a Web Server for Shockwave	126
Shockwave configuration for specific Web servers	126
Shockwave and MIME	127
Shockwave configuration for NSCA httpd	127
Shockwave configuration for tenants	128
Shockwave configuration for W3C httpd	129
Shockwave configuration for Quarterdeck's WebSTAR	129
Placing Shockwave Files on Your Server	130
Using Shockwave Material	007

Do's and Don'ts for Designing Shockwave Materials	132
Do	133
Don't	134
The Top Shockwave Resources	134
Chapter 9: Virtual Worlds with VRML	137
VRML's Brilliant History	138
And the winner was	138
VRML and HTML: What's the difference?	139
So, what's VRML good for?	140
VRML's characteristics	141
Nodes	141
Field types	142
Creating VRML Content	143
A virtual world in seven easy steps	143
Creation tools for your VRML world	144
Virtual Home Space Builder	144
Virtus Walkthrough Pro	145
Installing and Using VRML-Enabled Browsers	146
Stand-alone PC browsers	147
WorldView from InterVista	147
CosmoPlayer	147
Helper applications for standard HTML browsers	147
VRwave for Windows	147
Live3D	148
Configuring your browser for VRML	148
Configuring your Windows browser for VRML	149
Configuring a Macintosh browser for VRML	149
Configuring Your Web Server for VRML	150
Design Do's and Don'ts for VRML	150
Do	151
Don't	151
The Top VRML Resources	152
Chapter 10: Java Still Jams!	153
A Java Overview	153
What makes Java different?	154
About those terms	155
Simple	155
Object-oriented	155
Distributed	155
Interpreted	156
Robust	156
Secure	156
Architecture-neutral and portable	157
High performance	157
Multithreaded	157
Dynamic	157

Java's history: from seedling to coffee bean	158
Java's learning curve	159
Java versus C++ — the differences	160
Creating Java Content	160
About Java programming	161
Say "Hello World": creating a Java program	162
Finding and Using Java-Enabled Browsers	163
Referencing Java Materials in Your Web Pages	164
Java Design Do's and Don'ts	165
Do	165
Don't	166
Top Java Resources	166
Java's Future	167
Chapter 11: Scripting Alternatives	169
The Mechanics of Scripting	170
Programming with JavaScript	171
Differences between Java and JavaScript	172
JavaScript syntax and semantics	173
JavaScript resources	174
Programming with VBScript	175
Dynamic HTML	177
The skinny on DH	178
Benefits of Dynamic HTML	179
The big boys	180
To Script, or Not to Script?	180
Part III: Cool Web Applications	181
Chapter 12: Using Your Own Search Engine	183
What Is a Search Engine?	183
What Makes 'Em Tick?	184
Two Ways to Use Search Engines	185
Internet-wide search capability	186
Site-specific search capabilities	187
The Search Contenders	188
Chapter 13: Cussin' and Discussin' on the Web	191
An Intro to Forums	191
One Lump or Two?	192
Stand-alone versus integrated servers	193
Database: custom or current?	193
With or without whipped cream?	194
For your eyes only?	194

Playing God	194
Additional genetic traits	195
The Contenders	196
Chapter 14: When Push Comes to Shove	199
Push 101	199
Pondering Push	200
Push Prerogatives	201
HTML or proprietary?	201
Hot list or real content?	202
Browser or helper application?	202
Push Principles	202
Push in the Private Sector	203
Push Particulars	205
A Plethora of Push Products	205
The cheap ones	206
Take out a loan	206
 Part IV: Serving Up Your Web	 209
Chapter 15: Inviting the World onto Your Web	211
It's Show Time!	211
Stay Focused on Your Purpose	212
Keep Your Site in Charge	212
Publish, or Perish the Thought	213
It's good for business	214
. . . and it's good for pleasure	214
Back to the content	214
Dealing and Wheeling: Hooking Up with an Internet Service Provider ...	215
Where to draw the line?	216
What to work out with your provider	217
What else you need to find out	218
What ISPs are going to ask you	218
Webbing It Yourself: What Does That Mean?	219
Understand the tariffs	219
UNIX or no UNIX?	220
Try though you might, you can't escape maintenance	220
Are You Ready for Success?	221
The Answer to the Ultimate Question	222
 Chapter 16: If You Build It, Will They Come?	 223
Announcing Your Web Site to the World	223
Write a semi-formal announcement	224
Where to direct your announcement	225
Trolling the Usenet newsgroups	226
Niche or industry publications	227
The old-fashioned kind of networking	227

Stay on the Safe Side of Acceptable Use Policies	228
Giving Value Means Getting Value	229
Making Sure Your Web Catches the Right Prey	229

Chapter 17: The More Things Change . . . 231

The Two-Dimensional Text Trap	232
Who Says This Stuff Is Stale?	232
Check in on your pages regularly, Doctor Web	233
Keep your content current	233
Do your links point to nowhere?	233
Is your HTML passé?	234
If You Ask Them, They'll Tell You	234
Keeping Up with Changes	235
Maintenance Is an Attitude, and a Way of Life	236
When Things Change, They Also Break	236

Part V: Running a Successful Web Site 239

Chapter 18: What's Really in Your Web Site? 241

Know Your Web Server Administrator	242
Web Server Computer Platforms	243
UNIX	243
Macintosh	243
Windows NT	243
Web Server Software	244
Apache, NCSA, and the rest	244
Windows NT	245
WebSTAR and MacHTTPD	245
How Your Web Site Fits into the Whole Internet	245
Inventory Web Server Resources	246
Take Stock of Your Web Site	246
Lotsa docs (it's not a medical convention)	247
Graphics galore	247
The supporting cast of applications	248
Marvelous miscellany	248
"Organized Web Site" Is Not an Oxymoron	248
Where does your site live?	248
Picture your directory/file structure as a tree	249
You can't tell the territory without a map	249
Understanding all the pieces and parts	249
Using remote hyperlinks	250
What's the code situation like?	251
Any image maps in the picture?	251
Strategic Planning for Your Web Site	251

Chapter 19: Web Server Administration — the Easy Way	253
Web Server Hosting Options	254
Web server hosting services	254
Your friendly neighborhood ISP	257
Your organization's LAN	257
You!?	258
How the Web Server Fits into the Internet	259
The hardware: computer and telephone equipment	259
Web server software and (briefly) how it works	261
The Savvy Webmaster's Management Techniques	263
Laying out your Web space	264
Designing and handling the file system.....	264
Working with log files	265
Administration and monitoring tools	266
For More about Your Web Options	266
Weighing Costs Against Other Considerations	267
What's your bottom line?	267
Home-grown versus store-bought Web sites	268
Chapter 20: Managing the Web Publication Process	269
What Should Be Hanging in Your Web?	270
Planning for Now and the Future	271
Planning for Regular Updates	272
Designing Documents for Multiple Uses	273
Working with Creation and Production Staff	274
Enlisting staff support and assistance	275
Integrating the Web into the overall process	275
Conversion Is a Real Time-Saver	276
Doing conversion manually.....	276
The power (and limitations) of automated conversion	277
Seeking tools	277
Cyberleaf	278
HTML Transit	278
Web Publisher	280
WebWorks	281
What should you remember about HTML converters?	281
Déjà Vu — Elements of Page and Site Design	282
Give all pages a title	283
Make the most of text and hypertext links	283
Use graphics for maximum effect.....	283
Think in 24-D	284
Stringing pages together, the book way	284
Hierarchies are natural	285
Multiple tracks for multiple audiences	286
Extending your Web, a comment at a time	286

Chapter 21: Web Site Management Tools and Techniques 289

Managing Multitudes of Documents	290
How big is your Web?	290
Organize to untangle your Web	290
Document management systems	292
Examining the Alternatives	293
Web site managers	294
SITEMAN	294
LivePAGE	295
SiteMill	295
FrontPage	296
Let someone else manage your Web	298
Cool Management Tools	299
Doctor HTML	299
Weblint	300
The WebTechs HTML Validation Service	302
Acquiring the Perfect Tool Set	302
Web site usage tools and services	302
Log file analysis tools	303
Your Web Management Routine	304
The principles of Web site maintenance	304
Creating your Web management plan	305
The virtues of regular attention	306
Web administration is a part/full-time job	307

Chapter 22: Making Your Content Accessible 309

Of Spiders, Robots, Worms, and Other Agents	309
The past and future of Web agents	311
Agents in search engines and other WWW environments	312
Building search boundaries	312
What works globally also sometimes works locally	313
The Best of 'bots	313
Robot exclusion	314
They're baaaack!!	315
Just the Facts Ma'am: Meta-Information	315
Here I Am: Registering with Search Engines	317
Secret Agent Man	318

Part VI: Shortcuts and Tips Galore 319**Chapter 23: Maintenance Tips and Tricks 321**

Routine Is Everything	321
Remembering the content	322
Keeping track of dates	323
Updating your HTML	323
The Beauty of a Test Web	324
Overcoming Inertia Takes Constant Vigilance	325

Chapter 24: Advanced HTML Tips and Shortcuts	327
Don't Forget the Other Guys	327
Multiple Means of Delivery	329
Let the Users Choose	329
<TABLE> Alternatives	330
<FRAME> Alternatives	332
Knowing When to Split	332
Managing Miscellany	333
Adding Value for Value	333
Chapter 25: Extenuating Extensions	335
Warning! Plug In Now!	336
Shelling into Shockwave	337
Visualizing the Virtual with VRML	338
Jumping into Java	339
Effective Augmentation	340
Glossary	341
Index	365
IDG Books Worldwide Registration Card	Back of Book

Chapter 4

Style Sheets in Detail

In This Chapter

- ▶ Reviewing style sheets
 - ▶ Understanding style sheet syntax
 - ▶ Using external style sheets
 - ▶ Using embedded style sheets
 - ▶ Defining style with tags
 - ▶ Exploring some sample style sheets
 - ▶ Utilizing style sheet resources
-

In the beginning, the Web was a simple communication medium, used mostly by researchers and academia to exchange information. The information didn't have to be pretty or perfectly arranged, it just needed to be correct. As the Web spread to the business world, organizations developed a true understanding of the prowess of the Web and its ability to reach more people using fewer resources. The fact that users could have a measure of control over the layout of information in a Web document baffled those users who had only lived in a world of print and presentations. The inability to create columns and use different typefaces was frustrating to Web page authors. Never mind that HTML and the Web in general weren't really designed to support such detailed layout information — the pages just didn't look right!

Because the Web was not just a passing fad, the major browser developers and the World Wide Web Consortium recognized the importance of meeting this need for more user control of Web pages' layouts— without violating the multiplatform, open-ended state of the Web. *Style sheets* were the answer.

In this chapter, we review the Web style sheet concept, give a complete overview of the syntax behind style sheets, look at the three different ways you can include style information in your Web pages, walk you through three different style sheets, and point you to several style sheet resources on the Web. Whew! Just saying all that was hard . . . let's do it!

Understanding Style Sheets

The style sheet concept is not new, but in fact has been borrowed from word processing and page layout programs. In general, a style sheet defines a set of layout parameters for a document to ensure that similar elements in the document appear uniformly. In English, this statement means that all instances of the same element, such as a heading or bulleted item, look alike—the *style* applies the same font, margin, and paragraph specifications to each occurrence of that element.

For example, why should you have to work through five different menus each time you want to create a level one heading? Simply choosing Heading 1 from a style list (and having the text rendered in red, Times 24, a half inch from the base margin, and with an 18-point space afterward) is much easier and more reliable.

This advantage also applies to HTML documents. Before style sheets, you could only change the size and color of text in a Web page and you had to use the `` tag every time. HTML style sheets are like any other style sheet: They define how certain elements in a Web page should appear.



Style sheets also include already established HTML tags such as `<H1>` and `<H2>` and `` — your style sheet definitions override existing browser rendering for any tag, so you can take advantage of existing tags and just alter their appearance.

Taking another cue from standard style sheets, in HTML you can store style sheets individually, outside of the pages that reference them, and apply them to a collection of Web pages, which allows you to create one style template and apply it many times. These style sheets are called *external style sheets*. This flexibility creates a uniform look and feel throughout a large collection of Web documents.

To make style changes on the individual page level, you can include style markup within the document. Style markup works with (or overrides) any specifications in a referenced external style sheet. You can use style markup to customize the color, text font, and other attributes of a specific element, or create separate instances of an element, each with its own specific style.

Style sheets are not limited to one set of specifications per tag in a document. For example, you can have three separate level one headings defined in a style sheet, each with a different `CLASS` name, and apply whichever you want throughout the document. In the following section, we show you how to create multiple style definitions for a single tag, along with everything else you need to know to create your own style sheets.



Before we go any further, it's important to mention that style sheets are not fully implemented across all browsers and are not 100 percent backward-compatible with existing browsers. The style sheet information we present in the rest of this chapter is based on the Cougar specification (the latest HTML proposal). This means that all the style sheet elements and options we present in this chapter may not be usable right now. We can say with strong certainty that in the next few months, you will be able to use all the information and instructions in this chapter.



For the latest information, it's best to check the most recent Web resources at the following URLs:

- ✓ The World Wide Web Consortium Style Sheets page at:

<http://www.w3.org/pub/WWW/Style/>

- ✓ The Cougar HTML and Style Sheets specification at:

<http://www.w3.org/pub/WWW/TR/WD-style>

Discovering the Syntax of Style Sheets

After we've thrown out just enough information to make you really hungry for the how to's of style sheets, you're probably ready to satiate that desire and get down to the nitty-gritty. Style sheets are not difficult to understand if (and this is a big *if*) you've ever worked with other kinds of style sheets, you understand how HTML works, and you understand how browsers display Web pages. These are not easy prerequisites, but we're assuming that since you picked up this *MORE* book, you know something about HTML basics. (See *HTML For Dummies*, 3rd Edition, if you need to find out the fundamentals of HTML.)

By the book: style rules

Style sheets are essentially collections of rules that tell a browser how a document should appear. As with all other things HTML, these rules have a specific syntax. You can link these rules to your Web pages in multiple ways, but we get to that in few pages. The good news is that both external and embedded style sheets use the same rule syntax, and that individual tags have attributes to help you assign style specifics. When you learn the rule syntax, you're over the biggest hurdle.

Every rule has two components:

- ✓ **A selector:** Usually an HTML element such as ``, `<H1>`, or `<ADDRESS>`.
- ✓ **The style for the selector:** A selector's style can contain one or more predefined properties and their values. In the style `font-family: Times` the property being defined is `font-family`, and the value is `Times`.

You can create a rule by using the following syntax:

```
selector {property: value }
```

For example, the rule for a blue level one heading is:

```
H1 {color: blue}
```

You can include multiple properties and values for any one selector by separating the property-value pairs with semicolons. The syntax looks like this:

```
selector { property1: value1; property2: value2 }
```

For example, the rule for a blue level one heading in Helvetica font is:

```
H1 {color:blue; font-family: Helvetica}
```

These rules can apply to HTML elements (ones already defined by the DTD) and elements that you create yourself (new markup that you create for your own specific needs). If you want to create a custom style called “heading” that appears in blue Helvetica, rather than defining a style for the standard `<H1>` element, the style notation is:

```
heading {color:blue; font-family: Helvetica}
```

The preceding code creates a class called “heading,” which you can reference throughout your pages. You can also create multiple versions of an HTML tag by creating multiple classes. The following HTML code alters the standard `<H1>` tag and creates two new tags, one of class `red`, as well as one of class `blue`:

```
H1.red {color:red; font-family: Helvetica}
H1.blue {color:blue; font-family: Helvetica}
```

To specify which one you want to use in any given instance, use the `CLASS` attribute, as in the following code:

```
<H1 CLASS="red">
```

Pretty nifty, huh?

Properties

Properties can be divided into categories, which makes remembering them easier. The following sections look briefly at each family and any special syntax rules that a property might have.

Color and background properties

The following properties define the color scheme used within a document:

- ✓ **Background Attachment** specifies whether the image defined in `background-image` will scroll with the page or be fixed on the canvas. Here is an example:

```
P { background-image: url(http://www.site.com/↵
    background.gif); background-attachment: scroll }
```

- ✓ **Background Color** defines the background color using one of several predefined colors (discussed in the “Units” section later in this chapter) or the #RRGGBB color code. Here is an example:

```
P { background-color: blue }
```

- ✓ **Background Image** identifies the URL for a background image that appears on the page. Here is an example:

```
P { background-image: url(http://www.site.com/↵
    background.gif) }
```

- ✓ **Background Position** specifies where a background image will appear on a page, using the keywords top, center, bottom, left, middle, or right. Here is an example:

```
P { background-image: url(http://www.site.com/↵
    background.gif); background-position: center }
```

- ✓ **Background Repeat** determines how a background image will be repeated. By using repeat, the image tiles in the standard fashion; repeat-x only repeats the image horizontally; repeat-y is limited to vertical repetition; and no-repeat ensures that the background image is not repeated at all. Here is an example:

```
P { background-image: url(http://www.site.com/↵
    background.gif); background-repeat: repeat-y }
```

- ✓ **Color** defines the color of an element by using a predefined name or #RRGGBB notation. Here is an example:

```
P { color: blue }
```


- ✓ Background incorporates all of the other background properties into one value. Here is an example:

```
P { background: url(http://www.site.com/
background.gif) scroll blue center no-repeat }
```

Box properties

The following properties define margins and borders for text blocks. The measurements may be a little unfamiliar to you, but never fear, we explain them all in the section entitled "Units."

- ✓ Top Margin specifies the top margin of a text block by using length or percentage. Here is an example:

```
P {margin-top: 5em}
```

- ✓ Right Margin specifies the right margin of a text block using length or percentage. Here is an example:

```
P {margin-right: 5em}
```

- ✓ Bottom Margin specifies the bottom margin of a text block using length or percentage. Here is an example:

```
P {margin-bottom: 5em}
```

- ✓ Left Margin specifies the left margin of a text block using length or percentage. Here is an example:

```
P {margin-left: 5em}
```

- ✓ Margin uses four values to assign top, right, bottom, and left margins in that order. The following example assigns a margin of 5 to the top and bottom and 2 to the left and right sides:

```
P {margin: 5em 2em 5em 2em}
```

- ✓ Top Padding defines how much space should be inserted between the top border and the text using length or percentage. Here is an example:

```
P {padding-top: 10%}
```

- ✓ Right Padding defines how much space should be inserted between the right border and the text using length or percentage. Here is an example:

```
P {padding-right: 10%}
```

- ✓ Bottom Padding defines how much space should be inserted between the bottom border and the text using length or percentage. Here is an example:

```
P {padding-bottom: 10%}
```

- ✓ Left Padding defines how much space should be inserted between the left border and the text using length or percentage. Here is an example:

```
P {padding-left: 10%}
```

- ✓ Padding uses four values to assign top, right, bottom, and left border padding in that order using length or percentage. The following example assigns a padding of 10 percent to the top and bottom and 20 percent to the left and right sides:

```
P {margin: 10% 20% 10% 20%}
```

- ✓ Top Border Width defines how wide the top border should be using thin, medium, thick, or a defined length. Here is an example:

```
P {border-top-width: medium}
```

- ✓ Right Border Width defines how wide the right border should be using thin, medium, thick, or a defined length. Here is an example:

```
P {border-right-width: medium}
```

- ✓ Bottom Border Width defines how wide the bottom border should be, using thin, medium, thick, or a defined length. Here is an example:

```
P {border-bottom-width: medium}
```

- ✓ Left Border Width defines how wide the left border should be by using thin, medium, thick, or a defined length. Here is an example:

```
P {border-left-width: medium}
```

- ✓ Border Width uses up to four values to assign top, right, bottom, and left border widths in that order, using thin, medium, thick, or a defined length. Here is an example:

```
P {border-width: medium thick medium thick}
```

- ✓ Border Color uses up to four values to assign top, right, bottom, and left border colors in that order by using a predefined color or #RRGGBB notation. Here is an example:

```
P {border-color: blue red white green}
```

- ✓ **Border Style** uses up to four values to assign top, right, bottom, and left border styles in that order by using none, dotted, dashed, solid, double, groove, ridge, inset, or outset. Here is an example:

```
P {border-style: double solid solid solid}
```

- ✓ **Top Border** sets the width, style, and color of the top border. Here is an example:

```
P {border-top: thin dotted blue}
```

- ✓ **Right Border** sets the width, style, and color of the right border. Here is an example:

```
P {border-right: thin dotted blue}
```

- ✓ **Bottom Border** sets the width, style, and color of the bottom border. Here is an example:

```
P {border-bottom: thin dotted blue}
```

- ✓ **Left Border** sets the width, style, and color of the left border. Here is an example:

```
P {border-left: thin dotted blue}
```

- ✓ **Border** sets the width, style, and color of all borders. Here is an example:

```
P {border: thin dotted blue}
```

- ✓ **Width** defines in length or percentage the width of the space that the text will occupy. Here is an example:

```
P {width: 50%}
```

- ✓ **Height** defines the height of the space that the text will occupy in length or as automatically assigned. Here is an example:

```
P {height: auto}
```

- ✓ **Float** allows text to wrap around an element to the left, right, or not at all by using none. Here is an example:

```
P {float: left}
```

- ✓ **Clear** determines whether an element allows other elements to float on one or more sides using none, left, right, or both. Here is an example:

```
P {clear: both}
```


Classification properties

These properties provide information about how white space, lists, and other elements should be treated by a browser:

- ✓ **Display** creates one of four elements to be displayed in conjunction with the element it affects. These elements include `block`, creating a line break before and after the element; `inline`, which removes line breaks before and after the element; `list-item`, creating a bullet for the element without creating a list; and `none`, which turns off any display. The following example adds line breaks before and after the paragraph:

```
P {display: block}
```

- ✓ **Whitespace** defines how white space will be treated within the element. Choices include `normal`, collapsing all white space into one line; `pre`, rendering all hard returns regardless of the number; and `nowrap`, preventing line-wrapping without a `
` tag. Here is an example:

```
P {white-space: pre}
```

- ✓ **List Style Type** identifies what type of marker is used with a list item, including `disc`, `circle`, `square`, `decimal`, `lower-roman`, `upper-roman`, `lower-alpha`, `upper-alpha`, and `none`. Here is an example:

```
P {list-style-type: square}
```

- ✓ **List Style Image** identifies an image that is used as a list item marker. Here is an example:

```
P { list-style-image: url(bullet.gif)}
```

- ✓ **List Style Position** defines how a list marker is placed relative to the list item. The `inside` value forces text to wrap under the marker; `outside` indents text. Here is an example:

```
P ( list-style-position: inside)
```

- ✓ **List Style** defines the list style type, list style position, and list style image for any given list. Here is an example:

```
P { list-style: inside url(bullet.gif) square}
```

Font properties

These properties affect the way fonts appear and include everything from typeface to size:

- ✓ **Font Family** defines the typeface family or generic font type. The font family may be any named family, such as Times, Helvetica, or Geneva. Five generic font types exist, which include serif, sans-serif, cursive, fantasy, and monospace. You should include a generic font type whenever you define a specific font (in case the font you defined isn't installed on a user's machine). Any family name that includes spaces, such as "New Century Schoolbook", must be enclosed in quotation marks. Here is an example:

```
H1 {font-family: "New Century Schoolbook", sans-serif}
```

- ✓ **Font Size** specifies the size of the text. You can choose to specify one of these absolute sizes: xx-small, x-small, small, medium, large, x-large, or xx-large. Other options include larger, smaller, or a percentage relative to the base font size. You can also define the size in length. Here is an example:

```
H1 {font-size: xx-large}
```

- ✓ **Font Style** defines the style in which a font should be rendered. Choices include normal, italic, and oblique. Here is an example:

```
H1 {font-style: italic}
```

- ✓ **Font Variant** renders the text in either normal or small caps. Here is an example:

```
H1 {font-variant: small-caps}
```

- ✓ **Font Weight** defines the lightness or darkness of the text. Your choices include normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, and 900. Then numbers represent the actual darkness of the font weight, with 100 being lightest and 900 darkest. Here is an example:

```
H1 {font-weight: lighter}
```

- ✓ **Font** incorporates all of the other font properties into one value. Here is an example:

```
H1 {font: Helvetica, sans-serif xx-large italic small-caps lighter}
```

Text properties

These properties determine text spacing, alignment, line spacing, and more:

- ✓ **Letter Spacing** uses the length format to define the spacing between letters. Here is an example:

```
P { letter-spacing: +.05em}
```

- ✓ **Line Height** specifies the amount of space between baselines using length or percentage. Here is an example:

```
P { line-height: +2em}
```

- ✓ **Text Alignment** defines the alignment of text. Choose from left, right, center, or justify. Here is an example:

```
P { text-align: right}
```

- ✓ **Text Decoration** assigns one of five text modifications, including none, underline, overline, line-through, and blink. You can use this property to remove underlining from links. Here is an example:

```
A.visited { text-decoration: none}
```

- ✓ **Text Indentation** specifies the text indentation using length or a percentage. Here is an example:

```
P { text-indent: 5em}
```

- ✓ **Text Transformation** changes the case of text using capitalize, lowercase, uppercase, or none. The none value instructs the browser to leave the text as it was originally created by the author. Here is an example:

```
P { text-transform: capitalize}
```

- ✓ **Vertical Alignment** alters the alignment of text relative to the baseline. Choose from baseline, sub, super, top, text-top, middle, bottom, and text-bottom — or you can specify a percentage. Here is an example:

```
P { vertical-align: super}
```

- ✓ **Word Spacing** uses the length format to define the spacing between words. Here is an example:

```
P { word-spacing: +.05em}
```

Units

The lengths and percentages that we discuss in the preceding sections have specific rules that apply to them, as well as to URL and color notation. This section contains the information that you need in order to use all these units correctly in your style sheets:

- ✓ **Length Units:** You can use one of three relative length units in your style sheets: em is a measurement relative to the height of the element's font, so it adds or subtracts the specified number of ems from the actual height of the element's font; ex refers to the height of the letter x. The unit px stands for pixels as related to the height of the screen.

You can use a number to set the number of ems, exs, or pxs. Another option is to include the + or - sign with a number to indicate the base size plus or minus the supplied number. The legal absolute values are in (inches), cm (centimeters), mm (millimeters), pt (points, which are $\frac{1}{72}$ of an inch), or pc (picas, which are equal to 12 points).

- ✓ **Percentage Units:** Percentages can also be stand-alone numbers with the percentage sign (%), which represent a certain percentage of the screen or any other predefined value, such as the width of a text block. You can also use the + and - signs with percentages.
- ✓ **Color Units:** The predefined colors recognized by style sheets and most browsers are aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow. If you want use any other color, you must use #RRGGBB hexadecimal color notation. If you want to brush up on your hexadecimal notation, be sure to read the sidebar “A Hex Upon That Color” in Chapter 2.
- ✓ **URLs:** To link to a URL in a style sheet, use this notation: url(url.here.)

Whew, we bet you thought the properties were never going to end. The good news is that you have a wide variety of options to work with when creating style sheets. Remember that all of these properties may be used in external or embedded style sheets.

Linking to External Style Sheets

When you have all of the pieces and parts of a style sheet, how exactly do you link style definitions to your Web pages? It's so simple that it's scary. After you put together a style sheet, you can reference the style sheet from any HTML page by using the <LINK> tag. The HTML code looks like this:

```
<LINK REL=StyleSheet HREF="mystyle" TYPE="text/css"
      MEDIA=screen>
```

In the preceding code example, the REL attribute defines the link as a connection to a style sheet, HREF gives the actual location of the style sheet file, and the TYPE attribute defines a style sheet as CSS or *cascading* — browsers that don't support cascading style sheets will ignore the style. Cascading refers to a specific type of style sheet, one that can work well with other style sheets and embedded style information. Other types of style sheets have their own designations, such as text/dsssl for the Document Style Semantics and Specifications Language (DSSSL). The large majority of

the style sheets you create will be cascading. If you choose to use another type, you must read its documentation to find out its specific TYPE value. (**Remember:** Most browsers that support style sheets also support CSS, but if you're using some other type of style sheet, you will definitely want to include type information.) Finally, the MEDIA attribute in the preceding example defines the medium to which the style sheet is best suited. Most Webmasters create their pages to be viewed primarily on a computer screen, but this attribute will allow you to specify what medium your information is best presented in by using one of the following values:

- ✓ `print` is used for pages that should be printed.
- ✓ `screen` is used for pages that should be displayed on a computer screen.
- ✓ `projection` is used for pages that should be displayed in a projection format.
- ✓ `braille` is used for pages that should be output as Braille.
- ✓ `aural` is used for pages that will be read using a text-to-speech reader.
- ✓ `all` is used for pages that can be used by all output devices (this is the default media).

Many of these media values, `aural` and `braille` for example, are geared toward making Web information more accessible, whereas the `print` and `projection` media allow Web information to be formatted specifically for those output devices rather than trying to force a screen-oriented page to print well or show up clearly on a large presentation screen. In the future when these different media are better supported by the Web, you may have multiple style sheets for each kind of media. For now, it's just important to know your options.

To use the styles you've defined within your Web pages, you can use the CLASS attribute with any tag, or use the `` or `<DIV>` tags with the CLASS attribute to apply a style you've created to a section of your document. `` is a text-level element used to assign a style to text only, just like `` or `<CITE>`. If you create a style called `.introtext`, for example, that applies to only the first few words of a page, you can use this HTML code to reference the style:

```
<SPAN CLASS=introtext>
```

```
These are the first few</SPAN> words of the paragraph.
```

The words within the `` tags then have the `.introtext` style applied to them, but the rest of the text is left alone.

The `<DIV>` tag is a block-level element, one that separates text into logical groups, that can contain multiple paragraphs, headings, and other divisions. For example, if you create the style `.address`, use this HTML code to reference it in your document:

```
<DIV CLASS=address>
  Your address information here
<P>
  Your URL here
</DIV>
```

Embedded and Inline Style Specifications

Although you use the `<LINK>` tag to reference an external style sheet, you can use the `<STYLE>` tag to include style information directly within your HTML document, which is also known as an inline style. Any style information you include in an external style sheet can be written directly into an HTML page, but the style information cannot be referenced by other pages. You can, however, use an *inline style* to override or complement an external style sheet that an HTML page references. This enables you to write general style sheets that create an overall look and feel for your pages, but also retain page-level control by adding page-specific styles using the `<STYLE>` tag. To include an inline style, use this HTML code:

```
<HEAD>
<STYLE TYPE="text/css" MEDIA=screen>
  @hy
    BODY { background: url(foo.gif) blue; color: white }
    STRONG{ background: white; color: blue }
    .address { margin-left: 5em; margin-right: 5em }
  @hy>
</STYLE>
</HEAD>
```

You use the `` and `<DIV>` attributes to access these inline styles just as you would with an externally referenced style sheet.

Using Tags to Define Style

Several HTML tags have style built into their attributes, most notably the `` and `<BODY>` tags. The `` attributes include the `COLOR`, `FACE`, and `SIZE`, attributes for altering the type on a case-by-case basis. The `<BODY>` tag includes the `ALINK`, `BACKGROUND`, `BGCOLOR`, and `VLINK` attributes for page-level body style.

These two tags do not have the flexibility or extensive list of properties that style sheets have to offer. However, you can use these tags with external or embedded style sheets for specific style needs.

Setting a Good Example: Sample Style Sheets

In the preceding sections, we have described all the components of style sheets, but haven't really shown you a complete one yet. The best examples are those that actually work, so we include the following examples to show you some real-world applications of style sheets.

Back when we were learning HTML (when there were two whole books on the market to help us out), we learned by doing and by imitating others. The View Source command was the key to finding out how other people used HTML to create pages. Now, so many pages are created on-the-fly from databases or use complicated scripting that learning from other people's work isn't so easy anymore. When you view the source code of these pages, you will find quite a bit of code but very little HTML, if any at all. That makes it difficult to learn from other people's work.

When you create style sheets, taking what already works and then modifying it to fit your needs is always easier than starting from scratch, especially when the technology is new. Because so many style sheets are external (and the files not available to the world at large), in this section we include three separate style sheets, each with a slightly different style implementation, to give you a place to start.



You can use any of the elements in these sheets to build your own style sheet. Mix and match and modify to your heart's content. Remember, practice makes perfect. See the "Finding More Style Sheet Information" section for a rundown on what browsers support style sheets so that you can check your work and see whether you've accomplished what you set out to do.

News.com

This popular news site, run by the folks at C|Net, makes use of many different style techniques, including embedded style and tag style. This Web site's home page uses an embedded style sheet (see Listing 4-1) to establish parameters for the various types of information included in the page. The page itself a good example of how to combine scripting, embedded tables, and style sheets to create one great page.



We suggest that you visit the News.com page (found at <http://www.news.com/>) and view its source code to see the exceptional work. The code is clean, correct, well-written, and uses the best combination of HTML tags to combine several types of information into one coherent page. The content of the page changes daily, but the style sheet remains the same — an excellent example of how to use style sheets for pages that have frequently changing content. With styles already defined, all you have to do is update this page daily by replacing the actual information — the styles on the page remain consistent.

Listing 4-1

The C/Net Style Sheet

```
<STYLE>
<!@hy
.headline {font-family: Arial;
    font-size: 12pt;
    font-weight: bold;
    text-decoration: none;
    line-height: 100%;
    color: black
}

.headline2 {font-family: Arial;
    font-size: 11pt;
    font-weight: bold;
    text-decoration: none;
    line-height: 100%;
    color: black
}

.newsheadline {font-family: Arial Black;
    font-size: 18pt;
    font-weight: normal;
    text-decoration: none;
    line-height: 100%;
    color: black
}
```

```
.shorttakes    {font-family: Arial;
  font-size: 9pt;
  font-weight: medium;
  text-decoration: none;
  margin-left: 4pt;
  margin-right: 4pt;
  line-height: 12pt;
  color: black
}

.shorttakesbold {font-family: Arial;
  font-size: 9pt;
  font-weight: bold;
  text-decoration: none;
  margin-left: 4pt;
  margin-right: 4pt;
  line-height: 12pt;
  color: black
}

.text          {font-family: Times;
  font-size: 12pt;
  font-weight: normal;
  text-decoration: none;
  line-height: 13pt;
  color: black
}

.date          {font-family: Arial;
  font-size: 9pt;
  font-weight: medium;
  text-decoration: none;
  line-height: 11pt;
  color: black
}

.byline        {font-family: Arial;
  font-size: 9pt;
  font-weight: medium;
  text-decoration: none;
  line-height: 11pt;
```

(continued)

(continued)

```
        color: black
    }

    .menu {font-family: Times;
        font-size: 12pt;
        font-weight: normal;
        text-decoration: none;
        line-height: 14pt;
        color: black
    }

    .services {font-family: Times;
        font-size: 9pt;
        font-weight: normal;
        text-decoration: none;
        line-height: 14pt;
        color: black
    }

    .attribution {font-family: Arial;
        font-size: 9pt;
        font-weight: medium;
        text-decoration: none;
        line-height: 11pt;
        font-style: italic;
        color: black
    }

    .quote {font-family: Arial;
        font-size: 9pt;
        font-weight: bold;
        text-decoration: none;
        line-height: 10pt;
        color: black
    }

    .search {font-family: Arial;
        font-size: 9pt;
        font-weight: medium;
        text-decoration: none;
        line-height: 16pt;
        color: black
    }
```

```

.navbar    {font-family: Times News Roman;
            font-size: 10pt;
            font-weight: medium;
            line-height: 14pt;
            color: black
            }

.navbar2   {font-family: Times News Roman;
            font-size: 7pt;
            font-weight: medium;
            line-height: 14pt;
            color: black
            }

.jointext  {font-family: Arial;
            font-size: 10pt;
            font-weight: bold;
            text-decoration: none;
            line-height: 12pt;
            color: green
            }

A:visited  {color: green;
            }

A:link     {color: blue;
            }

@hy>
</STYLE>

```

Rather than alter existing HTML tags, this style sheet creates new style categories based on the type of information being displayed, whether it's a byline, attribution, a shorttake, or any of the other defined styles. In the following HTML excerpt from the same page, the shorttakes style is referenced using the `` tag and the `CLASS` attribute, and then modified using the `` tag.

```

<SPAN CLASS="shorttakes">

<FONT FACE="Arial" SIZE="-1">
NEWS.COM readers say free speech must be protected.

<A HREF="/SpecialFeatures/0,5,10190,00.html">Click here</A>
    for full results.

```


(continued)

```
<P>

</SPAN>

</FONT>
```

This code is a perfect example of using an embedded style sheet and a style tag together to implement a global style and then modifying it slightly to meet the needs of the page. The page has several different types of information, including headlines, shorttakes, and other specific content. The style sheets create a different style for each kind of information, providing the Webmaster with a quick, easy, and consistent way of presenting the information.

The Alertbox



The Alertbox is a semi-monthly article about the usability of the Web. The article is written by Jakob Nielsen, an engineer with SunSoft, the software division of Sun Microsystems. The Web page itself, located at <http://www.useit.com/alertbox/>, references an external style sheet called `useit_style.css` using the `<LINK>` tag in this bit of code:

```
<LINK TITLE="Useit House Style" REL=STYLESHEET HREF="/>
useit_style.css" TYPE="text/css">
```

View the page source to see this link for yourself, it's the first `<LINK>` tag after the `<META>` information. We dug around a bit and found the actual style sheet `useit_style.css` so you can have a good look at its innards (refer to Listing 4-2).

Listing 4-2

The Alertbox Style Sheet

```
BODY      {font-family: "'Times New Roman', Georgia, Times,
            'New York', serif" ;
            background: white;
            color: black;
            }

CODE      {font-family: "'Courier New', Courier, Monaco,
            monospace" ;
            }

SMALL     {font-family: "Verdana, 'Lucida Sans', Arial,
            Helvetica,
```

```

    Geneva, sans-serif" ;
    }

    HR {color: gray ;
    }

    .navbar {font-size: 100%;
    font-family: "Verdana, 'Lucida Sans', Arial, Helvetica,
    Geneva, sans-serif";
    background: #FFFF66 ;
    }

    .notrecommended {color: #666666 ;
    }

    .deemphasized {color: #666666 ;
    }

    TABLE {font-size: 90% ;
    font-family: "Verdana, 'Lucida Sans', Arial, Helvetica,
    Geneva, sans-serif" ;
    }

    TABLE.densetable {font-size: 80% ;
    }

    TR.summaryrow { font-size: 90%;
    background: #00FFFF ;
    }

    CAPTION {font-size: 100% ;
    }

    .embeddedfloat {float: right; margin-left: 3em
    }

    H1, H2, H3, H4, H5, H6 {font-family: "Verdana, 'Lucida
    Sans', Arial,
    Helvetica, Geneva, sans-serif" ;
    }

    UL.referencelist, .footnote {margin-left: 4em;
    text-indent: -4em ;
    font-size: 83%;
    font-family: "Verdana, 'Lucida Sans', Arial, Helvetica,
    Geneva, sans-serif";

```

(continued)

(continued)

```

list-style: none ;
}

A:link, .simulatedlink {color: blue ;
}

A:visited {color: purple ;
}

A:active {color: aqua ;
}

A.notrecommended:link {color: #6666FF ;
}

A.notrecommended:visited {color: #CC66CC ;
}

A.deemphasized:link {color: #6666FF ;
}

A.deemphasized:visited {color: #CC66CC ;
}

.overline {font-family: "Verdana, 'Lucida Sans', Arial,
Helvetica,
Geneva, sans-serif";
margin-bottom: -2ex ;
}

.updatecomment {font-size: 100%;
background: #00FFFF ;
}

.outsidecomment {font-size: 100%;
background: #E9E9E9 ;
}

```

The preceding style sheet uses a combination of altered standard tags such as `<CODE>`, `<SMALL>`, and `<CAPTION>`, as well as several specific styles that include `navbar`, `notrecommended`, and `embeddedfloat`. The navigation bar is really a table with the assigned style class, as this bit of markup shows:

```

<TABLE BGCOLOR=FFFF66 BORDER=1 COLS=1 WIDTH="100%"
CLASS=navbar>

```

To find out more about tables, be sure to read Chapter 2.

The page also incorporates tag-based styles (those that use attributes such as BGCOLOR and FONT to define style) and tags defined in the style sheet that work together, as shown in this code snippet:

```
<FONT FACE="Verdana, Lucida Sans, Arial, Helvetica, Geneva,
    sans-serif">

<SMALL>

How you can subscribe and
<A HREF="subscribe.html">get update notifications</A>
when a new Alertbox goes online

</SMALL>

</FONT>
```

, a standard HTML tag and attribute, supplies the font face, and <SMALL>, which was redefined in the referenced style sheet, changes the size of the text. This example shows how a tag-based style and a style defined by a style sheet can work together to create a combination style for a text block.

You can define and combine styles in many different ways. Planning your page layout and style sheets before actually implementing them is important. In the preceding example, the designer could have redefined the <TABLE> tag to meet the navbar specifications, rather than defining navbar and applying it to the <TABLE> tag. However, he would have to create a separate style specification each time he wanted to create a new table that wasn't a navigation bar. Better to leave a much-used tag alone and create classes to use when the standard style just won't do.

The Richmond Review



This literary magazine from the UK is an entirely Web-based publication. Even though it is not distributed in print, the magazine must still maintain the same consistency and standards to which print publications adhere. To maintain this consistency, the *Richmond Review* uses external style sheets, as well as embedded style sheets. Point your browser at this URL to visit the *Richmond Review*:

```
http://www.demon.co.uk/review/
```


Volume II, Issue #4 was the current edition posted to the site when we wrote this chapter. But regardless of the content, the external style sheet remains the same from issue to issue. View the source code and again you see the inevitable <LINK> tag, referencing the governing style sheet (style0.css in this case). The HTML code looks like this:

```
<LINK REL=STYLESHEET TYPE="text/css" HREF="style0.css"
      TITLE="style0">
```

Once again, we tracked down the actual style sheet; Listing 4-3 shows what we found.

Listing 4-3

The Richmond Review Style Sheet

```
BODY      { font-family: "Times New Roman";
            background: white;
            color: black;
            border: black;
            margin-left: 10%;
            margin-right: 10%;
            }

DIV.header { color: purple;
            font-family: "Arial";
            font-size: 85%;
            }

DIV.content { font-family: "Times New Roman";
            color: black;
            font-size: 100%;
            }

DIV.footer { color: purple;
            font-family: "Arial";
            font-size: 85%;
            }

DIV.footer A:link { font-family: "Arial";
                   font-size: 85%;
                   }

CITE      { font-weight: bold }

HR        { color: green
            }
```

```

HR.sub      { color: navy
              }

A:link      { color: navy;
              font-weight: extra-bold
            }

A:visited   { color: maroon;
              font-weight: bold
            }

A:active    { color: red;
              font-weight: light
            }

H2          { align: center;
              color: teal;
              font-family: "Arial";
              font-size: 120%
            }

H3          { align: center;
              color: teal;
              font-family: "Arial";
              font-size: 110%
            }

SPAN.mailto { color: green
              }

SPAN.isbn   { font-weight: bold
              }

```

In Listing 4-3, you see several classes defined for the same tag, such as the `DIV.header`, `DIV.content`, and `DIV.footer`, or the `SPAN.mailto` and the `SPAN.isbn` classes. These classes are all referenced in various places within the actual HTML document. The header class is used right after the `</HEAD>` tag to begin the document:

```

<HEAD>

<DIV CLASS=header>

<BODY>

```

(continued)

```

<H1 ALIGN=CENTER>
<IMG SRC="images/banner.gif" WIDTH="460" HEIGHT="50"
  ALT="The Richmond Review"></H1>

<H2 ALIGN=CENTER>
Vol. II, Issue #4

</H2>

<P>

</DIV>

```

The footer class ends the document and contains document and author information:

```

<DIV CLASS=footer>

<HR>

<P ALIGN=CENTER>

<A HREF="features/index.html">Features</A> |
<A HREF="library/index.html">Library</A> |
<A HREF="reviews.html">Reviews</A> |
<A HREF="scripts/search.html">Explore</A> |
<A HREF="rattler.html">RattleBag</A> |
<A HREF="http://www.bookpages.co.uk/twist/twist.plx?
  form=home.htx&SID=9">Bookshop</A> |
<A HREF="whoweare.html">WhoWeAre</A>

<P ALIGN=CENTER>

<CITE>The Richmond Review</CITE>

```

```
<P>  
  
</DIV>
```

The externally referenced style sheet is not the only source of style information this page draws upon. Embedded style information is also included with `<STYLE>` markup within the `<HEAD>` tags.

```
<STYLE TYPE="text/css">  
  
<!--@hy @import url(style0.css);-->  
  
H1 { color: blue }  
  
BODY { background: white } @hy>  
  
</STYLE>
```

Notice that the externally referenced style sheet has also been imported into the `<STYLE>` tag, so all the information contained in the external style sheet is effectively contained in this embedded style sheet, even though it is not listed line-by-line. The `H1` and `BODY` styles have been added to the style sheet for this page only. The `BODY` background specification is redundant because it is also included in the imported style sheet, but the `H1` information is new and adds to the imported specification.

This *Richmond Review* Web page shows how embedded and external style sheets can work together to create a consistent look and feel for a group of pages, but still retain page-level control. The way style sheets have been created, you are not forced to pick one way or the other, but can utilize all three methods for defining styles in your pages.

Finding More Style Sheet Information

The examples and preceding style syntax and linking sections give you solid information on how to create and implement style sheets in your pages. Many other Web resources contain information and links that you may find useful as you create your own style sheets. One useful site is the Web Design Group's Cascading Style Sheets Guide, which provides a tutorial and in-depth discussion of all aspects of style sheet design and implementation. They also have a great HTML tutorial that we have bookmarked. Here's the URL:

```
http://www.htmlhelp.com/reference/css/
```


As always, we can't overemphasize how useful the World Wide Web Consortium site is. For good measure, here is the URL one more time:

```
http://www.w3.org/pub/WWW/Style/
```

That's it for style sheets, as if it wasn't enough. After you create your first style sheet, the rest will be easy. The current full-release versions of Netscape and Internet Explorer support some parts of style sheets but not all. The newest versions, 4.0 for both, which are still in beta, will have full style sheet support, and you'll be seeing style sheets more and more on the Web. Be one of the pioneers and add them to your site today.

Chapter 10

Java Still Jams!

In This Chapter

- ▶ Looking at a Java overview
 - ▶ Creating content with Java
 - ▶ Installing and using Java-enabled browsers
 - ▶ Java-fying your Web pages
 - ▶ Programming do's and don'ts in Java
 - ▶ Finding top Java resources online
-

In the words of Sun Microsystems, Inc., Java's developer, Java is "a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithreaded, and dynamic language." Those of you who aren't walking dictionaries should read on for an explanation of those terms. In its own perverse way, this statement is incredibly accurate, although completely overcome by jargon!

A Java Overview

"Way cool. Stratospheric. White hot. Sizzling." The Web community describes Java with these words. What's so great about this object-oriented language with the funny name? Its multimedia capabilities, for one thing. Many people see Java as the most exciting avenue for bringing interactivity to the Web.

The unique features that Java offers, in combination with browsers such as Netscape Navigator and Internet Explorer, bring to the Web a form of real-time multimedia interactivity that has previously been available only from CD-ROMs. Instead of following hyperlinks, you get information in the form of moving pictures, audio, and video when you surf the Net with a Java-friendly browser. If you're looking for hotel information online, Java lets you walk through rooms on-screen to decide which one you like. You can plan stock or other financial portfolios with live data. And online shopping changes dramatically.

Java's specific features include the following:

- ✓ Cartoon-style animation
- ✓ Music that plays in the background while a user loads a Web page
- ✓ Inline sounds that play in real time while a user loads a Web page
- ✓ Real-time video
- ✓ Multiplayer interactive games

We've heard people compare Java's potential impact to the advent of the spreadsheet for PCs or the development of Mosaic, the first graphical Web browser. Java's future is regarded brightly in areas as diverse as electronic commerce, software distribution, and gaming. And as a bonus, Java is one of the most secure and virus-resistant languages ever created.

What makes Java different?

Java applets (small Java programs that you can include in HTML documents) are Java's most radical and brilliant creation. Applets greet you at Java-enhanced sites. Instead of viewing a table of contents at a Web site, you may encounter an applet showing a talking head that smiles, laughs, blinks its eyes, and tells you the site contents. As long as you're using a browser that supports Java, downloading specific software to run an applet isn't necessary; the Java environment in your browser automatically picks up the applet and runs it from your desktop.

This last feature sets Java apart from other languages. With a standard Web browser, you can't use a new content type (such as a special image format or game protocol) until the browser is updated to include that new type.

In stark contrast, Java compatibility is a feature that any browser is able to implement. Java sends the browser the requested content and the program needed to view it. Another bonus for you Web authors is that you no longer have to wait until a browser implements support for certain file types to use these types on your Web sites — instead, you can write the code yourself (in Java) and send your file to any user who requests it!

One of Java's most stellar features is its platform independence. A Java application can run on any platform and requires only that a single copy of the master version of that application be stored in a controlled location (on a Web server somewhere on the Internet). This process is truly revolutionary because it transforms the Net into a software distribution system and changes your Web browser into a platform capable of running an infinite variety of software.

Therefore, when you request a Java application over the Internet, the application runs on your local machine — without having to consider what type of hardware or software you use. This omniplatform capacity is a boon for system administrators as well because they need to worry about revising and controlling access to only one copy of the code. A Java-capable user can take advantage of applets from anywhere and everywhere.

Yet even among developers, Java's real power isn't fully realized. Professional developers can exploit Java's thundering power by creating tool kits that sit on top of raw Java. In turn, these tool kits should enable users to create powerful applications that are customized to their needs.

About those terms . . .

Now that we helped you recover from the original Java gibberish that we quoted in the first sentence of this chapter — and we hope that you are just a little excited about its potential — we want to explain those terms. Remember, we said that Java is “a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithreaded, and dynamic language.” In the interests of clarity, the following sections tell you what that *really* means.

Simple

Programmers call Java both *simple* and *elegant*. Java syntax eliminates the unnecessary and esoteric elements that plagued C++. For that reason, many people think that Java will ultimately replace C++ as a programming language of choice.

Object-oriented

Every Java class is directly or indirectly descended from the Object class — the grandfather of all Java classes. (*Class* refers to a method for defining a set of related objects that can inherit or share certain characteristics.) Likewise, everything in Java must be an object, unlike in C++. Only the most basic, primitive Java operations and data types (such as `int`, `while`, `for`, and so on) operate at a subobject level. In Java-speak, every class *extends* directly or indirectly from the Object class, making Java *object-oriented*.

Distributed

From its inception, developers designed Java to be run over a network (to be distributed). Consequently, Java features well-developed, inherent HTTP and TCP/IP capabilities. CORBA-style Interface Definition Language bindings grace Java; in programmerspeak, this statement means that Java programs are able to invoke remote procedures over a network from server-based

objects. And in something closer to English, this explanation means that Java programs are able to request remote services across the network, with some reasonable expectation that those requests will be fulfilled.

Interpreted

Java's unique, hard-core compiler converts any Java source code into a machine-independent format. This advanced conversion enables the source to run on any computer that is equipped with a Java Virtual Machine (a special platform-specific program that implements a Java runtime system, allowing the machine to interpret Java code).

The *bytecodes*, which result from the conversion, are the key to Java's impressive application portability. Although this portability is a groundbreaking feature, Java isn't completely different from other programming languages. That is, programmers who are well-versed in Java write application code and save it in files with a `JAVA` extension. These files, or programs, are then translated into a format that enables your computer to run them.

Robust

Java is considered *robust* because its design completely eliminates *pointer* manipulation from the language (pointers are names or locations of specific memory addresses and are often used by programmers to play games with memory allocation and data access, both of which can be fraught with peril). Eliminating this element removes a big source of runtime errors. Although pointer abstraction in languages such as C and C++ can benefit seasoned programmers, it can also cause memory leaks and other obscure runtime errors.

Another of Java's winning features is its garbage collection mechanism that remembers to deallocate memory. After an object is *thrown away* in the Java runtime environment (that is, after it has been used and is no longer needed in a program), other objects can reclaim and reuse the memory vacated by the thrown-away objects.

Secure

Java is *secure* because it is a client-side technology. Java programs are, in effect, downloaded from a host machine to your client machine by using conventional Web server transports (such as HTTP). Java uses safety precautions to prevent viruses from reaching your machine; before running any applet, the Java runtime system looks at incoming Java bytecodes to make sure that the program code is safe. Any code that is not safe is unequivocally rejected. Although this feature may be the safest thing going, that distinction attracts renegades who revel in code-cracking, so don't count on 100 percent safety. Still, Java's security should only improve as the language matures.

Architecture-neutral and portable

Java's byte-ordering issues (that is, the order in which it interprets bytes in memory when handling numbers and values) are negligible. Also, its bytecodes operate independently of any underlying architecture (the *architecture-neutral* feature means that it doesn't depend on hardware support to represent data). In addition, Java's character sets are Unicode-based for internationally *portable* applications. Today, the portable Java runtime system is available on a wide variety of systems, including Windows 95 and Windows NT, Solaris, HP/UX, SunOS 4.1.3, AIX, OS/2, and Digital UNIX (OSF/1). This availability means that the same program will run on all these operating systems (and their associated hardware) without requiring any modification.

High performance

To call Java a *high-performance* language may be slightly premature, because Java code seems to run 20 to 30 times slower for some CPU-intensive programs. The promise of high performance is definitely apparent, though. Sun Microsystems is committed to making Java's on-the-fly performance rival that of native C or C++ applications. Sun Microsystems' development of a just-in-time class compiler, which is able to provide more dynamic runtime support for Java's unique capabilities, is helping to improve Java performance across the board.

Multithreaded

Multithreaded means that Java can run multiple subtasks (called *threads*) within a single large application. Multithreading usually helps interactive performance. Java can create and manage its own threads and is inherently multithreaded; that is, Java has certain threads (such as the garbage collection system for reallocating memory) that run in low-priority mode.



Java borrowed the idea of synchronized keywords from the Mesa programming language and the Xerox Cedar environment. When code is wrapped with the *synchronized* keyword, its objects and methods may only be accessed by one object at a time. In other words, the *synchronized* keyword makes any associated code single-threaded, rather than inherently multithreaded (the normal Java default).

Dynamic

Java lets you patch applications *dynamically*, so (unlike patching in C++) you have to worry much less about recompilation. Because releasing a perfect piece of software is rare, making adjustments to your application at some point is practically inevitable. Java makes these adjustments much easier by deferring lots of its linkage manipulation until runtime, which basically eliminates C++'s superclass problems.

That's it for the gibberish, but aren't you glad we told you all about Java? If you can simply appreciate the incredible thought and effort that's gone into Java's design and development, we've succeeded in delivering the right message!

Java's history: from seedling to coffee bean

Java came out of the fertile breeding grounds of Sun Microsystems. The growing process began when a disgruntled employee, Patrick Naughton, told Sun Microsystems' chief Scott McNealy that he was leaving Sun Microsystems to go to NeXT Computer, Inc., because NeXT was doing more interesting work. McNealy asked Naughton to tell him (McNealy) what he (Naughton) thought was wrong with Sun Microsystems before leaving.

The list of suggestions that Naughton created was well-received by Microsystems executives, who gave him, along with a team of coworkers, carte blanche to pursue new research projects. The team code-named themselves Green and set to work on developing the fundamental elements of what would eventually become Java. They initially thought the key to success lay in the area of consumer electronics (such as interactive TV). Finally in 1994, after a series of disappointments, they realized that the Web was Java's ideal medium.

In the early days, the development team's metaphors were more organic than caffeinated; not only was the team named Green, but the language that emerged into Java was named Oak (because James Gosling, its creator, gazed on an oak tree from his office). But by 1995, Oak was renamed Java, and Naughton had written a Java interpreter for a Web browser called HotJava.

In May, Sun Microsystems formally announced Java and HotJava at its SunWorld '95 exposition. Netscape, always interested in cutting-edge technology, simultaneously announced its intention to license Java for Navigator 2.0. The buzz was thick from then on, because Java's potential to turn the Web on its ear (and garner a great deal of profit) was obvious. Sun Microsystems agreed to make Java and HotJava available free of charge, banking on the idea of making money by licensing the runtime code and Java development tools for commercial use.

Licensees must pass conformance tests in order to license Java and receive branding from Sun Microsystems, Inc. The purpose of this testing is to ensure that applets work on all branded products. Current Java license owners (besides Netscape) include Macromedia, Inc., Novell, Spyglass, Sybase, Symantec, Mitsubishi Electronics, IBM, Microsoft, and Adobe.

Even with such illustrious licensees, whether Sun Microsystems will make loads of money from Java is unclear (although Java's reputation has already helped Sun Microsystems' stock rise). One thing is sure — the Sun Microsystems' plan to make Java an essential programming language and a crucial part of the Web has already worked.

The 1995 release of JavaScript, a more accessible scripting language that nonprogrammers can use, also increases Java's market power. Microsoft's introduction of its own J++ Java development environment, and its increasingly heated endorsements of Java technology have also helped to legitimize Java in the minds of the largest of all development communities on the Web — the so-called "Wintel" (Microsoft Windows/Intel) developers.



This site contains a timeline of Java's evolution through the end of 1995:

<http://ils.unc.edu/blaze/java/javahist.html>.

Java's learning curve

As far as developers are concerned, Java's learning curve is minimal because its syntax is close to C++. However, for mere mortals uninitiated into the complex wonders of C or C++, Java is a complicated — and largely unnecessary — language to learn.

Unlike HTML, Java wasn't designed to be particularly accessible, and you don't really need to know Java just to write for the Web. Writing in Java can be compared to the more complicated and professional realm of creating CGI (Common Gateway Interface) programs.



The *lite* version of Java, called *JavaScript*, is much more viable than Java for most ordinary HTML authors to use (and Chapter 11 of this book covers this subject in brief). Sun Microsystems has also voiced its intentions to develop tool kits for nonprogrammers as well, and interesting development environments are also available from Symantec (Visual Café) and Borland (Latte).

The bottom line is: Nonprogrammers needn't worry about learning Java to enjoy its benefits. The only thing you need to experience Java's full-bodied flavor is a Java-enabled Web browser (such as a recent version of Netscape or Internet Explorer) that lets you install a Java plugin, which delivers Java's runtime interpreter. For those of you who are interested in programming in Java or JavaScript, read on for the exciting details.

Java versus C++ — the differences

Java features the following:

- ✓ **Single inheritance:** Keeps object relationships simple and easy to follow.
- ✓ **Garbage collection:** Reclaims available memory and lets Java applications run more efficiently.
- ✓ **Native multithreading:** Lets Java applications “divide and conquer” processing tasks, which also improves efficiency.

C++ features these elements:

- ✓ **A preprocessor:** Enables programmers to work with useful abstractions, without building them directly into the language.
- ✓ **Operator overloading:** Lets programmers play all kinds of tricks with variables and memory; although risky, this technique can provide substantial power when properly applied.
- ✓ **Header files:** Enables programmers to invoke standard collections of type definitions, variables, and library elements.
- ✓ **Pointers:** Lets subroutines and modules move addresses around, rather than values, which can speed up program behavior.
- ✓ **Multiple inheritance:** Lets programmers combine aspects of existing objects at will; although often confusing, this process is also incredibly powerful.

Creating Java Content

For nonprogrammers, creating Java content isn't nearly as easy as creating Web page content with HTML. But if Java's seductive programming power is too appealing for this information to daunt your interest, we suggest that you do some research before plunging into programming.



The following URL takes you to the Java Message Exchange, where you can ask questions and read about other people's experiences with the language:

<http://porthos.phoenixat.com/~warreng/WWWBoard/WWWBoard.html>

Sun Microsystems has a site that features helpful Java tutorials and covers topics such as writing Java programs, writing applets, creating a user interface, and much more:

<http://java.sun.com:80/tutorial/index.html>

Another excellent way to find out about Java is to look at Java source code wherever you can. Study and mimic examples of Java that you find on the Web — imitation is not only the highest form of flattery, but also a great way to understand a subject.

Unfortunately, Java source isn't as easy to see as HTML: You can't simply hit a View Source button and instantly immerse yourself in Java (because applets move as bytecodes across the Internet, not as source code). To look at Java source, you need to find a Java-related newsgroup (such as `comp.lang.java`) or a mailing list, or surf the Web and grab all the code you can find. But don't despair — you can find plenty of Java out there!

About Java programming . . .

You can use any word-processing program or editor to create and edit a Java program, as long as the tool you use can save files in ASCII format. Just remember: You must save all Java programs in a file ending with the `JAVA` extension.



Before you run your Java program, you must compile it with the Java compiler, usually a program named *javac*. The compiler translates Java source code into *bytecodes*, an intermediate form of code that changes Java language statements to machine instructions.

The bytecode doesn't go the final step of adopting a machine language that corresponds to a specific operating system and computer; the Java runtime environment does that step. Completing this final step is Java's key to platform independence and the reason that Java is both *interpreted* and *compiled*. After you compile a program, though, you're ready to roll.



If you encounter any problems with compilation or interpretation of Java code, the following site's troubleshooting tips should quiet your woes:

<http://sunsite.unc.edu/javafaq/>

Say "Hello World": creating a Java program

Demonstrating programming concepts with a simple "Hello World" program has been customary for a long time. This program, which prints the string Hello World to the screen, demonstrates the mechanics of writing code and constructing a simple program.

Before you start, make a directory or folder called JAVAHTML and create a subdirectory or subfolder called CLASSES within that directory. (Calling the first directory JAVAHTML isn't actually required, but you do need to call the subdirectory CLASSES.)

The code that we show next is a "Hello World" application written in Java. To run this application, type the following lines into a text file and save the file with the name HelloWorld.java in the JAVAHTML directory.

```
class HelloWorld {  
    public static void main (String args[]) {  
        System.out.println("Hello World!");  
    }  
}
```

Compile this program by typing `javac HelloWorld.java` at the command prompt (make sure that you're in the JAVAHTML directory).

If, for some reason, this program doesn't work, check out the following troubleshooting tips:



- ✓ Did you include the semicolon (;) after `System.out.println("Hello World!")`?
- ✓ Did you remember the closing bracket?
- ✓ Did you type everything exactly as it appears in the preceding code lines? Precise capitalization is essential because Java is case-sensitive.

The compiler places the executable output in a file named `HelloWorld.class` in the JAVAHTML directory after the program is successfully compiled. At this point, you can run the program simply by typing `java HelloWorld` at the command prompt. The program (hypothetically) responds by printing Hello World! on the screen.

Finding and Using Java-Enabled Browsers

You need the following components to use Java:

- ✓ A Java-enabled Web browser: HotJava, Netscape 2.0 or later, Internet Explorer 2.0 or later
- ✓ A Java compiler that turns Java source code into bytecodes
- ✓ A Java interpreter to run Java programs
- ✓ A text editor, preferably programming-oriented, such as BBEdit or Brief
- ✓ A suitable operating system (Windows 95 or Windows NT, Mac/OS 7.5 or better, many flavors of UNIX)

The following components are nice to have for your Java adventure, but not crucial:

- ✓ A Java debugger
- ✓ A Java class browser
- ✓ A Java-capable visual development environment
- ✓ Java documentation, either online information or any of the many books now available on this subject



You can find elements of these components, along with detailed instructions for their use, at the Sun Microsystems page:

<http://java.sun.com/>

Before Java's technology emerged, the Web was confined to static pages filled with only images and text. However, now that Java's on the scene, you not only have the same features as a CD-ROM — animation, multimedia, and instant interactivity — but when you're plugged into a network using Java, you also have live data, communication, and instant updates. Java, working with the Web, is a great one-two punch for anyone who realizes the full possibilities of this technology.

Referencing Java Materials in Your Web Pages

Incorporating Java materials into your Web pages can endow them with a whole new look and feel, as well as new potential. By using applets, you can actually walk your viewers through a sequence that you want them to see. The following list gives you tips for effective use of Java applets in your pages:

- ✓ Read about Java before you start trying to program with it. You can find a number of helpful resources on the Web and a number of books on Java as well.
- ✓ Surf to see what's going on in Java programming. The list at the end of this chapter provides URLs for a number of interesting Java sites.
- ✓ Select an applet that you would like to emulate. GAMELAN, at <http://www.gamelan.com/>, is a directory that offers many sample applets. Study the HTML code from the applets page and consult the code as you create your applet's parameters.
- ✓ Access an applet. You can't use applets in your pages unless you have the ability to access them! You have two paths of access: remote call and local compilation.
- ✓ Do some experimenting with `call` commands and figure out what HTML tag calls the applet for use on your pages. Directories called CLASSES are the repositories for all compiled applets.

To invoke a Java applet from inside a HTML document, use the HTML `<APPLET>` tag. The following code shows the general syntax for utilizing the `<APPLET>` tag:

```
<APPLET  
  CLASS="class name"  
  SRC="URL"  
  ALIGN="alignment"  
  HEIGHT="height in pixels"  
  WIDTH="width in pixels"  
  APPLET_SPECIFIC_ATTRIBUTES="values"  
...>
```

To preserve security, applets can't do certain things. The following list shows you some of the things that applets can't do:

- ✓ Make network connections to any host except the one where they originated.
- ✓ Read every system property.
- ✓ Define native methods or load libraries.
- ✓ Read or write files on the host that runs them.

If you think about these restrictions, most aim at preventing applets from violating system security or data integrity, primarily to discourage hackers from trying to use Java's normal behavior (of moving code from one system to another) for their evil purposes.

Java Design Do's and Don'ts

The following sections examine some areas to explore — and to avoid — when writing Java code.

Do

Do the following things when you design Web page elements with Java:

- ✓ Do make sure that your applet stops running when it's off-screen. If the browser is iconified (minimized) or displaying a page other than the one containing the applet, the applet generally shouldn't be running — doing so drains CPU resources. Your applet can run off-screen only if your applet code doesn't explicitly launch any threads.
- ✓ Do implement the `stop()` method if your applet code launches any threads. The `stop()` method can halt and destroy the threads you've launched by setting them to `null`. For example:

```
public synchronized void stop() {  
    if (refresh != null) {  
        public synchronized void stop() {  
            refresh.stop();  
            refresh = null;  
        }  
    }  
}
```

- ✓ Do give your users a way to stop an applet's behavior, especially when the applet does something that is potentially annoying, such as playing sounds (think of all those people secretly surfing at work!).

One way to stop an applet's behavior is to implement the `mouseDown()` method (in an applet that normally doesn't respond to mouse clicks) so that clicking the mouse suspends or resumes the thread. The following code shows one approach to using the `mouseDown()` method:

```
boolean threadSuspended = false;
    //an instance variable
public boolean mouseDown(Event e, int x, int y) {
    if (threadSuspended) {
        myThread.resume();
    } else {
        myThread.suspend();
    }
    threadSuspended = !threadSuspended;
    return true;
}
```

Don't

Don't do the following things when you design Web page elements with Java:

- ✓ Don't forget to remove or disable debugging output. Debugging output (usually created with `System.out.println()`) may be useful to you, but it may annoy or confuse users. To give your users feedback, try the status area at the bottom of the window or inside an applet's display area.
- ✓ Don't forget to end source URLs in your CLASSES directory with a slash.

Lots of other do's and don'ts exist for doing Java right. Read on for some excellent resources on this topic.

Top Java Resources

Not surprisingly, Sun Microsystems has an excellent collection of Java resources at java.sun.com/. In addition to that page, the following Web resources contain helpful and interesting information about Java:

- ✓ The Java Message Exchange is full of questions and answers:

<http://porthos.phoenixat.com/~warreng/WWWBoard/wwwboard.html>

- ✓ Gamelan is an excellent resource for Java applets:

<http://www.gamelan.com/>

- ✓ John December offers an impressive collection of Java information:

<http://www.december.com/works/java/info.html>

- ✓ This page contains useful information about Java, as well as downloads and links:

<http://sunsite.sut.ac.jp/java/>

- ✓ The following is a great page of Java links:

<http://www.nebulex.com/URN/devel.html>

- ✓ Sun Microsystems' helpful Java tutorials cover topics such as writing Java programs, writing applets, creating a user interface, and many more:

<http://java.sun.com:80/tutorial/index.html>

- ✓ Café Au Lait is a superlative resource listing:

<http://sunsite.unc.edu/javafaq/>

- ✓ If you have compiler or interpreter problems, this site is full of troubleshooting tips:

<http://java.sun.com/tutorial/troubleshooting/index.html>

- ✓ The IDG JavaWorld page is an exhaustive Java resource:

<http://www.javaworld.com/>

Java's Future

The idea that Java can automatically improve the quality of Web applications is not a given. Without vision, Java's applets alone can't push Web users into a higher realm of surf-nirvana. However, Java's potential is undeniable, and its possibilities in terms of multimedia and interactivity are staggering. If used well by developers, Java permanently enriches the Web.

Whether development continues with Java or something newer and more exciting comes along, you can't deny the growth and progress that Java has stimulated for the Web. Plus, Java is the world's greatest excuse for lame, caffeine-inspired puns! In the next chapter, we leave Java behind and move into the less rarefied, but more accessible, world of HTML scripting languages, including Java's cousin, JavaScript.

Chapter 11

Scripting Alternatives

In This Chapter

- ▶ Understanding scripting languages
- ▶ Jumping into JavaScript
- ▶ Venturing forth with VBScript
- ▶ Doing the Dynamic HTML thing
- ▶ Exploring new scripting alternatives

Somewhere between the basic capabilities in HTML markup that you know and love and the rich but complex possibilities conferred by CGI and Java programming lies a middle ground. This halfway point is also rife with possibilities to extend your Web site's look, feel, and level of interactivity.

Welcome to the Scripting Zone, an area of Web extension where you'll type bizarre and sometimes incomprehensible text to make your Web pages stand up and do tricks. Like HTML, scripting languages leave traces that anyone who views the source code for a Web page can see. But like the languages you'd use for CGI programs or Java applets, scripting languages confer more power and capability on their makers than does plain, unadulterated HTML.

Most experts believe that power and complexity are related for scripting languages, so it's important to understand that they're not truly general, nor infinitely extensible, as are real programming languages. However, they're also somewhat easier to learn and use than a full-blown programming language.

Best of all, most scripting languages (including all of those in this chapter) include lots of built-in functions, toolbars, widgets, interfaces, and other prefabricated goodies so that mere mortals who use a scripting language need do very little low-level, nuts-and-bolts programming. Instead, these tools use the built-in facilities of a scripting language as a set of building blocks to assemble surprisingly sophisticated extensions to Web pages at the cost of very little programming effort. Perhaps that's why scripting languages are quite hot right now—and it also should explain why this topic is in this edition of *MORE HTML For Dummies*!

Although each of the scripting languages discussed in this chapter has its strengths and weaknesses, any of them will provide a significant increase in what you can do with your Web pages, both as a page designer and as a content author.

The Mechanics of Scripting

All scripting languages rely on the presence of a special program, called an *interpreter*, to read the contents of a script while a document is open, and to take action based on that script's contents, immediately after each line is read. Because scripting relies on interpretation while the script is read, script languages depend on something called a runtime system, which means that the script is completely decoded and executed while it's running on a computer.

This system is in direct contrast to programming languages like C or Pascal, where a program called a *compiler* reads and interprets programs in advance. Values can be read or substituted at runtime from an input file or from user interaction, but compilation of the bulk of a program's code greatly reduces the amount of processing.

For this reason, interpreted languages, such as those used for Web scripting, run more slowly than compiled languages, because more processing is necessary to handle the code at runtime. That's also why you'll often see adjectives such as fast, efficient, or lightweight applied to scripting languages, because they indicate that such languages have been optimized to behave well at runtime, to require little in the way of system resources, and to impose no significant additional computing burden on the machines where they run.

Scripting languages are also sometimes called *embedded languages* because they usually appear within text for some other kind of language or text markup. For example, all the scripting languages covered in this chapter are built to appear within HTML documents, marked by the presence of the `<SCRIPT>` tag.

When the browser recognizes a `<SCRIPT>` tag, the following process begins:

1. The browser looks for a value for the LANGUAGE attribute (which may be something like JavaScript, VBScript, or Jscript).
2. Based on the value of the attribute, the browser invokes a runtime interpreter for that scripting language and passes control to the interpreter as soon as it's identified.

3. The interpreter takes over and executes the script until it reaches the closing `</SCRIPT>` tag.
4. The interpreter returns control to the browser, which continues with its job of HTML interpretation and rendering.

Because HTML is purely character-based, any scripting language that's embedded within HTML must share that character basis. This character basis means that you can use any text editor to create scripts for inclusion on your Web pages. But in many cases, particularly for VBScript, you'll find a variety of authoring tools that can help you build scripts. (In some cases, tools are available to help you build the HTML documents within which such scripts must occur.)

Programming with JavaScript

JavaScript isn't the same as Java, although the two are closely related. Created as a joint venture between Netscape and Sun, JavaScript is sometimes jokingly called *Java Lite*. (It was named *Mocha* before release — tired of the coffee metaphors yet? Keep reading.) JavaScript is a compact, object-based scripting language used to develop client and server applications. What can it do for you? Well, JavaScript provides a faster and more accessible means (than Java) to incorporate applets and other small programs into your Web site.

JavaScript is frequently regarded as Java's decaf version, which isn't wholly accurate because JavaScript addresses a different market niche. Dynamically typed languages (which permit type definitions to be added at will) that are less complex and more compact than Java, such as dBASE and HyperTalk, are considered to be JavaScript's real predecessors. They, like JavaScript, appeal to a wider audience than a highly specialized language like Java. Their accessibility owes to certain qualities: easy syntax; specialized, built-in functions; and minimal requirements for object creation and use.

JavaScript was explicitly developed for use with Netscape Navigator 2.0, where it provides a set of client-side APIs (Application Program Interfaces). These client-side APIs address the nuisances of static Web text. When JavaScript statements are embedded in an HTML page, they can then recognize and react to mouse clicks, page navigation, and `<FORM>` input. Although JavaScript lacks Java's strong type-checking and static typing mechanisms, it still supports the majority of Java's basic control flow constructs and expression syntax.

When you're working with JavaScript, you needn't use (or know anything about) classes. Instead, you acquire finished script components from a JavaScript resource (perhaps from a commercial product or an online collection of such materials) that make high-level properties (like `color` or `visible`) susceptible to scripting controls. As a result, you can set these properties to achieve the effects you desire in a Web document. JavaScript may offer the next best thing to writing code, and its capabilities enable many Web authors to include capabilities they wouldn't be able to include otherwise.

Differences between Java and JavaScript

JavaScript scripts run more slowly than Java applets because JavaScript interprets each line separately as it reads it from the HTML source file, but Java applets are immediately executed by the Java runtime environment within a browser.

JavaScript doesn't possess all the object-oriented features that Java has; in JavaScript, you can define objects but not object classes. JavaScript also lacks Java's object inheritance. Neither of these omissions is too significant for an average JavaScript user, because JavaScript's shorter scripts don't generally have the same reuse requirements that objects in Java do. In general, these omissions mean that JavaScript writers must content themselves with whatever predefined objects and attributes are available, rather than the richness and flexibility of full-blown Java, especially when it comes to recasting or tweaking objects to add functionality. But because JavaScript has been created to reduce complexity and speed development, anyone who needs to do extensive work with objects will be better served by working with Java directly, rather than using JavaScript.

Because developers created JavaScript specifically to enable HTML writers to support different HTML tags and to allow elements to interact with each other, input from one HTML form can influence HTML information inside another HTML page. In addition, JavaScript may change the text of the HTML page where it resides. By contrast, Java applets don't usually interact with HTML or change the text on a Web page. Instead, Java applets limit themselves to a specific region of a page that's reserved for their exclusive use (much like a separate display frame).

Also, JavaScript can deliver interactive Web forms more efficiently than Java. JavaScript enables you to handle error-checking with built-in event-handling mechanisms that are conveniently based upon user interaction with an HTML form.

The following lists highlight the main characteristics of Java and JavaScript. Comparing these lists should help you understand the differences between the two languages. The first list describes Java characteristics:

- ✓ Java code is compiled on a server before it's executed on the client.
- ✓ Java has strong typing requirements, which means that the program code must declare variable data types.
- ✓ Java applets consist of object classes with inheritance.
- ✓ Java has *static binding*, which means that its object references must exist when the Java code is compiled.

The following list describes JavaScript characteristics:

- ✓ The client interprets (it doesn't compile) JavaScript.
- ✓ JavaScript code is both integrated and embedded in HTML.
- ✓ JavaScript has loose typing, which means that the JavaScript code need not declare variable data types.
- ✓ Although JavaScript code uses built-in objects, it doesn't have classes or complete object-orientation.
- ✓ JavaScript has *dynamic binding*, which means that its object references are checked at runtime.

JavaScript syntax and semantics

The meat of any JavaScript program lies between the `<SCRIPT>` and `</SCRIPT>` tags, which define the program's starting and stopping points. `<SCRIPT>` marks the point where the browser hands control to a JavaScript interpreter, which reads all the lines in the HTML file until it encounters the `</SCRIPT>` tag. Between those two HTML tags is where the syntax and semantics of JavaScript truly reside.

The following line is an example of JavaScript code:

```
document.write("This text should appear before  
the body of the HTML page.<P>");
```

In this block of text, an object named `document` is being accessed, and its `write` attribute is being applied to the text inside the parentheses between the quotation marks.

This example illustrates the `object.attribute` syntax which defines the way that JavaScript statements work. The semicolon (;) marks the end of a statement to tell the interpreter that it's complete. Danny Goodman's *JavaScript Bible* (see the reference information in the next section) covers the syntax in great detail, and includes a terrific Quick Reference in Appendix A.

Viewing source code is a good way to figure out how to use JavaScript, just as it is when you're trying to learn Java. Examine the following JavaScript code:

```
<HTML>
<HEAD>
<TITLE>My First JavaScript Page</TITLE>
<SCRIPT LANGUAGE="Javascript">
<!-- The SCRIPT tag indicates the beginning of a
script; the value supplied for the LANGUAGE
attribute indicates that it's written in JavaScript.
Notice also that the script occurs in the HEAD
section of this particular HTML document.
-->

<!--
document.write("This text should appear before the body
of the HTML page.<P>");
// -->
<!-- The preceding HTML comment is used to "wrap"
JavaScript code so that browsers that don't under-
stand JavaScript will not display code that they
can't act upon. This is a common scripting trick.
Notice also that the syntax uses the document
object's write method to output literal text
(what's between the double quotes) that includes genuine
screen output, plus HTML tags.
-->

</SCRIPT>
</HEAD>
<BODY>
<H3> Check out my first JavaScript Web page! </H3>
</BODY>
</HTML>
```

JavaScript resources

As with most other Web-related technologies, you can use a search engine like Yahoo!, Excite, or AltaVista to produce gobs and gobs of JavaScript information by submitting JavaScript as a search string. The following are some of the best JavaScript resources:

- ✓ The Java Authoring Guide covers a range of information on JavaScript:

[http://home.netscape.com/eng/mozilla/2.0/handbook
javascript/index.html](http://home.netscape.com/eng/mozilla/2.0/handbook/javascript/index.html)

- ✓ The FAQ (Frequently Asked Questions) for JavaScript:

<http://www.his.com/~smithers/freq/beta/jsfaq.html>

- ✓ Danny Goodman's *JavaScript Bible*, 2nd Edition (IDG Books Worldwide, Indianapolis, IN, 1996, List price: \$39.99 with CD-ROM) is one of the best-ever resources on JavaScript. For information on this book, go to the following site:

<http://www.idgbooks.com>

There's certainly no shortage of resources on JavaScript, so dive right in if you're interested in learning more. Your Web pages — and your users — will thank you!

Programming with VBScript

Microsoft's full name for VBScript is Visual Basic Scripting Edition. VBScript is a subset of Microsoft's immensely popular Visual Basic development environment, a great favorite of software developers everywhere. VBScript has been implemented as a fast, portable, and efficient interpreter for use in Web browsers and other applications that can use ActiveX controls and Java applets. Support for VBScript is currently bundled with Microsoft Internet Explorer 3.0 (and higher-numbered versions) and with Internet Information Server 3.0.

What about Jscript?

Those of you who follow the latest trends are no doubt aware that Microsoft has floated yet another scripting language, called Jscript, for use with its Internet Explorer browser. So what's the difference between JavaScript and Jscript? And why should you care?

First, JavaScript originated with Netscape and Jscript originated with Microsoft, but the fundamental difference between the two is that JavaScript relies on underlying runtime mechanisms associated with its own interpreter and related mechanisms, but Jscript

depends on the presence of ActiveX Scripting support in the runtime environment. Practically speaking, this difference leads to the idea that it's better to use JavaScript where the primary audience for a site uses Netscape Navigator and use Jscript (or VBScript) where the primary audience uses Internet Explorer. Because a subset of JavaScript works with Internet Explorer, and Jscript works on Navigator when ActiveX support is loaded, this idea is not a cut-and-dried matter, but rather, a strong tendency.

Like JavaScript, VBScript programming relies on the use of the `<SCRIPT>` tag pair in HTML. The VBScript interpreter, which reads and reacts to the contents of the scripts that appear within HTML documents, handles everything between the opening `<SCRIPT>` and closing `</SCRIPT>`.

What VBScript supports that JavaScript does not is an intuitive, graphical interface for building and using scripts within HTML documents. For example, if you're working in an environment that supports Microsoft's ActiveX Control Panel, this interface means you can launch this tool and invoke all kinds of tools and wizards to help the process of scripting along at any point in building an HTML document.

VBScript is a strict subset of the Visual Basic for Applications language that's supported in Microsoft Excel, Project, Word, and Access, as well as in the Visual Basic 4.0 development system sold separately by Microsoft. VBScript is designed to be compact and low in overhead, for easy use across a (potentially slow) Internet connection. Because of VBScript's design characteristics, it does not use strict data typing, nor does it support file input/output or direct access to a machine's underlying operating system (these assumptions are wise for software that will run across a variety of platforms in any case).

When used in the Internet Explorer environment, VBScript acts just like JavaScript — in other words, it's based on a text interpreter that reads ASCII characters from within an HTML file invoked by the combination of the `<SCRIPT>` tag and the `LANGUAGE = "VBScript"` attribute. Like JavaScript, VBScript does not rely on stand-alone (or rather, stand-apart) applets like those used in pure Java. As with any scripting language, VBScript adds to the contents of the HTML documents in which it appears.

What's the scoop on ActiveX?

At this point, we've slung the term ActiveX enough that it cries out for definition:

ActiveX, which Microsoft once called Object Linking and Embedding, or OLE, is a collection of tools designed to bring audio, animation, and user interaction to Web documents. It is quite similar to the plugin technology developed for Netscape Navigator and used in Java applets. All three technologies make it possible to send small programs to a Web browser without requiring the involvement of any other special software on the desktop or back-end capabilities from a Web server. ActiveX supplies code from Microsoft that allows

developers to add dynamic or interactive components or controls to otherwise static Web pages. The general consensus is that ActiveX is Microsoft's competitive response to Sun Microsystems' Java technology (even though Microsoft itself has licensed Java, and offers its J++ Java development environment to its customers).

In short, ActiveX represents important Microsoft technologies necessary to permit Web and other documents to access persistent software objects and exchange information across a network.

Because VBScript can draw on a vast library of ActiveX controls, predefined buttons, toolbars, and other graphical widgets, it offers considerable capability when creating or defining interactive controls or onscreen forms for user input on Web pages. Because so many developers are already familiar with the VB development environment, they can leverage what they already know with VBScript.

The ActiveX Control Panel provides powerful document editing and a helpful graphical programming toolset that is useful when you're programming with VBScript. This tool makes it easy to extend existing HTML documents or author script-enhanced HTML documents from scratch, because of its powerful HTML layout controls and its menu-driven access to immense libraries of predefined functions, buttons, and other graphical objects. The ActiveX Control Panel is likened to "NotePad on steroids," in that it presents a text editor interface to the user, but mouse- and menu-driven development tools are never more than a few mouse clicks or keystrokes away. The ActiveX Control Panel includes Scripting Wizards that can help with script creation, an HTML layout tool that provides precise control over HTML page layout and object placement, and support for HTML authoring.

The only barrier to VBScript's path to world domination is that each VBScript script on a page forces a separate load of the VB runtime system. At around 3MB of memory per incarnation, this memory drain can quickly bring even serious systems to their knees. But performance limitations aside, VBScript offers serious capability and easy page extension to anyone with a modicum of interest in programming. Taken with the large number of books, tutorials, and explanatory materials available for Visual Basic and Microsoft's outstanding support for VBScript, it is a Web extension technology not only to be reckoned with, but worth considering for those environments where Internet Explorer is the dominant (or only) Web browser in use.

As you might expect, Microsoft's Web site provides one of the best points of departure for an investigation of VBScript's multifarious capabilities. Start at the VBScript home page, which you'll find at:

<http://www.microsoft.com/VBSCRIPT/>

From this site, you can obtain a copy of the latest FAQ, download the software and documentation, and obtain access to all kinds of related links and examples.

Dynamic HTML

Dynamic HTML is a new development in the area of Web scripting and HTML authoring. An initiative started by Microsoft, Dynamic HTML is now working

its way through the W3C to define a standard for inline scripting languages. Microsoft and its supporters dubbed this technology Dynamic HTML, but the W3C has labeled it the Document Object Model (DOM).

The W3C's efforts are focused toward establishing the DOM as a platform- and language-independent or neutral interface that enables programs and scripts to dynamically access and update content, style, and even structure displayed within a browser. Further enhancements may include a communications standard to allow inline scripts to communicate interactively with background CGIs or to incorporate processed data back into a presented page.

Vendors use the term “dynamic HTML” to describe the combination of the DOM (or DOM-like occurrences), HTML markup, style sheets, and inline scripts (such as JavaScript) to animate and dynamically alter Web documents. Microsoft, Netscape, and others have submitted proposals for the ultimate outcome of the DOM standard. Obviously, each vendor's proposal favors its current technology and products, but the W3C is determined to define a standard that offers no native advantage to any vendor.

The work at the W3C has only recently started, so it may be months before any public proposals or drafts of any kind are available. Until the W3C is able to make a declaration of DOM standards, the user community is at the mercy of those vendors who forge ahead and deploy their own versions of these proposed standards. In such situations the double-talk of vendors is highly evident — they claim to work with the W3C by establishing standards for the Internet community, but they persist with and deploy their own proprietary implementations. This tactic can affect the ultimate outcome of a standard by rallying public support and approval for one vendor's solution over another. This questionable policy keeps the user community in a perpetual state of flux. Unfortunately, it's also become one of the rules of the Web world!



Keep an eye on the DOM information page hosted by the W3C for the latest standards developments:

<http://www.w3.org/pub/WWW/MarkUp/DOM/>

At the time of publication, the only available document from this page was a preliminary version of standards requirements.

The skinny on DH

The surprising thing about Dynamic HTML is that so much is being made of an interface standard that adds little to the existing inline scripting capabilities. Adept programmers can create some of the features touted by Dynamic HTML without invoking a new standard. To be specific, Dynamic HTML is a

definition for an interface standard that adds an object model to HTML. This model enables scripts and other programs to interact dynamically with and alter page elements while a browser is displaying them. Dynamic HTML supports most of the common and popular inline scripting languages, including JavaScript, Java, and VBScript. The only really new or improved features that Dynamic HTML offers are the capability to interact with multimedia and database objects, support for the W3C Cascading Style Sheets (CSS) Specification, and backward browser compatibility.

Unless the W3C adds new HTML markup, or severely alters the assumed path to standardization, most Web authors and beginner programmers will be able to take advantage of Dynamic HTML without too much difficulty. Most existing expertise and experience with Web programming applies well to the construction of Dynamic HTML.

Benefits of Dynamic HTML

Although the final standard for Dynamic HTML is somewhere off in the foreseeable future, we can still generalize about its significance and capabilities. The benefits of this new method of Web publishing and design may pan out only as vague hopes and dreams, but today the following highlights gleam within current proprietary extensions. From our current vantage point, the promises of Dynamic HTML include:

- ✓ **Customized creative content:** Web designers and authors alike can alter the appearance and content of documents on-the-fly in response to user activity. Some of the dynamic features include tool tips over images and icons, fly-open contents, and alterations in font, color, or style of text when a user moves his mouse pointer over it. All of these page interactions occur without communication with the server, and operate as long as a page is displayed in a browser.
- ✓ **Rich multimedia:** Special effects, animations, audio mixing, transitions, vector graphics, and moving sprites are new additions that require far less data transfer than previous Web multimedia methods.
- ✓ **Fine layout control:** Just as a programmer can define the position of objects based on absolute screen position, so Web authors can use x, y, and z order absolute positioning to control object placement with Dynamic HTML.
- ✓ **Business data manipulation:** Web documents can manipulate content on-the-fly without communicating with a server or using high-end programming languages. One possible example would be sorting stock prices based on price or company name.
- ✓ **Decreased server load:** With more and more of the processing of Web documents occurring on the client side, servers will be able to handle more simultaneous users without increasing the servers' workloads.

- ✓ **Improved platform independence and cross-platform compatibility:** Because Dynamic HTML uses inline scripting directly supported by browsers, fewer platform-specific issues (such as plugins or helper applications) can prevent access to delivered content.
- ✓ **Down-level browser support:** Dynamic HTML features will automatically change themselves to static elements that conform to older browsers' capabilities.

The big boys

After submitting proposals to the W3C for consideration and standardization, both Microsoft and Netscape have forged ahead and implemented proprietary solutions into their latest browser versions. You can explore these respective companies' perspectives on how Dynamic HTML should function by visiting the following URLs:

- ✓ Microsoft Internet Explorer 4.0 with Dynamic HTML:

<http://www.microsoft.com/ie/ie40/browser/dynamic.htm>

- ✓ Netscape Communicator 4.0 with Dynamic HTML:

http://developer.netscape.com/library/documentation/DHTMLguid/dynamic_resources.html

To experience the thrill of Dynamic HTML, you'll have to download the 15-plus megabytes' worth of files for each of the 4.0 browsers. As we write this material, both Netscape Communicator and Internet Explorer are still in beta release, so you are on your own if you install a beta application. We recommend waiting for a final release, especially if your computer is a production tool.

To Script, or Not to Script?

The title of this section raises a question that many of you will have to ask — and answer — where your own Web pages are concerned. Because of the power and flexibility of the many scripting languages available, you should take this question quite seriously. Although some effort is involved in learning a scripting language (or development environment, if you'd prefer), the results can be nothing short of amazing. Because scripting is such a wonderful alternative to CGI programming (and requires no interaction with the ISP that hosts your Web pages), we think it's an important addition to any Web site. What you decide to do, however, is entirely up to you.

Enough of scripting already, the next chapter moves on to cover enriched forms of text display, such as Adobe's Portable Document Format (PDF).

Glossary

.....

absolute: When referring to the modification of path names or URLs, absolute means a full and complete specification (as opposed to a relative one).

Acrobat: A viewing program (helper application) from Adobe Systems. Acrobat is used for the portable document format (PDF) text display system. *See* Amber.

ActiveX: Microsoft's third-generation component architecture, which is based on the Component Object Model (COM). ActiveX objects may reside in stand-alone machines, on a LAN or intranet, or on the Internet. Specialized for the Internet, ActiveX objects are automatically downloaded if the object is not in the user's machine.

Afterburner: File compression software, used in conjunction with Shockwave from Macromedia, to compress native Director film strips for delivery over the Internet. Compression is usually by 60 percent or more.

agent: Any of a class of software programs capable of interacting with network or Internet resources on another user's behalf.

alternative selectors: HTML tag attributes that serve a similar function. For example, the functions of the CLASS and STYLE attributes can overlap.

Amber: The Netscape plugin version of Acrobat for viewing PDF files within Netscape.

anchor: A tagged text or graphic element in HTML that acts as a link to another location inside or outside a Web document. An anchor may also be a location in a document that acts as the destination for an incoming link. The latter definition usually applies in this book.

animation: A computerized process of creating moving images by rapidly advancing from one still image to the next.

anonymous ftp: A type of Internet file access that relies on the File Transfer Protocol (FTP) service, by which any user can typically access a file collection by logging in as *anonymous* and supplying his or her e-mail address as a password.

AppleScript: The scripting language for the Macintosh operating system, AppleScript is used to build CGI programs for Macintosh-based Web servers.

applet: A small application that performs a specific task, such as the Calculator in Microsoft Windows. In Internet speak, this term usually refers to a small Java application.

ASCII (American Standard Code for Information Interchange): A coding method that represents standard alphabetic, numeric, and other keyboard characters in computer-readable, binary format.

attribute: A named characteristic of an associated HTML tag. Attributes may be required or optional. Depending on the tag and the attribute, some attributes take values (if so, the syntax is `ATTRIBUTE="value"`).

authoring software: Programs that understand HTML tags and their placement. Some authoring programs can even enforce HTML syntax; others can convert from word-processing or document-formatting programs to HTML formats.

authoring tool: See authoring software.

back end: The server-side of client/server structure. This part of the processing is usually handled by programs running in obscurity on the server, out-of-sight (and mind) of most users.

bandwidth: Technically, the range of electrical frequencies that a device can handle. More commonly, bandwidth is a measure of a communications technology's carrying capacity.

Basic (Beginner's All-purpose Symbolic Instruction Code, also called *BASIC*): A programming language that is easy to learn and use. The most popular implementation of the language is QuickBasic from Microsoft Corporation.

beta test: The phase of software testing during which a program or system is turned over to a select group of users outside the development organization for use in more or less real-life situations.

BIOS (Basic Input-Output System): A basic driver software for a PC. The BIOS permits component devices to communicate with one another.

bitmapped: A graphic image that's represented as a matrix of binary on-off values, usually by conversion from some other graphics format.

body: One of the main identifiable structures of any HTML document. The body is usually trapped between the `<HEAD>` information and the footer information.

bookmark: The Netscape Navigator browser facility for building a list of URLs that users wish to keep for future reference.

Boolean: Computer shorthand for logical operations, such as those defined to combine or compare values (AND, OR, NOR, NOT, NAND, and so on).

'bot: See robot.

broken graphic icon: A standard icon that appears within a browser when an `` or other graphics tag is encountered, but the associated image file cannot be located or displayed.

browser: A Web access program that can request HTML documents from Web servers and render these documents on a user's display device. See client.

BSD (Berkeley Software Distribution): A flavor of UNIX that was particularly important in the late '70s and '80s when most of the enhancements and add-ons to UNIX appeared first in the BSD version (such as TCP/IP).

bugs: Small verminous creatures that sometimes show up in software in the form of major or minor errors, mistakes, and gotchas. Bugs got their name from insects that, having been attracted to the glow of the filament in a tube, were found in antiquated tube-based computers of the late '50s and early '60s.

burn: To convert a file from its original uncompressed Macromedia format to a compressed Shockwave format.

bytecode: The intermediate, semicompiled form of Java code created by the Java compiler *javac*.

C: A programming language developed at AT&T Bell Laboratories, C remains the implementation language for UNIX and the UNIX programmer's language of choice.

Cascading Style Sheets (CSS): The latest World Wide Web Consortium-sponsored specification for HTML document style sheets, which describe formatting and layout rules and conventions.

case-sensitive: Indicates that the case of the letters used in computer input is significant. For example, HTML tags can be input in any mixture of upper- and lowercase (HTML tags are not case-sensitive), but because HTML character entities (such as É) are case-sensitive, these entities must be reproduced exactly.

cast: The collection of objects in Macromedia Director. These objects may be graphics, sounds, animations, or other data that appears on the main display area.

CD-ROM (Compact Disc-Read-Only Memory): A computer-readable version of the audio CD; CD-ROMs can contain up to 650M of data, making them the distribution media of choice for many of today's large (some would even say *bloated*) programs and systems.

CERN (Conseil Européen pour la Recherche Nucleaire): The Center for High-Energy Physics in Geneva, Switzerland, and the birthplace of the World Wide Web.

change log: A record of changes made to elements within a Web site, including files, text entries, graphical objects, and so on. The change log is used to keep track of changes and enhancements during the maintenance process.

character entity: A way of reproducing strange and wonderful characters within HTML. Character entities take the form &string; where the ampersand (&) and semicolon (;) are mandatory metacharacters, and string names the character to be reproduced in the browser. Because character entities are case-sensitive, the string between the ampersand and the semicolon must be reproduced exactly.

character mode (also called text mode): Indicates that a Web browser can reproduce text data only. Character-mode browsers cannot produce graphics directly without the assistance of a helper application.

chatterbots: Software agents that provide help, give advice, and explain local conditions in chat rooms and other interactive environments on the Web.

clickable map: A graphic in an HTML file that has had a pixel coordinate map file created for it. This map file enables regions of the graphic to point to specific URLs for graphically-oriented Web navigation.

client: The end-user side of the client/server arrangement. Client typically refers to a consumer of network services. A Web browser is, therefore, a client program that talks to Web servers.

client/server: A model for computing that divides computing into two separate roles that are usually connected by a network: The client works on the end-user side of the connection and manages user interaction and display (input and output, and related processing), while the server works elsewhere on the network and manages data-intensive or shared processing activities (such as serving up the collections of documents and programs that a Web server typically manages).

comment: In programming, a descriptive statement in a source language program that is used for documentation.

Common Gateway Interface (CGI): The specification governing how Web browsers can communicate with and request services from Web servers. CGI is also the format and syntax for passing information from browsers to servers via forms or document-based queries in HTML.

Common Ground: A portable document format used to create input to one kind of text display engine available via the Web.

compiler: A special computer program that takes human-readable source code from some programming or scripting language and turns the code into a computer-readable binary equivalent.

computing platform: A way of referring to the kind of computer someone is using. This term encompasses both hardware (the type of machine, processor, and so on) and software (the operating system and applications) in use.

connections per minute: The measurement for the number of user requests that a Web server can simultaneously handle over the course of 60 seconds.

content: The reason users access Web documents and keep coming back for more. This is the information you provide to the public and is your major consideration in Web site development (although form is also important).

content-type: A MIME convention for identifying the type of data being transported over a network for delivery to any of a set of programs, including e-mail readers and Web browsers.

convention: An agreed-upon set of rules and approaches that enables systems to communicate with one another and work together.

conversion program: In general, a program that converts one type of format to another. In the context of the Web, the term refers to a program that reads a native word processor or page layout file format and converts it to a corresponding HTML layout.

CPU (Central Processing Unit): The master circuitry or chip that provides the brains of a typical computer.

cron: A UNIX utility that permits programs to be scheduled to run at specified times or at regular intervals. The *cron* utility is useful for handling automated maintenance tasks.

CSU/DSU (Channel Service Unit/Data Service Unit): A special translating piece of hardware, found between a network and a digital telephone line, that translates data between the two formats. CSU/DSU is most commonly used to attach a network router to a T1 or other digital telephone line.

daemon: A special type of program (usually in the UNIX world) that runs constantly in the background, ready to handle certain types of network service requests. E-mail, HTTP, and other Internet services depend on daemons to handle user service requests.

DBMS (Database Management System): Software that controls the organization, storage, retrieval, and security of information stored in a database.

DCR: File extension for a Director film strip that has been converted from its original format by using Afterburner. The file is destined for Web delivery.

default: In general computer-speak, a selection that occurs automatically in a program or instruction when the user hasn't made an explicit choice. For HTML, the default is the value assigned to an attribute when none is supplied.

desktop (desktop machine): The computer that a user typically has on his or her desktop. Desktop is a synonym for end-user computer or computer.

dial-up: A connection to the Internet (or some other remote computer or network) that is made by dialing up an access telephone number.

DIR: File extension for an original, editable Macromedia Director file.

Director (Macromedia Director): A large, complex software program used to create animated multimedia presentations on either a Macintosh or a PC. Director provides multimedia content for delivery via Shockwave technology.

directory path: The device and directory names needed to locate a particular file in any given file system. For HTML, UNIX-style directory paths usually apply.

directory structure: The underlying file container structure that describes the hierarchical organization of most computer file systems.

DNS (Domain Name Server): *See* domain names.

document: The basic unit of HTML information. The term refers to the entire contents of any single HTML file. Because this concept doesn't always correspond to normal notions of a document, we refer to what could formally be called *HTML documents* more or less interchangeably with *Web pages*, which is how these documents are rendered by browsers for display.

document headings: The class of HTML tags that we generically refer to as `<H*>`. Document headings enable Web-page authors to insert headings of various sizes and weights (levels 1 through 6) to add structure to their documents' contents. As structural elements, headings identify the beginning of a new concept or idea within a document.

document root: The base directory where HTML files and other components for any particular Web site are located on a Web server, or where the directories containing these components are ultimately attached.

document structure: The methods used to organize and navigate within HTML documents or related collections of documents.

document transfer rate: The speed at which a Web server can deliver pages to users in response to their document requests (usually measured in documents per minute).

document-based queries: One of two methods of passing information from a browser to a Web server. Document-based queries are designed to pass short strings of information to the server by using the `METHOD="GET"` HTTP method of delivery. This method is typically used for search requests or other short lookup operations.

domain names: The names used on the Internet as part of a distributed database system for translating computer names into physical addresses and vice versa.

DOS (Disk Operating System): The underlying control program used to make most Intel-based PCs run. MS-DOS, from Microsoft Corporation, is the most widely used implementation of DOS and provides the scaffolding atop which the equally widely used Microsoft Windows software runs. *See* OS.

download: To transfer a file from a server to a user across a network or the Internet.

DSSSL (Document Style Semantics and Specification Language): An early contender for HTML standardization. This formal language is used to define SGML style sheets. DSSSL users unanimously agree that this language is too complex and difficult for Web use.

DSSSL-Lite: A svelte subset of DSSSL. This style sheet language was considered for adoption as an HTML standard, but it, too, was judged too complex and difficult for ordinary mortals.

DTD (Document Type Definition): A formal SGML specification for a document. A DTD lays out the structural elements and markup definitions that can then be used to create instances of documents.

dumb terminal: A display device with attached keyboard that relies on the intelligence of another computer to drive its display and interpret its keyboard inputs. These devices were the norm in the heyday of the main-frame and minicomputer and are still widely used for reservation systems, point of sale, and other special-use applications.

Dynamic HTML: A set of special extensions to standard HTML that are interpreted and substituted as a Web page is delivered to a user. Versions have been proposed by both Netscape and Microsoft, and the World Wide Web Consortium is in the process of creating a proposal for a standard version.

electronic commerce: The exchange of money for goods or services via an electronic medium. Many companies expect electronic commerce to take the place of the majority of mail order and telephone order shopping by the end of the century.

e-mail (abbreviation for electronic mail): E-mail is the preferred method for exchanging information between users on the Internet (and other networked systems).

encoded information: A way of wrapping computer data in a special envelope to ship it across a network, encoded information refers to data-manipulation techniques that change data formats and layouts to make them less sensitive to the rigors of electronic transit. Generally, recipients (of encoded information) must decode the encoded information before using it.

Envoy: A portable document format and reader created by Novell for delivery and viewing of formatted documents over the Internet.

EPS (Encapsulated PostScript): A special form of PostScript that includes all external references within the file to make it self-containing and ready to render in any environment.

error message: Information delivered by a program to a user, usually to inform the user that things haven't worked properly, if at all. Error messages are an ill-appreciated art form and contain some of the funniest and most opaque language we've ever seen (also, the most tragic for their unfortunate recipients).

error-checking: A collection of software utilities that systematically checks input or data for errors. An error-checking utility can respond to any condition for which it has a programmed response.

Ethernet: The most common local-area networking technology in use today, Ethernet was developed at about the same time (and by many of the same people and institutions) as the Internet.

event-handling mechanism: A built-in software facility that permits a program to respond to conditions or circumstances as they occur.

external reference: A resource that is stored somewhere other than the location of a Web document or program.

FAQ (Frequently Asked Questions): A list of frequently asked questions that Usenet newsgroups, mailing list groups, and other affiliations of like-minded individuals on the Internet usually designate a more senior member of their band to assemble and publish, in an often futile effort to keep from answering them quite as frequently.

file dependencies: Contents of one file that depend on the contents of some other file (so that the containing file depends on the contained file).

file extension: In DOS, the 3-letter part of a file name after the period. In UNIX, Macintosh, and other file systems, file extension refers to the string after the right-most period in a file name. File extensions are used to label files as to type, origin, and possible use.

flame: A particularly hostile or nasty e-mail message ("That was a real flame."). As a verb, the term means to be a recipient of this type of message ("He got flamed.").

flamewar: The result when two or more individuals start exchanging hostile or nasty e-mail messages. This exchange is viewed by some as an art form and is best observed on Usenet or other newsgroups (where the `alt.flame` newsgroup is a good place to browse for examples).

font manipulation: In the display of a Web document, increasing or decreasing the default font size, typeface, and style based on HTML extensions or style sheet settings.

footer: The concluding part of an HTML document. The footer should contain contact, version, date, and attribution information to help identify a document and its authors.

forms: A mechanism that enables users to interact with servers on the Web. In HTML, forms are built on special markup that lets browsers solicit data from users and then deliver that data to specially designated input-handling programs on a Web server.

frames: An HTML construct that allows a number of documents to be displayed simultaneously within a browser, based on a master document that defines where the individual part displays onscreen. Although HTML can support a theoretically unlimited number of frame areas, most good designs use four or less.

front end: The client side of the client/server model, where the user views and interacts with information from a server. For the Web, browsers provide the front end that communicates with Web servers on the back end.

FTP (sometimes ftp; File Transfer Protocol): An Internet file transfer service based on the TCP/IP protocols. FTP provides a way to copy files to and from FTP servers elsewhere on a network.

full-text indexing: The ability to search on any string that occurs within a text file. Full-text indexing permits a collection of HTML or other documents to be searched by keyword, string, or substring.

fuzzy logic: A way of matching terms or keywords (for search activities) that finds terms or patterns that are somewhat like the terms supplied to drive the search, as well as exact matches.

garbage collection: The ability to reclaim and reuse computer memory that has been allocated within a program, after the variables or data structures that used that memory are no longer needed.

gateway: A type of computer program that knows how to connect two or more different kinds of networks, how to translate information from one network's format to the other's, and vice versa. Common types of gateways include e-mail, database, and communications.

GIF (abbreviation for Graphics Information File): One of a set of commonly used graphics formats within Web documents. GIF is commonly used because of its compressed format and compact nature.

global renaming: The capacity to change a name in one place on a Web site and have a software tool make equivalent changes to any other occurrences (of that same name) throughout the entire site. Global renaming is very useful when moving a site or changing existing content and references.

GNU free software license: A software license promulgated by the Free Software Foundation that permits royalty-free distribution of software, as long as all source code is included with the distribution package, and all changes to the original are clearly marked and attributed to their authors. This is also known as a General Public License (GPL).

gopher: A program/protocol developed at the University of Minnesota. Gopher provides for unified, menu-driven presentation of a variety of Internet services, including WAIS, Telnet, and FTP.

graphics: In HTML documents, files that belong to one of a restricted family of types (usually GIF or JPEG) that are referenced via URLs for inline display on Web pages.

grep (abbreviation for general regular expression parser): A standard UNIX program that looks for patterns found in files and reports on their occurrences. The grep program handles a wide range of patterns, including so-called regular expressions that can use substitutions and wild cards to provide powerful search-and-replace operations within files.

group: A named collection of nodes in VRML that acts as a single object within a virtual world.

GUI (Graphical User Interface, pronounced *gooeey*): A visually oriented interface that makes users' interaction with computerized information easier. GUIs make graphical Web browsers possible.

Gzip: A popular file compression technology, used largely among the UNIX community. Gzip is freely available through any of the countless GNU licensees.

HDTV (High Definition TeleVision): An emerging, fully digital, high-resolution form of video data that should become the next prevailing broadcast TV standard.

header files: Files containing information that identifies incoming data by MIME type (and subtype, where applicable).

heading: A markup tag in HTML that adds document structure. The term is sometimes used to refer to the initial portion of an HTML document between the `<HEAD> . . . </HEAD>` tags, where titles and context definitions are commonly supplied.

helper applications: Applications that help the Web browser deliver information to users. Today, browsers can display multiple graphics files (and other kinds of data); sometimes, browsers must pass particular files — for example, motion picture or sound files — over to other applications that know how to render the data they contain.

hierarchical filing system: A typical computer file system, where directories are organized in a hierarchical organization (one directory acts as the root, with a tree of other directories beneath it).

hierarchical structure: A way of organizing Web pages by using links that make some pages subordinate to others. *See* tree structure.

history list: A list of all the URLs a user visits during a session on the Web. This list provides the user with a handy way to jump back to any page that he or she has already visited while online. History lists normally disappear when the user exits the browser program.

host: An Internet-connected computer that supports one or more of the standard services, including Web access, e-mail, file transfer, and more.

hot key: A special keyboard combination that has the same effect as one or more mouse clicks or menu selections in a GUI interface (for example, CTRL+O means “open file” in many applications).

HotJava: The Java-enabled Web browser from Sun Microsystems, which is written in the Java programming language.

hotlist: A Web page that consists of a series of links to other pages, usually annotated with information about what’s available on that link. Hotlists act like switchboards to content information and are usually organized around a particular topic or area of interest.

HTML (HyperText Markup Language): The SGML-derived markup language used to create Web pages. Not quite a programming language, HTML nevertheless provides a rich lexicon and syntax for designing and creating useful hypertext documents for the Web.

HTTP or http (Hypertext Teleprocessing Protocol, or Hypertext Transfer Protocol): The Internet protocol used to manage communication between Web clients (browsers) and servers.

httpd (http daemon): The name of the collection of programs that runs on a Web server to provide Web services. In UNIX-speak, a daemon is a program that runs all the time and listens for service requests of a particular type; thus, an httpd is a program that runs continually on a Web server, ready to field and handle Web service requests.

hyperlink: A shorthand term for hypertext link. *See* hypertext link.

hypermedia: Any of a variety of computer media — including text, graphics, video, sound, and so on — available through hypertext links on the Web.

hypertext: A method of organizing text, graphics, and other data for computer use that lets individual data elements point to one another. Hypertext is a nonlinear method of organizing information (especially text).

hypertext link: A user-selectable document element (defined by special HTML markup) that, when selected, can change the user's focus from one document (or part of a document) to another.

IETF (Internet Engineering Task Force): The official standards organization (a suborganization of the Internet Architecture Board, or IAB) that governs all TCP/IP and other Internet-related standards.

image map (a synonym for clickable image): An overlaid collection of pixel coordinates for a graphic. An image map can be used to locate the region of a Web page graphic that a user selects by clicking the mouse. The map location, in turn, is used to select a related hypertext link for further Web navigation.

inheritance: Defines a relationship among objects in a hierarchy in object-oriented programming. Objects that are subordinate to other objects acquire attributes and characteristics because of that relationship. By inheritance, subordinate objects automatically include information defined for their parents in the hierarchy.

INIT: On the Macintosh, a program that runs during the computer's initialization phase to augment or modify the basic operating system's behavior or characteristics.

ink effects: Text-based colorization or animation effects in Macromedia Director.

input-handling program: For Web services, a program that runs on a Web server and is designated by the ACTION attribute of an HTML <FORM> tag. The job of the input-handling programs is to

field, interpret, and respond to user input from a browser. The program typically custom-builds an HTML document in response to a user request.

interleaved: A graphical image on the Web that displays onscreen in increasing levels of detail instead of from the top down. The levels of detail appear as individual lines of pixels separated by a constant interval during the display. This process gives interleaved images the appearance of being drawn in slices.

intermediate code: A way of representing a computer program in a form that's somewhere between human-readable source code and machine-readable binary executable code. Java bytecode is a form of intermediate code, ready to be converted into a binary executable form at the user's workstation through the agency of the Java runtime system.

internal link: A hyperlink on a Web site that links to a resource on the same site (or in some cases, within the same document).

Internaut: Someone who travels using the Internet (like *astronaut* or *argonaut*).

Internet: A worldwide collection of networks that began with technology and equipment funded by the U.S. Department of Defense in the 1970s that today links users in nearly every known country, who speak nearly every known language.

Internet Relay Chat (IRC): An Internet application and protocol that permits multiple users to communicate with one another by typing comments, all of which appear in the IRC application display, labeled by each user's supplied "handle" (also known as the user name).

Internet Service Provider (ISP): An organization that provides individuals or other organizations with access to the Internet, typically by purchasing large amounts of communications bandwidth and reselling small chunks of that bandwidth to its customers.

Internet Studio: A planned Microsoft software product (formerly code-named *Blackbird*) that will include Web document authoring, management, and maintenance utilities.

InterNIC (Internet Network Information Center): The Internet agency that handles IP address allocation and domain name registration facilities.

intranet: A TCP/IP-based network that uses standard Internet services and applications within the confines of a particular organization, to create a sort of “in-house Internet.”

IP (Internet Protocol): The specific networking protocol that ties computers together over the Internet. IP is also used as a synonym for the whole TCP/IP protocol suite. *See* TCP/IP.

IP address: A unique numeric address for a particular machine or physical interface on the Internet (or any other TCP/IP-based network). An IP address consists of four decimal octets separated by periods (such as, 108.28.36.51).

IPnG (IP Next Generation, or IPv6): An emerging replacement standard for TCP/IP that will broaden the address space for IP from its current ceiling of available addresses.

ISAPI (Internet Service Application Programming Interface): An Internet API developed by Microsoft that enables remote server applications.

ISDN (Integrated Services Digital Network): An emerging digital technology for telecommunications that offers

higher bandwidth and better signal quality than old-fashioned analog telephone lines. Not yet available in many parts of the United States or in the rest of the world.

ISO (International Standards Organization): The granddaddy of standards organizations worldwide. The ISO is comprised of standards bodies from many countries. The ISO sets most important communications and computing standards, such as the telecommunications and character code standards in this book.

Java: A specialized object-oriented programming language designed for creation of platform-independent, network deliverable, client-side applications. Some Java applications may be invoked within Web documents to add dynamic, ongoing behavior.

JavaScript: A scripting language that draws on underlying Java classes and objects to provide a simplified method to include dynamic behavior within Web documents.

JPEG or JPG (Joint Photographic Experts' Group): An industry association that defined a particularly compressible format for image storage that is designed for dealing with complex color still images (such as photographs). Files stored in this format usually take the extension JPEG (except on DOS or Windows machines, which use the three-character JPG equivalent). Today, JPEG is emerging as the graphics format standard of choice for use on the World Wide Web.

Kbps (Kilobits per second): A measure of communications speeds, in units of 2 to the 10th power bits per second ($2^{10} = 1024$, which is just about 1,000 and explains the quasi-metric K notation).

kerning: The relationships between characters as they appear on a document, including spacing and relative positioning.

KISS (Keep It Simple, Stupid!): A self-descriptive philosophy that's supposed to remind us to eschew obfuscation, except it's easier to understand!

LAN (Local Area Network): Typically, one of a variety of communications technologies that links computers together in a single building, business, or campus environment.

LaTeX: A specialized version of Donald Knuth's TeX typesetting program, LaTeX includes templates and definitions for creating book-length manuscripts.

layout element: In an HTML document, a layout element is a paragraph, list, graphic, horizontal rule, heading, or some other document component whose placement on a page contributes to its overall look and feel.

leading: The amount of space between lines of text (pronounced properly, rhymes with "bedding").

linear text: Shorthand for old-fashioned documents that work like this book does: by placing one page after the other, ad infinitum in a straight line. Even though books have indexes, pointers, cross-references, and other attempts to add linkages, users must apply these linkages manually (instead of clicking a mouse).

Lingo: A scripting language used within Macromedia Director to create and automate animation and other repetitive or time-based behaviors in a multimedia presentation.

link: A pointer in one part of an HTML document that can transport users to another part of the same document or to another document entirely. This capability puts the *hyper* in hypertext. In other words, a link is a one-to-one relationship or association between two concepts or ideas, similar to cognition. (The brain has triggers, such as smell, sight, and sound, that cause a link to be followed to a similar concept or reaction.)

link map: A directed graph in a Web site that shows the links between and among constituent documents.

linked media files: Multimedia content files that include references to other files within themselves. Shockwave for Director does not currently support these files for Internet-based delivery.

list element: An item in an HTML list structure tagged with the `` (list item) tag.

list tags: HTML tags for a variety of list styles, including ordered lists ``, unordered lists ``, menus `<MENU>`, glossary lists `<DL>`, or directory lists `<DIR>`.

listserv: An Internet e-mail handling program, typically UNIX-based, that provides mechanisms to let users manage, contribute and subscribe to, and exit from named mailing lists that distribute messages to all subscribed members daily. A common mechanism for delivering information to interested parties on the Internet, this program enables the HTML working group members to communicate with each other.

LOD (Level of Detail): The name of a specific root node in the VRML environment often acts as the basis for the construction of entire virtual worlds.

log files: Web server files that accumulate data about errors, data access, and user activity that may later be analyzed to determine site behavior, solve problems, and improve document relationships and designs.

logical markup: A number of HTML character-handling tags that exist to provide emphasis or to indicate the involvement of a particular kind of device or action.

Lynx: A widely used UNIX-based character-mode Web browser.

maintenance: The process of regularly inspecting, testing, and updating the contents of Web pages; also, an attitude that these activities are both inevitable and advisable.

majordomo: A set of Perl programs that automate the operation of multiple mailing lists, including moderated and unmoderated mailing lists and routine handling of subscribe/unsubscribe operations.

map file: A set of pixel coordinates on a graphic image that correspond to the boundaries of regions that users may select when using the graphic for Web navigation. This map file must be created by using a graphics program to determine regions and their boundaries, and then stored on the Web server that provides the coordinate translation and URL selection services.

markup: A method of embedding special characters (metacharacters) within a text file to instruct a computer program how to handle the contents of the file itself.

markup language: A formal set of special characters and related capabilities that define a specific method for handling the display of files that include

markup; HTML is a markup language that is an application of SGML and is used to design and create Web pages.

Mbps (Megabits per second): A measure of communications speeds, in units of 2^{20} bits per second ($2^{20} = 1,048,576$, which is just about 1,000,000 and explains the quasi-metric M notation).

metacharacter: A specific character within a text file that signals the need for special handling. In HTML, the angle brackets (< >), ampersand (&), pound sign (#), and semicolon (;) can all function as metacharacters.

method: A legal operation or transformation associated with a particular object in object-oriented programming. For HTML, a method refers to one of several types of data delivery associated with communicating <FORM> input from a user to a Web server.

MIDI (Musical Instrument Digital Interface): A computer-industry standard for interconnecting musical instruments, synthesizers, and computers. MIDI provides methods to translate music into computer data, and vice versa. Any file with a MIDI contains music or other multivoice signals.

MIME (Multipurpose Internet Mail Extensions): HTTP communications of Web information over the Internet rely on a special variant of MIME formats to convey Web documents and related files between servers and users, and vice versa.

mismatched tags: Opening a marked segment of text with one HTML tag, and attempting to close it with an invalid or incorrect closing tag (for example, <H1>This is wrong!</H2>).

modem (acronym for **modulator/demodulator**): A piece of hardware that converts between the analog forms for voice and data used in the telephone system and the digital forms for data used in computers. In other words, a modem lets your computer communicate by using the telephone system.

MOO (MUD, Object-Oriented): A particular implementation of a MUD, MOOs are built to be flexible and extensible, rather than bounded and static, like ordinary MUDs. *See* MUD.

Mosaic: A powerful graphical Web browser originally developed at the NCSA, now widely licensed and used for a variety of commercial browser implementations.

MPEG or **MPG** (Motion Picture Experts' Group): A highly compressed format designed for use in moving pictures or other multiframe-per-second media (such as video). MPEG cannot only provide tremendous compression (up to 200 to 1), but it also updates only elements that have changed onscreen from one frame to the next. This feature makes the MPEG format extraordinarily efficient as well. MPEG is the common file extension to denote files using this format and MPG is the three-letter equivalent on DOS and Windows systems (which can't handle four-letter file extensions).

MPPP (Multilink Point-to-Point Protocol): An Internet protocol that allows simultaneous use of multiple physical connections between one computer and another to aggregate their combined bandwidth and create a larger virtual link between the two machines.

MUCK (Multi-User Consensual Knowledgebase): A user-extendable, multiuser adventure game.

MUD (Multi-User Dungeon): A text-based virtual world where multiple users can collaborate and compete within an orchestrated fantasy environment.

multimedia: A method of combining text, sound, graphics, and full-motion or animated video within a single compound computer document.

multiple inheritance: Some object-oriented programming languages, such as C++, support multiple inheritance. In this kind of environment, an object can have multiple parents in the object hierarchy and can inherit characteristics and attributes from all parents.

multithreaded: An operating system concept in which the system or the applications that it supports can divide into logical subtasks, each of which can run independently, but all of which combine to provide a particular application or service.

MVS (Multiple Virtual Storage): A file system used on IBM mainframes and clones.

Native multithreading: A programming language with a built-in thread-handling mechanism (such as Java). Native multithreading enables programs to create and manage multiple execution threads explicitly.

navigate: The process of finding your way around a particular Web site or the Web in general.

navigation: The use of hyperlinks to move within or between HTML documents and other Web-accessible resources.

navigation bar: A way of arranging a series of hypertext links on a single line of a Web page to provide a set of navigation controls for an HTML document or a set of HTML documents.

navigation buttons: Using graphics to provide navigation links; otherwise, a form of navigation bar. *See* navigation bar.

NCSA (National Center for Supercomputing Applications): A research unit of the University of Illinois at Urbana, where the original Mosaic implementation was built and where the NCSA `httpd` Web server code is maintained and distributed.

nesting: In computer terms, one structure that occurs within another structure is said to be nested. In HTML, nesting happens most commonly with list structures, which may be freely nested within one another, regardless of type.

netiquette: The written and unwritten rules of behavior on the Internet (a networking takeoff on the term *etiquette*). When in doubt about whether an activity is permitted, ask first, and then act only if no one objects. (Check the FAQ for a given area, too. The FAQ often explicitly states the local rules of netiquette for a newsgroup, mailing list, or other group.)

network link: The tie that binds a computer to a network. For dial-in Internet users, the network link is usually a telephone link. For directly attached users, this link is whatever kind of technology (Ethernet, token-ring, FDDI, or so on) is in local use.

newsgroup: A named collection of messages collected and distributed via the Network News Transport Protocol (NNTP), through the public Usenet on the Internet, or through any number of private NNTP services (such as the Microsoft `msnews.micorosoft.com` NNTP server).

node: A basic object within the VRML environment, a node can appear to be an independent graphic within a scene or may be a constituent part of a group node.

numeric entity: A special markup element that reproduces a particular character from the ISO-Latin-1 character set. A numeric entity takes the form `&#nnn`, where `nnn` is the one-, two-, or three-digit numeric code that corresponds to a particular character.

object: The basic element within an object-oriented environment. Any single object represents a particular instance of a class of possible objects, where all share a common underlying definition and an associated set of methods.

object type: A named type of data structure in programming languages. An object's type is the specific data type associated with an object.

object-oriented: A programming methodology that concentrates on the definition of constituent parts of a program and the operations that can be performed on the parts (called methods).

obsolete elements: HTML markup codes that are no longer supported within the current HTML specifications.

OCR (Optical Character Recognition): A class of software that can “read” and interpret image data to convert pictures of text into a best-guess translation of actual textual data.

on-demand connection: A dial-up link to a service provider that is available whenever it's needed.

online: A term that indicates that information, activity, or communications are located on, or taking place in, an electronic, networked computing environment (such as the Internet). The opposite of online is offline, which is what your computer is as soon as you disconnect from the Internet.

operator overloading: The ability to use one type of operator (in programming languages, such as C or C++) on data that may not belong to that type (for example, to use the + operator to add two strings together).

orphan file: A file on a Web site that is not referenced by any other file (so you have no way to get to the orphan file, except through its absolute URL).

OS (Operating System): The underlying control program on a computer that makes the hardware run and supports the execution of one or more applications. DOS, UNIX, and OS/2 are all examples of operating systems.

packet: A basic unit (or package) of data that describes individual elements of online communications. In other words, data moves across networks like the Internet in packets.

pages: The generic term for the HTML documents that Web users view on their browsers.

paragraphs: The basic elements of text within an HTML document. <P> is the markup tag used to indicate a paragraph break in text (the closing </P> tag is currently optional in HTML).

parser: A program that reads text input for the purpose of recognizing and interpreting particular strings. A parser is the first part of a compiler, and it reads the source code to recognize language terms and operators, to build a formal representation of the program's contents and structure.

path, path name: See directory path.

PC (personal computer): A generic term that refers to just about any kind of desktop computer. Its original definition was as a product name for the IBM 8086-based personal computer, the IBM PC.

PDF (portable document format): The name of the document format for Acrobat from Adobe Systems, Inc. PDF is also used as the file extension for files in this particular format.

Perl: A powerful, compact programming language that draws from the capabilities of languages such as C, Pascal, and BASIC. Perl is emerging as the language of choice for CGI programs. Its emergence is partly owing to its portability and the many platforms that currently support it, and partly owing to its ability to exploit system services in UNIX quickly and easily.

pipe: The bandwidth of the connection in use between a user's workstation and the Internet (or the server on the other end of the connection, actually).

pixel: (abbreviation for picture element) A single group of phosphors on a CRT that creates a dot of color. When it is considered part of an image being displayed, this dot corresponds to a pixel.

plain text: Usually refers to vanilla ASCII text, as created or viewed in a simple text-editing program.

platform: Synonym for computer.

plugin: A Web browser add-in program that operates under the umbrella of the browser itself, thereby extending the program's overall capabilities (for example, Amber and Shockwave are both Netscape plugins).

port address: Lets a TCP/IP application know which program to talk to on the receiving end of a network connection. Because many programs may be running on a computer at one time — including multiple copies of the same program — the port address provides a mechanism to uniquely identify exactly which process the data should be delivered to.

portable document technology: See PDF.

PostScript: An Adobe page description language, used as a format for printing and display purposes by many programs and devices.

POTS (Plain Old Telephone System): The normal analog telephone system, just like the one you probably have at home.

PPP (Point-to-Point Protocol): A modern, low-overhead serial communications protocol, typically used to interconnect two computers via modem. Most Web browsers require either a PPP or SLIP connection in order to work.

preprocessor: A special program that parses source code or other text to translate embedded strings into (more verbose or complex) final forms before interpretation or compilation occurs.

property: A particular object attribute or its associated value.

proprietary: A data format, specification, or operation that's defined (and owned) by a company, rather than by a standards organization. Proprietary HTML markup is that markup created and used exclusively within certain browsers (such as the <CENTER> tag in Netscape, for example).

protocol: A formal, rigidly-defined set of rules and formats that computers use to communicate with one another.

provider: See service provider.

publication process: The process of gathering, organizing, and delivering to the Web information in the form of HTML documents and supporting materials.

Push: A Web-based information distribution technology that permits a special type of server (called a Push

server, naturally) to distribute information to a designated group of users, whenever new or changed content is posted to that server's information base. This technology lets changes and new materials be delivered to users almost as soon as such materials are recognized.

QED (Latin abbreviation for *Quod erod demonstrandum*, or "I have shown it"): A mathematical organization devoted to promoting effective uses of mathematics and computers, this group is advising the World Wide Web Consortium on the final form of HTML mathematics notation.

QuickTime: An Apple-derived multimedia/animation format. QuickTime is used across a broad variety of platforms today.

radio button: A type of interface control that's supported in HTML forms, where a selection of only one option out of a possible collection of options is permitted. Based on the old push-button radios so popular in cars in the '60s and '70s, the key concept is that only one selection at a time is allowed.

RAM (Random Access Memory): The memory used in most computers to store the results of ongoing work and to provide space to store the operating system and applications that are running at any given moment.

RealAudio: The audio helper application, from Progressive Network, that adds audio playback to a variety of Web browsers across multiple platforms.

relational database: A special type of database that organizes individual records into tables, where the columns specify the fields and attributes for all records, and the rows contain individual record instances.

relative: In the absence of the <BASE> tag in a URL, the link is relative to the current page's URL in which the link is defined. This arrangement makes for shorter, more compact URLs and explains why most local URLs are relative, not absolute.

request: In a client/server environment, clients request information to be delivered by a server. When you click a URL, you're implicitly requesting some server to deliver the corresponding HTML document or other resource to your workstation.

rescale: To resize a graphic image while maintaining its original aspect ratio and relative dimensions.

resource: Any HTML document or other item or service available via the Web. Resources are what URLs point to.

response: In a client/server environment, the server's reply to a user's request for information. The server supplies the information or provides some other kind of response that reports on the request itself (for example, invalid request or data not available).

return (short for carriage return): In text files, a return is what causes the words on a line to end and makes the display pick up at the leftmost location on the display.

RFC (Request for Comment): An official IETF standards document.

RGB (Red Green Blue): The name of a computer-based color representation scheme for graphics displays.

RGB code: A specific numeric value that corresponds to particular values for red, green, and blue components, used to designate particular colors and shades.

robot: A special Web-traveling program that wanders all over the place, following and recording URLs and related titles for future reference (for example, in search engines).

ROM (Read-Only Memory): A form of computer memory that allows values to be stored only once. After the data is initially recorded, the computer can only read the contents. ROMs are used to supply constant code elements, such as bootstrap loaders, network addresses, and other more or less unvarying programs or instructions.

root: The base of a directory structure above which no references are legal.

router: A special-purpose piece of Internet working gear that helps to connect networks together. A router is capable of reading the destination address of any network packet. The router can forward the packet to a local recipient if its address resides on any network that the router can reach or on to another router if the packet is destined for delivery to a network that the current router cannot access.

RTF (Rich Text Format): A platform-independent text representation often used as an intermediate format when converting one type of text, word processing, or page layout file into some other form. Also, the file extension for a Rich Text Format file.

sampling rates: The number of bits used to represent audio information over a particular time interval. The higher the sampling rate, the more true-to-life the corresponding sound information.

scene graph: Defines the ordering and precedence among all the VRML nodes in any virtual world. The earlier a node appears in a graph, the higher its precedence and the more impact it has on how the scene is rendered for display.

score: In Macromedia Director, the sequence of events and animation actions that are scheduled to occur as individual frames are played in sequence. The score also describes conditional processing and frame sequence alterations that can occur during playback.

screen: The glowing part on the front of your computer monitor where you see the Web do its thing (and anything else your computer shows you).

SEA (abbreviation for Self-Extracting Archive): File extension for a self-extracting version of the most popular Macintosh file compression format, Stuffit (.SIT).

search engine: A special Web program that can search the contents of a database of available Web pages and other resources to provide information that relates to specific topics or keywords supplied by a user.

search tools: Any of a number of programs that can permit HTML documents to become searchable by using the <ISINDEX> tag. This tag informs the browser of the need for a search window and behind-the-scenes indexing and anchoring schemes to let users locate particular sections of or items within a document.

security: A general term that describes how an operating system or network operating system establishes and manages user access to the resources and objects under that system's control.

server: A computer on a network whose job is to listen for particular service requests and to respond to those that it knows how to satisfy.

server root: Defines the root of a Web server's programs and documents. This concept is used to define the scope for file access permissions for users and administrators alike.

service provider: See Internet Service Provider.

setup: The phase at the beginning of the communications process when negotiating a network connection. At this point, protocol details, communication rates, and error-handling approaches are worked out, allowing the connection to proceed correctly and reliably thenceforth.

SGML (Standard Generalized Markup Language): An ISO standard document definition, specification, and creation mechanism that makes platform and display differences across multiple computers irrelevant to the delivery and rendering of documents.

shading: In a graphics environment, shading represents the results of the complex calculations that must occur to depict the effect of light and shadow on three-dimensional graphical objects.

shell: See UNIX shell.

Shocked: Often indicates that Web materials have been prepared for use with a Shockwave plugin for display.

Shockwave: The Macromedia technology (and plugin) used to create Internet-deliverable Director presentations and to display them within a Web browser.

single inheritance: The capacity to inherit attributes and characteristics from only one parent object (as is the case with Java) in object-oriented languages.

singleton: An HTML tag that has no corresponding closing tag. For example `<P>` and `<BASE>` are both singletons, even though `<P>` takes no attributes, and `<BASE>` does.

SIT (abbreviation for StuffIT archive): File extension for the native Stuffit Macintosh compressed-file format.

SLIP (Serial Line Interface Protocol): A relatively old-fashioned TCP/IP protocol used to manage telecommunications between a client and a server that treats the phone line as a slow extension to a network.

SMTP (Simple Mail Transfer Protocol): The underlying protocol and service for Internet-based electronic mail.

spider (Web spider, WebCrawler): A class of Internet software agents that tirelessly investigate Web pages and their links, while storing information about their travels for inclusion in the databases typically used by search engines.

SSL (Secure Sockets Library): A Netscape-designed Web commerce programming library, intended to help programmers easily add secure transactions across the Web.

stage: The display window where a presentation ultimately appears during playback in Multimedia Director.

stdin (UNIX standard input device): The default source for input in the UNIX environment. It is the input source for CGI programs as well.

stdout (UNIX standard output device): The default recipient for output in the UNIX environment. It is the output source for Web browsers and servers as well (including CGI programs).

style sheet: A document that rigorously describes how classes of markup are to be rendered in an HTML document display, including font selections, font styles, leading, kerning, and color schemes.

syntax checker: A program that checks a particular HTML document's markup against the rules that govern HTML's use; a recommended part of the testing regimen for all HTML documents.

syntax: Literally, the formal rules for how to speak. In this book, syntax describes the rules that govern how HTML markup looks and behaves within HTML documents.

T1: A high-speed (1.544 Mbps) digital communications link.

table: An HTML construct that allows data to be represented in onscreen areas called cells. The cells are organized into columns and rows.

tag: The formal name for an element of HTML markup, usually enclosed in angle brackets (`< >`).

TCP (Transmission Control Protocol): The transport layer protocol for the TCP/IP suite. TCP is a reliable, connection-oriented protocol that usually guarantees delivery across a network. See TCP/IP.

TCP/IP (Transmission Control Protocol/Internet Protocol): The name for the suite of protocols and services used to manage network communications and applications over the Internet.

Telnet: A means of running programs and using capabilities on other computers across the Internet. Telnet is the Internet protocol and service that lets you take a computer and make it emulate a dumb terminal over the network.

template: The skeleton of a Web page, including the HTML for its heading and footer and any consistent layout and navigation elements for a page or set of pages.

template-based search: A search that follows a set of values specified within a form.

tenant: A term applied to the Webmasters or administrators who manage a site located on somebody else's Web server (usually an ISP's).

terminal emulation: The process of making a full-fledged, stand-alone computer act like a terminal attached to another computer, terminal emulation is the service that Telnet provides across the Internet.

test plan: The series of steps and elements to be followed in conducting a formal test of software or other computerized systems. We strongly recommend that you write — and use — a test plan as a part of your Web publication process.

TeX: Donald Knuth's powerful typesetting environment, which pioneered the delivery of comprehensive, practical mathematical typesetting that defines the foundation on which HTML <MATH> notation is based.

text controls: Any of a number of HTML tags, including both physical and logical markup, text controls provide a method of managing the way that text appears within an HTML document.

text engine: A browser plugin or helper application that can render highly-formatted text and graphics based on a particular and specific format, either for printing or computer display.

text-mode: A method of browser operation that displays characters only. Text-mode browsers cannot display graphics without the assistance of helper applications.

throughput: The amount of data that can be "put through" a connection in a given period of time. Throughput differs from bandwidth in being a measure of actual performance, instead of a theoretical maximum for the medium involved.

thumbnail: A miniature rendering of a graphical image, used as a link to the full-sized version.

tiling: A technique for filling an entire region with graphics data that relies on taking a small area of graphics within that region and repeating it like a set of tiles to cover the area.

title: The text supplied between <TITLE> . . . </TITLE> defines the text that shows up on that page's title bar when displayed; the title is also used as data in many Web search engines.

transparent background: See transparent GIF.

transparent GIF: A specially rendered GIF image that takes on the background color selected in a browser capable of handling these GIFs. This process makes the graphic blend into the existing color scheme and provides a more professional-looking page.

tree structure: Computer scientists like to depict hierarchies in graphical terms, which makes them look like upside-down trees (a single root at the top, multiple branches below). File systems and genealogies are examples of tree-structured organizations that we're all familiar with, but examples abound in the computer world. This type of structure also works well for certain Web document sets, especially larger, more complex ones. See hierarchical structure.

unclosed elements: A marked region of HTML text that's missing a required closing tag (for example `<H1>This is wrong`).

UNIX: The operating system of choice for the Internet community at large and the Web community, too, UNIX offers the broadest range of tools, utilities, and programming libraries for Web server use.

UNIX shell: The name of the command-line program used to manage user-computer interaction. The shell can also be used to write CGI scripts and other kinds of useful programs for UNIX.

URI (Uniform Resource Identifier): Any of a class of objects that identify resources available to the Web. Both URLs and URNs are examples of URIs.

URL (Uniform Resource Locator): The primary naming scheme used to identify Web resources. URLs define the protocols to be used, the domain name of the Web server where a resource resides, the port address to be used for communication, and the directory path to access a named Web file or resource.

URL-encoded text: A method for passing information requests and URL specification to Web servers from browsers, URL encoding replaces spaces with plus signs (+) and substitutes special hex codes for a range of otherwise unreproduceable characters. This method is used to pass document queries from browsers to servers.

URN (Uniform Resource Name): A permanent, unchanging name for a Web resource. URNs are seldom used in today's Web environment. They do, however, present a method guaranteed to obtain access to a resource, as soon

as the URN can be fully resolved. (A URN can consist of human or organizational contact information, instead of resource location data.)

Usenet: An Internet protocol and service that provides access to a vast array of named newsgroups, where users congregate to exchange information and materials related to specific topics or concerns.

validator: A special token or password that accompanies a user's request for system resources, used to determine if the request may be satisfied or if it must be denied, based on the user's security profile.

VBScript: A special interpreted object-based scripting language developed as an offshoot of the Visual Basic programming language by Microsoft.

videoconferencing: A type of networked application where sounds and video signals are transmitted across a network, permitting users to communicate simultaneously via sounds and pictures.

ViewMovie: A Netscape plugin for viewing MOV animation files.

virtual shopping cart: A Web construct that lets users visit multiple Web pages, gathering a set of items that they've selected along the way, and pay for all items at once.

virus: A type of self-replicating program that seeks to distribute itself around a network, or the Internet, for either benign or malignant purposes.

VRML (Virtual Modeling Reality Language): VRML is a fully-fledged computer programming language designed to facilitate creation of complete, three-dimensional, graphical spaces called *virtual worlds*.

wanderer (a synonym for spider or robot) : A class of software agents that prowl the Web, following links and documenting what they find as they go. These programs provide much of the raw material that's organized by search databases for investigation by search engines.

watermark: A technique for maintaining a fixed background for HTML documents that stays the same, even as foreground materials scroll across the display.

Web: Shorthand for the World Wide Web (or W3). We also use Web in this book to refer to a related, interlinked set of HTML documents.

Web pages (a synonym for HTML documents): Sets of related, interlinked HTML documents, usually produced by a single author or organization.

Web server: A computer, usually on the Internet, that plays host to httpd and related Web-service software.

Web server administrator: The individual responsible for the setup, configuration, and maintenance of a Web server.

Web site: An addressed location, usually on the Internet, that provides access to the set of Web pages corresponding to the URL for a given site. Thus, a Web site consists of a Web server and a named collection of Web documents, both accessible through a single URL.

Webify: The process of converting a document of any kind into a Web-viewable format, typically by translation into HTML.

white space: The parts of a document or display that aren't occupied by text or other visual elements — the breathing room on a page. A certain amount of white space is essential to make documents attractive and readable.

Windows (MS-Windows): The astonishingly popular (and sometimes frustrating) GUI environment for PCs from Microsoft Corporation. Windows is the GUI of choice for most desktop computer users.

World Wide Web (WWW or W3): The complete collection of all Web servers available on the Internet, which comes as close to containing the "sum of human knowledge" as anything we've ever seen.

World Wide Web Consortium (W3C): The computer industry-funded technical organization that is jointly responsible for maintaining and managing HTML and related standards (along with the Internet Society's Internet Engineering Task Force, or IETF).

worm: A self-replicating computer program that seeks to visit as many locations on a network as possible. More of a nuisance than viruses, worms consume precious network bandwidth and are abhorred for that reason.

WWWInline: Another basic VRML node. WWWInline is often recommended as the root node when constructing a virtual world for Web display.

WYSIWYG (What You See Is What You Get): A term used to describe text editors or other layout tools (such as HTML authoring tools) that attempt to show their users onscreen what final, finished documents will look like.

X Windows: The GUI of choice for UNIX systems. X Windows offers a graphical window, icon, and mouse metaphor similar to (but much more robust and powerful than) Microsoft Windows.

Xobject: Externally defined data resources, such as sounds, graphics, or animated sequences, for inclusion within a Director film strip. Director support for Xobjects greatly expands the type and quality of materials that can be included within its playback environment.