

WEB PUBLISHER'S

CONSTRUCTION KIT

Second Edition
of best-selling
**HTML Web
Publisher's
Construction
Kit**

with

HTML 3.2

Publishing Your
Own HTML Pages on
the Internet

INCLUDES

NETMANAGE CHAMELEON,
SLIPKNOT, HOTMETAL,
HTML ASSISTANT, WINHTTPD,
LVIEW PRO, WHAM, CGI SCRIPTS
FOR UNIX AND WINDOWS,
NCSA'S HTTPD FOR UNIX,
AND MORE!

COVERS THE
LATEST VERSIONS
OF NETSCAPE
AND HTML



DAVID FOX, TROY DOWNING

Publisher • **Mitchell Waite**
Associate Publisher • **Charles Drucker**

Acquisitions Manager • **Jill Pisoni**

Editorial Director • **John Crudo**
Managing Editor • **John Crudo**
Copy Editor • **Scott Calamar, LightSpeed Publishing**
Technical Editor • **Miko Matsumura**

Production Director • **Julianne Ososke**
Production Manager • **Cecile Kaufman**
Production Editor • **Mark Nigara**
Cover Design and Production • **Sestina Quarequio and Karen Johnston**
Cover Illustration • **Rafael Lopez**
Illustrations • **Kristin Peterson**
Production • **Michele Cuneo**

© 1996 by The Waite Group, Inc.®
Published by Waite Group Press™, 200 Tamal Plaza, Corte Madera, CA 94925.

Waite Group Press™ is a division of Sams Publishing.

All rights reserved. No part of this manual shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, desktop publishing, recording, or otherwise, without permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. While every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

All terms mentioned in this book that are known to be registered trademarks, trademarks, or service marks are listed below. In addition, terms suspected of being trademarks, registered trademarks, or service marks have been appropriately capitalized. Waite Group Press cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any registered trademark, trademark, or service mark.

The Waite Group is a registered trademark of The Waite Group, Inc.
Waite Group Press and The Waite Group logo are trademarks of The Waite Group, Inc.
All other product names are trademarks, registered trademarks, or service marks of their respective owners.

Printed in the United States of America
95 96 97 98 • 10 9 8 7 6 5 4 3 2 1

Fox, David, 1973-

Web publisher's construction kit with HTML 3.2 / David Fox, Troy Downing.

p. cm.

Includes index.

ISBN: 1-57169-079-4

1. HTML (Document markup language) 2. World Wide Web (Information retrieval system)

I. Downing, Troy. II. Title.

QA76.76.H94F693 1996

025.04--dc20

96-30564

CIP

DEDICATION

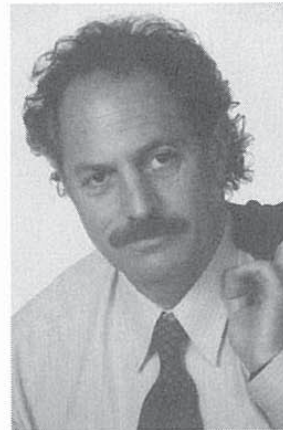
To Kelly, a friend if there ever was one.

— *David Fox*

I would like to dedicate this to my wife Laura for putting up with my late nights at work, and to my daughter Morgan—may this help pay your college tuition.

— *Troy Downing*

Message from the Publisher



WELCOME TO OUR NERVOUS SYSTEM

Some people say that the World Wide Web is a graphical extension of the information superhighway, just a network of humans and machines sending each other long lists of the equivalent of digital junk mail.

I think it is much more than that. To me, the Web is nothing less than the nervous system of the entire planet—not just a collection of computer brains connected together, but more like a billion silicon neurons entangled and recirculating electro-chemical signals of information and data, each contributing to the birth of another CPU and another Web site.

Think of each person's hard disk connected at once to every other hard disk on earth, driven by human navigators searching like Columbus for the New World. Seen this way the Web is more of a super entity, a growing, living thing, controlled by the universal human will to expand, to be more. Yet, unlike a purposeful business plan with rigid rules, the Web expands in a nonlinear, unpredictable, creative way that echoes natural evolution.

We created our Web site not just to extend the reach of our computer book products but to be part of this synaptic neural network, to experience, like a nerve in the body, the flow of ideas and then to pass those ideas up the food chain of the mind. Your mind. Even more, we wanted to pump some of our own creative juices into this rich wine of technology.

TASTE OUR DIGITAL WINE

And so we ask you to taste our wine by visiting the body of our business. Begin by understanding the metaphor we have created for our Web site—a universal learning center, situated in outer space in the form of a space station. A place where you can journey to study any topic from the convenience of your own screen. Right now we are focusing on computer topics, but the stars are the limit on the Web.

If you are interested in discussing this Web site or finding out more about the Waite Group, please send me e-mail with your comments, and I will be happy to respond. Being a programmer myself, I love to talk about technology and find out what our readers are looking for.

Sincerely,

Mitchell Waite, C.E.O. and Publisher

200 Tamal Plaza
Corte Madera, CA 94925
415-924-2575
415-924-2576 fax

Website:
<http://www.waite.com/waite>

CREATING THE HIGHEST QUALITY COMPUTER BOOKS IN THE INDUSTRY

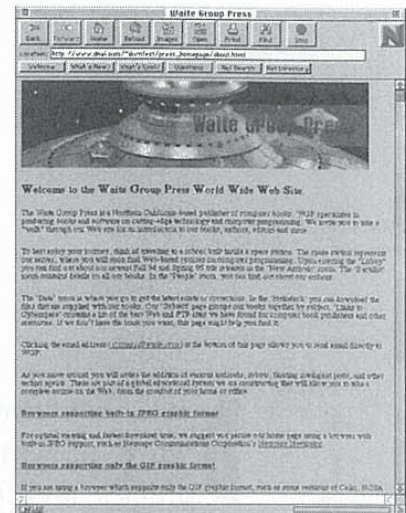
**Waite Group Press
Waite Group New Media**

Come Visit

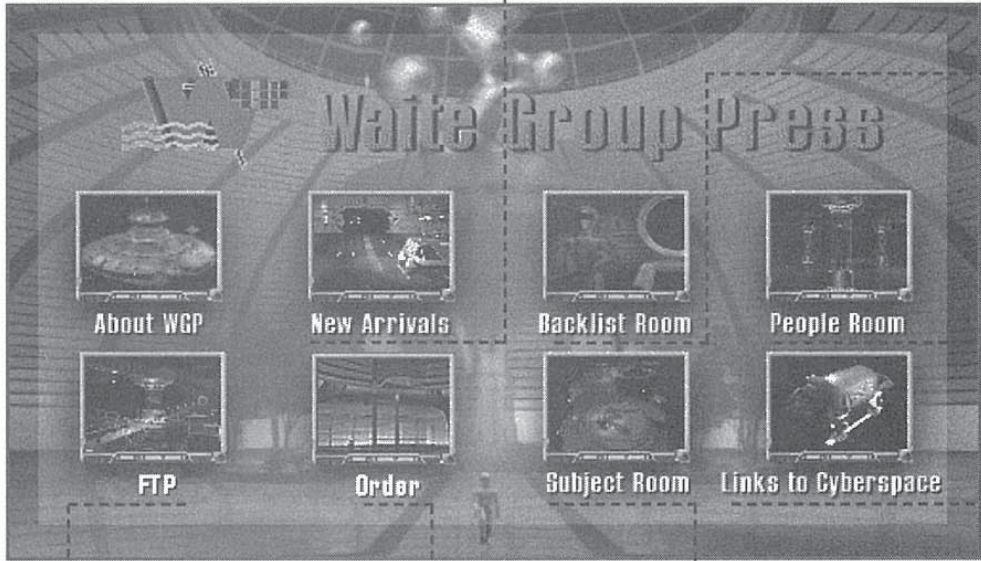
WAITE.COM

Waite Group Press World Wide Web Site

Now find all the latest information on Waite Group books at our new Web site, <http://www.waite.com/waite>. You'll find an online catalog where you can examine and order any title, review upcoming books, and send e-mail to our authors and editors. Our FTP site has all you need to update your book: the latest program listings, errata sheets, most recent versions of Fractint, POV Ray, Polyray, DMorph, and all the programs featured in our books. So download, talk to us, ask questions, on <http://www.waite.com/waite>.



The New Arrivals Room has all our new books listed by month. Just click for a description, Index, Table of Contents, and links to authors.



The Backlist Room has all our books listed alphabetically.

The People Room is where you'll interact with Waite Group employees.

Links to Cyberspace get you in touch with other computer book publishers and other interesting Web sites.

The FTP site contains all program listings, errata sheets, etc.

The Order Room is where you can order any of our books online.

The Subject Room contains typical book pages which show description, Index, Table of Contents, and links to authors.

World Wide Web:
<http://www.waite.com/waite>
Gopher: <gopher.waite.com>
FTP: <ftp.waite.com>

COME SURF OUR TURF—THE WAITE GROUP WEB



ABOUT THE AUTHORS

David Fox lives in New York City, where he writes novels, screenplays, articles, and technical stuff, depending on how broke he is. Currently he's keeping pretty darn busy programming for a major multimedia company. He's worked on several Waite Group Press titles, including *Love Bytes: The Online Dating Handbook* and *Web Publisher's Construction Kit with VRML/Live 3D*.



Troy Downing is a Research Scientist and Programmer specializing in Internet technologies at New York University's Media Research Lab. He is also adjunct faculty in the Information Technology Institute of NYU, where he teaches classes on Web Server Technologies, Java, CGI, and HTML authoring. Troy is a founder and part-owner of WebCal, LLC, an Internet software development company specializing in Web-based groupware. Troy has co-authored numerous Web-related books, including *HTML Web Publisher's Construction Kit*, *Java Primer Plus*, and *Java Virtual Machine*. Besides his involvement in Internet technologies, Troy is also an avid mountain biker, homebrewer, and is having fun being a father to his daughter, Morgan.



TABLE OF CONTENTS

INTRODUCTION.....	XXIV
PART I CONNECTING TO THE WEB	I
1 CATCHING THE INTERNET IN A WEB.....	3
2 WEB BROWSERS.....	49
3 LYNX.....	63
4 SLIPKNOT.....	89
5 NCSA MOSAIC.....	113
6 NETSCAPE ATLAS.....	137
7 A BRIEF LOOK AT OTHER BROWSERS.....	173
PART II CREATING WEB PAGES	193
8 WHAT CAN I DO?.....	195
9 THE GAME PLAN.....	275
10 THE HYPERTEXT MARKUP LANGUAGE.....	297
11 TEXT.....	313
12 GRAPHICS.....	375
13 SOUND.....	433
14 INTERACTIVITY.....	447
15 CGI SCRIPTS.....	479
16 HTML EXTRAS.....	545
17 JAVASCRIPT.....	593
18 JAVA.....	627
19 OTHER WEB RESOURCES.....	657
20 CONVERTING, TRANSLATING, OR CHEATING.....	675
21 HTML ASSISTANT.....	695
22 HoTMetaL.....	715
PART III WEAVING A WEB OF YOUR OWN	745
23 WHERE TO PLACE YOUR HTML DOCUMENTS.....	747
24 STARTING YOUR OWN WEB SITE.....	779
A SLIP SERVICE PROVIDERS.....	833
B OTHER SOFTWARE SOURCES.....	839
C WEB HELP RESOURCES.....	849
D ABOUT THE CD.....	855
INDEX.....	859

CONTENTS

PART I CONNECTING TO THE WEB	1
I CATCHING THE INTERNET IN A WEB	3
HOW THE WEB WAS WOVEN	6
CERN: The Concept	7
NCSA: The Tool	7
NETSCAPE: The Pretty Face	7
The New Fabric	8
THE WEB'S EVOLUTIONARY EDGE	8
WORLD WIDE WEB LAYOUT	9
The Server	10
The Client Browser	10
NAVIGATING WEB PAGES	10
Hypermedia Links	11
Search Indexes	13
THE HTTP PROTOCOL	14
URLS, URLS, URLS	15
The Composition of a URL	17
ACCESSING THE WEB	21
What Computer Do I Need?	21
What About the Modem?	22
What Type of Internet Account Do I Need?	22
SLIP ACCOUNTS	23
TCP/IP	24
Connecting with Windows 95 Dial-Up Networking	25
Installing Trumpet Winsock for Windows 3.1	28
Manual Login	31
Login Scripting	32
Logging Out	34
UNIX SHELL WEB BROWSERS	34
Text: Lynx or WWW	35
Graphics: SlipKnot	36
THE INTERNET ADAPTER	36
Determining Your Unix Platform	36
Getting Your License Code	37
Getting the Software	39
Installing The Internet Adapter	40
Configuring Winsock for The Internet Adapter	40
Exiting TIA	42

TABLE OF CONTENTS

HIGH-SPEED LEASED LINES	42
Permanent SLIP	42
High-Speed Lines	43
High-Speed Hardware	44
Connection Companies	44
SHOOTING DOWN TROUBLES	45
Your Modem Won't Respond	45
The "DNS Lookup Failed" Message	46
The "Unable to Resolve Host Name" Message	46
WHAT NOW?	46
2 WEB BROWSERS.	49
WHAT TYPE OF SOFTWARE IS A BROWSER?	51
WHICH BROWSER SHOULD I USE?	53
HELPFUL HANDS	54
Viewers	54
External Applications	54
The Talking MIME	55
TELNETING TO A PUBLIC BROWSER	57
WWW VIA E-MAIL	59
BROWSING WITHOUT THE INTERNET ALTOGETHER	60
Stand-Alone Browsers	60
Minimizing Winsock	60
Nullsock	60
WHAT NOW?	61
3 LYNX	63
LESSON #1: THE LYNX'S MEOW	67
LESSON #2: NAVIGATING	68
Following Lynx Links	69
Link Info	69
LOADING A SPECIFIC URL	69
Halting a Load	70
Exiting Lynx	70
LESSON #3: THE HISTORY LIST	70
LESSON #4: SCREEN-WASHING	71
LESSON #5: PRINTING A PAGE	72
Saving	72
Mailing	72
Printing	72
LESSON #6: GETTING TO THE SOURCE	73
Saving the Source	73
Returning to the Regular View	74
LESSON #7: EDITING A PAGE	74
LESSON #8: BOOKMARKS	74
Adding a Bookmark	75
Viewing Bookmarks	75
LESSON #9: THE INDEX	76

WEB PUBLISHER'S CONSTRUCTION KIT WITH HTML 3.2

LESSON #10: SEARCHING	76
Page Searching	76
Index Documents	77
LESSON #11: SAVING A MULTIMEDIA FILE	77
LESSON #12: FILLING OUT FORMS	78
The Form of a Form	78
User Validation	80
LESSON #13: USENET NEWS POSTING	80
Selecting a Group	80
Browsing Articles	81
Reading an Article	81
Posting an Article	82
LESSON #14: SENDING E-MAIL	82
LESSON #15: THE LYNX OPTIONS MENU	83
Editor	85
Bookmark File	85
Personal Mail Address	85
Searching Type	85
VI Keys	86
Emacs Keys	86
Keypad as Arrows or Numbered Links	86
User Mode	86
Local Execution Scripts or Links	87
WHAT NOW?	87

4 SLIPKNOT	89
TYING THE KNOT: INSTALLING	92
HOW SLIPKNOT SLIPS	93
SETTING UP THE TERMINAL	93
Setting Up the Host	93
Setting Up Communications	97
Setting Up the Terminal	97
LESSON #1: CONNECTING	98
Logging In	98
Logging Out	98
LESSON #2: NAVIGATING	99
Following a Link	100
Snagging a Particular URL	101
Stopping a Load	101
Loading a Local File	101
Retrieve Again!	102
LESSON #3: TURNING OFF IMAGES	102
LESSON #4: RETRIEVING MULTIMEDIA FILES	102
Viewing	103
Saving	103
FTP Files	103
LESSON #5: YOU'RE HISTORY	103
Cleaning Up	104

TABLE OF CONTENTS

Cleaning Up for Good	104
LESSON #6: THE PERMANENT SAVE	105
Local Keepers: Saving a Complete Web Page	105
Saving a Page	106
Remote Keepers: Bookmarks	106
Placing a Bookmark	106
LESSON #7: CHECKING OUT THE SOURCE	107
LESSON #8: CONFIGURING VIEWERS	107
Editing LView or WPLANY	107
Adding a New Viewer	108
LESSON #9: MAKING IT LOOK PRETTY	109
Setting Colors	109
Customizing Fonts	109
Background	110
LESSON #10: SWITCHING TO THE TERMINAL	112
LESSON #11: UPGRADING	112
WHAT NOW?	112
5 NCSA MOSIAC	114
INSTALLING MOSAIC	116
Installing WIN32S-OLE	116
Installing the Mosaic Browser	116
LESSON #1: NAVIGATING USING THE VIEW WINDOW	117
Following Hyperlinks	118
Moving to a Specific URL	119
The Right Mouse Button	119
Halting: The Secret Button	120
LESSON #2: IGNORING IMAGES	121
LESSON #3: MULTIMEDIA FILES	122
LESSON #4: HISTORY LIST	122
LESSON #5: THE HOTLIST	122
Getting Hot: Quickly Creating a Hot Item	124
Customizing Your Mosaic Menu	124
The Quicklist	126
Sharing Hotlists	126
LESSON #6: KEEPING THE PAGE	126
Save It	127
Load It to Disk	127
Print It	127
LESSON #7: SEARCHING	128
LESSON #8: THE SOURCE OF IT ALL	128
LESSON #9: FORMS	128
LESSON #10: ANNOTATING	131
LESSON #11: GOOD NEWS AND BAD NEWS	131
LESSON #12: FTP	132
LESSON #13: E-MAIL	133
Sending Mail	133
Mail to Developers	133
LESSON #14: PRESENTATION MODE	133

WEB PUBLISHER'S CONSTRUCTION KIT WITH HTML 3.2

LESSON #15: VIEWERS AND OTHER HELPERS	134
LESSON #16: OTHER MOSAIC OPTIONS	135
The Fonts You Want	135
Colors	135
Home Sweet Home	136
WHAT NOW?	136
6 NETSCAPE ATLAS	137
INSTALLING NETSCAPE	142
LESSON #1: SCAPING THE NET	142
Following Hyperlinks	144
Framed!	145
Moving to a Specific URL	145
Loading Indicators	145
Stop It!	145
The Web Page Appears	146
Safe and Secure	146
LESSON #2: GRAPHICLESS: NO AUTO LOADING	146
Loading Later	147
Hither and Dither	148
LESSON #3: RETRIEVING MULTIMEDIA FILES	148
LESSON #4: ANCIENT HISTORY	149
LESSON #5: BOOKMARKS	150
Dealing with Bookmarks	150
Adding or Deleting Bookmarks	151
Organizing Bookmarks	152
The Bookmark Files	154
LESSON #6: KEEPING A WEB PAGE	154
Saving to Disk	155
Loading	155
Printing	155
Mail Document	156
LESSON #7: SEARCHING A PAGE	156
LESSON #8: GETTING TO THE SOURCE	157
LESSON #9: FILLING IN FORMS	158
Filling 'er Out	159
User Validation	160
LESSON #10: THE NEWS	160
Setting Up a News Server	160
Getting the News	161
Subscribing to Newsgroups	161
Browsing Articles	161
Reading an Article	163
Posting an Article	164
LESSON #11: E-MAILING	165
Opening The Mail Window	166
Sending Mail	166
Receiving Mail	167
LESSON #12: GRABBING FTP FILES	168

TABLE OF CONTENTS

UPLOADING AN FTP FILE	168
LESSON #13: CONFIGURING VIEWERS	169
LESSON #14: CUSTOMIZING YOUR NETSCAPE EXPERIENCE	170
You've Got the Look	171
There's No Place Like Home	171
WHAT NOW?	172
7 A BRIEF LOOK AT OTHER BROWSERS	173
SPRY AIR MOSAIC	176
Features	176
Getting It	177
WINWEB	177
Features	178
Getting It	179
CELLO	180
Features	180
Getting It	181
SPYGLASS ENHANCED MOSAIC	181
Features	181
Getting It	181
NETCRUISER'S WEB BROWSER	182
Features	182
Getting It	182
QUARTERDECK MOSAIC 2.0	183
Features	183
Getting It	183
MICROSOFT'S INTERNET EXPLORER	183
Features	183
Getting It	184
NETMANAGE INTERNET CHAMELEON	184
INSTALLING THE CHAMELEON	185
SIGNING UP FOR AN INTERNET PROVIDER ACCOUNT	185
REGISTERING THE CHAMELEON SOFTWARE	187
CONNECTING TO THE INTERNET	188
WEB BROWSING with WebSurfer	190
OTHER INTERNET TOOLS	192
WHAT NOW?	192
PART II CREATING WEB PAGES	193
8 WHAT CAN I DO?	195
NAVIGATE: PLACES TO START	198
Best of the Web	198
Global Network Navigator	198
Einet Galaxy	198
Yahoo	200
INFORM: TERRIFIC TEXT	201
Manuals: NCSA Mosaic	201
Helpful Information: OncoLink	202

WEB PUBLISHER'S CONSTRUCTION KIT WITH HTML 3.2

A New Type of Text	202
HOME-MAKING: NO PLACE LIKE HOME PAGES	203
Corporate Pages	203
University Home Pages	205
Government Home Pages	207
Personal Home Pages	208
TEACH: EDUCATION	210
Geography Skills	210
Eric	211
Britannica Online	211
ENTERTAIN: GROOVY GRAPHICS	212
TV: Interactive TV Index	212
Movies: Buena Vista	212
Music: Underground Music Archive	212
Art: ArtSource	214
Sports: Internet Baseball Information Center	214
PUBLISH: PAPERS, BOOKS, MAGAZINES, AND MORE	215
Academic Papers	215
Multimedia Magazines	216
'Zines	217
Books	218
SHOWCASE: PUTTING YOUR TALENTS ONLINE	218
SELL: TAKING CARE OF BUSINESS	220
Stock Quotes	220
Networking over the Net	221
Job Searching	221
BUY: SHOPPING	221
The Internet Shopping Network	222
Direct Ordering	222
SPEAK FREE: POLITICS	224
PeaceNet	224
FedWorld	224
OBSERVE: THE NEWS	225
Headline News	225
Weather	226
INTERACT: FILL-IN FORMS	226
Personal Ads	226
Databases	226
Security	227
FIND: MAPS	227
Interactive Weather Map	227
Xerox Map Server	228
Map of Mars	228
TOUR: MUSEUMS	230
Le WebLouvre	230
Exploratorium	231
Museum of Paleontology	231
VISIT: VIRTUAL PLACES	232
San Diego	232

TABLE OF CONTENTS

Paris	233
EXPO Ticket Office	234
Downtown Anywhere	235
Internet Town Hall	236
BROWSE: LIBRARIES	236
SURPRISE: OTHER NEAT GIMMICKS	238
LabCam	238
The Cyrano Server	239
Lite-Brite	239
The Virtual Frog	240
Say	240
Bianca's Smut Shack	241
A Puzzler	242
THE LATEST AND GREATEST	243
WHAT NOW?	272
9 THE GAME PLAN	275
ENTER THE WEBMASTER	277
LESSON #1: WHO'S YOUR AUDIENCE?	278
Does the Audience Exist?	278
What's Already Out There?	279
Focusing In	279
LESSON #2: WHAT'S YOUR OBJECTIVE?	281
Sound and Video	282
Talking Back	282
Security	282
LESSON #3: THE HARD SELL	282
Astound!	283
Inform!	284
Entertain!	284
LESSON #4: OVERALL DESIGN	285
Point of Entry	285
The Home Page	285
Permanence	287
LESSON #5: PROBLEM PAGES	288
Art Bombardment	288
Graphic Gluttony	288
Multimedia Moderation	289
Textual Terseness	289
Java Jabbering	289
LESSON #6: GETTING THE RIGHT LOOK	290
Consistent Controls	291
The Look Should Fit the Structure	292
Cross-Browser Dressing	292
LESSON #7: LINKS TO OTHER PAGES	293
Stale Links	293
Senseless Links	293
LESSON #8: STUDYING THE STATS	294

WHAT NOW?	295
10 THE HYPERTEXT MARKUP LANGUAGE	297
WHERE TO START	300
Steal This Text	300
Convert This Text	302
HTML HISTORY	302
HTML FUNCTIONALITY	303
LIFE, LIBERTY, AND GOOD HTML	304
LESSON #1: HTML ELEMENTS	306
Specifying HTML	307
The Header	307
The Body	309
THE FUTURE OF HTML	310
WHAT NOW?	310
II TEXT	313
DESIGN FUNDAMENTALS	316
Consistency	317
Signing Your Page	317
LESSON #1: SPECIAL CHARACTERS	317
Punctuation	318
Enhanced Characters	318
Other Characters	320
LESSON #2: SECTION HEADINGS	322
Style: Drafting Good Headings	323
Aligning Headings	324
LESSON #3: PARAGRAPHS	324
LESSON #4: LINE BREAKS	326
No Break	327
Word Break	327
LESSON #5: TEXT STYLE	327
Physical Styles	328
Logical Styles	329
LESSON #6: SECTION STYLE	330
Preformatted Text	330
Blockquotes	332
Addresses	333
Styles to Watch Out For	333
LESSON #7: FONT SIZE AND COLOR	335
Relative Font Size	336
The Base Font	336
Making It Easier	336
Font Color	336
LESSON #8: CENTERING	337
LESSON #9: COMMENTS	337
LESSON #10: LISTS	338
Ordered Lists	339
Unordered Lists	341

TABLE OF CONTENTS

Customizing Lists	343
Definition Lists	343
Plain Lists	345
Menu Lists	346
Directory Lists	346
LESSON #11: HORIZONTAL LINES	346
In Thickness and in Health	347
Setting the Line's Width	347
Aligning Your Lines	347
Using a Solid Bar	348
LESSON #12: TABLES	349
Text Tables	349
Graphical Tables	349
HTML+ Tables	350
LESSON #13: HYPERTEXT	356
The Anchor Element	356
Links: HREF	358
Anchors: The Name Attribute	360
Title	363
Defining Relationships	364
Methods	366
LESSON #14: RELATIONSHIPS AND HYPERPATHS	366
Using Link	366
Nodes	367
Path	368
LESSON #15: MATHEMATICAL EQUATIONS	368
LESSON #16: DYNAMIC DOCUMENTS: PUSH OR PULL?	370
Server Push	370
Client Pull	370
LESSON #17: MARQUEES DE HAPPY	371
LESSON #18: POLISHING	373
WHAT NOW?	373
12 GRAPHICS	375
GRAPHIC FORMATS	378
GIF	378
JPEG	379
GRAPHIC VIEWERS	379
LView	379
WinGIF	381
LESSON #1: INLINED IMAGES	382
The IMG Command	383
Distant Images	383
Hyperimages	383
Crossing the Border	384
LESSON #2: ALIGNMENT	384
Absolute Alignment	385
Floating Alignment	386
Mind in the Gutter	387

WEB PUBLISHER'S CONSTRUCTION KIT WITH HTML 3.2

Clearing Space After Line Breaks	387
LESSON #3: TEXT ALTERNATIVES	388
LESSON #4: IMAGE SIZING	389
Auto-Scaling	390
Percentage Auto-Scaling	391
LESSON #5: TRANSPARENT GIFS	392
STEP 1: Isolate the Background	392
STEP 2: Selecting a Background Color	394
STEP 3: Making the Background Transparent	395
LESSON #6: CREATING SPACERS	399
LESSON #7: PERKS, GIZMOS, AND FRILLS	400
Bullets	400
Buttons / Icons	402
Bars	403
LESSON #8: MAKING GRAPHICS HUMBLE	403
Shrink Me	404
Thumbnails	406
The Flip Trick	407
LESSON #9: INTERLACED GIFS AND PROGRESSIVE JPEGS	408
Interlacing a GIF Image	408
Progressive JPEGs	409
LESSON #10: CLICKABLE IMAGEMAPS	410
STEP 1: Create the Image	411
STEP 2: Create the Map	411
MapEdit 1.1.2	413
STEP 3: Write the HTML	416
STEP 4: Set Up the Server	417
STEP 5: The Text-Based Index	418
LESSON #11: CLIENT-SIDE IMAGE MAPS	418
LESSON #12: WHAT'S YOUR BACKGROUND?	420
Background Color	421
Foreground Color	422
Getting Backgrounds	422
LESSON #13: EMBEDDING	423
THE SILVER SCREEN	423
Lights, Camera... Action?	424
QuickTime	424
MPEGPLAY	425
GRABBING GRAPHICS	425
Playing Picasso	425
Capturing	426
Image Hunting	427
WHAT NOW?	431
13 SOUND	433
HEARING AIDS	436
Installing SPEAK.EXE	437
Configuring SPEAKER.DRV	437
JAMMIN'	438

TABLE OF CONTENTS

WPLANY	438
WHAM	439
LESSON #1: HYPER SOUND	440
LESSON #2: AMBIENT SOUND	442
SOUNDING OFF	443
Audio Formats	443
Recording It Yourself	444
WHAM	444
Searching For That Right Sound	445
WHAT NOW?	445
14 INTERACTIVITY	447
HOW A FORM WORKS	450
LESSON #1: A FORM IS BORN	452
Action	452
Method	452
ENCTYPE	453
TARGET	453
LESSON #2: INPUT FIELDS	453
TYPE	454
NAME	455
DISABLED	456
MIN/MAX	456
LESSON #3: TEXT FIELDS	456
Text Field Variations	457
VALUE	458
SIZE	458
MAXLENGTH	458
LESSON #4: CHECKBOXES OR RADIO BUTTONS	458
VALUE	459
Checkboxes	459
Radio Buttons	460
Prechecked	461
LESSON #5: THE RESET AND SUBMIT BUTTONS	461
Customizing the Label	462
Using Icons	462
LESSON #6: TEXTAREAS	462
Size	463
WRAP	463
LESSON #7: SELECTION LISTS	464
SIZE	465
Multiple Selections	466
Options Attributes	466
LESSON #8: PUTTING IT ALL TOGETHER	467
A Basic Form	467
A Sample Message Form	468
LESSON #9: UPLOADING A FILE	470
LESSON #10: BUTTONS	471
LESSON #11: INCLUDING HIDDEN TEXT	471

LESSON #12: SEARCH INDEXES	472
ISINDEX	472
HREF	473
PROMPT	473
Book-Like Index Documents	473
LESSON #13: TESTING IT OUT	474
LESSON #14: WHAT THE DATA LOOKS LIKE	474
Search Indexes	475
The GET Method	475
POST	476
THE FUTURE OF FORMS	476
WHAT NOW?	477
15 CGI SCRIPTS	479
LESSON #1: WHAT IS CGI?	482
What Can I Do with a CGI Script?	484
LESSON #2: THE STANDARD CGI SCRIPTS	485
LESSON #3: HOW TO USE PRE-EXISTING SCRIPTS	488
Constructing a CGI URL	488
Specifying a Script Within a Form	489
Specifying a Script from the "Open URL" Interface	489
LESSON #4: HOW TO USE A SCRIPT TO ACCESS OTHER APPLICATIONS	490
LESSON #5: WRITING YOUR OWN SCRIPTS	493
Writing a Simple Script	493
Using Environment Variables in Scripts	497
Location and Status Headers	499
Security with CGI Scripts	501
LESSON #6: HANDLING FORM DATA	503
What Does Form Data Look Like?	503
HTTP Cookies	520
LESSON #7: SAMPLE SCRIPTS FOR UNIX, WINDOWS, AND MACINTOSH SERVERS ..	524
Unix	524
Easy Counter	530
Server-Push	533
Perl	535
DOS/Windows Scripts	539
Macintosh Scripts	541
WHAT NOW?	544
16 HTML EXTRAS	545
LESSON #1: INLINED VIDEO WITH INTERNET EXPLORER	548
When to Play It	549
Should There Be a Control Strip?	549
LOOP-D-LOOP	550
LESSON #2: FRAMING WEB PAGES	550
The Frame Document	551
Supporting The Frame-Impaired	558
LESSON #3: TARGETING A FRAME	559
Framing Yourself	560
The Acme Catalog	561

TABLE OF CONTENTS

LESSON #4: CHARGING UP NETSCAPE PLUG-INS	564
How to Install a Plug-in	568
How to Embed a Live Object	569
LESSON #5: LIVEMEDIA: STREAMED VIDEO AND AUDIO	569
Live Audio	570
Live Video	571
LESSON #6: OTHER STREAMED VIDEO AND AUDIO PLUG-INS	572
Voxware	572
VDOLive	575
PreVU	576
RealAudio	576
LESSON #7: LIVE3D AND THE VIRTUAL-REALITY MODELING LANGUAGE	578
Walking Through 3D Worlds	579
Creating Your Own VRML Files	581
LESSON#8: USING SHOCKWAVE	589
Installing Shockwave	589
Afterburning	589
Simply Shocking	591
WHAT NOW?	592
17 JAVASCRIPT	593
WHAT JAVASCRIPT DOES	596
WHAT JAVASCRIPT DOESN'T DO	597
INSERTING JAVASCRIPT INTO YOUR WEB PAGE	597
THE LANGUAGE	600
Objects	600
Functions	601
Methods	602
Variables	603
Event Handlers	603
JAVASCRIPT EXAMPLES	606
LUSCIOUS LINKS	606
After That Mouse!	606
You're History	607
FANTASTIC FORMS	609
Making Forms Smart	609
Processing Forms	610
Error-Checking	610
Interest Calculator Example	615
Beautifyng The Page	616
Scrolling Marquee	616
BUILT-IN OBJECTS AND FUNCTIONS	620
The String Object	621
The Math Object	622
The Date Object	622
Built-in Functions	625
WHAT NOW?	626
18 JAVA	627
JAVA'S JOLT	630

WEB PUBLISHER'S CONSTRUCTION KIT WITH HTML 3.2

THE BIG APPLLET	630
The ABCs of Alpha and Beta	632
An Applet a Day... ..	632
PLACES TO GET APPLETS	632
Applets from the Java Products Group	633
USING EXISTING APPLETS	636
Some Applets to Bite Into	637
Animation	638
Sound	642
Blinking/Nervous Text	642
Scrolling Marquee	643
Image Maps	646
CREATING YOUR OWN APPLLET	649
The Java Developer's Kit	650
Writing The Applet	651
Compiling It	653
Using It	653
Java Development Packages	654
WHAT NOW?	654
19 OTHER WEB RESOURCES	657
LESSON #1: GOPHER	660
LESSON #2: NET NEWS	662
LESSON #3: E-MAIL	665
LESSON #4: WIDE AREA INFORMATION SERVICE	667
LESSON #5: FILE TRANSFER PROTOCOL	668
What Happens to Downloaded Files?	669
How Can I Send a File Using FTP?	671
LESSON #6: TELNET	671
What Does Telnet Allow Me to Do?	671
What Happens When I Initiate a Telnet Session?	671
LESSON #7: ACCESSING LOCAL FILES	672
LESSON #8: URL SCHEMES	673
WHAT NOW?	673
20 COVERTING, TRANSLATING, OR CHEATING	675
WORDPERFECT TO HTML	679
Installing It	679
Writing Your Document	680
Converting	681
MICROSOFT WORD TO HTML	681
ANT_HTML	681
CU_HTML	687
The Internet Assistant	690
Other Tools	691
Netscape Navigator Gold	691
BOOKMARKS TO HTML	692
Lynx	692
Mosaic	693
Netscape	693

TABLE OF CONTENTS

Cello	693
WHAT NOW?	693
21 HTML ASSISTANT	695
INSTALLING THE ASSISTANT	698
LESSON #1: FROM TEXT TO HTML	698
LESSON #2: CREATING FROM SCRATCH	700
LESSON #3: THE TOOLBAR	700
Selection Tools	701
Insertion Tools	703
LESSON #4: INSERTING HYPERLINKS	703
External Links	704
Internal Links	705
Recycling URLs	706
LESSON #5: IMAGES	706
LESSON #6: USER TOOLS	706
Creating a User Tool	707
Entering Enter	708
LESSON #7: FINDING OR REPLACING	708
Finding	708
Replacing	708
LESSON #8: TOOLING WITH TOOLS	708
Undo	709
Auto Repeat	709
Putting an HTML Document on Hold	709
LESSON #9: TESTING 'ER OUT	709
Specifying a Browser	710
The Final Test	710
LESSON #10: PRINTING	710
LESSON #11: URL FILES	710
Grabbing URLs	711
Editing URLs	711
Combining URL Files	712
Saving It	712
LESSON #12: CONVERTING A BOOKMARK, HOTLIST, OR URL FILE TO HTML	712
WHAT NOW?	713
22 HoTMetaL	715
INSTALLING IT	718
LESSON #1: STARTING WITH A TEXT FILE	719
LESSON #2: STARTING FROM SCRATCH	719
Templates	720
Saving	721
LESSON #3: MARKING IT UP	721
Inserting	722
Surrounding	725
Changing	725
Deleting	726
The Smart List	726

WEB PUBLISHER'S CONSTRUCTION KIT WITH HTML 3.2

Pinning	726
LESSON #4: SPECIAL CHARACTERS	727
LESSON #5: WORKING WITH HYPERLINKS	728
Creating External Links	728
Creating Internal Links	729
LESSON #6: WORKING WITH GRAPHICS	730
Image Attributes	730
Viewing Images	731
LESSON #7: WORKING WITH FORMS	731
Input Field	731
TEXTAREA	733
Selection Lists	734
LESSON #8: FINDING SOMETHING	735
The Search Term	735
Search Options	736
Searching	737
LESSON #9: BREAKING THE RULES	737
LESSON #10: STYLES	737
Character	738
Separation	739
Load Styles	739
LESSON #11: TESTING IT OUT	740
Hiding Tags	740
Publishing	741
Previewing	741
LESSON #12: THE HTML BACKBONE	741
The Context Window	742
The Structure Window	742
WHAT NOW?	743

PART III WEAVING A WEB OF YOUR OWN..... 745

23 WHERE TO PLACE YOUR HTML DOCUMENTS..... 747

LESSON #1: TROUBLESHOOTING YOUR HTML DOCUMENTS	750
LESSON #2: HTML DOCUMENT ANALYSIS SERVICES	752
LESSON #3: FREE WEB POSTING SPOTS	755
LESSON #4: WEB SPACE FOR SALE	757
LESSON #5: POSTING	769
LESSON #6: UPLOADING DOCUMENTS	771
Uploading via FTP	771
Transferring Files with a Modem	772
LESSON #7: INSTALLING PAGES ON AN EXISTING WEB SERVER	772
LESSON #8: ANNOUNCING YOUR WEB PAGES	775
WHAT NOW?	777

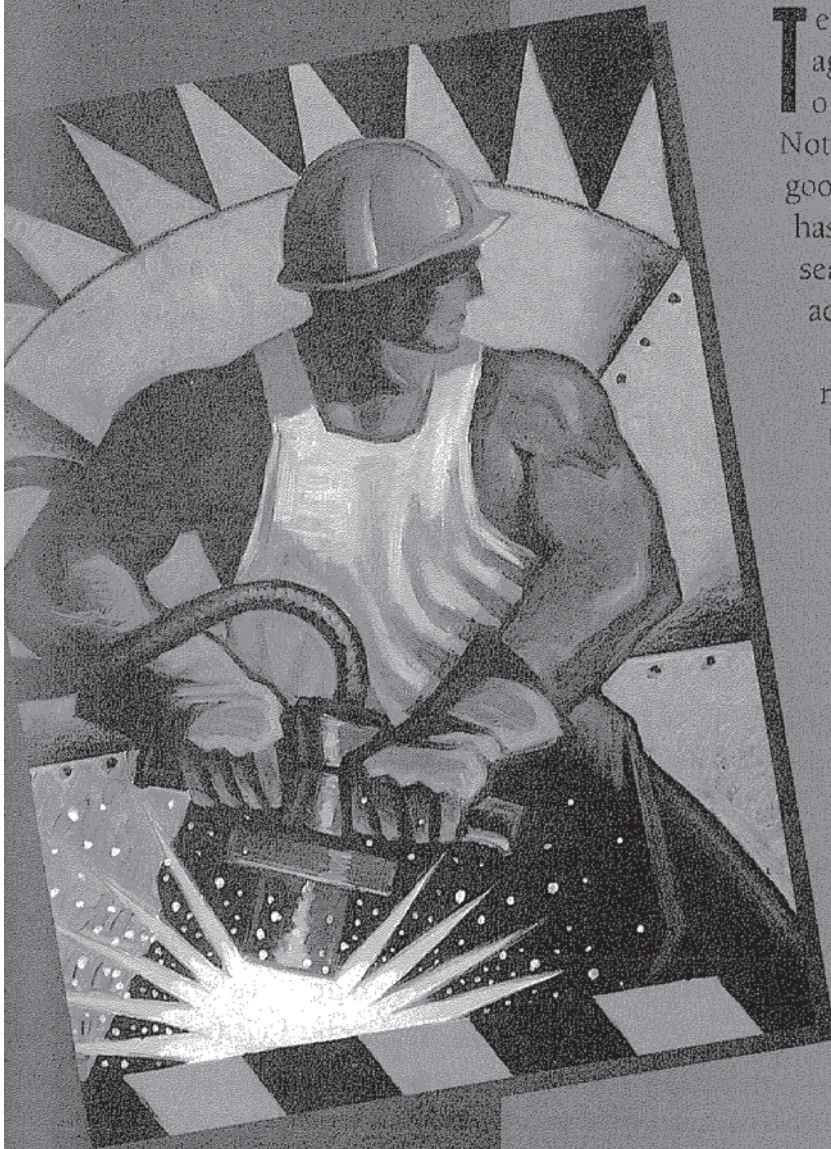
24 STARTING YOUR OWN WEB SITE..... 779

LESSON #1: YOUR WEB SERVER	782
LESSON #2: WEB SERVER SOFTWARE	782
LESSON #3: SERVER SOFTWARE SOURCES	785

TABLE OF CONTENTS

LESSON #4: TIPS AND CAUTIONS	786
Type of Machine	786
Memory	787
LESSON #5: SECURITY!	788
Transmission Security and Data Encryption	789
Pretty Good Privacy Encryption	791
Access Control, Passwords, and Other Filters	793
LESSON #6: HARDWARE	800
LESSON #7: INSTALLING A WEB SERVER	801
Unix Server Installation	801
Netscape's Latest Servers	812
Windows 95 And NT Servers	815
Windows 3.1 Server Installation	816
Macintosh Server Installation	826
LESSON #8: HYPERMEDIA GATEWAYS	831
WHAT NOW?	831
A SLIP SERVICE PROVIDERS	833
B OTHER SOFTWARE SOURCES	839
C WEB HELP RESOURCES	849
D ABOUT THE CD	855
INDEX	859

11



Text is where it all begins. Just a year ago, Internet crawlers pretty much dealt only with text and text-based menus. Not that this was terrible—the text was good, and there was tons of it. Digital text has all sorts of magic attributes. It can be searched, stored, copied, imported, or added on to.

Today's glitzy Web mosaics may make sparser use of the printed word than before, but text always forms the backbone. When designing a Web page, think about the text first. Keep in mind:

HTML People browsing your Web page may only have a text browser, such as Lynx. You don't want to spurn potential clients, associates, friends, or lovers just because their Internet accounts or computer systems can't support a graphical browser.

HTML Although graphics add spice to documents, at the heart of your Web site is information—and nothing can present raw facts and figures better than good old text.

Luckily, HTML provides plenty of text options and formats, allowing you to design layouts as fancy as your imagination. Web text is anything but plain. You can use headings, font sizes, lists, tables, and more, giving the page a crisp, professional layout.

DESIGN FUNDAMENTALS

Until HTML+ came along, Web text was somewhat limited. You could specify the following:

- HTML** Up to six levels of headings
- HTML** Paragraphs or line breaks
- HTML** Lists (bulleted, numbered, or definition style)
- HTML** Basic styles
- HTML** Horizontal separator lines
- HTML** Hypertext links and anchors

With the advent of HTML+, you can customize many of these elements so they look or act exactly the way you want. The latest HTML also adds these new text elements:

- HTML** Font sizing
- HTML** Alignment
- HTML** Tables



STYLE TIP: Be careful when using HTML+ or Mozilla (Netscape-supported HTML commands). People who use Lynx or another primitive browser may

see something completely different. If you plan on creating a fancy Web page, it's a good idea to design a simple page containing the same general information. You can then have a hypertext link at the top of your fancy page, saying something like:

THIS PAGE LOOK STRANGE? [Click here](#) for the simple text-based version!

Consistency

Use consistent punctuation, grammar, and heading styles. The more professional your page looks, the more people will enjoy it, learn from it, and not skip it. This applies to little things as well as big ones: If you use bulleted lists to organize your information on one Web page, try to use lists on all pages. If the theme of your document is particle physics, try not to get off track by talking about the history of cream cheese. If you want to discuss something else, include a separate link to it.

Signing Your Page

You should sign every Web page you create so interested parties can get in touch with you. If you change your document around a lot, you may also want to include the date of the last change. Most people put signatures at the bottom of the page, surrounded by the <ADDRESS> element (described in Lesson #6, Section Style, later in this chapter). You may also want to include a link to your e-mail address as a mailto: URL, so people can just click on your name to send you e-mail. You'll learn about links in Lesson #13, Hypertext, later in this chapter.

```
<ADDRESS>
```

```
<A HREF="mailto:jsmith@smartpants.edu">John Smith</A>, February 14, 1995
```

```
</ADDRESS>
```



STYLE TIP: Since the Web is viewed worldwide, don't use numerical dates. In America, for example, the date 2/1/95 means February first; in Europe it means January second. Write it out instead—February 1 or Jan 2 or whatever.

LESSON #1: SPECIAL CHARACTERS

The computer keyboard is really handy when it comes to letters of the English alphabet. But what about special currency symbols, or accent marks, or other languages altogether?

Punctuation

By now you know that HTML commands are indicated by the less-than and greater-than signs (< and >). But what if you want to include one of these signs as part of your text? HTML uses the ampersand (&) to indicate special characters, as shown in Table 11-1. To print out a special character, write the ampersand followed by a special code and a semicolon, as follows:

```
&charactercode;
```

For example, to write <WOW!> you'd use the commands:

```
&lt;WOW!&gt;
```

Table 11-1 An HTML trick for basic punctuation

HTML Text	Appears As
&	&
>	>
<	<
 	A nonbreaking space; use this to force a blank space



STYLE TIP: Since HTML ignores spaces it deems unnecessary, you can use the nonbreaking space character to force white space. You can then use this white space to separate text or even graphics.

Enhanced Characters

If your Web page includes any Spanish, French, or Latin words you'll probably want to use accented characters. Using the ampersand coding system, you can output most any type of extended or enhanced character as shown in Table 11-2.



STYLE TIP: Not every browser supports these enhanced characters. Be wary of overusing them.

For example, to write the word café, you'd write the command:

```
caf&eacute;
```

Table 11-2 Use these codes to print extended characters on your Web pages

HTML Text	Appears As	Description
Æ	Æ	(capital AE diphthong (ligature))
Á	Á	(capital A, acute accent)
Â	Â	(capital A, circumflex accent)
À	À	(capital A, grave accent)
Å	Å	(capital A, ring)

HTML Text	Appears As	Description
Ã	Ã	(capital A, tilde)
Ä	Ä	(capital A, dieresis or umlaut)
Ç	Ç	(capital C, cedilla)
É	É	(capital E, acute accent)
Ê	Ê	(capital E, circumflex accent)
È	È	(capital E, grave accent)
Ë	Ë	(capital E, dieresis or umlaut)
Í	Í	(capital I, acute accent)
Î	Î	(capital I, circumflex accent)
Ì	Ì	(capital I, grave accent)
Ï	Ï	(capital I, dieresis or umlaut)
Ñ	Ñ	(capital N, tilde)
Ó	Ó	(capital O, acute accent)
Ô	Ô	(capital O, circumflex accent)
Ò	Ò	(capital O, grave accent)
Ø	Ø	(capital O, slash)
Õ	Õ	(capital O, tilde)
Ö	Ö	(capital O, dieresis or umlaut)
Ú	Ú	(capital U, acute accent)
Û	Û	(capital U, circumflex accent)
Ù	Ù	(capital U, grave accent)
Ü	Ü	(capital U, dieresis or umlaut)
Ý	Ý	(capital Y, acute accent)
á	á	(small a, acute accent)
â	â	(small a, circumflex accent)
æ	æ	(small ae diphthong (ligature))
à	à	(small a, grave accent)
å	å	(small a, ring)
ã	ã	(small a, tilde)
ä	ä	(small a, dieresis or umlaut mark)
ç	ç	(small c, cedilla)
é	é	(small e, acute accent)
ê	ê	(small e, circumflex accent)
è	è	(small e, grave accent)
ð	ð	(small eth, Icelandic)
ë	ë	(small e, dieresis or umlaut mark)
í	í	(small i, acute accent)

Continued on next page

Continued from previous page

HTML Text	Appears As	Description
î	î	(small i, circumflex accent)
ì	ì	(small i, grave accent)
ï	ï	(small i, dieresis or umlaut mark)
ñ	ñ	(small n, tilde)
ó	ó	(small o, acute accent)
ô	ô	(small o, circumflex accent)
ò	ò	(small o, grave accent)
ø	ø	(small o, slash)
õ	õ	(small o, tilde)
ö	ö	(small o, dieresis or umlaut mark)
ß	ß	(small sharp s, German (sz ligature))
ú	ú	(small u, acute accent)
û	û	(small u, circumflex accent)
ù	ù	(small u, grave accent)
ü	ü	(small u, dieresis or umlaut mark)
ý	ý	(small y, acute accent)
ÿ	ÿ	(small y, dieresis or umlaut mark)
®	®	(registered trademark; supported by only a few browsers)
©	©	(copyright; supported by only a few browsers)

Other Characters

If you're interested in designing a Web page in a language other than English or containing hard-to-find characters, you have a few options:

HTML Some computers have automatic graphic filters to convert Roman characters into things like Arabic, Hebrew, Chinese, or Japanese. You can then use the standard ASCII set to create your Web pages. For those who don't own such filters, however, the screen will read as gibberish.

HTML The latest version of HTML has a CHARSET attribute for paragraphs, headers, and such, which can be used to specify alternate character sets, such as Hebrew, Japanese, or any other language a browser may support.

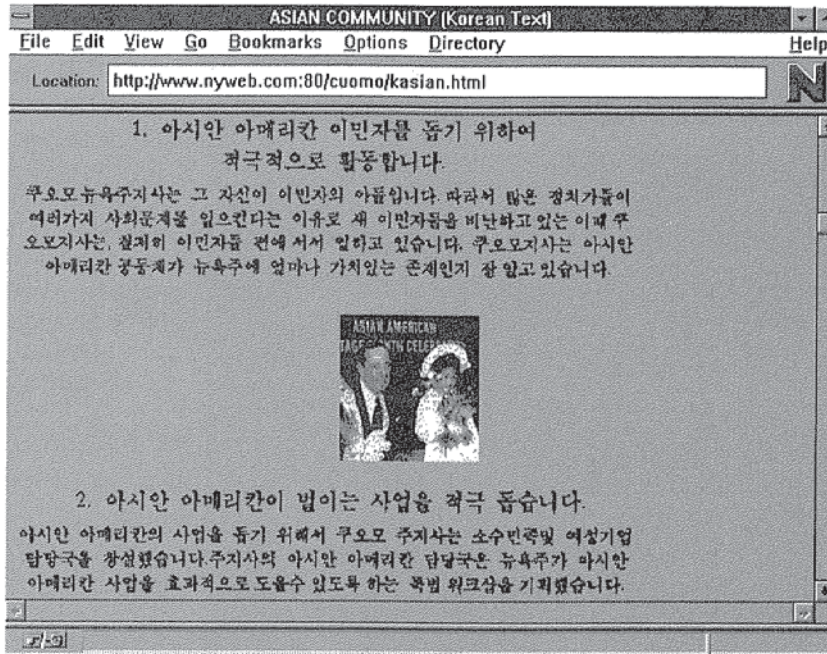


Figure 11-1 A Korean article on the New York Web

- HTML** Create your entire page using any foreign word processor or graphic design tool and then save the entire page as a GIF graphic. You can then make this graphic transparent (see the next chapter) and place it on your page for all to see. See Figure 11-1 for an example.
- HTML** You can obtain or create complete font sets with each letter stored in a separate transparent graphic file. These can be useful for foreign language headlines or small captions.

If you want to use complex mathematical equations, the latest version of HTML+ has some built-in mathematical functions. However, to be sure everybody can read your Web page, your best bet is to create the equations using a program such as LaTeX. You can then use the LaTeX2HTML converter (see Chapter 17) to form HTML pages.

Here are a couple of sites dedicated to multilingual servers:

<http://andrew.triumf.ca/cgi-bin/country-switch/>

<http://www.fer.uni-lj.si/>

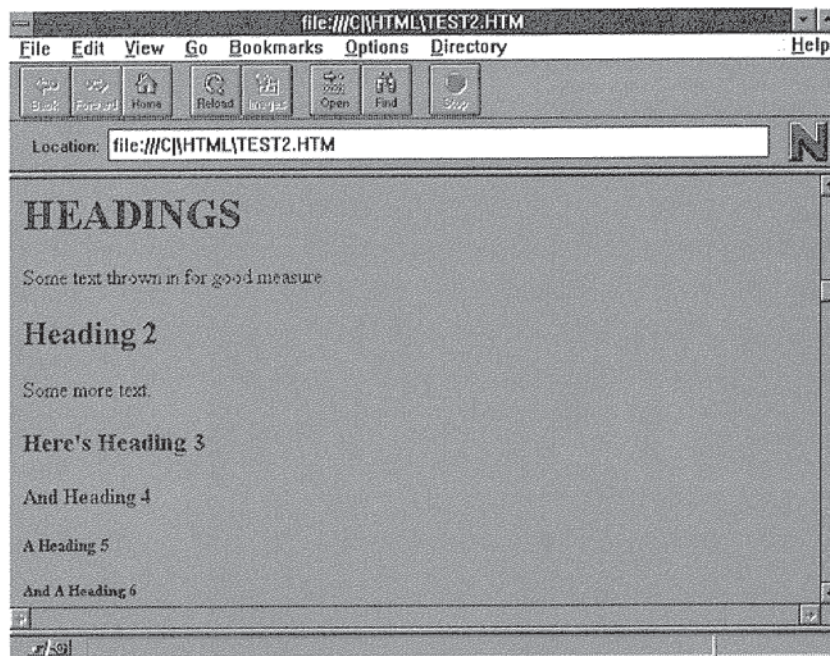


Figure 11-2 A hierarchy of headings

LESSON #2: SECTION HEADINGS

Headings organize your document into smaller, easily-found blocks of information. Headings will generally appear as in Figure 11-2.



STYLE TIP: Although HTML supports six (and in some cases, seven) levels of headings, it's a good idea to stick to four, otherwise your pages can get too complicated to wade through.

HTML headings take the form

```
<H#>heading</H#>
```

where # is the heading level—a number from one to six. For example, a document's main (largest) heading looks like:

```
<H1>Main Heading</H1>
```



STYLE TIP: The first heading is generally the same as the document title.

As in a printed document, the main headings should summarize the broadest ideas, with lower levels of headings forming a logical structure below them. Think of headings as your document's outline.

Here's the full HTML code for the headings and text in Figure 11-2:

```

<HTML>
<HEAD><TITLE>HEADINGS</TITLE></HEAD>
<BODY>
<H1>HEADINGS</H1>
Some text thrown in for good measure.
<H2>Heading 2</H2>
Some more text.
<H3>Here's Heading 3</H3>
<H4>And Heading 4</H4>
<H5>A Heading 5</H5>
<H6>And A Heading 6</H6>
</BODY>
</HTML>

```

In general, you should only use heading tags when you're actually defining a heading. In other words, don't stick a heading in the middle of a list. Although Netscape and Mosaic can handle these misplaced headings, other browsers may become confused.

Style: Drafting Good Headings

Although there's no steadfast rule, subheads generally should not stand alone. Always have at least two subheads in each section. Here's an example of good heading layout:

Desserts

 Pies

 Apple

 Key Lime

 Pumpkin

 Coconut Cream

 Cakes

 Cheese

 Chocolate

 Angel Food

Whereas a less-acceptable, splotchy layout would be:

 Soups

 Chicken

 Desserts

 Pies

 Apple

It also usually looks nicer if you follow each heading by at least one sentence of text instead of immediately tacking on another heading. Describe each subsection so that the reader can decide whether or not to study, skim, or skip it. For example:

```
<H2>Desserts</H2>
There are too many popular and sumptuous desserts to name. However,
the traditional American favorites have been pies, cakes, mousses, and
various fruit dishes.<P>
```

The final caveat with headings is to be sure not to skip a level. Try not to follow an <H1> with an <H3> heading. It confuses some browsers—and some people—when you jump around like that.

Aligning Headings

More advanced browsers such as Netscape allow you to align your headings using the ALIGN attribute. You have the following alignment options:

HTML Left (the default)

HTML Center

HTML Right

HTML Justify

For instance, to center the main heading, use the command:

```
<H1 ALIGN="center">The Heading</H1>
```

LESSON #3: PARAGRAPHS

A Web browser needs to know when one paragraph ends and when a new one begins. It's essential to separate one paragraph from the next. You can do this by using a plain <P> element. This element may appear at either the beginning or end of each paragraph—you don't need to designate both. For example, the text in Figure 11-3 was achieved by the code:

```
Once upon a time there was a little man. When I say little, I mean
little. I mean real little. He was so little most people would just
pass him by. Others would see him and smirk. He felt belittled.<P>
So this little man made himself a little plan. He decided he wasn't
going to take the belittling anymore. He was going to show those
giants a thing or two!<P>
The little man waited for a foggy day. The clouds came down low. Most
everybody was blinded, but not the little man. He could walk between
the confused people's legs, and walk he did. Eventually he reached his
destination: The chainsaw store!<P>
```

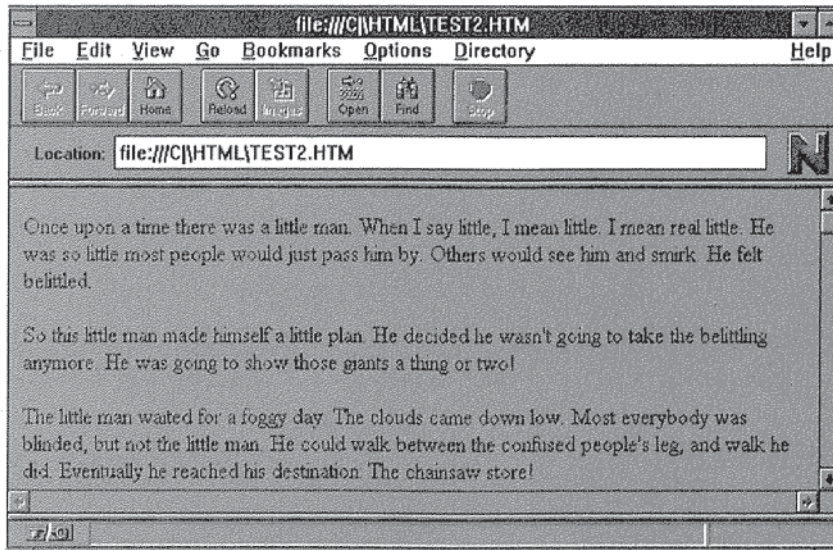


Figure 11-3 Picking a peck of paragraphs

NOTE: Remember, any extraneous blank spaces or lines in your HTML document are usually just ignored.

The latest HTML browsers let you define exactly how your paragraphs should look by using `<P>` at the beginning of your paragraph and `</P>` at the end. This HTML+ paragraph *container* allows you to define formatting attributes for each paragraph. For example, you can give each paragraph a unique ID, making it easy to reference, such as:

```
<P ID="Here">
```

More importantly, you can align center, left, right, justify, or indent each paragraph. For example, you could start each paragraph with the tag:

```
<P ALIGN="justify">
```

As of now, most Web browsers do not support many of these enhanced paragraph markers. Netscape always allows you to left, right, and center align any paragraph.

In general, it doesn't really matter whether you use one empty `<P>` marker or the `<P></P>` container. Use whichever technique seems most intuitive to you.



STYLE TIP: Some browsers will indent the beginning of paragraphs, others will leave a blank line or two. In any case, you don't need to worry about indentation. HTML takes care of this for you.

You do not need to place paragraph markings around, in, or after nonbody text, such as:

HTML Headings

HTML Lists

HTML Preformatted text

HTML Addresses

HTML Blockquotes

HTML Horizontal rule lines

These elements have their own types of separators. For example, when you define a heading, all Web browsers know to add appropriate spacing automatically.

LESSON #4: LINE BREAKS

Now your paragraphs are nice and tidy. But what if you want to put text on specific lines within the same paragraph? For example, although things like postal addresses, poems, and songs are stored in paragraphs, you want to break them up into lines. The solution is the `
` element.

For example, if your HTML document looks like

```
Hello there, dude.  
How are you today?  
I surely hope you are okay.<P>  
That's good to hear!
```

Then, when seen over a Web browser, the text appears like:

```
Hello there, dude. How are you today? I surely hope you are okay.  
That's good to hear!
```

However, if you insert the `
` element after each line, as follows:

```
Hello there, dude. <BR>  
How are you today? <BR>  
I surely hope you are okay.<P>  
That's good to hear!
```

Then the Web text appears on separate lines:

```
Hello there, dude.  
How are you today?  
I surely hope you are okay.
```

```
That's good to hear!
```

No Break

If you have a line of text that absolutely, positively should not be broken up, you can designate this using the `<NOBR>` element, as follows:

```
<NOBR>
This is a piece of text that, for whatever reason, we want all on the
same line.
</NOBR>
```

As of now, only Netscape supports this function.



STYLE TIP: Be careful with this! If your no-break text is too long, it'll scroll right off the screen. This can be a pain to read.

Word Break

If you have a specific no-break section and know exactly where you wouldn't mind the text breaking, you can insert the `<WBR>` element. A word break (`<WBR>`) does not force a break the way `
` does. Instead, it tells Netscape that, should a break be necessary, this is where it should be done.

You can use word breaks to achieve very precise blocks of text.



STYLE TIP: In general, text rolls over onto the new line once it reaches the end of the window. Be careful, because some HTML commands may break up text that is meant to be together. More advanced browsers like Netscape will only break up a line when it comes across empty space such as a space or a tab.

LESSON #5: TEXT STYLE

Most HTML browsers will display the following text emphasis styles:

HTML Bold

HTML Italic

HTML Underlined

HTML Preformatted Text

Highlighted text flows along with any neighboring text. You can also mix and match these elements to achieve, say, bold and italicized text.

There are two general style types: physical and logical. Physical styles tell the browser exactly what the text should look like. Logical styles, on the other hand, explain the kind of text being marked and let the browser do

whatever it does with that kind of text. The next two subsections explain the differences, and you can use whichever type you feel most comfortable with.

All styles have a start tag and an end tag. To highlight a word, phrase, or entire paragraph, surround the text by tags. For example, to create boldface text you'd type:

```
<B>This text will be bold,</B> but this text won't.
```



STYLE TIP: You can usually mix and match emphasis elements to achieve, say, bold and italicized text. You should note, however, that browsers have their own rules for interpreting overlapping styles. Some browsers interpret only the innermost element, which would make

```
<B><I>Bold and Italicized Text</B></I>
```

appear in italics but not boldface. In general, you should immediately surround any text you wish to highlight with its proper physical or logical style elements.

Physical Styles

Physical styles tell the Web browser exactly how a piece of text should appear. Most word processors make use of physical styles. However, each browser will implement a physical style in its own way, and not every browser supports all physical styles.

The supported physical styles are:

HTML **Bold text**

HTML <I> *Italicized text* </I>

HTML <TT> Typewriter fixed-width font text </TT>

HTML <U> Underlined text </U>

HTML+ has specifications for the following additional styles. Most of these are currently supported by browsers such as Mosaic and Netscape:

HTML ^{^{Super}script}

HTML _{_{Sub}script}

HTML <S> ~~Strikethrough~~ </S>

HTML <BLINK> Blinking text </BLINK>

HTML <CHANGE> Change bar </CHANGE>



STYLE TIP: Try not to use too many different types of emphasis in a document, or your text will appear peppered. Stick to one or two styles and be consistent about what type of text you use them for.

Logical Styles

Logical styles, in most cases, get the same results as physical styles. However, a logical style will usually be more consistent from browser to browser. A logical style is more like a description of the enclosed text and is useful for writing scholarly journals, manuals, or other articles involving many types of print.

Here are the most popular logical styles:

- HTML** <DFN>: A word being defined. This word may later appear in a glossary. (Usually displayed as italics.)
- HTML** : Emphasized text. Use this to give words some oomph. (Usually displayed as italics.)
- HTML** <CITE>: A citation. Used around titles of books, journals, movies, poems, etc. (Usually displayed as italics.)
- HTML** <CODE>: Computer code. If you include a program's source code listing, you may want to surround it by this element. (Usually displayed as a fixed-width font.)
- HTML** <KBD>: Keyboard entry. If you tell a user to type a specific set of commands, surround the commands with this element. (Usually displayed as a bold, fixed-width font.)
- HTML** <SAMP>: Sample status messages. (Usually displayed as a fixed-width font.)
- HTML** : Strong emphasis. Good for warnings or other text that you want to leap off the screen. (Usually displayed as bold.)
- HTML** <VAR>: A variable. If you show a command, you can use this element to designate parts of the command that the user should replace with a legal parameter. (Usually displayed as fixed-width italics.)
- HTML** <Q>: An inline quote. (Only Netscape supports this tag, surrounding the text by proper open and closed quotation marks.)

HTML <LANG>: A different language, or different language context. (HTML 3.0)

HTML <AU>: The name of an author. (HTML 3.0)

HTML <PERSON>: The name of a person. This tag is most useful if you want to quickly go through a long series of HTML documents and create an index or list of relevant people. (HTML 3.0)

HTML <ACRONYM>: An acronym. (HTML 3.0)

HTML <ABBREV>: An abbreviation for abbreviation. (HTML 3.0)

HTML <INS>: Newly inserted text. Most useful in legal documents. (HTML 3.0)

HTML : Deleted text. Also useful in legal documents. (HTML 3.0)

For example, if you wanted to emulate the following text:

To change directories in Unix, use the cd command, as follows:
cd directory-name

You would type the HTML sequence:

To change <DFN>directories</DFN> in Unix, use the cd command, as follows:<P>
<KBD>cd <VAR>directory-name</VAR></KBD><P>

LESSON #6: SECTION STYLE

Giving individual words a particular emphasis is a great way to clarify your text. HTML also offers the ability to mark entire paragraphs or text sections with a particular style.

Preformatted Text

If you have a table, indented program listing, a free-form poem, or a tiny piece of ASCII art, you want it to appear over the Web the same way it appears on your screen. The <PRE> element allows you to mark a section of text as preformatted. This text is then printed using the same line breaks, and fixed-width characters. For example the poem:

```
<PRE>
This is a poem,
                I'm not sure why I wrote it.
```

```
But it's a poem nonetheless,  

    Try not to over-quote it.
```

</PRE>

Would appear exactly as you entered it:

```
This is a poem,  

    I'm not sure why I wrote it.  

But it's a poem nonetheless,  

    Try not to over-quote it.
```



STYLE TIP: Hyperlinks can be used within <PRE> sections without any problems. You should avoid using other HTML tags within preformatted text, however, since there's no telling how they'll appear. Since line breaks are automatically interpreted, there's no need for <P> paragraph tags or
 line break tags. You should also avoid tabs, since every Web browser may have different tab settings. Use several blank spaces instead, which will always line up in a fixed-width font.

The Joy of Fixed-Width Fonts

Fixed-width fonts make each character the exact same size. With most other fonts, spaces, periods, and dashes are much shorter than alphanumeric characters—this usually gives the text a nice cosmetic appearance, but it can also make it hard to align different lines. For example, <PRE> lets you draw this cute little animal head:

```
  /\_/\
  (oo)\_/_
    (  )
     o
```

If you don't use the preformatted text style, such a picture would come out skewed:

```
  /\_/\
  (oo)\_/_
    (  )
     o
```

WIDTH

You can use the WIDTH attribute to let a Web browser know how long each line of text should be. The default is 80 lines. For example:


```
<PRE WIDTH=40>
```

This text will automatically be wrapped around after every forty characters. This can be useful if you're trying to create a small block of text, or some other similar effect.

```
</PRE>
```

will produce:

This text will automatically be wrapped around after every forty characters. This can be useful if you're trying to create a small block of text, or some other similar effect.

Blockquotes

The <BLOCKQUOTE> element is useful for any sort of quoted text. This text will usually appear italicized and centered as a block in the middle of the page, as shown in Figure 11-4.

For example, if you're writing a paper about René Descartes, you could include:

Descartes writes:

```
<BLOCKQUOTE>
```

Thus I lived, in appearance, just like those who have nothing to do but live a pleasant and innocent life and attempt to obtain the pleasures without the vices, to enjoy their leisure without ennui, and to occupy their time with all the respectable amusements available. But in reality I never desisted from my design and continued to achieve greater acquaintance with truth.

```
</BLOCKQUOTE>
```

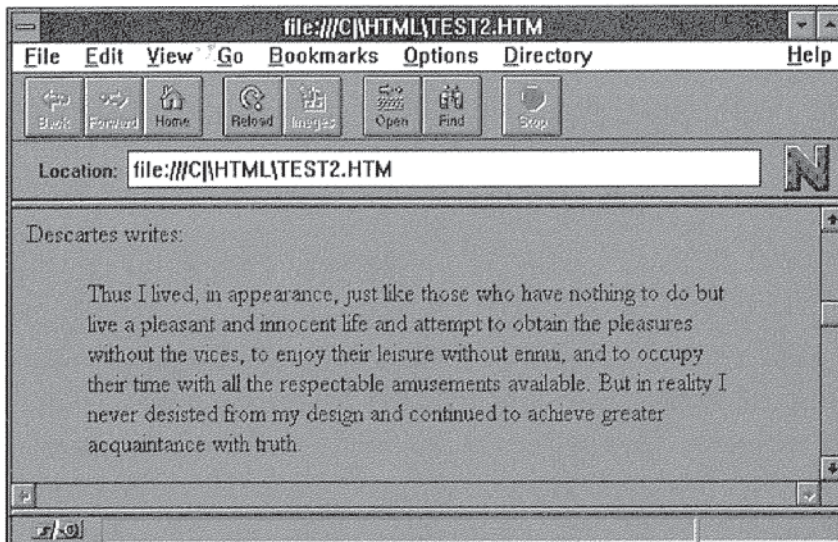


Figure 11-4 A nice way to quote a source

The latest specification of HTML (version 3) has abbreviated the <BLOCKQUOTE> element to just <QUOTE> and </QUOTE>. These shortened elements do not work with many browsers yet.

Addresses

You should surround any home address, office address, e-mail address, signature, or other personal information with the <ADDRESS> element. An address is generally put at the bottom of a document as a signature, or at the top as a sort of byline.

An address is usually printed in italics. It may also be aligned to the right or otherwise indented. For example, you might sign your documents:

```
<ADDRESS>
John Smith<BR>
123 No Way<BR>
Metropolis, IN 12345<BR>
TEL: (000)555-1234
</ADDRESS>
```



STYLE TIP: It's a good idea to include your e-mail address at the bottom of every Web page you create, so interested parties can reach you easily. You may want to use the mailto: URL (supported by most of the latest browsers) to turn your address into an automatic e-mailer. See Chapter 16 for more information.

Styles to Watch Out For

The latest HTML specification includes several additional styles, such as footnotes, margin notes, abstracts, notes, and iconized blocks. While no commercial browsers support these styles yet, they probably soon will.

Footnotes

If you have a source, or some additional information that may ruin the flow of your narrative, you'll soon be able to designate it as a footnote with the <FOOTNOTE> element. A footnote may appear at the end of the text, or may even show up in a pop-up box.

For example, the code:

```
Most Biblical sources give no exact year for Armageddon.
<FOOTNOTE>
A handful of mystics throughout the ages, such as Nostradamus, have
marked the year 2000 to herald the end of the world.
</FOOTNOTE>
This "judgment day" is supposed to be characterized by natural disas-
ters, fire and brimstone falling from the sky, and general turmoil.
```


Would create something that looked like the following:

Most Biblical sources give no exact year for Armageddon.* This "judgment day" is supposed to be characterized by natural disasters, fire and brimstone falling from the sky, and general turmoil.

You could then click on the asterisk to read the additional information.

Margin Notes

The <MARGIN> element will be similar to a footnote. Marginized text appears to the left or right of the current paragraph, or as a separate paragraph. You'll be able to use this method to stick important warnings, notes, or tips in the margins.

For example:

```
The Colt .38 Special revolver contains the typical 6 bullets.
<MARGIN>
When loading a gun be sure to face it away from you!
</MARGIN>
```

Abstract

The <ABSTRACT> element will be useful when you write a scholarly paper. If you surround the summary or article abstract by this element, you should get an abstract that looks like the one in a formal paper: in a small font, centered and justified beneath the paper's title. For example:

```
<H1>The Vicissitudes Of Neo-Templar Restructuring in a Postmodern
Era</H1>
<ABSTRACT>
I have no idea what this paper is about but I'm sure it's interesting.
No, really!
</ABSTRACT>
```

Note

You'll be able to use the <NOTE> element for notes, tips, or warnings. A note will then be set off in a special box or bold font. For example:

```
<NOTE>Be sure not to touch the two wires together!</NOTE>
```

Role

The <ROLE> element will mark a specific piece of text with an icon. You'll be able to give a role the following attributes, each of which displays a different icon:

 Simple: No icon

 Tip: A pointing finger

HTML Note: The blue information circle

HTML Warning: A yellow traffic warning sign

HTML Error: A stop sign

For example, the command:

```
<ROLE TIP>Counting to ten is much easier if you use your
fingers.</ROLE>
```

You'll also be able to use the SRC attribute to include your own icons. For example:

```
<ROLE SRC="caution.gif">Be careful not to touch the wires
together!</ROLE>
```

See the next chapter for details about including graphics.

LESSON #7: FONT SIZE AND COLOR

As of now, only the Netscape browser allows you to adjust font size. Adjusting fonts lets you design all sorts of special effects, as shown in Figure 11-5.

To change a font's size, just use the command:

```
<FONT SIZE=size>
```

The size value must be between 1 and 7. In general, a basic font is size 3. So, if you wanted to create a word that appeared to be shrinking, you could type:

```
<FONT SIZE=7>s<FONT SIZE=6>h<FONT SIZE=5>r<FONT SIZE=4>i<FONT
SIZE=3>n<FONT SIZE=2>k
```

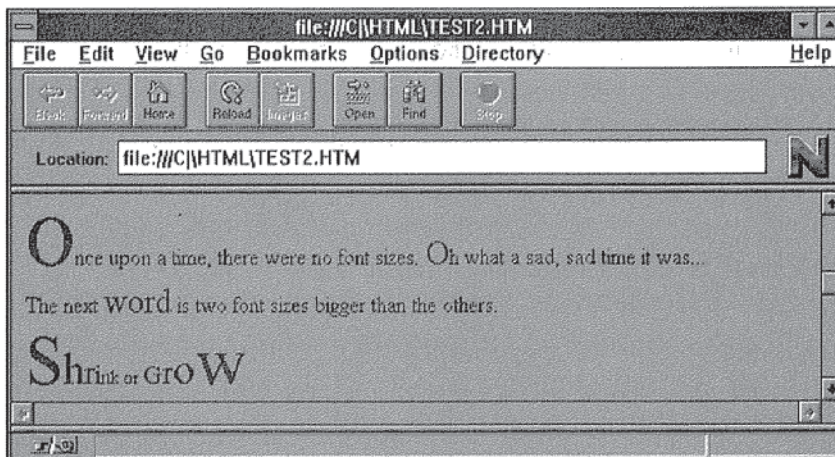


Figure 11-5 Changing font size to spice up your Web page

You can use the `` closing element to return to the default base font. For instance:

```
<FONT SIZE=7>This is big.</FONT> Let's go back to normal.
```



STYLE TIP: Be careful when embedding different-sized fonts alongside other text, especially if the text is more than one line long. The disparity in font size will mess up the vertical spacing between the lines.

Relative Font Size

You can also use the plus or minus character to change a font relative to the document's base font. For example, if you wanted a word to be two sizes bigger than the rest of the text, you would type:

```
The next <FONT SIZE=+2>word</FONT> is two font sizes bigger than the others.
```

The Base Font

The default font size is 3. If you want a document (or a section) to appear in a larger or smaller font, you can set the base font using the `<BASEFONT>` element. For example, to draw all text in the largest possible font, type:

```
<BASEFONT SIZE=7>
```

Making It Easier

HTML version 3 supports an easier way of quickly changing font size: The `<BIG>` and `<SMALL>` elements. To make text grow, simply surround it with the `<BIG>` and `</BIG>` tags:

```
Whatever size this text is, <BIG>this here will be bigger!</BIG> Back to regular size.
```

You can do the same trick using the `<SMALL>` and `</SMALL>` tags:

```
Whatever size this text is, you guessed it, <SMALL>this here will be smaller...</SMALL> Back to regular size.
```

Font Color

You can now specify color for a specific range of text using the `COLOR` attribute within the `` element. Simply set the color to a valid color name or hexadecimal color value. A hexadecimal color takes the form `#RRGGBB`, where `RR` is the red component, `GG` is the green component, and `BB` is the blue component of the color. For a list of popular colors and their hexadecimal values see Table 12-1 in the next chapter.

For example, to print text that starts off normal, becomes red, becomes green, and then becomes blue, you would write:

```
This is normal text here. <FONT COLOR=#ff0000>This is red. Cool,
eh?</FONT> <FONT COLOR=#00ff00>This is green, man.</FONT> And the text
coming up will be <FONT COLOR=#0000ff>blue as the sky on a clear
summer's day</FONT>.
```

Or, even more easily:

```
<FONT COLOR="blue">This will be blue</FONT>.
```

LESSON #8: CENTERING

Perhaps the most typically used style of text alignment is centering. Centered text is flashy, professional, and easy to read.

In previous sections, you learned about how you could alter the alignment of a paragraph or a heading by using the

```
<P ALIGN="center">
```

or

```
<H1 ALIGN="center">
```

attributes. This format is a bit unwieldy, however, and does not center everything you'd want it to.

Netscape has come up with a more convenient `<CENTER>` element. You can use this element to center any style, font, or manner of text. You can even use `<CENTER>` to center graphic images. Centering is simple:

```
<CENTER>
This text will be centered.<BR>
As will this!<P>
And this!<P>
</CENTER>
```

LESSON #9: COMMENTS

Putting comments in your HTML code helps others steal from you more effectively. The public structure of the Web encourages people to borrow each other's Web pages and improve on them, customizing them to their own needs. Everybody learns from everybody else.

Anything placed within comments will not be printed by a Web browser. The only way for somebody to see your comments is to download a listing of your HTML source.

To begin a comment, use the
`<!--`
 tag. At the end of each comment use the
`-->`
 tag.

A typical comment, then, would look like this:

```
<!-- This section jumps to a random home page. -->
```

Most browsers do not allow you to nest comments (use a comment within a comment). They also get very confused if you use a double-dash (--) anywhere in your comment. And although Netscape and other advanced browsers let you comment out several lines of actual HTML markup, other browsers will read commands inside of comments anyway and output a big mess.



STYLE TIP: In general, then, try to restrict your comments to simple one-liners preceding each section. If you absolutely need to use multiline comments, put a `<!--` at the start and a `-->` at the end of each line.

LESSON #10: LISTS

There are several types of HTML lists:

- HTML** Ordered: a numbered list
- HTML** Unordered: a bulleted list
- HTML** Definition: a list with a term followed by a few words or sentences of description
- HTML** Plain: a list with no numbers, bullets, or definitions
- HTML** Menu: a short list of items, like a plain list, but tighter
- HTML** Directory: a thin listing of many items in several columns

Lists can be mixed, matched, nested, and combined. In other words, you can have an unordered list within an ordered list, as shown in Figure 11-6.



STYLE TIP: You usually add a period to the end of a list item if it's a complete sentence. If not, lists usually look better with no punctuation.

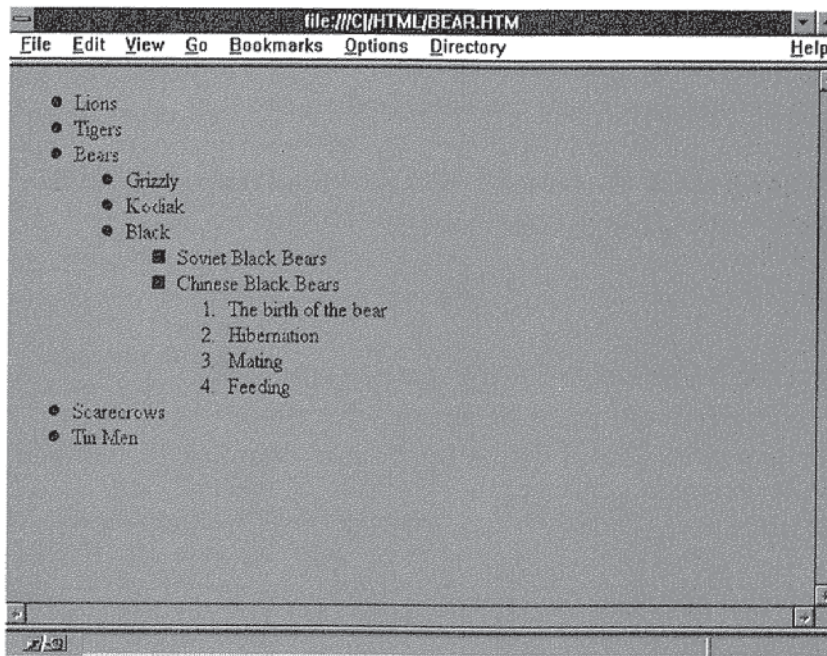


Figure 11-6 Nesting lists together to achieve an outline effect

Ordered Lists

One of the most common ways to express a number of items is to give them all numbers. These numerical, or ordered, lists are great for describing a series of instructions or set of problems.

An ordered list begins with the `` element and ends with ``. Each list item itself is preceded by the empty `` element. For example, to make this list:

1. Remove the cat from the box.
2. Pet the cat.
3. Feed the cat.
4. Place the cat back in the box.

You would use the following HTML instructions:

```
<OL>
  <LI>Remove the cat from the box.
  <LI>Pet the cat.
  <LI>Feed the cat.
  <LI>Place the cat back in the box.
</OL>
```




STYLE TIP: You can create a list without list elements to indent a block of text. This trick works with most browsers. For example, if you want to indent a special tip or idea, you could use the following code:

```
<UL>
This text will be indented, no matter how long the paragraph stretches
on. Pretty cool, eh?
</UL>
```

List Type

By default, an ordered list will count using standard numbers starting from 1, continuing with 2, ad infinitum. More advanced versions of HTML, such as Netscape's Mozilla, allow you to define the type of counting system you want. Use the TYPE attribute, as follows:

```
<OL TYPE=t>
```

In place of t you can fill in one of the following:

HTML A: capital letters

HTML a: lowercase letters

HTML I: large Roman numerals

HTML i: small Roman numerals

HTML 1: standard numbers (the default)

So, for example, if you used the HTML sequence:

```
<OL TYPE=a>
  <LI>Remove the cat from the box.
  <LI>Pet the cat.
  <LI>Feed the cat.
  <LI>Place the cat back in the box.
</OL>
```

Your list would appear as follows:

- a. Remove the cat from the box.
- b. Pet the cat.
- c. Feed the cat.
- d. Place the cat back in the box.

Why Start at One?

In some cases, you don't want your ordered list to begin with the number 1. You may be continuing a list from a previous section, or discussing a group of items from another list.

Using the START attribute, you can begin a list from any number you wish. For example, type

```
<OL START=4>
```

to begin your list with the number 4. The actual value of the list will still depend on the TYPE attribute. So, for example, if you designate:

```
<OL TYPE=I START=10>
  <LI>Remove the cat from the box.
  <LI>Pet the cat.
  <LI>Feed the cat.
  <LI>Place the cat back in the box.
</OL>
```

You'll get a list that looks like this:

```
X. Remove the cat from the box.
XI. Pet the cat.
XII. Feed the cat.
XIII. Place the cat back in the box.
```

Unordered Lists

The bulleted, or unordered, list is one of the best ways to express a series of points quickly. You can use bullets to list features, names, things in an inventory, or just about anything else. An unordered list begins with and ends with . Just like the ordered list, each list element is preceded by the element.

To create the statement

The biggest tourist sites in New York City are:

- The Statue of Liberty
- The World Trade Center
- The Empire State Building
- Rockefeller Center
- Central Park
- The Metropolitan Museum of Art

You would use this sequence:

The biggest tourist sites in New York City are:

```
<UL>
  <LI> The Statue of Liberty
  <LI> The World Trade Center
  <LI> The Empire State Building
  <LI> Rockefeller Center
  <LI> Central Park
  <LI> The Metropolitan Museum of Art
</UL>
```


Nesting Unordered Lists

You can easily create lists within lists, if you like, to achieve an “outline” effect. You can even mix and match unordered and ordered lists:

```
<UL>
  <LI> an item
  <LI> another item
  <LI> here's a nested list:
    <UL>
      <LI> a nested item
      <LI> another nested list:
        <OL>
          <LI> ordered item 1
          <LI> ordered item 2
        </OL>
      </UL>
    <LI> the last item
  </UL>
```



STYLE TIP: Although you can nest as many lists together as you wish, try to curb the nesting to three levels. Beyond this, a list becomes hard to read.

In most cases, your browser draws a small black circle as the first-level bullet. Lynx uses asterisks, and a browser like Netscape uses a small solid disc. As you nest lists, the bullets become indented and the style of bullet changes. Figure 11-6 shows a sample Netscape nested list: circular bullets mark the first and second unordered levels, and squares mark the next two levels.



STYLE TIP: As with headings, no nested list item should stand alone. It always looks much better to make each level of your list have two or more entries. Why else would it be called a list?

Pick Your Own Bullet

Using the TYPE attribute, Netscape and several other HTML+ browsers allow you to designate different styles of bullets. You have three choices, no matter which level of the unordered list you're at:

HTML disc

HTML circle

HTML square

So, to make a short list preceded by two squares:

```
<UL TYPE=square>
  <LI> Item 1
  <LI> Item 2
</UL>
```

Customizing Lists

You now have quite a bit of control over how your lists can look. But Netscape supports an even further level of list customizing. By adding the TYPE or VALUE attribute to the element, you can combine bullets and numbers within the same list, combine various types of numbers, or use various types of bullets.

For example, to make a list that counts backward by twos, using different types of numbers:

```
<OL START=8>
  <LI>Remove the cat from the box.
  <LI VALUE=6 TYPE=I>Pet the cat.
  <LI VALUE=4 TYPE=a>Feed the cat.
  <LI VALUE=2 TYPE=i>Place the cat back in the box.
</OL>
```

This code would produce:

```
8. Remove the cat from the box.
VI. Pet the cat.
d. Feed the cat.
ii. Place the cat back in the box.
```

To make a list with three different types of bullets, you could write:

```
<UL>
  <LI TYPE=disc> Item 1
  <LI TYPE=circle> Item 2
  <LI TYPE=square> Item 3
</UL>
```

Definition Lists

Definition lists are useful for any sort of listing that contains a piece of additional information for each list item. This is great for glossaries, detailed outlines, or even play scripts (where each speech appears as a “definition” for the name of the character saying the words). The list itself starts with the <DL> element and closes with the </DL> element.

Each item in a definition list has two parts, the term—designated by <DT>—and the definition—<DD>. The term—usually one word or phrase—must fit on a single line. The definition itself can take up as many lines as you like.

Figure 11-7 shows a sample definition list. Here’s the markup to create that list:

```
<DL>
  <DT> <B>The Statue of Liberty</B>
```

Continued on next page

Continued from previous page

```
<DD> This symbol of liberty and goodwill to all women and men was
donated to the United States by France. Situated on Liberty Island in
the New York Harbor, the statue is a sight which greeted American
immigrants for decades.
```

```
<DT> <B>The World Trade Center</B>
```

```
<DD> The tallest building in New York, and the second-tallest
building in the world. Also known as the Twin Towers.
```

```
<DT> <B>The Empire State Building</B>
```

```
<DD> Once the world's tallest building, it now ranks third. The
needle-like architecture makes this building a distinct New York land-
mark.
```

```
<DT> <B>The Metropolitan Museum of Art</B>
```

```
<DD> One of the world's premiere art museums, spanning most every
era and culture.
```

```
</DL>
```



STYLE TIP: It generally looks good to set the definition term in bold, italic, or some other sort of enhanced print. Note how each <DT> element in the above code is also surrounded by the attribute.

The <DL> element also has the optional COMPACT attribute. You can use this to create a list that appears condensed on some browsers.



STYLE TIP: Officially, you're not supposed to use more than one definition (<DD>) for each term (<DT>). If you want a definition that is longer than one paragraph, just use the <P> element and continue writing. Some browsers may get confused if you create a definition without an associated term. You can, however, list a term without including a definition for it.

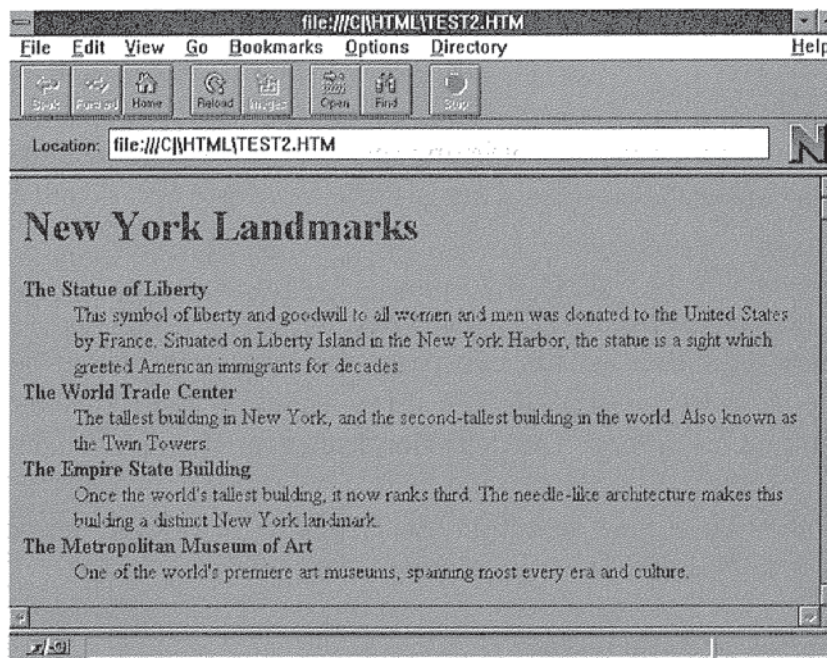


Figure 11-7 A standard definition list

Plain Lists

If you want to list items without using any fancy bullets, numbers, or definitions, you can easily create a plain list. Simply add the PLAIN attribute to any unordered list container. For example:

```
<UL PLAIN>
  <LI>First item
  <LI>Second item
  <LI>The Third
</UL>
```

A plain list can either shoot straight down the screen, or wrap over from line to line. This way, a list of twenty elements will be ordered in columns across the screen rather than wasting many pages. To wrap a list, use the WRAP attribute, with one of the following tags:

HTML vert

HTML horiz

For example, you could create a list that went across the screen horizontally:

```
Item1  Item2  Item3  Item4  Item5  Item6
Item7  Item8  Item9  Item10
```

by using the markup:

```
<UL PLAIN WRAP=horiz>
  <LI>Item1
  <LI>Item2
  ...
  <LI>Item10
</UL>
```

If you specified the wrap as vert, then the items would be listed vertically instead:

```
Item1  Item4  Item7  Item10
Item2  Item5  Item8
Item3  Item6  Item9
```

The <MENU> list and the <DIR> list, described in the next two sections, are a shorthand representation for popular plain lists. Try to use <MENU> and <DIR> whenever you can, since some browsers do not support the PLAIN attribute.



STYLE TIP: You can use the COMPACT attribute to give your plain lists a tighter appearance.

Menu Lists

A menu list looks very similar to an unordered list, except it usually consists of smaller paragraphs. The <MENU> element generally gets the same results as a plain list. You can use it to create a list that appears tighter, smaller, and with more white space:

```
<MENU>
  <LI>Caviar
  <LI>Champagne
  <LI>Beef Wellington
  <LI>Cheez Whiz
</MENU>
```

Directory Lists

The <DIR> element gives you a quick list of short words or phrases that wrap across the screen. In most cases, <DIR> gets the same results as the <UL PLAIN WRAP=vert> element. Each item in a directory list should be less than 20 characters long:

```
<DIR>
  <LI>John
  <LI>Mary
  <LI>Sue
  <LI>Bob
  <LI>Billy
  <LI>Kathy
  <LI>Frank
  <LI>Tim
  <LI>Elliot
  <LI>Mark
  <LI>Carlos.
</DIR>
```

This would create something similar to the following:

```
John   Billy   Elliot
Mary   Kathy   Mark
Sue    Frank   Carlos
Bob    Tim
```

Some browsers make no distinction between plain lists, menu lists, and directory lists.

LESSON #11: HORIZONTAL LINES

Having lots of text together on a screen can be difficult to read. Luckily, most every Web browser supports horizontal rule lines. Anytime you write

`<HR>` the browser draws a thin line completely across the screen. You can use these lines to mark the beginning or end of sections, to make headings look snazzy, to begin a footnote section, or to separate text from inline graphics or icons.

In Netscape and Mosaic the horizontal rule looks like a thin engraved line. Lynx uses a string of dashes.



STYLE TIP: Do not overuse horizontal rules. Too many lines muddle up the page with empty space, and make things hard to read.

In Thickness and in Health

Netscape and several other HTML+ browsers allow you to adjust the thickness of each horizontal line. Usually the line is 1 unit thick, but you can make it as thick as you like. Simply use the `SIZE` attribute. To make a line 5 units thick, type:

```
<HR SIZE=5>
```

Setting the Line's Width

What if you don't want your line to stretch all the way across the screen? Many times it's nice to have lines that are just as long as a nearby heading. Another nice Netscape feature allows you to determine a line's width by using the `WIDTH` attribute.

If you want, you can determine the exact width in pixels. When you decide how long to make a line, keep in mind that an average VGA screen is about 640 pixels wide. Here's what the element looks like:

```
<HR WIDTH=200>
```

Or, even more easily, you can just determine what percentage of the screen the line should take up:

```
<HR WIDTH=50%>
```

Aligning Your Lines

Finally, Netscape gives you the ability to easily align your lines. Since lines can be any width, why not push them against the right margin or center them? By setting the `ALIGN` attribute you can achieve some nice results.

There are three ways to align your lines:

HTML Left (the default)

HTML Right

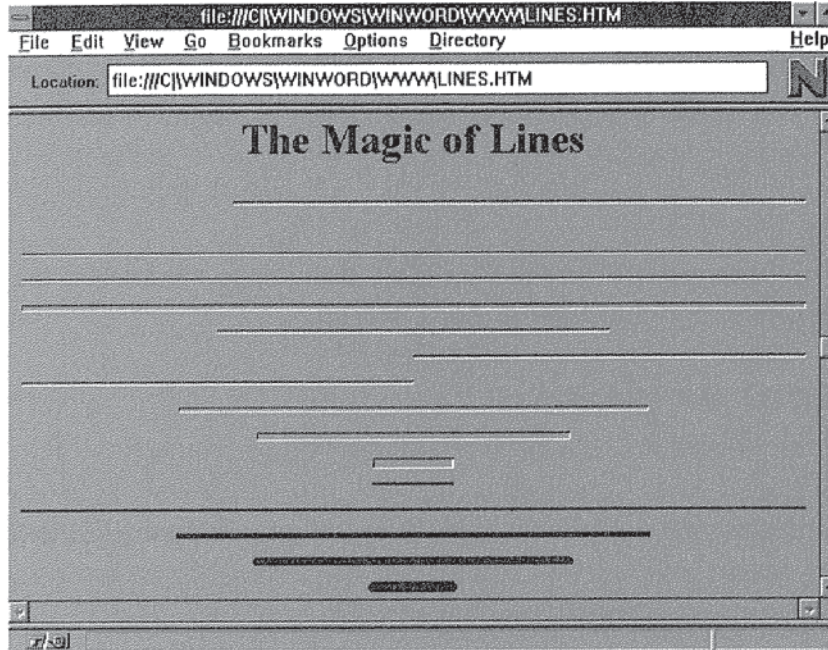


Figure 11-8 Going crazy with customized lines

HTML Center

For example, this element creates a half-sized line in the center of the screen:

```
<HR WIDTH=50% ALIGN=center>
```

Using a Solid Bar

The standard Netscape horizontal line looks as if it was carved into the monitor with a laser beam. If you want a more traditional solid line, simply specify the NOSHADE attribute as follows:

```
<HR NOSHADE>
```

Figure 11-8 shows the results of setting various thicknesses, widths, alignments, and solidities, as follows:

```
<HR>
<HR SIZE=2>
<HR SIZE=5>
<HR SIZE=3 WIDTH=50%>
<HR SIZE=3 WIDTH=50% ALIGN=right>
```

```

<HR SIZE=3 WIDTH=50% ALIGN=left>
<HR SIZE=4 WIDTH=60% ALIGN=center>
<HR SIZE=6 WIDTH=40% ALIGN=center>
<HR SIZE=8 WIDTH=10% ALIGN=center>
<HR WIDTH=10% ALIGN=center NOSHADE>
<HR NOSHADE>
<HR SIZE=4 WIDTH=60% ALIGN=center NOSHADE>
<HR SIZE=6 WIDTH=40% ALIGN=center NOSHADE>
<HR SIZE=8 WIDTH=10% ALIGN=center NOSHADE>

```

LESSON #12: TABLES

There are three ways to create tables:

- HTML** Using preformatted text
- HTML** Using graphics
- HTML** Using HTML+ commands

Text Tables

Every browser has the ability to use preformatted text. Remember, this type of fixed-width text is designated by the `<PRE>` tag and closed by the `</PRE>` element.

You can create your table by hand, then, using ASCII characters, as follows:

```

+-----+-----+-----+
+ Name | Date | Score |
+-----+-----+-----+
| Ralph | January | 10 |
| George | February | 20 |
| Mary | March | 15 |
| Sara | April | 25 |
+-----+-----+-----+

```

Table 3: Each contestant's final score.

This may not be the most professional look in the world, but it gets the job done.

Graphical Tables

You can use any software you'd like to create your table. Then capture the screen, convert the image into a GIF graphic file, and display this image on your Web page. See the next chapter for the specifics.

HTML+ Tables

The latest version of Mosaic and Netscape fully supports HTML+ table commands. Eventually, most browsers should support this feature. HTML table elements allow you to quickly slap together a very professional and sleek-looking table.



STYLE TIP: Since table commands are a relatively new feature, be sure to include a link to a standard preformatted text table. This way, people with browsers who can't view tables can still see your data.

You designate the start of a table using the `<TABLE>` element and end a table with `</TABLE>`. Within a table you can specify captions using the `<CAPTION>` element. Begin each row with the `<TR>` element. Header cells are designated using the `<TH>` element, and data cells are marked by the `<TD>` element.

Each cell can contain a word or a phrase, or entire paragraphs, lists, and headers—even graphical images. Figure 11-9 shows a sample table, which was created using the following code:

```
<TABLE BORDER>
  <CAPTION>Table 3: Each contestant's final score.</CAPTION>
  <TR><TH>NAME</TH><TH>DATE</TH><TH>SCORE</TH></TR>
  <TR><TD>Ralph</TD><TD>January</TD><TD>10</TD></TR>
  <TR><TD>George</TD><TD>February</TD><TD>20</TD></TR>
  <TR><TD>Mary</TD><TD>March</TD><TD>15</TD></TR>
  <TR><TD>Sara</TD><TD>April</TD><TD>25</TD></TR>
</TABLE>
```

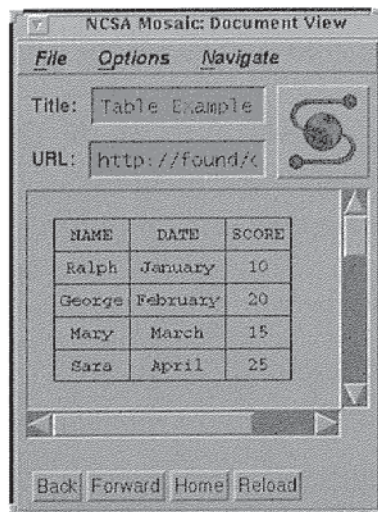



Figure 11-9 A simple but great-looking table created with HTML

Table Cells

Each row in a table begins with the <TR> element. If you like, you can end rows with </TR> though this is optional.







Any text that you want to head up your table begins with the <TH> element, and may end with the </TH> element. This text appears in a bigger, bolder font. Each cell of the table's actual data begins with the <TD> element, and may end with the optional </TD> element. Headings are usually created at the top or to the left of the data.

 **NOTE:** Rows can have different numbers of cells. The table is drawn accordingly.



STYLE TIP: When your Web browser comes across a table, it tries to figure out the table's size before it fills in any values. If the table is too wide to fit on the screen, it usually wraps text over within each cell. In this way, each cell acts as its own little window. Since this might make a table look awkward, try to limit the size of each cell. If you have too much data in one particular cell, you may want to use two separate tables.

A table's headings and data do not have to be plain text. You can place nearly anything within a table:

-  Images
-  Formatted text
-  Various text styles
-  Lists
-  Hyperlink anchors
-  Forms

For example, you could make a table as complicated as the following:

```
<TABLE BORDER>
  <CAPTION ALIGN=top>Table 6: Birds</CAPTION>
  <TR><TH>Famous Birds</TH><TH>Color</TH></TR>
  <TR>
    <TD>
      <OL>
        <LI> The Roadrunner</LI>
        <LI> Big Bird</LI>
        <LI> Chilly Willie</LI>
        <LI> Tweety Bird <IMG SRC= "Tweety.gif"></LI>
      </OL>
    </TD>
    <TD>
      <UL>
        <LI> Green and blue
        <LI> Yellow
        <LI> Black and white
        <LI>Yellow
      </UL>
    </TD>
  </TR>
</TABLE>
```

The Caption

Always designate the <CAPTION> element as the first command in your table. The caption is usually placed directly above the table in a separate font. You can also position the caption using the ALIGN attribute. For example:

```
<CAPTION ALIGN=bottom>Table 5. A neat table.</CAPTION>
```

would print the caption at the bottom of the table. To print the caption at the top, you'd use:

```
<CAPTION ALIGN=top>Table 5. A neat table.</CAPTION>
```

The Border

If you want to give your table a border, be sure to specify the BORDER attribute:

```
<TABLE BORDER>
```



STYLE TIP: In general, you can keep small tables free of borders. However, large tables can be hard to read unless each cell has a border.

Alignment

Data is usually centered within each cell. You can use the ALIGN attribute, however, to specify the horizontal alignment of cell headings or cell data:

HTML Left

HTML Center (the default)

HTML Right

Use VALIGN to specify the vertical alignment:

HTML Top

HTML Middle (default)

HTML Bottom

For example, to place the words "Top left" at the top left corner of a cell:

```
<TD ALIGN=LEFT VALIGN=TOP>Top left</TD>
```

NOWRAP

Use NOWRAP to specify whether or not text within cells should wrap around. If you don't want text-wrapping, use markup similar to the following:

```
<TD NOWRAP>
```


Merging Cells

Two cells can be merged to form one cell; this way you can create one header cell for two columns. To make a cell span more than one row, use the ROWSPAN attribute; use COLSPAN to make the cell larger than one column. You can make any cell heading or cell data as wide or as tall as you wish.

For example, the code:

```
<TABLE BORDER>
  <CAPTION>Wide rows</CAPTION>
  <TR><TH ROWSPAN=2></TH>
    <TH ROWSPAN=2>NUMBER OF<BR>CONTESTANTS</TH>
    <TH COLSPAN=2>SCORE</TH>
  </TR>
  <TR><TH>HIGHEST</TH>
    <TH>LOWEST</TH>
  </TR>
  <TR><TH ALIGN=LEFT>MALES</TH>
    <TD>20</TD>
    <TD>101</TD>
    <TD>23</TD>
  <TR><TH ALIGN=LEFT>FEMALES</TH>
    <TD>823</TD>
    <TD>103</TD>
    <TD>12</TD>
</TABLE>
```

produces the table in Figure 11-10.

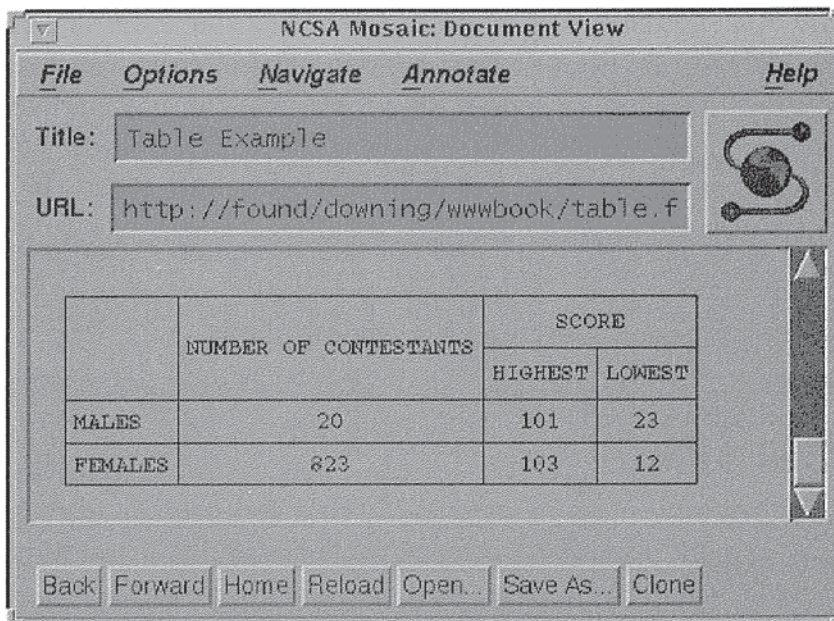


Figure 11-10 A table with merged cells

In other words, if you designate a cell with `ROWSPAN=2`, then it becomes two units high, merging with the cell directly below it. You only need to create one heading for a set of merged cells. If you make a cell with `COLSPAN=2` then it becomes two units wide, merging with the cell to the immediate right. Your Web browser automatically centers all headings, making merged cells easy to read.

In cases where you have labels along the top and side of a table, you can create an empty cell in the upper-left corner just by typing `<TH>` twice.

When merging a column of cells, you can force an entry to take up more than one line by using the `
` element.



STYLE TIP: Be sure you don't create row and column spans that cause cells to overlap. The resulting table is usually pretty hard to read.

Netscape's Table Extensions

Netscape creates lovely-looking tables, with three-dimensional borders, colors, and tight layouts. Some additional Netscape attributes let you customize the look of a table even further. For example, Mozilla HTML allows you to assign the table's `BORDER` attribute a value, giving different tables different looks. To create a table with a border four units thick:

```
<TABLE BORDER=4>
```

Spacing and Padding

Netscape also allows a `CELLSPACING` and `CELLPADDING` attribute as part of your `<TABLE>` element. By default, there are two units of space between each cell. If you want to tighten or loosen up a table, you can use markup similar to the following:

```
<TABLE CELLSPACING=4>
```

Padding is the space between a cell's border and its contents. Netscape usually pads a cell by one unit. If you want your cell's contents to be surrounded by more free space, you could use markup similar to:

```
<TABLE CELLPADDING=3>
```

Table or Cell Width

Finally, Netscape allows you to set the exact `WIDTH` of your table or of each cell. You can set a value (in pixels) or a percentage. For example, if you want your table to go across half the Web page, you could use the command:

```
<TABLE WIDTH=50%>
```

Size			
	Height	Weight	Best Score
John	5'10"	150	983
Sara	6'0"	164	901

Figure 11-11 Designing a complicated HTML table



STYLE TIP: Usually Netscape tries to create a table width that looks as good as possible. Only use WIDTH if your table is relatively small and you have an exact layout look in mind. Big tables can end up looking squashed.

You can also use the WIDTH attribute within the <TH> or <TD> elements. This lets you determine the size of a given cell. If you want a cell to take up a quarter of the table, you could use the markup:

```
<TD WIDTH=25%>Big ol' Cell</TD>
```

Design Hints

You might find it helpful to design your tables using a sort of graph, as shown in Figure 11-11. Each square on the graph is a cell. The thick lines represent the sizes of each cell. The bold print represents headings. You can then go through each cell with HTML commands, using the appropriate COLSPAN and ROWSPAN attributes and filling the cells with appropriate data. Each time you design a cell, cross it out, along with any other affected cells in the table. This prevents you from declaring the same cell twice, skewing the entire table.

For example, here's the code for a table based on the graph in Figure 11-11. Note that the caption doesn't show up on the graph, because it doesn't affect cell size or contents.

```
<TABLE>
  <CAPTION ALIGN=bottom>Each month's winners.</CAPTION>
```

Continued on next page

Continued from previous page

```
<TR><TH ROWSPAN=2><TH COLSPAN=2>SIZE<TH ROWSPAN=2>BEST<BR>SCORE
<TR><TH>HEIGHT<TH>WEIGHT
<TR><TH>JOHN<TD>5'10"<TD>150<TD>983
<TR><TH>SARA<TD>6'0"<TD>164<TD>901
</TABLE>
```

LESSON #13: HYPERTEXT

Let's review how hypertext works: You click on a word or phrase or image and then you're magically transported to a wonderful new world. In other words, the Web browser needs to know two pieces of information about each hypertext link:

HTML Which part of the document is the link?

HTML Where does the link lead?

The answer to the first question is the `<A>` link element. The answer to the second question is a URL (covered in detail in Chapter 1) and an anchor.

The Anchor Element

The anchor element puts the hyper in hypermedia. A link designates that the next piece of text or graphics is actually a hypertext doorway to a strange, new, and wonderful place. You designate an anchor or link using the

```
<A>
```

element.

When your link text is finished, you designate that using the closing element:

```
</A>
```

Link text can take many formats. In some cases it is the exact address, file name, or location of the place the user will be zapped to, as in the top half of Figure 11-12.

In most cases, though, a link can be much more subtle, as in the bottom half of Figure 11-12. The word "aquarium" can whisk the user to an online store specializing in aquariums and tropical fish supplies. The style of your document link is completely up to you.



STYLE TIP: Try to create links that don't muck up the flow of the text. People may convert your hypertext document into a plain text document so they can refer to it offline. Your text should stand on its own. You might want to avoid

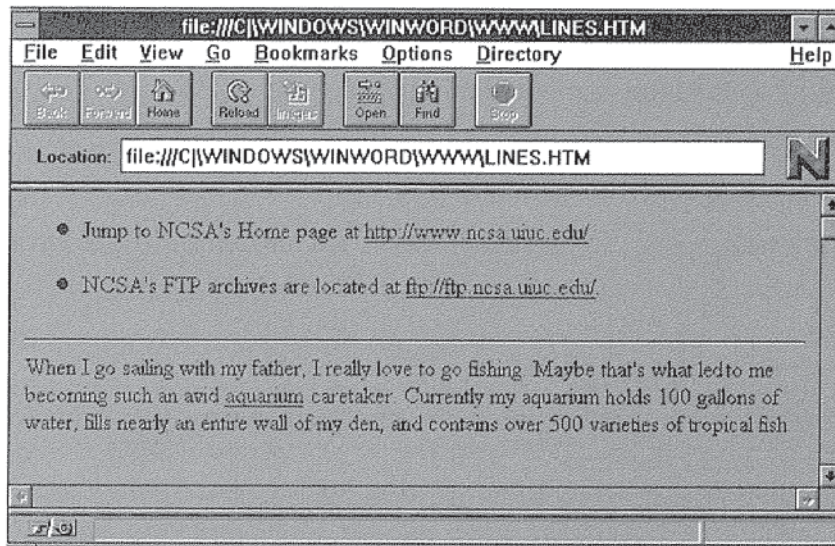


Figure 11-12 You can make anchors large and obvious (top) or subtle (bottom)

words like “click your mouse on this sentence to read about the history of skyscrapers.” Every Web user should know how to click. You’re also being “mouseist”—some sensitive person who doesn’t use a mouse may be offended. Instead, say something like, “There’s an excellent article on the history of skyscrapers by T.O. Tall”—and make the word “skyscraper” your link.

Links do not have to be plain body text. You can generally turn any sort of text—headings, lists, italicized or bold text, or anything else into a link. It’s perfectly fine to put an anchor within another HTML element. For example:

```
<H2><A HREF = "fishstore.html">Fish Stores</A></H2>
```

You should not, however, embed other HTML elements within an anchor, since some browsers may become confused. This is a no-no:

```
<A HREF = "fishstore.html"><H2>Fish Stores</H2></A>
```

You can even make a graphic into a hyperlink. The next chapter talks about how to do this.

The anchor element has a number of optional attributes, including:

- HTML** HREF: jump to another Web page or resource
- HTML** NAME: Destination for jumps from another part of the current document

HTML TITLE: Title of document

HTML REL and REV: Relationship between objects

HTML METHODS: How to link

An anchor is meaningless, however, unless it has either an HREF or a NAME.

Links: HREF

The HREF attribute is the hyperlink reference: the place where you want to jump to. The anchor takes the form

```
<A HREF="url.html">clickable link text goes here</A>
```

Replace "url.html" with the full URL of the destination document (inside the quotes). Any text or graphics between the <A> and the becomes an anchor, and most browsers will highlight it in blue. The user can then click on this blue text and automatically jump to the HREF location. See Figure 11-13 for a breakdown of a complete link.

WARNING: Do not create an anchor that doesn't lead to a valid location. Users will get confused and fed up clicking on links that don't exist. If you're not sure of a URL, you can either use empty quotes or just underline your link text: <U>Link Text</U>. You can then come back later and easily insert the real link. Also be sure to periodically check your remote links, since hypertext can often go stale as pages move or are erased.

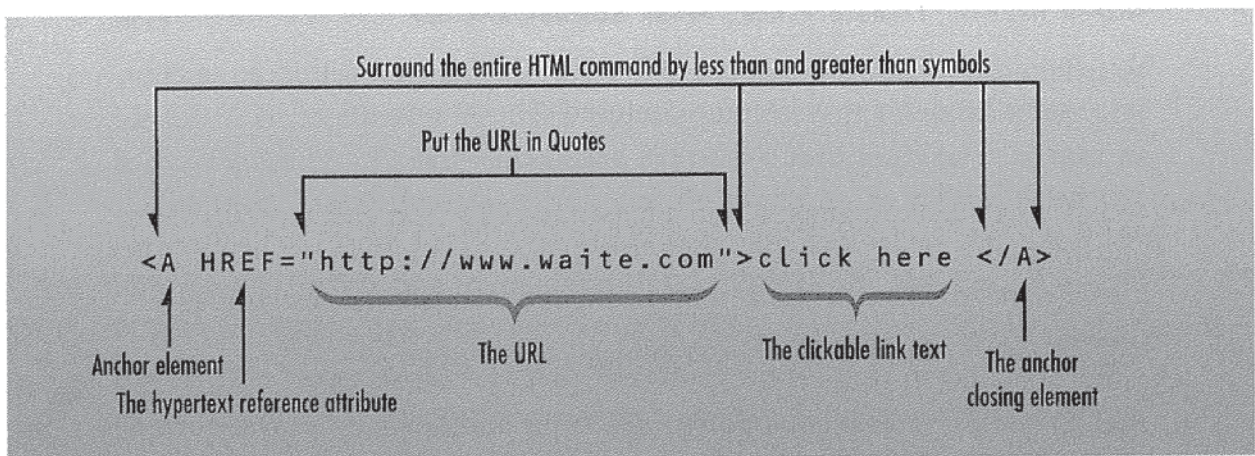


Figure 11-13 The elements in a hypertext link

HREFs often include absolute URLs—the specific addresses of distant Web documents, pictures, sounds, FTP files, gopher menus, telnet sites, newsgroups, e-mail, WAIS databases, and more. Any URL on the Internet can be jumped to. Most URLs are https, leading to another Web server. For example:

```
<A HREF="http://www.cern.edu/dsd/sd">CERN's home page.</A>
```

You can also jump to an FTP site, however:

```
<A HREF="ftp://ftp.netcom.com">Netcom's public FTP archives.</A>
```

Or even telnet to a remote computer:


```
<A HREF="telnet://world.com">Visit the World system.</A>
```

Most HREFs, however, are expressed as relative URLs—identifying documents, images, or other files that are available in the current directory. You can also indicate a relative URL location if you'd like to jump to an HTML document that is stored somewhere on the current Web server. For example, you can have users easily return to your own home page by using the anchor:

```
<A HREF="index.html">Click here to return to the home page.</A>
```

Or, if the file is stored in a different directory, you could indicate the entire pathname:

```
You should visit my friend <A HREF="/web/smith/public-  
html/home.html">John Smith's</A> home page.
```


 **NOTE:** Unix directories are separated by slashes, whereas DOS directories are separated by backslashes. If your HTML server is on a Windows or DOS PC, express filenames the way you normally would. If you are using a Unix machine, be sure to use a Unix directory structure. You may need to check with your system administrator if you're interested in using files in a directory different from your own. See Chapter 21 for detailed information about Web servers.

Relative URLs

When you can, stick to relative URLs. They're easier to type and easier to locate. All relative URLs consist of a simple filename. That file—whether it be a graphic, sound, movie, text file, or another HTML document—must be stored in the same directory as your Web page.


Relative URLs are especially useful if you must move your Web page to another location. For example, if your Internet provider goes out of business, you can transfer your entire Web directory to another machine. All relative URLs will still be valid—you won't need to retype a thing. Simply notify people of your new Web address.


When writing a relative URL, you don't need any slashes as long as the document you're linking to is in the current directory. If you call a URL that is in a different directory on the same machine, simply include the file's full pathname, which usually begins with a slash.


 **WARNING:** Be careful not to use partial pathnames. Such a setup may load up when you use it, but won't work for remote users.

Absolute URLs


You should link to an absolute URL when:

 You didn't create the document being jumped to.

 The document, for space or copyright reasons, is stored on a distant server.

 You're linking to a document that often moves around.

In most cases, you create a link to a distant URL because it fits in with the theme of your page. For example, if I'm interested in tango dancing, I may include a link to the official Tango Page in France. If you're involved with fractal research and there's an interesting movie about fractal animations in a colleague's directory at another university, use that person's complete URL.

 **STYLE TIP:** Don't be shy when it comes to using absolute URLs. Distant hyperlinks are what make the Web a web. Many people create a special Web page with links to their favorite places. In fact, you can convert your Lynx, Netscape, or Mosaic bookmark lists into HTML. Turn to Chapter 17 to find out how.

An absolute URL must take the form

```
protocol://www.domain/directory/filename
```

See the section on URLs in Chapter 1 for details.

Anchors: The Name Attribute

Suppose you have a hundred-page document. Are you going to force readers to scroll down through the entire thing if they're only interested in one section?

The NAME attribute turns the anchored text into an actual hypertext destination. In other words, if you designate a piece of text with a certain name, you can build in explicit jumps to it. This attribute takes the form:

```
<A NAME="label">Optional Text</A>
```

In the corresponding HREF, precede the anchor name with the pound sign (#), so the attribute takes the form:

```
<A HREF="#Label">Put your jump text here</A>
```

Once you designate a named anchor, you can jump to it from anywhere within the document (an internal link), or even from another document (an external link). See Figure 11-14.



STYLE TIP: Since some browsers can get confused by messy anchors, be sure to create anchors at the left margin of a given line. Try to limit your anchors to one word. Anchor names are case sensitive, so be sure to be consistent.

Internal Links: Creating a Table of Contents

The NAME attribute comes in most useful if you set up a table of contents as shown in Figure 11-15. Each line of this table can be a link to a heading that appears later in the same document.

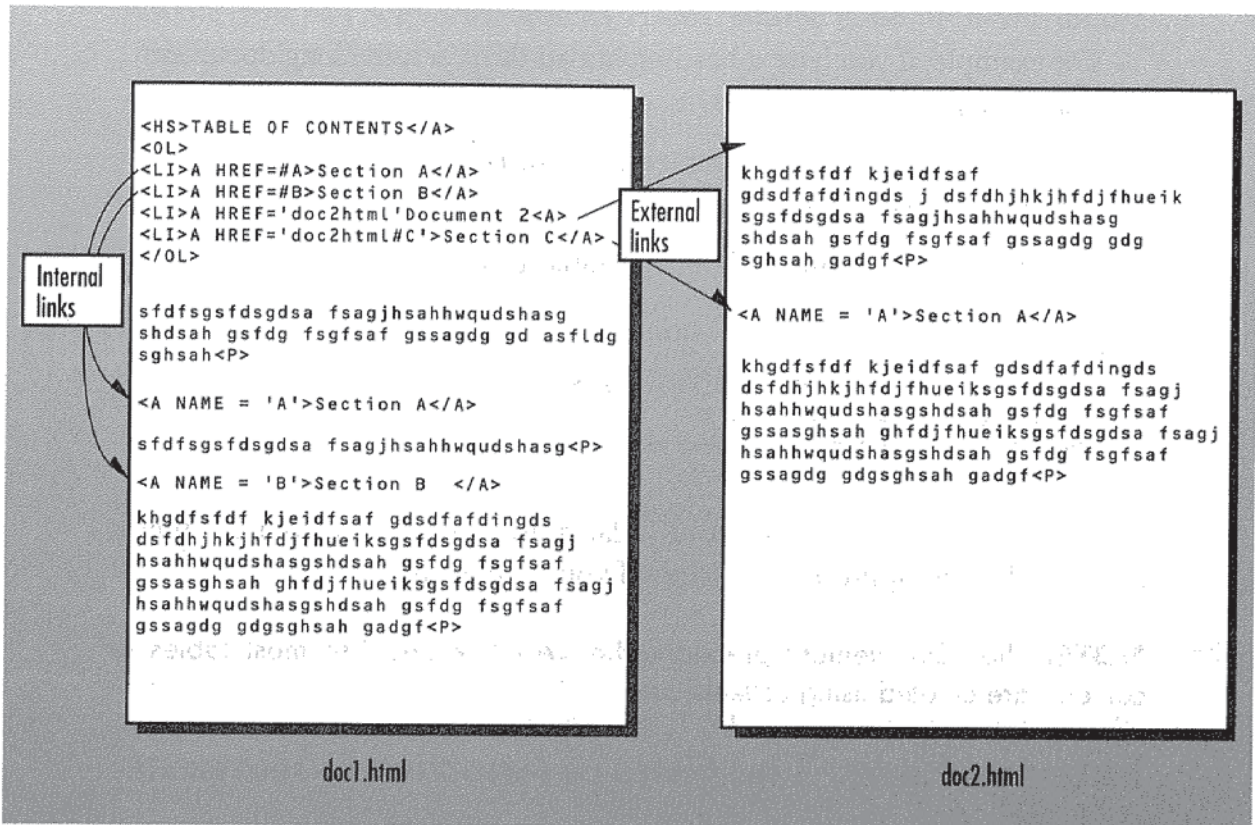


Figure 11-14 The difference between an internal link and an external one

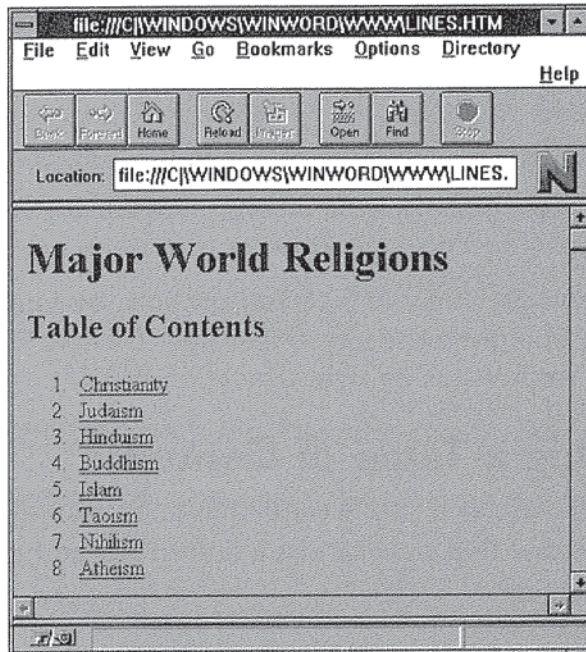


Figure 11-15 A hypertext table of contents

For example, if you have a heading buried deep in some long document:

```
<H2>Judaism</H2>
```

you can mark it as a link by giving it an anchor name:

```
<H2><A NAME="jew">Judaism</A></H2>
```

You can then create a jump from the table of contents by setting up the entries there in a form like this:

```
<OL>
  <LI><A HREF="#christ">Christianity</A>
  <LI><A HREF="#jew">Judaism</A>
  <LI><A HREF="#hindu">Hinduism</A>
</OL>
```

When someone clicks on the word Judaism in your table of contents they'll be warped to the appropriate section of your document.

NOTE: The `` element precedes the `<A>` link because most tables of contents are created using ordered lists.

External Links: The Guts of Another Document

Normally, when you jump to another document using the HREF attribute, you are taken to the beginning of the text. If the NAME attribute is also included,

however, you can begin reading a document from any particular point. This is very useful. Simply append the #NAME text to the end of the URL.

Suppose, for example, you're creating a document about the Middle East. It might be useful to create a link to your document about Major World Religions. When you got to talking about Israel's Jewish populace, for example, you could include the following hyperlink:

```
<A HREF="religions.html#jew">Click here to read some more about Judaism</A>.
```

Assuming there's a properly labeled section in the religions.html document, your readers will be warped directly there.



STYLE TIP: If you have many Web pages dealing with the same topic, you may want to create one master Table Of Contents page with external links to each section of each document. You could then include a button at the bottom of every Web page that returns the user to your table of contents. This makes a lot of material easy to flip through.

You can even create external links to distant absolute URLs. For example, if you find a huge article about marine life on some distant system, you can scan that article's HTML source code until you find a NAME tag. For example, you may find:

```
<H1><A NAME="sharks">Sharks: Deep Sea Killers</A></H1>
```

If you happen to be discussing sharks within your own Web page, you can create a link to the shark section of the marine life article:

```
An <A HREF="http://www.sea.org/marinelife.html#sharks>excellent article</A> about sharks is published by the SEA Organization.
```



NOTE: In the latest HTML+ specification, the NAME attribute has been changed to ID, and is usually a part of the <P> paragraph marker, rather than the <A> anchor. For instance:

```
<P ID="jew">
```

However, for the time being, most browsers do not support this. You should stick to the NAME attribute.

Title

The TITLE attribute is optional, and usually not necessary. TITLE should be used only when creating a link to a non-HTML document. Every HTML document should have its own title. However, telnet sites, Gopher menus, FTP files, and text files do not have official Web titles. Inserting a title in the anchor displays a title at the top of the browser's screen.

This takes the form:

```
<A HREF="plaintext.txt" TITLE="A Plain Text Document">Click here</A>
```

A TITLE is usually informational, acting more as a comment than a useful tag.

Defining Relationships

The REL and REV attributes are used within advanced HTML documents. These two markers define the exact relationship between the linked document and the current document. REL and REV are only used in addition to the HREF attribute.

Most hypertext links consist of a source document (yours) and the destination document (the URL specified in HREF).

REL

The REL attribute expresses a forward relationship between the source and the destination. You can use REL as follows:

```
<A HREF="http://www.here.there/path/crazy.html" REL="Useindex">Click here</A>
```

Here are the most useful relationships you can create between documents:

- HTML** Acyclic: You cannot return to the source document from the destination document.
- HTML** Annotation: The destination document just contains notes, edits, or small additions to the source document.
- HTML** Approves: The destination document has been approved by the source.
- HTML** Embed: The destination document is included in the source document. Some browsers use this to display two documents at once.
- HTML** History: The destination document lists the many versions of the source.
- HTML** Includes: The source document includes the destination within some sort of group.
- HTML** Interested: The owner of the source document is interested in the destination.

- HTML** Made: The source document is a home page or description of the destination's author.
- HTML** Owns: The person who owns the current source document also owns the destination document.
- HTML** Precedes: The source document precedes the destination document. This is useful if you have a book or manual consisting of a string of many Web pages. Often, you can click the browser's Next or Previous arrows to navigate. Each document can be preceded by only one other document.
- HTML** Present: Whenever the source document is presented, the destination document must also be retrieved.
- HTML** Refutes: The destination document refutes an argument made in the source. See Supports.
- HTML** Reply: The destination document is a reply to a question or issue raised in the source.
- HTML** Search: The destination document should be searched, not shown. Most browsers will ask the user to type in a search term. This allows you to create an index or database.
- HTML** Supersedes: The destination document is a previous version of the source.
- HTML** Supports: The destination document supports an argument made in the source. See Refutes.
- HTML** UseGlossary: The destination has the glossary definition of terms that are listed in the source.
- HTML** Useindex: The destination document is a search index.

Most REL tags are pretty much arbitrary and can be used by you to keep track of your Web documents.

REV

The REV attribute is the exact opposite of REL. For example, a link from the source to the destination with:

```
<A HREF="document2.html" REL="Precedes">Click here to view the next
document in the series.</A>
```

Is the same as a link from the destination to the source with:

```
<A HREF="document1.html" REV="Precedes">Click here to view the
previous document in the series.</A>
```

Methods

You almost never need to worry about this attribute. The METHODS tag tells the httpd server which method types the linked-to URL supports.

LESSON #14: RELATIONSHIPS AND HYPERPATHS

You now know how to jump to another document. But what if you want the jump to be automatic? You can create an ordered list of several documents, forming a sort of guided tour of your Web pages.

Once you create a hyperpath, a user would have two ways of accessing adjacent documents along the path:

HTML Clicking on the anchor, as usual

HTML Pressing the Next or Previous button in the browser's toolbar



STYLE TIP: Many browsers won't support forward and backward links. If you're presenting a multipage document, it's always a good idea to include a set of buttons or phrases at the bottom of your page that can take the user to and fro. For example:

```
[Table Of Contents] [Next] [Previous]
```

Which can be created using HTML similar to the following:

```
<A HREF="toc.html" REL="Precedes"> [Table Of Contents] </A>
<A HREF="page5.html" REL="Precedes"> [Next] </A>
<A HREF="page3.html" REV="Precedes"> [Previous] </A><P>
```

Using Link

HTML has a <LINK> element which uses the REL and REV attributes, similarly to the <A> element. For example:

```
<LINK HREF=URL REL="next">
```

The REL attribute takes arguments similar to:

HTML Contents: The destination document is a Table of Contents.

HTML Next: The destination document is the next document in the sequence.

- HTML** Previous: The destination document is the previous document in the sequence.
- HTML** Parent: The destination document is the parent of the current document.
- HTML** Help: The destination document has lots of help for topics in the current document.

The best way to make one document automatically follow another is to use the <LINK> element's REL attribute. The LINK is usually placed in the document's <HEAD>. For example, if you have three documents: doc1.html, doc2.html, and doc3.html, you could put these links at the beginning of your third document:

```
<HEAD>
<TITLE>Sample Document 3 of 10</TITLE>
<LINK HREF="doc2.html" REL="previous">
<LINK HREF="doc4.html" REL="next">
<LINK HREF="toc.html" REL="contents">
</HEAD>
```

and this set of links at the beginning of document 4:

```
<HEAD>
<TITLE>Sample Document 4 of 10</TITLE>
<LINK HREF="doc3.html" REL="previous">
<LINK HREF="doc5.html" REL="next">
<LINK HREF="toc.html" REL="contents">
</HEAD>
```


You could continue doing this for all the documents. Some browsers then allow you to press the Next or Previous buttons to cycle between these ten documents. In the future, browsers may even include an automatic Table of Contents button.

Nodes

The latest specification of HTML also has commands that will make it easy to develop complex hypertext paths. To create stepping stones along a hypertext river, use the <A> anchor's NODE attribute. You can create as many nodes as you like. You usually want to put your nodes in a list for a user to see:

```
<UL>
  <LI><A HREF="node1.html" NODE>The first stop on our tour.
  <LI><A HREF="node2.html" NODE>The second stop.
</UL>
```


A browser reads these nodes, in order, and places them in its own list. A user can then use the Next and Previous button to cycle between the nodes.

 **NOTE:** Modern browsers do not yet support the NODES command.

Path

Eventually, you'll be able to use NODES to create a special HTML document which lines up a specific path. If you want to call this path document, you could use the anchor's PATH attribute. For example, if you have a set of paths listed in the file paths2.html, you could use the command:

```
<A HREF="paths2.html" PATH>Another path</A>
```


The second path would then be added onto any existing nodes.


LESSON #15: MATHEMATICAL EQUATIONS


As of now, there are no widespread Web browsers that can handle equations. However, the proposed new HTML version 3 format has some great mathematical additions. Since the next generation of browsers will most likely implement these elements, it's worth going over them.


HTML+ has elements for all basic mathematical functions. To designate an equation, you would type the $element. To end the equation, use$.


A browser that supports mathematical symbols would have a set of common escape characters, such as:


 `∫`: Integral symbol


 `∞`: Infinity

 `&tanh;`: Tangent

 `α`: Greek Alpha

 `β`: Greek Beta

 `θ`: Greek Theta

 `κ`: Greek Kappa

$$F(t) = \int_n^{\infty} j^t dt$$

Figure 11-16 A sample equation created by HTML+

You can create an equation, then, using a command similar to the following:

```
<math>
  F(t) = &int;<sub>n</sub><sup>&infin;</sup> j<sup>t</sup> dt
</math>
```

which would create the equation in Figure 11-16. The subscript and superscript would automatically center the appropriate numbers over and under the integral sign.

Netscape and Mosaic already support <SUB> subscripts and <SUP> superscripts, making it possible to create simple exponents and such.

The HTML+ math specification also calls for the <BOX> element to group a set of numbers of equations together. The <OVER> element could be used to create fractions with a numerator and a denominator. An <ARRAY> element would print out nice-looking matrices. A <ROOT> element would be used to express various roots.

You can read more about HTML+ math commands and other HTML+ specifications by jumping to:

<http://www11.w3.org/hypertext/WWW/MarkUp/HTMLPlus/>

LESSON #16: DYNAMIC DOCUMENTS: PUSH OR PULL?

The latest version of Netscape Navigator supports dynamic documents by using the server push and client pull. This lets your Web browser receive new data periodically. For example, a Web page with the latest weather map could be updated every ten minutes, or a Web page with stocks could be updated any time a stock's price changed. A video camera can even be used to send the latest frame to your Web browser, forming a live animation or movie.

Server Push

Server push is when the Web server delivers information to a Web page. Unlike a standard Web page, however, the connection between the server and the Web client remains intact. This way, whenever the server decides to update information, the Web page is immediately reloaded.

You can use server push to create very detailed sets of dynamic documents. You can even dynamically change a single image within a Web page, providing video or animation (albeit with a very slow frame rate) within an otherwise unchanging page.

To find out more about how to create CGI scripts which promote server pushing, check out the following URL:

<http://home.netscape.com/home/demo/1.1b1/pushpull.html>


Client Pull

Client pull is similar to server push, but gives the Web page itself more power. A Web page can be programmed to tell a server to reload its data after a given amount of time. The server waits and then automatically gives the Web page the new information.

Client pull is accomplished by using the HTTP-EQUIV="Refresh" attribute in the META element (an HTML3 element that simulates a document's header). The number of seconds to wait is specified using the CONTENT attribute. For example, the following markup should be the first line of your HTML document if you want the Web page to reload itself after 3 seconds:

```
<META HTTP-EQUIV="Refresh" CONTENT=3>
```

The Web page will continue to reload every three seconds.

 **NOTE:** You can use a CONTENT of 0 seconds to load up the given document as quickly as possible.

To load up a different document after a set amount of seconds, simply specify a full URL using the URL attribute. For example, to load the page at <http://www.smartpants.edu/sample.html> after a minute, use the following markup as the first line in your HTML document:

```
<META HTTP-EQUIV="Refresh" CONTENT=60
URL=http://www.smartpants.edu/sample.html>
```

You can use this trick to flip between two documents, animate a sequence of images, or create an automatic “slide show” of many Web pages.

KEY NOTE: To stop a dynamic document while it is loading, a user needs to either close the window, click on a hyperlink, or use the history list to return to a previous document.

LESSON #17: MARQUEES DE HAPPY

Microsoft's Internet Explorer has a unique HTML extension which lets you add scrolling text marquees across your Web pages. These marquees work similar to the scrolling LED signs you might see at the top of a bus or along the sides of buildings in Time's Square. This allows you to put a long message or series of text in a very tiny space. It also draws attention to your Web page, making it stand out from flat, unmoving pages.

Simply drop your marquee anywhere into your HTML document using the `<MARQUEE>` and `</MARQUEE>` tags to surround the text you want to scroll. To quickly drop a marquee into your Web page you could use text similar to the following:

```
<MARQUEE>This text will scroll across my Web page if you're looking at
this using Internet Explorer. Groovy, eh? I hope you like it because I
worked so hard to achieve this effect.</MARQUEE>
```

It's easy to customize the look, feel, and speed of this scrolling using a number of attributes. You can decide how you'd like the text to move by using the BEHAVIOR attribute. You can set BEHAVIOR to any of the following values:

HTML *scroll*: The text begins off the screen and gradually moves across the marquee until it scrolls off the other side of the screen. This is the default behavior.

HTML *slide*: The text begins off-screen and moves to the other side of the screen, where it stops at the margin. This draws immediate attention to the text, but then keeps the text in place from then on.

HTML *alternate*: The text bounces back and forth between both sides of the screen.

You can use the `DIRECTION` attribute to control which direction the text should move. Use *left* if you'd like the text to boogey from left to right (the default value), or use *right* if you want to scroll in the opposite direction.

Use the `SCROLLAMOUNT` and `SCROLLDELAY` attributes to set the speed of the scroll or text movement. Set `SCROLLAMOUNT` to the number of pixels that the text should move between each frame of scrolling. A high value will make the text jittery. A low value will make the text extra-smooth. Set `SCROLLDELAY` to the number of milliseconds between each frame redraw. A higher value will make the text scroll slower. You can play with these two values until the text moves at the speed and smoothness you desire.

By default, your marquee will loop infinitely. You can ensure this by setting the `LOOP` attribute to `-1` or *infinite*. You can also specify a set number of loops. For example, if you want the message to loop twice before stopping use `LOOP=5`.

Use the `BGCOLOR` attribute to set the background color of your marquee. Set it to an RGB hexadecimal triplet (see Table 12-1 for color values).

Finally, you can adjust the size of the marquee itself by using the `HEIGHT` and `WIDTH` attributes. You can set these values to a set number of pixels or to a percentage of the total screen. For example, to make a marquee which takes up 5% of the vertical screen size and is 300 pixels wide use:

```
<MARQUEE WIDTH=300 HEIGHT=5%>This is my marquee.</MARQUEE>
```

Use the `HSPACE` and `VSPACE` attributes to adjust how big the marquee's margins should be. The value should be a set number of pixels.

You can also choose how the marquee will align to any nearby text by using the `ALIGN` attribute. Set this to *top* if you want adjacent text to align with the top of the marquee, *middle* to center the marquee, or *bottom* to align with the marquee's bottom.

So, if you want text to bounce back and forth (starting from the right) within a 300-pixel wide marquee with a blue background at a very fast speed you could use markup similar to:

```
<MARQUEE BEHAVIOR=alternate DIRECTION=right SCROLLDELAY=5
BGCOLOR=#0000ff WIDTH=300>This text will bounce back and
forth.</MARQUEE>
```



STYLE TIP: Since Internet Explorer is the only browser which supports this tag, be aware that any text you put here will appear normal within Netscape

or any other browser. You might want to surround this text with the bold tag or some other tag to plan accordingly.

LESSON #18: POLISHING

Proofread your pages for spelling and grammar errors. The page will be seen by millions of adoring fans around the world. Don't publish something flawed.

You should also double-check your HTML syntax, keeping a sharp eye out for these common problems:

- HTML** Unpaired elements—that is, start-something elements with no corresponding closings (it's easy to forget the slash “/” in </H1>).
- HTML** Missing opening or closing quotes around URLs ()
- HTML** Never-ending comments (<!--How come nothing else on my Web page is getting displayed?!?!?)
- HTML** HTML elements with no closing greater-than bracket (<A NAME=“hello”)
- HTML** Using the less-than, greater-than, or ampersand sign as part of your text (<, >, &).
- HTML** Special characters with no semicolon (>)
- HTML** URLs with uppercase letters when the system wants lowercase () or vice versa

There are several programs and services that run through your HTML code, checking for such errors. You can read about them in Chapter 23, Where to Place Your HTML Documents.



STYLE TIP: Remember, a Web page is always a work in progress. Polish it and polish it again 'til it shines.

WHAT NOW?

Your Web page already looks pretty good, doesn't it? Well, just wait until you add graphics. Flip to the next chapter. Now that you can create hyperlinks, you can read about interesting things to link to in Chapter 12.

If you can hardly wait to put your pages on the Web, check out Chapter 23, Where to Place Your HTML Documents.