

Valuable Bonus
Disk with Ready-
to-Use Templates
Enclosed!

"[This book] explained how to do things in English, not in
technoid. I was able to accomplish things in less time..."
— Melissa Olson, St. Paul, MN, on HTML For Dummies



NEW!
Revised &
Updated!

HTML FOR DUMMIES[®]

2ND EDITION

A Reference for the Rest of Us!

by **Ed Tittel**

Author of IDG Books'
NetWare[®] For Dummies[®], 2nd Edition

& **Steve James**

The Fun and Easy Way[™]
to Improve Your Web Pages
Using HTML (Hypertext
Markup Language)

Uncover the Latest on HTML
Tags Including Tables, Math,
Netscape, and Internet
Explorer Extensions

The World Wide Web
Publishing Process
— Explained in
Plain English

FREE HTML
"Cheat Sheet"
Inside!



MS-HOF002955

Petitioner has added romanettes to
pages (i)-(iv). Otherwise, it leaves
the original page numbering.

MICROSOFT CORP.
EXHIBIT 1022

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

Text and art copyright © 1996 by 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.: 96-75002

ISBN: 1-56884-647-9

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

2E/RR/QT/ZW/IN

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

Distributed by Macmillan Canada for Canada; by Computer and Technical Books for the Caribbean Basin; by Contemporanea de Ediciones for Venezuela; by Distribuidora Cuspide for Argentina; by CITEC for Brazil; by Ediciones ZETA S.C.R. Ltda. for Peru; by Editorial Limusa SA for Mexico; by Transworld Publishers Limited in the United Kingdom and Europe; by Al-Maiman Publishers & Distributors for Saudi Arabia; by Simron Pty. Ltd. for South Africa; by IDG Communications (HK) Ltd. for Hong Kong; by Toppan Company Ltd. for Japan; by Addison-Wesley Publishing Company for Korea; by Longman Singapore Publishers Ltd. for Singapore, Malaysia, Thailand, and Indonesia; by Unalis Corporation for Taiwan; by WS Computer Publishing Company, Inc. for the Philippines; by WoodsLane Pty. Ltd. for Australia; by WoodsLane Enterprises Ltd. for New Zealand.

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 translations, contact Marc Jeffrey Mikulich, Director, Foreign & Subsidiary Rights, at IDG Books Worldwide, 415-655-3018 or fax 415-655-3295.

For sales inquiries and special prices for bulk quantities, write to the address above or call IDG Books Worldwide at 415-655-3200.

For information on using IDG Books Worldwide's books in the classroom, or ordering examination copies, contact the Education Office at 800-434-2086 or fax 817-251-8174.

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 representation 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 any particular purpose and shall in no event be liable for any loss of profit or any other commercial damage, including but not limited to special, incidental, consequential, or other damages.

Trademarks: All brand names and product names used in this book are trademarks, registered trademarks, or trade names of their respective holders. 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.

Contents at a Glance



Introduction	1
Part I: Building Better Web Pages	7
Chapter 1: The Web's THE Place to Be!	9
Chapter 2: Getting Hyper	27
Chapter 3: What's in a Page?	39
Part II: A Tour of HTML Basics	55
Chapter 4: What's a Markup Language?	57
Chapter 5: Pigeonholing Page Contents: HTML Categories	69
Chapter 6: Introducing the Unrepresentable: HTML Entities	121
Chapter 7: Building Basic HTML Documents	131
Chapter 8: Beyond Basics: Adding Flair and Impact to Your Pages	151
Part III: Advanced HTML	171
Chapter 9: Going High-Rise: Building Complex Pages	173
Chapter 10: Strictly Pro Forma: Using Forms for Feedback	197
Chapter 11: The Map's the Thing!	213
Chapter 12: Stick Out Your Neck! HTML Extensions	225
Part IV: Beyond HTML? CGI Programs and "Real" Applications	243
Chapter 13: The Common Gateway Interface (CGI)	245
Chapter 14: Help Them Find Their Way: Aids to Web Navigation	265
Part V: Call the Exterminator! Debugging Web Pages	273
Chapter 15: Testing, Testing, 1-2-3	275
Chapter 16: It Doesn't Matter What YOU Think	285
Part VI: Going Public: Serving Up Your Web Pages	291
Chapter 17: So You've Woven a Web: What Now?	293
Chapter 18: If You Build It, Will They Come?	305
Chapter 19: The More Things Change	315
Part VII: It's Tool Time! HTML Development Tools and More	321
Chapter 20: Using UNIX Uniquely	323
Chapter 21: More Macintosh Madness	333
Chapter 22: Webbing Up to Windows	343
Part VIII: The Part of Tens	357
Chapter 23: The Top Ten HTML Do's and Don'ts	359
Chapter 24: Ten Design Desiderata	365
Chapter 25: Decimating Web Bugs	371
Chapter 26: Ten Ways to Decide to "Build or Buy"	377
Glossary	383
Index	399
License Agreement	413
About the Disk	415
Reader Response Card	Back of Book

Table of Contents

<i>Introduction</i>	7
About This Book	1
How to Use This Book	2
Assume Makes an A** Out of U & Me	3
How This Book Is Organized	3
Part I: Building Better Web Pages	3
Part II: A Tour of HTML Basics	4
Part III: Advanced HTML	4
Part IV: Beyond HTML? CGI Programs and "Real" Applications	4
Part V: Call the Exterminator! Debugging Web Pages	5
Part VI: Going Public: Serving Up Your Web Pages	5
Part VII: It's Tool-time! HTML Development Tools and More	5
Part VIII: The Part of Tens	5
Icons Used in This Book	6
Where to Go from Here	6
 <i>Part I: Building Better Web Pages</i>	 7
Chapter 1: The Web's THE Place to Be!	9
From Small Things, Big Things Sometimes Come	9
What is the Web, and where is it strung?	10
And now, a word from our sponsor	10
Before the Web: Other Internet Navigation Tools	11
FTP (no, it's not about delivering flowers — that's FTD!)	11
Burrowing around in Gopherspace	12
The beauty of mailing lists and electronic mail	12
Usenet	12
Wide-Area Information Service (WAIS)	13
Why is the Web a "Big Deal"?	13
Of Browsers and Search Tools	14
Lynx	14
Netscape	15
Internet Explorer	15
Mosaic	16
Uniform Resources on the Web	16
URLs hold the keys to the Web	16
Making those URL keys fit	17

Danger! Explosive Growth	18
A Scintillating Survey of the Web, Worldwide	19
Jumping-off points galore	19
Search pages, anyone?	20
Under the Hood: How the Web Works	20
Networking Takes Protocols	21
How Webs talk: the HyperText Transfer Protocol (HTTP)	21
The straight dope on HTTP	24
HTML: Hypertext Markup Language	24
Accessing the Web	25
Chapter 2: Getting Hyper	27
HTML Basics	28
Of Links and Sausages	28
Jumping around inside documents	29
Jumping across documents (and services)	29
You've Used Hypertext, without Even Knowing It	30
Beyond Text, There's Multimedia	31
Hyperhelpers: useful "helper" applications	33
The value of visuals	35
Mavens of multimedia	36
Bringing It All Together with the Web	37
Chapter 3: What's in a Page?	39
It's All in the Layout	39
What Are You Trying to Say?	40
Who's listening?	41
Design springs from content — and intent	41
Grab the audience's attention	44
They're after the goods . . . don't get in the way!	44
What should they remember?	45
Meet the Elements of Page Design	45
Tagging text	45
Titles and labels	46
Text and hypertext links	47
Using graphics effectively	48
Overcoming two-dimensional thinking	48
Part II: A Tour of HTML Basics	55
Chapter 4: What's a Markup Language?	57
A Markup Language Is Not a Form of Graffiti	57
A syntax is not a levy on cigarettes!	58
Elements of HTML syntax	59

Standard Generalized Markup Language (SGML)	61
Generalized Markup Covers Many Sins	62
Building better pieces and parts	63
Where HTML fits under the SGML umbrella	63
Welcome to HTML	64
Delivering content to a variety of platforms	64
The four-plus faces of HTML	65
HTML Elements!	66
Go to the head of the document	66
The bulk's in the document body	67
The good stuff's in the graphics and links	67
A footer may be optional, but it's a good idea	67
Ladies and Gentlemen, Start Your Engines!	68
Chapter 5: Pigeonholing Page Contents: HTML Categories	69
HTML Syntax Redux	69
Syntax conventions are no party!	70
Decoding a complex metacharacter example: ALIGN = ?	72
Interesting HTML properties	72
HTML Categories	75
HTML Tags	79
The rundown on attributes	80
Tag information layout	80
The HTML tag team	82
Chapter 6: Introducing the Unrepresentable: HTML Entities	121
Entities Don't Have to Be an Alien Concept	121
Producing Special Characters	122
Nothing Ancient about the ISO-Latin-1 HTML	123
Chapter 7: Building Basic HTML Documents	131
The Template's the Thing!	132
Page Layout: Top to Bottom	133
What's in a Name? Thinking Up Good Titles and Headings	135
Titles	135
Headings	136
Building Better Document Bodies	141
Textual sound bites — NOT!	142
Balanced composition	142
Controlling long pages	142
The bottom line on bodies	143
Good Bones: Building Strong Paragraphs	143
Listward Ho: Choosing List Structures	144
The unordered list	145
The ordered list	145
The definition list	146

Hooking Up: Linking Your Pages	147
Links to pages within your Web are relative	147
Links to the world outside of your Web are physical	148
Choose your hyperlinks with care	149
Chapter 8: Beyond Basics: Adding Flair and Impact to Your Pages	151
Borrowing Can Lead to Sorrow	151
Eye-Catchers: Logos, Icons, and Other Little Gems	152
Horizontal rules — but rainbow lines bring smiles	155
Colored balls beat list dots	155
Icons	156
Graphics as hyperlinks	157
Logos	157
Building Graphic Page Layouts	161
Working with graphics files	161
GIF and JPEG file formats	162
Rules for graphical thumbs	163
Footers Complete Your Page	163
Use a URL line as part of your page	166
Copyright	166
Counting coup: versions, dates, and times	166
Pointers to the Author or Owner	167
The e-mail link for feedback	168
Forms for feedback	168
Comment Your HTML Documents for Posterity	168
 Part III: Advanced HTML	 171
Chapter 9: Going High-Rise: Building Complex Pages	173
There's No Place Like Home	173
Organization	174
It's story (board) time, boys and girls!	177
First Things First: List 'Em Out	177
Sketch the Web	177
Board the whole story	178
Anchors Away: Jumping Around Your Documents	179
Linking to text in another page	179
Linking to text within a page: table of contents links	180
Jumping to Remote Pages	181
Hypertext links to outside resources	182
Jump pages	182
A special <LINK>	183
The Nesting Instinct: Lists Within Lists	184
Sampling Sophisticated Pages	186
HTML For Dummies home page	186
Shockwave Productions HTML page	191

Really Cool: Java, VRML, Shockwave	193
JAVA	194
VRML	194
Shockwave	195
Chapter 10: Strictly Pro Forma: Using Forms for Feedback	197
What HTML Forms Are For	197
Living within Your Forms Limitations	198
Beware of browser!	198
Sorry, servers.	199
What's in a Form?	199
Forms involve two-way communication	200
Tag! You're a form	201
Using Form Tags	201
Setting the <FORM> environment	202
Knowing what's (in)coming: the <INPUT> tags	203
Being <SELECT>ive	206
<TEXTAREA> lets users wax eloquent . . . or profane!	208
Formulating Good Attitudes	211
Chapter 11: The Map's the Thing!	213
Where Are You? (Using Clickable Maps)	213
Cosmic Cartography: What It Takes to Present Maps on the Web	214
Warning: different maps for different servers	216
Dealing with shapes in maps	216
Building and Linking to CERN Map Files	217
The menu bar map file	218
Creating and storing map files	218
Using map files	219
Defining a clickable map in your HTML document	219
Building and Linking to NCSA Map Files	219
The button bar map file	220
Creating and storing map files	220
Using map files	220
Defining a clickable map in your HTML document	221
"The Map Is Not the Territory"	222
Of Clickable Maps and URLs	223
Chapter 12: Stick Out Your Neck! HTML Extensions	225
If Your Browser Can't See It, Is It Really There?	226
The State of the HTML Art	226
A "simple explanation" for browser diversity	226
Viva la difference: adding value . . . or adding confusion?	227
What's in Store for HTML?	227
HTML 2.0 Rules!	228
HTML 3.0 is still developing . . . but it's already passé!	228

Proposed Standard HTML Extensions	229
Accessing Tag Information on the Diskette	238
Nonstandard HTML Extensions	239
Netscape's changes to HTML markup	239
New Entities:	240
Microsoft's Internet Explorer	240
Some Cool Colors	240
For Tag Info, See "tagindex.htm"	241
The Perils of (In)Compatibility	241

Part IV: Beyond HTML? CGI Programs and "Real" Applications 243

Chapter 13: The Common Gateway Interface (CGI) 245

The "Common Gateway" Is NOT a Revolving Door!	246
Describing CGI programs	246
What's going on in a CGI program?	247
What's in CGI Input?	248
Short and sweet: the "extended URL" approach	248
Long-winded and thorough: the input-stream approach	250
Handling environment variables	251
Forming Up: Input-Handling Programs	253
Coding CGI	254
Ladies and gentlemen: Choose your weapons!	254
Example 1: What time is it?	256
Example 2: Counting page visits	257
Example 3: Decoding clickable map coordinates	259
Installing and Using CGIs	263

Chapter 14: Help Them Find Their Way: Aids to Web Navigation 265

There's a Spider on the Web	265
Webcrawlers and Search Engines	266
Don't wait for the spider to catch your site — register it	267
Searching Your Documents for Details	268
The Bigger Things Get, The Easier It Is to Get Lost!	269
Documentary Integuments: Indexes, Jump Tables, and Internal Links ...	271
Doing Things the Database Way	271
Stay Away from Diminishing Returns	272
Where's the Search Leading Us?	272

Part V: Call the Exterminator! Debugging Web Pages ... 273

Chapter 15: Testing, Testing, 1-2-3	275
Why Test Your Pages?	275
What You Think Doesn't Count	276
Expect (and Test for) the Unexpected	276
In Vitro Vitrification: Alpha Testing Methods	277
WebBuilder, test thy HTML thyself	278
Iteration, iteration, iteration	280
Written test plans	280
Alpha Test Plan	281
Introduction	281
Schedule and resources	282
Tester form	282
System configurations	283
Test method and evaluation	283
Performance and functionality testing	284
Limits and boundary checking	284
The A-Team's Finished!	284
Chapter 16: It Doesn't Matter What YOU Think	285
Beta Testing	285
Finding beta testers	286
Cycling ahead of the gremlins	287
Gamma Testing and Beyond?	287
There's nothing crazy about a sanity check!	288
Stick it to me: the importance of feedback	288
Knowing When to Quit (Testing)	290
 Part VI: Going Public: Serving Up Your Web Pages	 291
Chapter 17: So You've Woven a Web: What Now?	293
Inviting the World to Your Door	293
Stay Focused on Your Purpose	294
Make Sure Your Users Know Who's in Charge!	294
Publish, or Perish the Thought!	295
It's good for business	295
. . . and it's good for pleasure!	296
Back to the content	296
Dealing and Wheeling: Making Arrangements with an	
Internet Service Provider	296
Where to draw the line?	298
What to work out with your provider	299
Webbing It Yourself: What's That Mean?	301
Understand the tariffs	301
UNIX, or no UNIX?	301
Try though you might, there's no escaping maintenance!	302

Are You Ready for Success?	303
The Answer to the Ultimate Question	304
Chapter 18: If You Build It, Will They Come?	305
Announcing Your Web Site to the World	305
Write a "semi-formal" announcement	306
Where to direct your announcement	307
Trolling the Usenet newsgroups	308
Niche or industry publications	309
The old-fashioned kind of networking	309
Staying on the Safe Side of Acceptable-Use Policies	310
NSFNET Backbone Acceptable-Use Policy	311
Giving Value Means Getting Value	312
Making Sure Your Web Catches the Right Prey	312
Chapter 19: The More Things Change	315
Quack! Quack! The Two-Dimensional Text Trap Redux	316
Who Says This Stuff Is Stale? (The Mold's a Dead Giveaway . . .)	316
Check in on your pages regularly, Doctor Web!	317
Keep your content current!	317
Do your links point to nowhere?	317
Is your HTML passé?	317
If You Ask Them, They'll Tell You	318
Keeping Up with Changes	318
Maintenance is an Attitude and a Way of Life!	319
When Things Change, They Also Break	320
 <i>Part VII: It's Tool Time! HTML Development Tools and More</i>	 <i>321</i>
Chapter 20: Using UNIX Uniquely	323
Diving for Treasure in the UNIX HTML Tools Sea	324
Standing Alone Amidst the UNIX HTML Editors	324
A.S.H.E.	325
Phoenix	326
HoTMetaL Pro	326
NaviPress	328
EMACS modes and templates	328
Filtering and Converting Your UNIX Files	330
WebMaker 2.1	330
WebWorks Publisher	331
UNIX Web Server Search	332

Chapter 21: More Macintosh Madness 333

Surveying the Orchard of Macintosh HTML Tools	333
Biting into Standalone HTML Editors for Macintosh	334
Webtor	334
HTML Pro	335
PageMill	335
HoTMetaL Pro	336
NaviPress	337
Netscape Navigator Gold	338
Tackling Text Editor Extensions and Templates	338
BBEdit HTML Extensions	339
BBEdit HTML Tools	339
Magnificent Miscellaneous Mac Tools	340
Web Server Primer	340
WebSTAR / MacHTTP Web server	340
NetWings	341

Chapter 22: Webbing Up to Windows 343

Surveying the Field of HTML Software Tools	343
Looking at Standalone HTML Authoring Systems for Windows	344
HoTMetaL Pro / HoTMetaL Free	345
FrontPage	346
NaviPress	347
HTMLed Pro	348
HotDog	349
InContext Spider	349
Netscape Navigator Gold	350
Word Processor Plug-Ins	351
Filters and File Converters	352
HTML Transit	352
WebMaker 2.1	353
WebWorks Publisher	353
A Windows Web Server Primer	354

Part VIII: The Part of Tens 357**Chapter 23: The Top Ten HTML Do's and Don'ts 359**

Remember Your Content!	359
Structuring Your Documents	360
Keeping Track of Tags	360
Making the Most from the Least	361
Building Attractive Pages	361
Avoiding Browser Dependencies	362
Evolution, Not Revolution	362
Navigating Your Wild and Woolly Web	363

Beating the Two-Dimensional Text Trap	363
Overcoming Inertia Takes Constant Vigilance	364
Chapter 24: Ten Design Desiderata	365
Creating Page Layouts	365
Building a Graphical Vocabulary	366
Using White Space	366
Formatting for Impact	367
Enhancing Content	367
Making Effective Use of Hypermedia	368
Aiding Navigation	368
Forming Good Opinions	368
Knowing When to Split	369
Adding Value for Value	369
Chapter 25: Decimating Web Bugs	371
Making a List and Checking It Twice	371
Mastering the Mechanics of Text	372
Lacking Live Links Leaves Loathsome Legacies	372
Looking for Trouble in All the Right Places	373
Covering All the Bases	374
Tools of the Testing Trade	374
Fostering Feedback	374
Making the Most of Your Audience	375
Chapter 26: Ten Ways to Decide to “Build or Buy”	
Your Web Services	377
Understanding Objectives	378
Counting Your Pennies	378
Projecting and Monitoring Traffic	378
How Much is Too Much?	379
Managing Volatility	379
Communicating Corporately	380
Reaching Your Audience	380
“Web-ifying” Commerce	381
Understanding Your Options	382
Overcoming Success	382
Glossary	383
Index	399
License Agreement	413
About the Disk	415
Reader Response Card	Back of Book

Chapter 1

The Web's THE Place to Be!

In This Chapter

- ▶ Defining the World Wide Web
 - ▶ Examining other Internet search tools
 - ▶ Making the most of the Web — browsers and search tools
 - ▶ Examining Web background and terminology
 - ▶ Looking at exploding Web growth
 - ▶ Interpreting Web pages
 - ▶ Accessing the Web
-

To understand HTML, you first have to understand the environment that it serves and the world in which that environment operates. HTML is a text-based markup language that provides the underpinnings for one of the most exciting information search and navigation environments ever developed: This environment is called the World Wide Web (Web, WWW, or W3, for short). The Web represents a major step forward in making all kinds of information accessible to average folks like you and me.

From Small Things, Big Things Sometimes Come

Tim Berners-Lee and his colleagues at the European Laboratory for Particle Physics (CERN) in Geneva, Switzerland, had no idea what they were starting when they began hacking together their ideas for the Web back in March of 1989. Nevertheless, they succeeded in starting a strange and wonderful thing that has taken the whole Internet community by storm. Their mission was to build an online system for ordinary users to easily share data, without having to master arcane commands or esoteric interfaces. By 1992, users outside CERN were creating Web pages, and developers were creating powerful, graphical browsers for a broad range of desktop computers and workstations. By 1993, the Web had emerged as the most popular tool for Internet *surfing*.

What's in the words?

As you read through this book, you'll probably encounter some words of Web jargon that you don't recognize along the way. You'll also encounter a fair number of acronyms, like HTTP, that may not make a lot of sense to you, either. Don't worry, this is pretty normal when tackling a new subject in the computer world, where gibberish is the norm and acronyms proliferate like mushrooms! The good news is we've included a

glossary at the end of the book that contains definitions for most terms that you may not know and for all acronyms that we could find herein. Therefore, if you see a word you don't know, check the glossary: You may find enlightenment there! If not, drop us an e-mail and make us explain the word to you; maybe we don't know what it means either!!

To get started surfing, you need a way to access the Internet, a Web browser, and information about where to enter the Web. After that, you can scan the information that shows up on your screen and follow chains of information for the rest of your life (without ever again having to come up for air).

But wait a minute! Before you get lost in its infinite strands, you may want to consider a few more details about the Web's workings. (But don't let us stop you from trying it out — just check in right here when you come back!)

What is the Web, and where is it strung?

By now, you probably have a vague idea that the Web is a vast, amorphous blob of text, image, audio, and video data that is scattered across networks and computers worldwide. Hence comes the name, World Wide Web.

And now, a word from our sponsor . . .

According to Tim Berners-Lee, one of the Web's chief architects (and a founding father for the original development at CERN), the World Wide Web is "the universe of network-accessible information, an embodiment of human knowledge. . . . It has a body of software, and a set of protocols and conventions. W3 uses hypertext and multimedia techniques to make the Web easy for anyone to roam, browse, and contribute to." (From a Web page written by Berners-Lee that you can find at <http://www.w3.org/pub/WWW/WWW/> entitled *The World Wide Web*). By working with HTML, you too can roam, browse, and contribute to the Web!

Before the Web: Other Internet Navigation Tools

To understand the extraordinary impact of W3, you may want to look at previous Internet navigation tools. These other tools require considerably more user expertise than do Web browsers. While you're taking the trip down memory lane, please keep the following in mind: Although Web browsers supplant the functionality of many navigation tools, they work with these tools as well. Through HTML links, browsers call on other services to locate and retrieve files, messages, and other goodies from the vast Internet storehouse.

FTP (no, it's not about delivering flowers—that's FTD!)

FTP (File Transfer Protocol) is a cross-platform tool for transferring files to and from computers anywhere on the Internet. *Cross-platform* means that you don't have to use the same kind of computer operating system to access files on a remote system. Figure 1-1 shows a graphical FTP menu; notice the PC file system on the left (what's on your machine) and the remote file system on the right (what's on the FTP server). By navigating the directories, you can copy files between the two systems, as your access rights allow. (In English, *navigating the directories* means finding the location in a file system where the files you're after reside; and *access rights* refer to your ability to see, copy, delete, or write files within the FTP server's directories.)

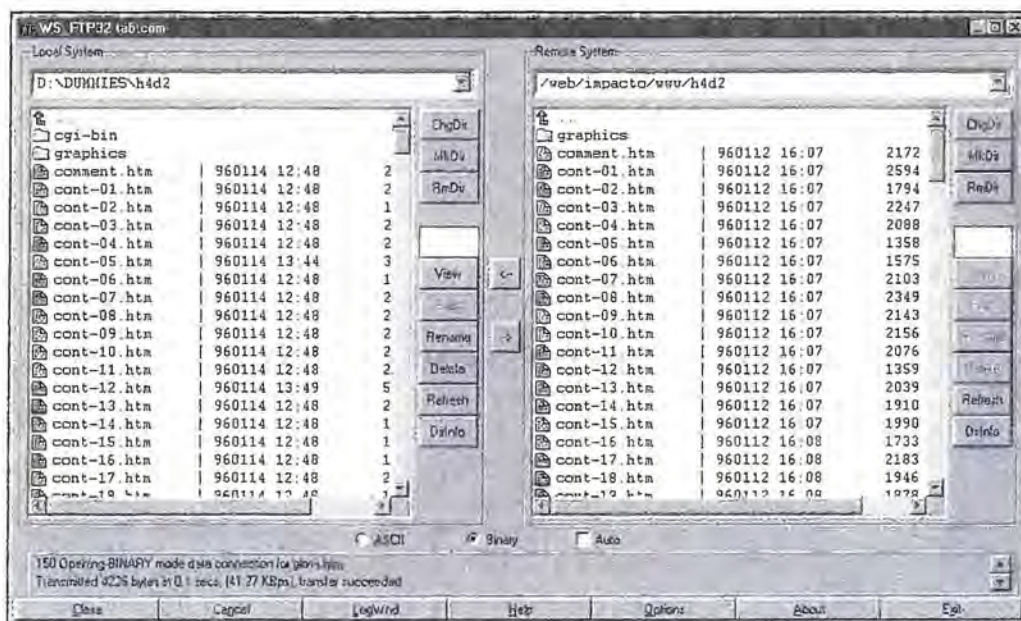


Figure 1-1:
A graphical
view of FTP.

Burrowing around in Gopherspace

Gopher is the creation of a team of dedicated programmers at the University of Minnesota, home of “The Golden Gophers.” More than just a totemic animal, Gopher is a good tool to use when browsing for files on the Internet. Gopher servers are extensively interlinked, much like the Web. In addition, all Gopher interaction occurs through a consistent menu interface that makes all systems look alike. (See Figure 1-2.) You can search by keyword or filename, so that you have more flexibility in finding your way around.

Figure 1-2:
The
primeval
Gopher
at the
University of
Minnesota.



The beauty of mailing lists and electronic mail

A little-known bit of trivia is that much information on the Internet is available through e-mail and through tools like FTP and Gopher. By stating the proper requests to the right e-mail servers — thereby accessing mail service programs such as *listserv* and *majordomo* — users with sufficient savvy can get to just about anything on the Internet.

Usenet

Usenet is a worldwide message system, where anyone can read and post articles. Usenet organizes its articles into named groups by topic and focus. These groups have varying degrees of internal organization — from strict moderation to freeform conversation. In some cases, you can approach Usenet with a specific question and come away with an answer immediately, but other queries can go unanswered for weeks on end. Persistence, coupled with an appreciation for Usenet’s workings and *netiquette*, is your key to success. (Figure 1-3 shows a listing of Usenet messages.)

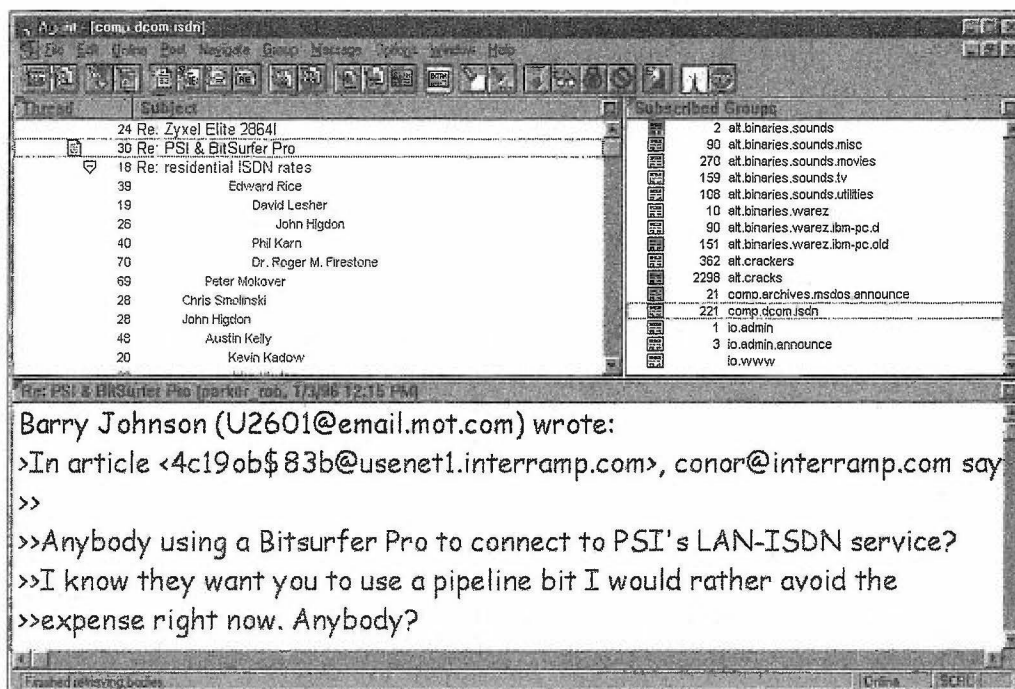


Figure 1-3:
A listing of
Usenet
messages.

Wide-Area Information Service (WAIS)

WAIS (pronounced "ways" or "wase") is one of a handful of Internet search tools that can spread itself across the network to scour multiple archives and handle multiple data formats. It searches tirelessly throughout the entire Internet for the information you request. WAIS can help you find things, especially when you're not sure of precise filenames, menu entries, or other name-specific information.



To learn more about the Internet, take a trip to your local bookstore. You'll find no shortage of Internet-related titles there. (We counted over 100 on our last visit!) You should pay particular attention to John R. Levine, Carol Baroudi, and Margaret Levine Young's *The Internet For Dummies*, 3rd Edition and Levine and Young's *More Internet For Dummies*, 2nd Edition, (both published by IDG Books Worldwide, Inc., 1995).

Why is the Web a "Big Deal"?

We hope that you can come up with answers to this question on your own by now. But we'll run this one down, just to be sure: The World Wide Web is a major development in information access on the Internet. W3 is a big deal because it covers an astonishing amount of ground; it makes finding the way around huge collections of data easy and intuitive for users; and it hides most of the ugly details of how to grab and use information on the Internet.

Any one reason would make the Web important and useful; all of them together make it a genuine step forward in the way we use and share information as a part of our daily lives.

Of Browsers and Search Tools

For most end users, their Web access software — called a *browser* or a Web client — is the most important piece of Internet software that they use. Today, you can find many browsers for PCs running Windows, a more limited selection for DOS-only machines, and several options for the Macintosh, UNIX machines, and other platforms. All graphical Web browsers share a common, point-and-click approach to interacting with information. Even character-based browsers, like Lynx, still make it easy to pick and follow links by selecting the appropriate highlighted text. In the next few paragraphs, we give you a cursory look at browsers.

Lynx

Lynx is a text-only Web browser. That is, it cannot display or deliver graphical or multimedia elements (although it can be configured to display graphics using an external file viewer on appropriate systems). Even so, Lynx provides useful Web functionality for users on so-called *dumb terminals* because it supports keyboard navigation and boldface display of hypertext links (which is where we think that the program got its name: Lynx = links; get it?). Figure 1-4 shows a sample Lynx display.

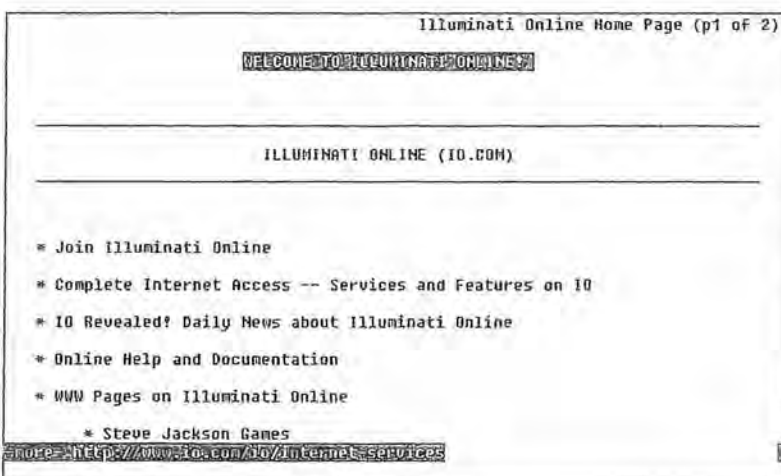


Figure 1-4:
Lynx uses
text displays
effectively
for Web
pages.

Netscape

Netscape, the brainchild of Marc Andreessen, is the Internet's most popular graphical Web browser. It offers clear evidence of its developer's wisdom and experience, and it includes some advanced features not found in other Web browsers. The list of features, advancements, and add-ons to Netscape is a list that boggles the mind — with additions made almost daily. Available both as shareware and in a commercial release, Netscape provides one of the best and most popular Web interfaces that we've encountered anywhere. See Figure 1-5 for a look at Netscape's welcome screen.

Internet Explorer

You can get Internet Explorer, Microsoft's entry into the Web browser arena, in the Plus! add-on package for Windows 95. However, you can obtain the most recent release from the Microsoft Web site at <http://www.microsoft.com>. (We're told that Microsoft is planning to release 16-bit versions that run on Windows 3.1, 3.11, and Windows for Workgroups; they're even planning a port to NT. Perhaps by the time you read this, you can find all these elements at the Microsoft Web site.) Although it's a new player in this game, Internet Explorer has already made significant wrinkles in the Web community. Like Netscape, it has proprietary HTML tag elements in its list of features. The Internet Explorer is quickly becoming a popular interface to the Web and promises to cover Windows users of all stripes soon!

Figure 1-5:
Netscape is
used by
over 70
percent of
Web
surfers.



Mosaic

Mosaic is a graphical Web browser developed by a team of programmers at the National Center for Supercomputing Applications (NCSA). One of the original programmers, Marc Andreessen, eventually left NCSA to form his own Internet service company — Netscape. Mosaic was the first graphical browser for the Web, and it continues to hold its own against the *newbie* browsers. You can find many flavors of Mosaic today, including ones for X Windows (primarily for UNIX systems), Microsoft Windows, and the Macintosh. Mosaic was the first *full-featured* graphical browser and has spawned many clones.

Uniform Resources on the Web

The most significant feature of the World Wide Web is its ability to shield and even protect users from the harsh UNIX environment of the Internet. With the Web, locating information is greatly simplified. You find Web resources identified with special names — called *URLs* or *Uniform Resource Locators* — which describe the protocols that you use to access the resources and point to the resources' Internet locations.

URLs hold the keys to the Web

As you examine a URL for a specific HTML file, it looks something like this:

`http://www.w3.org:80/hypertext/WWW/Addressing/Addressing.html#spot`
|—1—|—2—| 3 |—————4—————|————5————|—6—|

This URL is composed of six parts that work as follows:

1. **Protocol/data source:** For network resources, this part is usually the name of the protocol used to access the data that resides on the other end of the link. The syntax for this part of the name is as follows:

- ✓ **ftp://** — points to a file accessible through the File Transfer Protocol.
- ✓ **gopher://** — points to a file system index accessible through the Gopher protocol.
- ✓ **http://** — points to a hypertext document (typically, an HTML file) accessible through the HyperText Transfer Protocol.
- ✓ **mailto://** — links to an application that allows you to compose a message to send through e-mail to a predefined address.

- ✓ **news://** — points to a Usenet newsgroup and uses the Network News Transfer Protocol (NNTP) to access the information.
- ✓ **telnet://** — links to a remote log-in on another Internet computer, typically to select from a predefined menu.
- ✓ **WAIS://** — points to a Wide-Area Information Server on the Internet and provides access to a system of indexed databases.
- ✓ **file://** — indicates that the file is local and is not a public Web page (that is, not available outside your directory or local network). Use this syntax for local data (typically, HTML files from your desktop machine's hard disks or other drives), but note that the syntax can vary from browser to browser. If you're desperate for a more complete discussion of accessing local files, see the sidebar "URL syntax and punctuation for local file access."

2. **Domain name:** The domain name for the Web server where the desired Web page or other resource resides.
3. **Port address:** In most cases, the default port address for HTTP is :80 (and you can omit it), but you may see URLs with other numbers in use; this number identifies the process address that a Web session needs to connect with. In general, if a number appears in the URL, it's a good idea to include it (even if the number is the default :80).
4. **Directory path:** The location of the Web page in the Web server's file system.
5. **Object name:** The actual name of the HTML file for the desired Web page or the name of any other resource that you require.
6. **Spot:** Sometimes, getting users to the HTML file isn't enough: You want to drop them at a particular location *within* the file. By preceding the name of an HTML *anchor* with a pound sign (#) and tacking it onto the HTML file name, you can direct a browser to jump right to a specific location. Using this structure is especially handy for larger documents, where readers might otherwise need to do quite a bit of scrolling to get to the right information.

Making those URL keys fit



All in all, the most important thing to remember about URLs is to enter them *exactly as they're written* because they don't work if they're not exactly correct. When you use a Web browser, cutting and pasting URLs into a hotlist, bookmark, or text file is better than writing them out by hand because you reduce the possibility of a transcription error.



URL syntax and punctuation for local file access

When trying to access local files, you can first look for a menu selection in your browser that lets you search your local file system (like "Open File" or "Open Local File"). If that doesn't work, we've had pretty good results with the following approach:

```
file:///<drive ID>|<directory spec>/
      <filename>
```

Notice the three forward slashes after the colon. After the drive ID (which would be a letter for DOS or the volume name for Macintosh, NetWare, and so on), use a vertical

bar character (|) in place of a colon. Then, when specifying the directory path (spec), use forward slashes (/) to separate directory levels. Follow the path with the exact name of the file, and you should be able to access it with your browser.

If the preceding strategy doesn't work for you, look in your browser's Help file for enlightenment: Often, it's just waiting a few screens away. Search on a phrase such as "Open File" or "Open Local File," and you should get the information that you need.



For more information on URLs, consult this URL:

```
http://www.w3.org/hypertext/WWW/Addressing/Addressing.html
```

This resource describes the details for URL syntax and supported protocols and points to specifications and other documents on the subject. A word to the wise: The CERN site gets a lot of traffic, so you may get *timed out* trying to connect. (In English, *timed out* means that you wait forever, and then your browser tells you that it can't retrieve the page!) We've had good luck getting there on off-hours, like 3 a.m. eastern standard time.

Danger! Explosive Growth

The word *exploding* conjures up an image of something that isn't safe to crawl onto, but exploding is just what the Web is doing in terms of growth. Even though its introduction (in 1991) makes it one of the newest Internet applications around, the Web has already become the most popular Internet application of all time. According to the work of one Web aficionado, former MIT student Michael Gray (now a principal at net.genesis, a successful Internet consultancy), the Internet has an annual growth rate of over 1,000 percent (or it increases 10 times each year, if you like smaller numbers)!



Wherever you get your statistics about W3, you find the unanimous opinion that Web usage is growing dramatically and user ranks are swelling robustly. The only real question then becomes, "How can I possibly manage to find what I really need out there on the Web?" For Web publishers, this question translates into, "How can I let the people know where my pages are?"

Following are the URLs for the references that we mentioned here:

✓ Michael Gray's "Growth of the Web"

<http://www.netgen.com/info/growth.html>

✓ A good general source for Web and Internet statistics with links to most other Web and Internet surveys and statistics worth mentioning (including "official" and personal efforts in that direction)

<http://WWW.Stars.com:80/Vlib/Misc/Statistics.html>

✓ We also recommend that you check out the Graphics, Visualization, and Usability Center's fourth WWW User Survey at

http://www.cc.gatech.edu/gvu/user_surveys/survey-10-1995/

A Scintillating Survey of the Web, Worldwide

At this point, you should have some idea about the Web's origins; now you can take a quick look at some of the many treasures that W3 offers.

Jumping-off points galore

Every browser comes with a predefined home page; many offer excellent starting points for your Web travels. Three browsers in particular — Netscape, Internet Explorer, and NCSA Mosaic — offer outstanding home pages with "Starting Points," topic indexes, and search capabilities to help you locate items of interest. However, the Web is a mystical thing — whose circumference is nowhere, and whose center, everywhere — so no "perfect starting point" really exists.

Search pages, anyone?

You can find a variety of search pages on the Web. These pages provide a link to background applications that can examine loads of data repositories on the Internet and, based on keywords that you supply, can return URLs matching the topics that you want.



The major search pages are nicely represented in a number of places, but we find the Yahoo! "Searching the Web" page to be most useful. The URL for this page is

```
http://www.yahoo.com/Computers_and_Internet/Internet/World_Wide_Web/  
Searching_the_Web/
```

Try it on for size; pick a search engine and try a search with a term of particular interest. (For best results, pick something specific like *coriander*, instead of something general like *spices*.)

As you travel around the Web, pay attention to page layouts and to the use of indexes, graphics, and hotlists. You can glean a great deal from these examples, both good and bad. You can also select View Source from your browser's menu to see the HTML code that represents each of these pages!

Under the Hood: How the Web Works

Now that you know what the Web is, where it came from, and what a big deal it has become, it's time to start grappling with how W3 actually works. Despite the volume of connected information and the different ways of presenting and delivering that information, the Web works through a basic set of mechanisms.

The Web is more than just the browser that you use on your desktop. The hidden structure of the Web is just as important as the utility that you use to access it. The Web has two labor-handling divisions: storage/retrieval and display/input. The Web *server* (located elsewhere on the network) typically handles the storage and retrieval part. The browser on the user's workstation (sometimes called a *client*) handles the display of information and recognition of input, when appropriate.

In the grand scheme of the computer world, this approach to handling information delivery is called *client/server computing*. Client/server has become an industry buzzword; nevertheless, the approach does confer some appreciable benefits:

- ✓ Because this approach divides the processing load, clients can concentrate on providing the best possible interface to users. Also, the client's concentration simplifies the offering of cool graphical displays and powerful visual controls.
- ✓ Likewise, servers can concentrate on maximizing their ability to service lots of requests; the division of labor lets the Web servers handle tens of thousands of resource requests per day.
- ✓ Another benefit of the client/server split derives from the location where the server stores the information that clients use: Residence on a server makes information ideal for sharing; permits better control over the data; and lets information providers decide how much power and capability that they want their servers to provide.
- ✓ By keeping dollars and data concentrated in one place — namely, at the server — the client/server approach helps to maximize performance on the server (where it's going to do the greatest good for the most users). The server environment also ensures information protection through backups, more rigorous control of data access, and accurate logging of request statistics (known as *hits*).

In short, clients handle the job of user interaction, and servers tackle the rapid retrieval and delivery of information. Client/server capabilities are well-realized on the World Wide Web, which features powerful, graphical clients (browsers) and fast, powerful servers. By working together, these elements contribute to the Web's burgeoning popularity.

Networking Takes Protocols

In diplomatic circles, a protocol is a set of rules that keeps professionals, friends and enemies alike, from making fools of each other (or themselves). For networks, methods of bulletproof communication are equally necessary and appreciated. Thus, you shouldn't be surprised to learn that the rules and formats that govern the methods by which computers communicate over a network are also called *protocols*.

How Webs talk: the HyperText Transfer Protocol (HTTP)

HTTP is an Internet protocol for a specific application — the World Wide Web. It provides a way for Web clients and servers to communicate with one another, primarily through the exchange of messages from clients (like "Give me this." or "Get me that.") and servers (like "Here's the page you asked for." or "Huh? I can't find what you're looking for.").



Acronymophobes, beware!

One thing you've got to realize, if you're going to become a real WebMaster, is that when you climb onto the Web, you're joining the Internet community. If there's ever been an unabashed bastion for acronyms — those multiletter combinations that nerds use to refer to things like personal computers (PCs), a disk operating system (DOS), random-access memory (RAM), or a compact disc, read-only-memory (CD-ROM) — it's with the Internet crowd.

Because we're talking about Web lore, the term *WebMaster* is a ubiquitous name for a person who holds the Web protocols on high, is a

veteran of the Web trenches, and lives, eats, and breathes the Web. You might be lucky and good enough to be called a WebMaster yourself — someday!

So, if your most fiendish nightmare is of drowning in a bowl of alphabet soup, maybe you'd better rethink your Web-oriented efforts! That's because networking in general, and the Internet in particular, is a field that revels in acronyms. When it comes to discussing Internet protocols, you'll find no better gathering spot for bizarre alphanumeric combinations.

To fully understand HTTP, you need to fully understand TCP/IP. A longish sort of acronym, *TCP/IP* stands for Transfer Control Protocol/Internet Protocol, which is the name given to the full set of protocols used on the Internet. But to begin writing good Web documents, you don't need to know much about either topic. However, we have included a list of reference materials for you masochists out there.



Welcome to the Nebulous Zone . . .

It's a place where different kinds of computers can freely exchange information with one another, where mere implementations bow to the demands of an all-encompassing standard. TCP/IP is a world unto itself: More bits use TCP/IP in a day on today's Internet than are required to store every piece of printed material known to mankind before 1950.

When it comes to TCP/IP, there's a lot to learn and a lot to know. Covering TCP/IP in any depth is way beyond the scope of this book. Therefore, we'd like to give you some choice references:

- ✓ The Internet is the subject of at least two other . . . *For Dummies* books: *The Internet For Dummies*, 3rd Edition (IDG Books Worldwide, Inc., 1995) by John R. Levine, Carol Baroudi, and Margaret Levine Young; and *More Internet For Dummies*, 2nd Edition (IDG Books Worldwide, Inc., 1995) by Levine and Young. Both books are a good place for beginners to start investigating the basics of TCP/IP.
- ✓ John Quarterman and Smoot Carl-Mitchell are the authors of *Practical Internetworking with TCP/IP and UNIX* (Addison-Wesley, 1993). This book is aimed at the system or

network administrator for a TCP/IP network who wants to understand how and why things work.

- ✓ *TCP/IP For Dummies* by Marshall Wilensky and Candace Leiden (IDG Books Worldwide, Inc., 1995) is a great place to continue your TCP/IP investigations. In addition to covering the topic in wonderfully amusing detail, the book provides a gentle introduction to TCP/IP that is hard to beat.
- ✓ Matthew Flint Arnett is the first in a series of 14 coauthors for *Inside TCP/IP* (New Riders Press 1994), another book aimed at helping those who must run a TCP/IP network or internetwork, or those who must oversee an Internet connection.
- ✓ O'Reilly & Associates covers TCP/IP with a Nutshell handbook for UNIX system administrators, *TCP/IP Network Administration*, by Craig Hunt.
- ✓ A truly definitive look at TCP/IP comes from Douglas E. Comer, author of *Internetworking with TCP/IP*, a 3-volume set (Prentice-Hall 1991, 1991, 1993; Volumes 2 and 3 were authored with David L. Stevens). Comer's books are widely regarded as the best general references on the subject.

- ✓ Another comprehensive 2-volume treatise on TCP/IP, called *TCP/IP Illustrated, Volumes 1 and 2* is available from W. Richard Stevens (assisted by Gary R. Wright on the second volume, Addison-Wesley 1994). These books are more up to date than Comer's, and they offer detailed "war stories" taken straight from life on the Internet. For a reference that brings many salient TCP/IP details together in one place, this set is a good choice.

- ✓ The ultimate authority on TCP/IP comes from a standards body called the Internet Architecture Board (IAB). Within the IAB, the Internet Engineering Task Force (IETF) drafts and maintains Internet standards of all kinds, including those for protocols, in the form of numbered documents called "Requests for Comment" (RFCs). See Table 1-1 for more information on the IETF RFCs.

For a listing of all the current protocol-related RFCs, consult RFC 1720 "Internet Official Protocol Standards," which is available in at least three ways. (If 1720 isn't current anymore, it'll tell you it's been made obsolete by a new document, and you can follow a link to the new reigning standard.)

If you take the time to poke around in the RFC collection, you'll be going straight to the horse's mouth, where TCP/IP and related matters are concerned!

Table 1-1 **Three methods for examining RFCs**

<i>Service</i>	<i>Method</i>
e-mail	Send e-mail to <code>mailserv@ds.internic.net</code> and type <code>file/ftp/rfc/rfc1720.txt</code> in the message body
FTP	Anonymous FTP to <code>ds.internet.net</code> (password = your e-mail address). Look in directory <code>rfc/</code> for the file named <code>rfc1720.txt</code>

(continued)

Table 1-1 (continued)

Service	Method
Web	<URL: http://www.cis.ohio-state.edu/htbin/rfc/rfc1720.html > for the contents of RFC 1720 <URL: http://www.cis.ohio-state.edu/hypertext/information/rfc.html > has general RFC information

The straight dope on HTTP

Protocols that link clients and servers together must handle requests and responses. Consequently, you'll not be surprised to know that information exchanges on the Web happen in four parts, all classed as specific message types for HTTP.

- ✓ **Connection:** Occurs when a client tries to connect to a specific Web server (your browser may display a status message such as Connecting to HTTP Server). If the client can't make the connection, the attempt usually times out and the browser displays a Connection timed out message.
- ✓ **Request:** The client asks for the Web resource that it's looking for. The request includes the protocol to use (which indicates the type of resource), the name of the object to find, and information about how the server should respond to the client.
- ✓ **Response:** Now it's the server's turn. If the server can deliver the requested object, it responds with a delivery in the manner requested by the client. If it can't deliver, the server sends an error message explaining why not.
- ✓ **Close:** After the server transfers information in response to the request, the connection between client and server is closed. You can easily reopen the connection with another request — for example, by clicking on a link in the current object — that jumps you back to reestablish the connection.

After it completely transfers the requested object, HTTP has done its job. Then the browser must interpret and display what the server delivered, and another strand in the Web unfurls.

HTML: Hypertext Markup Language

After the Web server returns the response to a Web request, the browser takes over to interpret and display the information.

HTML is a *markup language* that describes the structure of a Web document's content plus some behavioral characteristics. All Web browsers are able to understand and interpret this standard language. HTML is a subset of a larger markup language, SGML, called Standard Generalized Markup Language. (That's as much as you need to know about SGML in order to write Web documents.)

HTML is a way of representing text and linking that text to other kinds of resources — including sound files, graphics files, multimedia files, and so on — that allows the concurrent display of different kinds of data and lets different resources augment and reinforce one another.

As delivered by a Web server, HTML is nothing more than a plain text file that includes two kinds of text:

- ✓ **The content:** Text or information for display or playback on the client's screen, speakers, and so on
- ✓ **The markup:** Text or information to control the display or to point to other information items in need of display or playback

Also, a browser must be prepared to convert a third kind of data — encoded files — and to hand them off to the right kind of helper application. This hand-off may involve a graphics program for an icon or image, a sound player program to handle audio, a video player program to play back a video file, or any other program necessary to reproduce a particular kind of information.

HTML files include both control information (tags) and content (text), which together describe the appearance and contents of Web pages. The HTML language also provides the mechanisms for tying in other Internet protocols and services on the Web — like FTP, Gopher, Usenet, e-mail, WAIS, telnet, and HTTP — so that Web pages can deliver many kinds of resources.

Accessing the Web

The crucial ingredient in gaining access to the Web — the one that lives up to the “World Wide” in its name — is the attachment to the Internet. In fact, the biggest constraint on your enjoyment of the Web is likely to be the size of the pipe that connects you to the Internet (and it to you). The term *pipe* refers to how much data the connection between you and the Web server can accommodate; like a water pipe, the more capacious the connection between you and your server, the faster things can move. Because waiting for screens to complete is a big drag, the faster the data goes, the better you'll like it!

Following are the two basic ways to go for your Internet link-ups:

- ✓ Over the telephone system and into another computer or network that's connected to the Internet
- ✓ Over a network and onto the Internet (or onto another computer that's properly connected)

You need to contact your ISP (Internet service provider) to get detailed information on how to maximize your speed and access to the Web.

Chapter 2

Getting Hyper

In This Chapter

- ▶ Understanding basic HTML concepts
 - ▶ Linking up the strands in the Web
 - ▶ Looking for hypertext examples
 - ▶ Getting past hypertext — to hypermedia
 - ▶ Going for the graphics
 - ▶ Dealing with multimedia display/playback
 - ▶ Bringing multiple media together on the Web
-

The real secret behind the Hypertext Markup Language is that there is no secret: Everything's out in the open in an HTML document, just waiting for the right interpretation. The beauty of HTML is its simple content — just a stream of plain characters — which makes virtually any text editor a potential HTML generator. The challenge of using HTML is working with its sensitivities. That is, HTML relies on the order in which its characters occur and the way that they're used to produce the right results.

Even though HTML can be forgiving (when you use *some* browsers to view documents with *certain* omitted or misstated elements), the best way to use HTML is to understand and work within its structure. Because your readers can use so many different browsers, the only way to get consistent Web page appearance and behavior is to know the rules for creating HTML documents and to use them to your readers' advantage!

This chapter presents the fundamental ideas behind HTML and introduces the concepts and operation of hypertext. Along the way, we hope that you begin to appreciate some of the basic principles behind building well-structured, readable Web pages.

HTML Basics

HTML stands for HyperText Markup Language. Its name reflects the two key concepts that make it work (and that make the World Wide Web such an incredible phenomenon):

- ✓ **Hypertext:** A way of creating multimedia documents; also a method for providing links within and between documents
- ✓ **Markup language:** A method for embedding special tags that describe the structure as well as the behavior of a document (not a way of discussing a preschooler's efforts with crayons on the wall!)

The simplicity and power of HTML markup let anyone create Web documents for private or public use. The power of hypertext, with its built-in support for multimedia and document links, creates the threads that compose the incredible breadth and reach of the Web. Making Web documents is so easy and straightforward that anyone can do it — as long as you can play the game by its rules.

Of Links and Sausages

HTML supports links within the same document and to completely different data elements elsewhere on the Web. Both types of links work the same way: You simply put the correct HTML tags around text or graphics to create an *active* (linked) area. Then when readers visit your Web site and click on the active area, they're transported to another spot within the same file, to another document in the same Web, or off to another place somewhere out on the Net. Figure 2-1 shows the *HTML For Dummies Hotlist* that includes links to several Web resources. (They're the items beneath the heading "Here are a few of our favorite Web spots:") Each of these links can carry you off to a valuable source of information.

The method of indicating a link varies from one browser to the next, but all browsers give some kind of visual clue that you're selecting an active area of the screen. You may see text underlined in a bright color, the font changed to bold, or a graphic outlined in a contrasting color. You may see image maps without direct visual clues other than a changing set of coordinates (in the browser's status line) that correspond to the location of your mouse cursor on-screen. Using this type of link is a bit like playing an adventure game: You're not always sure where you're going or what you'll find after you get there!

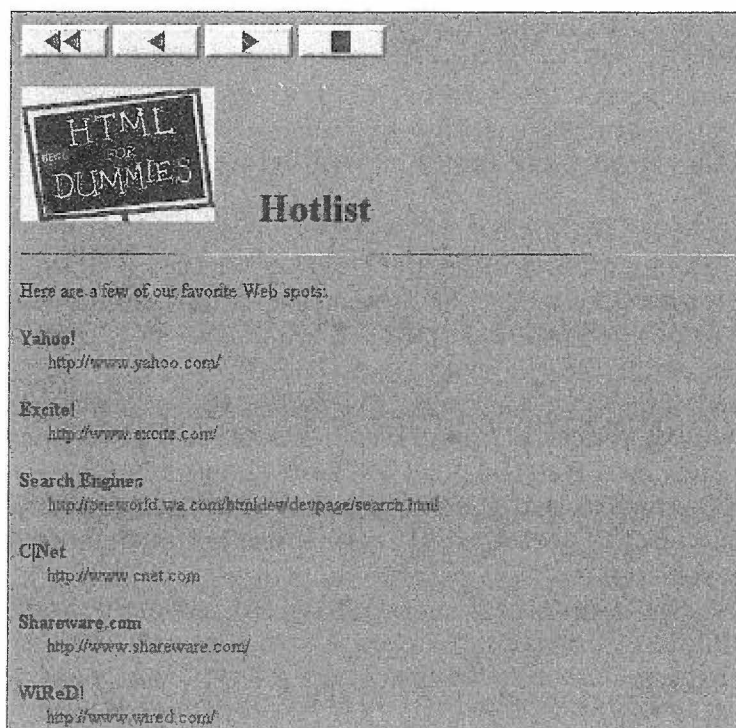


Figure 2-1:
A simple
HTML link
example.

Jumping around inside documents

One variety of link connects points inside the same document; such links are called *internal anchors*. WebMasters often use this kind of link from a table of contents (at the top of an HTML file) to the related sections throughout the document. Also, you can use this method to jump directly to the start of the document from its end; it beats the heck out of scrolling up through a long file.

This points to a key advantage of hypertext — namely, the capability to circumvent the linear nature of paper documents by providing rapid, obvious ways to navigate within (or among) documents. Sure, you can do the same thing with the table of contents or the index of a paper book by looking up and flipping the pages yourself (what a pain!). Hypertext automates this task for you, so that you can effortlessly jump around inside and between documents by creating links that are a snap (or actually, a click) to use.

Jumping across documents (and services)

By using a combination of links and Uniform Resource Locators (URLs), HTML can link your page to other Web pages, no matter where they are on the Web. Because URLs can reference a variety of protocols and services on the Internet (not just other HTML files), you can link through telnet, WAIS, Gopher, FTP, Usenet newsgroups, or even e-mail.

The same hypertext technology that lets you jump around inside a document also lets you reach any other resource available through the Web, as long as you know its exact URL (address). HTML's simple, text-tagging technique erases the difference between *here* and *there*. With a single click of the mouse or selection of the right text field, you are there, or the information is here. . . . Anyway, you get it.

Creation of the Web as we know it today comes from the work of individual Web weavers who include links to other documents from their Web sites. That is, the documents may constitute the Web, but the links among the documents reveal the Web's true power. Those links make up the strands in the Web that tie people, ideas, and arbitrary locations together.



Whenever you reference a URL in an HTML document, try to cut and paste that URL from a browser if at all possible. You'll save time on typing — some URLs can get pretty long — and even better, you'll be sure to get the reference right. For an additional check, test the URL with a browser to make sure that it's still valid before you copy it.

You've Used Hypertext, without Even Knowing It

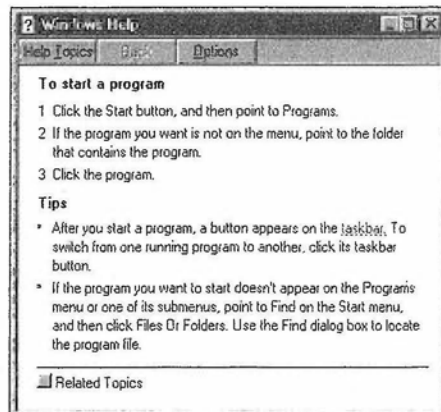
At this point, you may be saying something like, "What is this hypertext stuff?" Although you may see hypertext as strange and exotic, you're probably much more familiar with it than you think. Your own desktop computer undoubtedly contains several hypertext applications that you use regularly.

For example, if you're a Microsoft Windows user, every time you run the Windows Help utility, you're using a hypertext application. Figure 2-2 shows a Microsoft Windows Help screen. Notice the buttons near the top of the screen: "Help Topics," "Back," and "Options." These buttons work much like the navigation tools for a Web browser; they can let you return to where you've been (Back), or examine the range of available links within a document (Help Topics). The underlined phrases (such as, "taskbar") are links to text within the help system. They work much like the HTML links via your browser.

Macintosh users can identify HyperCard or SuperCard as forms of hypertext; others will have to take our word for it and not assume that these are games of chance! Here again, these applications have concepts in common with the Web: some kind of home page, various navigation tools (shaped arrows and specific commands), multimedia data, and plenty of links to select.

Likewise, UNIX users who are familiar with FrameMaker or other multimedia authoring tools can find some common ground with the Web's hypertext capabilities. Once again, the concepts of a home page, document navigation

Figure 2-2:
The
Microsoft
Windows
Help engine
is a
well-known
example of
hypertext.



and linkage, and integrated multimedia help to realize the notion of hypertext and how it operates.

The difference between the preceding examples and the Web is the kinds of links that they support: Only the Web offers the capability of jumping across the Internet and following links to other Web documents and servers. Enabling this feature is what gives HTML its unique value and, of course, what's made the Web possible.

Beyond Text, There's Multimedia

If you include nontext files (such as sound, graphics, and video) in Web pages, you must employ a certain amount of alchemy. Shipping Web information in a format called *MIME* (Multipurpose Internet Mail Extensions) makes it possible for a Web server to deliver multiple forms of data to your browser in a single transfer. Actually, MIME is a technique designed to bundle attachments within individual e-mail files. The following paragraphs tell a little more about how this works.

After a MIME file with attachments shows up at your workstation, additional processing begins immediately. The text portion of the message file arrives first. It contains the text-only HTML page description and lets the browser get right to work building and displaying the text portions of the page. Because the text arrives first, you may see placeholders or icons for graphics when you first see the Web page. Then the graphics or other forms of data replace these placeholders after their related attachments arrive.

While the user views the Web page, the browser receives attachments in the background. As they come in, the browser identifies these attachments by a file type or by description information in the attachment tag (as specified by the MIME format). After the browser identifies a file, it can handle the file's playback or display. Table 2-1 shows a list of common file types used on the Web, including expansions for the inevitable acronyms that such files often entail.

Table 2-1 Common Sound, Graphics, and Motion-Video Formats on the Web

<i>Extension</i>	<i>Format</i>	<i>Explanation</i>
Sound Formats		
RA	RealAudio	Used with RealAudio Web Server and RealAudio Player add-on for browsers.
SBI	Sound Blaster Instrument	Used for a single instrument with Sound Blaster cards (multiinstrument: IBK).
SND, AU	8 kHz mulaw	Voice-grade sound format used on workstations (such as Sun, NeXT, HP9000, and so on).
WAV	Microsoft Waveform	Sound format used in Windows for event notification.
Still-Video (Graphics) Formats		
GIF	Graphics Interchange Format	Compressed graphics format commonly used on CompuServe, easy to render multiplatform. Can be interleaved or not, depending on how image is created.
JPEG, JPG	Joint Photographic Experts Group	Highly compressed format for still images, widely used for multiplatform graphics.
PDF	Portable Document Format	Adobe's format for multiplatform document access through its Acrobat software.
PS	PostScript	Adobe's type description language, used to deliver complex documents over the Internet.
XBM	X-Window Bitmap	Image bitmap used by X-Windows, primarily on UNIX workstations.
Motion-Video Formats		
AVI	Audio Video Interleaved	Microsoft's Video for Windows standard format; found on many CD-ROMs.
DVI	Digital Video Interactive	Another motion-video format, also found on CD-ROMs.

<i>Extension</i>	<i>Format</i>	<i>Explanation</i>
FLI	Flick	Autodesk Animator motion-video format.
MOV	QuickTime	Apple's motion video and audio format; originated on the Macintosh, but also available for Windows.
MPEG, MPG	Motion Picture Experts Group	Full-motion video standard using frame format similar to JPEG with variable compression capabilities.



Several Web sites contain large amounts of information on file formats and programs.

- ✓ **Common Internet File Formats:** Contains an annotated list of audio, graphic, and multimedia file formats with links to applications that use them.

<http://www.travelresource.com/form-pc.html>

<http://www.travelresource.com/form-mac.html>

- ✓ **GRAPHICS:** Maintained by Martin Reddy at the University of Edinburgh, this site contains "everything you ever wanted to know" about graphics and links to additional resources.

<http://www.dcs.ed.ac.uk/%7Emxr/gfx/>

- ✓ **Multimedia File Formats on the Internet:** A Beginner's Guide for PC Users, by Allison Zhang.

<http://ac.dal.ca/~dong/contents.htm>

Hyperhelpers: useful "helper" applications

When referenced by a Web page, nontext data shows up as attachments to the HTML file. Sometimes the browser itself handles playback or display, as often is the case for simple, two-dimensional graphics (including .GIF and .JPEG files). Even so, these may be handled by other applications (especially when using character-mode Web browsers like Lynx).

But when other kinds of files show up and need special handling capabilities (beyond the scope of most browsers), the browser hands off the files to other applications for playback or display. These *helper* applications have the built-in smarts to handle the formats and processing necessary to deliver the contents of specialized files on demand. The process normally works something like this:

- ✓ The browser builds a page display that includes an active region (underlined or outlined in some way) to indicate the attachment of a sound, video, or animation playback.
- ✓ If the user selects the active region (the link), the browser calls on another application to handle playback or display.
- ✓ The other application takes over and plays back or displays the file.
- ✓ After the helper completes the display or playback, the browser reasserts control, and the user can continue on (or select the active region again, and get another playback or display).

A standard part of browser configuration supplies the names and locations of helper programs to assist the browser when such data arrives. If the browser can't find a helper application, it simply doesn't respond to an attempt to display or play back the requested information.

For example, WPlany and Wham are two common shareware sound player applications for PCs running Windows. As part of their configuration or setup, most browsers supply a method for linking particular file types (like the .SBI and .WAV file extensions common on the PC) with a helper application.

By establishing an association between the sound playback application (WPlany or Wham) and the related audio file extensions (.SBI and .WAV), the browser automatically launches the application you designate when it encounters files with those extensions. This causes the sounds to play (which, we assume, is a good thing!).



For comprehensive listings of PC, Mac, and UNIX helper applications and links to their sources, start your search at one of the following URLs:

<http://www.travelresource.com/help.html>
<http://charlotte.acns.nyu.edu/internet/helper/>
http://www2.netscape.com/assist/helper_apps/index.html
<http://wwwhost.cc.utexas.edu/learn/use/helper.html>

Some useful helper applications for Windows include

- ✓ **For still graphics:** Lview is a good, small graphics viewer that can handle .GIF, .PCX, and .JPEG files. It also supports interesting image editing capabilities. (See Chapter 9 for more details.)
- ✓ **For video:** You'll want to use QuickTime for Windows (for QuickTime movies) or MPEGplay for .MPEG video files.
- ✓ **For PostScript viewing:** Ghostview for Windows works with a companion program called Ghostscript that allows users to view or print PostScript files from any source, including the Web. Because so many documents on the Internet have the PostScript format, we find these to be useful programs.

All in all, a good set of helper applications can make your browser even more effective at bringing the wonders of the Web to your desktop. With the right help, your browser can play back or render just about anything you'll run into!

The value of visuals

Without a doubt, graphics add impact and interest to Web pages, but that extra punch doesn't come without a price. You can easily get carried away by the appeal of pictures and overdo their use on your Web page. (Overdoing applies as much to the small images that you use as buttons and visual on-screen controls as it does to large images that help to dress up the Web page.)

Therefore, when you use graphics, remember these two important things:

- ✓ Not everybody who reads your page can see the graphics. Readers may not see the graphics because they use a character-mode browser (that can't display them) or because they switch off their graphics display (a common option on most Web browsers to conserve *bandwidth*, or their ability to move data, and improve response time).
- ✓ Graphics files — even compressed ones — can sometimes be quite large, often ten or more times greater in size than the HTML file that they're attached to. Moving graphics across the network takes time and consumes bandwidth. Also, it penalizes users with slower modems far more than those attached directly to the Internet.

Sometimes graphics are essential — for example, when you are using a diagram or illustration to explain your material. In other situations, impact is important — such as on a home page, where you'll be making a first impression. Under these circumstances, using graphics is perfectly appropriate, but be sensitive to the different capabilities and bandwidths of your readers.



Following are rules of thumb for using graphics effectively in your Web pages. (As you examine the work of others, see what happens to your attitude when you find that they violated these rules.)

- ✓ Keep your graphics small and uncomplicated whenever possible. This reduces file sizes and keeps transfer times down.
- ✓ Keep file sizes smaller by using compressed formats (like .GIF and .JPEG) when you can.
- ✓ Create a small version (called a *thumbnail*) of your graphic to include in your Web page. If you must use larger, more complex graphics, link the created thumbnail on your Web page to the full-size version of the graphic elsewhere. This linking spares casual readers the impact of downloading the larger version every time they access the page (and keeps Internet usage under better control, making you a better *netizen*!).

- ✓ Keep the number of graphic elements on a page to a minimum. Practically speaking, this means at most two or three graphic items per page, where one or two items are compact, iconlike navigation controls and another is a content-specific graphic. Here again, the idea is to limit page complexity and to speed up transfer times.

Sometimes, the temptation to violate these rules of thumb is nearly overwhelming. If you must break the rules, be sure to run your results past some disinterested third parties. (You'll learn more about testing techniques in Part V of this book.) Watch them read your pages if you can. Listen carefully to their feedback to see whether you've merely bent these rules or broken them to smithereens!

Also, remembering that not everybody who accesses the Web can see your graphics should help to keep you humble. For readers who don't have access to graphical browsers, try to think of ways to enhance their reading experience even without graphics.

Mavens of multimedia

The rules that go for graphics go *double* for other forms of multimedia. If graphics files are large when compared to HTML text files, then sound and video files are HUGE. They are time dependent; therefore, the longer they play, the bigger they are, and the longer the browser takes to download them to your computer. Although they're appealing and definitely increase the interest level for some topics, sound and video files are not germane to most topics on the Web. Use them sparingly or not at all, unless your Web site is an Internet radio show or movie theater.

With the advent of the smaller, faster display and interactivity tools for the Web (such as Java, VRML, and Shockwave), the Web will undoubtedly use more and more multimedia applications. Java *applets* work with Netscape, Mosaic, and other browsers to allow quick display of animated graphics and other special effects. VRML (Virtual Reality Modeling Language) is similar to HTML but provides 3-D viewing capabilities and more. Shockwave is the name of Macromedia's plug-in to Netscape 2.0 that allows Director movies to play inside Web pages. The use of these tools goes far beyond the scope of this book, but you can look forward to using them in your second generation Web page. If you're interested in these tools, you can get more information at the following URLs:



Java — Sun Microsystems

<http://www.javasoft.com> or <http://java.sun.com>

VRML

<http://www.vrml.org/>

Shockwave—Macromedia

<http://www.macromedia.com>



Once again, the trick is to make large files available through links instead of including them on pages that everyone tries to download. Labeling such active regions with the file's size is also a good idea so that people know what they're in for if they choose to download. (For example: Warning! This points to a 40K sound byte of a barking seal.)

Bringing It All Together with the Web

Let's get down from multimedia hyperspace and back to the cyberspace world of your future Web page. Now that you understand the basic concepts behind the Web and HTML and have met (briefly) some tools of the trade, you need to read the following paragraph carefully. It embodies the essence of your Web pages.

The three most important factors in building good Web pages are content, content, and content. (Get the idea?) If the content is well-organized, engaging, and contains links to interesting places, your Web site can be a potent tool for education and communication. If your Web site is all flash, sharing it can be an exercise in sheer frustration (and humiliation for the WebMaster . . . that's you!). Therefore, if you put your energy into providing high-quality content and link your readers to other high-quality, content-filled pages, your Web site will be a howling success. If you don't, your site will be the electronic equivalent of a ghost town!

In the next chapter, you'll discover what's involved in using — and building — documents for the Web. Just fly into our parlor for a look at what's in (and on) a Web page.

Chapter 3

What's in a Page?

In This Chapter

- Discovering layout basics
- Knowing what you want to say
- Understanding your readers
- Finding out about page flow

The trick to understanding HTML lies in knowing how to separate the two components of the HTML file: the *content* and the *controls*. (Controls are also known as *tags* or *markup*.) You can present the majority of *content* in a plain text (ASCII) file with no tagging whatsoever. When you look at an HTML source file, you see *markup* in the file that doesn't show up when your browser displays the page. (It's the characters that show up inside the HTML bracket markers (<>) that control the way characters appear on the screen; if you're puzzled, don't worry — you see plenty of examples throughout the rest of this book.) Even though markup doesn't show up on-screen, it is crucial to controlling how Web pages look and act.

The really interesting parts of HTML are the combinations of markup and content such as the commands used to title pages or control textual guideposts — headers, graphics, lists of elements, and so on. Before you can read, write, and understand HTML, you must be able to mentally separate the structure of a document from its controls.

Building good Web pages requires that you not only understand the distinction between content and controls but also use it for its best effects. In this chapter, you'll begin to appreciate the components of a Web page, and you'll find out how to bring these pieces together to create readable Web documents.

It's All in the Layout

The human eye and brain are marvelous instruments. They are capable of scanning incredible amounts of material and zeroing in on the things that are most important to the reader. As a Web page designer, your mission (should you

decide to accept it) is to aid the reader's eye in locating the page's salient features quickly and efficiently. Nothing communicates this concern — or your lack of it — more quickly than a document's layout.

The layout is the overall visual arrangement of the elements in your document. Layout isn't concerned with the placement of individual text elements on a page. Instead, layout involves the number of elements on a page, how they're arranged, and how much white space is around them.

The layout of a document — whether a Web page, a letter to your sister, or an advertisement in a magazine — is a crucial part of communicating with the reader. For materials where reader interest is mandatory, layout may not appear to be important at all (this may explain why tax forms are so boring to look at). For documents where interest must be generated, layout is almost as important as the information that the document delivers.

Think of all the boring textbooks that you've slogged (slept?) through, with pages and pages of text and the occasional graphic crammed between two half-page paragraphs. By comparison, think of a magazine or television advertisement that you've seen recently. The people who designed the advertising grabbed your interest by creating eye-catching images, by using appealing language, and by delivering arresting combinations of elements.

When it comes to building Web pages, your job can be as challenging as the one faced by advertising designers. You shouldn't assume that your material is of such great interest to the world that your Web pages can stress content at the expense of layout. You are competing with millions of Web sites on the WWW. Even if your site's content is completely unique, you can encourage your readers to visit more often and to link your site to theirs by making the layout as inviting as possible.

Because attention to layout adds to the accessibility of any document, you do your readers a service by building a good layout: At the same time that you make your page more pleasant to read, you deliver the content quickly and effectively, which is everyone's desire in the information age.

What Are You Trying to Say?

As markup languages go, HTML is fairly simple and easy to learn. Unfortunately, this fact creates a nearly overwhelming temptation to rush out and start building Web pages right away. In fact, you're probably wondering when we're going to get around to putting HTML tags on text, aren't you?

Well, whether you're an individual trying to share information with others or an organization trying to advertise its products and services, the impetus to publish online ASAP is powerful. Nevertheless, we advise you to step back and do a little analysis and design work, instead of trying to build "killer Web pages" on the fly.

Who's listening?

Knowing your audience is a critical requirement for building Web pages that people can use. If you don't know who you want visiting your Web site and why you want them to visit, you're just putting together a *vanity page* that a few Web surfers may pass through once. Don't get all bent out of shape — vanity pages comprise the majority of the personal pages on the Web. If you really have a reason for your Web site, you want to know how to emphasize it to your potential readers.

It's important to start your Web document's design based on certain assumptions about your audience. Although this is true for advertisements and for encyclopedias, the focus on form and excitement is a little more intense and urgent for ads than it is for encyclopedias. You could do worse than to create the initial interest and impact of an advertisement, but you'll want to go farther than most ads go to deliver the depth of content that your audience is most likely to want, even if it isn't entirely encyclopedic in nature.

How can you get to know your audience? Think of it as a form of hunting: Identify your target group and then start hanging around their haunts, whether in cyberspace or in the real world. Watch and listen to them. When you recognize their interests, you can target their needs and duly consider the factors that can hook them into your content. You must deliver solid, usable information to your target group so that they come back for more and spread the word about what you have to offer.

Design springs from content — and intent

Web pages built around long documents with complex ideas take more forethought and are more difficult to design. That's a big surprise . . . NOT. Short, single-concept documents are easy to make into good Web pages . . . not necessarily true either. No matter how long, short, complex, or simple your content is, you still need to follow this basic principle: Design springs from content, as form follows function. Also, remember the audience that you intend to reach; emphasize the high points that your research shows to be of interest to your potential readers.

Creating an outline before you start writing (or programming in HTML, for that matter) is usually sufficient to help you organize your information. An outline helps you to determine the order of topics and the need for graphics, sound, or other multimedia information. It also gives you a blueprint to follow as you construct the document that expands the outline.

Identifying the topics and major elements of your document highlights the relationships between those components (and possibly between the components and other information sources on the Web). Therefore, outlining your content plays a key role in establishing links and presenting your readers with visual clues on how to read and navigate the information in your document.

The intent behind a document — to inform, educate, persuade, or question — plays a major role in your design.

- ✓ If your goal is to inform, you are less inclined to include numerous eye-catching displays and more inclined to direct readers' vision to the highlights of the information that your document contains.
- ✓ If your goal is to persuade or sell, you try to hook the readers' attention with compelling visuals and riveting testimonials, and then you follow through with the important details.
- ✓ If your goal is to question something, you raise the issues to be queried early on and provide pointers to additional discussion and related information afterward.



In each case, the goal behind the document strongly conditions its execution and delivery. That's why understanding your intent is so important to building the right kind of document.

Establish your key messages

At the beginning of your document design process you need to answer the question at the head of this section — “What Are You Trying to Say?” Approach this task by outlining the key ideas or messages that you want to convey; put the most important ones first. Then follow each main idea with any relevant information that makes your case, proves your point, or otherwise substantiates what you're saying.

If you follow this exercise carefully, you find much of the content emerging gracefully from your outline. The important relationships among the various elements of your document (and other documents) more or less establish themselves as you work your way through the outlining process.

Think about superstructure and information flow

Superstructure refers to the formal mechanics of how you communicate a document's organization and navigation. It includes elements such as

- ✓ a table of contents
- ✓ a set of common controls
- ✓ an index
- ✓ a glossary of technical terms

In short, the superstructure is the wrapping that you wind around your content so that readers can find their way to the information they want and understand what they're reading or viewing. For any given document, you may not need

every element of superstructure, but for most documents — especially longer ones — some elements of superstructure are helpful, and others are absolutely essential.

The TOC (table of contents)

From a lifetime of exposure to printed materials, we've come to expect a table of contents that lays out the topics and coverage at the beginning of most documents. The beauty of hypertext is that you can build links that take readers directly from any entry in the TOC to the corresponding information in the document. Thus, the TOC becomes not just an organizational map for your document, but a convenient navigation tool.

Common controls for all screens

To promote readability and familiarity within your Web site, include common controls in each individual document. These common controls can be a set of clickable icons: to page backward or forward, to jump back to the TOC, or to return to your home page. If you include a search tool for keywords in the document, make the tool accessible from any point in any of your documents. Whatever you do, establish a common look and feel for your pages; then your readers can navigate much more easily. Consistency may be "the last refuge for the unimaginative," but it does promote familiarity and ease of use!

An index or a search engine

Helping readers locate keywords or individual topics can ensure that they get the best use from your content. Another great thing about hypertext is that an old-fashioned index may be unnecessary. Because all your content is online and accessible to the computer, you can often replace the functionality of an index with a built-in search engine for your Web pages.

Chapter 14 presents a discussion of search engines, including tools that you can use to provide index capabilities for your documents.

A glossary helps manage specialized terms and language

If you're covering a subject that's full of jargon, technical terms, or other forms of arcane gibberish (beloved by experts and feared by newcomers), try to include a glossary with your Web pages. Fortunately, HTML includes a text style specifically built for defining terms, which helps you to construct a glossary whenever you need one. You'll still have to come up with the definitions yourself — unless you can find a Web site with a glossary that you can link to your page. That's the beauty of the Web!



Grab the audience's attention . . .

If you've ever had the pleasure of viewing a movie at a THX-equipped theater, you're probably familiar with the phrase "The Audience Is Listening." It comes up at the conclusion of the THX demonstration, which usually happens immediately before the feature presentation.

The THX demonstration consists of the simple "THX" graphic that fills the entire screen. A loud, sustained orchestral chord, overlaid with a powerful pipe-organ note, accompanies the graphic. The musical effect starts quietly, builds to a peak over 20 seconds, and then fades away on a low pedal-tone that you swear moves the entire theater. The sensation is similar to standing at the end of the runway while a jet fighter comes toward you and takes off over your head at 200+ mph. It definitely (and almost deafeningly) gets your attention. Having raised the audience nearly out of its seats, the demonstration concludes with the words "The Audience Is Listening."



Even though we don't recommend this audio effect for most Web pages, we do encourage you to grab your readers' attention when they first glimpse your page. Nothing does this as effectively as a tasteful image coupled with a brief, compelling introduction to your page. Include information that tells why your page is important, what it contains, and how to get around. Get your readers interested, get them oriented, and then they'll be hooked!

They're after the goods . . . don't get in the way!

After you have your readers' attention, help them to get to the real content of your Web site easily. Your superstructure should be visible, but don't let it get in the way. If you include pointers to direct your readers to more-detailed content, make the pointers obvious and easy to distinguish from the rest of the page. An overly complicated design, layout, or information flow can prevent easy access to the information.



This advice translates into some important rules of thumb, particularly for introductory materials. Keep your welcoming (home) page simple and elegant. Use short, direct sentences. Keep your focus on the topic(s) at hand. Use the superstructure to emphasize your content. For a complex Web site, you can include a link to an "About this site" page for those who want to understand the site's structure and function.

What should they remember?

A well-known educational principle reports that people exposed to new materials generally remember ten percent of the content — at best. When designing your Web documents, keep asking yourself, “What ten percent do I want my readers to remember?” This question helps you focus on the really important ideas so that you can direct the audience to them and reinforce them throughout your Web pages.

Also true, if somewhat sad, is that most readers remember a limited volume of material as well. Remembering ten percent of the concepts in a document doesn't translate into remembering ten percent of its overall content: Would you remember ten out of a hundred pages in a document that you'd never seen before? This sometimes means presenting less information than you may otherwise be inclined to convey, in the interest of concentrating on what's really important. Therefore, save yourself some time and trouble, and concentrate on the important stuff right from the outset!

Don't be too ambitious in your document's coverage: Strongly related concepts linger in memory far better than loosely linked or unrelated ones. As with so much else in life, sticking to your focus is a key ingredient for successful communication.

Meet the Elements of Page Design

Okay, you've been exposed to some important design concepts for building Web pages. At this point, you should be ready to meet the elements that make up an HTML document. These elements may sound familiar because many are integral parts of any well-written document. Others may be less familiar, perhaps because of the terminology or because the concepts — like hypertext links — don't correspond to normal printed materials.

Nevertheless, the following are the building blocks that make up Web pages. After we give you a tour of these basic elements, we conclude with a discussion of information flows and design elements for Web documents.

Tagging text

Including tags along with text is what separates HTML from any ordinary ASCII file. In HTML, tags are enclosed within angle brackets: for example, a document head is indicated with a `<HEAD>` tag. Most HTML tags travel in pairs, so that the `<HEAD>` tag marks the beginning of a document head and a corresponding `</HEAD>` marks its end.

Some tags include particular values, called *attributes*, that help describe a pointer or a reference to an external data element. Other attributes label information to be communicated back to a Web server; still others add to the physical description of a display object (for example, the alignment or dimensions of a graphical element).

Attributes provide the source and destination information for links within and across documents. These link attributes (usually called *link anchors* or *anchors*, for short) describe the relationship between two named locations in the Web, whether the locations are in the same document or are in different documents.



A linked location is indicated by a document reference (for access to other documents), a location reference (for a point inside the same document), or a combination of the two (for a point inside another document). For more information on such links, please read about `<BASE>`, `<A>` (anchor), and `<LINK>` tags in Chapter 5.

Many HTML tags require that certain attributes be specified, others can take on optional attributes and values, and still others never acquire attributes. You'll find out how to tell the difference as you discover the elements of document design presented in this book.

Titles and labels

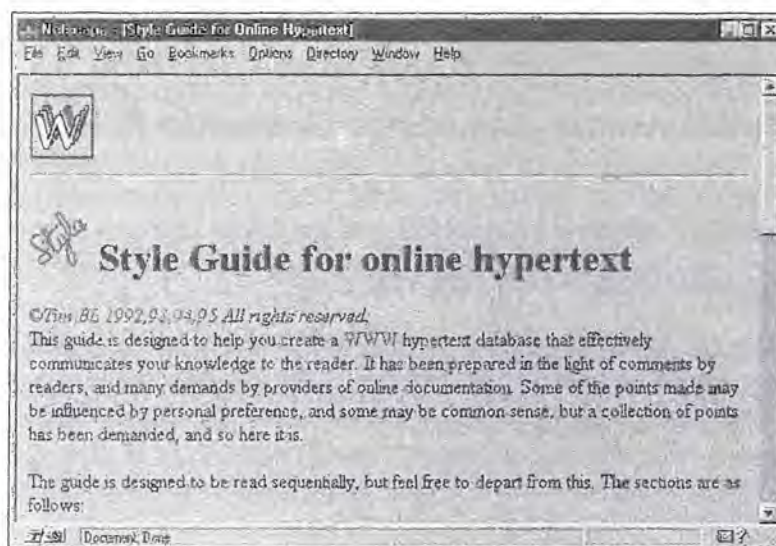
Every HTML document should have a title that serves to identify it to the readers. Also, titles have three other important aspects:

- ✓ You can save titles in hotlists, which makes them a navigation tool for readers who use hotlists.
- ✓ Titles allow Web walkers to decipher what HTML documents are about, which makes the corresponding entries that these walkers create search databases more accurately and usefully.
- ✓ Titles help you manage your documents, especially when they're complex and voluminous.

When you display a Web document, the title typically shows up in the window's title bar. Figure 3-1 shows the viewing window for an HTML Style Guide with the title, "Style Guide for online hypertext," prominently displayed.

Labels aren't required, but they're a good document-organizing tool. Labels help identify sections or topic areas in a document and provide better navigation for readers, especially when you use them for links. As you'll see in Parts III and IV of this book, if you use the `NAME` attribute inside an anchor, you can direct an HTML link to a named section in a document as well as to the head of that document. Anchors also signal to other browsers, "Hey, if you want to point to me, reference me by my anchor's `NAME` attribute."

Figure 3-1:
The document's title shows up on the window's title bar.



```
<A NAME="Mexican Dove">Linda Paloma</A>
```

If some author, or even the original author, wants to link to the Linda Paloma area of the current document, all they need to do is create an HTML link definition by using the anchor's NAME (in this case, "Mexican Dove").

Text and hypertext links

Anyone writing HTML has only one kind of link available — a unidirectional association between a source and a target. But this one kind of link has four purposes in HTML files:

- ✓ **Intradocument linking** provides a way to move from one location to another location inside the same document.
- ✓ **Interdocument linking** provides a way to move from one document to another document.
- ✓ **Linking to an *agent* program** that acts on behalf of the Web server provides a way for a document to handle a query or provide a service (such as information gathering).
- ✓ **Linking to a nontext object** provides a way to access a graphic, sound, video, or other multimedia file.

Interestingly enough, the links from inside HTML files to other types of data — like sound, graphics, motion video, and so on — are what enable the hypertext aspects of HTML. Along with these external links, HTML's inter- and intradocument links create the connections that compose the Web. Remember, no matter how they're used, links contribute to the Web's appealing look and feel.

Using graphics effectively

First impressions are critically important; therefore, you want to use your most potent, eye-catching images on the initial page that your readers see. However, bigger images mean longer download times — especially for dial-in users. You must balance the trade-off between the positive impact of a catchy image and the negative impact of *wait time*. (And you must realize that some readers have no graphics capabilities at all!)



What does all this mean for effective use of graphics?

- ✓ Use graphics only when they add to the impact, intelligibility, or value of a page. A picture can be worth a thousand words — by providing a flowchart of a complex process, for example. But if the related words are only tangential to the content, an image may end up detracting from your document's ability to communicate with its readers.
- ✓ Keep your graphics as small as possible, both in terms of image space and file size. The two are related, but you can reduce file sizes by using .JPEG (with more data compression) to render large images from relatively small files. Be sure to check the image on a lower-resolution display and adjust the compression almost to the point where resolution begins to suffer.
- ✓ Insert a *thumbnail* version of a large graphic (if you must use it) and link it to the full-size version. Also, tell users the file size of the full version, so that they can decide how badly they want to see it (translation: how long they want to wait).
- ✓ Keep the number of graphical elements per page to a minimum. Each graphic adds to the time that is needed to build a page. If you must use numerous graphical elements on a page, keep them small in order to minimize transfer time. In other words, multiple icon-sized elements on a page are fine; more than one or two large graphical elements typically is not.

Overcoming two-dimensional thinking

Although hypertext is new and exciting, the legacy of thousands of years of linear text is difficult to overcome. In other words, even though document designers can do incredibly nifty and creative things with linking and

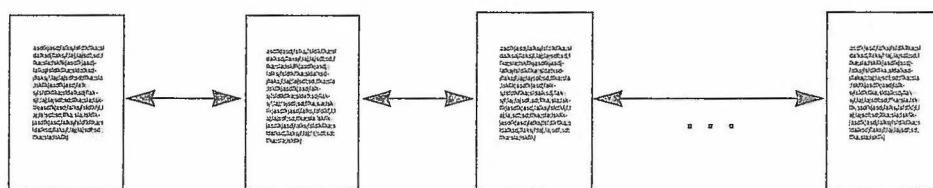
hypermedia, they have to fight the nearly overwhelming tendency to make their documents read like books. You must exploit the hypertext capabilities of HTML displays and links in appealing and useful ways, or your Web pages won't live up to their potential or the expectations of your readers.

In the sections that follow, you have a chance to examine some common organizational techniques for building Web documents; you also encounter documents that can exist only on the Web.

Stringing pages together the old-fashioned way

Some pages demand to be read in sequence: for example, those presenting a narrative that builds on previous elements. In such a case, string the pages together as shown in Figure 3-2. If you have a document of five pages or more — if you believe Tim Berners-Lee in his *Hypertext Online Style Guide* — you should chain them together sequentially anyway.

Figure 3-2:
Chain pages
together to
read them in
sequence.

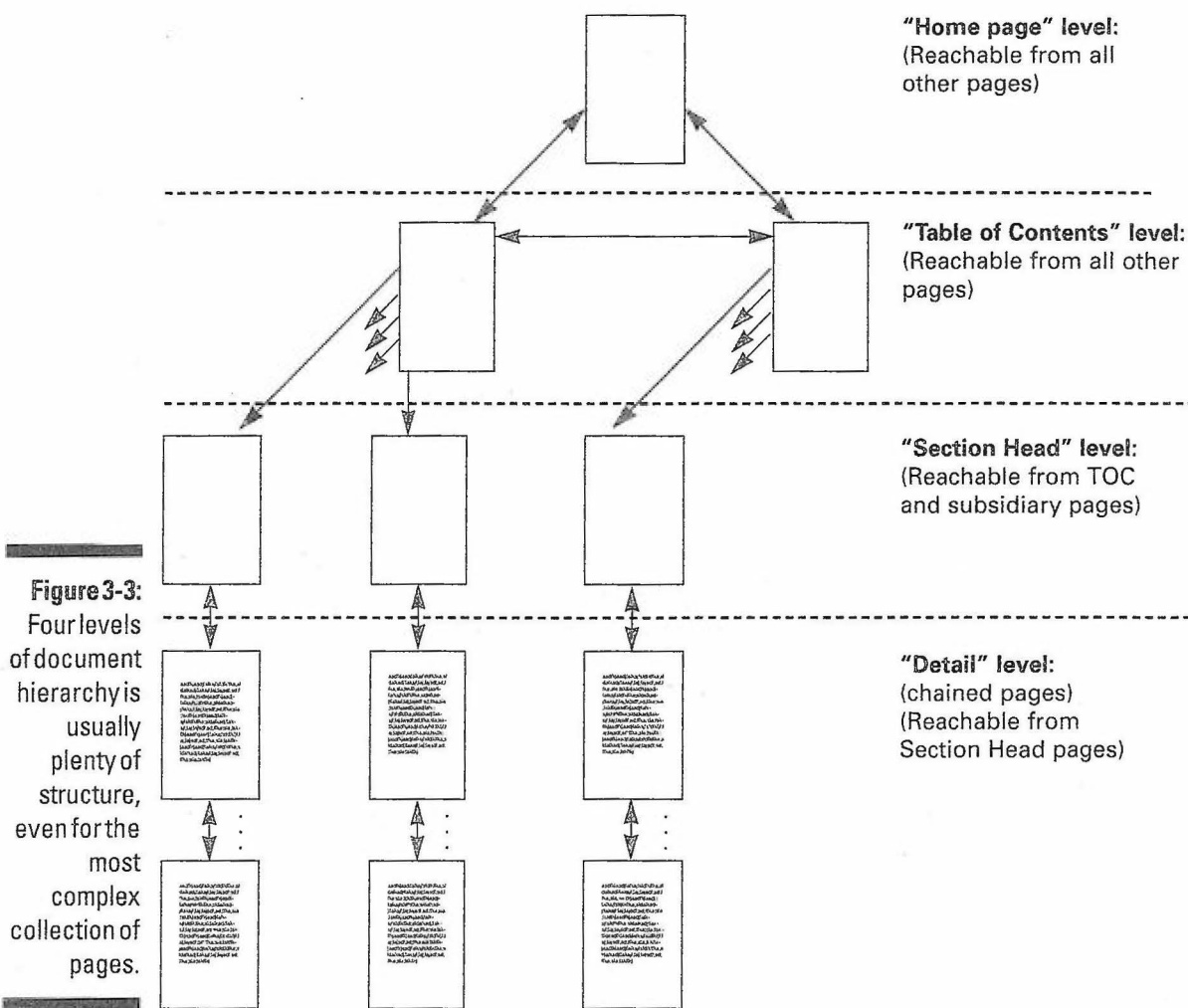


The nice thing about hypertext is that you can chain pages together forward and backward; then you can easily “turn” pages in either direction. Don’t be afraid to include other appropriate links in this basic structure (such as links to other HTML documents, a glossary, or other points inside your document).

Hierarchies are easy to model in HTML

If you commonly construct documents from an outline, a hierarchical approach to document links should immediately make sense. Most outlines start with major ideas and divisions that you refine and elaborate on to wind up with all the details of a formal document. Figure 3-3 shows a four-level hierarchical document structure that organizes the entire work.

HTML itself has no limits on the levels of hierarchies that you can build; the only limit is your (and your audience’s) ability to handle complexity. For both your sakes, we suggest keeping the hierarchy from getting too big or too deep.

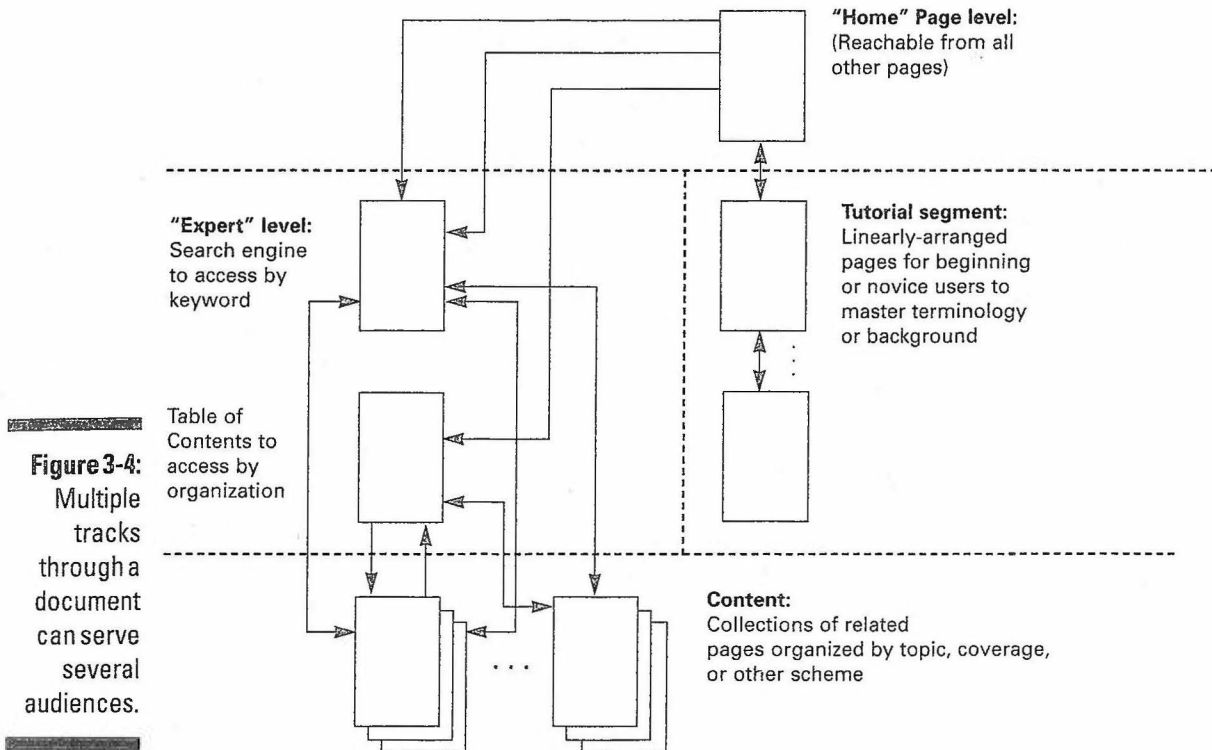


Multiple tracks for multiple audiences

It isn't unusual to build a document that includes different levels of information to meet the needs of different audiences. By using HTML, you can easily interlink basic introductory documents (like a tutorial or technical overview) with more pointed, in-depth reference materials. That way, you can design a home page that points beginners to a tutorial and leads them through an overview before assaulting them with the down-and-dirty details of your "real" content.

This kind of organization, depicted in Figure 3-4, lets you notify experienced readers how to bypass the introductory materials and access your in-depth content directly. Such an approach lets you design for multiple audiences without doing a lot of extra work.

The organization in Figure 3-4 differs somewhat from that depicted in Figures 3-2 and 3-3. It emphasizes the links between related documents more than the flow of pages within those documents. In fact, the kind of document pictured in Figure 3-4 would probably combine elements from both a linear and a hierarchical structure in its actual page flows. The tutorial section would be read from front to back (or at least a chapter at a time), but the reference section would be consulted by topic (and only rarely read all the way through).



A bona fide Web wonder: the "hotlist" or "jump page"

Some of the best resources we've located on the Web consist of nothing more than a list of annotated references to other documents. The lists usually relate to one or more specific topics. You can see this kind of document structure in Figure 3-5, which shows a single page pointing off to multiple pages in various locations. In this example, the picture fails to do complete justice to the concept. For a better illustration, you need to look at a real Web page. Consequently, we advise you to check out the URL listed below — it's quite convincing as an example of what a good hotlist can do!

There are more good hotlists on the Web than you can shake a stick at. Of course, we suggest that you start by looking at the *HTML For Dummies Hotlist* Page.



<http://www.outer.net/html4dum/hotlists.htm#top>

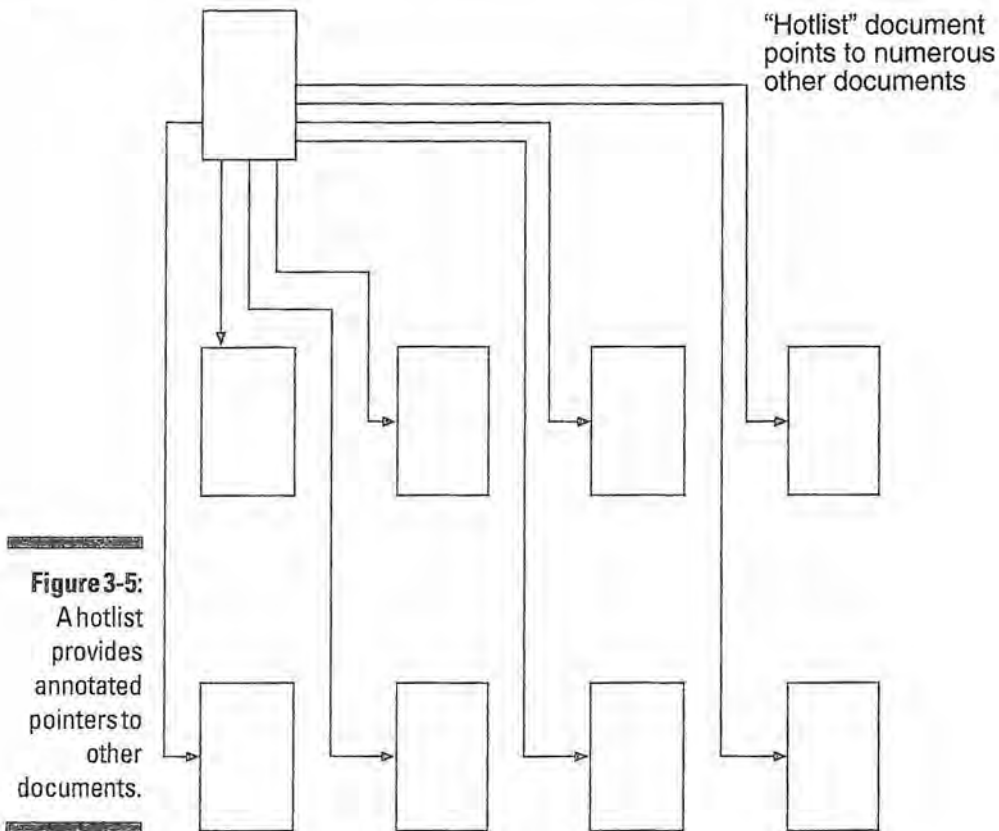


Figure 3-5:
A hotlist
provides
annotated
pointers to
other
documents.

Extending the Web, a piece at a time

Another kind of Web page solicits input from readers who, by their input, help to create an open-ended document. Readers contribute comments, additional text, and hypermedia or add to an ongoing narrative. The structure of such a document is hard to predict and, therefore, hard to depict. Suffice it to say that this kind of Web document can grow like a colony of coral, more by accretion than by prior organization or design.

For an example of this kind of living, ongoing document, consult the following URL.



<http://bug.village.virginia.edu/>

WAXweb is a hypermedia implementation of a feature-length, independent film, *WAX or The Discovery of Television Among the Bees* (David Blair, 85:00, 1991). WAXweb is a large hypermedia database available over the Internet. It features an authoring interface that lets users collaborate in building onto the story. WAXweb includes thousands of individual elements, ranging from text to music, motion videos, and video transcriptions of motion picture clips. For users with VRML-capable browsers, WAXweb also offers a pretty nifty VRML implementation, as well.

The only limitations on how you structure documents are those imposed by your need to communicate effectively with your audience. After you realize that, you can use the various page-flow and organizational techniques that we've outlined here for their best effects.

Now that you know the first building blocks of Web-page design — layout and organization — it's time to initiate you into the world of markup languages. In Chapter 4, you discover what HTML looks like and how you use it to structure a document. Here comes the good stuff!!

Chapter 4

What's a Markup Language?

In This Chapter

- Defining HTML markup conventions
- Understanding HTML's roots
- Introducing the basic HTML control characters, elements, attributes, and entities

In this chapter, you'll finally see what HTML looks like. You'll begin to appreciate what's involved in a markup language and start to understand how to use HTML to create your own Web pages. Because this chapter is an overview, you won't be able to run right out and start building pages after you read it, but you should have a pretty good idea of the pieces and parts that make HTML do its thing.

A Markup Language Is Not a Form of Graffiti

HTML represents a way to take ordinary text and turn it into hypertext, just by adding special elements that instruct Web browsers how to display its contents. These special elements are called markup tags.

The purpose of a markup language is to give either machines or humans clues about the structure, content, and behavior of a document. Markup comes in two types: descriptive and procedural. HTML is a descriptive markup language. A descriptive markup language describes the structure and behavior of a document. This focus allows an author to concentrate more on content and structure, and less on formatting and presentation. Here's an example:

```
<H1>Cooking for One</H1> - display level one header  
<OL> - begin ordered (numbered) list  
<LI>Take Lean Cuisine out of fridge.
```

```
<LI>Place in microwave.  
<LI>Set timer for 5 min.  
</OL> – end ordered list
```

The tags `H1`, `OL`, and `LI` describe an object and its components. It's up to a browser to render these properly on your screen. In fact, from browser to browser, each of these objects could be rendered differently. But the important thing is that each browser has a meaningful way to display a first-level header (`H1`) and a numbered list (`OL` and `LI`). From a portability standpoint, this is a good thing.

The other kind of markup language, procedural, describes formatting rather than structure. An example is *troff*, the dinosaur UNIX markup language, which looks like this:

```
.center .12 .Helvetica .bold Table of Contents
```

These *troff* tags tell the output device to render an object as “centered, 12 point Helvetica bold”:

Table of Contents

As you can imagine, this type of markup is not very portable and is difficult to maintain.

HTML tags not only govern how a browser displays the contents of an HTML document, but also control how the browser can draw in graphics, video, sound, and so on to create a multimedia experience.

In the same vein, still other HTML tags instruct browsers how to handle and display hypertext links, either within the same HTML file or to other documents or Web-accessible services. The key to building attractive, readable Web pages is knowing how to use HTML markup to highlight and organize your content.

A syntax is not a levy on cigarettes!

When it comes to any kind of formal, computer-readable language, you invariably find a set of rules governing its terms and their order of placement. This set of rules is called the *syntax* of the language. In HTML the syntax describes how a Web browser can recognize and interpret the instructions contained in the markup tags.

What makes HTML particularly interesting is that it's all pure text — in fact, HTML can work completely within the confines of the ASCII 7-bit character set (ISO 646), which contains only 128 distinct viewable characters. Nevertheless, HTML can handle display of so-called higher-order ASCII characters, which normally require eight bits to represent directly and are sometimes called the 8-bit ASCII or Latin-1 character set (ISO 8859/1). Quite a bit of work is under way to make HTML able to handle the Unicode character set, which supports up to 65,535 different characters (of which over 34,000 are defined). The Unicode set includes non-Roman alphabets such as Chinese (Han) ideograms, Hebrew characters, and lots of other interesting stuff.

This diversity lets HTML display things like accents, umlauts, and other diacritical marks often associated with non-English languages (or loan words, like *résumé*, from those languages), by including instructions on what characters to represent as a part of the markup. In other words, HTML can provide the related instructions to a browser even if you can't always see things in the same format when you're writing the HTML. For example, `<OL>` and `<LI>` don't look like a numbered list inside your text file, but they do when the browser interprets them.

Elements of HTML syntax

The special control characters that separate HTML markup from ordinary text are the left and right angle brackets: `<` (left bracket) and `>` (right bracket). These characters indicate that the browser should pay special attention to what they enclose. And here's your big chance to learn another buzzword: *parser*. Inside the browser software, a parser does the work of reading and constructing the display information. A parser reads information in the HTML file and decides which elements are markup and which ones aren't, permitting the browser to take appropriate action.



In HTML, left and right angle brackets can enclose all kinds of special instructions called *tags*. The next six chapters of this book are devoted to introducing, explaining, and demonstrating HTML tags.

In the meantime, here's a general introduction to the way HTML tags look and behave: A tag takes a generic form that looks something like this:

```
<TAG-NAME (ATTRIBUTE="VALUE") . . . >Text(</TAG-NAME>)
```

(By the way, text within the tags is case insensitive, but for readability, we use all caps to make HTML tags stand apart from other text.)

Let's look at the pieces of this generic form:

- ✓ **<TAG-NAME>:** All HTML tags have names. For example, `H1` is a level-one header; `OL` is an ordered list. Tags are surrounded by angle brackets that mark their contents for special attention from the parser software.
- ✓ **{ATTRIBUTE(="VALUE")...}:** Some HTML tags require or permit certain named attributes to be associated with them. Here, the notation using curly braces — for example, {ATTRIBUTE} — indicates that attributes may be present for some tags, but not for others. In the same vein, some attributes require that values be associated to them, as when supplying the name of an HTML document for a hypertext link; other attributes may not require associated values at all, which is why (= "VALUE") is also in curly braces.

For example, in an `<IMG>` tag used to point to a graphics file, a required attribute is `SRC` (source), to provide a pointer to the file where the graphic resides, as in `<IMG SRC="../../gifs/redball.gif">`. This syntax proves yet again how important it is to stay close to your source! Finally, the ellipsis (...) indicates that some HTML tags may even include multiple attributes, each of which may take a value or not, depending on the circumstances.

- ✓ **Text:** This is the content that's modified by a tag. For example, if the tag were a document title, the HTML string

```
<TITLE>HTML For Dummies Home Page</TITLE>
```

would display the words "HTML For Dummies Home Page" in the title bar at the top of a graphical browser's window. Enclosing this text within `<TITLE>` and `</TITLE>` tags marks it as the title for the document.

- ✓ **{</TAG-NAME>}:** The closing tag name is denoted by the left angle bracket (`<`), followed by a forward slash (`/`), the tag name, and finally a closing right angle bracket (`>`). The curly braces indicate that this element does not always occur. However, over 70 percent of the HTML tags require a closing tag as well as an opening tag, so the omission of a closing tag should be considered the exception, rather than the rule. Don't worry; most, if not all, browsers will simply ignore closing tags if they're not necessary.

The majority of HTML tags don't require the assignment of attributes, so don't be too overwhelmed by their full-blown formal syntax. By and large, most tags resemble the `<TAG-NAME>Text</TAG-NAME>` layout like `<TITLE>HTML For Dummies Home Page</TITLE>`.

The ampersand (&) is another special HTML control character. It's used to denote a special character for HTML content that might not belong to the 7-bit ASCII character set (like an accent grave or an umlaut), or that might otherwise be interpreted as a markup character (like a left or right angle bracket). Such tagged items are called character entities, and can be expressed in a number of ways. For example, the string `è` produces a lowercase E with a grave accent mark (è), while the string `<` produces a left angle bracket (`<`).



Chapter 6 supplies the complete set of HTML character entities. If you work in a language other than English, you might want to consider building macros to replace familiar higher-order ASCII characters in your HTML files. Such macros would let you automate this kind of search-and-replace maneuver as a post-processing step. Then, you could avoid keying seven or eight characters of HTML to produce one output character (this is what computer science nerds like to call an *unfavorable input-output ratio*)! Even better, check out some of the HTML authoring tools in Part VIII: Most of them will automate this kind of thing for you.

In the next section, we'll uncover the roots of HTML, namely SGML, Standard Generalized Markup Language.

Standard Generalized Markup Language (SGML)

Technically speaking, HTML is not a programming language, nor can an HTML document be called a program. Normally, a program is defined as a set of instructions and operations to be applied to external data (usually called the input).

HTML combines instructions within the data to tell a display program, called a *browser*, how to render the data that the document contains. Even though it's not a programming language per se, HTML provides plenty of structuring and layout controls to manage a document's appearance and the linkage mechanisms necessary to provide hypertext capabilities.



Warning: Entering the Acronym Zone

Nearly any writing, speech, or documentation that has to do with computers will be rife with technical jargon and gibberish, no matter what the specific subject might be. To make matters even more confusing, the cognoscenti would much rather abbreviate frequently used technical terms like Hypertext Markup Language as HTML or Standard Generalized Markup Language as SGML, instead of having to say all the words for the concept each time it's mentioned.

The use of strings of first letters for technical terms — such as ROM for read-only memory, RAM for random-access memory, CPU for central

processing unit, and so on — happens across the board in the computer world, so you might as well get used to it! Because this kind of shorthand is called an acronym (which literally means *from the point or head of the name* in Greek), we've got to warn you that you've entered a world where acronyms are commonplace. To make your lives a little easier, dear readers, we've tried to include all of the acronyms that we use in this book in the Glossary at the end. That way, if you forget what a particular string of letters means, you'll at least be able to look it up!

Actually, HTML is defined by a particular type of document — called a *Document Type Definition*, or *DTD* — within the context of SGML. Thus, any HTML document is also an SGML document that represents a specific subset of SGML capability.

Generalized Markup Covers Many Sins

SGML originated with work begun at IBM in the 1960s to overcome the problems inherent in moving documents across multiple hardware platforms and operating systems. IBM's efforts were called GML, for General Markup Language. GML was originally targeted for local use at IBM, rather than as a generic way of representing documents. This development was the first publish-once, multiplatform strategy for document preparation — a concept that's become extremely popular today.

GML's originators — Charles Goldfarb, Ed Mosher, and Ray Lorie (the original "GML") — realized by the 1970s that a more general version of markup would make documents portable from one system to another. Their work led ultimately to the definition and birth of SGML in the 1980s, which is today covered by the ISO 8879 standard.

SGML is a powerful and complex tool for representing documents of all kinds. It offers the ability to create document specifications that can then be used to define and build individual documents that conform to those specifications.

Some commercial and government institutions, like the Department of Defense (DoD), have adopted SGML. The DoD mandates that contractors and subcontractors now submit all documentation to the government in SGML. The DoD even mandates the DTDs to which the contractors' documents must conform.

Several quotes from the SGML standard, ISO 8879, should help to illustrate its aims, while underscoring a strong relationship to HTML. First, there's the notion of the markup process:

Text processing and word processing systems typically require additional information to be interspersed among the natural text of the document being processed. This added information, called "markup," serves two purposes:

- a) Separating the logical elements of the document; and
- b) Specifying the processing functions to be performed on those elements.

Charles F. Goldfarb, *The SGML Handbook*, (Clarendon Press, Oxford, 1990, 5).

In stature and in scope, SGML far outstrips HTML; it is used to define complex document types like those used for military standard (Milspec) documents or aircraft maintenance manuals, whose specifications alone can run into thousands of pages.

The notion of *generalized markup* is what makes SGML's document definition system so all-encompassing and powerful. Goldfarb has this to say on that subject:

...“generalized markup” [...] does not restrict documents to a single application, formatting style, or processing system. Generalized markup is based on two novel postulates:

- a) Markup should describe a document's structure and other attributes rather than specify processing to be performed on it, as descriptive markup need be done only once and will suffice for all future processing.
- b) Markup should be rigorous so that the techniques available for processing rigorously-defined objects like programs and data bases can be used for processing documents as well. (Goldfarb 7-8)

Even though SGML, like HTML, is oriented toward producing documents, the goal is to make those documents behave more like programs, that is, to behave predictably in a computer-oriented world.

Building better pieces and parts

The whole idea behind SGML is to create a formal method for describing the sections, headings, styles, and other components that make up a document so that references to individual items or entries in a document can be described by such definitions. This description lets a document be rendered in a consistent way, no matter what platform displays it. In simplistic terms, SGML is a general-purpose tool for describing documents of just about any conceivable kind.

In its most generic form, an SGML document comes in three parts:

- ✓ A description of the legal character set and the characters used to distinguish plain text from markup tags
- ✓ A declaration of the document type, including a listing of the legal markup tags it may contain
- ✓ The document itself, which includes actual references to markup tags, mixed with the content for the document

Where HTML fits under the SGML umbrella

All three document parts (see the preceding bulleted list) do not have to be included in the same physical file. In fact, HTML works from the same set of definitions for the first two items so that only the contents and tags that make up an HTML document need to be included with the document itself, as described in the third item.

For HTML, the ISO Latin-1 character set defines character entities for higher-order ASCII characters, along with the angle brackets and forward slash used to indicate markup tags. The declaration of document type comes from a standard DTD for HTML.

Therefore, HTML can be conveyed by pure ASCII text files that conform to the definitions and requirements covered in the first two bulleted items to create instances of documents, the third bulleted item. In other words, all that HTML consists of is content text, character entities, and markup tags.

Welcome to HTML

Despite its more limited nature, HTML shares several important characteristics with its SGML *parent* — namely

- ✓ A character-based method for describing and expressing content
- ✓ A desire to deliver that content equally to multiple platforms
- ✓ A method for linking document components (and documents) together to compose compound documents

Although HTML may be less generalized than SGML, it still leaves plenty of room for unique and powerful expressions, as any quick perusal of Web pages illustrates. Though it may be less than completely generalized, HTML's tags and entities can still do justice to a broad range of content!

Delivering content to a variety of platforms

HTML's tremendous power and appeal come from its capacity to service character-mode and graphical browsers with identical content. The look and feel of any document's content remains the same, subject only to the display limitations of whatever browser you're using.

When you add the capacity to group multiple, related sources of information together — text, graphics, sound, and video — and the capacity to link documents together, the result is *hypertext*. HTML's combination of a simple concise form, powerful controls, and hypertext linkages helps to explain the overwhelming popularity of the Web as an information retrieval and investigation tool.

The four-plus faces of HTML

In a manner of speaking, all there is to HTML besides content is a collection of character entities and markup tags. Some purists insist on remaining within the confines of the SGML DTD for HTML, but the number and kinds of tags and entities used in various Web environments (especially in some of the more advanced browsers) continue to expand with each passing day and each new version of browser software.

Although some browsers may recognize tags and entities unknown to other browsers, HTML includes the convention that all unrecognized markup is ignored. You may lose some of the finer formatting controls with some browsers (like the `<BLINK>` tag, currently recognized only by Netscape), but the content should still be quite accessible.

Today, HTML has several standardization levels, numbered zero through three:

- 0 The original text-only markup language developed for prototype browsers at CERN prior to HTML's release to the general public. You shouldn't see any tools that remain at this level, except for historical curiosities.
- 1 The initial public implementation of HTML markup, which included the ability to reference graphical elements in addition to text controls. Many browsers — for example, Lynx and cello — still operate at level 1 HTML.
- 2 The current implementation of HTML markup, which includes all markup elements for level 1 plus tags for interactive forms. Most graphical browsers — like Mosaic, Netscape, and WinWeb or MacWeb — support level 2 HTML.
- 3 The name of another HTML level that is no longer under development as such. Throughout the rest of the book, we refer to this level generically as *beyond 2.0*, when we're not referring to some specific HTML standards document.

The group that manages the HTML standards, known as the World Wide Web Consortium or W3C, realized that it took too long for the agglomeration of standards known as HTML 2.0 to make it through the approval process. Thus, it decided to treat the components of what was called HTML 3.0 individually, to get them through the approval process faster. Today, these elements include things like mathematics notation, tables, frames, and more. You'll often hear them called HTML 3.0, but we think it's important for you to recognize that this nomenclature is no longer exact. (In Chapter 12, we cover this information in detail.)

To find out what's current for HTML, you can always go trolling on the Web itself. You should be able to find the current specification, in the form of the HTML DTD, along with online documentation on HTML markup and usage, as well as current information on SGML.

HTML Elements!

Well-structured HTML documents come in three parts, consisting of

- ✓ A **head** that identifies the document as HTML and establishes its title
- ✓ A **body** that contains the content for a Web page. This part holds all displayed text on a page (except for the title), as well as all links to graphics, multimedia information, locations inside the same HTML file, or other Web documents.
- ✓ A **footer** that labels the page by identifying its author, date of creation, and version number (if applicable)

In reality, HTML is very forgiving, so you can sometimes get away with skipping some of these elements. As a matter of good style and practice, however, we strongly recommend that any pages you design begin with the information necessary for all three elements.

Go to the head of the document

An entire HTML document should be bracketed by the identification tags `<HTML>` to open the document and `</HTML>` to close it. These tags identify the DTD for the document to an SGML-sensitive program, to allow it to interpret the document's contents properly. You can get away with omitting this tag for most browsers, but with the increasing convergence of HTML and SGML, doing so may limit the shelf life of your Web pages.

An optional line may sometimes precede your document head. It is called a *document type prolog* and describes, in SGML, that the following HTML document complies to the indicated level of the HTML DTD. Here is an example:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
```

By deciphering this line, you can tell that the HTML document conforms to the HTML 2.0 DTD distributed by the Internet Engineering Task Force (IETF). You can also tell that the DTD is `PUBLIC` and is not system dependent. Finally, you can tell that the HTML tag set is defined by the English language (the `EN` in the `DOCTYPE` statement above). Local SGML parsers can also use this `DOCTYPE` statement to validate the HTML document; that is, the parsers check the document's syntax for conformance and correctness. This prolog will become increasingly important in the future.



A document title is flagged by the HTML tags `<TITLE>`, to open the heading section, and `</TITLE>` to close it. Within this section, you commonly find the actual title and possibly some other document heading tags, which we cover in

more detail in Chapters 5 and 7. The important thing about an HTML document heading is that it should identify itself in an informative and catchy way.

HTML also includes a pair of tags, `<HEAD>` and `</HEAD>` to physically identify the head of a document. As with the `<HTML>` and `</HTML>` tags, many browsers happily let you skip this tag, but for readability and structure, we recommend including it anyway.

The bulk's in the document body

The real content for any HTML document occurs in the body section, which is enclosed within `<BODY>` and `</BODY>` tags. The body is where you describe your document's layout and structure by using a variety of tags for text headings, embedded graphics, text paragraphs, lists, and other elements. Not surprisingly, the majority of HTML tags occur within a document's body, simply because that's where all the beef is!



Chapter 5 covers all HTML tags in brief form, and Parts III and IV of this book provide lots of examples. Remember, this is just the overview.

The good stuff's in the graphics and links

You also find all hypertext content within the document's body section. This content can take the form of references to graphics or other files, as indicated by the appropriate use of text tags like `<IMG>` for in-line graphics. Or, you can have links to other points within the same document or to outside documents by using the *anchor* tags (`<A>`, `</A>`) with the appropriate attributes. In fact, because anchors point to generalized URLs (not just other HTML documents), you can also use them to invoke services like FTP to transfer multimedia files from within your Web pages.



Chapters 8, 11, 12, and 14 offer lots of details about using graphics within HTML documents, including good sources for material, appropriate usage, and other sorts of graphical information goodies. By the time you're through with this book, you'll be slinging HTML graphics around with the best WebMasters!

A footer may be optional, but it's a good idea

Technically speaking, HTML doesn't include a separate tag to denote a page footer — that is, there's no `<FOOT>` and `</FOOT>` pair to label the information that would typically appear in a footer. Nevertheless, we strongly recommend that you include a footer on every Web page that you create, just for the record.

“What’s in a footer?” you ask. Well, it should contain information to describe the page and its author(s). A good footer also helps to identify a document’s vintage and contents and lets interested readers contact the owners of the page if they spot errors or want to provide feedback.

For your own sake, it’s also smart to include date and version information in each of your HTML files. Doing this enables you to recognize what version you’re dealing with whenever you look at a page, plus it provides a great reminder of how stale your pages are getting. As any real Web-head knows, the older the information, the more likely it is to be out-of-date!



Chapter 8 provides lots of details about page footers and includes a particularly fine example of what a good footer should look like. Be sure not to miss it.

Ladies and Gentlemen, Start Your Engines!

Okay, so now you’ve seen the basic elements of HTML. In the chapters that follow, we introduce you to the details, including more tags than you meet at a Red Tag sale at your local department store. As you come to know and love what HTML can do, you can apply what you’re learning to build great Web pages.

Chapter 9

Going High-Rise: Building Complex Pages

In This Chapter

- Expanding your home page into a Web
- Moving around inside your documents and local Web
- Jumping to remote sites
- Sampling and analyzing sophisticated Web pages
- 3-D and beyond

You're probably not satisfied with your nice, simple, single-screen home page. Because of all the wonderful stuff you've seen out there, you really want to make a Web of pages with all sorts of great material in them, right? That's pretty natural, and it doesn't compromise the KISS principle either.

If you recall, we suggested that you'd want to make more pages as you expanded your Web. But the more pages you add, the more difficulty your users will have in finding their way around. While you're growing your own Web, your most important job is to make your users' journey through it as enjoyable as possible. In fact, you've already discovered the necessary methods and techniques in previous chapters. Now it's time to put them to use.

This chapter covers important aspects of creating complex Webs. After this discussion, we show you examples of more advanced Web pages with comments about their elements and layout.

There's No Place Like Home

Home isn't only your home page; it's your own Web, the local constellation of planets orbiting your home page. It's your local turf in cyberspace, where Web surfers can find the information that you think is important. But even if your

site is fantastic and beautiful, users can be put off if they have trouble navigating your Web. That's why you need a clear mental picture of its fully developed organization before you start expanding things.

Organization

If you listen closely to your content, it can tell you the organizational style that it needs — or rather, demands. Hierarchical style, linear style, and randomly interlinked combinations of these two styles are the standard organizational structures used in most Webs. These structures — which we'll call *Web structure* — are illustrated in Figures 9-1, 9-2, and 9-3.

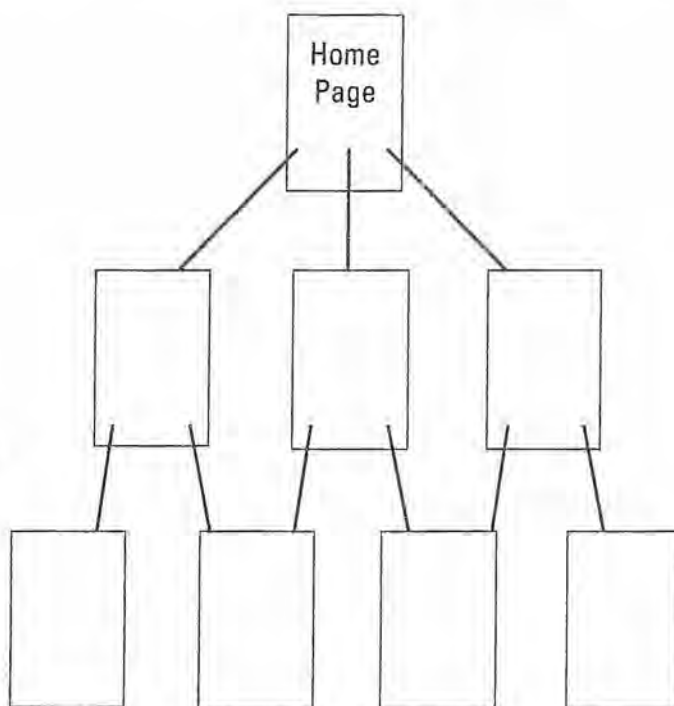


Figure 9-1:
The
hierarchical
structure
looks like a
family tree.



It's straightforward to "Web-ify" a linear document, but the converse is not true: Organizing a random collection of ideas and concepts into a linear document is very difficult, if not impossible. When designing and using links within or among Web documents, you need to be clear about the organization and interconnectedness of their content.

The hierarchical, or tree, structure is used most often in Web designs. It's logical and has a familiar look to most computer users (hints: hard disk file trees and GUI help systems). This kind of organization is easy for users to navigate, especially when you include links back to the home page on each page.

When using a hierarchy, your information should progress from the most general (trunk) level, or a table of contents on your home page, to the most detailed content in the outermost leaves. Your content dictates the divisions in the tree, but you can include interesting links between seemingly unrelated branches to better inform your users. Also, try providing multiple links to individual pages. In this way, the structure includes aspects of the index of a book, as well as its table of contents.

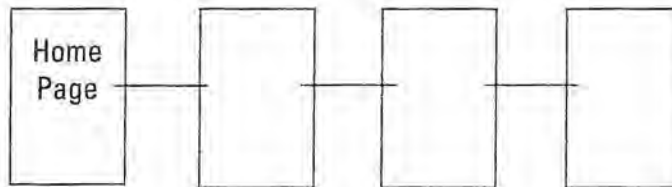
Keep in mind that readers can enter your Web space from somewhere other than your home page, so make sure that you give navigational clues to *jumped-in* users. You want them to find your home page or other relevant pages easily. It's frustrating to land on a page whose URL you obtained by e-mail from some cohort who says, "Check out this page," only to be forced to blunder around because you can't tell where home is!

Also, remember your readers experience the *lost in hyperspace* problem quite often. That's one excellent reason to provide navigational clues at the top and bottom of each Web page that you create. For an example of good style (even if we do say so ourselves), check out this URL:



<http://www.outer.net/html4dum/>

Figure 9-2:
Linear
structure.



Simple, booklike, but also *rigid and confining* are common descriptions of the linear structure. If your information presents a series of steps or follows a process from start to finish, linear structure is a fitting choice for your document organization. It keeps users on track and out of trouble. Here, you can make good use of links to "next page," "previous page," and "top or start page."

Be sure to put links to your home page (or the starting point) on each page in a linear structure. Without such links, users dropped into the middle of your Web have only the browser's controls to get out. If you trap them this way, they'll talk about you and your Web on the Net, but the talk won't be flattering!

The WWW itself is a Web structure. It's a great example of the fantastic freedom of movement and free-flowing design that are implicit in this kind of loosely linked environment. Also, Web structure can quickly demonstrate how easy it is to get lost.

Being lost on the Web isn't quite the same as being lost in a big city. Users can always use the browser's controls to go back to the previous URL or go to another URL. If you want users to get the most out of your Web, you need to provide obvious hyperlinks from each of your pages to other appropriate pages within your own Web or to other locations on the WWW.

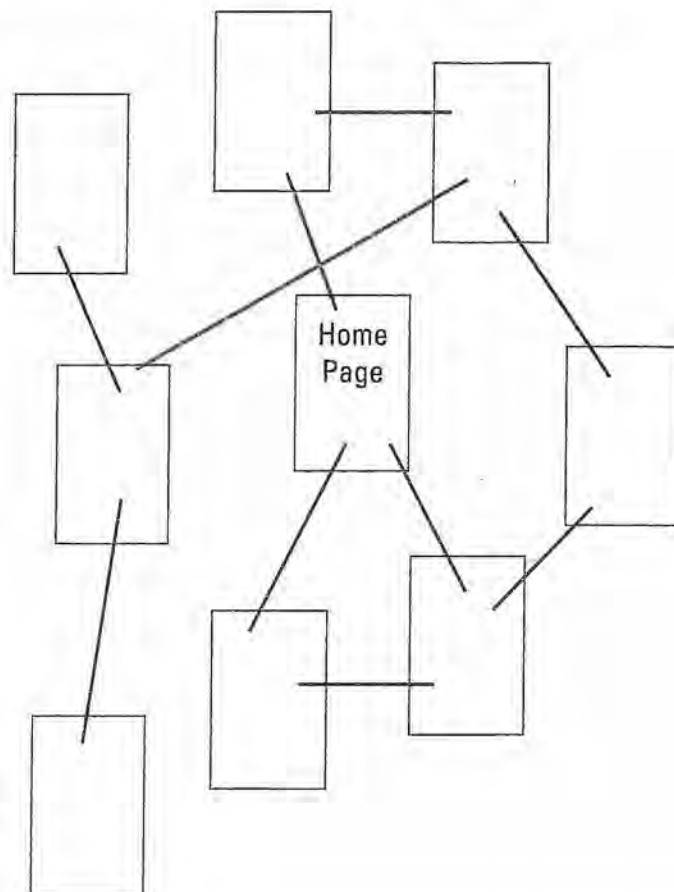


Figure 9-3:
Web
structure.



Providing structure without constraining users' freedom to explore your space is the goal of any well-designed Web structure. If your information on related subjects is extensive, put hyperlinks within the text to specific paragraphs in other pages where the user can see more detailed information about the content. Be careful how you do this, though — as we mentioned in Chapter 7, too much linkage can be just as detrimental as not enough!

Hypertext linking is the most time-consuming part of HTML document development. Do it well, and your users will love you forever. We discuss this issue in more detail later in this chapter.

When you build a complex Web structure, always, always, always put a link to your home page on each page. It's also a great idea to reproduce the URL for each page in its footer in small type. If you provide this information, users can return to any specific page in your Web by using that URL later on, even if they didn't add it to their bookmark lists.

It's story (board) time, boys and girls!

Remember doing the sketch of your home page along with Chapter 7? It's time for you to find your pencil and paper again. This time, you'll be drawing your Web structure. For a personal Web, pencil and paper will probably do nicely.

First Things First: List 'Em Out

Make a list of the major pieces of information that you want to include in your Web. These major points will probably be the links on your home page and may be similar to the following:

- ✓ Who you are
- ✓ What you do
- ✓ Personal history
- ✓ Your family
- ✓ Hobbies
- ✓ Sports
- ✓ WWW land (jump list)

Sketch the Web

In the preceding example, there are no other topics to consider, so a simple hierarchical structure looks appropriate. The simple sketch of this structure (shown in Figure 9-4) should look familiar.

Using this sketch to analyze your home page, you can see some links that aren't readily apparent from looking at the HTML document. These links exist among the "Who you are" page and the "What you do," "Personal history," and "Sports" pages as well as among the "Your family" and the "Hobbies" and "Sports" pages. All these secondary pages are therefore linked to the "Home page."