

# CORE

## CSS



## PRENTICE HALL PTR CORE SERIES

*Core — Visual Basic 5*, Cornell & Jezak

*Core — Web Programming*, Hall

*Core — Java Foundation Classes*, Topley

*Core — Java Networking*, Niemeyer

*Core — CSS*, Schengili-Roberts

# CORE

## CSS



KEITH SCHENGILI-ROBERTS

Prentice Hall PTR, Upper Saddle River, NJ 07458  
[www.phptr.com](http://www.phptr.com)

Editorial/Production Supervision: Joanne Anzalone  
Acquisitions Editor: Greg Doench  
Editorial Assistant: Mary Traegy  
Marketing Manager: Bryan Gambrel  
Buyer: Alexis Heydt  
Cover Design: Talar Agasyan  
Cover Design Direction: Jerry Votta  
Art Director: Gail Cocker-Bogusz  
Series Design: Meg VanArsdale



© 2000 Prentice Hall PTR  
Prentice-Hall, Inc.  
A Simon & Schuster Company  
Upper Saddle River, NJ 07458

Prentice Hall books are widely used by corporations and government agencies for training, marketing, and resale. The publisher offers discounts on this book when ordered in bulk quantities.

For more information, contact  
Corporate Sales Department  
Prentice Hall PTR  
One Lake Street  
Upper Saddle River, NJ 07458  
Phone: 800-382-3419; FAX: 201-236-7141  
E-mail (Internet): [corpsales@prenhall.com](mailto:corpsales@prenhall.com)

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

All product names mentioned herein are the trademarks of their respective owners.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-0832456-4

Prentice-Hall International (UK) Limited, London  
Prentice-Hall of Australia Pty. Limited, Sydney  
Prentice-Hall Canada Inc., Toronto  
Prentice-Hall Hispanoamericana, S.A., Mexico  
Prentice-Hall of India Private Limited, New Delhi  
Prentice-Hall of Japan, Inc., Tokyo  
Pearson Education Asia Pte. Ltd., Singapore  
Editora Prentice-Hall do Brasil, Ltda., Rio de Janeiro



This book is dedicated to my dear daughter,  
Vanessa Erika Roberts.



# *Contents*

ACKNOWLEDGEMENTS . . . . . XXIX

PREFACE . . . . . XXXI

Who You Are . . . . . xxxii

How This Book Is Organized . . . . . xxxiii

Conventions Used in This Book . . . . . xxxv

Further Information . . . . . xxxvi

Feedback . . . . . xxxvii

## *PART 1:*

*THE ORIGINS OF CASCADING STYLE  
SHEETS . . . . . 2*

1 THE BEGINNING OF HTML AND THE “BROWSER WARS” 4

The World Wide Web Consortium and Cascading Style Sheets . . . . . 7

2	HTML AND ITS RELATIONSHIP TO CSS. . . . .	10
	Adding Cascading Style Sheets to Web Pages	13
	The <STYLE> Tag	14
	The <SPAN> and <DIV> Tags	16
	The <LINK> Tag	22
3	BROWSER ADOPTION OF CSS . . . . .	26
	Browser Differences	29
 <i>PART 2:</i>		
	<i>CSS1 . . . . .</i>	<i>34</i>
4	OVERVIEW . . . . .	36
	Chapter Descriptions	38
5	IMPLEMENTATION OF BASIC CSS CONCEPTS . . . . .	42
	Grouping	44
	Inheritance	45
	CLASS and ID as Selectors	48
	Contextual Selectors	53
	Comments	57
6	THE CASCADE . . . . .	60
	Cascading order	61
	!!IMPORTANT	66

7	CSS UNITS . . . . .	70
	Length Units	71
	Percentage Units	76
	Color Formats	77
	URLs	81
8	PSEUDO-CLASSES AND PSEUDO-ELEMENTS . . . . .	84
	ANCHOR	86
	FIRST-LINE	89
	FIRST-LETTER	92
	Pseudo-Elements in Selectors and Combining Multiple Pseudo-Elements	95
9	FONT PROPERTIES . . . . .	98
	The FONT-FAMILY Property	100
	The FONT-SIZE Property	104
	The FONT-STYLE Property	111
	The FONT-VARIANT Property	113
	The FONT-WEIGHT Property	116
	The FONT Property	121
	THE COLOR AND BACKGROUND	
10	FAMILY OF PROPERTIES . . . . .	126
	The COLOR Property	127
	The BACKGROUND-COLOR Property	129
	The BACKGROUND-IMAGE Property	132

The BACKGROUND-REPEAT Property 135

The BACKGROUND-ATTACHMENT Property 138

The BACKGROUND-POSITION Property 141

The BACKGROUND Property 147

## 11 THE TEXT FAMILY OF PROPERTIES . . . . . 152

The LETTER-SPACING Property 154

The WORD-SPACING Property 157

The LINE-HEIGHT Property 160

The VERTICAL-ALIGN Property 163

The TEXT-ALIGN Property 166

The TEXT-DECORATION Property 170

The TEXT-INDENT Property 173

The TEXT-TRANSFORM Property 176

## 12 THE BOX FAMILY OF PROPERTIES . . . . . 178

The BORDER-STYLE Property 181

The BORDER-COLOR Property 184

The BORDER-WIDTH Property 188

The BORDER-BOTTOM Property 192

The BORDER-LEFT Property 194

The BORDER-RIGHT Property 196

The BORDER-TOP Property 199

The BORDER-BOTTOM-WIDTH Property 202

The BORDER-LEFT-WIDTH Property 204

	The BORDER-RIGHT-WIDTH Property	206
	The BORDER-TOP-WIDTH Property	210
	The BORDER Property	212
	The CLEAR Property	215
	The FLOAT Property	219
	The HEIGHT Property	226
	The WIDTH Property	229
	The PADDING-BOTTOM Property	231
	The PADDING-LEFT Property	233
	The PADDING-RIGHT Property	235
	The PADDING-TOP Property	238
	The PADDING Property	240
	The MARGIN-BOTTOM Property	243
	The MARGIN-LEFT Property	246
	The MARGIN-RIGHT Property	248
	The MARGIN-TOP Property	251
	The MARGIN Property	254
13	THE CLASSIFICATION FAMILY OF PROPERTIES. . . . .	260
	The DISPLAY Property	261
	The WHITE-SPACE Property	264
	The LIST-STYLE-TYPE Property	267
	The LIST-STYLE-IMAGE Property	272
	The LIST-STYLE-POSITION Property	275
	The LIST-STYLE Property	278

*PART 3:*

	<i>CSS2</i> . . . . .	284
14	OVERVIEW . . . . .	286
15	SELECTORS, PSEUDO-ELEMENTS AND PSEUDO-CLASSES	292
	Universal Selector	293
	Child Selectors	294
	Adjacent Sibling Selectors	294
	Attribute Selectors	295
	Pseudo-Elements and Pseudo-Classes	296
16	NEW MEDIA TYPES . . . . .	300
17	THE BOX FAMILY OF PROPERTIES . . . . .	306
	The Border “Side” Color Properties	308
	The Border “Side” Style Properties	311
	Additional Values for the BORDER Sub-Family of Properties	315
18	VISUAL FORMATTING FAMILY OF PROPERTIES . . . . .	318
	The DISPLAY Property	320
	The Set of Positioning Properties	325
	The Z-INDEX Property	334
	Text Direction Properties: DIRECTION and UNICODE-BIDI	336



	DETAILED VISUAL FORMATTING	
19	FAMILY OF PROPERTIES . . . . .	340
	The MIN-WIDTH and MAX-WIDTH Properties	341
	The MIN-HEIGHT and MAX-HEIGHT Properties	344
20	VISUAL EFFECTS PROPERTIES . . . . .	348
	OVERFLOW	349
	CLIP	352
	VISIBILITY	355
	GENERATED CONTENT, AUTOMATIC NUMBERING	
21	AND LISTS . . . . .	360
	The CONTENT Property plus the BEFORE and AFTER Pseudo-Elements	362
	The QUOTES Property	366
	The COUNTER-INCREMENT and COUNTER-RESET Properties	369
	MARKER-OFFSET	373
	LIST-STYLE-TYPE	375
22	PAGED MEDIA FAMILY OF PROPERTIES . . . . .	378
	SIZE	380
	MARKS	382
	The Page Break Properties	384
	PAGE	390
	WIDOWS and ORPHANS	391

23	FONT FAMILY PROPERTIES . . . . .	394
	FONT-STRETCH	396
	FONT-SIZE-ADJUST	398
	The @FONT-FACE Pseudo-Element	401
24	TEXT FAMILY PROPERTIES . . . . .	406
	TEXT-SHADOW	407
25	TABLE FAMILY PROPERTIES . . . . .	412
	CAPTION-SIDE	413
	TABLE-LAYOUT	417
	Controlling Table Borders	419
	EMPTY-CELLS	423
	SPEAK-HEADER	426
26	USER INTERFACE PROPERTIES. . . . .	428
	The CURSOR Property	429
	The OUTLINE Sub-Family of Properties	432
	The OUTLINE-WIDTH Property	432
	The OUTLINE-STYLE Property	434
	The OUTLINE-COLOR Property	436
	OUTLINE	437
27	AURAL STYLE SHEET PROPERTIES. . . . .	440
	The VOLUME Property	442

The SPEAK Property	445
The Pause Sub-Set of Properties	447
The Cue Sub-Set of Properties	451
The PLAY-DURING Property	456
The AZIMUTH Property	458
The ELEVATION Property	462
The SPEECH-RATE Property	465
The VOICE-FAMILY Property	468
The PITCH Property	470
The PITCH-RANGE Property	473
The STRESS Property	475
The RICHNESS Property	476
The Speech Sub-Group of Properties	479

## *APPENDICES*

A	CSS1 PROPERTY, PSEUDO-ELEMENT, PSEUDO-CLASS, AND CONCEPTS REFERENCE . . . . .	482
	Alphabetical Listing of CSS Properties	483
	Pseudo-Elements and Pseudo-Classes	521
	Implementation of Basic CSS Concepts	522
	The Cascade	528

	CSS2 PROPERTY, PSEUDO-ELEMENT AND PSEUDO-CLASS	
B	REFERENCE . . . . .	532
	Alphabetical Listing of CSS2 Elements	533
	Pseudo-Classes and Pseudo-Elements	609
	Media Types	611
	COLORS, UNITS OF MEASURE, PERCENTAGE UNITS AND	
C	URLS . . . . .	614
	Colors	615
	Units of Measure	622
	Percentage Units	624
	URLs	626
	<i>INDEX . . . . .</i>	<i>629</i>

## TABLE OF FIGURES

Figure 2-1	Web page displaying the effects of some simple CSS code. . . . .	12
Figure 2-2	The effects of the sample code for the simple <SPAN> and <DIV> examples displayed. . . . .	20
Figure 2-3	The effects of two separate formatting.css files on the same Web page. . . . .	24
Figure 3-1	The Internet Explorer 5.x Web browser displayed. . . . .	30
Figure 3-2	The Opera Web browser displayed. . . . .	31
Figure 3-3	The Netscape Navigator 4.6 browser displayed. . . . .	32
Figure 3-4	The Mozilla browser (Build M10) displayed. . . . .	33
Figure 5-1	A Web page displaying grouped properties. . . . .	46
Figure 5-2	A Web page displaying inherited properties. . . . .	47
Figure 5-3	Multiple ID and CLASS attributes displayed on a single Web page as seen within Internet Explorer. . . . .	52
Figure 5-4	A nested, unordered list displaying the effects of contextual selectors. . . . .	55
Figure 5-5	More Contextual selectors at work in Internet Explorer. . . . .	57
Figure 6-1	The cascading order code as seen in Opera 3.6. . . . .	64
Figure 6-2	The Cascading Order Code as Seen in Internet Explorer 5.0. . . . .	65
Figure 6-3	!IMPORTANT in action, in this case forcing all paragraphs to display blue text. . . . .	67
Figure 7-1	Various units of measure set to an equivalent indent value of 1 inch from the left margin. . . . .	75
Figure 7-2	The effects of percentage values applied to margin values. . . . .	78
Figure 7-3	The various CSS color value settings displayed. . . . .	81
Figure 7-4	An example of referencing an external file using an URL reference in CSS code. . . . .	83
Figure 8-1	The Effects of the CSS ANCHOR Pseudo-Properties Displayed in Internet Explorer 5.0. . . . .	88
Figure 8-2	The Effects of the FIRST-LINE Pseudo-Element Displayed in the Opera 3.6 Browser. . . . .	92
Figure 8-3	The Effects of the FIRST-LETTER Pseudo-element Displayed in the Opera 3.6 Browser. . . . .	94
Figure 8-4	Combination of the Effects of the FIRST-LETTER and FIRST-LINE Pseudo-Properties Displayed in the Opera 3.6 Browser. . . . .	97

Figure 9-1	The results of the FONT-FAMILY code displayed. . . . .	102
Figure 9-2	Results of the more specific FONT-FAMILY code displayed. . . . .	103
Figure 9-3	The results of the name values for FONT-SIZE displayed in Netscape Navigator. . . . .	106
Figure 9-4	FONT-SIZE displaying the effects of the LARGER and SMALLER settings. . . . .	107
Figure 9-5	Specific measurement unit sizes for FONT-SIZE displayed. . . . .	108
Figure 9-6	The results of percentage values applied to FONT-SIZE as displayed in Internet Explorer. . . . .	110
Figure 9-7	The results of the name values for FONT-SIZE displayed in Internet Explorer. . . . .	111
Figure 9-8	The three values for FONT-STYLE displayed. . . . .	113
Figure 9-9	The three values for FONT-STYLE displayed in Opera. . . . .	115
Figure 9-10	The three values for FONT-STYLE displayed in Internet Explorer. . . . .	116
Figure 9-11	The nine numerical values for FONT-WEIGHT displayed in Netscape Navigator. . . . .	118
Figure 9-12	The BOLDER and LIGHTER values for FONT-WEIGHT displayed in Internet Explorer. . . . .	120
Figure 9-13	The NORMAL and BOLD values for FONT-WEIGHT displayed. . . . .	121
Figure 9-14	Internet Explorer displaying the sample FONT code. . . . .	124
Figure 10-1	HTML and CSS color values displayed in Netscape Navigator. . . . .	129
Figure 10-2	BACKGROUND-COLOR associated with various HTML elements on a Web page. . . . .	131
Figure 10-3	BACKGROUND-IMAGE used associated with several different HTML tags. . . . .	134
Figure 10-4	The effects of BACKGROUND-REPEAT as seen within Internet Explorer. . . . .	137
Figure 10-5	The effects of multiple BACKGROUND-REPEAT values set within a Web page as viewed within Netscape Navigator for Windows. . . . .	138
Figure 10-6	BACKGROUND-ATTACHMENT used to fix the background image in place as seen within Internet Explorer. . . . .	140

Figure 10-7	BACKGROUND-POSITION set to a specific X,Y pixel value, with BACKGROUND-REPEAT set to NO-REPEAT. . . . .	143
Figure 10-8	BACKGROUND-POSITION set to a specific X,Y pixel value, with no BACKGROUND-REPEAT value set. . . . .	144
Figure 10-9	BACKGROUND-POSITION set to a value of 50% for both X and Y, centering the background image in the browser's window. . . . .	145
Figure 10-10	BACKGROUND-POSITION set to a RIGHT BOTTOM with NO-REPEAT set for BACKGROUND-REPEAT. . . . .	146
Figure 10-11	The BACKGROUND property "in action". . . . .	149
Figure 11-1	The use and abuses the LETTER-SPACING property can be put to. . . . .	156
Figure 11-2	The effects of WORD-SPACING as seen within the Opera browser. . . . .	159
Figure 11-3	Using FONT-SIZE and LINE-HEIGHT to set up "double-spacing" on a Web page. . . . .	162
Figure 11-4	Setting LINE-HEIGHT to a value of 200% in order have "double-spacing" on a Web page. . . . .	163
Figure 11-5	The SUB and SUPER values of VERTICAL-ALIGN displayed in Internet Explorer 5.0. . . . .	166
Figure 11-6	Using the four TEXT-ALIGN values to paragraphs of text in Internet Explorer. . . . .	168
Figure 11-7	Going beyond the specification: The four TEXT-ALIGN values work with non-block level elements in Netscape Navigator. . . . .	169
Figure 11-8	Four out of the five values of TEXT-DECORATION displayed in Internet Explorer. . . . .	172
Figure 11-9	The TEXT-INDENT sample code as seen within Netscape Navigator. . . . .	175
Figure 11-10	The four values for TEXT-TRANSFORM as seen within Internet Explorer. . . . .	177
Figure 12-1	The basic margin, border and padding properties displayed. . . . .	180
Figure 12-2	All of the BORDER-STYLE attributes as seen from within Internet Explorer for the Macintosh. . . . .	184
Figure 12-3	The various CSS color format displayed using the BORDER-COLOR property as seen in Internet Explorer. . . . .	187
Figure 12-4	The various CSS color format displayed using the BORDER-COLOR property as seen in Netscape Navigator. . . . .	187

Figure 12-5	Various numerical values applied to the BORDER-WIDTH property as seen in Internet Explorer. . .	190
Figure 12-6	Various name values applied to the BORDER-WIDTH property as seen in Netscape Navigator. . . . .	191
Figure 12-7	The code sample for BORDER-BOTTOM property as seen in Internet Explorer. . . . .	194
Figure 12-8	The code sample for BORDER-LEFT property as seen in Internet Explorer. . . . .	197
Figure 12-9	The code sample for BORDER-RIGHT property as seen in Internet Explorer. . . . .	199
Figure 12-10	The code sample for BORDER-TOP property as seen in Internet Explorer. . . . .	201
Figure 12-11	Code example for BORDER-BOTTOM-WIDTH displayed in Internet Explorer. . . . .	204
Figure 12-12	Code example for BORDER-LEFT-WIDTH displayed in Internet Explorer. . . . .	207
Figure 12-13	Code example for BORDER-RIGHT-WIDTH displayed in Internet Explorer. . . . .	209
Figure 12-14	Code example for BORDER-TOP-WIDTH displayed in Internet Explorer. . . . .	212
Figure 12-15	Sample code for BORDER displayed. . . . .	215
Figure 12-16	Sample code for CLEAR displayed. . . . .	218
Figure 12-17	Sample code for FLOAT and <IMG SRC> where ALIGN=LEFT is set displayed in Internet Explorer. . . . .	222
Figure 12-18	All three values for the FLOAT property displayed in Internet Explorer. . . . .	223
Figure 12-19	FLOAT property used in conjunction with margins set to an image as displayed in Internet Explorer. . . . .	224
Figure 12-20	Several examples of the FLOAT property displayed in a series in Internet Explorer. . . . .	226
Figure 12-21	Sample HEIGHT code examples as seen in Internet Explorer. . . . .	228
Figure 12-22	Sample WIDTH code examples as seen in Internet Explorer. . . . .	231
Figure 12-23	Code example for PADDING-BOTTOM displayed. . . . .	233
Figure 12-24	Code example for PADDING-LEFT displayed. . . . .	235
Figure 12-25	Code example for PADDING-RIGHT displayed. . . . .	237
Figure 12-26	Code example for PADDING-TOP displayed. . . . .	239
Figure 12-27	Code example using measurement values for PADDING displayed. . . . .	242



Figure 12-28	Code example using a percentage value with PADDING displayed. . . . .	243
Figure 12-29	Code example using a set measurement value and a percentage value with the MARGIN-BOTTOM property. . . . .	245
Figure 12-30	Code example using a set measurement value, a percentage value and a negative value with the MARGIN-LEFT property. . . . .	248
Figure 12-31	Code example using a set measurement value, a percentage value and a negative value with the MARGIN-RIGHT property. . . . .	251
Figure 12-32	Code example using a set measurement value and a percentage value with the MARGIN-TOP property. . . . .	253
Figure 12-33	Code example for the MARGIN property as seen in Internet Explorer. . . . .	256
Figure 12-34	Code example for the MARGIN property as seen in Netscape Navigator. . . . .	257
Figure 13-1	Code example for the DISPLAY property as seen in Internet Explorer. . . . .	263
Figure 13-2	Code example for the WHITE-SPACE property as seen in Netscape Navigator (note that the NOWRAP property is not supported). . . . .	267
Figure 13-3	All of the LIST-STYLE-TYPE property values displayed within Internet Explorer. . . . .	271
Figure 13-4	Sample LIST-STYLE-IMAGE code displayed within Internet Explorer. . . . .	274
Figure 13-5	Sample LIST-STYLE-POSITION code displayed within Internet Explorer. . . . .	278
Figure 13-6	Sample LIST-STYLE code displayed within Internet Explorer. . . . .	281
Figure 17-1	The effect of all of the Border-“Side”-Color properties displayed using different color value types. . . . .	310
Figure 17-2	The Border “Side” Style Properties as Seen within the Mozilla Browser. . . . .	315
Figure 18-1	An illustration of the intended effects of the COMPACT and RUN-IN values for the DISPLAY property. . . . .	322
Figure 18-2	Sample code using the MARKER value for the DISPLAY property. . . . .	324
Figure 18-3	An example of POSITION: ABSOLUTE as seen within Internet Explorer 5.0. . . . .	330

Figure 18-4	Image of a Sopwith Camel airplane positioned relative to that of the Fokker airplane, as seen within Internet Explorer 5.0. . . . .	331
Figure 18-5	The FIXED value for the POSITION property displayed in the Mozilla browser. . . . .	333
Figure 18-6	Z-INDEX sample code, as seen in Mozilla. . . . .	336
Figure 19-1	Effects of the MIN-WIDTH and MAX-WIDTH properties displayed within the Mozilla browser. . . . .	344
Figure 19-2	Effects of the MIN-HEIGHT and MAX-HEIGHT properties displayed within the Mozilla browser. . . . .	346
Figure 20-1	Effects of the OVERFLOW property displayed in Mozilla. . . . .	352
Figure 20-2	Effects of the CLIP property as it should be displayed. . . . .	354
Figure 20-3	Effects of the VISIBILITY property as seen in the Mozilla browser. . . . .	357
Figure 20-4	The effects of the VISIBILITY property on two adjacent tables, one with COLLAPSE set to the middle row, and the second with HIDDEN to the middle row. . . . .	359
Figure 21-1	CONTENT example as displayed within the Mozilla browser. . . . .	365
Figure 21-2	QUOTES example as displayed within the Mozilla browser. . . . .	368
Figure 21-3	The COUNTER-INCREMENT and COUNTER-RESET sample code as it ought to be displayed. . . . .	373
Figure 21-4	Three new CSS2 values as seen in the Mozilla browser. . . . .	377
Figure 22-1	Portrait and Landscape printing modes displayed. . . . .	381
Figure 22-2	Crop and Cross marks displayed. . . . .	383
Figure 23-1	The effects of the various FONT-STRETCH values illustrated. . . . .	398
Figure 23-2	Different fonts set to the same point size (12 points). . . . .	399
Figure 23-3	The same fonts with similar legibility values. . . . .	399
Figure 24-1	Sample TEXT-SHADOW code as it should be displayed in a compliant browser. . . . .	410
Figure 25-1	The CAPTION-SIDE code example as it should be displayed. . . . .	416
Figure 25-2	Example code for the BORDER-COLLAPSE and BORDER-SPACING properties displayed. . . . .	422
Figure 25-3	The "empty-cell" effect illustrated. . . . .	423

Figure 25-4	The effects of the EMPTY-CELLS property as seen within the Mozilla browser. . . . .	426
Figure 27-1	Using named values with AZIMUTH to place sound around a listener. . . . .	460
Figure 27-2	Using named values with AZIMUTH to place sound above and below a listener. . . . .	464

## TABLE OF LISTINGS

Listing 2-1	Simple CSS Example . . . . .	12
Listing 2-2	formatting.css #1 . . . . .	23
Listing 2-3	formatting.css #2 . . . . .	23
Listing 5-1	CLASS as Selector . . . . .	48
Listing 5-2	ID as Selector . . . . .	49
Listing 5-3	CLASS and ID as Selectors . . . . .	51
Listing 5-4	Contextual selectors . . . . .	53
Listing 5-5	Contextual selectors #2 . . . . .	56
Listing 5-6	Comments . . . . .	58
Listing 6-1	Cascading Order . . . . .	62
Listing 6-2	!IMPORTANT . . . . .	66
Listing 7-1	Length Units . . . . .	72
Listing 7-2	Percentages . . . . .	76
Listing 7-3	Color Formats . . . . .	79
Listing 7-4	URLs . . . . .	82
Listing 8-1	Anchor . . . . .	87
Listing 8-2	First-Line . . . . .	90
Listing 8-3	First-Letter . . . . .	93
Listing 8-4	Pseudo-Elements in Selectors . . . . .	95
Listing 9-1	Relative FONT-SIZE Values . . . . .	107
Listing 9-2	FONT-SIZE Takes a Percentage Value . . . . .	109
Listing 9-3	Font-Style . . . . .	112
Listing 9-4	FONT-VARIANT . . . . .	114
Listing 9-5	FONT-WEIGHT 2 . . . . .	119
Listing 9-6	Font . . . . .	123
Listing 10-1	Color . . . . .	128
Listing 10-2	Background-Color . . . . .	130
Listing 10-3	Background-Image . . . . .	133
Listing 10-4	Background-Repeat . . . . .	136
Listing 10-5	Background-Position #4 . . . . .	145
Listing 10-6	Background . . . . .	148
Listing 10-7	Background . . . . .	149
Listing 11-1	Letter-Spacing . . . . .	155
Listing 11-2	Word-Spacing . . . . .	158
Listing 11-3	Line-Height . . . . .	161

Listing 11-4	Line-Height #2 . . . . .	162
Listing 11-5	Vertical-Align Example . . . . .	165
Listing 11-6	Text-Align . . . . .	167
Listing 11-7	Text-Decoration . . . . .	171
Listing 11-8	Text-Indent . . . . .	174
Listing 12-1	Border-Style . . . . .	182
Listing 12-2	Border-Color . . . . .	186
Listing 12-3	Border-Width . . . . .	189
Listing 12-4	Border-Width #2 . . . . .	190
Listing 12-5	Border-Bottom . . . . .	193
Listing 12-6	Border-Left . . . . .	196
Listing 12-7	Border-Right . . . . .	198
Listing 12-8	Border-Top . . . . .	201
Listing 12-9	Border-Bottom-Width . . . . .	203
Listing 12-10	Border-Left-Width . . . . .	206
Listing 12-11	Border-Right-Width . . . . .	208
Listing 12-12	Border-Top-Width . . . . .	211
Listing 12-13	Border . . . . .	214
Listing 12-14	Clear . . . . .	217
Listing 12-15	Float #2 . . . . .	221
Listing 12-16	Float #3 . . . . .	224
Listing 12-17	Float #4 . . . . .	225
Listing 12-18	Height Example . . . . .	227
Listing 12-19	Width . . . . .	230
Listing 12-20	Padding-Bottom Example . . . . .	232
Listing 12-21	Padding-Left Example . . . . .	234
Listing 12-22	Padding-Right . . . . .	237
Listing 12-23	Padding-Top Example . . . . .	239
Listing 12-24	Padding Example #1 . . . . .	241
Listing 12-25	Margin-Bottom . . . . .	244
Listing 12-26	Margin-Left . . . . .	247
Listing 12-27	Margin-Right . . . . .	250
Listing 12-28	Margin-Top . . . . .	253
Listing 12-29	Margin Example #1 . . . . .	255
Listing 13-1	Display Example . . . . .	263
Listing 13-2	White-Space . . . . .	266
Listing 13-3	List-Style-Type . . . . .	269
Listing 13-4	List-Style-Type . . . . .	274

Listing 13-5	List-Style-Position . . . . .	276
Listing 13-6	List-Style . . . . .	280
Listing 15-1	First-Child Example . . . . .	297
Listing 16-1	Media Types Example . . . . .	303
Listing 17-1	Border-“Side”-Color Example . . . . .	310
Listing 17-2	Border-“Side”-Style Example . . . . .	314
Listing 18-1	Display Example #1 . . . . .	321
Listing 18-2	Display Example #2 . . . . .	323
Listing 18-3	Position Example #1 . . . . .	329
Listing 18-4	Position Example #2 . . . . .	330
Listing 18-5	Position Example #3 . . . . .	332
Listing 18-6	Z-Index Example . . . . .	334
Listing 18-7	Unicode-Bidi and Direction Example . . . . .	339
Listing 19-1	Min-Width and Max-Width Example . . . . .	343
Listing 19-2	Min-Height and Max-Height Example . . . . .	346
Listing 20-1	Overflow Example . . . . .	351
Listing 20-2	Clip Example . . . . .	354
Listing 20-3	Visibility Example . . . . .	356
Listing 20-4	Visibility Example #2 . . . . .	357
Listing 21-1	Content Example . . . . .	363
Listing 21-2	Content Example #1 . . . . .	364
Listing 21-3	Quotes . . . . .	366
Listing 21-4	Quotes Example . . . . .	367
Listing 21-5	Counter-Increment and Counter-Reset . . . . .	370
Listing 21-6	Counter-Increment and Counter-Rest . . . . .	371
Listing 21-7	Counter-Increment and Counter-Rest Example . . . . .	372
Listing 21-8	Marker-Offset . . . . .	374
Listing 21-9	List-Style-Type - CSS2 Example . . . . .	376
Listing 22-1	Size Example . . . . .	381
Listing 22-2	Page-Break-Before . . . . .	385
Listing 22-3	Page-Break-After . . . . .	386
Listing 22-4	Page-Break-Inside . . . . .	387
Listing 22-5	Page-Break-Before Example . . . . .	388
Listing 22-6	Page-Break-After Example . . . . .	388
Listing 22-7	Page-Break-Inside Example . . . . .	389
Listing 22-8	Page Example . . . . .	390
Listing 23-1	Font-Stretch Example . . . . .	397
Listing 23-2	@Font-Face Example . . . . .	401

Listing 24-1	Text-Shadow Example . . . . .	409
Listing 25-1	Caption-Side Example . . . . .	415
Listing 25-2	Table-Layout Example . . . . .	418
Listing 25-3	Border-Collapse . . . . .	419
Listing 25-4	Border-Spacing . . . . .	420
Listing 25-5	Border-Collapse and Border-Spacing Example . .	421
Listing 25-6	Empty-Cells Example #2 . . . . .	424
Listing 27-1	Volume Example . . . . .	444
Listing 27-2	Speak Example . . . . .	446
Listing 27-3	Pause Properties Example 1 . . . . .	449
Listing 27-4	Pause Properties Example 2 . . . . .	449
Listing 27-5	Pause Properties Example 3 . . . . .	450
Listing 27-6	Cue Properties Example . . . . .	455
Listing 27-7	Play-During Example . . . . .	457
Listing 27-8	Azimuth Example . . . . .	461
Listing 27-9	Elevation Example . . . . .	464
Listing 27-10	Speech-Rate Example . . . . .	467
Listing 27-11	Voice-Family Example . . . . .	470
Listing 27-12	Pitch Example . . . . .	472
Listing 27-13	Pitch-Range Example . . . . .	474
Listing 27-14	Stress Example . . . . .	476
Listing 27-15	Richness Example . . . . .	478
Listing 27-16	Speak-Punctuation and Speak-Numeral Example .	481
Listing C-1	Color Formats . . . . .	616
Listing C-2	Length Units . . . . .	623
Listing C-3	Percentages . . . . .	625
Listing C-4	URLs . . . . .	626





# Acknowledgments



Behind the creation of any book there are always people who are working or helping out behind the scenes in order to make things possible. For this book to be possible, I had to spend the time sequestering myself away for days at a time in order to write code, test it, check the specification to see how it ought to have worked, laugh, then re-write the code if necessary and check it against other platforms and browsers. A lot of people aided me in my work, and this is the space in which I'd like to take the time to thank them in print.

First of all, I'd like to thank my wife and daughter for allowing me to take the time out to write this book. I'd also like to thank my mother-in-law Jean Strimling, who took care of my daughter Vanessa when deadlines loomed. Thanks also goes to Bill Wood IV for the great editing job, and to Dave Truman for taking my text and creating a printable book out of it. For those few screenshots and diagrams that no browser is currently capable of handling, I'd like to thank graphics wizard Andy Yadegar for whipping them up for me. Thanks also to Unix-wizard Sean Healy, who gave me access to the machines I needed to test out CSS compatibility under different operating systems. Last but not least, thanks goes to Prentice Hall editor Greg Doench for taking the leap of faith in letting me run with the subject of CSS and creating this book as a result.



# *Preface*



Welcome to Core CSS. Though Cascading Style Sheets (CSS) have been around for a few years now, it is still underutilized on the Web. The original CSS specification was released by the World Wide Web Consortium back in December of 1996. Since then both Microsoft and Netscape have been slow to implement CSS fully in their respective browsers, and at the time of writing (November 1999) neither the current versions of Internet Explorer nor Netscape Navigator implement CSS fully. Despite this, there are real signs that there is a real drive to make CSS a critical part of Web design in the near future. Browser manufacturers like Opera have pushed for full CSS compatibility, and Netscape has made real efforts to incorporate CSS1 fully and parts of CSS2 into Mozilla, the “precursor” browser to the final release of what will one day become Netscape Navigator 5.0.

Web authors are beginning to realize that CSS provides the power they have been asking for in order to have greater control over how things appear and work on screen. Part of being able to use this power is to understand how it all works. That’s where this book comes into the picture.

## Who You Are

You are a Web author who is looking to expand the capabilities of your Web pages. You know that CSS opens the doors to a wide range of possibilities, but want to learn more about how to make the most of it. Or perhaps you know that CSS will solve some of your most pernicious Web formatting problems, but shy away from using it because you have heard that it can produce varying results when viewed under different browsers or under different operating systems. If you fit either of these circumstances, then this book is for you.

Arguably one of the reasons why CSS has not been adopted as quickly as many other Web technologies have been is the lack of good, solid information as to how it should work. There is also a wide disparity between how CSS is supposed to work, and how it actually works in the major browsers.

This book takes a practical, pragmatic look at the current state of affairs regarding CSS, and guides the reader through how CSS works. This book provides the information Web authors need in order to understand not only how CSS should work, but how it actually works in current major browsers. It does not confine itself only to one operating system, but takes a look at how CSS works under browsers working under multiple operating systems. With this knowledge, Web authors will know what CSS properties are “safe” for use, and which to avoid.

More than that though, this book also provides information as to the future of CSS with an in-depth look at CSS2. CSS2 is a relatively recent specification that takes the original CSS specification further, bringing the Web to new display devices, providing much greater control over the positioning of onscreen elements, even providing Web authors with the control as to how Web pages should *sound* and much more. Browser support for CSS2 is limited at the moment, but it promises to come to the fore in the near future. This book provides the Web author with a guide as to what to expect when CSS2 is widely implemented.

You do not have to be an expert at understanding how the Hypertext Markup Language (HTML) works, but the book does assume you have a basic understanding of both HTML and the Web. The book assumes no prior knowledge of CSS. It will not only serve those Web authors who are just starting out using CSS, but should stand in good stead as a handy reference for those occasions when you need to look up how a particular CSS property works.

## How This Book Is Organized

This book is divided into three parts, with a number of appendices designed to provide the Web author with quick reference material to have on hand when writing CSS code.

Part One, “The Origins of Cascading Style Sheets”, begins by looking at how HTML developed, and the “browser wars” that necessitated the need for Cascading Style Sheets, in an attempt to rein in the burgeoning number of HTML tags unleashed onto Web authors by Microsoft and Netscape. It also looks at the role of the World Wide Web Consortium and the underlying goals behind the development of the first CSS specification. The relationship between HTML and CSS is then explored, looking at the ways in which CSS code can be incorporated into Web pages using such HTML tags as `<STYLE>`, `<SPAN>`, `<DIV>` and `<LINK>` and the HTML attributes `STYLE`, `CLASS` and `ID`. The concept of “cascading” is also explained in brief with its many rules plainly laid out. The final chapter in this part looks at how CSS is actually implemented in various releases of the major browsers.

Part Two, “CSS1”, is devoted wholly to explaining all of the properties belonging to the CSS1 specification in detail. At the same time, the practical uses for each CSS1 property is also emphasized, along with code examples and information explaining any known quirks as to how (or if) a specific CSS1 property’s functionality is implemented in a current browser. This part is broken down into nine chapters, each exploring a section of the CSS1 specification and its properties. The first four chapters explore some CSS fundamentals and how well they work in current browsers. Chapter 5 (“Implementation of Basic CSS Concepts”) and Chapter 6 (“The Cascade”) look at how well these basic concepts (such as inheritance, grouping CSS code and cascading rules) are implemented in the major browsers. Chapter 7 (“CSS Units”) looks at and explains the many different fundamental units of measure that can be used in conjunction with certain CSS properties. The remaining chapters in this part proceed to cover the CSS1 properties and their functionality in detail. Chapter 8 (“Pseudo-Classes and Pseudo-Elements”) looks at how pseudo-classes and pseudo-elements can be utilized. Chapter 9 (“Font Properties”) looks at the many font-related properties in detail, explaining how they can be used to precisely lay out the type of font display to be seen by the user. Chapter 10 (“The Color and Background Family of Properties”) examines how the versatile family of `COLOR` and `BACKGROUND` properties can be used effectively. Chapter 11 (“The Text Family of Properties”) discusses how the many text-related CSS properties can be used

to control how text is displayed onscreen. Chapter 12 (“The Box Family of Properties”) explains and examines the box set of CSS properties, which determine how a wide variety of Web elements such as headers, images and paragraph can have their appearance enhanced. Finally, Chapter 13 (“The Classification Family of Properties”) looks at those properties designed to fundamentally alter the way in which certain onscreen elements are to be displayed within the browser.

Part Three, “CSS2”, examines how the properties of this relatively recent specification are to be used. The CSS2 specification is meant as an additional standard built on top of the CSS1 specification, and is meant to be an adjunct to it. Since many of the properties in the CSS2 specification are already defined and explained in Part Two, only those properties that are wholly new to CSS2 are covered. This part is broken down into a total of fourteen chapters. As with the previous part, the first three chapters of this part are devoted to some CSS2 fundamentals. The first chapter (“Overview”) provides a concise summary of the new features introduced under the CSS2 specification. Chapter 15 (“Selectors, Pseudo-Elements and Pseudo-Classes”) looks at the powerful new selectors, pseudo-elements and pseudo-classes that are available for use. Chapter 16 (“New Media Types”) introduces the concept of media types, and examines how Web pages can be modified so that they can be displayed in other media such as print or “talking browsers”. All of the remaining chapters of this part look in turn at all of the new or significantly enhanced CSS properties. Chapter 17 (“The Box Family of Properties”) looks at the important changes made to this important family of properties as well as the many new properties that are introduced under the CSS2 specification. Chapter 18 (“Visual Formatting Family of Properties”) peers at the properties comprising this new family of functions that gives Web authors much greater control over how things appear onscreen. Chapter 19 (“Detailed Visual Formatting Family of Properties”), deals with those properties that are meant to handle the fine details in the display of many onscreen elements. Chapter 20 (“Visual Effects Properties”) looks at those properties designed to enable Web authors to control how text or other Web elements are displayed when they exceed the dimensions of the box within which they are contained. Chapter 21 (“Generated Content, Automatic Numbering and Lists”) explains how the new functions that are provided with CSS2 enable Web authors to control and enhance content that is automatically generated by the browser, including such things as the numbering and display of lists. Chapter 22 (“Paged Media Family of Properties”) explores those elements related to crafting Web pages so that they can be printed in the precise way that a Web author desires. Chapters 23 and 24

(“Font Family Properties” and “Text Family Properties”) looks at those new and expanded properties introduced under CSS2 to the font and text families of properties respectively. Chapter 25 (“Table Family Properties”) looks into this new class of properties that provide greater control over how tables should be displayed onscreen. Chapter 26 (“User Interface Properties”) examines another new class of properties designed to provide greater control over the display of such things as cursors and the outlines that surround such things as buttons or text fields in forms that denote a “focus” for user input. The final chapter, Chapter 27 (“Aural Style Sheet Properties”) explores a new class of properties designed to enable the Web author to determine how a Web page could be spoken aloud by a browser which speech capabilities.

There are a number of appendices that are designed as a quick reference for Web authors who need to look up how a particular CSS property functions, and whether or not it is supported in the major browsers. Appendix A is devoted to an alphabetical listing of all CSS1 properties, their values, plus sample code and whether or not it is supported in the major browsers. Similarly, Appendix B is an alphabetical listing of all CSS2 properties and their values plus sample code demonstrating how each property is to be used. Appendix C looks briefly at all of the ways colors can be implemented in CSS, as well as the ways various units of measure, percentage units and URLs can be used.

There is also a set of color-coded charts detailing those CSS1 properties and functions that have and have not been adopted within the various versions of the major browsers. These are designed as “spot” guides to tell a Web author whether or not a particular CSS1 property is “safe” to use. There are three such color charts: the first lists only the “safe” CSS1 properties, the second lists only the “unsafe” CSS1 properties, and the final chart displays *all* CSS1 properties, “safe” and “unsafe”.

## Conventions Used in This Book

Courier font is used to indicate HTML code, both in the listings and in the shorter code extracts that you’ll find included in the text. The same font is also used to indicate HTML tags and CSS property names (such as `BACKGROUND-COLOR`). In some cases, we show a code extract and then explain how to modify it to change its behavior. In this case, the code that is added or modified is shown in **courier font**.

Icons are used to call out material that is of significance and that the reader should be alerted to:



Core Note, Alert, Tip

**Note:** *This is information that deserves special attention, such as an interesting fact about the topic at hand, or that the reader may want to keep in mind.*

**Alert:** *This is information that, while useful, may cause unexpected results or serious frustration.*

**Tip:** *This is particularly useful information that will save the reader time, highlight a valuable programming tip, or offer specific advice on increasing productivity.*

## Further Information

CSS is an ever-evolving subject. At the time of writing, the processes were already set in motion to produce an official specification for CSS3, but were in too early a state to be covered effectively in this book. The definitive place to find information about CSS is the extensive material devoted to the official specifications that can be found at the World Wide Web Consortium's Web site, which can be found at:

<http://www.w3.org/Style/CSS>

It is also a good idea for any Web master to keep abreast of the latest developments in browser technology. The two acknowledged major players in the industry are Netscape and Microsoft. You can find out more about the latest CSS developments in Netscape Navigator by going to Netscape's home page at:

<http://home.netscape.com>

You can find out more information about how CSS can be used with Microsoft's Internet Explorer from Microsoft's Internet Explorer Web site at:

<http://www.microsoft.com/windows/ie>

(A minor caveat — to the annoyance of many, Microsoft is well known for frequently changing the URLs of major sections of their Web site. If this



URL does not work, try going to the Microsoft home page at <http://www.microsoft.com> and begin looking for links about Internet Explorer).

It is also worth pointing out where to find information about the two other browsers referenced in this book, both of which (at the time of writing) incorporate more CSS functionality than either Netscape Navigator or Internet Explorer. Both of these browsers would make for good additions to a Web master's "toolbox" when checking whether your CSS code functions the way you want it to. Opera is a browser that almost fully understands all CSS1 code, and more information about it can be found from Opera Software's Web site at:

<http://www.opera.com>

Mozilla is a project of Netscape Corporation to produce a browser incorporating open standards such as CSS. The latest version of their browser understands CSS1 and a major portion of CSS2. It is fully expected by many to lay the groundwork for the much-anticipated Netscape Navigator 5.0. More information about the Mozilla browser can be found at:

<http://www.mozilla.org>

For people seeking more information as to how CSS can be used, there is a single Usenet newsgroup wholly devoted to the subject that is well worth checking out:

<comp.infosystems.www.authoring.stylesheets>

## Feedback

No book is perfect and this one is unlikely to be an exception to that rule. Even though it has been through a long period of revision and technical review, there are, of course, errors still to be found and improvements still to be made. If you find an error or if there is something that you think might make the book more useful, we want to know about them. Please send comments and corrections to the following e-mail address:

[robertsk@wave.home.com](mailto:robertsk@wave.home.com)



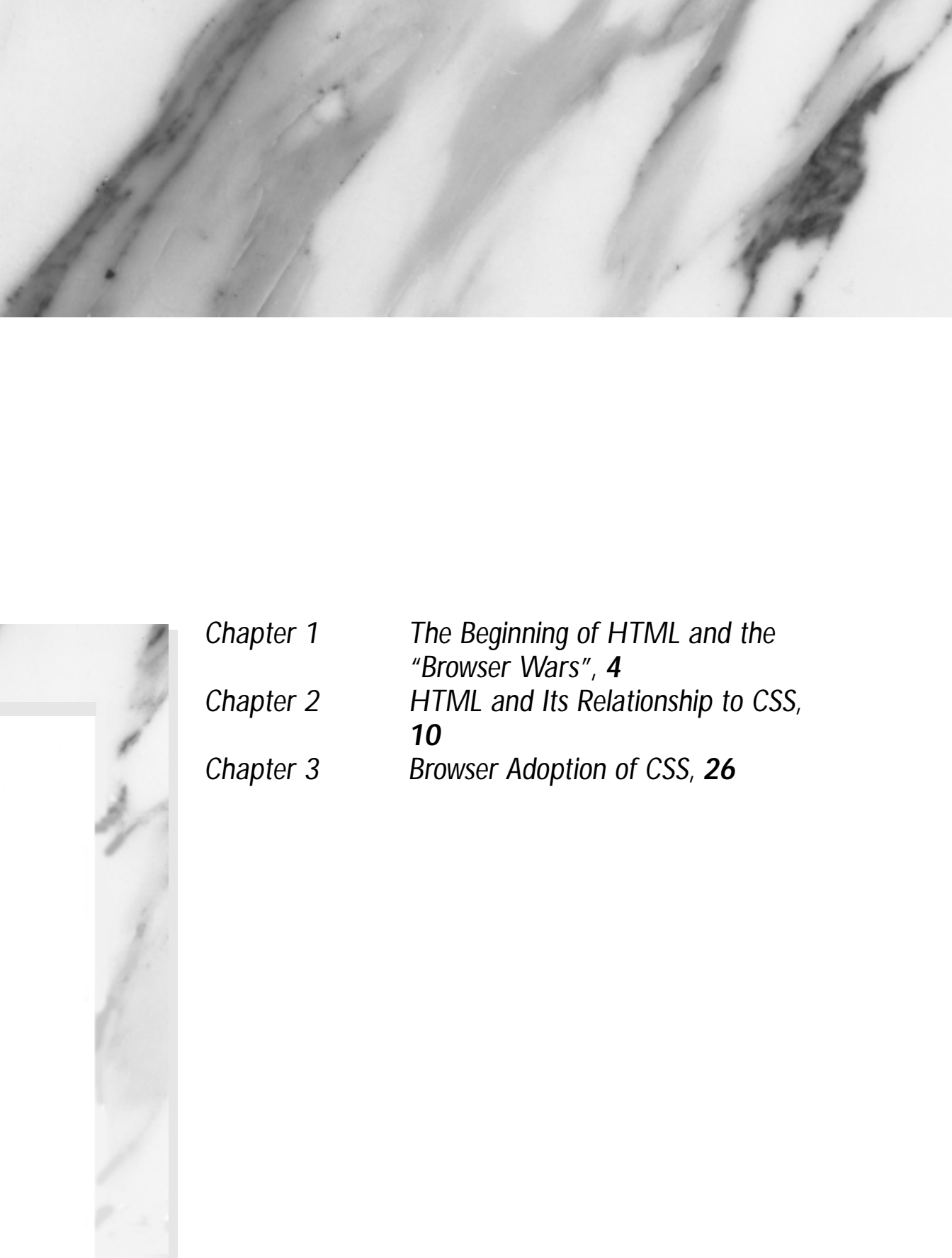
# CORE

## CSS

# *Part* 1



## THE ORIGINS OF CASCADING STYLE SHEETS



<i>Chapter 1</i>	<i>The Beginning of HTML and the “Browser Wars”, 4</i>
<i>Chapter 2</i>	<i>HTML and Its Relationship to CSS, <b>10</b></i>
<i>Chapter 3</i>	<i>Browser Adoption of CSS, <b>26</b></i>

# THE BEGINNING OF HTML AND THE “BROWSER WARS”

## Topics in this Chapter

- The Birth of the Web and HTML
- Increasing Browser Manufacture Competition
- Too Many Tags!
- W3C Devises the CSS Standard
- The Goals of CSS



# *Chapter*

# 1

**T**he World Wide Web came into being in 1991 as the result of a project by computer engineer Tim Berners-Lee at the European Centre for Nuclear Research (CERN) in Geneva, Switzerland. The goal was to get different computers to communicate with each other and display information across the Internet. Berners-Lee's work soon blossomed into the Web we know today.

The Web was founded on two major protocols: the Hypertext Transport Protocol ("HTTP"), which actually carries Web page information across the Internet, and the Hypertext Markup Language (better known as "HTML") specification, which determines the way a Web page is displayed in a browser. In the days of the original HTML 1.0 specification, things were pretty simple, with only a few HTML tags, primarily to handle such things as modest text formatting, hypertext links and images.

The Web grew quickly, however, especially after Web browsers broke out of the academic community in 1993 with the release of the Mosaic browser from the National Center for Supercomputing Applications. Mosaic was the first robust Web browser available for the popular Microsoft Windows and Macintosh operating systems. The initial version of Netscape Navigator, one of the first major commercial browsers, followed soon after, in 1994.

Netscape sought to make their Web browser stand out from the rest by creating new HTML tags for Web authors — tags that could only be viewed within Netscape's browser (or any other browser that adopted Netscape's new

tags). Many of these new tags provided desirable options for Web authors, giving them the ability to add such things as tables, frames and enhanced text formatting. This strategy helped Netscape to quickly grab a majority of the Web-browser market.

But browser development did not end with Netscape Navigator. While it may seem hard to believe now, back in the mid-1990s there were several dozen commercial Web browsers vying for market share, most of which have now fallen by the wayside. (In fact, the author worked on one of these, the now largely forgotten “Cyberjack,” produced by Delrina in 1995). One of Netscape’s main competitors was Spyglass, which had a license to sell a commercial version of the Mosaic browser. Microsoft’s Bill Gates realized that the Web was going to be important to the future of computing and communications, and so Microsoft decided to get into the browser market by purchasing the rights to Spyglass’s browser code. The initial version of Microsoft’s Internet Explorer was launched shortly thereafter.

Adopting a strategy similar to Netscape’s, Microsoft also began to incorporate new HTML tags into its browser — tags whose effects could only be seen when viewed within Internet Explorer. Thanks to the many features that these new tags brought to Web design (and the fact that Microsoft offered their browser for free), Internet Explorer soon took off in popularity.

This was the real start of the “browser wars”, where Netscape and Microsoft competed for “mind share” by creating new HTML tags in an attempt to sway Web authors to incorporate them into their Web pages. The result was an overabundance of new HTML tags for Web authors. Today, Netscape Navigator and Internet Explorer are the two browsers of choice for at least 90% of all Web users.

Unfortunately for the Web community, this conflict means that Internet Explorer and Netscape Navigator don’t agree about a lot of things. Both companies have introduced all sorts of HTML enhancements and technologies, some of which have been wildly successful (such as Netscape’s HTML table standard or Internet Explorer’s `<FONT>` tag) and some of which have failed to gain wide acceptance (such as Netscape’s layers standard or Internet Explorer’s Active Channels). Even when the two companies adopt the same types of technology within their browsers, there are often incompatibilities between the different versions (a point with which any JavaScript developer will readily agree).

In particular, the competing HTML tags have caused real headaches for Web authors. This crowd of new tags has been slowly subverting the idea that Web pages should be viewable by all browsers on all platforms — one of the factors that helped make the Web a popular medium in the first place.



These days, Web authors often have to jump through a number of hoops in order to make Web pages function and display properly in the two major browsers, even with the standard tags that are common to both. If a Web author decides to take advantage of a feature particular to one browser, the problems of compatibility are compounded. Some Web authors simply decide to craft their Web pages to look best in one browser, to the detriment to those using a different browser. A few Web authors go so far as to write code that will detect the type of browser being used, and then deliver a Web page optimized for that browser. While in some ways this might be seen as an ideal circumstance, few Web authors have the time or resources to do this properly. The majority simply design their Web site to look best in the two major browsers, but this restricts the Web author to code Web pages that often cater to the “lowest common denominator” in Web design and functionality.

Many of the non-standard HTML tags introduced in both Netscape Navigator and Internet Explorer make it hard or impossible for people using command-line browsers (such as Lynx) or people with disabilities (such as blind users who use browsers that read content aloud) to “view” a Web site.

As a result of these issues, most professional Web authors must test their Web pages against different versions of each of the major browsers and several of the less popular browsers in order to examine their Web pages for general compatibility.

Clearly, something had to be done before HTML became too awkward and unwieldy for use. This is where the World Wide Web Consortium enters the picture.

## The World Wide Web Consortium and Cascading Style Sheets

Tim Berners-Lee created the Web at CERN, and the initial standards for HTML 1.0 and HTML 2.0 were governed by them. But CERN’s main focus is as an international center for particle physics research, and with the growth in popularity of the Web, CERN abdicated its role as the standards-setting body for HTML in 1994. It passed the torch to a new body called the World Wide Web Consortium (better known simply as “W3C”). The W3C has convinced major software companies, including Netscape Communications, Microsoft, IBM, Novell, Sun Microsystems and many more, to become members of this standards body. This arrangement provides the software firms with lines of communication to other member firms and to a body recognized

as authoritative in the devising and setting a workable standard for HTML. The W3C is designed to be a neutral meeting ground, where competing companies can come together to contribute to and comply with future Web standards.

The W3C recognized the need to introduce some stability to HTML, and tried to consolidate existing HTML standards, first with the official HTML 3.2 specification, and later with the HTML 4.0 specification. These specifications adopted many of the HTML tags made popular in both Internet Explorer and Netscape Navigator, even though these tags often failed to conform to one of HTML's main guiding principles, that the markup should reflect the structure of a document rather than its physical layout.

HTML was not designed with sophisticated page layout properties in mind, but rather with the intention of displaying the logical structure of a document by using such things as headers, various heading levels, body text, and text-formatting tags that describe how the contents should be displayed (like the emphasis or teletype tags). This was done because there are many different varieties of computer out there, and they could not be expected to display content in exactly the same way. By defining logical constants to form a Web document, it was left up to the programmer to create software that would represent these structures in a fashion that would work for a particular computer or a specific operating system. In this way, a Web document was designed to be easily intelligible to anyone using virtually any computer.

But much of the drive by the browser manufacturers to provide Web authors with proprietary tags was spurred by the Web authors' desire for better control over the layout of elements on a Web page, similar to what already exists in desktop publishing programs. Over time, however, these extra tags have increased the overall complexity of Web pages, in addition to having produced Web pages that could be incompatible between Web browsers. While the HTML 3.2 and 4.0 specifications consolidated the existing state of HTML, action had to be taken to curb the development of new HTML tags. A flood of new tags from either browser manufacturer would only further impede the goal of creating an HTML standard that would enable Web pages that could be viewed by all browsers on any type of computer.

The W3C came up with the idea of Cascading Style Sheets (CSS) to head off the need by browser manufacturers to introduce even more HTML tags. CSS is an effort by the W3C to minimize the need for the introduction of new *physical* formatting tags (such as the font or table tags) by browser manufacturers — chiefly Microsoft and Netscape — by making such tags unnecessary. CSS is a compromise, providing the page layout features that Web authors want by adding CSS formatting elements to existing HTML tags.

CSS retains much of the logical structure of a Web page while delivering many of the page layout features in a way that is both easy to understand yet powerful in its effects.

Using CSS, it is possible to do the following on a Web page:

- specify the exact point size of text
- add indentations to text
- set margins within a Web page
- add new formatting elements to a Web page, such as borders around text
- use such measurement units as inches and centimeters to set precise sizes for text or images displayed on a Web page
- create a distinctive style for individual Web pages or sets of Web pages
- change the fundamental way a tag is supposed to be displayed onscreen
- set precisely where a background image is displayed, and whether it should be repeated across or down a Web page
- alter the spacing between letters, between words or lines of text, and much more

CSS's long list of features provides many of the page layout and typographical functions Web authors have been looking for, although it still provides less layout or typographic control than desktop publishing programs. CSS is really a mechanism for modifying how content is displayed on a Web page, and not a comprehensive layout tool. Despite this, CSS provides Web authors much finer control over where and how things appear on a Web page than can be achieved using regular HTML.


While CSS is a powerful tool designed to address the needs of Web authors who want greater control over how a Web page is displayed, it does not control the *behavior* of Web page elements. CSS does not affect fundamental functions such as hyperlinks or the way embedded objects behave within a Web page.

Development on the CSS specification is ongoing, and so in order to further enhance and extend the capabilities of Web browsers without the need for new HTML tags, the W3C introduced CSS2 in 1998. CSS2 expands upon the capabilities contained in the original CSS specification (now known as CSS level 1, or "CSS1") by adding greater control over such things as font control, how a Web page is to be displayed under different environments and even how a page should *sound*.

Make no mistake, CSS is definitely a valuable asset in Web page design.

# HTML AND ITS RELATIONSHIP TO CSS

## Topics in this Chapter

- 
- How CSS can be Incorporated into HTML Code
  - The <STYLE> Tag
  - The <SPAN> Tag
  - The <DIV> Tag
  - The <LINK> Tag
  - The STYLE Attribute
  - The CLASS Attribute
  - The ID attribute

# Chapter 2

CSS does not replace HTML code on a Web page; instead, it is added to existing HTML tags, augmenting them. You do not have to change the way you write HTML code for your Web pages, as CSS can simply sit on top of your existing code. In fact, once you get used to using CSS, you may find yourself using fewer HTML tags in favor of CSS. For example, the `<FONT>` tag is superceded by various CSS properties that give you far greater control over how text is displayed onscreen. And once you understand the basics of CSS, you'll find it easy to understand and to use.

CSS works by adding CSS properties to a specific HTML tag, which tells the browser how the tag should be modified in its display. A typical CSS statement (or “rule”) consists of two parts:

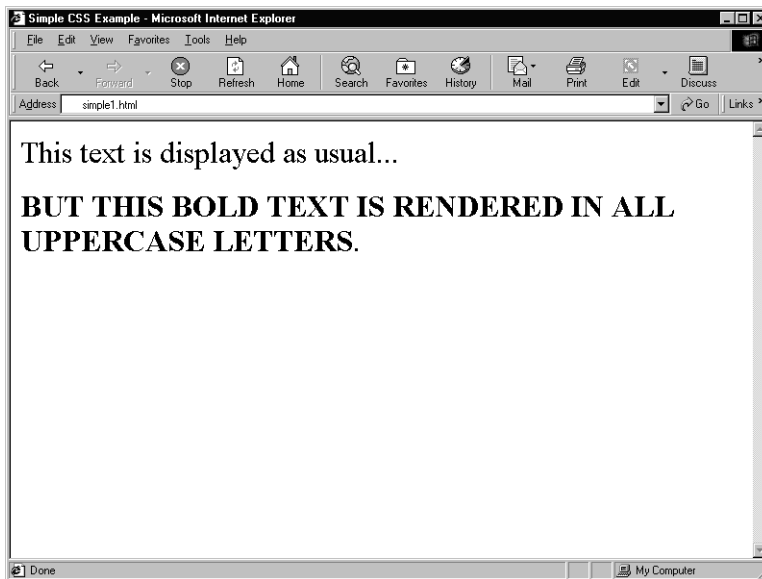
- the element to be changed, known as the “selector”
- a statement that describes how the “selector” should be displayed, known as the “declaration”

Here is a simple example of some HTML code that includes a few simple CSS rules:

**Listing 2-1 Simple CSS Example**

```
<HTML>
<HEAD>
<TITLE>Simple CSS Example</TITLE>
<STYLE>B {TEXT-TRANSFORM: UPPERCASE}</STYLE>
</HEAD>
<BODY STYLE="FONT-SIZE: X-LARGE">
This text is displayed as usual...<BR>
<B>but this bold text is rendered in all uppercase letters</B>.
</BODY>
</HTML>
```

The effects of this code can be seen in the following illustration:



**Figure 2-1** Web page displaying the effects of some simple CSS code.

There are two separate CSS modifications being made to the HTML tags that appear in the previous code example. In the first example, set between the `<STYLE>` tags near the top of the Web page, the selector is the bold tag (`<B>`), but without the usual brackets — see below) and the declaration assigns

to the `TEXT-TRANSFORM` property the value `"UPPERCASE"`: (`{TEXT-TRANSFORM: UPPERCASE}`). The second CSS example affects the `<BODY>` tag, which is the selector, modifying it with the declaration `"FONT-SIZE: X-LARGE"`. This applies the effect of an extra large font (the equivalent of `<FONT SIZE="6">`) to all of the text that appears on the Web page. This is a relatively simple example of what can be done with a couple of CSS elements, but the combined effects of some well-chosen CSS code can easily make for Web pages that stand out from the crowd.

## Adding Cascading Style Sheets to Web Pages

There are several ways in which CSS information can be specified within a Web page; we saw two in the previous code example. In this section, we look at the HTML tags and attributes that enable Web authors to add CSS code to their Web pages.

The different ways in which CSS information can be added to a Web page are as follows:

- It can be embedded within the header of a Web page
- It can be embedded within the body of a Web page (usually in certain sections or individual tags)
- It can be specified within a separate Web page

Each of these will be explored in this section.

There are four HTML tags and three HTML attributes that can be used to add CSS to a web page. The HTML tags are `<STYLE>`, `<SPAN>`, `<DIV>` and `<LINK>`. The HTML attributes are `STYLE`, `CLASS` and `ID`. The three tags are most commonly used to modify a section of a Web page or a particular set of HTML tags. The attributes, which can be added to almost every existing HTML tag, are designed to modify the display of the particular HTML tag with which it is associated.

As noted above, there is a special formatting style used to signify the CSS modifications to be associated with a particular HTML tag. The specified tag (the selector) is not surrounded by the usual angle brackets, but is instead left “naked” at the beginning of a line. The declaration is contained within “curly” brackets (`{` and `}`), and a simple space separates the selector and the declaration. Many beginners to CSS often make the simple mistake of adding

angle brackets to the selector, or using the wrong style of bracket type in the declaration. Browsers are finicky when it comes to CSS rules, and it is important to get this format right.

## The <STYLE> Tag

As we saw in the initial CSS example, it is possible to add CSS code within the header of a Web page. This is done by using the <STYLE> tag, which is used to set CSS properties to a Web tag that is applied to the whole of a Web document.

### <STYLE> ... </STYLE>

Description: Used to add Cascading Style-Sheet information to specific HTML tags.

Attributes:

#### ***DISABLED***

Indicates that the particular CSS function should not be enabled (i.e., displayed).

#### ***MEDIA - ALL / BRAILLE / PRINT / PROJECTION / SCREEN / SPEECH***

Specifies the type of medium for which the CSS definitions are designed.

#### ***TITLE - string***

Used to “name” a particular CSS style that a browser can be told to use.

#### ***TYPE - MIME type***

Specifies the MIME type of the content enclosed by the style tag.

Sample Code:

```
<HTML>
<HEAD>
<STYLE>
B {COLOR: BLUE}
```



```
</STYLE>
</HEAD>
<BODY>
<B>This bold text is blue.</B>
</BODY>
</HTML>
```

The <STYLE> tag is always contained within the header of a Web page, and unless overridden elsewhere in the Web page, its effects are applied globally to the selected HTML tags contained within it.

The <STYLE> tag has several attributes: DISABLED, MEDIA, TITLE and TYPE, all of which are optional — and few of which are ever actually used in practice, frankly. It is important to know the values of the attributes in those cases where it may be useful, however.

The DISABLED attribute — which is an unofficial attribute for <STYLE> that only supported within Internet Explorer version 4.0 and higher — indicates that the particular CSS function should not be “enabled” (in other words, that its effects should not be displayed). This is for use in circumstances where you may wish to disable the CSS formatting display in a particular place on the Web page. If you use this particular attribute, the content associated with the selected tag is still displayed, but it not formatted in the CSS style indicated.

The MEDIA attribute is used to specify the type of medium for which the CSS formatting is to be used. There are five distinct values: ALL, BRAILLE, PRINT, PROJECTION, SCREEN and SPEECH, reflecting the different types of media in which the Web page can be “viewed”. The default value is SCREEN. If a device exists to translate the Web page into a different medium, the device will use the CSS elements associated with the particular style used for it. This is an official HTML 4.0 specification attribute for the <STYLE> tag, and it is only supported within Internet Explorer version 4.0 and higher. Much of this is actually superceded by the more robust CSS2 specification, which deals in more detail with a wider variety of display media — see Part 3 for more detail.

The TITLE attribute is used to “name” a particular CSS style contained within a <STYLE> tag. The idea is to enable a Web author to create a gallery of different formatting styles, which the author can then selectively apply to individual Web pages within a Web site. This attribute is part of the HTML 4.0 specification, but is not currently implemented by either of the two major browsers.

The TYPE attribute simply specifies the MIME type contained by the <STYLE> tag. There are two possible values for this at the moment: “text/css”,

which applies directly to CSS, and “text/javascript”, which, not surprisingly, applies to JavaScript code. For the most part, the text/css value is simply assumed, and the two major browsers do not have any problems interpreting CSS code when this value is not present. This attribute is part of the HTML 4.0 specification, and is recognized in Internet Explorer and Netscape Navigator.

The <STYLE> tag can also be used with two of the near-universal HTML attributes: ID and CLASS. ID and CLASS can be added to virtually all HTML tags, and are used for applying more specific CSS formatting to a particular group of HTML tags or to a unique instance of an HTML tag appearing on a Web page. More detailed information about their use can be found in Chapter 5, “Implementation of Basic CSS Concepts”.

## The <SPAN> and <DIV> Tags

Other HTML tags can be used to apply CSS to a specific section of a Web page, enabling the Web author to set special CSS characteristics for that section, visually differentiating it from the rest of the Web page. This can be accomplished by using the <SPAN> and <DIV> tags.

### <SPAN> ... </SPAN>

Description: Designed to set CSS information for a specific inline section within a Web page.

Attributes:

**CHARSET = ISO-#**

Indicates the international character coding being used.

**HREF = URL**

Points to a Web page that is viewed when the content contained within the <SPAN> tag is activated.

**HREFLANG = string**

Specifies the base language of the resource indicated in the HREF attribute.

***MEDIA = ALL / BRAILLE / PRINT / PROJECTION /  
SCREEN / SPEECH***

Specifies the type of medium for which the CSS definitions are designed.

***REL = URL***

Indicates the URL of a subdocument.

***REV = URL***

Indicates the URL of a parent document.

***STYLE = CSS element; CSS element(s)***

Used to add CSS formatting to the HTML code.

***TARGET = frame name***

Specifies the name of the frame (if any) within which the content set by HREF should be viewed.

***TITLE - string***

Used to “name” a particular CSS style that a browser can be told to use.

***TYPE = MIME type***

Specifies the MIME type of the content enclosed by the style tag.

Sample Code:

```
<HTML>
<HEAD>
<TITLE>Simple Span Example</TITLE>
<STYLE>
P {COLOR: GREEN}
</STYLE>
</HEAD>
<BODY STYLE="FONT-SIZE: X-LARGE">
<P>
Look for the <SPAN STYLE="COLOR: RED">red text</
SPAN> that appears within the green text.
</P>
</BODY>
</HTML>
```

## **<DIV> ... </DIV>**

Description: Designed to set CSS information for a specific block elements within a Web page.

Attributes:

***ALIGN = LEFT / RIGHT / CENTER / JUSTIFY***

Specifies the alignment of the content.

***CHARSET = ISO-#***

Indicates the international character coding being used.

***CLEAR = NONE / LEFT / RIGHT / ALL***

Used whenever a “floating” element exists on a Web page, to break the line and start the text that appears after it in the next clear margin.

***HREF = URL***

Points to a Web page that is viewed when the content contained within the <DIV> tag is activated.

***HREFLANG = string***

Specifies the base language of the resource indicated in the HREF attribute.

***MEDIA - ALL / BRAILLE / PRINT / PROJECTION /  
SCREEN / SPEECH***

Specifies the type of medium for which the CSS definitions are designed.

***NOWRAP***

Suppresses word wrap.

***REL = URL***

Indicates the URL of a subdocument.

***REV = URL***

Indicates the URL of a parent document.

***STYLE = CSS element; CSS element(s)***

Used to add CSS formatting to the HTML code.

***TARGET* = frame name**

Specifies the name of the frame (if any) within which the content set by HREF should be viewed.

***TITLE* - string**

Used to “name” a particular CSS style that a browser can be told to use.

***TYPE* = MIME type**

Specifies the MIME type of the content enclosed by the style tag.

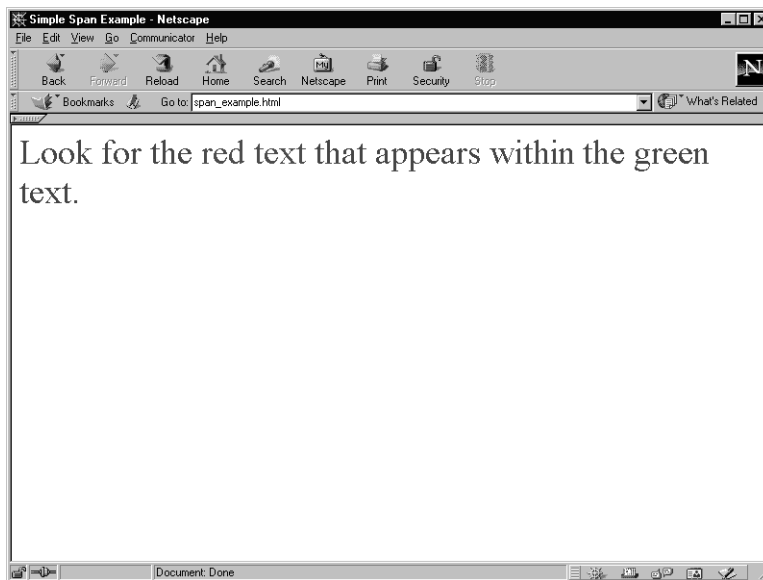
Sample Code:

```
<HTML>
<HEAD>
<TITLE>Simple Div Example</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: X-LARGE">
<P>
This text is displayed using the default display
values.
<DIV STYLE="FONT-FAMILY: COURIER; COLOR: LIME">
<H1>A Lime-Colored, Courier-Font Header</H1>
</DIV>
Back to the default values.
</BODY>
</HTML>
```

The <SPAN> tag is designed to temporarily override any existing CSS information that may have already been specified, and is meant to be used as an inline element. The <DIV> tag works in the same manner, but is supposed to be applied to block level elements.

Inline elements refer to those common formatting HTML tags that are usually contained within a line of text, such as <B> or <I>. These elements do not necessarily begin or end a separate line onscreen. Tags such as <H1> or <P> do, and they are known as block elements. <SPAN> and <DIV> work in the same way, but are simply meant to be applied to different types or groups of HTML tags. As you will see later in the book, some CSS elements can only be used with inline or block-level Web elements (i.e., HTML tags).

The <SPAN> and <DIV> tags both have only one required attribute: STYLE. The STYLE attribute is to be used to further refine the CSS element displayed. The STYLE attribute is the third near-universal attribute, along with ID and CLASS, and is probably the most commonly used of the three.



**Figure 2-2** The effects of the sample code for the simple `<SPAN>` and `<DIV>` examples displayed.

Using `STYLE`, you can directly set the CSS rule to apply to a specific HTML tag (or with the aid of the `<SPAN>` and `<DIV>` tags, to a group of such tags). Its structure has already been explained, but you will note one important addition in the sample code provided for the `<DIV>` tag: how to add more than one CSS attribute to an HTML tag. This can be accomplished by adding a semi-colon (“;”) between two or more CSS statements, forming a compound CSS rule that is associated with the specific HTML tag.

The `<SPAN>` and `<DIV>` tags share many additional attributes, most of them added beginning with the official HTML 4.0 specification. None of these other attributes are as yet implemented in current browsers, but it is important to know what they are intended to do pending their adoption with the major browsers.

The `CHARSET` attribute is used to specify the international character coding used for the hyperlink. The default value is the standard Latin-1 ISO standard, which is used by most European languages — including English. The `TYPE` attribute is used to specify the MIME type of the file type to which the hypertext anchor points, and works in exactly the same way as it does with the `<STYLE>` tag. Ditto the `MEDIA` attribute and its associated values.

The HREFLANG attribute is designed to set the base language of the resource to which it is being linked. The HREF attribute enables the Web author to make the content contained within a set of <SPAN> or <DIV> tags act as a hypertext link. The associated TARGET attribute specifies the name of the frame within which the link should be displayed, if the content is contained within a framed Web page.

The TITLE attribute is used to “name” a particular CSS style contained within a <SPAN> or <DIV> tag. Using the name the Web author has assigned to the tag, the Web author can create a number of different formatting styles, and the author can then apply different styles to individual sections of the Web page. This attribute is part of the HTML 4.0 specification, but is not currently implemented by either of the two major browsers.

The <DIV> tag has three additional attributes not used by <SPAN>: ALIGN, CLEAR and NOWRAP. The ALIGN attribute specifies the alignment of the content it contains, while CLEAR is used with any “floating” elements (such as an image) that it might contain, and NOWRAP suppresses the normal word wrap characteristics. It should be noted that these attributes, with the possible exception of ALIGN, are rarely used in practice.

The code examples for both the <SPAN> and <DIV> also serve to introduce the meaning behind the “Cascading” part of the term “Cascading Style-Sheets”. Quite simply, more specific CSS rules tend to override global ones. The designers of CSS realized that there would be situations in which a Web author would want to do just that. When this situation occurs, the browser “cascades” from the more general to the more specific rule. This cascading order is defined by the following rules:

Situation: A browser is presented with a number of CSS statements, some of which conflict with each other.

1. If a statement is applied to a “parent” HTML tag (such as <BODY>) and tags that are its “children” have explicit CSS statements of their own, the CSS statement that applies to the parent rules.
2. The CSS statements are then sorted by explicit weight, and any marked ! IMPORTANT (covered in Chapter 6) take precedence over the others.
3. Any style sheets information that may be imported through the <LINK> tag will be overruled any CSS statements contained within the Web page.
4. More specific CSS statements overrule more general ones.

5. If two or more CSS statements appear that have the same weight, the one specified last wins.

This concept is explored in more depth in Chapter 6, “The Cascade”.

## The <LINK> Tag

There is a third way CSS information can be added to a Web page: by specifying CSS elements in a completely different Web page, which are then linked to the original page. By specifying the display and layout attributes in a separate Web page, you can customize the style for a set of Web pages that use ordinary HTML code. To do this, you create a Web page containing nothing but CSS information. Then, you can “link” another Web page to the CSS page, by using the <LINK> tag in the header of the target Web page. This is potentially the most appealing for Web authors who manage large Web sites.

### <LINK>

Description: The <LINK> tag indicates where a Web document fits into the hierarchy of the Web site.

Attributes:

#### **REL = URL**

Indicates the URL of a subdocument.

#### **REV = URL**

Indicates the URL of a parent document

#### **TYPE - MIME type**

Specifies the MIME type of the content enclosed by the style tag.

Sample Code:

```
<HTML>
<HEAD>
<TITLE>Link Example</TITLE>
<LINK REL=STYLESHEET HREF="formatting.css"
TYPE="text/css">
```



```
</HEAD>
<BODY>
<H1>Link Example</H1>
<P>
This Web page will appear very different depending
on the content of the <B>formatting.css</B> page
that it is linked to. This demonstrates the power
behind the use of the <I>LINK</I> tag to easily
make global changes to a number of different Web
pages at once.
</BODY>
</HTML>
```

The sample code provided in the description of the <LINK> tag shows a fairly ordinary Web page that contains a level-one header, a paragraph tag, a bold tag and an italics tag, all contained within a standard body tag. In the header of this Web page, there is a <LINK> tag that references a separate file called `formatting.css`. The `formatting.css` file is a simple text file that contains nothing but CSS formatting information in much the same format as used by the <STYLE> tag, as in the following two example files:

#### Listing 2-2 `formatting.css` #1

```
<!-- formatting.css #1 -->
BODY {FONT-SIZE: X-LARGE}
B {FONT-SIZE: XX-LARGE; COLOR: RED}
I {FONT-SIZE: LARGE; COLOR: BLUE}
P {COLOR: GRAY}
H1 {COLOR: WHITE; BACKGROUND: BLACK}
```

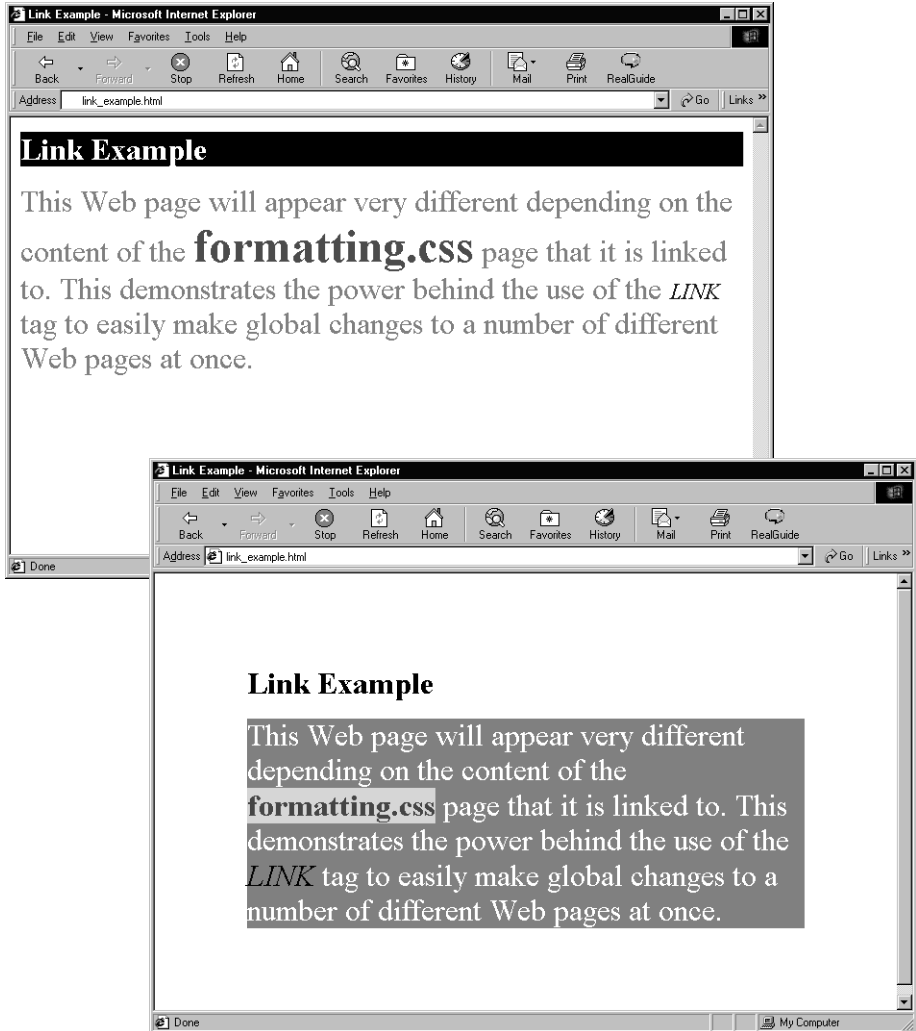
---

#### Listing 2-3 `formatting.css` #2

```
<!-- formatting.css #2 -->
BODY {FONT-SIZE: X-LARGE; MARGIN: 1 IN}
B {COLOR: GREEN; BACKGROUND: YELLOW}
I {COLOR: BLACK}
P {BACKGROUND: GRAY; COLOR: WHITE}
H1 {BORDER-COLOR: RED}
```

---

You can see how each of these two files are interpreted by the browser in the following two illustrations:



**Figure 2-3** The effects of two separate `formatting.css` files on the same Web page.

This is a mere taste of what can be done using CSS within Web pages. The examples shown here have employed only a small subset of the CSS elements available, but nevertheless demonstrate the power behind CSS.

Unfortunately, not all CSS elements are universally adopted within the two major browsers, which means that Web authors have to be aware of what CSS code should and should not be used.

# BROWSER ADOPTION OF CSS

## Topics in this Chapter

- Internet Explorer 3.x
- Internet Explorer 4.x
- Internet Explorer 4.5 (Macintosh) and 5.x
- Opera 3.6
- Netscape Navigator 4.X



# Chapter 3

While both Netscape and Microsoft have agreed to implement CSS in their respective browsers, neither have done so fully as yet. This means that Web authors who want to use CSS have to know which CSS elements they can and can't use, as well as the significant differences in the way adopted CSS elements behave in both browsers. Much of the information contained in the rest of this book deals with the ways CSS is currently implemented in the two major browsers, looking at the way particular CSS elements work.

It must be stressed that at the time of writing (November 1999), neither Internet Explorer nor Netscape Navigator apply the whole of the CSS1 (nor CSS2 for that matter) within their respective browsers. Both use only a subset of the complete specification as laid out by the W3C. Since both companies are members of the W3C, however, they will adopt the full set of CSS elements over time — at least that is the hope. CSS1 (and CSS2) will only be useful to Web authors if they are widely adopted and fully implemented by browser manufacturers. If they are not, CSS could easily disappear. There is a precedent for this: the official HTML 3.0 specification was widely ignored by browser manufacturers.

The piecemeal way in which Netscape Navigator and Internet Explorer have adopted CSS elements is holding CSS back from wider adoption within the Web authoring community. In many cases, CSS elements that are supported in Internet Explorer are not supported in Netscape Navigator, or they

are not supported in either browser. Sometimes only certain values of a CSS element are supported, or they only work when associated with certain HTML tags. There are even cases where a CSS element has been adopted for use within a beta (or “preview”) version of a browser and then later dropped in the next beta release — presumably an oversight, but not exactly something to inspire confidence in CSS for a Web author. There are also significant differences in the way CSS code works in the same browser operating under different operating systems.

In general, if a certain feature is already supported within one browser, you can most likely use a CSS element to set it, too. For example, the `TEXT-DECORATION` element has a value called `BLINK` that flashes the text on and off repeatedly. It is supported within Netscape Navigator — where you could use the `<BLINK>` tag to do much the same sort of thing — but not in Internet Explorer, which has never supported `<BLINK>`. On the other hand, there is a feature seen only in Internet Explorer called “watermarking” that allows the Web author to create a background image that remains fixed upon the page even when a user scrolls up or down. There is a CSS value (“`BACKGROUND-ATTACHMENT: FIXED`”) that does the same thing. Not surprisingly, this CSS element is supported within Internet Explorer, but not within Netscape Navigator.

Knowing which CSS elements to use and which not to use is half the battle in constructing workable CSS code. It can only be hoped that over time all CSS elements will be supported equally well within the two major browsers. As time goes by and more Web authors become familiar with the details of CSS (and as more Web browsers fully implement and are able to display Web pages that use CSS), the demand and need for new HTML tags will be significantly diminished.

Depending on how things turn out, CSS will either be a triumph or a resounding failure for the W3C, ultimately determining the evolution and development of HTML and of the relevancy of the W3C itself.

## Browser Differences

As a Web author, it is worth your while to get to know the major differences and the quirks that determine how well CSS is adopted within a particular browser. An understanding of the strengths and weaknesses of a given browser will help you to choose which CSS elements will be most effective in your Web pages.

### *Internet Explorer 3.x*

IE 3.x was the first major browser to implement CSS, although only a rudimentary fashion. It did not implement many CSS properties, and those that were implemented were rarely implemented fully. Very few people use Internet Explorer 3.x today, so it is probably not worthwhile to try and design pages that would look good in it. This is just as well, since designing Web pages that use CSS for Internet Explorer 3.x is pretty much an exercise in frustration, anyway. In short, don't plan on designing a CSS-oriented Web site that has to be "safe" for whatever Internet Explorer 3.x users are still out there.

For this reason, in the summary listing of "safe" and "unsafe" elements provided at the end of this book, Internet Explorer 3.x is not included, as it would render too many CSS elements as "unsafe".

### *Internet Explorer 4.x*

Microsoft greatly extended CSS compatibility within the initial 4.x version of Internet Explorer. This version of Internet Explorer was the first to show that a major browser vendor was serious about implementing CSS.

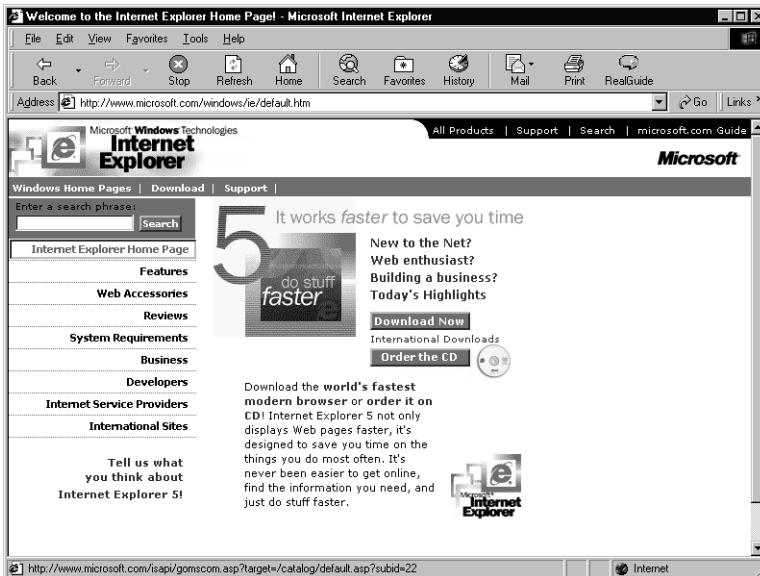
This version of Internet Explorer was not without its drawbacks, however. While it supported much of the background, border, font, margins, lists and padding CSS element families at least partially, it could not support a number of other elements, such as `DISPLAY` or `CLEAR/FLOAT` in the Macintosh version at all. There were also some minor variances worth noting in the ways the browser implemented the same CSS elements across different operating systems. Problems implementing the `TEXT-ALIGN` property have been noted, as have more general problems when implementing some basic CSS functions, such as containment, inheritance and dealing with certain length units. Although superseded by version 5.x, at the time of writing there are

still enough Internet Explorer 4.x users out there for it to be beneficial to keep its limitations in mind.

## Internet Explorer 4.5 (Macintosh) and 5.x

At the time of writing, Internet Explorer 5.x is the best version of a *major browser* that implements CSS. But “best” is not “perfect”. Many people had hoped that this version would fully support CSS, but, unhappily, it does not.

Microsoft first made major improvements to the Macintosh 4.5 release of Internet Explorer, adding full or improved support for several CSS elements that were previously unsupported or only partially supported. Upon this framework lay the basis of much of the work that was subsequently done on the Windows and UNIX versions of Internet Explorer 5.0. For the most part, Internet Explorer 5.0 handles CSS code best, and a majority of the examples illustrated in this book use this version of the browser. It is worth noting, however, that the Macintosh 4.5 version is actually better in a few cases, such as its more comprehensive handling of the border and background family of elements.

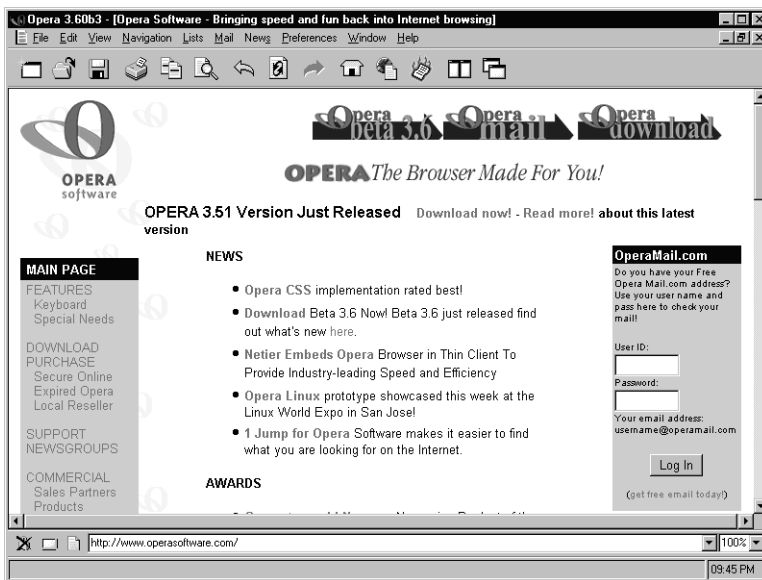


**Figure 3-1** The Internet Explorer 5.x Web browser displayed.



## Opera 3.6

Though not a “major” browser, Opera deserves mention because of its extensive support for CSS. At present, there are only a handful of CSS properties it does not fully support, and there are few cases where it doesn’t at least partially support some CSS property. If you are planning to use CSS code extensively in your Web pages, it is a good idea to keep a “reference” copy of the latest version of the Opera browser. You can then see how your CSS code should work, and then compare that to what you see in Internet Explorer and Netscape Navigator.



**Figure 3-2** The Opera Web browser displayed.

## Netscape Navigator 4.X

At the time of writing, Netscape Navigator is the least capable of the major browsers when it comes to rendering CSS properties. CSS was first implemented in the 4.0 version of the browser, but there has been little to no improvement in the 4.5 and 4.6 versions that have appeared since then. It is weak in its support of some background, border and font families of elements, in several cases not supporting them at all.

The CSS implementation of Netscape Navigator is at least consistent between the different operating systems — if your CSS code works in the Windows version of Netscape Navigator, you can be sure that it also works in the Macintosh and UNIX versions of the browser.

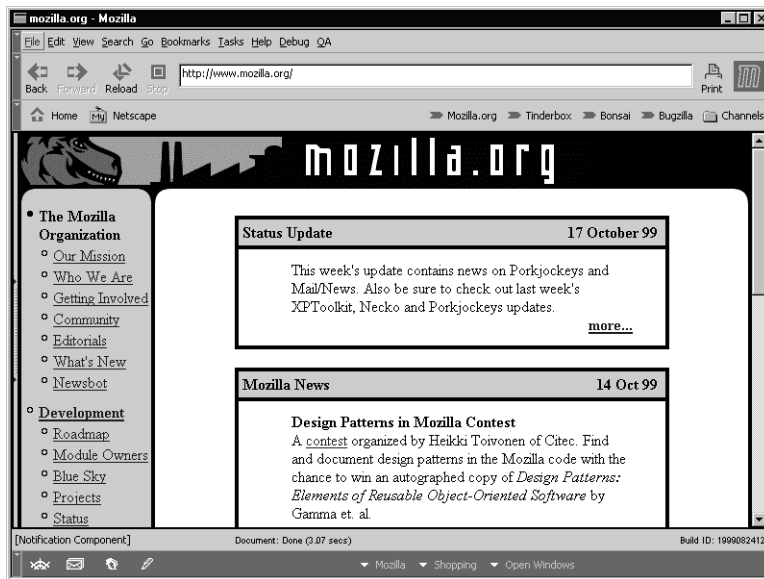


**Figure 3-3** The Netscape Navigator 4.6 browser displayed.

Netscape Navigator may not remain at the bottom of the CSS compatibility heap for long. An offshoot group affiliated with Netscape, called Mozilla.org, is working on a version of the browser that complies fully with many Web standards, including CSS1. Mozilla.org produces what can best be described as an “interim” version of a browser, whose component parts will one day most likely be incorporated in Netscape Navigator 5.0, which has a tentative (and arguably ambitious) release date currently slated for the first quarter of 2000. At the time of writing, the CSS1 component for the “Milestone 10” test build of the program was considered 98% complete, with only some outstanding bugs to be fixed. Furthermore, it seems as though there will be at least some support for a number of CSS2 properties as well, including positioning elements, extended table resources and the additional selectors it offers. Since Mozilla.org has not committed to supporting these CSS2 features in the final shipping version of Netscape Navigator 5.0, at this point in time it cannot be said whether or not the features currently supported in

the most recent build may not be stripped away in the final shipping product, for reasons of file size or feature completeness. However, these developments aimed at achieving full CSS compatibility bode well for the future of CSS.

The Mozilla browser is not included in the “safe” and “unsafe” CSS listing at the back of the book because it is a preview version of the browser, and features can change from one preview version to another.



**Figure 3-4** The Mozilla browser (Build M10) displayed.

If Netscape Navigator 5.0 is finally released having reached its goal to fully support CSS1 elements, Web authors can only hope that this will spur Microsoft into following suit in the next major revision of its Internet Explorer browser.

# *Part* 2



**CSS1**

<i>Chapter 4</i>	<i>Overview, <b>36</b></i>
<i>Chapter 5</i>	<i>Implementation of Basic CSS Concepts, <b>42</b></i>
<i>Chapter 6</i>	<i>The Cascade, <b>60</b></i>
<i>Chapter 7</i>	<i>CSS Units, <b>70</b></i>
<i>Chapter 8</i>	<i>Pseudo-Classes and Pseudo-Elements, <b>84</b></i>
<i>Chapter 9</i>	<i>Font Properties, <b>98</b></i>
<i>Chapter 10</i>	<i>The Color and Background Family of Properties, <b>126</b></i>
<i>Chapter 11</i>	<i>The Text Family of Properties, <b>152</b></i>
<i>Chapter 12</i>	<i>The Box Family of Properties, <b>178</b></i>
<i>Chapter 13</i>	<i>The Classification Family of Properties, <b>260</b></i>

# OVERVIEW

## Topics in this Chapter

- CSS1 Overview
- Chapter Descriptions



# Chapter 4

This part provides a detailed look at each of the Cascading Style Sheet properties belonging to the CSS1 specification. You will find extensive sample codes showing how each property works, plus screen shots displaying how the code does (or does not) function in various browsers. Following the description of how each CSS property is supposed to work, browser compatibility is addressed, providing Web authors with a clear idea as to which CSS properties are “safe” or “unsafe” to use.

Anybody looking to learn how CSS can actually be used will find this section particularly helpful, as it reflects the current “state of the art” of CSS implementation in the major browsers at the time of writing (summer 1999). It is also designed to serve as a handy, detailed reference for Web authors trying to determine which CSS properties to add to their Web pages, and which to avoid due to problems with browser compatibility.

This part of the book looks at all of the following “families” of CSS implementations and properties:

- Implementation of Basic CSS Concepts
- The Cascade
- CSS Units
- Pseudo-Classes and Pseudo-Elements
- Font Properties
- The Color and Background Family of Properties

- The Text Family of Properties
- The Box Family of Properties
- The Classification Family of Properties

## Chapter Descriptions

This book looks first at the fundamental concepts of CSS (such as cascading, inheritance and units of measure) that must be implemented for a browser to even begin to be thought of as “CSS compliant”. These are covered in the chapters “Implementation of Basic CSS Concepts”, “The Cascade” and “CSS Units”. The rest of the chapters in this part are devoted to a detailed examination of all of the CSS properties: “Pseudo-Classes and Pseudo-Elements”, “Font Properties”, “The Color and Background Family of Properties”, “The Text Family of Properties”, “The Box Family of Properties” and “The Classification Family of Properties”.

### *Implementation of Basic CSS Concepts*

This chapter looks at the fundamental functions that underpin CSS. These functions help to define how CSS works, and determine to a large extent whether or not a given Web browser is capable of being CSS compliant — browsers that do not at least partially implement these functions are incapable of dealing with CSS code. Such topics as containment in HTML, grouping CSS properties and values together, inheritance and other basic CSS fundamentals will be covered.

### *The Cascade*

The first “C” in “CSS” stands for “Cascade”, a key idea that helps determine the “weight” of one CSS property over another. Under the right context, one CSS property should overrule the actions of another. The chapter titled “The Cascade” looks at this set of crucial features and explains the circumstances as to which CSS rule “rules”. Many of the design features covered in this chapter are also essential to functioning CSS in modern browsers.



## ***CSS Units***

One of the chief advantages of CSS is that it gives Web authors greater control in setting lengths of measure (for such things as margins and font sizes) to elements contained on a Web page. The chapter called “CSS Units” takes a look at all of the different ways Web authors can set such things as color, units of measure, percentage values and including URLs to CSS code. This chapter also covers features that are also crucial to fully functioning CSS in modern browsers — many of the CSS properties covered later in this part rely heavily upon the units presented in this chapter.

## ***Pseudo-Classes and Pseudo-Elements***

Pseudo-classes and pseudo-elements are a number of CSS properties designed to change the way hypertext anchors are displayed and produce specific typographic effects for the first letter and first line of text in a paragraph of text. This chapter also looks at how you can combine these elements effectively within a single Web page, and begins a more concentrated look into the various CSS properties that Web authors can actively set.

## ***Font Properties***

The chapter titled “Font Properties” takes a look at the properties used to alter the way fonts are displayed. This includes such things as specifying the “family” to which a font belongs (such as “san-serif” or “fantasy”), the size of the on-screen font, whether a font is displayed normally or in italics, its boldness value and more.

## ***The Color and Background Family of Properties***

This chapter takes an in-depth look at the various CSS properties used to set color values and determine the placement of background images on Web pages. The `COLOR` property in particular is a crucial value in CSS, as it is an element most often used in conjunction with other CSS properties, such as font or text properties. The family of background properties includes values that enable a Web author to specify exactly where a background image should appear on a Web page, and how it is “tiled” across or down the page. These values contain many features Web authors will find simply impossible to do using regular HTML.

## ***The Text Family of Properties***

This chapter examines all of the properties dealing with the way text is displayed on a Web page. These properties differ from the font family of properties in that they control the look and placement of text as it appears on screen. These properties include values that will let you create underlined text, text that blinks, indented text and more.

## ***The Box Family of Properties***

This chapter covers the single largest family of properties in the CSS pantheon. There are several different types of properties in this family, governing how and where such things as margins, borders, padding and the placement of block level elements appear on a Web page. The name “box” applies to the way these elements surround and modify block level elements (such as paragraphs or headers) that are by their nature rectangular in shape, as they each take up an entire horizontal line of space. Since these properties are meant to be applied to both the inside and the outside of these block level elements, they can be made to look like a series of concentric “boxes” around the block level element.

## ***The Classification Family of Properties***


The classification family of properties deals with CSS properties designed to change the fundamental way certain elements on a Web page are categorized, which can lead to a change in the way that they are displayed on screen. Using some of these elements, you can change a block level element to appear as if it is an inline element, or change the way in which list items are displayed — including tacking on images as list markers.

For a more concise description of the CSS1 properties listed here, see Appendix A, “CSS1 Property, Pseudo-Element, Pseudo-Class, and Concepts Reference”, which lists all of the CSS1 properties in alphabetical order. Appendix C, “Colors, Units of Measure, Percentage Units and URLs”, provides a quick listing of the types of values that can be added to CSS properties that can take color values, length measures, percentages or hypertext links.



# IMPLEMENTATION OF BASIC CSS CONCEPTS

## Topics in this Chapter

- 
- Containment in HTML
  - Grouping
  - Inheritance
  - Class as Selector
  - ID as Selector
  - Contextual Selectors
  - Comments

# *Chapter*

# 5

This chapter will explain how some of the more basic concepts behind CSS are implemented. In order for browsers to understand and implement CSS formatting, they must be aware of the ways in which CSS properties can be combined within the HTML contained on a Web page. Recognition of these basic CSS concepts within Web browsers is crucial to the implementation of CSS within a browser. Thankfully, all of these features are at least partially implemented in the major browsers.

This chapter focuses on the following CSS concepts:

- Containment in HTML
- Grouping
- Inheritance
- Class as Selector
- ID as Selector
- Contextual Selectors
- Comments.

## Grouping

Grouping is a basic CSS concept designed to make things simpler for Web authors by allowing them to combine several CSS rules in a single statement. This can be very handy, and can considerably shorten the length of your CSS code as you customize exactly how you want different HTML tags to appear on a Web page. Grouping allows you to add the same basic formatting rules to common groups of formatting elements, such as headers or paragraphs.

### Grouping

Description: Grouping is designed to shorten the overall length of style sheet code by allowing the Web author to add several formatting properties together in a single, compact statement.

Sample code:

```
<STYLE>
H1, H2, H3 {FONT-FAMILY: HELVETICA;}
P {FONT-SIZE: 12pt; FONT-FAMILY: COURIER;
  FONT-VARIANT: NORMAL; COLOR: NAVY;}
</STYLE>
```

<b>Table 5-1 Grouping Support in Major Browsers</b>
---

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>NN 4.x</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Safe

Quite often, Web authors may only want to add a single CSS rule to a given HTML tag, as in the following code example:

```
<STYLE>
H1 {FONT-FAMILY: HELVETICA;}
</STYLE>
```

But what if you wanted to format all of your major headers using the Helvetica font? You could write it all out, assigning the CSS rule to each header you intend to use on the Web page, or you could simply use the CSS

grouping concept to add the same formatting to all of those headers, as shown in the following code fragment:

```
<STYLE>
H1, H2, H3 {FONT-FAMILY: HELVETICA;}
</STYLE>
```

This code ensures that the Helvetica font will be used for all of the <H1>, <H2> and <H3> headers on the Web page.

The grouping concept also means that you can add several CSS properties to a single HTML tag, customizing it so that it is displayed in exactly the way you want. For example, the following CSS code example adds several CSS declarations to a single HTML tag:

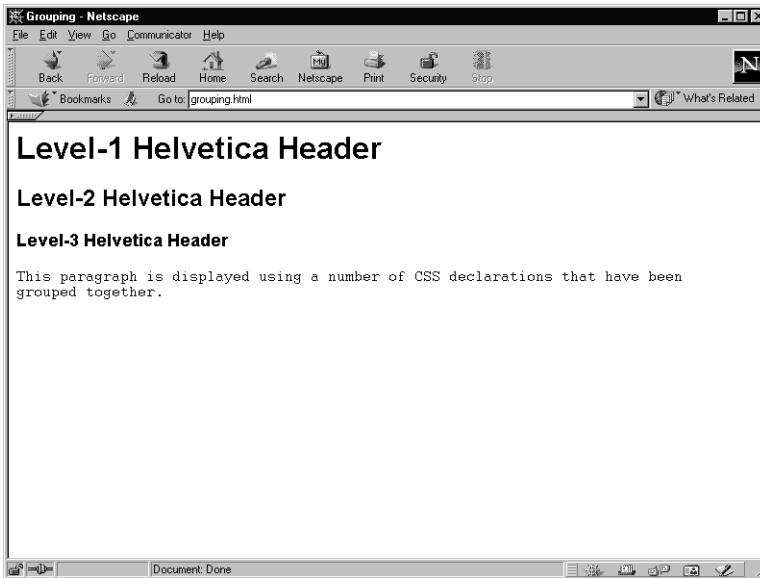
```
<P STYLE="FONT-SIZE: 12pt; FONT-FAMILY: COURIER;
FONT-VARIANT: NORMAL; COLOR: NAVY;">
This paragraph is displayed using a number of CSS
declarations that have been grouped together.
</P>
```

Grouping allows your code to be more compact, while at the same time making it easier to read (and understand). It also arguably encourages better layout for CSS properties, as it is a natural thing to want to format similar elements (such as headers) using common formatting elements, such as the font they should all display. By grouping CSS properties in this manner, you can easily set a genuine “style sheet” for a page or set of pages that is short, concise and fosters good (rather than wild) layout. Figure 5–1 shows the effects of the CSS code examples from this section — notice how the grouping of the Helvetica font applied to the header code presents an orderly progression of headers that share the same look.

Grouping is fully supported in all of the more recent versions of Internet Explorer and Netscape Navigator.

## Inheritance

One of the chief concepts behind CSS is that CSS properties should be able to take on, or “inherit”, the display properties of dominant HTML tags. For example, if you set specific properties for the <BODY> tag, all of the HTML elements that fall under this tag (which includes all displayable Web elements) should take on these same display properties. In other words, the “child” elements (tags such as <P>, <I>, <B> etc.) should take on the display properties set by the “parent” (in this case, <BODY>). In effect, all of the child



**Figure 5-1** A Web page displaying grouped properties.

elements will take on the display properties set to the parent; further changes made to any child element are additive.

## Inheritance

Description: A browser should allow “child” elements to take on the properties of a “parent” element.

Sample Code:

```
<P STYLE="FONT-SIZE: 14pt; FONT-STYLE: ITALIC;
COLOR: BLUE;">
```

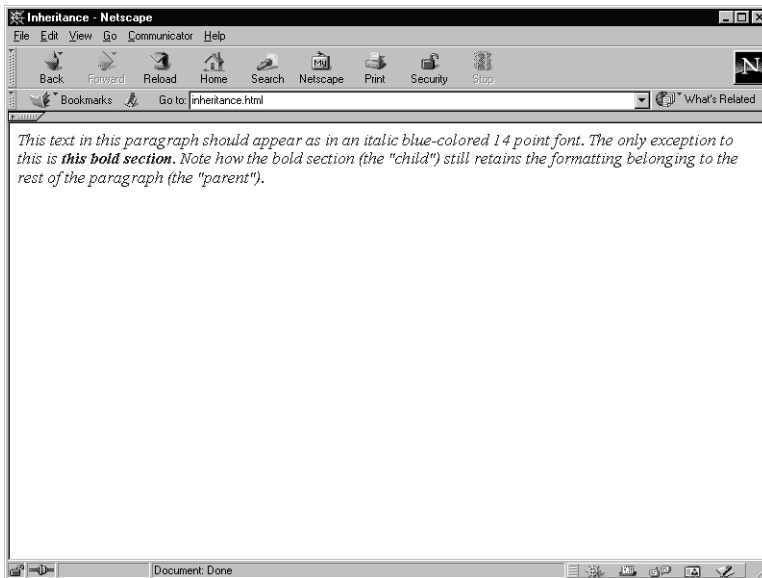
This text in this paragraph should appear as in an italic blue-colored 14 point font. The only exception to this is **this bold section**. Note how the bold section (the "child") still retains the formatting belonging to the rest of the paragraph (the "parent").

```
</P>
```



<b>Table 5-2 Inheritance Support in Major Browsers</b>
--

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> (Mac)	<i>IE 4.5</i> (Mac)	(Win. & UNIX)	<i>NN 4.x</i>	(Mac & UNIX)	<i>Opera</i> 3.6
Unsafe	Safe	Partial	Safe	Safe	Safe	Safe	Safe



**Figure 5-2** A Web page displaying inherited properties.

If you take a look at Figure 5-2, which uses the code from the definition section, you will note that all of the text in the paragraph is set to a large, blue and italic text. The section that appears in bold (which is a “child” element to the paragraph it is contained within) retains all of these formatting elements, and adds it own.

This basic CSS concept is present in all of the current versions of Internet Explorer and Netscape Navigator.

## CLASS and ID as Selectors

In preparation for the full implementation of CSS, the forward-looking HTML 4.0 specification added two new attributes: `CLASS` and `ID`. These new attributes can be applied universally to any HTML tag meant to be displayed within the body of a Web page. This makes sense, since CSS is designed to alter the display of visible elements on a Web page, and by definition that means HTML tags that are displayed within the body of a Web page, and not, say, in the header of a Web page.

The idea behind `CLASS` and `ID` is that they identify the HTML tags with which they are associated as “targets” for CSS declarations that have previously been defined within the header of a Web page. `CLASS` and `ID` differ in that `CLASS` is intended for general formatting across many elements on a Web page, whereas `ID` is meant for specific formatting to a single instance.

### *CLASS as Selector*

Description: `CLASS` is an HTML attribute that enables Web authors to “name” a set of formatting properties, which can then be called upon later in a Web page.

Sample Code:

#### **Listing 5-1 CLASS as Selector**

```
<HTML>
<HEAD>
<TITLE>CLASS as Selector</TITLE>
<STYLE>
H1.RED {COLOR: RED;}
H1.GREEN {COLOR: GREEN;}
H1.BLUE {COLOR: #0000FF;}
</STYLE>
</HEAD>
<BODY>

<H1 CLASS="RED">A Red Header</H1>

<H1 CLASS="GREEN">A Green Header</H1>
```

**Listing 5-1 CLASS as Selector (continued)**

```
<H1 CLASS="BLUE">A Blue Header</H1>

</BODY>
</HTML>
```

**Table 5-3 CLASS Selector Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Partial	Partial	Partial	Partial	Safe	Safe	Safe

## *ID as Selector*

Description: ID is an HTML attribute that enables Web authors to assign a common type of formatting element that can be called upon by different HTML tags.

Sample Code:

**Listing 5-2 ID as Selector**

```
<HTML>
<HEAD>
<TITLE>ID as Selector</TITLE>
<STYLE>
#GREEN {COLOR: GREEN; FONT-WEIGHT: BOLD;}
H1#GREEN { letter-spacing: 0.5em }
</STYLE>
</HEAD>
<BODY>

<H1>A Header</H1>

<P ID="GREEN">Here is some green text.</P>
```

**Listing 5-2 ID as Selector (continued)**

```
<H2 ID="GREEN">A Green Header</H2>

<P>This text is not green or bold.</P>

</BODY>
</HTML>
```

**Table 5-4 ID Selector Support in Major Browsers**

				<i><b>IE 5.0</b></i>		<i><b>NN 4.0</b></i>	
<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01</b></i> <i><b>(Mac)</b></i>	<i><b>IE 4.5</b></i> <i><b>(Mac)</b></i>	<i><b>(Win. &amp;</b></i> <i><b>UNIX)</b></i>	<i><b>NN 4.x</b></i>	<i><b>(Mac &amp;</b></i> <i><b>UNIX)</b></i>	<i><b>Opera</b></i> <i><b>3.6</b></i>
Unsafe	Partial	Partial	Partial	Partial	Partial	Partial	Partial

CSS attributes to be associated with either CLASS or ID are initially set within the header of a Web page. CSS rules for both CLASS and ID are defined within a pair of <STYLE> tags contained within the header, much like any other CSS rules that may be set there. What makes CLASS and ID stand out is their specificity, as any CSS rule that uses these elements will overrule any other CSS properties that may already be associated with a given HTML tag.

Individual selectors are given “names” that are later called upon by a CLASS attribute associated with that tag, separated by a period in the header of the Web page, as seen in the following CSS rule:

```
H1.RED {COLOR: RED;}
```

In this case the name “RED” has been associated with the <H1> tag. An <H1> tag can call upon this CSS rule by using CLASS and the name associated with this rule in the body of the Web page, as can be seen in the following line of code:

```
<H1 CLASS="RED">A Big, Red Header</H1>
```

In this case, the CSS declaration {COLOR: RED;} will be applied to this header. If CLASS="RED" was not used in the previous example, the header would be displayed using the browser’s default values for displaying a top-level header. This gives you an idea of the flexibility that CLASS can provide; it

enables Web authors to call upon a CSS rule anywhere within a Web page without having to explicitly set a CSS rule for each and every HTML tag to be modified.

When using ID, a CSS rule is preceded by a hash symbol (“#”) and then the “name”, as can be seen in the following line of code:

```
#GREEN {COLOR: #33FF33; FONT-WEIGHT: BOLD;}
```

Just as with CLASS, individual HTML tags call upon these names (and their corresponding rules) by adding the ID attribute to the tag and then specifying the name in quotes, as can be seen in this line of code:

```
<H1 ID="GREEN">A Light Green, Bold Header</H1>
```

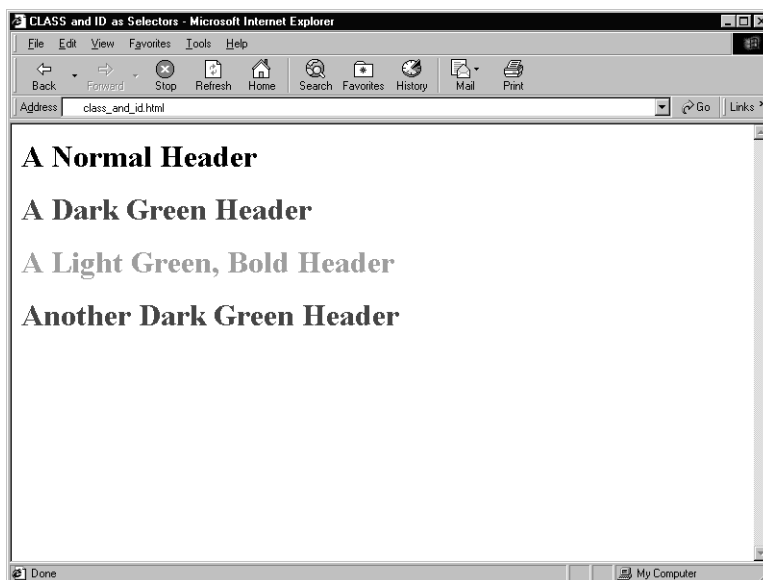
Whereas CLASS can be used with multiple elements on a Web page, ID is meant for use with only a single element. The next example of Web code demonstrates how you can use multiple ID and CLASS attributes in a Web page:

### Listing 5-3 CLASS and ID as Selectors

```
<HTML>
<HEAD>
<TITLE>CLASS and ID as Selectors</TITLE>
<STYLE>
H1.GREEN {COLOR: GREEN;}
#GREEN {COLOR: #33FF33; FONT-WEIGHT: BOLD;}
</STYLE>
</HEAD>
<BODY>
<H1>A Normal Header</H1>
<H1 CLASS="GREEN">A Dark Green Header</H1>
<H1 ID="GREEN">A Light Green, Bold Header</H1>
<H1 CLASS="green">Another Dark Green Header</H1>
</BODY>
</HTML>
```

Even though the name “GREEN” is applied in two different instances, the browser is able to make the distinction between which set of CSS rules should be applied to each header, based on whether it is called by ID or CLASS.

You can also see from this example that the names used do not have to be case-sensitive — both “GREEN” and “green” are equally effective as names.



**Figure 5-3** Multiple ID and CLASS attributes displayed on a single Web page as seen within Internet Explorer.

In terms of browser compatibility, both ID and CLASS should be considered safe to use. In fact, the “Partial” ratings given to the latest versions of Internet Explorer and Netscape Navigator result from the fact that neither meets the requirements of the official CSS specification exactly in situations that are either unlikely to appear or do not interfere with these properties’ functions. For example, in the most recent versions of Internet Explorer, the ID and CLASS properties accept digits at the beginning of a class selector (e.g., #24U {COLOR: RED;} or H1.24U {COLOR: RED;}); if they worked strictly according to the specification, they shouldn’t accept this. The CLASS property works as designed in the most recent versions of Netscape Navigator, but as for ID, it does not recognize the hash symbol (“#”) after a specified associated HTML element, (e.g., P#FRED {COLOR: RED;}), and it should. As long as Web authors are aware of these conditions and avoid them, using ID and CLASS in Web pages is perfectly safe.

# Contextual Selectors

Another basic CSS concept is that Web browsers should be able to selectively apply CSS formatting rules based upon the context in which a particular CSS property appears on a Web page. We have already seen an example of this in the previous section, as the browsers are able to distinguish which CSS rule should be applied to an `ID` or `CLASS` property when the name “GREEN” was used.

In a more general sense, contextual selectors are meant to be applied in certain situations or combinations of HTML tags appear on a Web page.

## *Contextual Selectors*

Description: Web browsers that understand CSS should be able to selectively apply CSS formatting rules based on the context in which they appear.

Sample Code:

### Listing 5-4 Contextual selectors

```
<HTML>
<HEAD>
<TITLE>Contextual selectors</TITLE>
<STYLE>
UL LI {FONT-SIZE: X-LARGE; COLOR: BLACK;}
UL UL LI {FONT-SIZE: LARGE; COLOR: GRAY;}
</STYLE>
</HEAD>
<BODY>
<UL>
<LI>An extra-large list item in a black font
<UL>
    <LI>A large sub-list item in a grey font
    <LI>Another large sub-list item in a grey font
</UL>
<LI>Another extra-large list item in a black font
</UL>

</BODY>
</HTML>
```

---

**Table 5-5 Contextual Selector Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Safe	Safe	Safe	Safe	Safe	Safe	Safe	Safe

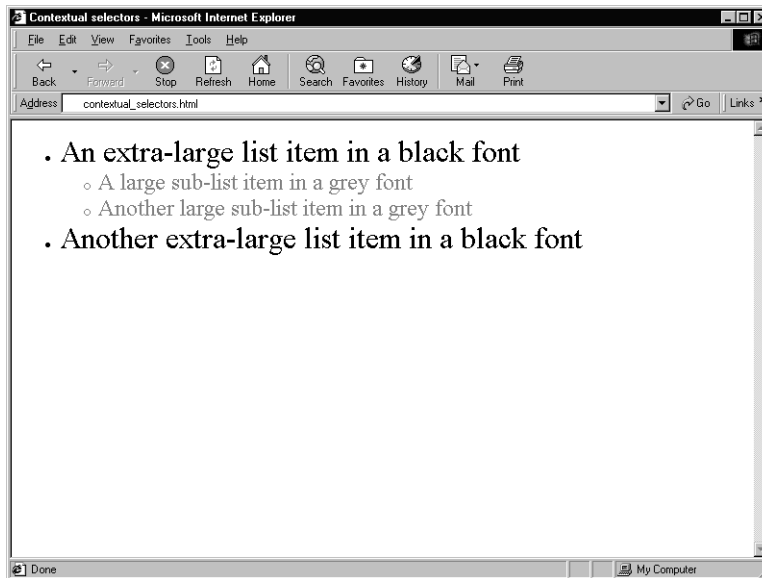
For example, say you have a couple of nested, unordered lists on a Web page, and you want them to be displayed differently when they appear, for emphasis. You could set things so that those list elements that appear in the “main” list appears larger and in a different color than those that appear in the sub-list, as in the following code:

```
<STYLE>
UL LI {FONT-SIZE: X-LARGE; COLOR: BLACK;}
UL UL LI {FONT-SIZE: LARGE; COLOR: GRAY;}
</STYLE>
```

You can see the results in Figure 5–4, which applies this code to a Web page that contains a set of nested, unordered lists.

This basic CSS concept can be applied to many other useful situations. For example, you can set nested, ordered lists to display different numeral types using the `LIST-ITEM` CSS property, so that the “outer” list elements are preceded by decimals and the nested list is preceded by roman numerals. It can also be handy in situations where you want to distinguish between elements that would ordinarily be rendered in the same way. For example, say you set all of your `<I>` and `<EM>` tags to the color value “GREEN”. But if you have both tags in a single sentence, then you will lose any distinction between the two elements. The Contextual selector rule means that you can apply a special CSS rule that allows you to set a different set of display properties whenever these two tags appear together. The following code example and illustration displays both of these examples at work:





**Figure 5-4** A nested, unordered list displaying the effects of contextual selectors.

**Listing 5-5 Contextual selectors #2**

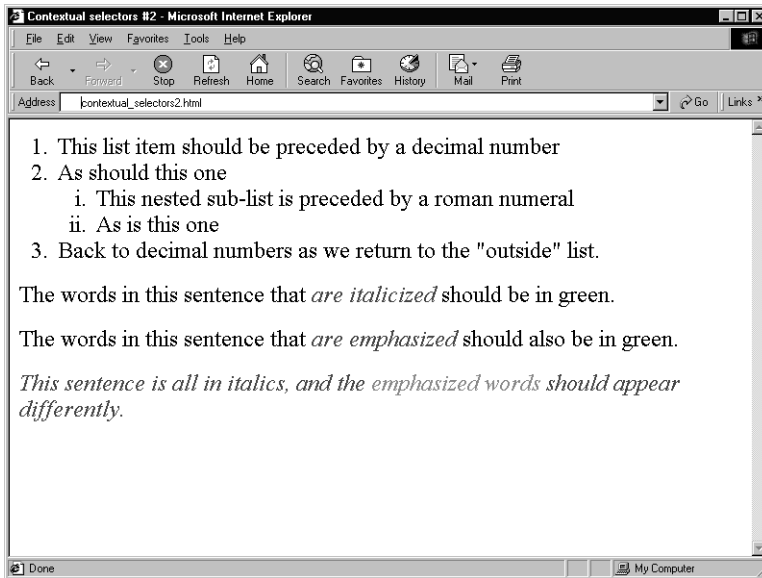
```
<HTML>
<HEAD>
<TITLE>Contextual selectors #2</TITLE>
<STYLE>
OL {LIST-STYLE: DECIMAL;}
OL OL {LIST-STYLE: LOWER-ROMAN;}
I {COLOR: GREEN;}
EM {COLOR: GREEN;}
I EM {COLOR: LIME;}
</STYLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE;">

<OL>
<LI>This list item should be preceded by a decimal number
<LI>As should this one
<OL>
<LI>This nested sub-list is preceded by a roman numeral
<LI>As is this one
</OL>
<LI>Back to decimal numbers as we return to the "outside" list.
</OL>

<P>
The words in this sentence that <I>are italicized</I> should be
in green.
<P>
The words in this sentence that <EM>are emphasized</EM> should
also be in green.
<P>
<I>This sentence is all in italics, and the <EM>emphasized
words</EM> are designed to appear differently.</I>

</BODY>
</HTML>
```

---



**Figure 5-5** More Contextual selectors at work in Internet Explorer.

Note how the special contextual CSS rule applies a different color value when the `<I>` and `<EM>` tags are both being applied within the same sentence, overriding the value when both appear singly.

Both Netscape Navigator and Internet Explorer implement contextual selectors fully, so they are safe to use. It should be noted, however, that the `LIST-ITEM` CSS property used in the last example is not universally supported.

## Comments

Experienced Web authors are aware of how handy comments can be to HTML code. In HTML, comments can be used to explain why certain actions or formatting decisions are made, or to mark distinct sections in HTML code. For the same reasons, it is handy to be able to add comments to CSS code, so that you (or someone else) have an idea of what is going on in your code, and why.

## Comments

Description: In order to explain how or why certain CSS values have been applied, it is necessary for the Web author to be able to add comments to the CSS code.

Sample Code:

### Listing 5-6 Comments

```
<HTML>
<HEAD>
<TITLE>Comments</TITLE>
<STYLE>
H1 {FONT-SIZE: X-LARGE; COLOR: BLACK; BACKGROUND: GRAY;}
/* Sets the formatting for the top-level headers */
</STYLE>
</HEAD>
<BODY>

<H1>Large, Funky-Looking Header</H1>

<H1 STYLE="FONT-SIZE: LARGE; COLOR: #333333;
BACKGROUND: YELLOW;" /* Regular top-level header formatting
style has been over-ruled in this case*/>
Another, Smaller Header</H1>

</BODY>
</HTML>
```

### Table 5-6 Comment Support in Major Browsers

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Safe

In HTML, comments are added to a Web pages by using “<!--” at the beginning of the comment and “-->” at the end of the comment, as in the following example:

```
<!-- This HTML comment will not be displayed -->
```

CSS comments follow a different model, based upon the commenting style used in the C programming language. The beginning of a CSS comment begins with a “/\*” and ends with the opposite, “\*/”. Here’s how this can be applied:

```
/* This CSS comment will not be displayed */
```

You have to be careful to be sure that when you add CSS comments, they are contained *within* CSS code statements, otherwise they may be displayed within the Web page. For example, you can insert a CSS comment within the boundaries of the <STYLE> tag in the header of a Web page, as in the following example:

```
<STYLE>
H1 {FONT-SIZE: X-LARGE; COLOR: BLACK;
BACKGROUND: GRAY;}
/* Sets the formatting for the top-level headers */
</STYLE>
```

If you are adding a comment to inline CSS code, make sure that it appears within the boundaries of the HTML tag, as in this example:

```
<H1 STYLE="FONT-SIZE: LARGE; COLOR: #333333;
BACKGROUND: YELLOW;" /* Regular top-level header
formatting style has been over-ruled in this
case */>
Another, Smaller Header</H1>
```

Comments are fully supported in the latest versions of Internet Explorer and Netscape Navigator. Since it is arguably a relatively easy thing for the browser manufacturers to implement (for after all, no actual change in the display of an element is required), the ability to add comments is found in virtually all CSS compliant browsers. Still, Web authors have to be careful to get the order of the comments right (“/\*” followed by “\*/”), and to make sure that the comments fall within the CSS code and not outside of it.

# THE CASCADE

## Topics in this Chapter

- Cascading Order
- The !IMPORTANT Property



# Chapter 6

One of the key design features behind CSS is that it is possible to specify more than one style sheet at a time within a document. This chapter takes a look at the rules governing how cascading works, and how you can increase the “weight” of one CSS statement by using the `! IMPORTANT` property.

## Cascading order

Cascading order is one the fundamental features of the CSS specification. It determines how much “weight” is to be applied to a given CSS rule, based on the situation in which it appears. In essence, more specific CSS statements overrule more general ones, and if two or more CSS statements with the same weight appear in a Web page, it is the last one that is applied. This is an over-simplification, but is a useful way to remember how things are supposed to work.

Here are the actual cascading rules, presented in more detail:

Situation: A browser is presented with a number of CSS statements, some of which conflict with each other.

- If a statement is applied to a “parent” HTML tag (such as <BODY>) and tags that are its “children” have explicit CSS statements of their own, the CSS statement that applies to the parent rules.
- The CSS statements are then sorted by explicit weight, and any marked !IMPORTANT (covered in the next section) take precedence over the others.
- Any style sheet information that may be imported through the <LINK> tag will be overruled any CSS statements contained within the Web page.
- More specific CSS statements overrule more general ones.
- If two or more CSS statements appear that have the same weight, the one specified last wins.

The generalization made earlier really applies only to the last two items presented here, but it is a good rule of thumb and can arguably be applied to the first three items in more general terms.

## ***Cascading Order***

Description: The cascading order selects specific selectors over more general ones, and if there are two CSS rules with the same weight, the last one specified is the one that is used.

Sample Code:

### **Listing 6-1 Cascading Order**

```
<HTML>
<HEAD>
<TITLE>Cascading Order</TITLE>
<LINK REL="stylesheet" TYPE="text/css" HREF="one.css">
<LINK REL="stylesheet" TYPE="text/css" HREF="two.css">
<STYLE>
LI {COLOR: OLIVE;}
OL LI {COLOR: NAVY;}
UL LI {COLOR: YELLOW;}
LI.LIME {COLOR: LIME;}
OL LI#MAR {COLOR: MAROON;}
.ORDER {COLOR: BLUE;}
.ORDER {COLOR: SILVER;}
.DECORATION {TEXT-DECORATION: LINE-THROUGH;}
</STYLE>
```



### Listing 6-1 Cascading Order (continued)

```

</HEAD>

<BODY STYLE="FONT-SIZE: LARGE; FONT-COLOR: BLACK">

<OL>
<LI>This first-level list item should be navy
<LI>Ditto with this line; neither should be olive
<UL>
<LI>This unordered sub-list item should be yellow
<LI CLASS="LIME">This one ought to be lime colored
</UL>
<LI ID="MAR">This should be maroon
<LI>This list-item should default to navy.
</OL>

<P CLASS="ORDER">
This paragraph should be rendered in a silver color, taking
on the second value defined for the class
"<CODE>order</CODE>".
</P>
<P CLASS="DECORATION">
This text should be struck-through and not appear with an
overline or underline, as the overline code and underline
codes which appear in the two linked style sheets should be
overridden by the code contained in this page.
</P>
</BODY>
</HTML>

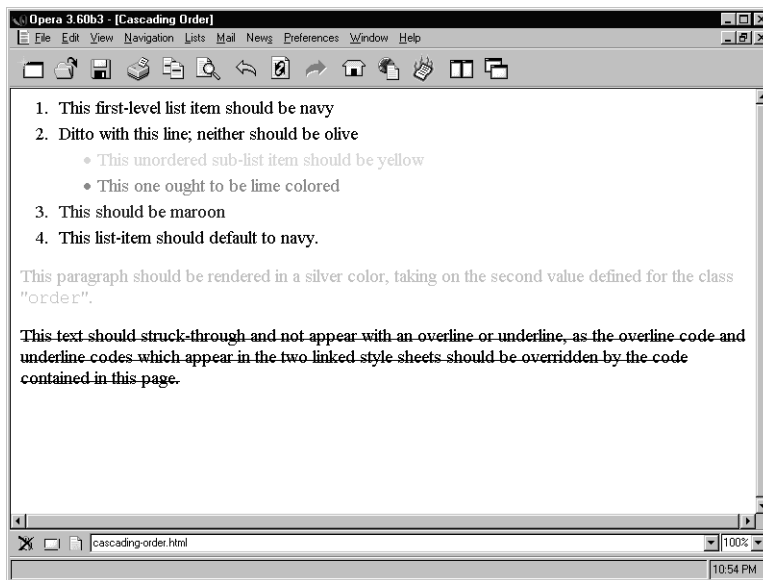
```

### Table 6-1 Cascading Order Support in Major Browsers

				<i>IE 5.0</i>			<i>NN 4.0</i>
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> (Mac)	<i>IE 4.5</i> (Mac)	(Win. & UNIX)	<i>NN 4.x</i>	(Mac & UNIX)	<i>Opera</i> 3.6
Partial	Partial	Partial	Partial	Partial	Partial	Partial	Safe

The code example above that appears with the definition will put any browser to the test when it comes to cascading rules. On the whole, both

Netscape Navigator and Internet Explorer do well with this, but only Opera 3.6 gets it completely right, as can be seen in Figure 6–1.



**Figure 6–1** The cascading order code as seen in Opera 3.6.

Notice how the initial color value set to the `<LI>` tag is olive, but this is overruled by the following statement, which says that any `<LI>` tag associated with an ordered list (`<OL>`) should be displayed in navy. This very specific CSS statement overrules the previous one. In turn, the next statement associates the color yellow whenever an `<LI>` tag is contained within an unordered listing (`<UL>`). You can go to another level of specificity when you reference a CSS statement using either a CLASS or ID statement in the code; in each of the two cases presented in this Web page, these values carry more weight than the default value specified. The two `.ORDER` CSS statements are a case where there are two identical CSS statements with the same weight. In this case, the latter of the two (“`COLOR: SILVER;`”) is what should be applied by the browser.

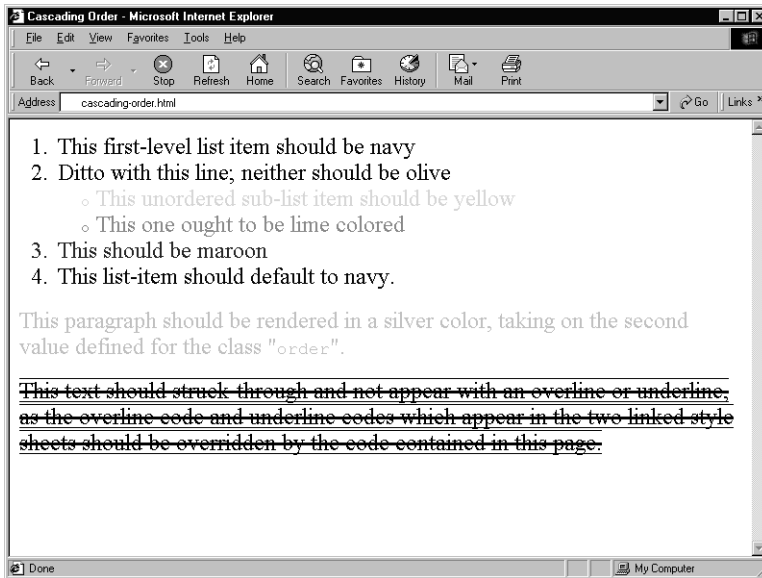
The final example is somewhat tricky, and its background needs to be explained. First off, there are two imported style sheets referenced by this Web page: `one.css` and `two.css`. `one.css` contains the following code:

```
.DECORATION {TEXT-DECORATION: UNDERLINE;}
```

`two.css` has this code:

```
.DECORATION {TEXT-DECORATION: OVERLINE;}
```

When `CLASS="DECORATION"` is specified in the code, the value that should be applied is `TEXT-DECORATION: LINE-THROUGH;`, which is specified directly within this Web page, not those contained within the two separate, external style sheets. This is where Internet Explorer 5.0 currently stumbles, as it applies all of these statements instead of the final one. This can be seen in Figure 6–2:



**Figure 6–2** The Cascading Order Code as Seen in Internet Explorer 5.0.

It should be noted that this is the only place where Internet Explorer 5.0 has problems, and it is easy enough to remedy as long as you can keep track of any CSS code being applied through external style sheets. Netscape 4.6 handles this code very poorly. When you are planning on writing CSS code that relies upon a number of cascading elements, your best bet at this point is to try to ensure that you have as few CSS conflicts as possible. Otherwise, if you run into troubles, it may be difficult to predict exactly which CSS elements will be overridden.

## !IMPORTANT

The `!IMPORTANT` property enables Web authors to increase the “weight” applied to a certain rule, ensuring that it overrules any other CSS properties that might otherwise be specified. The `!IMPORTANT` property is useful in situations where you want to ensure that a certain formatting style is always applied to a document.

### ***!IMPORTANT***

Description: Web authors can increase the weight of a particular CSS declaration by using “!IMPORTANT” in their code.

Sample Code:

#### **Listing 6-2 !IMPORTANT**

```
<HTML>
<HEAD>
<TITLE>!IMPORTANT</TITLE>
<STYLE>
P {COLOR: BLUE !IMPORTANT;}
P {COLOR: RED;}
P.FUCHSIA {COLOR: FUCHSIA;}
</STYLE>
</HEAD>
<BODY>

<P>
This line of text should be in blue.

<P STYLE="COLOR: RED;">
This line of text should also be in blue, even though it is
set to red.

<P CLASS="FUCHSIA">
This line of text should also be in blue, even though it is
set to fuchsia.

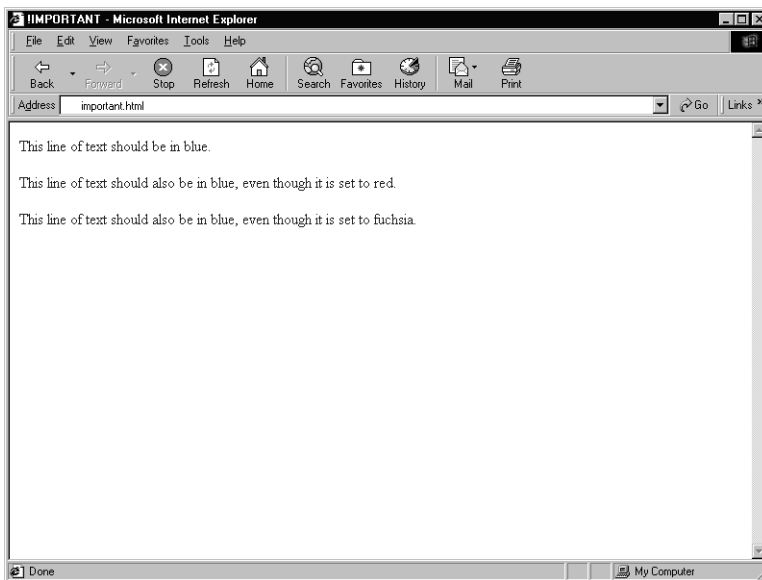
</BODY>
</HTML>
```

---

**Table 6-2 !IMPORTANT Support in Major Browsers**

				<i>IE 5.0</i>			<i>NN 4.0</i>
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i>	<i>IE 4.5</i>	<i>(Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>(Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Unsafe	Unsafe	Safe	Unsafe	Unsafe	Safe

In the code sample included with the definition of this term, the initial paragraph tag is set to the color blue, and !IMPORTANT is attached to it. Two further paragraph tags are set to different colors, but the end result, as shown in Figure 6–3 is that the initial blue value applied to the first paragraph overrules all of the other values that have been set.



**Figure 6-3** !IMPORTANT in action, in this case forcing all paragraphs to display blue text.

In the normal order of things, the initial value given to the paragraph tag would be overruled by those values that follow it. In this case the !IMPORTANT property is given the most “weight” by the browser, and consequently forces all of the text in the other paragraphs of the Web page to appear blue.

The `!IMPORTANT` property is best used in situations where you want to ensure that a particular CSS rule is applied to a page, perhaps when you are unsure of all of the properties being imported through linked CSS style sheets, or wish to override CSS code that you know appears later in the Web page.

Unfortunately, support for this property is spotty in the most recent browsers: only Internet Explorer 5.0 (and Opera 3.6) is safe. Because of this, its use cannot be recommended at this time, especially since it would most likely be used in situations where it is “important” that the elements that should be formatted in the way intended by the `!IMPORTANT` property.



# CSS UNITS

## Topics in this Chapter

- Length Units
- Percentage Units
- Color Formats
- URLs





# *Chapter*

# 7

CSS can provide Web authors with finer control over such things as setting the size, width, and color of HTML elements on a Web page, as well as adding URLs. This is accomplished using different types of measurement units. This chapter looks at the following CSS units and how they work:

- Length Units
- Percentage Units
- Color Formats
- URLs

These CSS units are all summarized in Appendix C “Colors, Units of Measure, Percentage Units and URLs”.

## Length Units

One of the chief benefits of CSS to Web authors is the fact that you can finally specify many standard units of measure to such things as borders, font sizes, and margins. There are eight units of measure that can be used with CSS properties: inches, millimeters, centimeters, picas, points, pixels, m-

lengths and x-heights. All of these units of measure can be specified by using a two-letter contracted form: inches = IN, millimeters = MM, centimeters = CM, pica = PC, point = PT, pixels = PX, m-length = EM, and x-height = EX. Units of measure are not case-sensitive, so you could use either “MM” or “mm” to set a measure for a given CSS rule when using millimeters.

## *Units of Measure*

There are eight units of measure that can be used in combination with CSS properties:

- inches – IN
- millimeter – MM
- centimeter – CM
- picas – PC
- point – PT
- pixels – PX
- m-length – EM
- x-height – EX

Sample code:

### **Listing 7-1 Length Units**

```
<HTML>
<HEAD>
<TITLE>Length Units</TITLE>
<STYLE>
.NOLEFTMARGIN {MARGIN-LEFT: 0;}
.INCH {MARGIN-LEFT: 1IN;}
.MM {MARGIN-LEFT: 25.4MM;}
.CM {MARGIN-LEFT: 2.54CM;}
.PICA {MARGIN-LEFT: 6PC;}
.POINT {MARGIN-LEFT: 72PT;}
.EX {MARGIN-LEFT: 12EX;}
.EM {MARGIN-LEFT: 6EM;}
.PIXEL {MARGIN-LEFT: 96PX;}
</STYLE>
</HEAD>
```

**Listing 7-1 Length Units (continued)**

```
<BODY STYLE="FONT-FAMILY: MONOSPACE;">
<P CLASS="NOLEFTMARGIN">
This paragraph has no left margin. All of the other
paragraphs that follow have margins that should be
set the equivalent of one inch.
</P>
<P CLASS="INCH">
This paragraph has a left margin set to 1 inch.
</P>
<P CLASS="MM">
This paragraph has a left margin set to 25.4 millimeters.
</P>
<P CLASS="CM">
This paragraph has a left margin set to 2.54 millimeters.
</P>
<P CLASS="PICA">
This paragraph has a left margin set to 6 picas.
</P>
<P CLASS="POINT">
This paragraph has a left margin set to 72 points.
</P>
<P CLASS="EX">
This paragraph has a left margin set to 12 ex.
</P>
<P CLASS="EM">
This paragraph has a left margin set to 6 em. The major
browsers calculate 1 ex = 1/2 em, which is not
technically correct, but at least it is consistent. ;-)
</P>
<P CLASS="PIXEL">
This paragraph has a left margin set to 96 pixels (which is
equivalent to 1 inch on the monitor displaying this code).
</P>

</BODY>
</HTML>
```

---

**Table 7-1 Measuring Unit Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Partial	Partial	Partial	Partial	Partial	Partial	Partial	Partial

There are several different types of measures available to the Web author, ranging from simple units of length to typographic measurement units. Inches are a standard measure of length in the imperial system of measurement, and should be familiar to all Americans. There are 25.4 millimeters to an inch. Similarly, millimeters and centimeters are standard units of length under the metric system. There are 10 millimeters to the centimeter, and there are 2.54 centimeters to the inch. Picas and Points are standard typographical units of measure. There are 72 points to the inch, 12 points to 1 pica, and 6 picas equal 1 inch. In Web browses, the typical default value for text display is 12 points for variable-width fonts (typically Times Roman), and 10 points for fixed-width fonts (typically Courier).

m-length and x-height refer to the length and height respectively of the letters “m” and “x” in a given font. In a fixed-width font such as Courier, the “m” and “x” lengths should be the same.

Pixels are a standard unit of measure for gauging height and width on computer screens. Depending on the type of computer monitor being used and the resolution being displayed, a typical pixel measures between 0.25 – 0.35 millimeters.

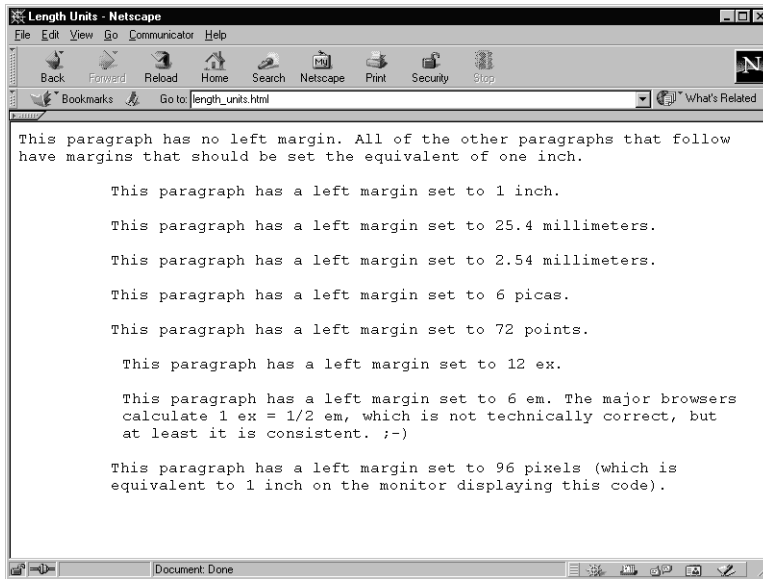
These units of measure can be divided into three types: absolute, relative and device-dependant. Inches, millimeters, centimeters, points and picas are all absolute measures. In theory, you should be able to take a ruler to your computer screen and verify that 1 inch on screen corresponds to a standard inch in length.

m-length and x-height are relative units of measure that depend on the font currently being used. Since these lengths are dependant on the size of the font being used, the m-length and x-height of a given 10 point font will necessarily be smaller than used in a 16 point font.

Pixels are a device-dependent unit of measure, as different monitors set to different screen resolutions can yield very different resolutions. Since the results will always be consistent between monitors, however, it is considered

safe to use pixels, although there may be differences in terms of the absolute scale of the item being displayed.

Figure 7–1 shows how the code example from the definition section is displayed. All units of measure used have been set to indent a line of text the equivalent of 1 inch. Note the results that are actually displayed.



**Figure 7–1** Various units of measure set to an equivalent indent value of 1 inch from the left margin.

Only the m-length and x-height units fail the test, and this is because the browser calculates 1 ex = 1/2 em, which is not technically correct. This behavior is consistent in both Internet Explorer and Netscape Navigator, and it is for this reason that they are only given a “Partial” rating when it comes to conforming to the CSS specification. In practice, however, there is little call for using m-length or x-height when a more convenient measure can be used instead, and since the results are consistent between the two major browsers, using units of length is, in general, a safe thing to add to your CSS code.

It should also be noted that Netscape Navigator is finicky when interpreting units of measure in CSS code. When using any of these units of measure, make sure that there are no spaces between the numeral and measurement unit being used. In other words, when specifying an inch, write it as “1in”, and not “1 in”, or Netscape will ignore it. Internet Explorer is more forgiving

when it comes to dealing with units of measure, and it will correctly display any CSS code that uses the space, as in “1 in” or “2.54 cm”. This is a yet another reason you should test your CSS code in both browsers before publishing to the Web.

## Percentage Units

In addition to using units of measure in your CSS code, you can also use percentage units to set the width and height of various elements displayed on screen. The behavior is really no different than when using percentage values in standard HTML code, such as using `<HR WIDTH=50%>` to set a hard rule to half the width of the browser window. Similarly, percentage values in CSS are always relative to another value, most typically the height or width of the browser window. Percentages are always specified using a numeral followed by a percentage sign, in exactly the same way as with a standard HTML tag.

### *Percentage Units*

Using a number followed by a percentage sign (“%”) always specifies percentage values in CSS. Percentage values are always relative to another value, such as the width or height of the browser window.

Sample code:

#### **Listing 7-2 Percentages**

```
<HTML>
<HEAD>
<TITLE>Percentages</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: X-LARGE">
<P STYLE="MARGIN-LEFT: 25%;">
This text is indented 25% of the browser window to the
left. Here's some extra text to make this more apparent.
</P>
```

Listing 7-2 Percentages (continued)

```
<P STYLE="MARGIN-RIGHT: 25%;">
This text is indented 25% of the browser window to the
right. Here's some extra text to make this more apparent.
</P>

</BODY>
<HTML>
```

Table 7-2 Percentage Units Support in Major Browsers

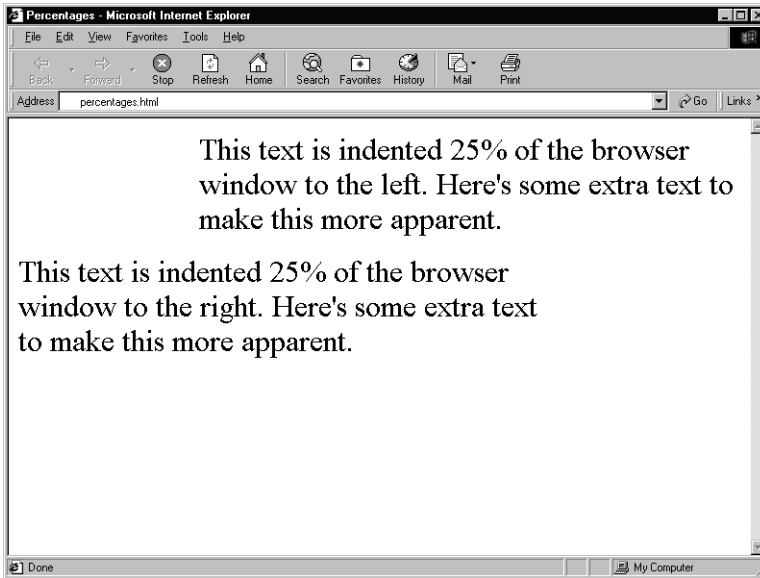
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Safe	Safe	Safe	Safe	Safe	Safe	Safe	Safe

One of the advantages of using percentage values in CSS is that they give you greater flexibility than can be achieved using standard HTML code, and should give you consistent displays across different computer monitor resolutions. Figure 7–2 displays the results of the code used in the definition section, showing how you can easily set the left and right margins for text (or other elements) on a Web page.

Percentage values are fully supported by all browsers that support CSS.

Color Formats

Web authors have long been used to adding color to such things as backgrounds and fonts in their Web pages. Web authors could use either a pre-defined color name (i.e., “BLUE”, “MAROON”, “FUCHSIA” etc) or a hexadecimal value (i.e., “#FF0000” for red) to set the color. CSS expands this use of color to almost every displayable HTML tag, and also provides additional ways to set color values.



**Figure 7-2** The effects of percentage values applied to margin values.

## Colors

There are five ways you can specify colors in CSS:

- Using a pre-defined color name (i.e., “BLUE”)
- Using a full hexadecimal color value preceded by a hash mark (i.e., “#00FF00”)
- Using a compressed hexadecimal color value preceded by a hash mark, where each number is duplicated (i.e., “#0F0”)
- Using a three-digit RGB (“Red Green Blue”) color value preceded by “RGB”, with each digit specifying the amount of color to be used on a scale from 0 to 255, and the digits contained in round brackets (i.e., RGB(0,255,0))
- Using a three-value RGB percentage color value preceded by “RGB”, with each digit specifying the percentage of color to be used on a scale from 0% to 100%, and the percentage values contained within round brackets (i.e., RGB(0%,100%,0%)).



Sample code:

Listing 7-3 Color Formats

```
<HTML>
<HEAD>
<TITLE>Color Formats</TITLE>
</HEAD>

<BODY STYLE="FONT-SIZE: LARGE">
<P STYLE="COLOR: LIME;">
This text should be lime green <CODE>(STYLE="COLOR:
LIME;" )</CODE>.
</P>
<P STYLE="COLOR: #00FF00;">
This text should be lime green <CODE>(STYLE="COLOR:
#00FF00;" )</CODE>.
</P>
<P STYLE="COLOR: #0F0;">
This text should be lime green <CODE>(STYLE="COLOR:
#0F0;" )</CODE>.
</EM>
<P STYLE="COLOR: RGB(0,255,0);">
This text should be lime green <CODE>(STYLE="COLOR:
RGB(0,255,0);" )</CODE>.
</P>
<P STYLE="COLOR: RGB(0%,100%,0%);">
This text should be lime green <CODE>(STYLE="COLOR:
RGB(0%,100%,0%);" )</CODE>.
</P>

</BODY>
</HTML>
```

Table 7-3 Color Format Support in Major Browsers

				IE 5.0			NN 4.0
IE 3.02	IE 4.x	IE 4.01	IE 4.5	(Win. &	NN 4.x	(Mac &	Opera
		(Mac)	(Mac)	UNIX)		UNIX)	3.6
Partial	Safe	Partial	Safe	Safe	Safe	Safe	Safe

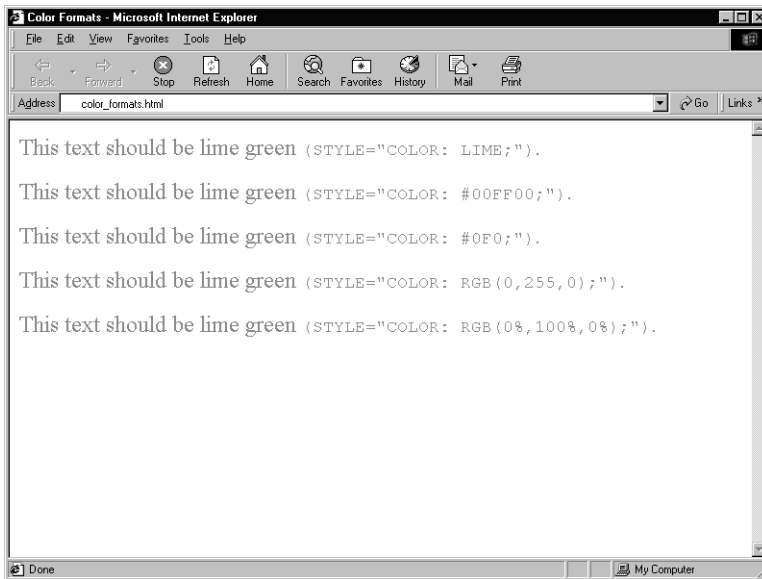
In addition to using a pre-defined color name or hexadecimal color value, you can also use a “compressed” hexadecimal color value, a three-digit RGB (“Red Green Blue”) color value, and a three-digit RGB percentage color value.

The “compressed” hexadecimal color value works by using three digits to represent the six normally used. Each single digit is effectively “doubled”, so that the full hexadecimal value for the color green, “#00FF00”, would be represented in the compressed format as “#0F0”. This is an easy, shorthand way to write simple hexadecimal color values, and easily lends itself to the use of the “safe” color palette, which uses multiples of the hexadecimal numbers 00, 33, 66, 99, CC and FF. (For more information on of the “safe” color palette, see Appendix C, “Colors, Units of Measure, Percentage Units and URLs”).

CSS also introduces two ways to add color in terms of RGB values, which is a standard way of representing color in many image manipulation programs. In the first method, the values for red, green and blue are set using a numerical scale ranging from 0 to 255. This value is contained in round brackets and is preceded by “RGB” to set the type of color value being used. Green would be represented using RGB(0,255,0). You can also use percentage values instead of numerals. Under this setting, green would be represented by RGB(0%,100%,0%). Of the two systems, the first more closely corresponds to the standard full hexadecimal color value system, as they both use 256 gradations in color, while the percentage RGB value system uses a less precise 100-point scale. Figure 7–3 shows how the code used in the definition section is displayed, with all of the color settings set to the same value of the color green.

Color formats are fully supported in the most recent versions of Internet Explorer and Netscape Navigator. It should be noted, however, that the RGB values are not recognized in the Macintosh version of Internet Explorer 4.0.1, although this has been fixed in the subsequent version.

For more information on color names, “safe” hexadecimal color values and more, see Appendix C “Colors, Units of Measure, Percentage Units and URLs”.



**Figure 7-3** The various CSS color value settings displayed.

## URLs

It is possible to add URL references within some CSS code, allowing Web authors to add such things as backgrounds to Web pages, tables or other displayable elements, or links to external CSS formatting pages.

### URLs

When an URL is used in a CSS rule, using the word “URL” and the link contained in round brackets specifies it.

Source code:

### Listing 7-4 URLs

```
<HTML>
<HEAD>
<TITLE>URLs</TITLE>
<STYLE>
BODY {BACKGROUND: URL(http://captmondo.dynip.com/cssbook/images/
background.gif); MARGIN-RIGHT: 100PX; MARGIN-LEFT: 100PX;}
</STYLE>
</HEAD>
<BODY STYLE="FONT-SIZE: X-LARGE";>
This Web page uses a background image imported from an outside
source, and has margins set to 100 pixels
on both sides so that the text does not stray over the dark
yellow "sides" set by the background image.

</BODY>
</HTML>
```

**Table 7-4 URL Support in Major Browsers**

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> <i>(Mac)</i>	<i>IE 4.5</i> <i>(Mac)</i>	<i>(Win. &amp;</i> <i>UNIX)</i>	<i>NN 4.x</i>	<i>(Mac &amp;</i> <i>UNIX)</i>	<i>Opera</i> <i>3.6</i>
Safe	Safe	Safe	Safe	Safe	Partial	Partial	Safe

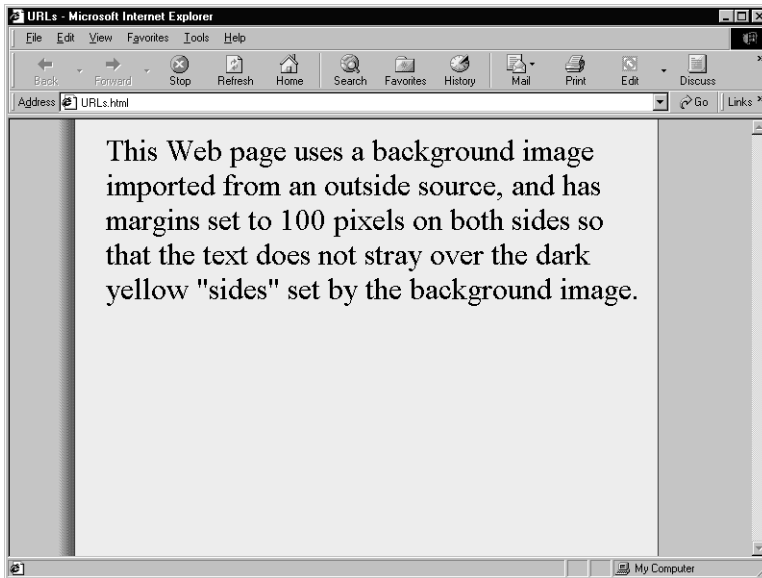
Specifying URLs in CSS is slightly different than specifying URLs in HTML. In HTML, you would reference a background image using a standard URL format, as in the following line of code:

```
<BODY BACKGROUND=" http://www.klgroup.com/images/
background.gif">
```

To specify this same URL in CSS code, you place the actual URL in round brackets, preceded by "URL" to denote it as such, as in the following example CSS code:

```
<BODY STYLE="BACKGROUND: URL(http://
www.klgroup.com/images/background.gif);">
```

Figure 7-4 shows how the example code in the definition section works. Note how the additional margin settings added to the CSS code tile the background image within certain boundaries – something that cannot be accomplished using straight HTML.



**Figure 7-4** An example of referencing an external file using an URL reference in CSS code.

Use caution when using URLs in your CSS code. The most recent version of Internet Explorer fully supports this specification, but Netscape Navigator supports it only partially. Netscape Navigator incorrectly interprets any given URL in relation to the document's URL, rather than the stylesheet's URL. What this means is that using relative URLs without an "http:" reference is impossible, making it hard to test Web code that uses CSS URL references if you are not using a Web server set up locally on your computer. This is yet another reason why you should always test your Web code in the two major browsers before publishing to the Web.

# PSEUDO-CLASSES AND PSEUDO- ELEMENTS

## Topics in this Chapter

- The ANCHOR Pseudo-Element
- The FIRST-LINE Pseudo-Class
- The FIRST-LETTER Pseudo-Class
- Pseudo-Elements in Selectors and Combining Multiple Pseudo-Elements



# Chapter 8

It is common for Web browsers to display a link differently based upon whether or not it has already been visited by the user, and to acknowledge that the user is clicking on it. Web authors have had the ability to set the colors of links on their Web pages for some time now using HTML, but CSS gives Web authors a new way to do things, while providing the flexibility to change more than the color of the links. The Pseudo-classes `ANCHOR`, `FIRST-LINE` and `FIRST-LETTER` deal with these situations.

There are a number of common typographical layout techniques (such as drop-cap letters, or special text formatting applied to the first line or letter of text in a paragraph) that, while commonplace in print media, are not easy to reproduce using standard HTML. These do not conform to the idea of typical structural elements contained in a Web page, but are better thought of as pure typographical layout formatting features. CSS enables the Web author to accomplish this using the pseudo-elements `FIRST-LINE` and `FIRST-LETTER`.

This chapter looks at the following pseudo-classes and pseudo-elements and shows how they can be used:

- `ANCHOR`
- `FIRST-LINE`
- `FIRST-LETTER`

- Pseudo-Elements in Selectors and Combining Multiple Pseudo-Elements

## ANCHOR

The `ANCHOR` pseudo-element controls how links are handled on a Web page. Many browsers display a link differently depending on whether or not it has already been visited, or is actively being clicked on by the user. The `ANCHOR` pseudo-element is actually a set of sub-properties that enable the Web author full control over how links appear on a Web page.

### ANCHOR

Description: Determines how links are displayed.

Values:

- `A:LINK` – Determines how a “resting” (i.e., unclicked) link appears on a Web page.
- `A:ACTIVE` – Determines how a link should appear when clicked (or otherwise selected) by the user.
- `A:VISITED` – Determines how the link should appear after it has been visited.

Sample code:

```
<STYLE>
A:LINK {COLOR: FUCHSIA;}
A:ACTIVE {COLOR: OLIVE;}
A:VISITED {COLOR: BLACK;}
</STYLE>
```

**Table 8-1 ANCHOR Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Partial

The `ANCHOR` pseudo-element is comprised of the following sub-properties: `A:LINK`, `A:ACTIVE` and `A:VISITED`. `A:LINK` defines how an unclicked and



unvisited link (in other words, a link “at rest”) should appear on a Web page. The `A:ACTIVE` property defines how a link should appear when it is clicked or otherwise selected by the user, and the `A:VISITED` property determines what the link should look like after it has been visited. These CSS pseudo properties are designed to work in exactly the same way as the anchor attributes you can add to the `<BODY>` tag using HTML, as the following line of code shows:

```
<BODY LINK="FUCHSIA" VLINK="BLACK" ALINK="OLIVE">
```

In this case, the HTML attribute `LINK` is equivalent to the CSS `A:LINK`, `VLINK` equivalent to `A:VISITED` and `ALINK` is equivalent to `A:ACTIVE`. The following Web page displays CSS code that is equivalent to the HTML code:

### Listing 8-1 Anchor

```
<HTML>
<HEAD>
<TITLE>Anchor</TITLE>
<STYLE>
A {COLOR: NAVY;}
A:LINK {COLOR: FUCHSIA;}
A:ACTIVE {COLOR: OLIVE;}
A:VISITED {COLOR: BLACK;}
#REDLINK {COLOR: #FF0000; FONT-SIZE: X-LARGE;}
</STYLE>
</HEAD>

<BODY STYLE="FONT-SIZE: LARGE;">
```

None of the links on this page should be a `<A HREF="link1.html">navy color</A>` as they are over-ridden by the subsequent anchor values. Before they are clicked they should appear as a `<A HREF="link2.html">fuschia color</A>`, an `<A HREF="link3.html">olive color</A>` when clicked, and `<A HREF="link4.html">black</A>` once it has been visited.

### Listing 8-1 Anchor (continued)

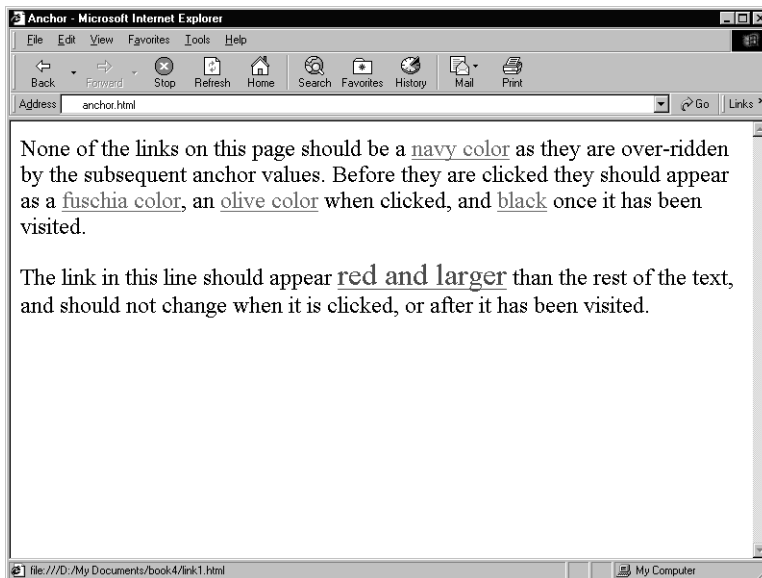
```
<P>
```

The link in this line should appear `<A HREF="link4.html" ID="REDLINK">red and larger</A>` than the rest of the text, and should not change when it is clicked, or after it has been visited.

```
</BODY>
```

```
</HTML>
```

The results of this page can be seen in Figure 8–1:



**Figure 8–1** The Effects of the CSS ANCHOR Pseudo-Properties Displayed in Internet Explorer 5.0.

In this code, the “resting” link is displayed in fuchsia, active links in olive and visited links in black.

There’s more going on with the code in this Web page, however. Note that in this case, the CSS information is contained within the header of the page. This makes sense, as you would not want to set individual values for how each

link should appear on a Web page by adding CSS properties to each and every individual link on a page. Also note how the initial value is set to the following:

```
A {COLOR: NAVY;}
```

This would normally mean that all tags on the Web page should appear in a navy color. Due to cascading rules, however, all of the subsequent color values assigned to the regular, active and visited links should overrule this setting. This works properly in Internet Explorer 5.0, but not in Netscape Navigator 4.x, which proceeds to color all of the links on the page to navy. Cascading rules can be seen in action again with the link contained in the final line of text on the Web page. The link contains a reference to the specific anchor value “REDLINK”, which renders the link in red (using the hexadecimal value “#FF0000”) and in a larger font size than the rest of the text. Because this is a specific link value, it overrides previous anchor values. This code also shows how CSS can be more flexible than the standard HTML way of doing things, as you can also tack on other display properties (relative font size) to the links on a Web.

Internet Explorer 5.0 displays all of the properties for the `ANCHOR` pseudo-properties properly, but Netscape Navigator 4.x only renders the `A:LINK` property; all other values are ignored. This is odd considering that if these same link values were formulated in HTML instead of CSS, it would work properly in Netscape Navigator.

Despite the fact that this group of pseudo-elements is not rendered correctly in Netscape Navigator, you are relatively safe using them in Web pages, so long as you do not make the color scheme you choose for your links crucial to navigating the Web page. In other words, don't use a background color normally associated with a link value, such as blue or red.

## FIRST-LINE

`FIRST-LINE` is a pseudo-element meant to reproduce the type of text sometimes seen in the first lines of newspapers and magazines, setting it apart from the rest of the text in a paragraph. It changes the way the first line of text in a paragraph is displayed. Using the `FIRST-LINE` pseudo-element, you can change the color, background, spacing or other ways in which the first line of text in a paragraph is displayed.

## FIRST-LINE

Description: Sets how the first line of text in a paragraph should be displayed.

Sample code:

```
<STYLE>
P:FIRST-LINE {FONT-VARIANT: SMALL-CAPS; COLOR:
NAVY;}
</STYLE>
```

**Table 8-2 FIRST-LINE Support in Major Browsers**

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> (Mac)	<i>IE 4.5</i> (Mac)	(Win. & <i>UNIX</i> )	<i>NN 4.x</i>	(Mac & <i>UNIX</i> )	<i>Opera</i> <i>3.6</i>
Unsafe	Unsafe	Unsafe	Unsafe	Unsafe	Unsafe	Unsafe	Safe

The following Web page code shows how it can be used:

**Listing 8-2 First-Line**

```
<HTML>
<HEAD>
<TITLE>First-Line</TITLE>
<STYLE>
P:FIRST-LINE {FONT-VARIANT: SMALL-CAPS; COLOR: NAVY;}
</STYLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE;">

<BR>

<P>
The first line, and only the first line of this paragraph,
should be displayed in a navy color and in
small caps. All of the subsequent lines of this paragraph
should appear in the default text color, which
is black. Here is some extra text in order to make the
effect clear.
</P>
```

**Listing 8-2 First-Line (continued)**

Here is some more text. The first line of this text is not displayed in small caps nor in a navy color.

<P>

<TABLE WIDTH="450" BORDER>

<TR>

<TD>

<P>

The first line, and only the first line of this paragraph, should be displayed in a navy color and in small caps. All of the subsequent lines of this paragraph should appear in the default text color, which is black. Here is some extra text in order to make the effect clear.

</P>

Here is some more text. The first line of this text is not displayed in small caps nor in a navy color.

</TD>

</TR>

</TABLE>

</BODY>

</HTML>

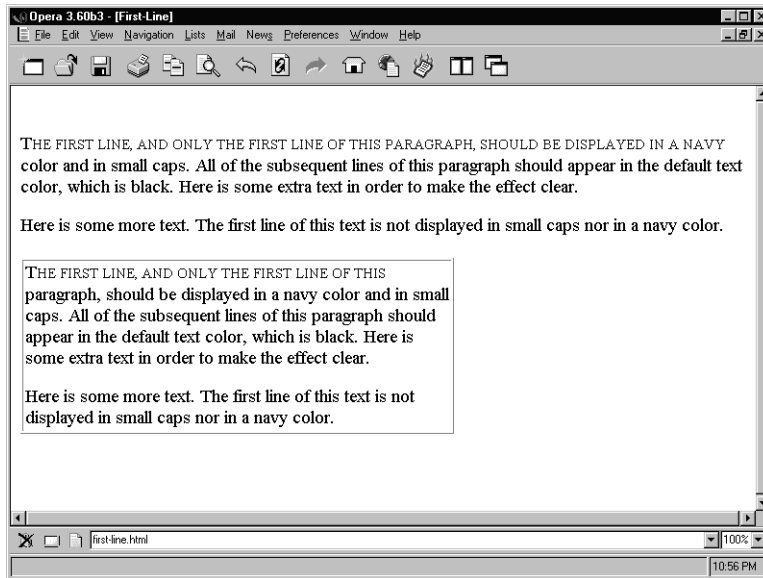
---

The results of this page can be seen in Figure 8–2.

Notice that only the first line of text in a paragraph is altered in its display, as opposed to the first sentence of a paragraph. Note also how the size of the line of text determines how much text will be rendered in the FIRST-LINE format; the first line of text that appears in the small table has a shorter first line, and subsequently fewer words are formatted under the FIRST-LINE properties specified.

The FIRST-LINE pseudo-element is designed to handle only certain types of additional CSS formatting properties: the various font color and background properties, plus the WORD-SPACING, LETTER-SPACING, TEXT-DECORATION, TEXT-TYPE, LINE-HEIGHT and CLEAR CSS properties.

Neither Internet Explorer 5.0 nor Netscape Navigator 4.x display the desired formatting when the FIRST-LINE pseudo-element is specified; the snap-shot used to illustrate this pseudo-element in action comes from the



**Figure 8-2** The Effects of the FIRST-LINE Pseudo-Element Displayed in the Opera 3.6 Browser.

Opera 3.6 browser, which displays this particular CSS pseudo-element as designed.

## FIRST-LETTER

FIRST-LETTER is another pseudo-element, designed specifically to give either a “drop cap” or “initial cap” appearance to the first letter than appears in a paragraph, as seen in many newspapers and magazines.

### ***FIRST-LETTER***

Description: Sets how the very first letter in the first line of text in a paragraph should be displayed.

Sample code:

```
<STYLE>
P {FONT-SIZE: 12pt; LINE-HEIGHT: 12pt;}
P:FIRST-LETTER {FONT-SIZE: 18pt; FLOAT: LEFT;
COLOR: NAVY; FONT-WEIGHT: BOLD;}
```

&lt;/STYLE&gt;

**Table 8-3 FIRST-LETTER Support in Major Browsers**

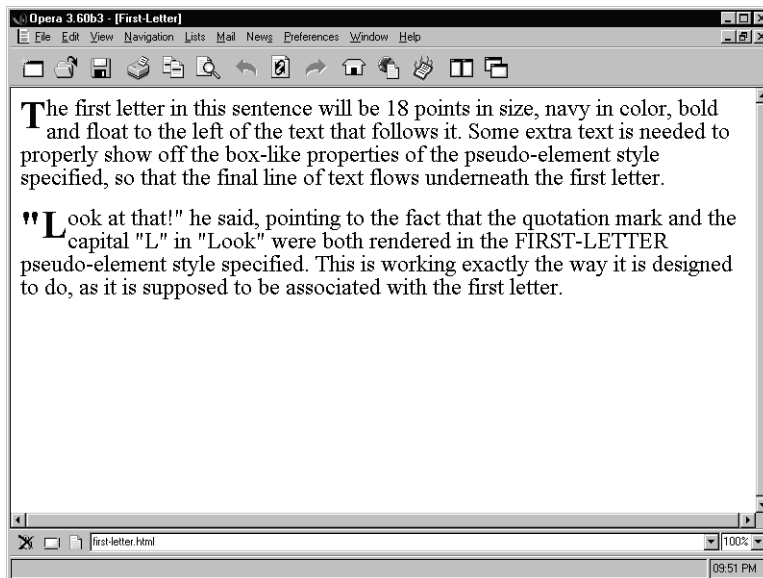
				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> (Mac)	<i>IE 4.5</i> (Mac)	(Win. & UNIX)	<i>NN 4.x</i>	(Mac & UNIX)	<i>Opera</i> 3.6
Unsafe	Unsafe	Unsafe	Unsafe	Unsafe	Unsafe	Unsafe	Safe

This particular pseudo-element takes a bit of planning in order to get the effect to look just right. The following Web page code demonstrates how a proper drop-cap effect can be accomplished:

**Listing 8-3 First-Letter**

```
<HTML>
<HEAD>
<TITLE>First-Letter</TITLE>
<STYLE>
P {FONT-SIZE: 18PT; LINE-HEIGHT: 18PT;}
P:FIRST-LETTER {FONT-SIZE: 30PT; FLOAT: LEFT; COLOR: NAVY;
FONT-WEIGHT: BOLD;}
</STYLE>
</HEAD>
<BODY>
<P>
The first letter in this sentence will be 18 points in
size, navy in color, bold and float to the left of the text
that follows it. Some extra text is needed to properly show
off the box-like properties of the pseudo-element style
specified, so that the final line of text flows underneath
the first letter.
<P>
"Look at that!" he said, pointing to the fact that the
quotation mark and the capital "L" in "Look"
were both rendered in the FIRST-LETTER pseudo-element style
specified. This is working exactly the way
it is designed to do, as it is supposed to be associated
with the first letter.
</BODY>
</HTML>
```

Figure 8–3 shows how this code is displayed in the Opera 3.6 browser.



**Figure 8–3** The Effects of the FIRST-LETTER Pseudo-element Displayed in the Opera 3.6 Browser.

Note how the first letter in each of the sentences displayed take on the values assigned to the `FIRST-LETTER` pseudo-element: the letter should be 30 points in size, navy in color, bold and float to the left of the text that follows it. Note the line of code that precedes the `FIRST-LETTER` pseudo-element sets the text and the line-height that follows the first letter to 18 points in height. This is done so that the text aligns properly with the top of the first letter, otherwise the following text would float slightly above the first letter.

The second line of text shows another intended effect of the `FIRST-LETTER` pseudo-element: any quotation mark that precedes the first-letter takes on the same properties as the first letter itself. This is desirable, as paragraphs often begin with quotation marks. Note that this is not designed to work with other symbols, such as a round bracket (i.e., a parenthesis) or mathematical symbols.

In order to create an initial cap instead of a drop cap in the code example shown earlier, all you would have to do would be to remove the `FLOAT: LEFT;` rule from the code to produce the desired effect.



Many of the same formatting restrictions that apply to `FIRST-LINE` apply to the `FIRST-LETTER` pseudo-element: it can take the various font color and background properties, plus the `WORD-SPACING`, `LETTER-SPACING`, `TEXT-DECORATION`, `TEXT-TRANSFORM`, `LINE-HEIGHT` and `CLEAR` CSS properties.

The `FIRST-LETTER` pseudo-element is not supported in the most recent versions of either Internet Explorer or Netscape Navigator. You can see the intended effect of the `FIRST-LETTER` pseudo-element in the Opera 3.6 browser.

## Pseudo-Elements in Selectors and Combining Multiple Pseudo-Elements

By looking at the previous code examples covered in this section, you may have wondered how you can combine pseudo-selectors with other CSS properties, or how to differentiate between combinations of pseudo-elements.

Pseudo-elements can be combined with classes in selectors so that they can be specifically selected for. Pseudo-elements can also be tacked onto a selector. Figure 8–4, which is an adaptation of what was seen earlier in the `FIRST-LINE` section, shows both pseudo-elements combined with classes at work:

### Listing 8-4 Pseudo-Elements in Selectors

```
<HEAD>
<TITLE>Pseudo-Elements in Selectors</TITLE>
</HEAD>
<STYLE>
P.INITIAL:FIRST-LINE {FONT-VARIANT: SMALL-CAPS; COLOR:
NAVY; FONT-SIZE: 20PT;}
BODY P:FIRST-LETTER {COLOR: BLUE; FONT-SIZE: 30PT;}
BODY {FONT-SIZE: 20PT;}
</STYLE>
<BODY>
```

**Listing 8-4 Pseudo-Elements in Selectors (continued)**

```
<P CLASS="INITIAL">
```

The first line, and only the first line of this paragraph, should be displayed in a navy color and in small caps, and because the INITIAL class is selected for this paragraph, the first letter of the first line will appear as a large, blue letter. All of the subsequent lines of this paragraph should appear in the default text color, which is black.

```
</P>
```

Here is some more text. The FIRST-LINE and FIRST-LETTER pseudo-element values do not apply to this sentence because it is outside of previous closed paragraph.

```
<P>
```

This is a new paragraph, but because the INITIAL class is not selected, only the FIRST-LETTER value will be displayed.

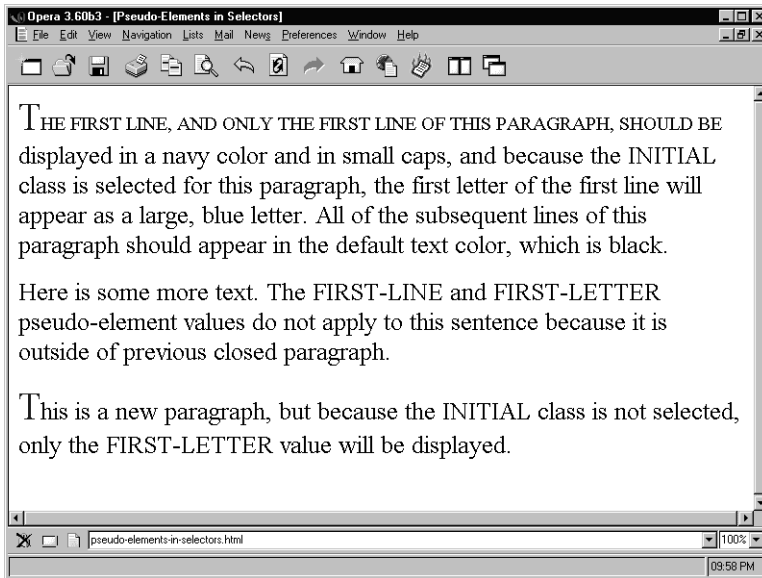
```
</P>
```

```
</BODY>
```

```
</HTML>
```

---

Note how the first paragraph combines the display values set by FIRST-LINE and FIRST-LETTER, but that the third paragraph only shows the effects of FIRST-LETTER. That is because the effects of the FIRST-LINE pseudo-element are set using the class P.INITIAL, which is selected for in the first paragraph by using the code `<P CLASS="INITIAL">`. The FIRST-LETTER pseudo-element has been tacked onto the BODY selector, which applies the value of a large blue letter to the first letter that appears in a paragraph. Note how the second paragraph, which is not contained within a paragraph tag, does not take on either of these values (nor is it supposed to, since it lies outside of the end-paragraph tag).



**Figure 8-4** Combination of the Effects of the FIRST-LETTER and FIRST-LINE Pseudo-Properties Displayed in the Opera 3.6 Browser.

Since neither Netscape Navigator 4.x nor Internet Explorer 5.x support all of the pseudo-elements covered in this section, there is little point using this sort of code in a Web page as yet. The browser used to illustrate the code “in action” is Opera 3.6, which ably demonstrates how things are supposed to work according to the official CSS specification.

# FONT PROPERTIES

## Topics in this Chapter

- The FONT-FAMILY Property
- The FONT-SIZE Property
- The FONT-STYLE Property
- The FONT-VARIANT Property
- The FONT-WEIGHT Property
- The FONT Property



# Chapter 9

Web authors have always wanted to have greater control over the formatting of font properties on Web pages. Netscape tried to oblige the Web authoring community by introducing the `<FONT>` tag in Netscape Navigator 1.0, which allowed authors to change the size of the displayed text by using the `SIZE` attribute. Microsoft added two new attributes, `COLOR` and `FACE`, with the release of Internet Explorer 1.0, the first of which was adopted within Netscape Navigator 2.0 and the latter in version 3.0. Despite the efforts by Netscape and Microsoft to improve the `<FONT>` tag (which still continues, as Netscape recently introduced `POINT-SIZE` and `WEIGHT` attributes to the `<FONT>` tag in Netscape Navigator 4.0), Web authors will find greater flexibility with the CSS `FONT` family of properties.

This chapter takes a look at the `FONT` family of CSS properties, which consists of the following:

- `FONT-FAMILY`
- `FONT-SIZE`
- `FONT-STYLE`
- `FONT-VARIANT`
- `FONT-WEIGHT`
- `FONT`



It can take any of the following values: CURSIVE, FANTASY, MONOSPACE, SANS-SERIF and SERIF. One of the drawbacks of the <FONT> tag is that you had to specify a particular font name (such as “Arial”, “Times Roman” or “Courier”) in order to try and get it to appear on screen. The FONT-FAMILY CSS property tries to get around this by specifying the type or “family” of font to be displayed. Given a particular value, the browser does its best to come up with a match, so if you specify MONOSPACE you may get a Courier font to appear on screen, whereas if you specify SERIF, you may get a Times Roman font instead. This CSS property is still dependant on the user’s computer to come up with a matching font family, but the font family categories encompass a broader spectrum of fonts that could only be accomplished with a <FONT> tag containing multiple values attached to the FACE attribute. In other words, you could choose to include the following HTML code in order to have a get a serif font displayed on the user’s screen:

```
<FONT FACE="Garamond, AGaramond, Times, Times  
Roman, CG Times, Home, Goudy"><B>This sentence is  
displayed in a serif font</B></FONT>
```

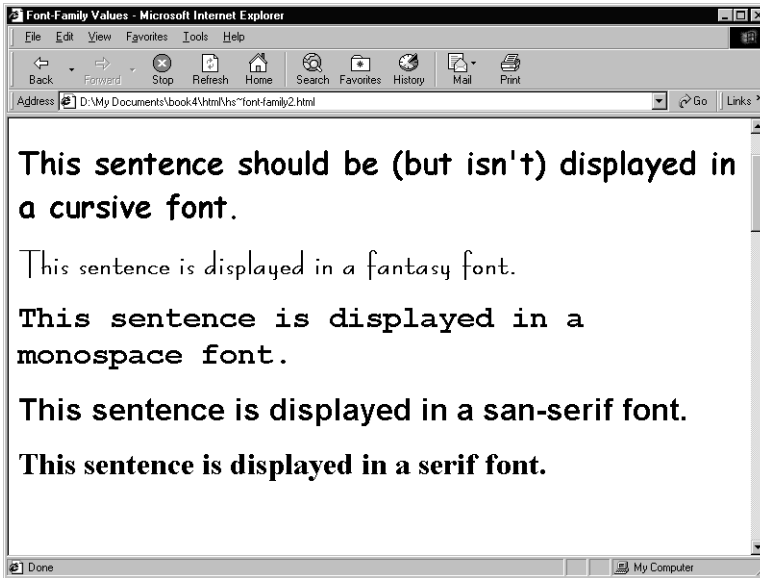
Or you could use the following, more concise CSS code instead:

```
<B STYLE="FONT-FAMILY: SERIF">This sentence is  
displayed in a serif font</B>
```

The following code example shows the different values that can be applied to FONT-FAMILY, whose results are displayed in Figure 9–1:

```
<B STYLE="FONT-FAMILY: CURSIVE">This sentence  
should be (but isn't) displayed in a cursive  
font.</B>  
<P>  
<B STYLE="FONT-FAMILY: FANTASY">This sentence is  
displayed in a fantasy font.</B>  
<P>  
<B STYLE="FONT-FAMILY: MONOSPACE">This sentence  
is displayed in a monospace font.</B>  
<P>  
<B STYLE="FONT-FAMILY: SANS-SERIF">This sentence  
is displayed in a san-serif font.</B>  
<P>  
<B STYLE="FONT-FAMILY: SERIF">This sentence is  
displayed in a serif font.</B>
```

The illustration shows how the browser tries to interpret and then apply the correct font family based on what it can find on the user’s system. The ini-



**Figure 9-1** The results of the FONT-FAMILY code displayed.

tial line of code specifies that a cursive (i.e., handwriting) font should be displayed in the browser. This doesn't happen because the browser is unable to find a close match, although it does its best and displays a font that certainly looks different from the default browser font. The browser is able to find font family matches for the rest of the code, and displays the text accordingly.

There is an inherent trade-off when you use the FONT-FAMILY CSS property in this manner: what you lose in specificity you gain in terms of the greater likelihood of getting a font match and in brevity of code.

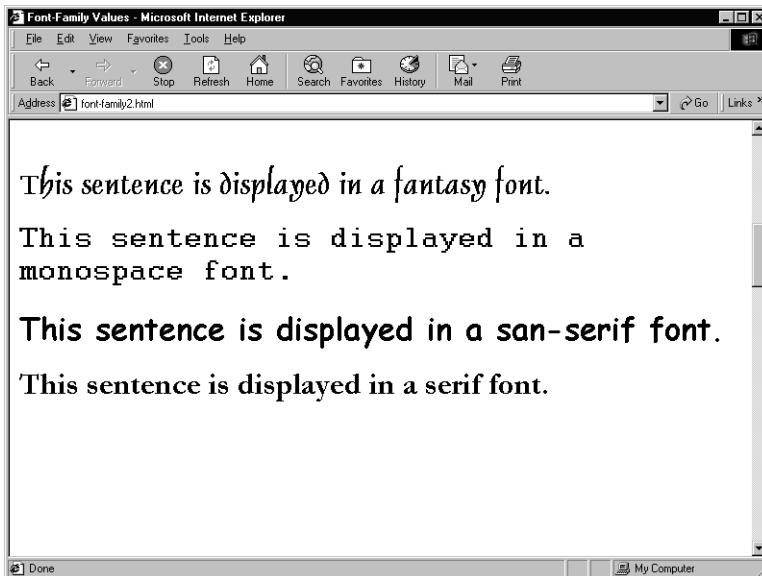
While this is a fairly simple way of specifying font appearance on a Web page, FONT-FAMILY also lets you combine more specific font name information in a similar manner to the <FONT> tag. For example, you might want to make sure that, of all of the serif fonts available, a Times font (such as Times New Roman) takes precedence. This is done by specifying the name of the specific font you want in single quotes, backed up by a more general font name, followed by a FONT-FAMILY value that the font belongs to. Here's what that code would look like in the case of trying to specify the Times New Roman font:

```
<H1 STYLE="FONT-FAMILY: 'Times New Roman', Times,
SERIF">Serif font</H1>
```



The browser interprets the code by moving from the specific font name, if it can't find a match, to the family to which the font belongs, and then to the generic font family type if no match has been found. To see this in action, compare the following code and illustration to the first one:

```
<B STYLE="FONT-FAMILY: 'Dauphin', Dauphin,  
FANTASY">This sentence is displayed in a fantasy  
font.</B>  
  
<P>  
  
<B STYLE="FONT-FAMILY: 'Courier', Courier,  
MONOSPACE">This sentence is displayed in a  
monospace font.</B>  
  
<P>  
  
<B STYLE="FONT-FAMILY: 'Comic Sans MS', Comic,  
SANS-SERIF">This sentence is displayed in a san-  
serif font.</B>  
  
<P>  
  
<B STYLE="FONT-FAMILY: 'Garamond', Garamond,  
SERIF">This sentence is displayed in a serif  
font.</B>
```



**Figure 9-2** Results of the more specific FONT-FAMILY code displayed.

Note that although the same font families are being used, the more specific references select a different type of font within a font family.

This CSS property is safe to use, as it is fully supported in versions 4.0 and up of Internet Explorer and Netscape Navigator.

## The FONT-SIZE Property

The `FONT-SIZE` property is used to set the size of the text to be displayed. It can do this in any one of three ways: by specifying the size in a relative term to the default browser font, in relative terms to the last specified font size, or with a precise size value.

In many ways, the `FONT-SIZE` CSS property is like the `<FONT>` tag when its `SIZE` attribute is used, but again the CSS property provides greater flexibility than the HTML tag.

### ***FONT-SIZE***

Description: Sets the display size of text.

CSS Family Type: Font

Values:

- `LARGE` | `MEDIUM` | `SMALL` | `X-LARGE` | `X-SMALL` | `XX-LARGE` | `XX-SMALL` - Sets the size of the text. Values range from `XX-SMALL` (smallest) to `XX-LARGE` (largest).
- `LARGER` | `SMALLER` - Changes the size of the current specified element relative to a previously specified value.
- *n* value - Specific unit value for the specified element. Can also take a separate value that corresponds to line-height.
- % value - Sets a percentage for the size of a font relative to the base font size.

Sample code:

```
<H1 STYLE="FONT-SIZE: 48 pt">Monster-Sized  
Heading</H1>
```

**Table 9-2 FONT-SIZE Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Partial	Partial	Partial	Partial	Partial	Safe	Safe	Safe

The FONT-SIZE property provides seven specific name values for setting the size of the font displayed. Ranging from smallest to largest, they are: XX-SMALL, X-SMALL, SMALL, MEDIUM, LARGE, X-LARGE and XX-LARGE. They are equivalent to the <FONT> tag using the attributes ranging from SIZE="1" to SIZE="7". The following code and illustration shows how the text is displayed:

```
<B STYLE="FONT-SIZE: XX-SMALL">This sentence is set
to XX-SMALL.</B>

<P>

<B STYLE="FONT-SIZE: X-SMALL">This sentence is set
to X-SMALL.</B>

<P>

<B STYLE="FONT-SIZE: SMALL">This sentence is set
to SMALL.</B>

<P>

<B STYLE="FONT-SIZE: MEDIUM">This sentence is set
to MEDIUM.</B>

<P>

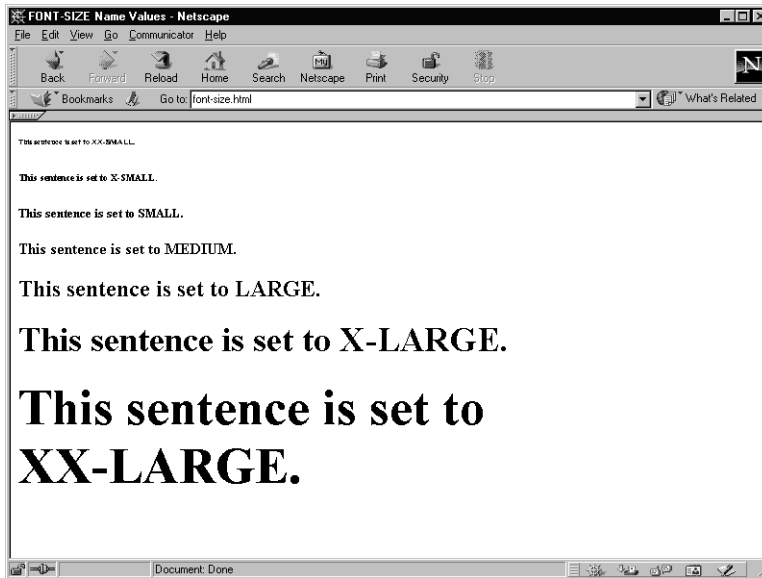
<B STYLE="FONT-SIZE: LARGE">This sentence is set
to LARGE.</B>

<P>

<B STYLE="FONT-SIZE: X-LARGE">This sentence is set
to X-LARGE.</B>

<P>

<B STYLE="FONT-SIZE: XX-LARGE">This sentence is
set to XX-LARGE.</B>
```



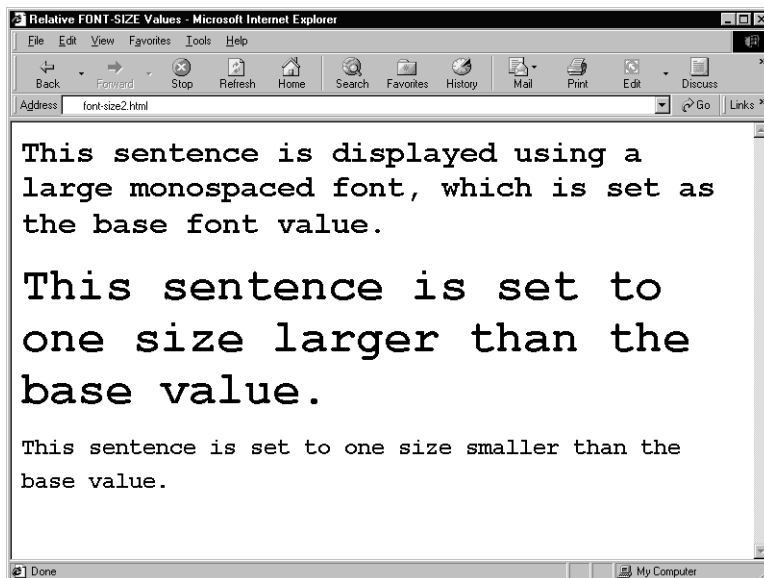
**Figure 9-3** The results of the name values for FONT-SIZE displayed in Netscape Navigator.

It is also possible to set the font displayed to a value that is one size larger or smaller than the base font size value being used. This is analogous to the way you can set the HTML tag `<BASEFONT>` to a particular value, and then set the font size relative to that value (by using `<FONT SIZE="+1">` or `<FONT SIZE="-1">`). This can be done in CSS by using the values `LARGER` and `SMALLER`.

In the following code example, the `FONT-SIZE` for the `<BODY>` tag has been set to the name value `LARGE`. The last two paragraphs of bolded text are set to one size larger and one size smaller than the default `X-LARGE` value.

**Listing 9-1 Relative FONT-SIZE Values**

```
<HTML>
<HEAD>
<STYLE>
B {FONT-FAMILY: MONOSPACE;}
</STYLE>
<TITLE>Relative FONT-SIZE Values</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: X-LARGE;">
<P>
<B>This sentence is displayed using a large monospaced
font, which is set as the base font value.</B>
<P>
<B STYLE="FONT-SIZE: LARGER">This sentence is set to one
size larger than the base value.</B>
<P>
<B STYLE="FONT-SIZE: SMALLER">This sentence is set to one
size smaller than the base value.</B>
</BODY>
</HTML>
```



**Figure 9-4** FONT-SIZE displaying the effects of the LARGER and SMALLER settings.

Keep in mind when using the LARGER and SMALLER values that they are relative to the base font size that has been set; they are not relative to each other. If that were the case, the SMALLER value seen in Figure 9-4 would be the same size as the first line, which is set to the default value of X-LARGE. Instead, it is one size smaller than extra-large, equivalent to a value of LARGE.

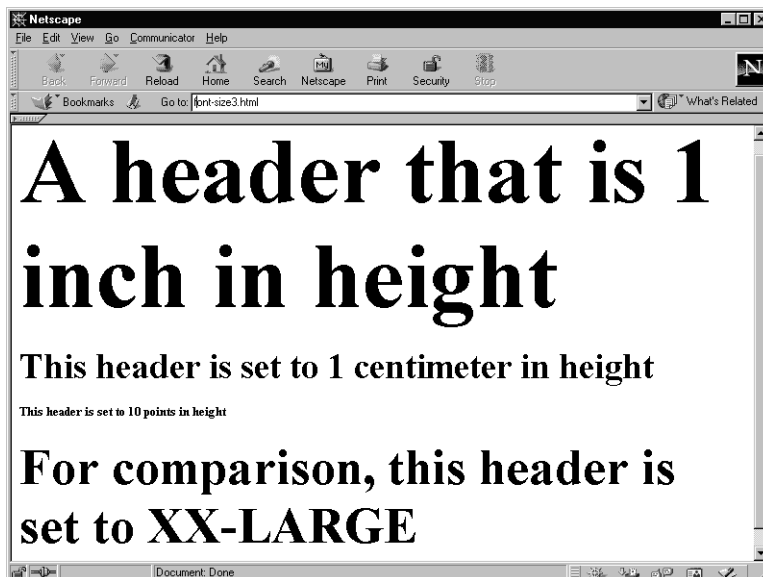
The most powerful feature of the FONT-SIZE property is that you can specify a particular size (or more properly, the height) of a font using a standard unit of measure. For example, you can set a font to be 1 inch or centimeter in height, or set the specific point value for the font. The following code shows how this can be accomplished:

```
<H1 STYLE="FONT-SIZE: 1in">A header that is 1 inch  
in height</H1>
```

```
<H2 STYLE="FONT-SIZE: 1cm">This header is set to 1  
centimeter in height</H2>
```

```
<H3 STYLE="FONT-SIZE: 10pt">This header is set to  
10 points in height</H3>
```

```
<H1 STYLE="FONT-SIZE: XX-LARGE">For comparison,  
this header is set to XX-LARGE</H1>
```



**Figure 9-5** Specific measurement unit sizes for FONT-SIZE displayed.

FONT-SIZE is also capable of taking on a percentage value. The percentage value that is specified is relative to the size of the base font. The following code and illustration show how percentage values work with FONT-SIZE.

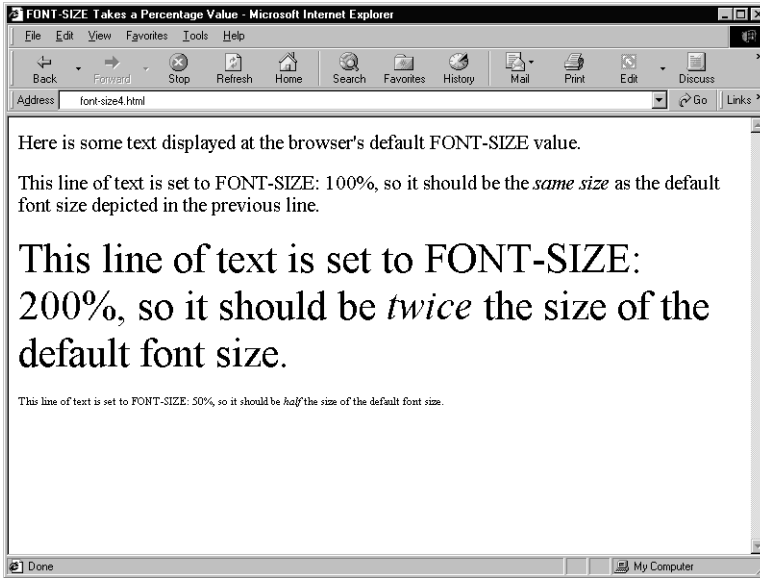
### Listing 9-2 FONT-SIZE Takes a Percentage Value

```
<HTML>
<HEAD>
<TITLE>FONT-SIZE Takes a Percentage Value</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: 16pt">
Here is some text displayed at the browser's default FONT-
SIZE value.
<P STYLE="FONT-SIZE: 100%">
This line of text is set to FONT-SIZE: 100%, so it should
be the <I>same size</I> as
the default font size depicted in the previous line.
</P>
<P STYLE="FONT-SIZE: 200%">
This line of text is set to FONT-SIZE: 200%, so it should
be <I>twice</I> the size
of the default font size.
</P>
<P STYLE="FONT-SIZE: 50%">
This line of text is set to FONT-SIZE: 50%, so it should be
<I>half</I> the size
of the default font size.
</P>
</BODY>
</HTML>
```

---

As you can see in Figure 9–6, the 100% value corresponds exactly to the default font size. 200% displays the font at twice the default size, and 50% reduces the displayed font to a miniscule font half the size of the default font size.

The FONT-SIZE property gives the Web author control over the exact sizing of fonts displayed on screen. You can use any of the standard units of measure recognized by CSS (see “CSS Units” earlier in this part for more information).



**Figure 9-6** The results of percentage values applied to FONT-SIZE as displayed in Internet Explorer.



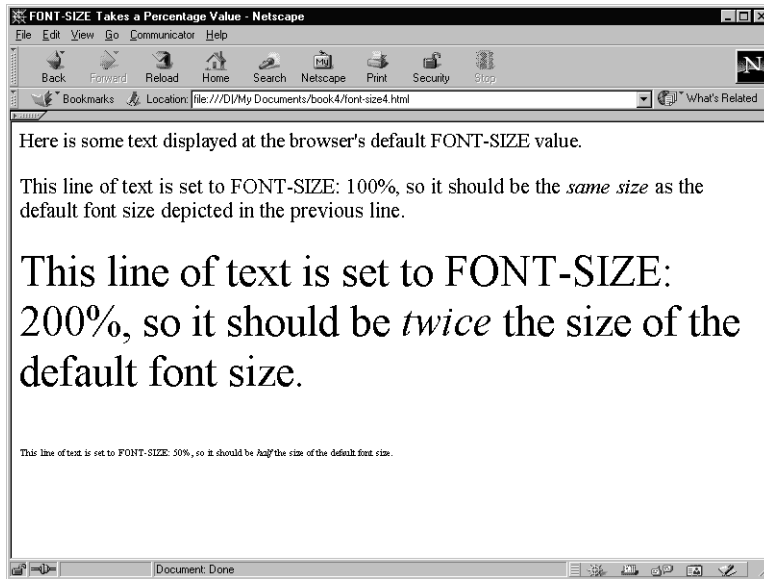
#### Core Tip

*Be careful whenever you are writing code containing measurement units, as Netscape Navigator is finicky when it comes to reading them. Make sure there is no space between the numerical value and the measurement value (i.e., "1in") or Netscape will not recognize it as a legitimate value and will ignore it. Internet Explorer is much more forgiving, and will display a value correct if there is a space between the number and measurement unit (i.e., "1 in"), which is, for most people, a more natural way of writing the figure.*

The FONT-SIZE property is fairly safe to use, as it is fully supported in Netscape Navigator and largely supported within Internet Explorer. The only problem within Internet Explorer is all of the named FONT-SIZE values are one size smaller than they should be, so that the value SMALL is equivalent to the default value, when it should be MEDIUM. This means that you can expect the named values to be displayed differently (although consistently) in Netscape Navigator and Internet Explorer. As an example of the effect, see



Figure 9–7 that shows the same code used in the first example, and notice the difference in size.



**Figure 9–7** The results of the name values for FONT-SIZE displayed in Internet Explorer.

## The FONT-STYLE Property

The FONT-STYLE property exists primarily as a way of telling a font to display itself in an italic setting. It takes one of three possible values: ITALIC, NORMAL and OBLIQUE. NORMAL tells the browser to use the default (i.e., non-italic) setting for a font. ITALIC and OBLIQUE are equivalent, and should display a font in its italic (or “oblique”) setting.

### ***FONT-STYLE***

Description: Sets whether or not the displayed font is italicized.

CSS Family Type: Font

Values:

- **ITALIC | NORMAL** - Determines whether or not text is italicized
- **OBLIQUE** – To be used to create italicized text (used for fonts that do not understand the term “italic”).

Sample code:

```
<I STYLE="FONT: OBLIQUE">This text is set to  
oblique, </I>
```

**Table 9-3 FONT-STYLE Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Partial	Safe	Safe	Safe	Safe	Partial	Partial	Safe

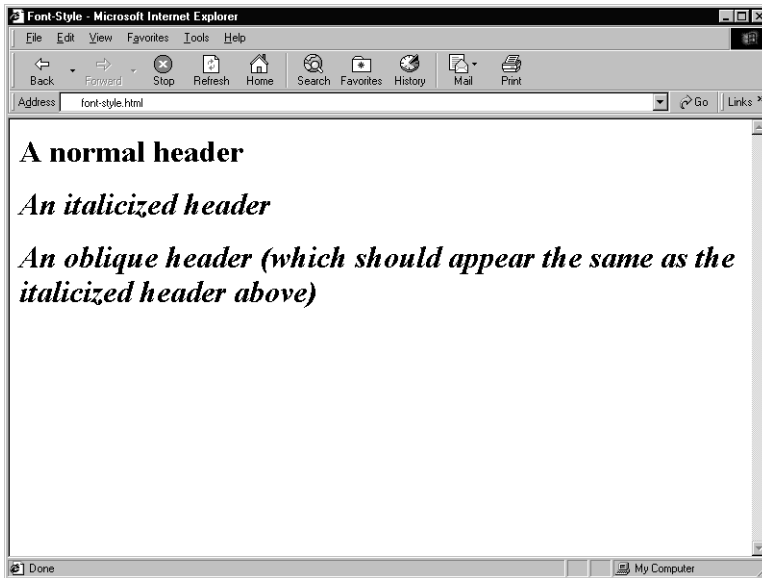
The value **OBLIQUE** is present for those fonts that do not understand the value **ITALIC**; they are equivalent terms.

The following code and illustration show how the three values for **FONT-STYLE** work:

**Listing 9-3 Font-Style**

```
<HTML>
<HEAD>
<TITLE>Font-Style</TITLE>
</HEAD>
<BODY>
<H1 STYLE="FONT-STYLE: NORMAL">A normal header</H1>
<H1 STYLE="FONT-STYLE: ITALIC">An italicized header</H1>
<H1 STYLE="FONT-STYLE: OBLIQUE">An oblique header (which  
should appear the same as the italicized header above)</H1>
</BODY>
</HTML>
```

This property is fully supported in Internet Explorer, but only **NORMAL** and **ITALIC** are recognized within Netscape Navigator.



**Figure 9-8** The three values for FONT-STYLE displayed.

This is one of those of CSS properties where you really have to wonder why anyone bothered to create it. It was likely done in order to round out the various font family properties, but it is unlikely to replace the italics (<I>) tag in usage, whose functions it emulates. The tag form is much easier to remember and to write than its CSS equivalent — and for the moment, it is better supported within the major browsers than its CSS equivalent.

## The FONT-VARIANT Property

The rather obscurely named FONT-VARIANT essentially sets whether or not a font should be displayed in small capital letters (better known as “small caps”).

### ***FONT-VARIANT***

Description: Determines how text is displayed using capital letters.

CSS Family Type: Font

Values:

- NORMAL | SMALL-CAPS
- These values are toggles to turn the small-caps effect on and off.

Sample code:

```
<H1 STYLE="FONT-VARIANT: SMALL-CAPS">this text is
displayed in small capital letters</H1>
```

**Table 9-4 FONT-VARIANT Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Safe

There are only two values for FONT-VARIANT: NORMAL (the default value) and SMALL-CAPS.

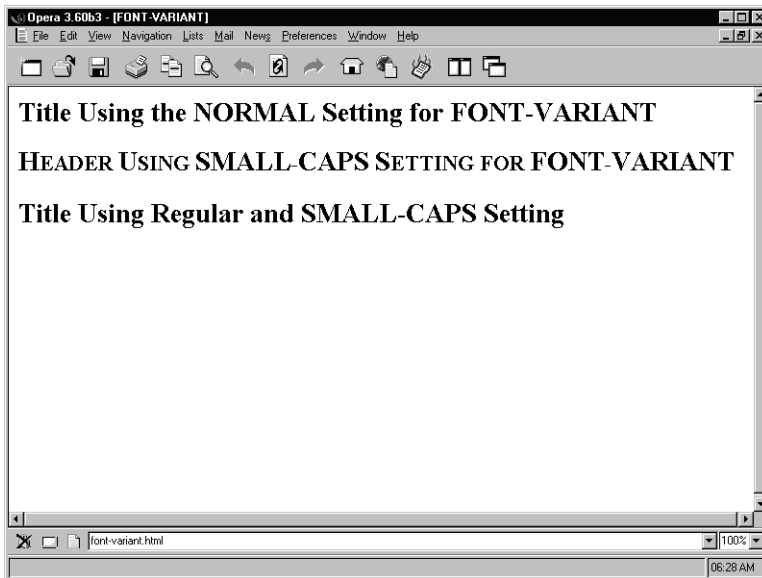
This property is not supported in Netscape Navigator, and while it is supported in Internet Explorer, the SMALL-CAPS value is not as robust as it ought to be according to the CSS specification. Take a look at the following code sample, and then see how it is displayed in both Internet Explorer and Opera.

**Listing 9-4 FONT-VARIANT**

```
<HTML>
<HEAD>
<TITLE>FONT-VARIANT</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: XX-LARGE">
<P>
<B STYLE="FONT-VARIANT: NORMAL">Title Using the NORMAL
Setting for FONT-VARIANT</B>
<P>
<B STYLE="FONT-VARIANT: SMALL-CAPS">Header Using SMALL-CAPS
Setting for FONT-VARIANT</B>
```

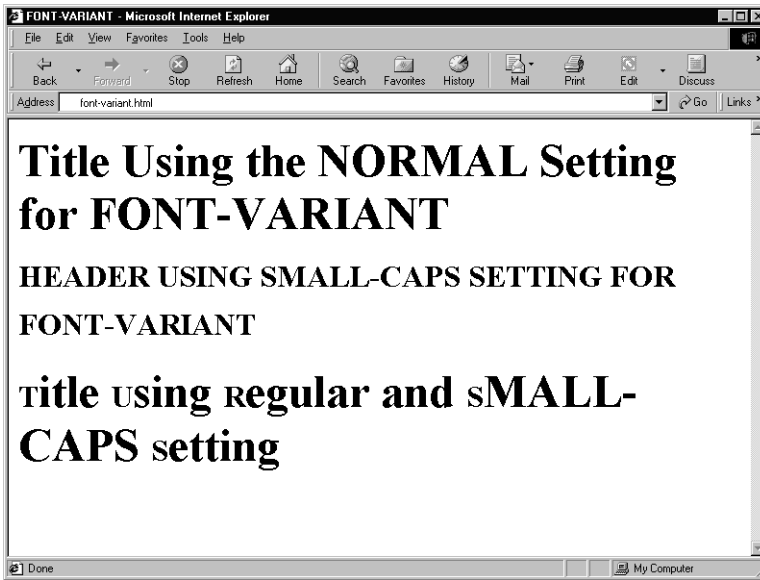
**Listing 9-4 FONT-VARIANT (continued)**

```
<P>
<B STYLE="FONT-VARIANT: SMALL-CAPS">T</B><B>itle</B>
<B STYLE="FONT-VARIANT: SMALL-CAPS">U</B><B>sing</B>
<B STYLE="FONT-VARIANT: SMALL-CAPS">R</B><B>egular and</B>
<B STYLE="FONT-VARIANT: SMALL-CAPS">S</B><B>MALL-CAPS</B>
<B STYLE="FONT-VARIANT: SMALL-CAPS">S</B><B>etting</B></B>
</BODY>
</HTML>
```



**Figure 9-9** The three values for FONT-STYLE displayed in Opera.

Notice that in Internet Explorer FONT-STYLE: SMALL-CAPS is displayed in all caps. This is not exactly according to specifications, although it is good enough to qualify as a match. Opera, on the other hand, displays the same code more in the spirit of the specification, with true small caps. It also does not capitalize the first letter of the article “for”. To drive the point home, notice the major difference in the way the final line of code is displayed between the two browsers. In Opera, the line height of the letters is the same (as it should be), whereas in Internet Explorer they are of different sizes.



**Figure 9-10** The three values for FONT-STYLE displayed in Internet Explorer.

Despite this, Internet Explorer does at least recognize and attempt to display the difference, and for this it can be said to conform to the specification. Opera just does a better job of it.

## The FONT-WEIGHT Property

The FONT-WEIGHT property is used to set the “thickness” of a given font. Think of it as the “bold” counterpart to FONT-STYLE, with more options.

### **FONT-WEIGHT**

Description: Sets the thickness of the displayed font.

CSS Family Type: Font

Values:

- 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 - Absolute font weight values.
- BOLD | NORMAL - Sets whether or not the specified text element uses a bold face.

- **BOLDER | LIGHTER** - Changes the weight of the current specified element relative to that of a previously specified value.

Sample code:

```
<I STYLE="FONT: BOLD">This italicized text is also
set to bold using CSS.</I>
```

**Table 9-5 FONT-WEIGHT Support in Major Browsers**

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> (Mac)	<i>IE 4.5</i> (Mac)	(Win. & UNIX)	<i>NN 4.x</i>	(Mac & UNIX)	<i>Opera</i> 3.6
Partial	Partial	Partial	Partial	Partial	Partial	Partial	Partial

There are three ways you can set **FONT-WEIGHT**: as an absolute value ranging from 100 to 900, as a relative measure to the default value, and as an absolute name value.

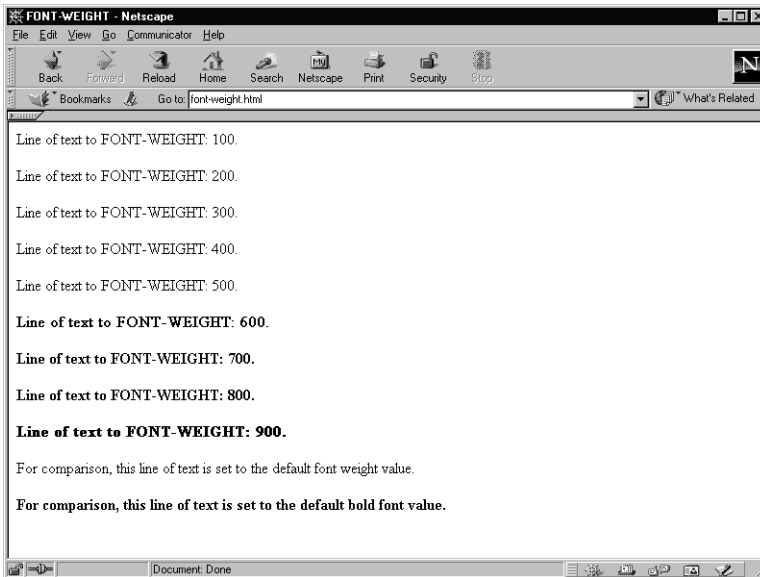
The first way introduces a new concept to Web authors, that of absolute numerical values for the weight of a font. In typographical terms, the values roughly correspond to the following series of values from smallest to heaviest: Book, Medium, Bold, Heavy, Black and Extra Black. You may have noticed that this series is not composed of nine different values, and this has been taken into account in the official CSS specification. Wherever “holes” appear in font weight, the browser is supposed to fill them in by representing the missing values in the same way as either the next available value up or down. According to the specification, the value 400 is to correspond to the normal, default weight, and 700 to the weight of bold. The major browsers render these two values correctly, but there is no smooth progression from the least-to most-heavy weight, as can be seen in the following illustration of this test code:

```
<P STYLE="FONT-WEIGHT: 100">Line of text to FONT-
WEIGHT: 100.</P>
<P STYLE="FONT-WEIGHT: 200">Line of text to FONT-
WEIGHT: 200.</P>
<P STYLE="FONT-WEIGHT: 300">Line of text to FONT-
WEIGHT: 300.</P>
<P STYLE="FONT-WEIGHT: 400">Line of text to FONT-
WEIGHT: 400.</P>
<P STYLE="FONT-WEIGHT: 500">Line of text to FONT-
WEIGHT: 500.</P>
```

```

<P STYLE="FONT-WEIGHT: 600">Line of text to FONT-
WEIGHT: 600.</P>
<P STYLE="FONT-WEIGHT: 700">Line of text to FONT-
WEIGHT: 700.</P>
<P STYLE="FONT-WEIGHT: 800">Line of text to FONT-
WEIGHT: 800.</P>
<P STYLE="FONT-WEIGHT: 900">Line of text to FONT-
WEIGHT: 900.</P>
<P>For comparison, this line of text is set to the
default font weight value.
<P><B>For comparison, this line of text is set to
the default bold font value.</B>

```



**Figure 9-11** The nine numerical values for FONT-WEIGHT displayed in Netscape Navigator.

As you can see, there are really only three different font weights supported, and there is no smooth progression of font weight to be seen. It should be noted that this behavior is consistent between both Internet Explorer and Netscape Navigator.

FONT-WEIGHT also has two relative name values, **BOLDER** and **LIGHTER**, which work in the same way as the **LARGER** and **SMALLER** relative name values associated with FONT-SIZE. **BOLDER** is supposed to make the selected text heavier than the default, and **LIGHTER** is supposed



to make the selected text less heavy than the default. You can see how this works as the following code is displayed in Internet Explorer:

### Listing 9-5 FONT-WEIGHT 2

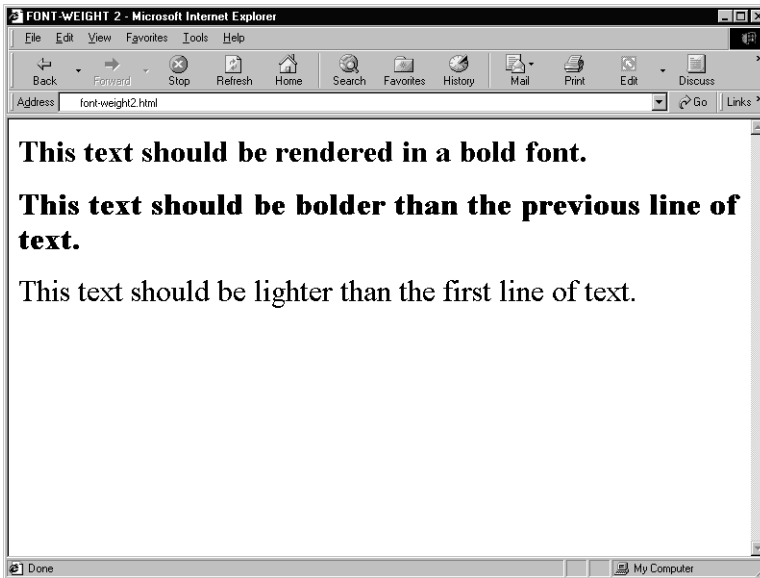
```
<HTML>
<HEAD>
<TITLE>FONT-WEIGHT 2</TITLE>
<STYLE>
P {FONT-WEIGHT: BOLD; FONT-SIZE: X-LARGE;}
</STYLE>
</HEAD>
<BODY>
<P>This text should be rendered in a bold font.</P>
<P STYLE="FONT-WEIGHT: BOLDER;">This text should be bolder
than the previous line of text.</P>
<P STYLE="FONT-WEIGHT: LIGHTER;">This text should be
lighter than the first line of text.</P>
</BODY>
</HTML>
```

---

Keep in mind when using the **BOLDER** and **LIGHTER** that they are relative to the base font weight that has been set; they are not relative to each other. If that were the case, the **SMALLER** value seen in Fig. 9–12 would be the same weight as the first line, which is set to the default value of bold. Internet Explorer supports **BOLDER** and **LIGHTER** fully, but Netscape Navigator supports neither.

Finally, **FONT-WEIGHT** also supports the two absolute name values **NORMAL** and **BOLD**. These two values do pretty much what you would expect: they turn bolding “on” or “off” for selected text. The following code and illustration show these two values in action:

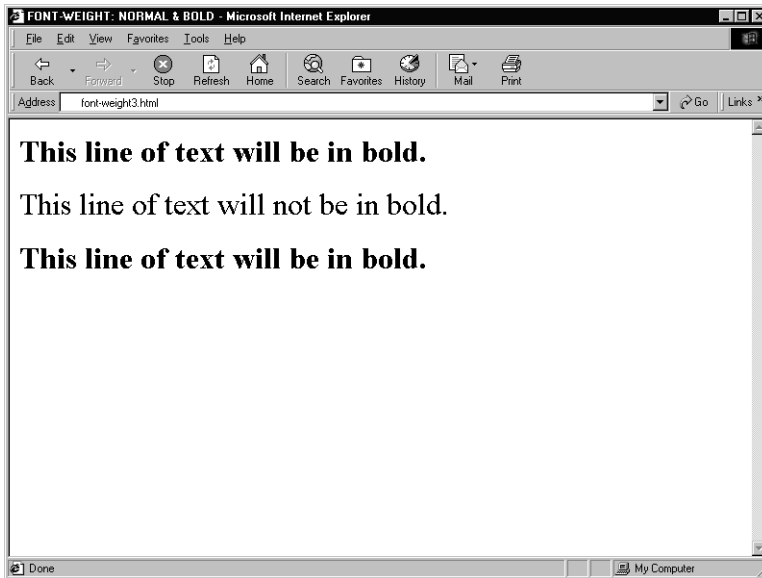
```
<B>This line of text will be in bold.</B>
<P>
<B STYLE="FONT-WEIGHT: NORMAL;">This line of text
will not be in bold.</B>
<P>
<B STYLE="FONT-WEIGHT: BOLD;">This line of text
will be in bold.</B>
```



**Figure 9-12** The BOLDER and LIGHTER values for FONT-WEIGHT displayed in Internet Explorer.

Figure 9-13 shows off the effects of the NORMAL and BOLD settings: NORMAL turns off what would otherwise be bolded text, and BOLD turns it back on. Notice that the BOLD setting does not make the bolded text any bolder than it would otherwise be; compare it to the first line of bold text that appears without being modified by any CSS properties. Both of these values are supported in the latest versions of Internet Explorer and Netscape Navigator.

In terms of usage, it is particularly unfortunate that the full range of absolute numerical values are not supported, as that would at least give Web authors a greater range over the “weight” of text that could be displayed on screen. Since the CSS specification does not insist that the browser implement the full range of values in order to be considered compliant, it is unlikely we will see the major browsers bothering to adopt this feature fully anytime soon.



**Figure 9-13** The NORMAL and BOLD values for FONT-WEIGHT displayed.

## The FONT Property

This is a “shortcut” property that allows the Web author to quickly set fonts in the manner possible under the FONT-FAMILY, FONT-SIZE, FONT-STYLE, FONT-VARIANT and FONT-WEIGHT CSS properties. Essentially, you can do anything with the FONT value that you can do with any of the other font family properties. If you are planning on using a number of font family properties in your code, you may find this simple but versatile CSS property useful to make all the font settings you want.

### **FONT**

**Description:** This is an abbreviated property that allows the Web author to quickly set fonts in the manner possible under the FONT-FAMILY, FONT-SIZE, FONT-STYLE, FONT-VARIANT and FONT-WEIGHT CSS properties.

CSS Family Type: Font

Values:

- CURSIVE | FANTASY | MONOSPACE | SANS-SERIF | SERIF - Sets the type of visual characteristics of font “family” name that can be specified.
- *font\_family* - The name of the font to be displayed
- LARGE | MEDIUM | SMALL | X-LARGE | X-SMALL | XX-LARGE | XX-SMALL - Specifies the size of the text. Values range from XX-SMALL (smallest) to XX-LARGE (largest).
- LARGER | SMALLER - Changes the size of the current specified element relative to a previously specified value.
- *n* value - Specific unit value for the specified element.
- % value – Sets a percentage for the size of a font relative to the base font size.
- ITALIC | NORMAL - Determines whether or not text is italicized
- OBLIQUE - Sets an italic-like property if the font used is sans-serif.
- 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 - Absolute font weight values.
- BOLD | NORMAL - Sets whether or not the specified text element uses a bold face.
- BOLDER | LIGHTER - Changes the weight of the current specified element relative to a previously specified value.

Sample code:

**Fantasy font**

### Table 9-6 FONT Support in Major Browsers

[illegible]

The following code and illustration show how you can “compress” your font CSS code into a single, concise statement:

### Listing 9-6 Font

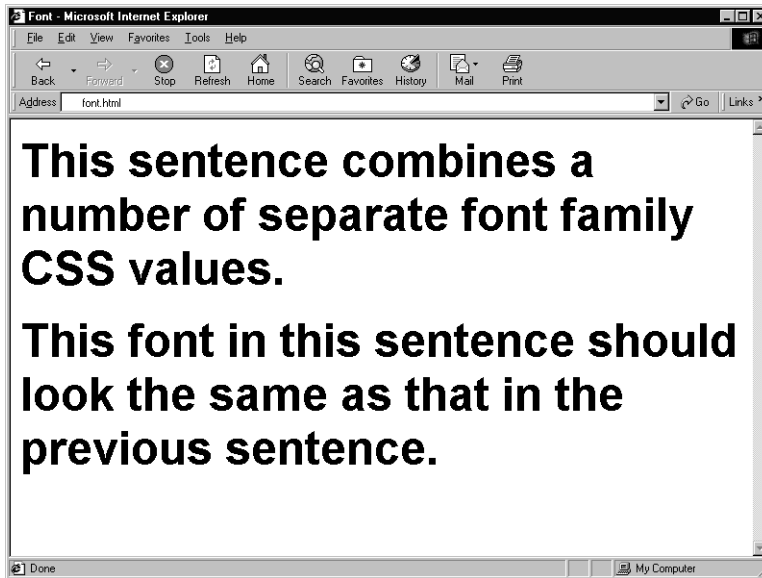
```
<HTML>
<HEAD>
<TITLE>Font</TITLE>
</HEAD>
<BODY>

<B STYLE="FONT-SIZE: 36pt; FONT-WEIGHT: BOLD; FONT-FAMILY:
SANS-SERIF;">
This sentence combines a number of separate font family CSS
values.</B>
<P>
<B STYLE="FONT: BOLD 36pt SANS-SERIF;">This font in this
sentence should look
the same as that in the previous sentence.</B>

</BODY>
</HTML>
```

---

The FONT property is not fully supported in either Netscape Navigator or Internet Explorer, since not all of its sub-properties are fully supported. There is an additional problem when using the FONT property in that both major browsers are highly sensitive to the ordering of the modifiers. If, instead of using `<B STYLE="FONT: BOLD 40pt SANS-SERIF;">` you used `<B STYLE="FONT: 40pt BOLD SANS-SERIF;">`, you would get very different results depending on the browser being used. Internet Explorer renders that code in a 40 point font that is neither bold nor san-serif, while Netscape Navigator only renders the text in bold. Use caution when using this CSS property.



**Figure 9-14** Internet Explorer displaying the sample FONT code.



# THE COLOR AND BACKGROUND FAMILY OF PROPERTIES

## Topics in this Chapter

- The COLOR Property
- The BACKGROUND-COLOR Property
- The BACKGROUND-IMAGE Property
- The BACKGROUND-REPEAT Property
- The BACKGROUND-ATTACHMENT Property
- The BACKGROUND-POSITION Property
- The BACKGROUND Property





# *Chapter* 10

The Color and Background family of properties enable Web authors to add color values to virtually any displayable element, and tile backgrounds exactly according to specifications. The set of color and background properties provide a much broader range of choices for Web authors than can be had with plain old HTML.

This family consists of the following properties, which are examined in this chapter:

- COLOR
- BACKGROUND-COLOR
- BACKGROUND-IMAGE
- BACKGROUND-REPEAT
- BACKGROUND-ATTACHMENT
- BACKGROUND-POSITION
- BACKGROUND

## The COLOR Property

The COLOR CSS property simply sets the color to be displayed for the HTML tag with which it is associated. It can take any of the color value types

that can be associated with CSS. For example, it can take a standard color name (such as “BLUE”) or a hexadecimal color value (such as “#0000FF”, which is blue), both of which are standard ways of setting color values in HTML. In CSS, you can also specify color using a “compressed” hexadecimal color value (such as “#00F”, which is blue). You can also use a three-digit RGB (“Red Green Blue”) color value (where RGB(0,0,255) is blue), and a three-digit RGB percentage color value (where RGB(0%,0%,100%) is blue).

COLOR works in the same way as the COLOR attribute does with the <FONT> tag or a number of other HTML tags. The following code displays HTML and CSS code that produce the same results, which can be seen in Figure 10-1:

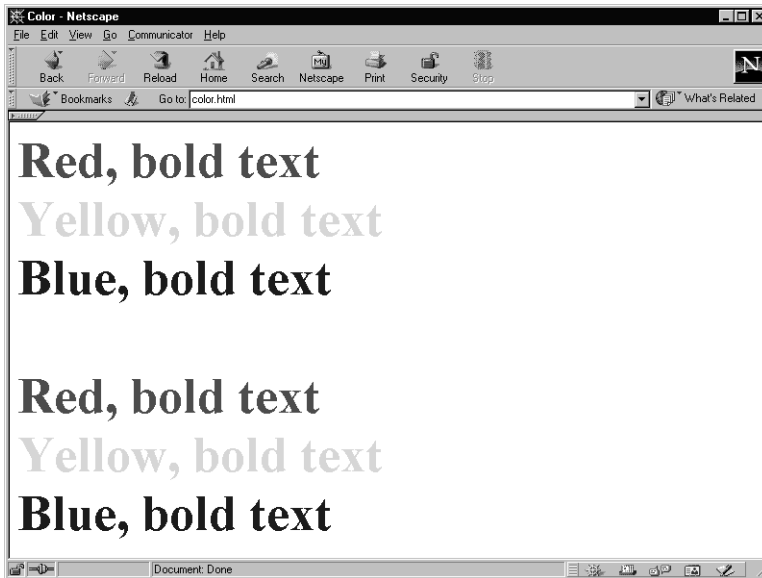
### Listing 10-1 Color

```
<HTML>
<HEAD>
<TITLE>Color</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: XX-LARGE";>
<FONT COLOR="RED"><B>Red, bold text</B></FONT><BR>
<FONT COLOR="YELLOW"><B>Yellow, bold text</B></FONT><BR>
<FONT COLOR="BLUE"><B>Blue, bold text</B></FONT>
<P>
<B STYLE="COLOR: RED">Red, bold text</B><BR>
<B STYLE="COLOR: YELLOW">Yellow, bold text</B><BR>
<B STYLE="COLOR: BLUE">Blue, bold text</B>
</BODY>
</HTML>I Meet
```

---

The COLOR property is supported in the latest versions of both Netscape Navigator and Internet Explorer, and is considered “safe” to use.

A final note of warning to non-American readers of this book: make sure you use the American spelling of the word “COLOR” — the British spelling (“COLOUR”) simply will not work in CSS code.



**Figure 10-1** HTML and CSS color values displayed in Netscape Navigator.

## The BACKGROUND-COLOR Property

The BACKGROUND-COLOR property sets the background color of an HTML element, and for the most part functions in exactly the same way as the BGCOLOR attribute most commonly associated with the <BODY> tag.

### **BACKGROUND-COLOR**

Description: Sets the background color of an element.

CSS Family Type: Color and Background

Values:

- *color name* or numerical color value
- TRANSPARENT - renders the specified element transparent

Sample code:

```
<I STYLE="BACKGROUND-COLOR: FFFF00">Italicized
text on a yellow background.</I>
```

Table 10-1 BACKGROUND-COLOR Support in Major Browsers

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Safe

What makes this property so versatile is that it can be added to virtually any HTML tag. Here’s a Web page that shows some examples of how it can be added to many different HTML tags, plus an illustration of it in action:

Listing 10-2 Background-Color

```
<HTML>
<HEAD>
<TITLE>Background-Color</TITLE>
</HEAD>
</HTML>

<BODY STYLE="BACKGROUND-COLOR: RGB(0,0,0); FONT-SIZE: X-LARGE">

<I STYLE="BACKGROUND-COLOR: #FFFF00">Italicized text on a
yellow background.</I>

<H1 STYLE="BACKGROUND-COLOR: AQUA">A Light Blue Header</H1>

<I STYLE="BACKGROUND-COLOR: GREEN">Text with a green
background.
<B STYLE="BACKGROUND-COLOR: TRANSPARENT">This should be in
green too.</B>
More text with a green background.</I>

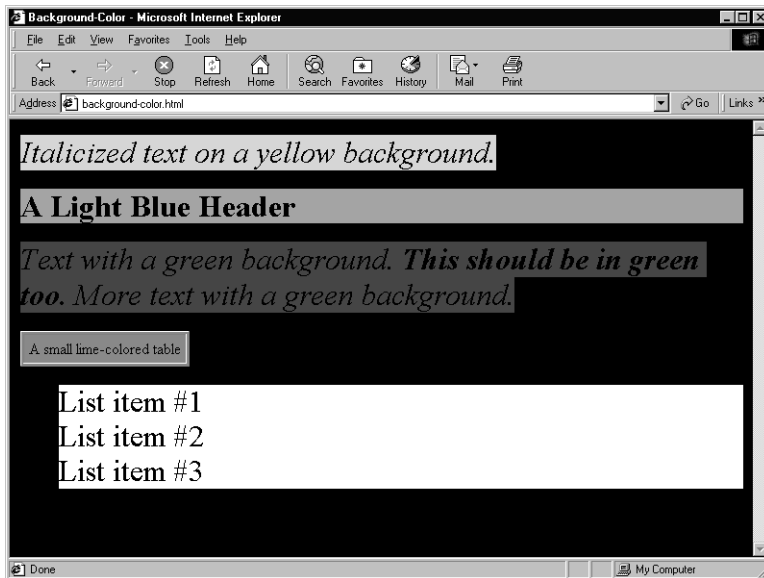
<P>
```

**Listing 10-2 Background-Color (continued)**

```
<TABLE BORDER CELLPADDING="5" STYLE="BACKGROUND-COLOR: LIME">
<TR>
<TD>A small lime-colored table</TD>
</TR>
</TABLE>

<OL STYLE="BACKGROUND-COLOR: WHITE">
<LI>List item #1
<LI>List item #2
<LI>List item #3
</OL>

</BODY>
</HTML>
```



**Figure 10-2** BACKGROUND-COLOR associated with various HTML elements on a Web page.

This property is fully supported in Internet Explorer, and largely supported in Netscape Navigator too. The glitch in Netscape Navigator comes

with how it supports the `TRANSPARENT` value. In the official specification, a `TRANSPARENT` setting is supposed to let the background color “shine through”. For example, the following code snippet (which is part of the code used in Figure 10–2) is not displayed the same way in both browsers:

```
<I STYLE="BACKGROUND-COLOR: GREEN">Text with a  
green background.  
<B STYLE="BACKGROUND-COLOR: TRANSPARENT">This  
should be in green too</B>  
More text with a green background.</I>
```

The sentence that uses the `TRANSPARENT` value should also be in green because its “parent” (i.e., the element of the Web page to which it belongs, in this case the enclosing `<I>` tags) is also set to green. Netscape instead shows the underlying black of the Web page — dramatic perhaps, but this is not the way it is supposed to work.

## The `BACKGROUND-IMAGE` Property

`BACKGROUND-IMAGE` is designed to set a graphic image as a background image for an element. It works in much the same manner as does the `BACKGROUND` attribute to the `<BODY>` tag, but its CSS counterpart can be added to more than just the `<BODY>` tag.

### ***BACKGROUND-IMAGE***

Description: Sets a graphic image as a background image.

CSS Family Type: Color and Background

If this element is specified, the CSS properties `BACKGROUND-REPEAT`, `BACKGROUND-ATTACHMENT` and `BACKGROUND-POSITION` can also be used with it. A color value can also be added.

Values:

- `URL(image)`

Sample code:

```
<BODY STYLE="BACKGROUND-IMAGE: URL(canvas.gif)">
```

**Table 10-2 BACKGROUND-IMAGE Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Safe

It takes a URL value in a manner demonstrated in the following piece of code:

```
<BODY STYLE="BACKGROUND-IMAGE: URL(picture.gif)">
```

The BACKGROUND-IMAGE property can be associated with any HTML tag that can be displayed on screen. For example, the following Web page (illustrated in Figure 10-3) gives you an idea of how it can be used:

**Listing 10-3 Background-Image**

```
<HTML>
<HEAD>
<TITLE>Background-Image</TITLE>
</HEAD>
</HTML>

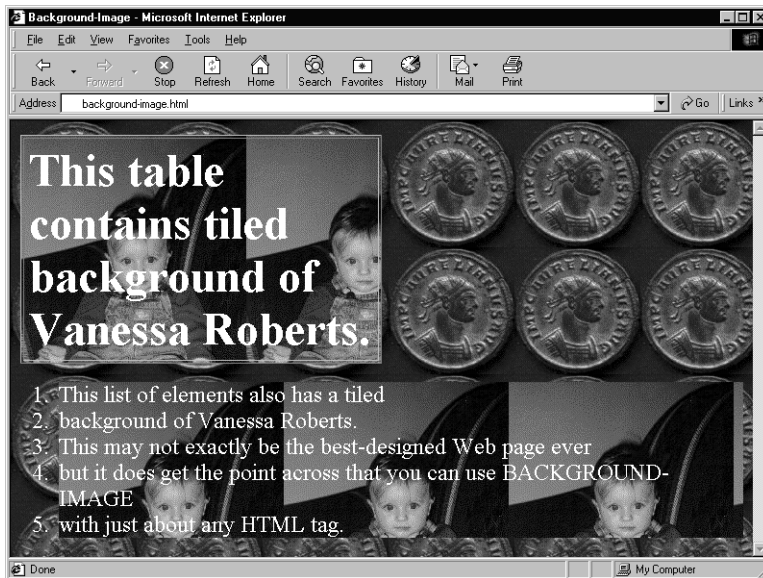
<BODY STYLE="BACKGROUND-IMAGE: URL(aurelian.jpg);">
<P>
<TABLE BORDER WIDTH="50%" CELLPADDING="5"
STYLE="BACKGROUND-IMAGE: URL(vanessa.jpg);">
<TR>
<TD><B STYLE="FONT-SIZE: XX-LARGE; COLOR: WHITE;">This
table contains tiled background of Vanessa
Roberts.</B></TD>
</TR>
</TABLE>
<P>
```

**Listing 10-3 Background-Image (continued)**

```

<OL STYLE="BACKGROUND-IMAGE: URL(vanessa.jpg); FONT-SIZE:
LARGE; COLOR: WHITE;">
<LI>This list of elements also has a tiled
<LI>background of Vanessa Roberts.
<LI>This may not exactly be the best-designed Web page ever
<LI>but it does get the point across that you can use
BACKGROUND-IMAGE
<LI>with just about any HTML tag.
</OL>
</BODY>
</HTML>

```



**Figure 10-3** BACKGROUND-IMAGE used associated with several different HTML tags.

This property is fully supported in both Internet Explorer and Netscape Navigator, so it is safe to use. It does exhibit one quirk in Netscape Navigator, in that it pretty much has to be applied to an image file that is contained in the same directory as the Web page. This can be annoying in circumstances



where you want to use a universal style sheet throughout a Web site using many directory levels, although this can be got around by using an absolute URL (i.e., `URL(http://www.vebulon.ca/picture.gif)`, for example). This is all very well when you are running and testing your code directly from a Web server, but can pose a problem if you are running on a local system that does not have a Web server. (For more information on using URLs in CSS, see “URLs” in Chapter 7, “CSS Units”).

The CSS properties `BACKGROUND-REPEAT`, `BACKGROUND-ATTACHMENT` and `BACKGROUND-POSITION` can also be used to modify the display characteristics of `BACKGROUND-IMAGE`. These will all be treated separately in this chapter.

## The BACKGROUND-REPEAT Property

`BACKGROUND-REPEAT` is always used in conjunction with the `BACKGROUND-IMAGE` property, and it determines how an image will be tiled on a Web page. The default value is `REPEAT`, which works the same way that `<BODY BACKGROUND="image.gif">` does by simply repeating the background image both across and down the Web page. `BACKGROUND-REPEAT` comes with two other values, `REPEAT-X` and `REPEAT-Y`, which tile the image horizontally and vertically, respectively.

### ***BACKGROUND-REPEAT***

Description: If the `BACKGROUND-IMAGE` property is set, this additional value specifies how the image is repeated on the Web page.

CSS Family Type: Color and Background

Values:

- `REPEAT` - Image is horizontally and vertically tiled.
- `REPEAT-X` - Image is horizontally tiled.
- `REPEAT-Y` - Image is vertically tiled.
- `NO-REPEAT` - The image is not repeated.

Sample code:

```
<BODY STYLE="BACKGROUND-IMAGE: URL(gordian3.jpg);  
BACKGROUND-REPEAT: REPEAT-X">
```

**Table 10-3 BACKGROUND-REPEAT Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Partial	Partial	Safe	Safe	Partial	Partial	Safe

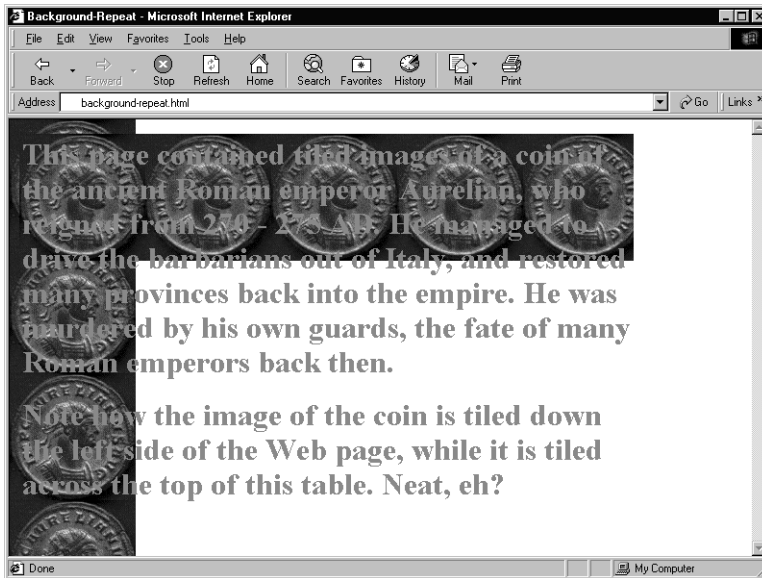
As an example, the following Web page code tiles the image file in a strip that runs down right side of the Web page, and also across the top of a table:

**Listing 10-4 Background-Repeat**

```
<HTML>
<HEAD>
<TITLE>Background-Repeat</TITLE>
</HEAD>

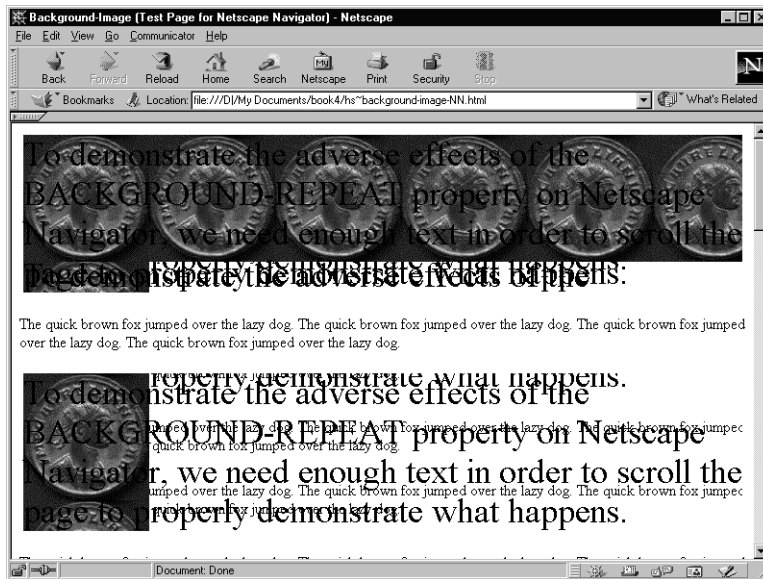
<BODY STYLE="BACKGROUND-IMAGE: URL(aurelian.jpg); BACKGROUND-
REPEAT: REPEAT-Y;">

<TABLE STYLE="BACKGROUND-IMAGE: URL(aurelian.jpg); BACKGROUND-
REPEAT: REPEAT-X;">
<TR>
<TD>
<B STYLE="FONT-SIZE: X-LARGE; COLOR: LIME;">This page contained
tiled images of a coin of the ancient Roman emperor Aurelian, who
reigned from 270 - 275 AD. He managed to drive the barbarians out of
Italy, and restored many provinces back into the empire. He was
murdered by his own guards, the fate of many Roman emperors back
then.
<P>
Note how the image of the coin is tiled down the left side of the
Web page, while it is tiled across the top of this table. Neat, eh?
</B>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```



**Figure 10-4** The effects of BACKGROUND-REPEAT as seen within Internet Explorer.

This property is supported fully in the most recent versions of Internet Explorer and Netscape Navigator, but Netscape is given only a “Partial” rating. This is because it is definitely not *safe* to use this property in Netscape Navigator when you are planning on using multiple instances of the BACKGROUND-REPEAT property on a single page. Figure 10-5 demonstrates what happens when you have multiple instances where the BACKGROUND-REPEAT property is used, and the page is then scrolled.



**Figure 10-5** The effects of multiple BACKGROUND-REPEAT values set within a Web page as viewed within Netscape Navigator for Windows.

The result, as you can see, is a mess — bad enough to make the page unreadable. This effect is not seen in the Unix or Macintosh versions of Netscape Navigator, although they have a problem when BACKGROUND-REPEAT is combined with BACKGROUND-POSITION when the latter is set to a percentage value.

## The BACKGROUND-ATTACHMENT Property

The BACKGROUND-ATTACHMENT property is designed to set how the background image should move when the browser window is scrolled. When you scroll a page in a browser, the typical behavior for the background is that it, the text and any other elements on the Web page scroll in unison. With the BACKGROUND-ATTACHMENT property, you have another choice.

## BACKGROUND-ATTACHMENT

Description: If BACKGROUND-IMAGE property is set, this property specifies how the background image should move when the browser window is scrolled.

CSS Family Type: Color and Background

Values:

- SCROLL - Image moves when the browser window is scrolled.
- FIXED - Image does not move when the browser window is scrolled.

Sample code:

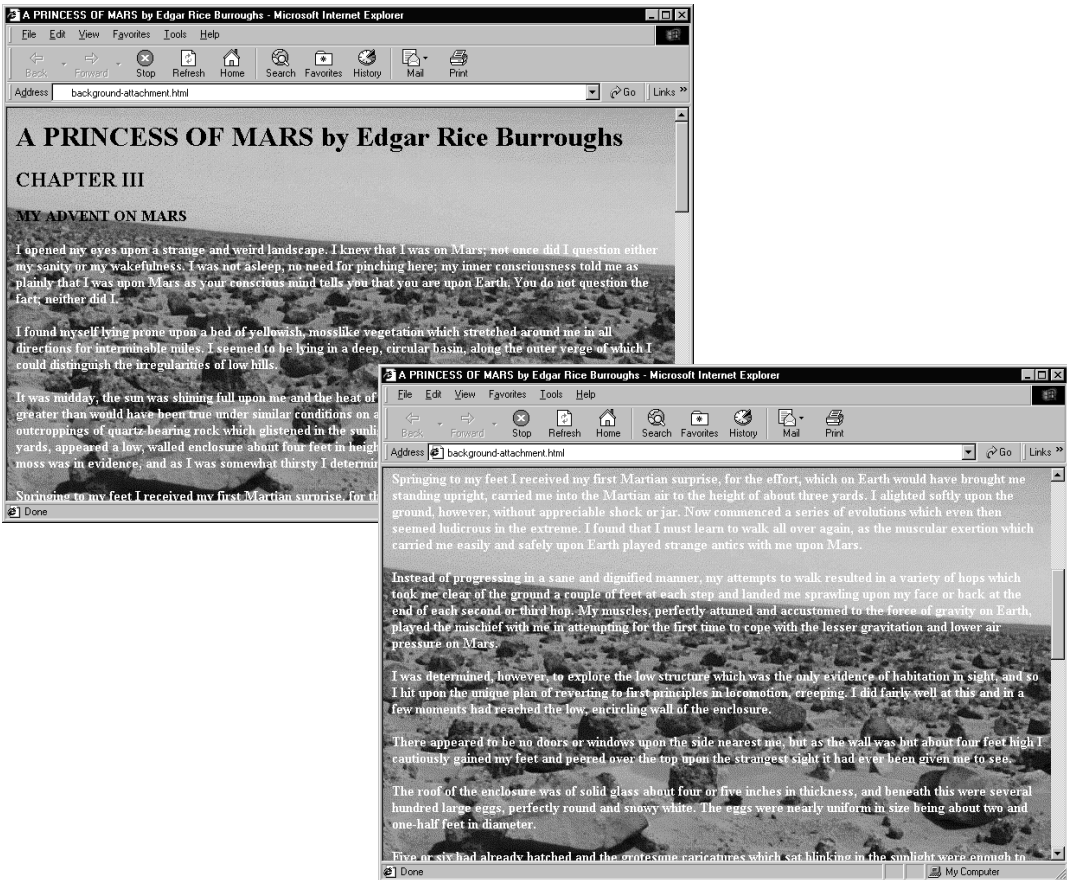
```
<BODY STYLE="BACKGROUND-IMAGE: URL(canvas.gif);
BACKGROUND-ATTACHMENT: FIXED">
```

**Table 10-4 BACKGROUND-ATTACHMENT Support in Major Browsers**

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i>	<i>IE 4.5</i>	<i>(Win. &amp;</i>	<i>NN 4.x</i>	<i>(Mac &amp;</i>	<i>Opera</i>
		<i>(Mac)</i>	<i>(Mac)</i>	<i>UNIX)</i>		<i>UNIX)</i>	<i>3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Safe

The BACKGROUND-ATTACHMENT property is always associated with BACKGROUND-IMAGE and it can take one of two possible values: SCROLL and FIXED. SCROLL is the default, and is what you typically see when you scroll any Web page with a background image – the background scrolls with you. FIXED is supposed to keep the background in place while the browser moves the text and other elements of the Web page instead. You can see the effect of this in Figure 10-6: the text in the Internet Explorer scrolls, but the background does not. The effect makes it seem as though the text is “floating” above the background image.

Internet Explorer has an HTML equivalent for this CSS property: <BODY BGPROPERTIES="FIXED">, whereas Netscape Navigator has never supported this extension to the <BODY> tag. Not surprisingly, therefore, Internet Explorer supports the BACKGROUND-ATTACHMENT value fully, while Netscape Navigator only partially supports it: essentially, it supports the default SCROLL value, but not FIXED. This qualifies as partial support for the



**Figure 10-6** BACKGROUND-ATTACHMENT used to fix the background image in place as seen within Internet Explorer.

property, but the FIXED value is the only “interesting” value for BACKGROUND-ATTACHMENT.

Although it has been around for awhile, the BGPROPERTIES="FIXED" attribute for the HTML tag <BODY> is little used, so many users are initially surprised when they see the effect in action. When put into effect (using either HTML or CSS) this “trick” can actually be a little unsettling to the viewer, so it is best used sparingly. In addition, as when choosing any background image, make sure that you are choosing one that will not interfere with the readability of your text.

While it is possible to add this property to other HTML tags, it really only makes sense to add the BACKGROUND-ATTACHMENT property to the <BODY> tag. As all other elements on the Web page are affected when you scroll the browser window, you will never see the effects of a value such as BACKGROUND-ATTACHMENT: FIXED displayed on anything but the <BODY> tag.

Despite the fact that this property is only fully supported within Internet Explorer, it is considered safe for use because it will not hinder the display of the background in Netscape Navigator. Just keep in mind that the image cannot be made to remain in place when the browser contents are scrolled, and design your Web page accordingly. In other words, do not create your Web page so that the absence of a non-scrolling background will adversely affect your page's functionality.

## The BACKGROUND-POSITION Property

The BACKGROUND-POSITION property provides a feature that is otherwise impossible in regular HTML: the exact positioning of a background image on a Web page. It is always used in conjunction with the BACKGROUND-IMAGE property, and can take either a precise X,Y pixel value, a percentage value or a special keyword value.

### ***BACKGROUND-POSITION***

Description: If the BACKGROUND-IMAGE property is specified, this property specifies where the image first appears, and then describes how it should be tiled.

CSS Family Type: Color and Background

Values:

- X% Y% - Percentage is in reference to the dimensions of the browser window display.
- X Y - Represents the absolute coordinate position of the image.
- (LEFT/CENTER/RIGHT) | (TOP/CENTER/BOTTOM) - Keywords representing screen positions. Left keyword is the X-position and the right keyword is the Y-position for the image.

Sample code:

```
<BODY STYLE="BACKGROUND-IMAGE:
URL(victorinus.jpg); BACKGROUND-POSITION: 100 50">
```

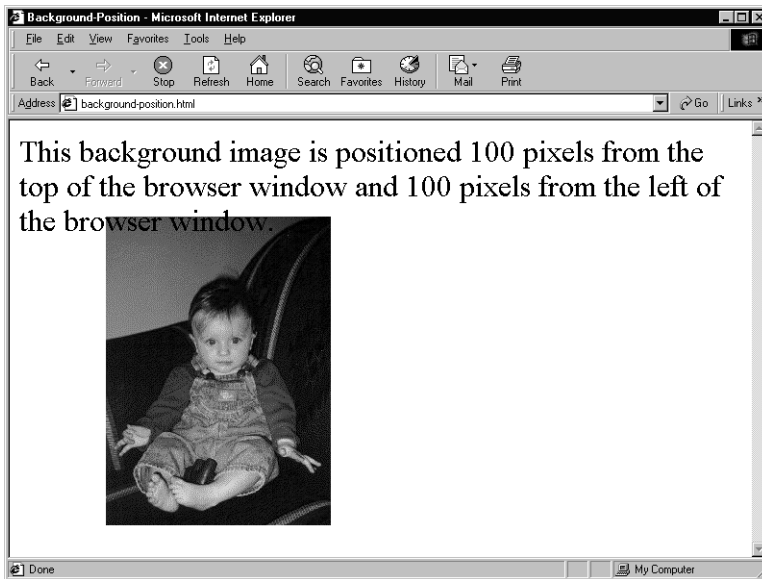
**Table 10-5 BACKGROUND-POSITION Support in Major Browsers**

		<i>IE 4.01</i>	<i>IE 4.5</i>	<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>(Mac)</i>	<i>(Mac)</i>	<i>(Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>(Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Safe

You can set an X,Y value that tells the background image where it should initially be displayed. X is the value in pixels that the image will appear from the top of the browser window, and Y is the number of pixels the image will appear from the left of the browser window. The following code, depicted in Figure 10–7, shows BACKGROUND-POSITION in action:

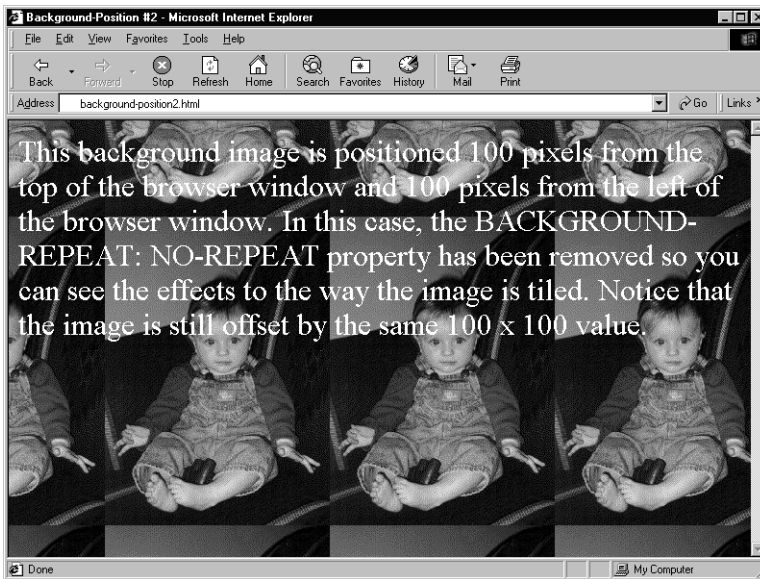
```
<BODY STYLE="BACKGROUND-IMAGE: URL(vanessa.jpg);
BACKGROUND-POSITION: 100 100; BACKGROUND-REPEAT:
NO-REPEAT;">
<P STYLE="FONT-SIZE: X-LARGE;">
This background image is positioned 100 pixels
from the top of the browser window and 100 pixels
from the left of the browser window.
</P>
```





**Figure 10-7** BACKGROUND-POSITION set to a specific X,Y pixel value, with BACKGROUND-REPEAT set to NO-REPEAT.

What this code does is position the image called `vanessa.jpg` and begin tiling it 100 pixels from the right and 100 pixels from the top. Notice that the additional property `BACKGROUND-REPEAT: NO-REPEAT` has been set; this has been done to show off the effect more clearly. When it is removed, the image is simply tiled over the whole of the Web page, with the first image offset by the value determined by `BACKGROUND-POSITION`. You can see the effect when the `BACKGROUND-REPEAT` value is removed in Figure 10-8.

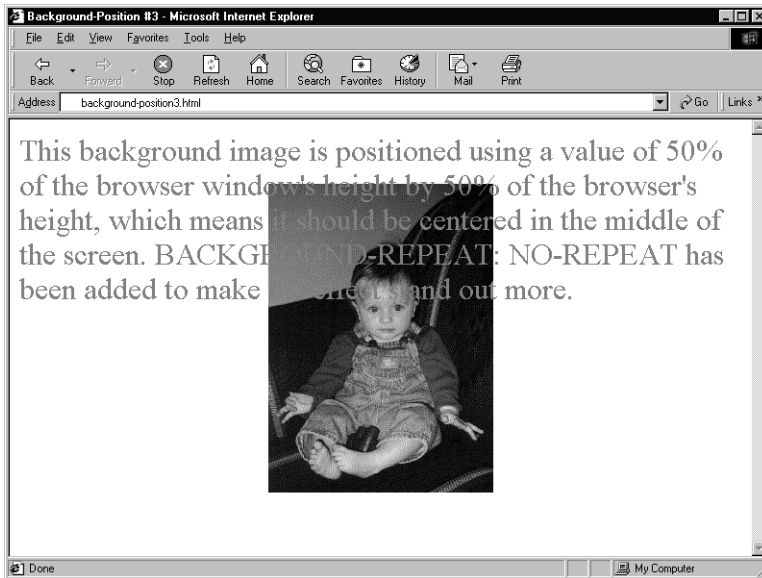


**Figure 10-8** BACKGROUND-POSITION set to a specific X,Y pixel value, with no BACKGROUND-REPEAT value set.

BACKGROUND-POSITION can also take a percentage value, where the X and Y values are set relative to the width of the browser's window. If it is set to a value of 50% for both values, as in the following code sample, the image will be centered in the middle of the browser window.

```
<BODY STYLE="BACKGROUND-IMAGE: URL(vanessa.jpg);
BACKGROUND-POSITION: 50% 50%; BACKGROUND-REPEAT:
NO-REPEAT;">
<P STYLE="FONT-SIZE: X-LARGE; COLOR: GRAY;">
This background image is positioned using a value
of 50% of the browser window's height by 50% of
the browser's height, which means it should be
centered in the middle of the screen.
BACKGROUND-REPEAT: NO-REPEAT has been added to
make the effect stand out more.
</P>
```

A number of keyword values can also be used to position a background image with the BACKGROUND-POSITION property. The keywords “LEFT”, “CENTER” and “RIGHT” can be used for the X value, and “TOP”, “CENTER” and “BOTTOM” can be used for the Y value. These keyword values



**Figure 10-9** BACKGROUND-POSITION set to a value of 50% for both X and Y, centering the background image in the browser's window.

offer a simple shortcut for Web authors who want to ensure that a background image always begins in any of the corners of a browser window or aligned within its center. This has an edge over percentage or pixel values, as you can never be sure how wide your users will size their browser's window. The following code, depicted in Figure 10-10, shows the X and Y values set to RIGHT and BOTTOM, respectively (again, using BACKGROUND-REPEAT: NO-REPEAT):

#### Listing 10-5 Background-Position #4

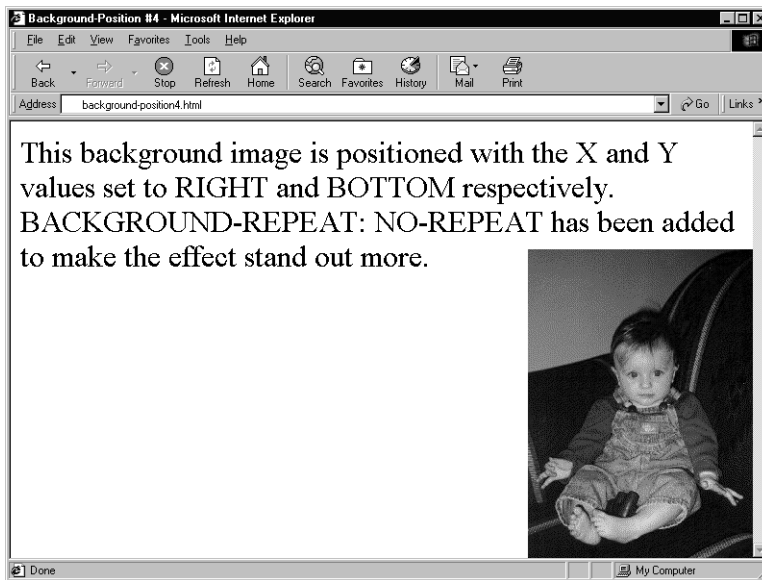
```
<HTML>
<HEAD>
<TITLE>Background-Position #4</TITLE>
</HEAD>

<BODY STYLE="BACKGROUND-IMAGE: URL(vanessa.jpg); BACKGROUND-
POSITION: RIGHT BOTTOM; BACKGROUND-REPEAT: NO-REPEAT;">
```

**Listing 10-5 Background-Position #4 (continued)**

```
<P STYLE="FONT-SIZE: X-LARGE;">  
This background image is positioned with the X and Y values  
set to RIGHT and BOTTOM respectively.  
BACKGROUND-REPEAT: NO-REPEAT has been added to make the  
effect stand out more.  
</P>  
  
</BODY>  
</HTML>
```

---



**Figure 10-10** BACKGROUND-POSITION set to a RIGHT BOTTOM with NO-REPEAT set for BACKGROUND-REPEAT.

This property is arguably most-effective when combined with the `BACKGROUND-REPEAT` property (as can be seen in the many code examples in this section). If you want the background image to stand out on your Web page, you can use the `BACKGROUND-POSITION` and `BACKGROUND-REPEAT` properties effectively for this purpose.

Of the two major browsers, the BACKGROUND-POSITION property is only recognized within Internet Explorer. When this code is tried in Netscape Navigator, it simply tiles the image as it would any other background image, completely ignoring the BACKGROUND-POSITION property.

## The BACKGROUND Property

The BACKGROUND property is a “shortcut” property, allowing Web authors quick access to all of the values contained in the rest of the BACKGROUND family of properties, namely: BACKGROUND-COLOR, BACKGROUND-IMAGE, BACKGROUND-REPEAT, BACKGROUND-ATTACHMENT and BACKGROUND-POSITION. It works in much the same manner as the FONT property in providing quick access to all of the values contained in the font family.

### **BACKGROUND**

Description: Sets the background color or image. It can take on any value contained in the background family of properties.

CSS Family Type: Color and Background

Values:

- *color name* or *numerical color value*
- URL(image)
- SCROLL - Image moves when the browser window is scrolled.
- FIXED - Image does not move when the browser window is scrolled.
- TRANSPARENT – renders the specified element transparent
- X% Y% - Percentage is in reference to the dimensions of the browser window display.
- X Y - Represents absolute coordinate position of the image.
- (LEFT/CENTER/RIGHT) | (TOP/CENTER/BOTTOM) - Keywords representing screen positions. Left keyword is the X-position and the right keyword is the Y-position for the image.
- REPEAT - Image is horizontally and vertically tiled.
- REPEAT-X - Image is horizontally tiled.
- REPEAT-Y - Image is vertically tiled.
- NO-REPEAT - The image is not repeated.

Sample code:

```
<B STYLE="BACKGROUND: GREEN">This text is displayed
against a green background.</B>
```

**Table 10-6 BACKGROUND Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Partial	Partial	Partial	Safe	Safe	Partial	Partial	Safe

All of the values associated with `BACKGROUND` property have already been discussed in detail with each of the other background family properties. `BACKGROUND` is a handy shortcut for setting all of the values you need quickly. The following two code samples are equivalent to each other, and are depicted in Figure 10-11:

Code Sample #1:

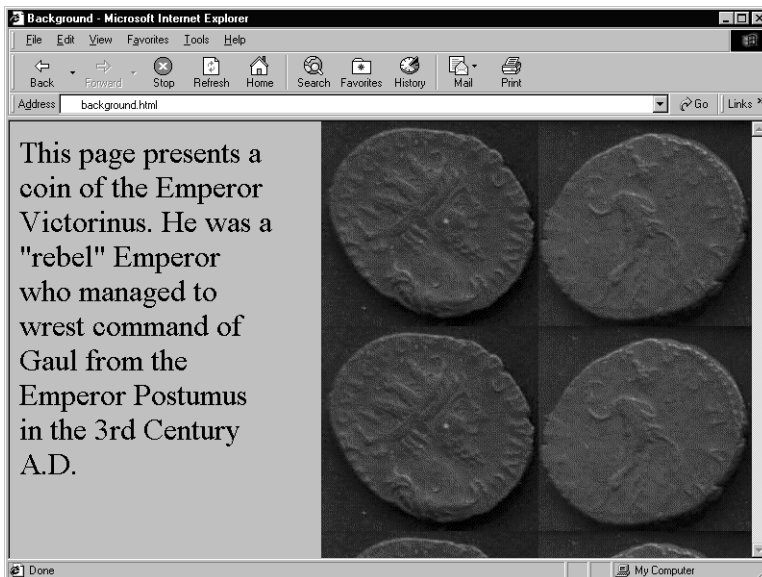
**Listing 10-6 Background**

```
<HTML>
<HEAD>
<TITLE>Background</TITLE>
</HEAD>
<BODY STYLE="BACKGROUND-COLOR: SILVER; BACKGROUND-IMAGE:
URL(victorinus.jpg); BACKGROUND-REPEAT: REPEAT-Y;
BACKGROUND-ATTACHMENT: FIXED; BACKGROUND-POSITION: TOP
RIGHT;">
<P STYLE="FONT-SIZE: X-LARGE; MARGIN-RIGHT: 65%">
This page presents a coin of the Emperor Victorinus. He was
a "rebel" Emperor who managed to wrest command of Gaul from
the Emperor Postumus in the 3rd Century A.D.
</P>
</BODY>
</HTML>
```

## Code Sample #2

**Listing 10-7 Background**

```
<HTML>
<HEAD>
<TITLE>Background</TITLE>
</HEAD>
<BODY STYLE="BACKGROUND: SILVER URL(victorinus.jpg) REPEAT-
Y FIXED TOP RIGHT">
<P STYLE="FONT-SIZE: X-LARGE; MARGIN-RIGHT: 65%">
This page presents a coin of the Emperor Victorinus. He was
a "rebel" Emperor who managed to wrest command of Gaul from
the Emperor Postumus in the 3rd Century A.D.
</P>
</BODY>
</HTML>
```



**Figure 10-11** The BACKGROUND property "in action".

As you can easily see, the second code sample is much more concise. This property is fully supported in the most recent versions of Internet Explorer, but only partially supported in Netscape Navigator (which only fully supports the `BACKGROUND-IMAGE` property).


Unlike the somewhat finicky `FONT` shortcut property, it is possible to rearrange the ordering of the various values for `BACKGROUND` without interfering with the presentation of the elements displayed on screen.





# THE TEXT FAMILY OF PROPERTIES

## Topics in this Chapter

- 
- The LETTER-SPACING Property
  - The WORD-SPACING Property
  - The LINE-HEIGHT Property
  - The VERTICAL-ALIGN Property
  - The TEXT-ALIGN Property
  - The TEXT-DECORATION Property
  - The TEXT-INDENT Property
  - The TEXT-TRANSFORM Property

# *Chapter*

# 11

The family of text properties enables Web authors to change the manner in which text is displayed on screen. They differ from the font family of properties in that the text family is designed to accomplish a number of typographical display options, irrespective of the font being used. Using the text family of properties, you can alter the spacing between lines, words or individual letters in your text, change the alignment of your text and much more.

The text family is composed of the following properties, all of which will be examined in detail in this chapter:

- LETTER-SPACING
- WORD-SPACING
- LINE-HEIGHT
- VERTICAL-ALIGN
- TEXT-ALIGN
- TEXT-DECORATION
- TEXT-INDENT
- TEXT-TRANSFORM

## The LETTER-SPACING Property

The `LETTER-SPACING` property does just that: it adds spacing between individual letters on a Web page. It can take one of two possible values: the default keyword value `NORMAL`, or a specific measurement value.

When used effectively, the `LETTER-SPACING` property enables the Web author to help spread out what text they have across the Web page. This technique is common in page layout programs to better space words across a single line of text, usually to fully justify it (i.e., to have even left and right margins). This technique may not fully translate to the Web, since it is exceedingly difficult to know where a specific word will appear on screen, making it hard to fully justify the line of text in this manner. The `LETTER-SPACING` property can be used effectively for eye-catching displays, however.

### LETTER-SPACING

Description: Specifies the spacing value between letters.

CSS Family Type: Text

Values:

- `NORMAL` - The browser's default letter-spacing value.
- *length (units)* - Sets the length value between letters as a unit of measurement.

Sample code:

```
<B STYLE="LETTER-SPACING: 1cm">Some wide letter  
spacing!</B>
```

**Table 11-1 LETTER-SPACING Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Safe

The `LETTER-SPACING` property can take any unit of measure that has been allotted to CSS (see the CSS Units chapter for more detailed information on which units of measure can be used).

The best use for the LETTER-SPACING property is for emphasis. It is great for making your headers stand out, or for emphasizing particular words or phrases in your text. It is tricky to use LETTER-SPACING for its intended typographical purpose of fully justifying text across a line, and that use is not recommended (If this is your goal, TEXT-ALIGN is better suited for this purpose.)

The following code example shows some good and not-so-good examples of the LETTER-SPACING property in action:

### Listing 11-1 Letter-Spacing

```
<HTML>
<HEAD>
<TITLE>Letter-Spacing</TITLE>
</HEAD>

<BODY STYLE="FONT-SIZE: LARGE;">

<H1 ALIGN="CENTER" STYLE="LETTER-SPACING: 0.5CM;">This
Header Stands Out</H1>
<H2 ALIGN="CENTER" STYLE="LETTER-SPACING: 0.3CM;">On the
Web, Letter Spacing is Best Used for Emphasis</H2>

<P>
```

Again, within a line of regular text, the LETTER-SPACING property is best used for `<EM STYLE="LETTER-SPACING: 0.2CM;">emphasis</EM>`, as it really stands out from the rest of the text being displayed.

```
<P>
```

In typographical circles, letter spacing is most often used `<B STYLE="LETTER-SPACING: 0.1CM; FONT-WEIGHT: NORMAL;">to` help a line of text to be fully justified across the width of the page. However, `<B STYLE="LETTER-SPACING: 0.1CM; FONT-WEIGHT: NORMAL;">on the Web it</B>` is hard to know exactly where a line of text will be placed on the page.

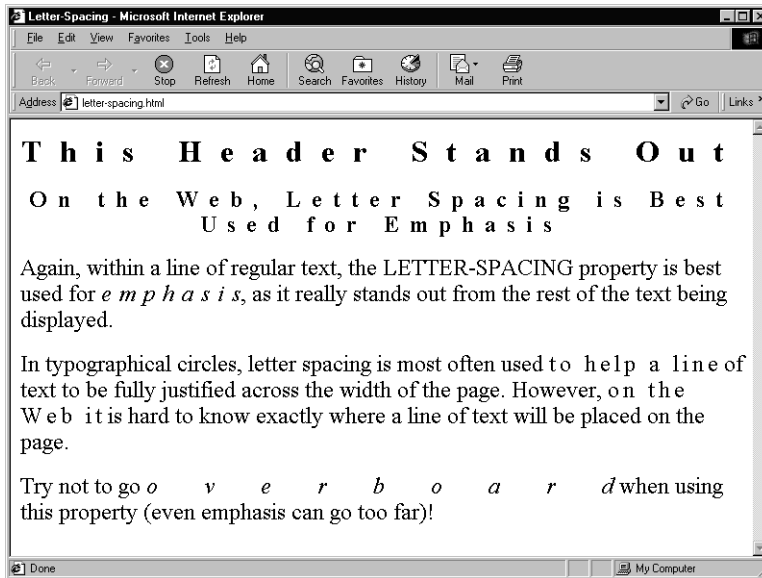
### Listing 11-1 Letter-Spacing (continued)

```
<P>
```

```
Try not to go <EM STYLE="LETTER-SPACING:
0.5IN;">overboard</EM> when using this property (even
emphasis can go too far)!
```

```
</BODY>
```

```
</HTML>
```



**Figure 11-1** The use and abuses the LETTER-SPACING property can be put to.

Figure 11-1 gives you a good idea as to how the LETTER-SPACING property should and should not be used. When used in moderation, the LETTER-SPACING property helps to emphasize headers or words in a line of text. It does not hold up well when trying to justify a line of text (as it is impossible to know the exact width of the browser window your viewers will be using), or when you use an extreme value to space out the individual letters. Use this property sparingly.

LETTER-SPACING is fully supported in Internet Explorer, but not at all within Netscape Navigator. Accordingly, if you plan on using this property, do not use it in such a way that meaning will be lost to viewers using Netscape Navigator. When using the LETTER-SPACING property in a paragraph, a good compromise would be to use it in conjunction with tags (such as <EM>, <B> or <I> ) that will still convey the emphasis you want to get across to the reader.

## The WORD-SPACING Property

What the LETTER-SPACING property does for letters, the WORD-SPACING property does for words. Again, it was designed primarily for typographical reasons, typically to fully justify text across a line, but as we have already seen in the example for the LETTER-SPACING property, this use is not recommended for the Web. Instead, the WORD-SPACING property is best used for emphasis. Unfortunately, the WORD-SPACING property is not yet supported in either of the two major browsers.

### WORD-SPACING

Description: Sets the spacing between words.

CSS Family Type: Text

Values:

- **NORMAL** - The browser default word spacing value is used.
- ***length (units)*** - Sets the length value between words as a unit of measurement.

Sample code:

```
<I STYLE="WORD-SPACING: 1cm">The words in this
sentence are all spaced 1cm apart.</I>
```

**Table 11-2 WORD-SPACING Support in Major Browsers**

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i>	<i>IE 4.5</i>	<i>(Win. &amp;</i>	<i>NN 4.x</i>	<i>(Mac &amp;</i>	<i>Opera</i>
		<i>(Mac)</i>	<i>(Mac)</i>	<i>UNIX)</i>		<i>UNIX)</i>	<i>3.6</i>
Unsafe	Unsafe	Safe	Safe	Unsafe	Unsafe	Unsafe	Safe

The WORD-SPACING property can take any unit of measure that has been allotted to CSS (see the CSS Units chapter for more detailed information on what unit of measure can be used).

Again, like the LETTER-SPACING property, the best use for the WORD-SPACING property is for emphasis. It is great for making your headers stand out, or for emphasizing particular words or phrases in your text. It is tricky to use LETTER-SPACING for its intended typographical purpose of fully justifying text across a line, so that particular use is not recommended.

The following code example and accompanying illustration displays WORD-SPACING put to effective and not-so-effective uses:

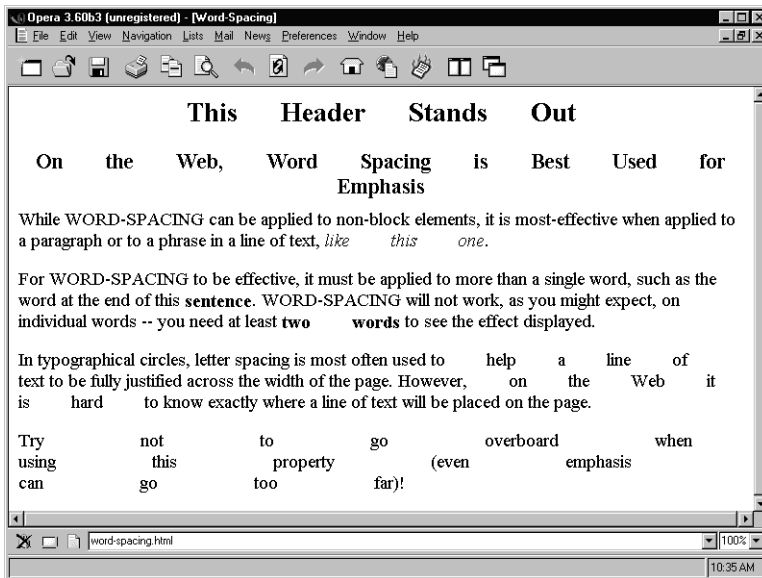
### Listing 11-2 Word-Spacing

```
<HTML>
<HEAD>
<TITLE>Word-Spacing</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE;">
<H1 ALIGN="CENTER" STYLE="WORD-SPACING: 2CM;">This Header
Stands Out</H1>
<H2 ALIGN="CENTER" STYLE="WORD-SPACING: 2CM;">On the Web, Word
Spacing is Best Used for Emphasis</H2>
While WORD-SPACING can be applied to non-block elements, it is
most-effective when applied to a paragraph or to a phrase in a
line of text, <I STYLE="WORD-SPACING: 2CM;">like this one</I>.
<P>
For WORD-SPACING to be effective, it must be applied to more
than a single word, such as the word at the end of this <B
STYLE="WORD-SPACING: 2CM;">sentence</B>. WORD-SPACING will not
work, as you might expect, on individual words -- you need at
least <B STYLE="WORD-SPACING: 2CM;">two words</B> to see the
effect displayed.
<P>
In typographical circles, letter spacing is most often used <B
STYLE="WORD-SPACING: 2CM; FONT-WEIGHT: NORMAL;">to help a line
of text</B> to be fully justified across the width of the
page. <B STYLE="WORD-SPACING: 2CM; FONT-WEIGHT:
NORMAL;">However, on the Web it is hard to</B> know exactly
where a line of text will be placed on the page.
```



## Listing 11-2 Word-Spacing (continued)

```
<P STYLE="WORD-SPACING: 5IN;">
Try not to go overboard when using this property (even
emphasis can go too far)!
</P>
</BODY>
</HTML>
```



**Figure 11-2** The effects of WORD-SPACING as seen within the Opera browser.

Using code similar to that used in the LETTER-SPACING section, Figure 11-2 shows how WORD-SPACING should and should not be used. Again, it is best used for emphasis. Notice that you have to have at least two words for the effects of WORD-SPACING to be seen — it will not work on a single word in isolation. It does not hold up well when trying to justify a line of text (since it is impossible to know the exact width of the browser window your viewers will be using). (Again, if your aim is at justifying the text, TEXT-ALIGN is more appropriate for the job.) When it is set to an extreme value, as can be seen in the final paragraph displayed on the page, it can make the resulting text hard to read.

The `WORD-SPACING` property is not well supported in the two major browsers. It is not supported in any version of Netscape Navigator, and is only supported in the recent Macintosh versions of Internet Explorer, so its use is not recommended. The property is fully supported in the most recent version of the Opera browser.

## The `LINE-HEIGHT` Property

The `LINE-HEIGHT` property is designed to allow a Web author to set the distance between lines of text on a Web page. Using this property, you can space out the text that appears on your Web page, creating effects like “double-spacing” on your Web pages. It can take one of three sets of values: the default `NORMAL` value (equivalent to “single-spacing”), a specific height value using a measurement value, or a percentage value relative to the font’s height.

### ***LINE-HEIGHT***

Description: Sets the distance between lines of text on a Web page.

CSS Family Type: Font

Values:

- `NORMAL` - Sets the line height to the default value used by the browser.
- *`n`* - Sets a multiple value for the current height of the font in use.
- *%* - Sets a percentage value in relation to the size of the current font in use.
- *`length (units)`* - Specifies the length value as a unit of measurement.

Sample code:

```
Baseline<BR>  
<H1 STYLE="LINE-HEIGHT: 2.5in">This appears 2  
inches down from the break element at the top of  
the page!</H1>
```

**Table 11-3 LINE-HEIGHT Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Partial	Safe	Safe	Safe	Safe	Safe	Safe	Safe

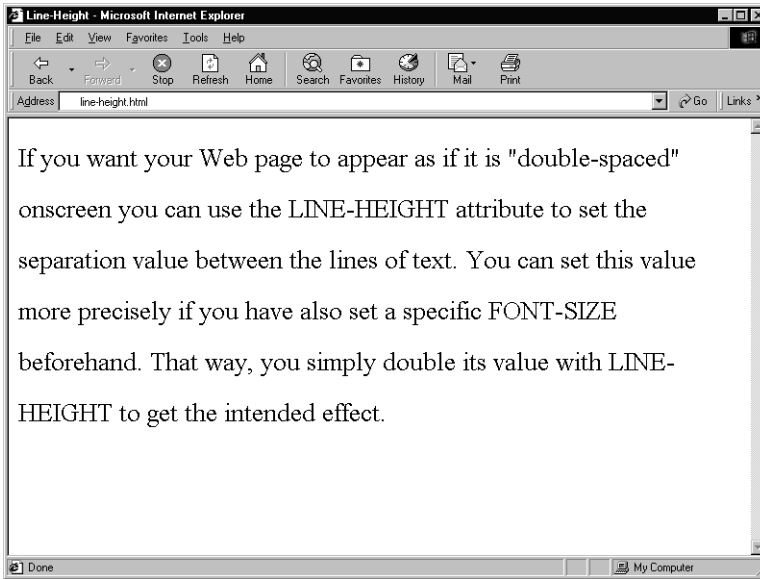
The `LINE-HEIGHT` property can take any unit of measure that has been allotted to CSS (see the CSS Units chapter for more detailed information on which units of measure can be used).

If you are aiming for a “double-spacing” look on your Web page, it is best to use the `LINE-HEIGHT` property in conjunction with the `FONT-SIZE` property. That way, you can simply set the exact size for the font of the text to be displayed, and then simply double that value for `LINE-HEIGHT` to get the intended effect. You can see that effect in the following code, which is also displayed in Figure 11-3:

**Listing 11-3 Line-Height**

```
<HTML>
<HEAD>
<TITLE>Line-Height</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: 20PT;">
<P STYLE="LINE-HEIGHT: 40PT;">If you want your Web page to
appear as if it is "double-spaced" on-screen you can use
the LINE-HEIGHT attribute to set the separation value
between the lines of text. You can set this value more
precisely if you have also set a specific FONT-SIZE
beforehand. That way, you simply double its value with
LINE-HEIGHT to get the intended effect.
</P>
</BODY>
</HTML>
```

Another way to achieve the same effect quickly and easily is to simply use a percentage value. The percentage value you set is relative to the height of the existing font size, so if you set `LINE-HEIGHT` to a value of 200%, it will always

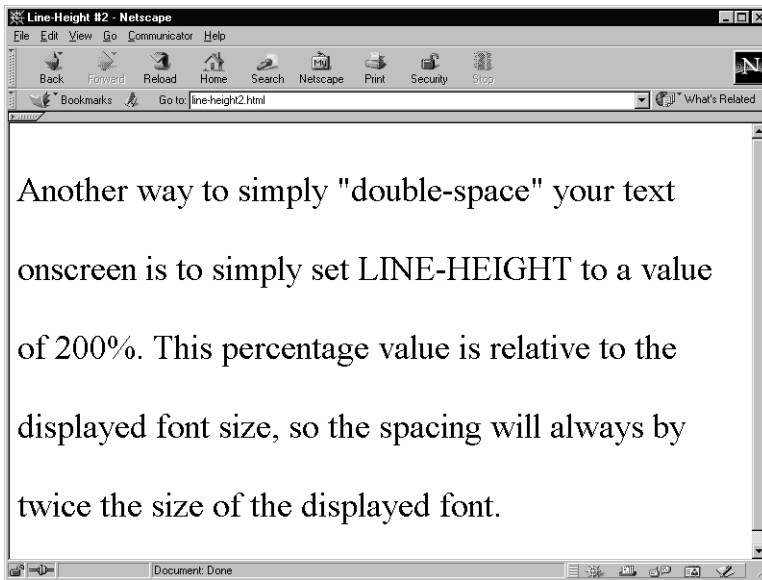


**Figure 11-3** Using FONT-SIZE and LINE-HEIGHT to set up “double-spacing” on a Web page.

display the text on the Web page as double-spaced. This effect can be seen in the following code example, which is displayed in Figure 11-4:

#### Listing 11-4 Line-Height #2

```
<HTML>
<HEAD>
<TITLE>Line-Height #2</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: X-LARGE;">
<P STYLE="LINE-HEIGHT: 200%;">Another way to simply
"double-space" your text onscreen is to simply set LINE-
HEIGHT to a value of 200%. This percentage value is
relative to the displayed font size, so the spacing will
always be twice the size of the displayed font.
</P>
</BODY>
</HTML>
```



**Figure 11-4** Setting LINE-HEIGHT to a value of 200% in order have “double-spacing” on a Web page.

The LINE-HEIGHT property is supported in the recent versions of Internet Explorer and Netscape Navigator, and is therefore safe to use. If you are using specific units of measure when using LINE-HEIGHT, remember that you should not leave a space between the value and the unit of measure (i.e., use “12PT” not “12 PT”) or your code will be unsafe with Netscape Navigator.

## The VERTICAL-ALIGN Property

The VERTICAL-ALIGN property is designed to let a Web author align the placement of text on a Web page with respect the vertical placement of another element on the Web page — most typically an image. It belongs to the text family of CSS properties and its numerous values equate to various positioning values normally associated with text.

VERTICAL-ALIGN

Description: Sets the vertical positioning of the HTML tag with which it is associated.

Media Group: Visual  
CSS2 Family Type: Visual formatting (Details)  
Values:

- BASELINE | BOTTOM | MIDDLE | SUB | SUPER | TEXT-TOP | TEXT-BOTTOM | TOP - Aligns the text element in relation to the specified value.
- *n* value - Sets a length unit value.
- % value – Sets a length percentage value relative to the line-height of the containing block.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<B STYLE="VERTICAL-ALIGN: SUB">Subscript-aligned
text</B> <I STYLE="FONT-SIZE: 16pt">Large,
italicized text.</I>
<B STYLE="VERTICAL-ALIGN: SUPER">Superscript-
aligned text</B> <I STYLE="FONT-SIZE:
16pt">Large, italicized text.</I>
```

Table 11-4 VERTICAL-ALIGN Support in Major Browsers

				IE 5.0		NN 4.0	
IE 3.02	IE 4.x	IE 4.01 (Mac)	IE 4.5 (Mac)	(Win. & UNIX)	NN 4.x	(Mac & UNIX)	Opera 3.6
Unsafe	Partial	Partial	Partial	Partial	Unsafe	Unsafe	Partial

The VERTICAL-ALIGN property can take any one of eight different values: BASELINE, MIDDLE, SUB, SUPER, TEXT-TOP, TEXT-BOTTOM, BOTTOM and TOP. BASELINE aligns the baseline of the element to that of the parent element, while MIDDLE aligns the text to the middle of the associated element. SUB and SUPER are the equivalent of subscript and superscript. TEXT-TOP aligns the top of the element to the top of the parent, and TEXT-BOTTOM does the same thing, but to the bottom of the parent element instead. BOTTOM and TOP are two additional values designed to be used in relation to the existing line of text on which the element is situated, so

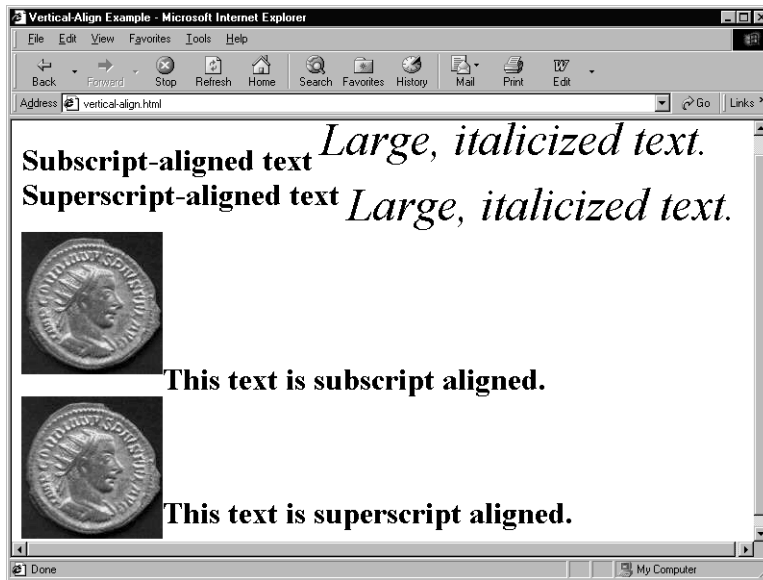
that TOP aligns the element to the tallest element on that line of text, and bottom does the same for the lowest element.

VERTICAL-ALIGN is a good property for aligning text alongside such things as images, and many of its values have direct counterparts to the <IMG> tag in the HTML 4.0 specification. Unfortunately, VERTICAL-ALIGN is not well supported. In Internet Explorer, only the SUB and SUPER values work, and it does not work at all in Netscape Navigator. The following code example (displayed in Figure 11–5 in Internet Explorer 5.0) shows the SUB and SUPER values at work:

### Listing 11-5 Vertical-Align Example

```
<HTML>
<HEAD>
<TITLE>Vertical-Align Example</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: X-LARGE;">
<B STYLE="VERTICAL-ALIGN: SUB">Subscript-aligned text</B>
<I STYLE="FONT-SIZE: XX-LARGE">Large, italicized text.</I><BR>
<B STYLE="VERTICAL-ALIGN: SUPER">Superscript-aligned
text</B> <I STYLE="FONT-SIZE: XX-LARGE">Large, italicized
text.</I><BR>
<IMG SRC="gordian3.jpg"><B STYLE="VERTICAL-ALIGN:
SUB">This text is subscript aligned.</B><BR>
<IMG SRC="gordian3.jpg"><B STYLE="VERTICAL-ALIGN:
SUPER">This text is superscript aligned.</B><BR>
</BODY>
</HTML>
```

---



**Figure 11-5** The SUB and SUPER values of VERTICAL-ALIGN displayed in Internet Explorer 5.0.

## The TEXT-ALIGN Property

The TEXT-ALIGN property is a very handy property for quickly aligning text on a Web page. It can take one of four values: LEFT, RIGHT, CENTER and JUSTIFY. It is designed to be applied to block level elements, such as headers or paragraphs, rather than inline formatting elements (such as <B> or <I>, for example), although as we shall see, Netscape Navigator goes beyond the specification in this regard.

### TEXT-ALIGN

Description: Sets the alignment of the text.

CSS Family Type: Text



Values:

CENTER | JUSTIFY | LEFT | RIGHT - Aligns the text contained by the specified element.

Sample code:

```
<P STYLE="TEXT-ALIGN: CENTER">Centered text.</P>
<P STYLE="TEXT-ALIGN: JUSTIFY">Justified text.</P>
<P STYLE="TEXT-ALIGN: LEFT">Left-aligned text.</P>
<P STYLE="TEXT-ALIGN: RIGHT">Right-aligned text.</P>
```

**Table 11-5 TEXT-ALIGN Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Partial	Safe	Partial	Partial	Safe	Safe	Safe	Safe

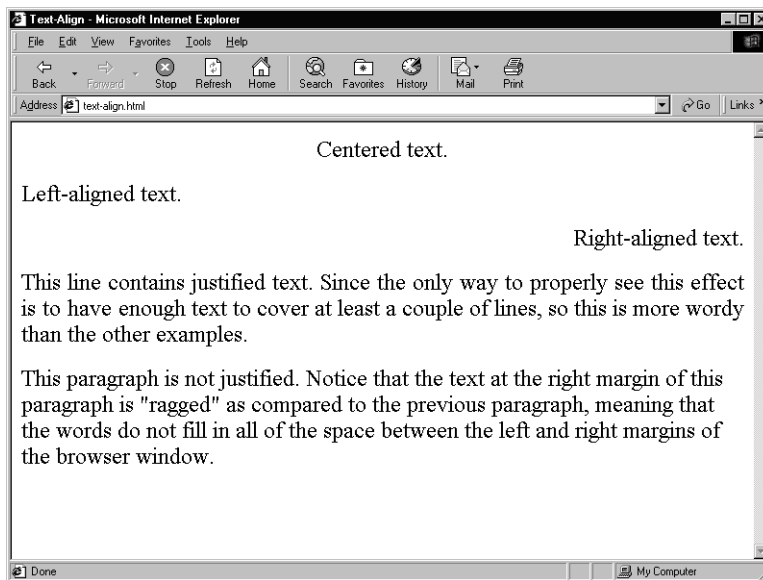
The following code and subsequent illustration show how the TEXT-ALIGN property can be used to format block level elements (in this case, paragraphs) within Internet Explorer:

**Listing 11-6 Text-Align**

```
<HTML>
<HEAD>
<TITLE>Text-Align</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE;">
<P STYLE="TEXT-ALIGN: CENTER">Centered text.</P>
<P STYLE="TEXT-ALIGN: LEFT">Left-aligned text.</P>
<P STYLE="TEXT-ALIGN: RIGHT">Right-aligned text.</P>
<P STYLE="TEXT-ALIGN: JUSTIFY">This line contains justified
text. Since the only way to properly see this effect is to
have enough text to cover at least a couple of lines, so
this is more wordy than the other examples.</P>
</P>
```

### Listing 11-6 Text-Align (continued)

```
<P>
This paragraph is not justified. Notice that the text at
the right margin of this paragraph is "ragged" as compared
to the previous paragraph, meaning that the words do not
fill in all of the space between the left and right margins
of the browser window.
</P>
</BODY>
</HTML>
```

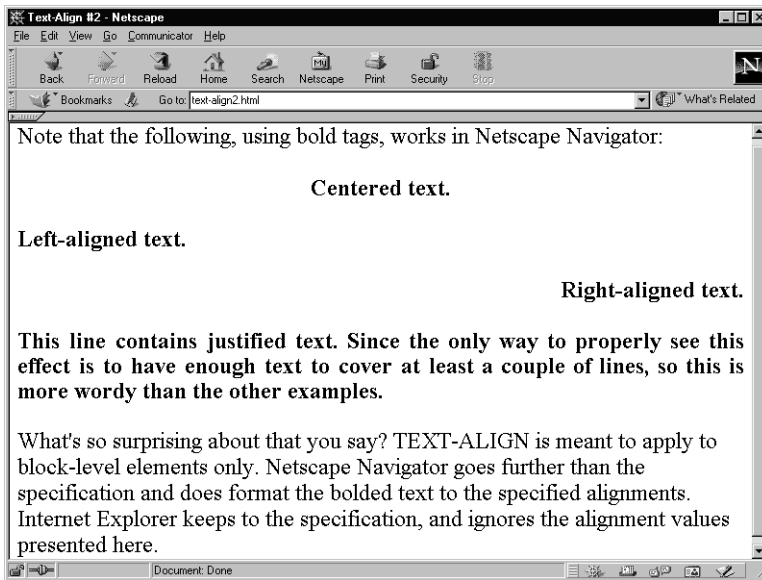


**Figure 11-6** Using the four TEXT-ALIGN values to paragraphs of text in Internet Explorer.

For the most part these values are self-explanatory. For those unfamiliar with the term, justification refers to how text is displayed between the margins. Full justification (which is what the JUSTIFY value sets) means the browser will space the text contained within the paragraph fully across the two margins. Notice how the unjustified sentence that follows the justified

example is “ragged” — the words at the end of the right-margin are not flush against it.

The TEXT-ALIGN property is fully supported in all of the recent Windows versions of Internet Explorer and Netscape Navigator. Internet Explorer for the Macintosh does not support the JUSTIFY value for TEXT-ALIGN. It is worth pointing out that in this case Netscape Navigator supports TEXT-ALIGN beyond the intents of specification by applying the alignment values to non-block elements. The effects of this can be seen in Figure 11-7. This use for TEXT-ALIGN is not recommended, as it may lead to some unexpected results.



**Figure 11-7** Going beyond the specification: The four TEXT-ALIGN values work with non-block level elements in Netscape Navigator.

## The TEXT-DECORATION Property

The `TEXT-DECORATION` property enables Web authors to set “decorative” properties to their text displays. This does not mean, as you might first think, that you can add such things as curlicues or baroque fluting to the individual letters in your text. Instead, `TEXT-DECORATION` is designed to let you add underlines, overlines or strikethroughs to your text, or even make your text blink on and off.

### ***TEXT-DECORATION***

Description: Adds a “decorative” property to the HTML tag specified.

CSS Family Type: Text

Values:

- `BLINK` - Text blinks.
- `LINE-THROUGH` - Text contains a line through the middle (strikethrough text).
- `NONE` - No decoration is added.
- `OVERLINE` - Text is displayed with a line running above it.
- `UNDERLINE` - Text is underlined.

Sample code:

```
<A HREF="fred.html" STYLE="TEXT-DECORATION:
NONE">This link is not underlined</A>

<H1 STYLE="TEXT-DECORATION: OVERLINE">Header with
an Overline</H1>

<H1 STYLE="TEXT-DECORATION: UNDERLINE">Header
with an Underline</H1>

<P STYLE="TEXT-DECORATION: LINE-THROUGH">
Ignore this sentence

<P>

<I STYLE="TEXT-DECORATION: BLINK">This text
blinks</I>
```

**Table 11-6 TEXT-DECORATION Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Partial	Partial	Partial	Partial	Partial	Partial	Partial	Partial

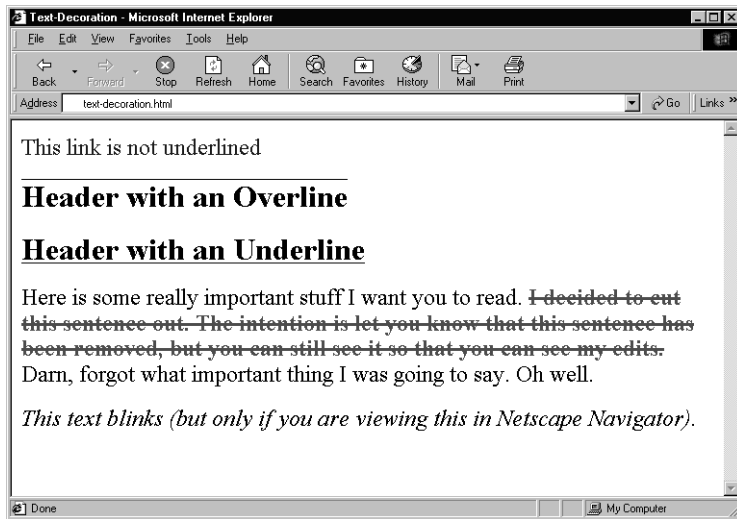
The TEXT-DECORATION property can take any one of five keyword values: BLINK, LINE-THROUGH, OVERLINE, UNDERLINE and NONE. The BLINK value is analogous to setting text within the <BLINK> tag, first instituted in the original version of Netscape Navigator. The tag is considered “infamous” by many, as there is no control over the rate of blinking, and over time it is guaranteed to annoy the user reading the page. Unfortunately, the TEXT-DECORATION property offers no extra control over the rate or duration of blinking. The LINE-THROUGH value behaves in the same way as the little-used <S> tag, which puts a line through the text. Struck-through text is universally recognized to indicate that edits to the text have been made. The OVERLINE value adds a line above the text being displayed, and has no HTML tag equivalent. The UNDERLINE value works in the same way as the little-used <U> tag, placing a line underneath the text. Caution should be used when using this value, as some users may mistake it as a hyperlink, which is also typically underlined. The NONE value, which is the default, can also be put to good use. It can remove the underline normally associated with hyperlinks, so that if you have a need for such a thing, it can easily be accomplished using this value. All of these values are used in the following code example, and the results can be seen in Figure 11-8:

**Listing 11-7 Text-Decoration**

```
<HTML>
<HEAD>
<TITLE>Text-Decoration</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE;">
<A HREF="fred.html" STYLE="TEXT-DECORATION: NONE">This link is
not underlined</A>
```

### Listing 11-7 Text-Decoration (continued)

```
<H1 STYLE="TEXT-DECORATION: OVERLINE">Header with an
Overline</H1>
<H1 STYLE="TEXT-DECORATION: UNDERLINE">Header with an
Underline</H1>
Here is some really important stuff I want you to read.
<B STYLE="TEXT-DECORATION: LINE-THROUGH; COLOR: RED;"> I
decided to cut this sentence out. The intention is let you know
that this sentence has been removed, but you can still see it
so that you can see my edits.</B> Darn, forgot what important
thing I was going to say. Oh well.
<P>
<I STYLE="TEXT-DECORATION: BLINK;">This text blinks (but only
if you are viewing this in Netscape Navigator).</I>
</BODY>
</HTML>
```



**Figure 11-8** Four out of the five values of TEXT-DECORATION displayed in Internet Explorer.

The TEXT-DECORATION property is only partially implemented in the major browsers. The real stickler is the BLINK value. It is supported within Netscape Navigator, which originated the <BLINK> tag. Internet Explorer has

never implemented the effects of the <BLINK> tag, and it does not implement the BLINK value, either. Internet Explorer does implement all of the other values, however. The only value Netscape Navigator does not support is the overline value. In a quirk of Web administrivia, according to the official CSS specification, the BLINK value does not have to be implemented in order for the browser to be said to be compliant with this property. For the sake of clarity however, Internet Explorer is given only a “Partial” rating, even though it can be said otherwise to comply with the stated specifications for this property.

## The TEXT-INDENT Property

The TEXT-INDENT property is used for indenting text from the left margin. It differs in function from the various margin properties in that it only indents the first line of text from the left margin. It can either take a specific measurement value or a percentage value that corresponds to the width of the browser window.

### **TEXT-INDENT**

Description: Specifies indents from the left margin or from a block-element.

CSS Family Type: Text

Values:

- *length (units)* - Specifies the length value as a unit of measurement.
- % - Specifies the distance as a percentage of the browser window display.

Sample code:

```
<H1 STYLE="TEXT-INDENT: 2in">This text is indented  
from the margins by 2 inches.</H1>
```

**Table 11-7 TEXT-INDENT Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Safe	Safe	Safe	Safe	Safe	Safe	Safe	Safe

The `TEXT-INDENT` property can take any unit of measure that has been allotted to CSS (see the CSS Units chapter for more detailed information on which units of measure can be used). It can also take a percentage value, which is relative to the width of the browser window.

The `TEXT-INDENT` property is meant for use with block level elements, such as headers, but both major browsers go beyond the specification to some degree. They can differ when it comes to displaying the same code, however. If an on-screen element begins a fresh line, Internet Explorer will do its best to indent the item. Netscape Navigator will always indent an on-screen element, even from within a sentence. Both also accept negative values, which is according to the specification. The following code and illustration give you an idea as to how `TEXT-INDENT` can be used:

**Listing 11-8 Text-Indent**

```
<HTML>
<HEAD>
<TITLE>Text-Indent</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE;">
<H1 STYLE="TEXT-INDENT: 2IN; FONT-SIZE: LARGER;">This
header is indented by 2 inches.</H1>
<P>
<B STYLE="TEXT-INDENT: 2IN;">So is this bold text.</B>
<P>
<TABLE BORDER STYLE="TEXT-INDENT: 2IN;">
<TR>
<TD>
This table is indented from side of the browser margin by 2
inches.
```

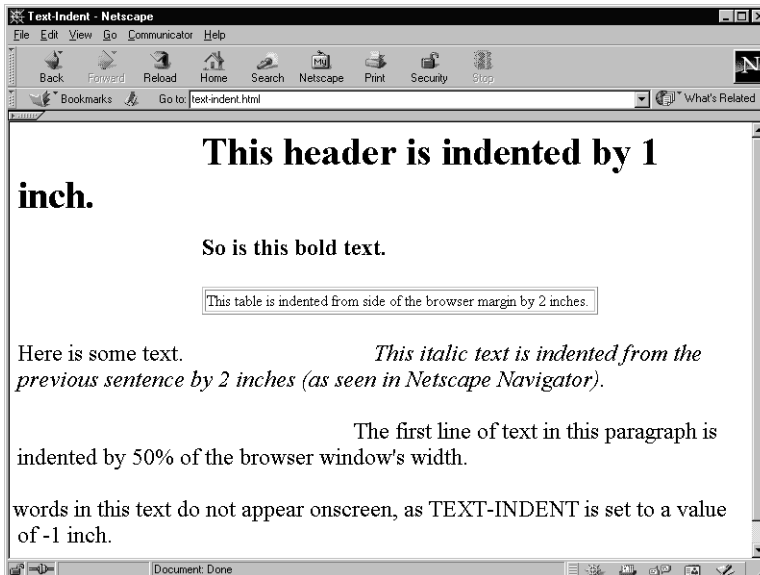


### Listing 11-8 Text-Indent (continued)

```

</TD>
</TR>
</TABLE>
<P>
Here is some text. <I STYLE="TEXT-INDENT: 2IN;">This italic
text is indented from the previous sentence by 2 inches (as
seen in Netscape Navigator).</I>
<P STYLE="TEXT-INDENT: 50%;">The first line of text in this
paragraph is indented by 50% of the browser window's
width.</P>
<P STYLE="TEXT-INDENT: -1IN;">This first words in this text
do not appear onscreen, as TEXT-INDENT is set to a value of
-1 inch.</P>
</BODY>
</HTML>

```



**Figure 11-9** The TEXT-INDENT sample code as seen within Netscape Navigator.

The same code is interpreted differently in Internet Explorer. In Internet Explorer, the table is flush to the margin of the browser window, and the text

within the table is indented 2 inches. Also, it does not render the italic sentence 2 inches away from the rest of the line in which it appears. Internet Explorer is arguably more “correct” when it comes to interpreting the official CSS specification in this regard, but what is important for Web authors to know is that while both Internet Explorer and Netscape Navigator both fully implement this property, they can display the same code in different ways.

## The TEXT-TRANSFORM Property

The `TEXT-TRANSFORM` property is another of those CSS properties whose name doesn’t adequately describe its function. Many new users might be forgiven for thinking that perhaps it does something to change or otherwise “transform” the text on the page to something else altogether. While the term is correct in typographical terms, for the layperson, all it really does is control the case (as in “upper” or “lower” case) of the words it encloses. `TEXT-TRANSFORM` has four different values: `CAPITALIZE`, `LOWERCASE`, `UPPERCASE` and `NONE`. Each of these values are fairly self-explanatory; they affect the text they enclose in the ways they describe, although it should be said that `CAPITALIZE` in this case means “initial capitals”.

### ***TEXT-TRANSFORM***

Description: Specifies the type of case to be used on the text.

CSS Family Type: Text

Values:

`CAPITALIZE` - Sets the initial letter of a word in capital letters.

`LOWERCASE` - Sets all text to lower case.

`NONE` - No transformation is performed.

`UPPERCASE` - Sets all text to upper case.

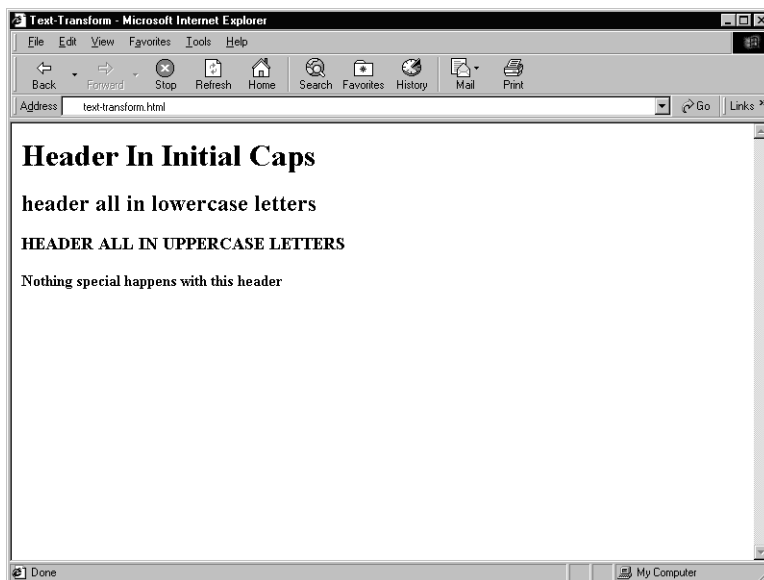
Sample code:

```
<H1 STYLE="TEXT-TRANSFORM: CAPITALIZE">Header in  
initial caps</H1>  
<H2 STYLE="TEXT-TRANSFORM: LOWERCASE">HEADER ALL  
IN LOWERCASE LETTERS</H2>  
<H3 STYLE="TEXT-TRANSFORM: UPPERCASE">header all  
in uppercase letters</H3>  
<H4 STYLE="TEXT-TRANSFORM: NONE">Nothing special  
happens with this header</H4>
```

**Table 11-8 TEXT-TRANSFORM Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Safe

Figure 11–10 shows the effect of the code seen in the definition section for this property. Note in particular how the browser takes the second and third headers and reverses the case in which they were written.




**Figure 11–10** The four values for TEXT-TRANSFORM as seen within Internet Explorer.

The TEXT-TRANSFORM property is fully supported in both Internet Explorer and Netscape Navigator.

# THE BOX FAMILY OF PROPERTIES

## Topics in this Chapter

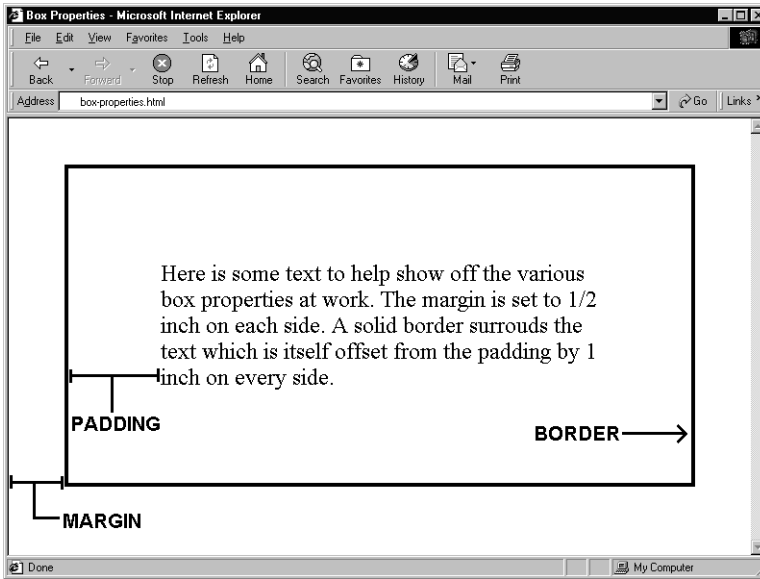
- 
- The BORDER-STYLE Property
  - The BORDER-COLOR Property
  - The BORDER-WIDTH Property
  - The BORDER-BOTTOM, BORDER-TOP, BORDER-LEFT and BORDER-RIGHT Properties
  - The BORDER-“Side”-WIDTH Properties
  - The BORDER Property
  - The CLEAR Property
  - The FLOAT Property
  - The HEIGHT and WIDTH Properties
  - The PADDING-“Side” Properties
  - The MARGIN-“Side” Properties

# *Chapter* 12

**T**he box family of CSS properties is the largest CSS family, with 28 separate properties. While this sounds impressive, the majority of these properties are only a slight variation upon a common formatting theme. Most of these properties can be collected together into one of three sub-groups of similar properties (they share the same name prefix): the border, margin, and padding groups. The border sub-group controls the type, color and placement of borders around elements on a Web page. The margin sub-group handles where margins should appear between elements on a Web page, and the padding sub-group takes care of spacing between a Web element and a border. There are four separate properties (`CLEAR`, `FLOAT`, `HEIGHT` and `WIDTH`) that do not fit neatly into any of these categories, but these “miscellaneous” properties all have a role to play in placing and sizing elements on a Web page.

All of these properties refer to the spacing, placement or size of elements that appear on screen. The family name “box” is derived from the fact that all of these properties are designed to work with block-level elements and control how they are placed or otherwise arranged on a Web page. The three main categories of properties actually determine the placement of elements on a Web page in concentric rectangles. Margin values set the distance between the edge of the browser window and the border (if any) that appears around text or images on a Web page. Padding, in turn, separates the text

from the border. In essence, what you get is a bunch of nested “boxes”, one fitting neatly inside the other. Figure 12–1 provides a good example of this.



**Figure 12–1** The basic margin, border and padding properties displayed.

This chapter looks in detail at the following properties:

- BORDER-STYLE
- BORDER-COLOR
- BORDER-WIDTH
- BORDER-BOTTOM
- BORDER-LEFT
- BORDER-RIGHT
- BORDER-TOP
- BORDER-BOTTOM-WIDTH
- BORDER-LEFT-WIDTH
- BORDER-RIGHT-WIDTH
- BORDER-TOP-WIDTH
- BORDER
- CLEAR
- FLOAT
- HEIGHT

- WIDTH
- PADDING-BOTTOM
- PADDING-BOTTOM
- PADDING-LEFT
- PADDING-RIGHT
- PADDING-TOP
- PADDING
- MARGIN-BOTTOM
- MARGIN-BOTTOM
- MARGIN-LEFT
- MARGIN-RIGHT
- MARGIN-TOP
- MARGIN

## The BORDER-STYLE Property

The `BORDER-STYLE` property determines the type of border that the browser will display around a given element on a Web page, such as text or an image. Using this property, you can add stylish and eye-catching borders to your Web page. Up until now, the only HTML equivalent was to create a table and turn on the `BORDER` attribute. There is no way to otherwise change the solid border that would be displayed. With `BORDER-STYLE` you can play with up to nine different values for the type of border that can be displayed. It is unfortunate that only a few of these values are supported in Netscape Navigator and Internet Explorer.

### ***BORDER-STYLE***

Description: Sets the type of border to be displayed.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- `DASHED` - A dashed border line is displayed.
- `DOTTED` - A dotted border line is displayed.
- `DOUBLE` - A double-line is displayed.
- `INSET` - A 3D inset line is displayed.
- `GROOVE` - A 3D grooved line is displayed.
- `NONE` - No border is displayed, no matter what `BORDER-WIDTH` value is present.

- **OUTSET** - A 3D outset line is displayed.
- **RIDGE** - A 3D ridged line is displayed.
- **SOLID** - A solid border line is displayed.

Example:

```
<H1 STYLE="BORDER-STYLE: SOLID">Header with Inset
Border.</H1>
```

**Table 12-1 BORDER-STYLE Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Partial	Safe	Safe	Partial	Partial	Partial	Partial

There are a number of 2D and 3D border types. The 2D borders are created by the values **DASHED**, **DOTTED**, **DOUBLE** and **SOLID**, while the 3D borders are created by the **INSET**, **OUTSET**, **GROOVE** and **RIDGED** values. Of the 2D border types, the **DASHED** value displays a border made up for dashes, the **DOTTED** value displays a border made from small dots, the **DOUBLE** value creates a border made from a double-line, and **SOLID** displays a solid border style. Of the 3D border types, **INSET** creates a border that appears recessed, while **OUTSET** creates a border that appears the opposite of **INSET**. **GROOVE** displays a 3D grooved border line and **RIDGE** creates a 3D ridged border line. The value **NONE** ensures that no border is displayed. You can see all of these properties put to use in the following code sample:

**Listing 12-1 Border-Style**

```
<HTML>
<HEAD>
<TITLE>Border-Style</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: MEDIUM;">
<P ALIGN="CENTER" STYLE="BORDER-STYLE: DASHED; BORDER-
COLOR: GRAY; BORDER-WIDTH: THICK;">Text surrounded by a
dashed border style.</P>
```

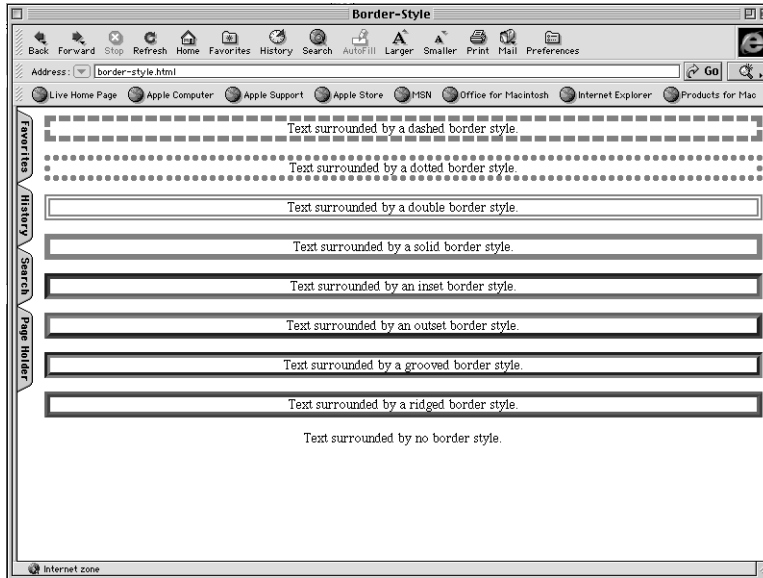


**Listing 12-1 Border-Style (continued)**

```
<P ALIGN="CENTER" STYLE="BORDER-STYLE: DOTTED; BORDER-
COLOR: GRAY; BORDER-WIDTH: THICK;">Text surrounded by a
dotted border style.</P>
<P ALIGN="CENTER" STYLE="BORDER-STYLE: DOUBLE; BORDER-
COLOR: GRAY; BORDER-WIDTH: THICK;">Text surrounded by a
double border style.</P>
<P ALIGN="CENTER" STYLE="BORDER-STYLE: SOLID; BORDER-COLOR:
GRAY; BORDER-WIDTH: THICK;">Text surrounded by a solid
border style.</P>
<P ALIGN="CENTER" STYLE="BORDER-STYLE: INSET; BORDER-COLOR:
GRAY; BORDER-WIDTH: THICK;">Text surrounded by an inset
border style.</P>
<P ALIGN="CENTER" STYLE="BORDER-STYLE: OUTSET; BORDER-
COLOR: GRAY; BORDER-WIDTH: THICK;">Text surrounded by an
outset border style.</P>
<P ALIGN="CENTER" STYLE="BORDER-STYLE: GROOVE; BORDER-
COLOR: GRAY; BORDER-WIDTH: THICK;">Text surrounded by a
grooved border style.</P>
<P ALIGN="CENTER" STYLE="BORDER-STYLE: RIDGE; BORDER-COLOR:
GRAY; BORDER-WIDTH: THICK;">Text surrounded by a ridged
border style.</P>
<P ALIGN="CENTER" STYLE="BORDER-STYLE: NONE; BORDER-COLOR:
GRAY; BORDER-WIDTH: THICK;">Text surrounded by no border
style.</P>
</BODY>
</HTML>
```

Support for this property is minimal. Internet Explorer for Windows and Unix recognizes only the **DOUBLE**, **SOLID** and **NONE** values — all other values for **BORDER-STYLE** result in a border identical to **SOLID**. Netscape Navigator is capable of displaying all of the values for **BORDER-STYLE** with the exception of **DASHED** and **DOTTED**. Oddly enough, only the recent Macintosh versions of Internet Explorer are capable of displaying all of the values for **BORDER-STYLE**, as can be seen in Figure 12–2, which displays the sample code on Internet Explorer 4.5 for the Macintosh

There are a number of other restrictions worth knowing about when using **BORDER-STYLE**. When using **BORDER-STYLE** in Netscape Navigator, make sure you use the **BORDER-WIDTH** property as well, otherwise nothing will be displayed. Also, beware using the color value **BLACK** in combination with the **INSET**, **OUTSET**, **RIDGE** and **GROOVE** values, as Internet Explorer



**Figure 12-2** All of the BORDER-STYLE attributes as seen from within Internet Explorer for the Macintosh.

does not display the intended effect, but rather colors the inner and outer borders the same value. Since the default value for any border is BLACK, it is recommended that you always set a non-BLACK color value when using BORDER-STYLE for the sake of Internet Explorer users. It should be said that Netscape Navigator does fine when the value is black. One last thing — try not to use the values “RIDGED” or “GROOVED” when you mean to “RIDGE” or “GROOVE”, a common slip-up when setting these values.

## The BORDER-COLOR Property

The BORDER-COLOR property sets the color for the border to be displayed. Previous to the introduction of this property, the only way you could add color to borders was to use the BORDERCOLOR attribute, or the Internet Explorer-only BORDERCOLORLIGHT or BORDERCOLORDARK in conjunction with the <TABLE> HTML tag. The BORDER-COLOR property opens things up so that a Web author can add color to any block-level element.

## BORDER-COLOR

Description: Sets the color of the border sides.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- *color name* or *numerical color value* - Sets the color for the border.

Sample code:

```
<H3 STYLE="BORDER-COLOR: OLIVE">Header with a  
Colored Border</H3>
```

**Table 12-2 BORDER-COLOR Support in Major Browsers**

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i>	<i>IE 4.5</i>	<i>(Win. &amp;</i>	<i>NN 4.x</i>	<i>(Mac &amp;</i>	<i>Opera</i>
		<i>(Mac)</i>	<i>(Mac)</i>	<i>UNIX)</i>		<i>UNIX)</i>	<i>3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Partial

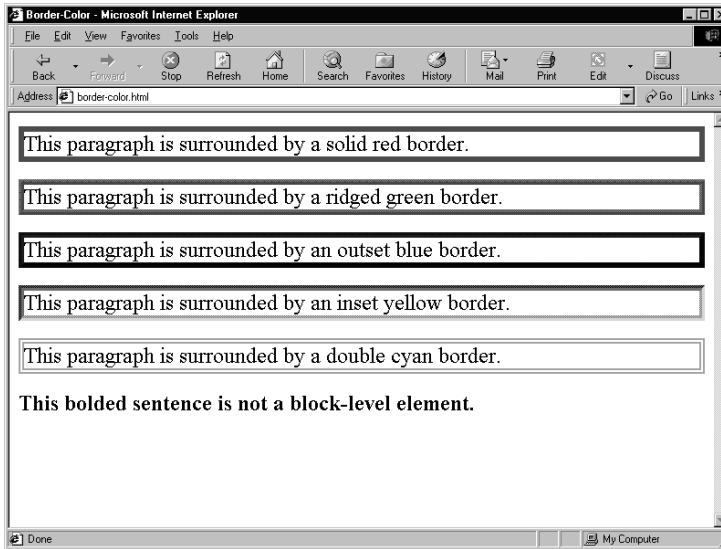
The `BORDER-COLOR` property, like all other color CSS properties, allows Web authors to set a color value in a variety of ways. For example, it can take a standard color name (such as “RED”) or a hexadecimal color value (such as “#FF0000”, which is also red), both of which are standard ways of setting color values in HTML. In CSS, you can also specify color using a “compressed” hexadecimal color value (such as “#0F0”, which is green). You can also use a three-digit RGB (“Red Green Blue”) color value (where RGB(0,0,255) is blue), and a three-digit RGB percentage color value (where RGB(0%,0%,100%) is blue). (For more information on ways of specifying color, please see the “Color Formats” section in the CSS Units chapter).

The following code example (seen in Figure 12–3) demonstrates all of the different ways of implementing color with the `BORDER-COLOR` property in a single Web page.

**Listing 12-2 Border-Color**

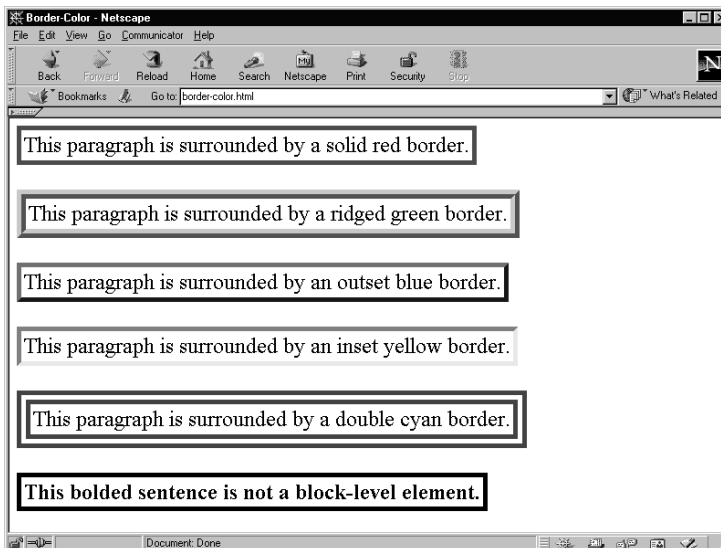
```
<HTML>
<HEAD>
<TITLE>Border-Color</TITLE>
</HEAD>
</HTML>
<BODY STYLE="FONT-SIZE: LARGE;">
<P STYLE="BORDER-COLOR: RED; BORDER-STYLE: SOLID; BORDER-
WIDTH: THICK;">This paragraph is surrounded by a solid red
border.</P>
<P STYLE="BORDER-COLOR: #00FF00; BORDER-STYLE: RIDGE;
BORDER-WIDTH: THICK;">This paragraph is surrounded by a
ridged green border.</P>
<P STYLE="BORDER-COLOR: #00F; BORDER-STYLE: OUTSET; BORDER-
WIDTH: THICK;">This paragraph is surrounded by an outset
blue border.</P>
<P STYLE="BORDER-COLOR: RGB(255,255,0); BORDER-STYLE:
INSET; BORDER-WIDTH: THICK;">This paragraph is surrounded
by an inset yellow border.</P>
<P STYLE="BORDER-COLOR: RGB(0,100%,100%); BORDER-STYLE:
DOUBLE; BORDER-WIDTH: THICK;">This paragraph is surrounded
by a double cyan border.</P>
<B STYLE="BORDER-COLOR: BLACK; BORDER-STYLE: SOLID; BORDER-
WIDTH: THICK;">This bolded sentence is not a block-level
element.</B>
</BODY>
</HTML>
```

Both Internet Explorer and Netscape Navigator implement this property, but they each have their own special ways of doing it. Internet Explorer keeps strictly to the specifications, applying the value only to block-level elements, such as headers or paragraphs. Netscape Navigator correctly interprets all of the color values, but can only display a border if `BORDER-WIDTH` is also set. It also adds a border to non-block-level elements, and instead of extending the border across the browser window (as would ordinarily be expected of a block-level element), the border instead only fits the text or other Web page elements it contains. You can see the differences in the effects of the same code displayed in Netscape Navigator in Figure 12-4. Essentially, the prop-



**Figure 12-3** The various CSS color format displayed using the BORDER-COLOR property as seen in Internet Explorer.

erty works well in both browsers, but as with all of the border sub-family of properties, keep the differences in mind when using it.



**Figure 12-4** The various CSS color format displayed using the BORDER-COLOR property as seen in Netscape Navigator.

## The BORDER-WIDTH Property

The `BORDER-WIDTH` property is used to set the thickness value for the border to be displayed. It can either take a absolute pixel value, or any one of three special name values. This is a particularly important property for Netscape Navigator, as this property needs to be present for the browser to recognize any of the other border properties.

### ***BORDER-WIDTH***

Description: Sets the thickness of the border for the specified element. One to four border sides can be set.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- `n [ n n n ] measurement_units`
- Sets the thickness of the border.
- If 1 value is present, all borders are set to the numerical value specified
- If 2 values are present, the top and bottom, and then the side borders take the numerical values specified
- If 3 values are present, the top, right and left sides, then bottom take the numerical values specified
- If 4 values are present, the top, right, bottom, then left borders take the numerical values specified
- `THIN | MEDIUM | THICK` - Sets the relative thickness of the border.

Sample code:

```
<H2 STYLE="BORDER-WIDTH: 5 5 15 5">Header with  
Surrounding Border</H2>
```

**Table 12-3 BORDER-WIDTH Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Partial

This property can take from one to four separate numerical values, which set the width of the border selected. The default value is pixels, so if no measurement unit is specified (as in the examples used in this section), the value given is assumed to be a pixel value. If multiple values are to be set using other units of measurement, each value in turn must be set with a specific unit of measurement — one measurement unit value *does not* apply to all automatically. If a single numerical value is present, all borders are set to that value. If two numerical values are present, the top and bottom take the first value, and the side borders take the second. If three values are present, the top, right and left sides, and then the bottom take the numerical values specified. Finally, if four values are present, the top, right, bottom, then left borders take the numerical values specified. When specifying these values, they must be separated by a space, as can be seen in the following code example.

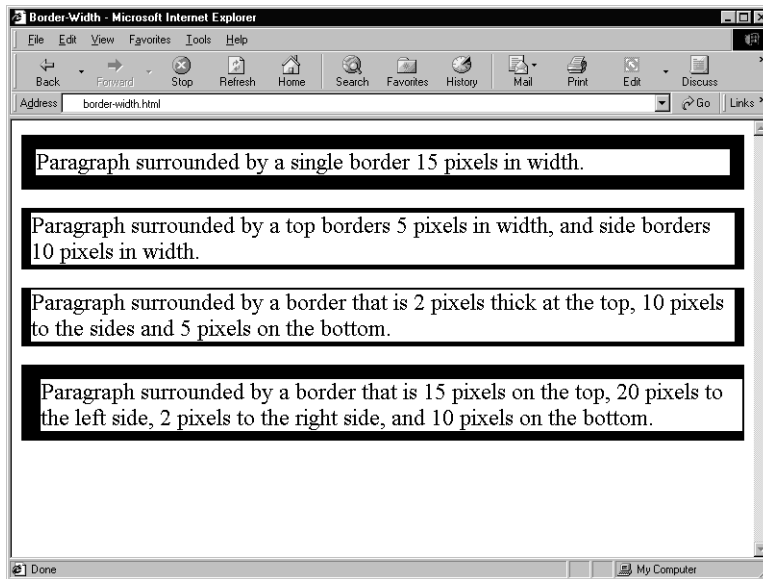
### Listing 12-3 Border-Width

```
<HTML>
<HEAD>
<TITLE>Border-Width</TITLE>
</HEAD>
</HTML>
<BODY STYLE="FONT-SIZE: LARGE;">
<P STYLE="BORDER-WIDTH: 15; BORDER-STYLE: SOLID;">Paragraph
surrounded by a single border 15 pixels in width.</P>
<P STYLE="BORDER-WIDTH: 5 10; BORDER-STYLE:
SOLID;">Paragraph surrounded by a top borders 5 pixels in
width, and side borders 10 pixels in width.</P>
<P STYLE="BORDER-WIDTH: 2 10 5; BORDER-STYLE:
SOLID;">Paragraph surrounded by a border that is 2 pixels
thick at the top, 10 pixels to the sides and 5 pixels on
the bottom.</P>
<P STYLE="BORDER-WIDTH: 15 2 10 20; BORDER-STYLE:
SOLID;">Paragraph surrounded by a border that is 15 pixels
on the top, 20 pixels to the left side, 2 pixels to the
right side, and 10 pixels on the bottom.</P>
</BODY>
</HTML>
```

---

Both Internet Explorer and Netscape Navigator implement this property, although in slightly different ways. Internet Explorer applies the property only to block-level elements, such as headers and paragraphs, while Netscape Navigator will apply it to any on-screen element. The only major difference

you have to remember in terms of actual display characteristics is that Internet Explorer extends the border the full width of the browser window, whereas Netscape Navigator only uses the width of the text or other elements it contains.



**Figure 12-5** Various numerical values applied to the BORDER-WIDTH property as seen in Internet Explorer.

The BORDER-WIDTH property can take one of three specific name values: THIN, MEDIUM and THICK. It is also possible to combine one or more of these named values in the same way as with the numerical values. Examples of all of these named values can be seen in the following code, which is displayed in Figure 12-6.

#### Listing 12-4 Border-Width #2

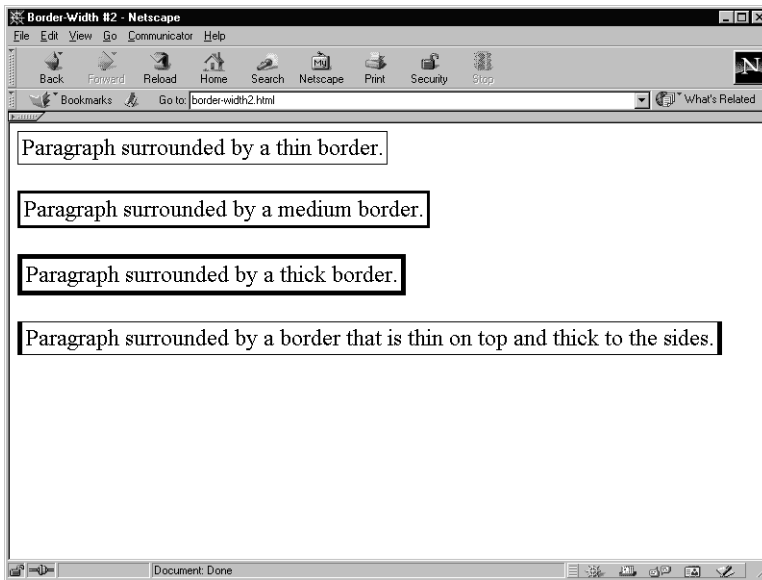
```
<HTML>
<HEAD>
<TITLE>Border-Width #2</TITLE>
</HEAD>
</HTML>
<BODY STYLE="FONT-SIZE: LARGE;">
```



**Listing 12-4 Border-Width #2 (continued)**

```
<P STYLE="BORDER-WIDTH: THIN; BORDER-STYLE: SOLID;">Paragraph
surrounded by a thin border.</P>
<P STYLE="BORDER-WIDTH: MEDIUM; BORDER-STYLE: SOLID;">Paragraph
surrounded by a medium border.</P>
<P STYLE="BORDER-WIDTH: THICK; BORDER-STYLE: SOLID;">Paragraph
surrounded by a thick border.</P>
<P STYLE="BORDER-WIDTH: THIN THICK; BORDER-STYLE:
SOLID;">Paragraph surrounded by a border that is thin on top
and thick to the sides.</P>
</BODY>
</HTML>
```

---



**Figure 12-6** Various name values applied to the BORDER-WIDTH property as seen in Netscape Navigator.

## The BORDER-BOTTOM Property

The `BORDER-BOTTOM` property sets the value to be assigned to the border at the bottom of the box *only*. It can take on all of the values belonging to the `BORDER-STYLE`, `BORDER-COLOR` and `BORDER-WIDTH` properties. It functions in the same way as its sibling properties `BORDER-TOP`, `BORDER-RIGHT` and `BORDER-LEFT`.

### ***BORDER-BOTTOM***

Descriptions: Sets the display value for the bottom border of the specified HTML tag.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement\_units
- Sets the thickness of the border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.
- *color name* or *numerical color value* - Sets the color for the border.
- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what `BORDER-WIDTH` value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.

Sample code:

```
<EM STYLE="BORDER-BOTTOM: DOUBLE BLUE">Text with a  
double blue underline.</EM>
```

**Table 12-4 BORDER-BOTTOM Support in Major Browsers**

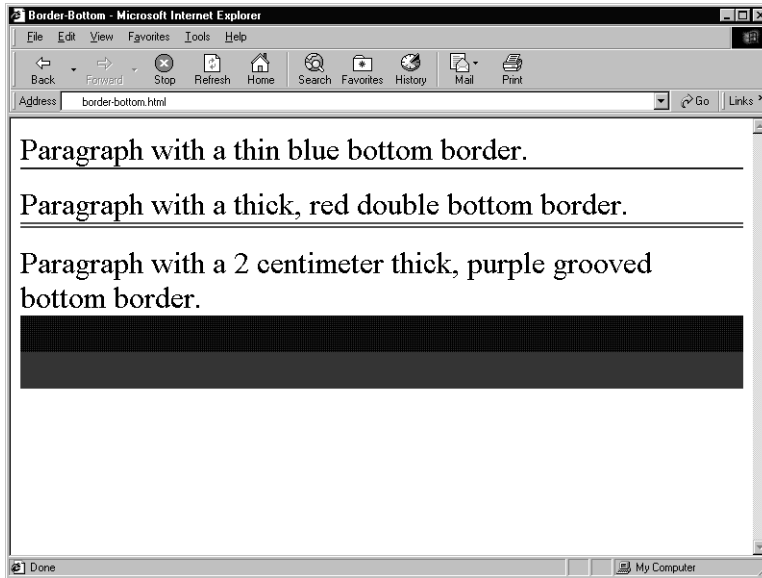
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Partial

The BORDER-BOTTOM property is something of a “shortcut” property, as it is capable of taking on all of the other values associated with BORDER-STYLE, BORDER-COLOR and BORDER-WIDTH properties. It can take a single numerical value or named value that sets the width, color value and border style. What the BORDER-BOTTOM property does is to set the display properties that are applied to the bottom border only — no other borders are seen. The following code (displayed in Figure 12–7) shows how it can be used:

**Listing 12-5 Border-Bottom**

```
<HTML>
<HEAD>
<TITLE>Border-Bottom</TITLE>
</HEAD>
</HTML>
<BODY STYLE="FONT-SIZE: X-LARGE;">
<P STYLE="BORDER-BOTTOM: THIN SOLID BLUE;">Paragraph with a
thin blue bottom border.</P>
<P STYLE="BORDER-BOTTOM: THICK DOUBLE #FF0000;">Paragraph
with a thick, red double bottom border.</P>
<P STYLE="BORDER-BOTTOM: PURPLE GROOVE 2CM;">Paragraph with
a 2 centimeter thick, purple grooved bottom border.</P>
</BODY>
</HTML>
```

This property is fully supported in Internet Explorer (with the proviso that, depending on the operating system, it may not support all border styles — see the BORDER-STYLE property earlier in this chapter for more info). It is not supported in Netscape Navigator.



**Figure 12-7** The code sample for BORDER-BOTTOM property as seen in Internet Explorer.

## The BORDER-LEFT Property

The BORDER-LEFT property works in exactly the same manner as the BORDER-BOTTOM property, but in this case sets the value to be assigned to the border to the left of the box *only*. It can take on all of the values belonging to the BORDER-STYLE, BORDER-COLOR and BORDER-WIDTH properties. It functions in the same way as its sibling properties BORDER-TOP, BORDER-RIGHT and BORDER-BOTTOM.

### ***BORDER-LEFT***

**Description:** Specifies how the left border associated with an HTML tag should be displayed.

**CSS Family Type:** Boxes (Sub-family: Borders)

**Values:**

- *n* measurement units

- Sets the thickness of the border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.
- *color name* or *numerical color value* - Sets the color for the border.
- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.

Sample code:

```
<ADDRESS STYLE="BORDER-LEFT: 1.0cm AQUA">123
Underwater Drive<BR>An aqua-tic address.</ADDRESS>
```

**Table 12-5 BORDER-LEFT Support in Major Browsers**

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> (Mac)	<i>IE 4.5</i> (Mac)	(Win. & UNIX)	<i>NN 4.x</i>	(Mac & UNIX)	<i>Opera</i> 3.6
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Partial

The BORDER-LEFT property is something of a “shortcut” property, as it is capable of taking on all of the other values associated with BORDER-STYLE, BORDER-COLOR and BORDER-WIDTH properties. It can take a single numerical value or named value that sets the width, color value and border style. What the BORDER-LEFT property does is to set the display properties that are

applied to the left border only — no other borders are seen. The following code (depicted in Figure 12–8) shows how it can be used:

### Listing 12-6 Border-Left

```
<HTML>
<HEAD>
<TITLE>Border-Left</TITLE>
</HEAD>
</HTML>
<BODY STYLE="FONT-SIZE: X-LARGE;">
<P STYLE="BORDER-LEFT: MEDIUM SOLID RED;">Paragraph with a
thin red left border.</P>
<P STYLE="BORDER-LEFT: THICK RIDGE #0000FF;">Paragraph with
a thick, blue ridged left border.</P>
<P STYLE="BORDER-LEFT: 1IN GREEN DOUBLE;">Paragraph with a
1 inch thick, green double border to the left.</P>
</BODY>
</HTML>
```

---

This property is fully supported in Internet Explorer (with the proviso that, depending on the operating system, it may not support all border styles — see the `BORDER-STYLE` property earlier in this chapter for more info). It is not supported in Netscape Navigator.

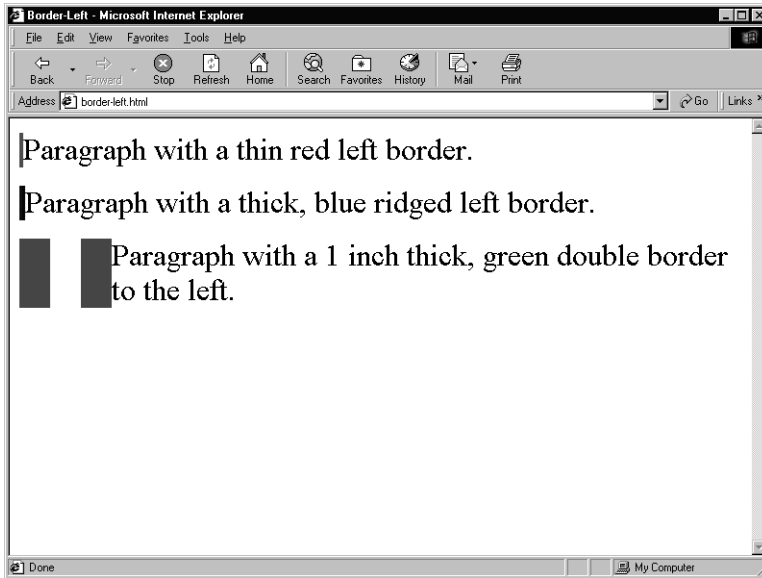
## The BORDER-RIGHT Property

The `BORDER-RIGHT` property works in exactly the same manner as the `BORDER-LEFT` and `BORDER-BOTTOM` properties, but in this case sets the value to be assigned to the border to the right of the box *only*. It can take on all of the values belonging to the `BORDER-STYLE`, `BORDER-COLOR` and `BORDER-WIDTH` properties. It functions in the same way as its sibling properties `BORDER-TOP`, `BORDER-LEFT` and `BORDER-BOTTOM`.

### ***BORDER-RIGHT***

Description: Specifies how the right border associated with an HTML tag should be displayed.

CSS Family Type: Boxes (Sub-family: Borders)



**Figure 12-8** The code sample for BORDER-LEFT property as seen in Internet Explorer.

#### Values:

- *n* measurement units
- Sets the thickness of the border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.
- *color name* or *numerical color value* - Sets the color for the border.
- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.

#### Sample code:

```
<H5 STYLE="BORDER-RIGHT: THICK DOUBLE
YELLOW">Header with Yellow Right Border</H5>
```

Table 12-6 BORDER-RIGHT Support in Major Browsers

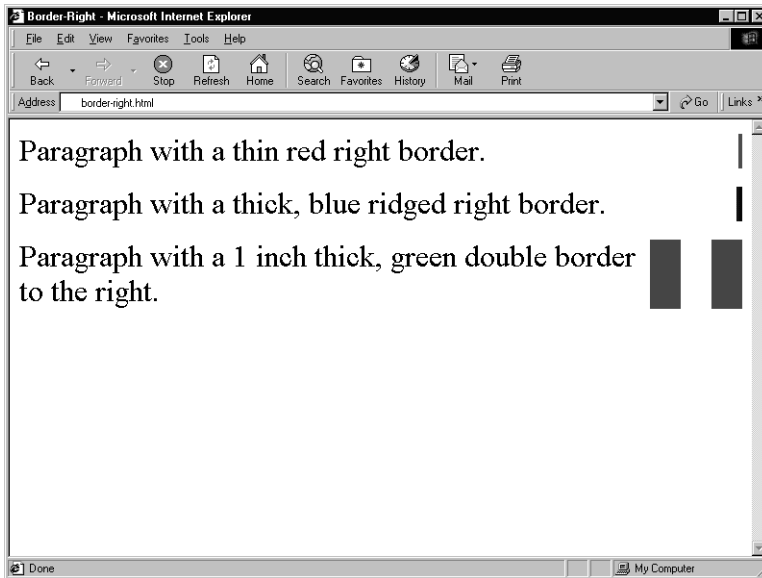
				IE 5.0		NN 4.0	
IE 3.02	IE 4.x	IE 4.01 (Mac)	IE 4.5 (Mac)	(Win. & UNIX)	NN 4.x	(Mac & UNIX)	Opera 3.6
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Partial

The BORDER-RIGHT property is a type of “shortcut” property, capable of taking on all of the other values associated with BORDER-STYLE, BORDER-COLOR and BORDER-WIDTH properties. It can take a single numerical value or named value that sets the width, color value and border style. What the BORDER-RIGHT property does is to set the display properties that are applied to the right border only — no other borders are seen. The following code (depicted in Figure 12–9) shows how it can be used:

Listing 12-7 Border-Right

```
<HTML>
<HEAD>
<TITLE>Border-Right</TITLE>
</HEAD>
</HTML>
<BODY STYLE="FONT-SIZE: X-LARGE;">
<P STYLE="BORDER-RIGHT: MEDIUM SOLID RED;">Paragraph with a
thin red right border.</P>
<P STYLE="BORDER-RIGHT: THICK RIDGE #0000FF;">Paragraph
with a thick, blue ridged right border.</P>
<P STYLE="BORDER-RIGHT: 1IN GREEN DOUBLE;">Paragraph with a
1 inch thick, green double border to the right.</P>
</BODY>
</HTML>
```





**Figure 12-9** The code sample for BORDER-RIGHT property as seen in Internet Explorer.

This property is fully supported in Internet Explorer (with the proviso that, depending on the operating system, it may not support all border styles — see the BORDER-STYLE property earlier in this chapter for more info). It is not supported in Netscape Navigator.

## The BORDER-TOP Property

The BORDER-TOP property works in exactly the same manner as the BORDER-RIGHT, BORDER-LEFT and BORDER-BOTTOM properties we have already looked at, and sets the value to be assigned to the border to the top of the box *only*. It can take on all of the values belonging to the BORDER-STYLE, BORDER-COLOR and BORDER-WIDTH properties.

## BORDER-TOP

Description: Specifies how the top border associated with an HTML tag should be displayed.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units
- Sets the thickness of the border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.
- *color name* or *numerical color value* - Sets the color for the border.
- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.

Sample code:

```
<TT STYLE="BORDER-TOP: THIN DASHED
00FF00">Teletype text with a top-border</TT>
```

**Table 12-7 BORDER-TOP Support in Major Browsers**

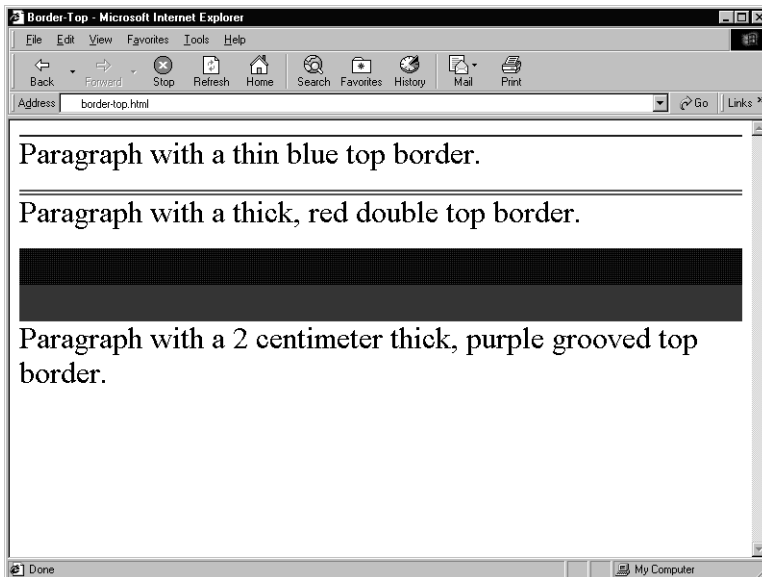
				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> (Mac)	<i>IE 4.5</i> (Mac)	(Win. & UNIX)	<i>NN 4.x</i>	(Mac & UNIX)	<i>Opera</i> 3.6
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Partial

The BORDER-TOP is capable of taking on all of the other values associated with BORDER-STYLE, BORDER-COLOR and BORDER-WIDTH properties. It can take a single numerical value or named value that sets the width, color value and border style. The BORDER-TOP property sets the display properties that

are applied to the top border only — no other borders are seen. The following code (depicted in Figure 12–10) shows how it can be used:

### Listing 12-8 Border-Top

```
<HTML>
<HEAD>
<TITLE>Border-Top</TITLE>
</HEAD>
</HTML>
<BODY STYLE="FONT-SIZE: X-LARGE;">
<P STYLE="BORDER-TOP: THIN SOLID BLUE;">Paragraph with a
thin blue top border.</P>
<P STYLE="BORDER-TOP: THICK DOUBLE #FF0000;">Paragraph with
a thick, red double top border.</P>
<P STYLE="BORDER-TOP: PURPLE GROOVE 2CM;">Paragraph with a
2 centimeter thick, purple grooved top border.</P>
</BODY>
</HTML>
```



**Figure 12-10** The code sample for BORDER-TOP property as seen in Internet Explorer.

This property is fully supported in Internet Explorer (with the proviso that, depending on the operating system, it may not support all border styles — see the `BORDER-STYLE` property earlier in this chapter for more info). It is not supported in Netscape Navigator.

## The `BORDER-BOTTOM-WIDTH` Property

The `BORDER-BOTTOM-WIDTH` property is designed to change the display of the bottom border. It differs from the `BORDER-BOTTOM` property in that it also displays a full border around all of the other sides of the box — only the bottom border is altered. It can take on all of the values belonging to the `BORDER-WIDTH` property. It functions in the same way as its sibling properties `BORDER-TOP-WIDTH`, `BORDER-RIGHT-WIDTH` and `BORDER-LEFT-WIDTH`.

### ***BORDER-BOTTOM-WIDTH***

Description: Sets the thickness of the bottom border.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units
- Sets the thickness of the bottom border.
- `THIN` | `MEDIUM` | `THICK` - Sets the relative thickness of the border.

Sample code:

```
<BLOCKQUOTE STYLE="BORDER-BOTTOM-WIDTH:
THICK">"This is an important quote,"</BLOCKQUOTE>
he said.
```

**Table 12-8 `BORDER-BOTTOM-WIDTH` Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Partial

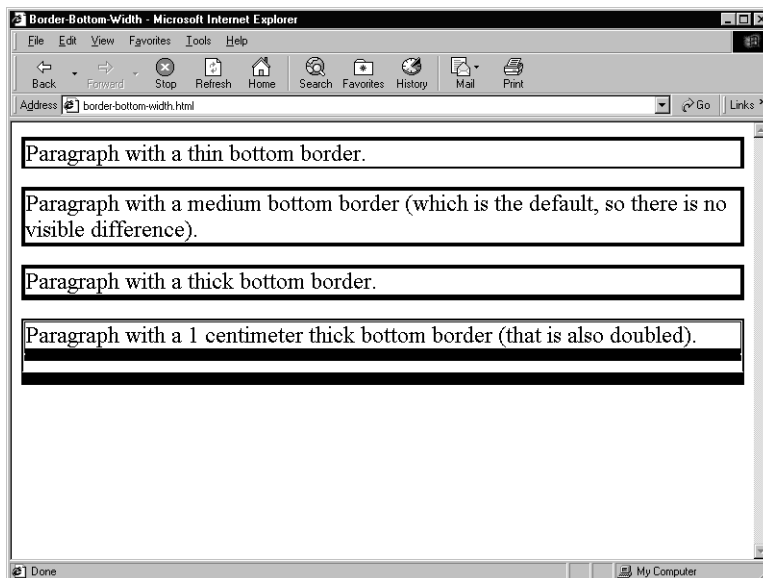
BORDER-BOTTOM-WIDTH can take one of two different sets of values: either a numeric value that sets the size of the border, or a name value. The name value can be one of either THIN, MEDIUM or THICK, where MEDIUM is the default border width. If a numeric value is given and no measurement unit is specified, the default value is set to a pixel width. Remember that when you are using this property, you are only setting the thickness value for the bottom width of the border, so this property is most often used in conjunction with other border family properties that set the display characteristics for the rest of the border. The following code (illustrated in Figure 12-11) gives you an idea as to how it can be used:

### Listing 12-9 Border-Bottom-Width

```
<HTML>
<HEAD>
<TITLE>Border-Bottom-Width</TITLE>
</HEAD>
</HTML>
<BODY STYLE="FONT-SIZE: LARGE;">
<P STYLE="BORDER-BOTTOM-WIDTH: THIN; BORDER-STYLE:
SOLID;">Paragraph with a thin bottom border.</P>
<P STYLE="BORDER-BOTTOM-WIDTH: MEDIUM; BORDER-STYLE:
SOLID;">Paragraph with a medium bottom border (which is the
default, so there is no visible difference).</P>
<P STYLE="BORDER-BOTTOM-WIDTH: THICK; BORDER-STYLE:
SOLID;">Paragraph with a thick bottom border.</P>
<P STYLE="BORDER-BOTTOM-WIDTH: 1CM; BORDER-STYLE:
DOUBLE;">Paragraph with a 1 centimeter thick bottom border
(that is also doubled).</P>
</BODY>
</HTML>
```

---

This attribute is fully supported in Internet Explorer, but only partially implemented in Netscape Navigator. Netscape Navigator displays the previous code example in the same manner as it is supposed to be displayed when the BORDER-BOTTOM property is set — in other words, it only displays the bottom border, instead of just enhancing it. In addition, this property is not supposed to work if the BORDER-STYLE property is not set. Internet Explorer correctly deals with its absence, whereas Netscape Navigator does not.



**Figure 12-11** Code example for BORDER-BOTTOM-WIDTH displayed in Internet Explorer.

## The BORDER-LEFT-WIDTH Property

The BORDER-LEFT-WIDTH property is designed to change the display of the left border. It differs from the BORDER-LEFT property in that it also displays a full border around all of the other sides of the box — only the left border is altered. It can take on all of the values belonging to the BORDER-WIDTH property. It functions in the same way as its sibling properties BORDER-TOP-WIDTH, BORDER-RIGHT-WIDTH and BORDER-BOTTOM-WIDTH.

### ***BORDER-LEFT-WIDTH***

Sets the thickness of the left border.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units

- Sets the thickness of the left border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.

Sample code:

```
<H4 STYLE="BORDER-LEFT-WIDTH THIN">Text with a thin
border to the left.</H4>
```

**Table 12-9 BORDER-LEFT-WIDTH Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Partial

BORDER-LEFT-WIDTH can take one of two different sets of values: either a numeric value that sets the size of the border, or a name value. The name value can be one of either THIN, MEDIUM or THICK, where MEDIUM is the default border width. If a numeric value is given and no measurement unit specified, the default value is set to a pixel width. Remember that when you are using this property, you are only setting the thickness value for the left width of the border, so this property is most often used in conjunction with other border family properties that set the display characteristics for the

rest of the border. The following code (illustrated in Figure 12–12) gives you an idea as to how it can be used:

### Listing 12-10 Border-Left-Width

```
<HTML>
<HEAD>
<TITLE>Border-Left-Width</TITLE>
</HEAD>
</HTML>
<BODY STYLE="FONT-SIZE: LARGE;">
<P STYLE="BORDER-LEFT-WIDTH: THIN; BORDER-STYLE:
SOLID;">Paragraph with a thin left border.</P>
<P STYLE="BORDER-LEFT-WIDTH: MEDIUM; BORDER-STYLE:
SOLID;">Paragraph with a medium left border (which is the
default, so there is no visible difference).</P>
<P STYLE="BORDER-LEFT-WIDTH: THICK; BORDER-STYLE:
SOLID;">Paragraph with a thick left border.</P>
<P STYLE="BORDER-LEFT-WIDTH: 1CM; BORDER-STYLE:
DOUBLE;">Paragraph with a 1 centimeter thick left border
(that is also doubled).</P>
</BODY>
</HTML>
```

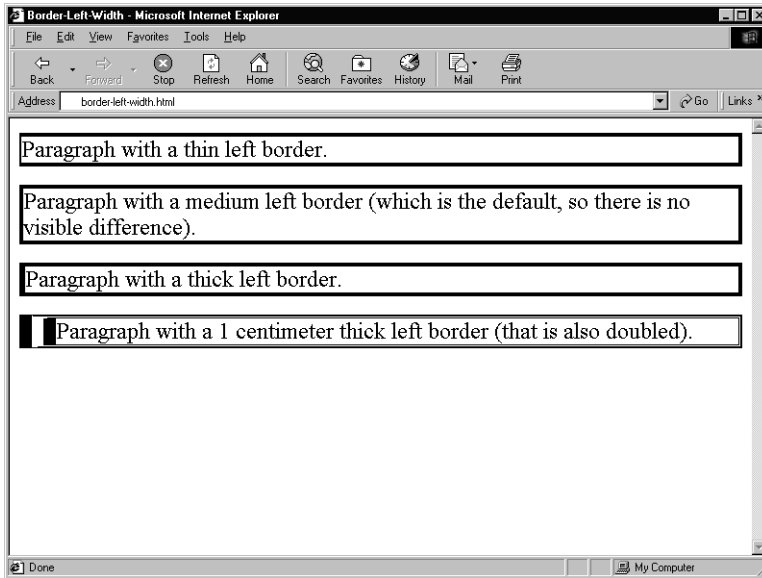
---

This attribute is fully supported in Internet Explorer, but only partially implemented in Netscape Navigator. Netscape Navigator displays the previous code example in the same manner as it is supposed to be displayed when the `BORDER-LEFT` property is set — in other words, it only displays the left border, instead of just enhancing it. In addition, this property is not supposed to work if the `BORDER-STYLE` property is not set. Internet Explorer correctly deals with its absence, whereas Netscape Navigator does not.

## The BORDER-RIGHT-WIDTH Property

The `BORDER-RIGHT-WIDTH` property is designed to change the display of the right border. It differs from the `BORDER-RIGHT` property in that it also displays a full border around all of the other sides of the box — only the right





**Figure 12-12** Code example for BORDER-LEFT-WIDTH displayed in Internet Explorer.

border is altered. It can take on all of the values belonging to the BORDER-WIDTH property. It functions in the same way as its sibling properties BORDER-TOP-WIDTH, BORDER-BOTTOM-WIDTH and BORDER-LEFT-WIDTH.

## ***BORDER-RIGHT-WIDTH***

Description: Sets the thickness of the right border.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units
- Sets the thickness of the right border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.

Sample code:

```
<SUP STYLE="BORDER-RIGHT-WIDTH 0.2in">Superscript  
with border to the right.</SUP>
```

**Table 12-10 BORDER-RIGHT-WIDTH Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Partial

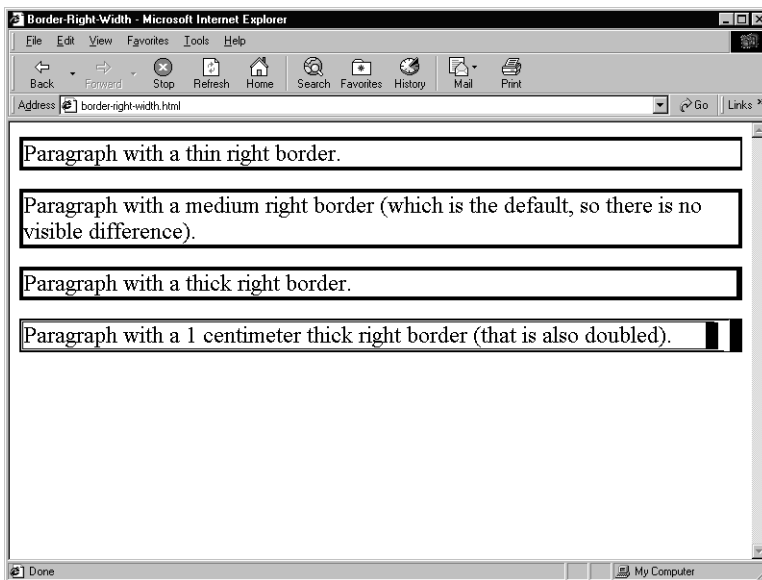
BORDER-RIGHT-WIDTH can take one of two different sets of values: either a numeric value that sets the size of the border, or a name value. The name value can be one of either THIN, MEDIUM or THICK, where MEDIUM is the default border width. If a numeric value is given and no measurement unit specified, the default value is set to a pixel width. Remember that when you are using this property, you are only setting the thickness value for the right width of the border, so this property is most often used in conjunction with other border family properties that set the display characteristics for the rest of the border. The following code (illustrated in Figure 12-13) gives you an idea as to how it can be used:

**Listing 12-11 Border-Right-Width**

```
<HTML>
<HEAD>
<TITLE>Border-Right-Width</TITLE>
</HEAD>
</HTML>
<BODY STYLE="FONT-SIZE: LARGE;">
<P STYLE="BORDER-RIGHT-WIDTH: THIN; BORDER-STYLE:
SOLID;">Paragraph with a thin right border.</P>
```

**Listing 12-11 Border-Right-Width (continued)**

```
<P STYLE="BORDER-RIGHT-WIDTH: MEDIUM; BORDER-STYLE:
SOLID;">Paragraph with a medium right border (which is the
default, so there is no visible difference).</P>
<P STYLE="BORDER-RIGHT-WIDTH: THICK; BORDER-STYLE:
SOLID;">Paragraph with a thick right border.</P>
<P STYLE="BORDER-RIGHT-WIDTH: 1CM; BORDER-STYLE:
DOUBLE;">Paragraph with a 1 centimeter thick right border
(that is also doubled).</P>
</BODY>
</HTML>
```



**Figure 12-13** Code example for BORDER-RIGHT-WIDTH displayed in Internet Explorer.

This attribute is fully supported in Internet Explorer, but only partially implemented in Netscape Navigator. Netscape Navigator displays the previous code example in the same manner as it supposed to be displayed when the BORDER-RIGHT property is set — in other words, it only displays the right border, instead of just enhancing it. In addition, this property is not supposed

to work if the `BORDER-STYLE` property is not set. Internet Explorer correctly deals with its absence, whereas Netscape Navigator does not.

## The `BORDER-TOP-WIDTH` Property

The `BORDER-TOP-WIDTH` property is designed to change the display of the top border. It differs from the `BORDER-TOP` property in that it also displays a full border around all of the other sides of the box — only the top border is altered. It can take on all of the values belonging to the `BORDER-WIDTH` property. It functions in the same way as its sibling properties `BORDER-LEFT-WIDTH`, `BORDER-RIGHT-WIDTH` and `BORDER-BOTTOM-WIDTH`.

### ***BORDER-TOP-WIDTH***

Description: Sets the thickness of the top border.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units
- Sets the thickness of the top border.
- `THIN` | `MEDIUM` | `THICK` - Sets the relative thickness of the border.

Sample code:

```
<SUB STYLE="BORDER-TOP-WIDTH 1mm">Subscript with
very thin top border.</SUB>
```

**Table 12-11 `BORDER-TOP-WIDTH` Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Partial

`BORDER-TOP-WIDTH` can take one of two different sets of values: either a numeric value that sets the size of the border, or a name value. The name

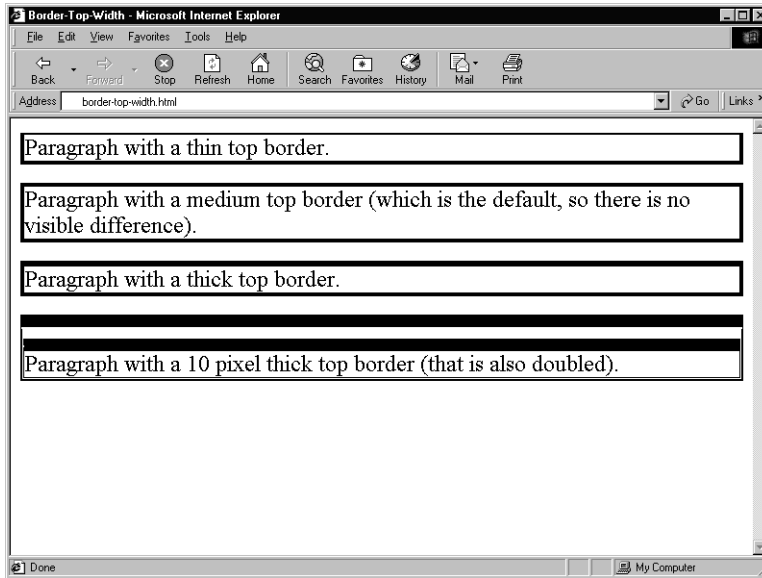
value can be one of either THIN, MEDIUM or THICK, where MEDIUM is the default border width. If a numeric value is given and no measurement unit specified, the default value is set to a pixel width. Remember that when you are using this property, you are only setting the thickness value for the top width of the border, so this property is most often used in conjunction with other border family properties that set the display characteristics for the rest of the border. The following code (illustrated in Figure 12–14) gives you an idea as to how it can be used:

**Listing 12-12 Border-Top-Width**

```
<HTML>
<HEAD>
<TITLE>Border-Top-Width</TITLE>
</HEAD>
</HTML>
<BODY STYLE="FONT-SIZE: LARGE;">
<P STYLE="BORDER-TOP-WIDTH: THIN; BORDER-STYLE:
SOLID;">Paragraph with a thin top border.</P>
<P STYLE="BORDER-TOP-WIDTH: MEDIUM; BORDER-STYLE:
SOLID;">Paragraph with a medium top border (which is the
default, so there is no visible difference).</P>
<P STYLE="BORDER-TOP-WIDTH: THICK; BORDER-STYLE:
SOLID;">Paragraph with a thick top border.</P>
<P STYLE="BORDER-TOP-WIDTH: 1CM; BORDER-STYLE:
DOUBLE;">Paragraph with a 1 centimeter thick top border
(that is also doubled).</P>
</BODY>
</HTML>
```

---

This attribute is fully supported in Internet Explorer, but only partially implemented in Netscape Navigator. Netscape Navigator displays the previous code example in the same manner as it supposed to be displayed when the BORDER-TOP property is set — in other words, it only displays the top border, instead of just enhancing it. In addition, this property is not supposed to work if the BORDER-STYLE property is not set. Internet Explorer correctly deals with its absence, whereas Netscape Navigator does not.



**Figure 12-14** Code example for BORDER-TOP-WIDTH displayed in Internet Explorer.

## The BORDER Property

The BORDER property is a type of “shortcut” property that can allow the Web author to set most, but not all of the properties normally reserved for the other border sub-family of properties.

### **BORDER**

**Description:** This is an abbreviated element that allows the Web author to specify BORDER-STYLE, BORDER-WIDTH and BORDER-COLOR elements easily.

**CSS Family Type:** Boxes (Sub-family: Borders)

**Values:**

- *n* measurement units - Sets the thickness of the border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.

- *color name* or *numerical color value* - Sets the color for the border.
- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.

Sample code:

```
<STRONG STYLE="BORDER: RIDGE RED">A strong
statement.</STRONG>
```

**Table 12-12 BORDER Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Partial	Partial	Safe	Partial	Partial	Partial	Partial

Using BORDER, you can use all of the border styles you could normally set through the BORDER-STYLE property, set the width normally reserved for the BORDER-WIDTH property, and set the color normally reserved for the BORDER-COLOR property. You cannot set the values normally associated with BORDER-TOP, BORDER-RIGHT, BORDER-BOTTOM, BORDER-LEFT, BORDER-TOP-WIDTH, BORDER-RIGHT-WIDTH, BORDER-BOTTOM-WIDTH and BORDER-LEFT-WIDTH. Put simply, you can only set global border values. This means it can only take one numerical or name value to substitute for BORDER-WIDTH, and if you need to set one border value different than another, you will have to use BORDER with one of the other properties that control that type of thing. In addition, when using border, you must set the border style to be displayed — without a typical BORDER-STYLE property value set, nothing is displayed. The

following code (illustrated in Figure 12–15) gives you an idea as to how it works:

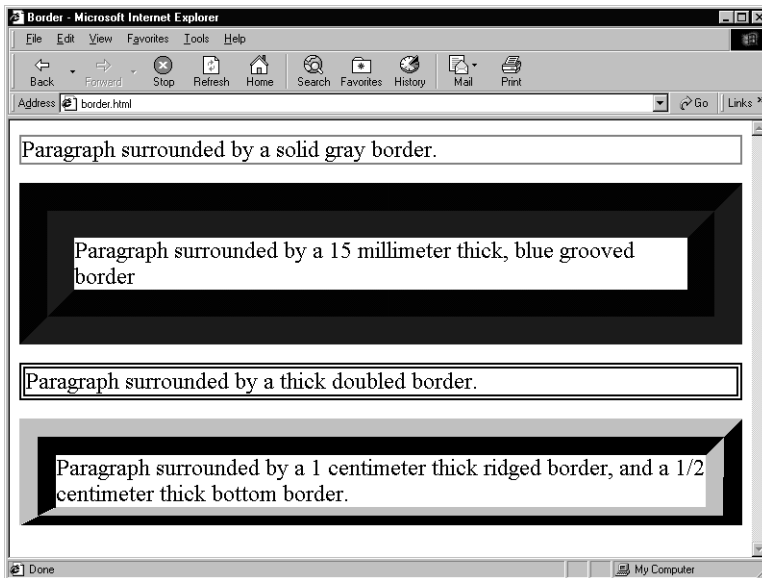
### Listing 12-13 Border

```
<HTML>
<HEAD>
<TITLE>Border</TITLE>
</HEAD>
</HTML>
<BODY STYLE="FONT-SIZE: LARGE;">
<P STYLE="BORDER: THIN SOLID GRAY;">Paragraph surrounded by
a solid gray border.</P>
<P STYLE="BORDER: 15MM GROOVE #0000FF;">Paragraph
surrounded by a 15 millimeter thick, blue grooved border</
P>
<P STYLE="BORDER: THICK DOUBLE;">Paragraph surrounded by a
thick doubled border.</P>
<P STYLE="BORDER: 1CM RIDGE; BORDER-BOTTOM-WIDTH:
0.5CM">Paragraph surrounded by a 1 centimeter thick ridged
border, and a 1/2 centimeter thick bottom border.</P>
</BODY>
</HTML>
```

---

This property is only partially supported under both Internet Explorer and Netscape Navigator. Essentially, they support all of the values already supported under the `BORDER-STYLE`, `BORDER-WIDTH` and `BORDER-COLOR` properties.





**Figure 12-15** Sample code for BORDER displayed.

## The CLEAR Property

The CLEAR property belongs to the family of boxes and margins. It is designed to tell the browser where textual elements should appear on screen. The CLEAR property is used to set how the specified element will allow other Web page elements to “float” along its sides, determining where the text that appears after the element should appear in relation to it. In this way, it works in exactly the same manner as the CLEAR attribute as used with the <BR> HTML tag.

### **CLEAR**

Description: Sets whether the specified element will allow other elements to “float” along its sides.

CSS Family Type: Boxes

Values:

- **BOTH** - The specified element will be moved below floating elements on either side.
- **LEFT** - The specified element will be moved below floating elements on the left side only.
- **NONE** - Any floating elements are allowed on either side of the specified element.
- **RIGHT** - The specified element will be moved below floating elements on the right side only.

Example:

```
<B STYLE="CLEAR: BOTH">This text should appear
below the image.</B>
```

**Table 12-13 CLEAR Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Partial	Partial	Safe	Safe	Partial	Partial	Safe

**Table 12-14 Clear / Float Variant Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Partial	Unsafe	Partial	Partial	Unsafe	Unsafe	Safe

When text follows an image, it generally flows immediately after it in the same vertical space occupied by the image. Using the `CLEAR` property, you can control where text should appear in relation to the image. This can be handy when you have multiple images appearing on a Web page, and you want to be sure that your text is associated with the right image, and that the images don't appear in a jumble on the page. For example, the following code will create such a jumble:

```
<IMG SRC="trajan.jpg" ALIGN="LEFT">
```

```
This text should appear beside the previous image.  
<IMG SRC="trajan.jpg" ALIGN="LEFT">
```

What you will get is the first image appearing on the left, some text appearing to its right, and the next image appearing underneath the text and to the right of the first image. Not pretty. To solve this problem, you could use `<BR CLEAR="ALL">` after the text to force the second image to appear below the first, eliminating the jumble on screen. You can now also use the CLEAR CSS property to accomplish the same thing.

The CLEAR property uses the following modifiers: LEFT, RIGHT, BOTH and NONE. LEFT tells the browser to display the text below the image when the image is set to `ALIGN="LEFT"`. RIGHT tells the browser to display the text below the image when the image is set to `ALIGN="RIGHT"`. BOTH should be used when you have two images aligned left and right, and you want text to appear *beneath* both of them. Finally, NONE places the text *between* two images aligned left and right. The following code (displayed in Figure 12–16) demonstrates all of these elements put to use:

#### Listing 12-14 Clear

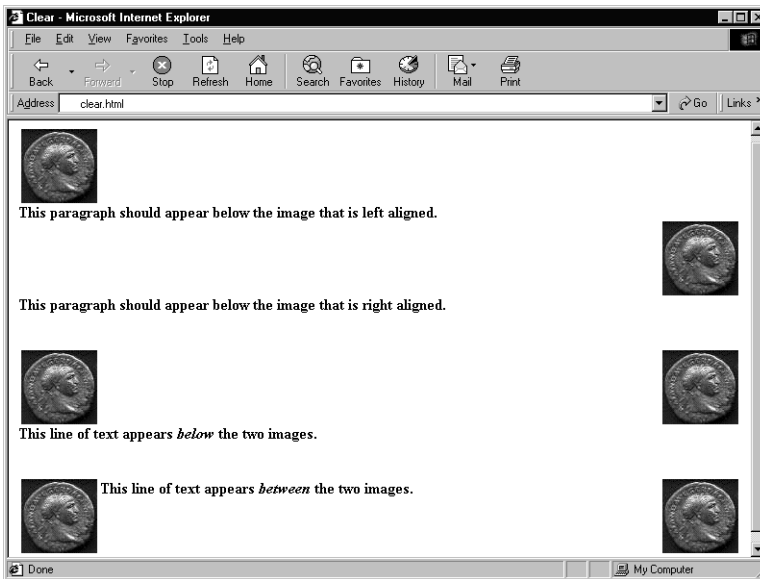
```
<HTML>  
<HEAD>  
<TITLE>Clear</TITLE>  
</HEAD>  
<BODY>  
<P>  
<IMG SRC="trajan.jpg" ALIGN="LEFT">  
<B STYLE="CLEAR: LEFT">  
This paragraph should appear below the image that is left  
aligned.  
</B>  
<BR CLEAR="ALL">  
<IMG SRC="trajan.jpg" ALIGN="RIGHT">  
<B STYLE="CLEAR: RIGHT">  
This paragraph should appear below the image that is right  
aligned.  
</B>
```

### Listing 12-14 Clear (continued)

```

<P>
<BR CLEAR="ALL">
<IMG SRC="trajan.jpg" ALIGN="LEFT">
<IMG SRC="trajan.jpg" ALIGN="RIGHT">
<B STYLE="CLEAR: BOTH">
This line of text appears <I>below</I> the two images.
</B>
<P>
<BR CLEAR="ALL">
<IMG SRC="trajan.jpg" ALIGN="LEFT">
<IMG SRC="trajan.jpg" ALIGN="RIGHT">
<B STYLE="CLEAR: NONE">
This line of text appears <I>between</I> the two images.
</B>
</BODY>
</HTML>

```



**Figure 12-16** Sample code for CLEAR displayed.

Admittedly, much the same thing can be accomplished with less apparent fuss using a table layout to arrange the text and image, but the `CLEAR` property provides an alternate way of doing things.

The `CLEAR` property is supported equally well in the latest versions of both Netscape Navigator and Internet Explorer. It is considered a “safe” CSS property to use. However, be careful if you end up using `CLEAR` along with the `FLOAT` property, as Internet Explorer only partially supports this combination of properties, and Netscape Navigator does not support the combination at all.

## The FLOAT Property

The `FLOAT` property determines how a specified element wraps around another, “parent” element as it floats on a Web page. While it sounds simple, it is arguably one of the hardest things for browsers to get right, and at the moment no browser is fully capable of rendering it properly.

The `FLOAT` property belongs to the CSS family of boxes and margins, and is most typically used to tell a browser where to place a text either immediately before or after an image. To put things in more concrete terms, think of an image on a Web page and some text that follows immediately after it. Browsers typically place the text flush to the bottom of the image. The `FLOAT` property is designed to allow the Web author to have that text appear beside that image instead. This is handy, because otherwise the image would take over the Web page, especially if the image is a large one.

### ***FLOAT***

Description: Sets how elements can wrap around another, “parent” element as it floats on a Web page.

CSS Family Type: Boxes

Values:

- `NONE` - Other elements cannot be displayed to float around the specified element.
- `LEFT` - Other elements can be displayed to the left margin of the specified element.
- `RIGHT` - Other elements can be displayed to the right margin of the specified element.

Example:

```
<IMG SRC="aurelian.jpg" STYLE="FLOAT:
LEFT">STYLE="FLOAT: LEFT" is set for the image.
```

**Table 12-15 FLOAT Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Partial	Partial	Partial	Partial	Partial	Partial	Partial

**Table 12-16 Float / Margin Variant Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Safe

**Table 12-17 Float Elements in Series Variant Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Safe

**Table 12-18 Float on Text Elements Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Partial	Partial	Safe	Safe	Partial	Partial	Partial

FLOAT can take on one of three properties: LEFT, RIGHT and NONE. LEFT displays the element to the left margin of that specified, and RIGHT displays the element to the right margin, and NONE sets the float value to the browser default (which should be flush with the bottom of the image).

Probably the best way to understand how this works is to look at some code and then take a look at the results. For example:

```
<IMG SRC="aurelian.jpg">This is the default value  
for the browser. <BR CLEAR="ALL">
```

What you will get is an image that appears with some text that displayed flush with the bottom of the image. What FLOAT does is very similar to the function of the ALIGN attribute for the <IMG> tag. The following two lines of code produce the same results:

```
<IMG SRC="aurelian.jpg" ALIGN="LEFT">In this case,  
ALIGN="LEFT" is set for the image. <BR CLEAR="ALL">  
  
<P>  
  
<IMG SRC="aurelian.jpg" STYLE="FLOAT: LEFT">In  
this case, ALIGN="LEFT" is not set but  
STYLE="FLOAT: LEFT" is set for the image.
```

Similarly, if you switch ALIGN="LEFT" with ALIGN="RIGHT" and substitute STYLE="FLOAT: LEFT" with STYLE="FLOAT: RIGHT" you will also get the same results, although this time with the image flush to the right margin and the text to the left. You can see the results of this code in Figure 12–17.

As for the value NONE, it should work in the same way as the browser's default value when no float setting is given. All three values for float are displayed in the following code, whose results can be seen in Figure 12–18.

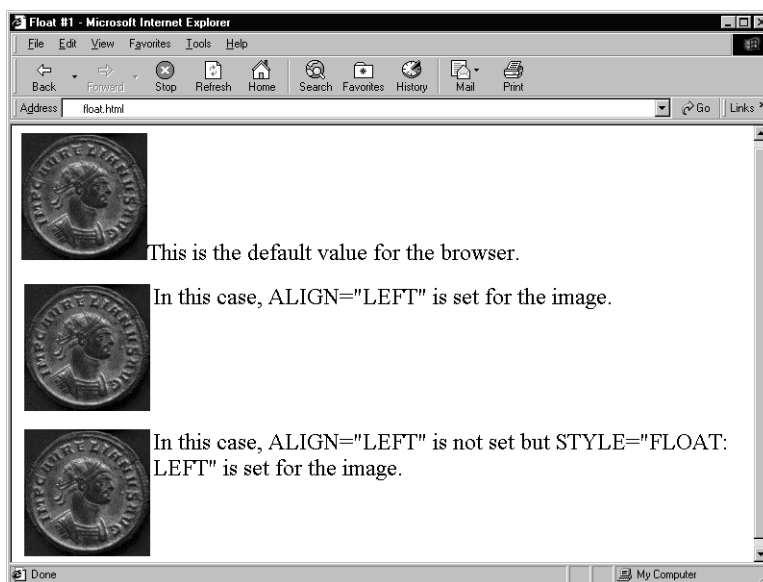
### Listing 12-15 Float #2

```
<HTML>  
<HEAD>  
<TITLE>Float #2</TITLE>  
</HEAD>  
<BODY STYLE="FONT-SIZE: LARGE">  
<IMG SRC="aurelian.jpg" STYLE="FLOAT: NONE;">Image set to  
STYLE="FLOAT: NONE".  
<BR CLEAR="ALL">
```

**Listing 12-15 Float #2 (continued)**

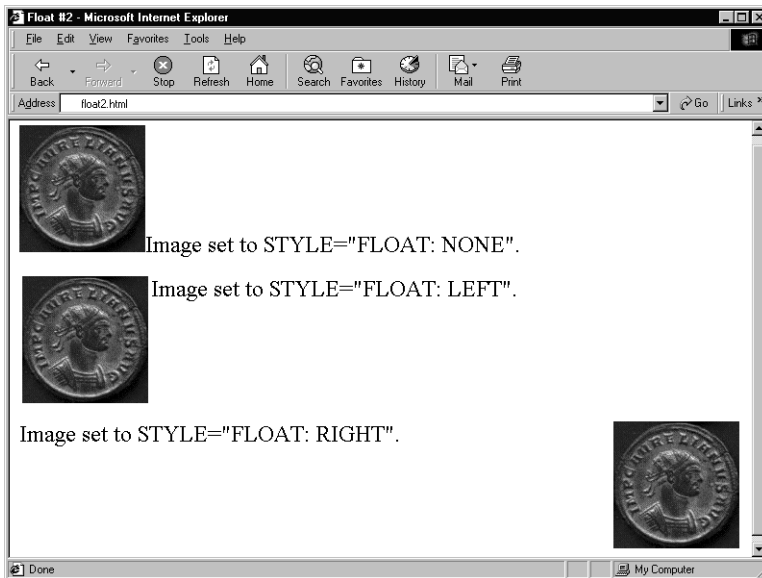
```
<P>
<IMG SRC="aurelian.jpg" STYLE="FLOAT: LEFT;">Image set to
STYLE="FLOAT: LEFT".
<BR CLEAR="ALL">
<P>
<IMG SRC="aurelian.jpg" STYLE="FLOAT: RIGHT;">Image set to
STYLE="FLOAT: RIGHT".
</BODY>
</HTML>
```

---



**Figure 12-17** Sample code for FLOAT and <IMG SRC> where ALIGN=LEFT is set displayed in Internet Explorer.





**Figure 12-18** All three values for the FLOAT property displayed in Internet Explorer.

Internet Explorer beats Netscape Navigator hands down when it comes to supporting the FLOAT property. Internet Explorer almost fully supports the property, even under fairly extreme circumstances. Netscape Navigator supports some of the basic values, but it falls down when trying to handle anything more complicated.

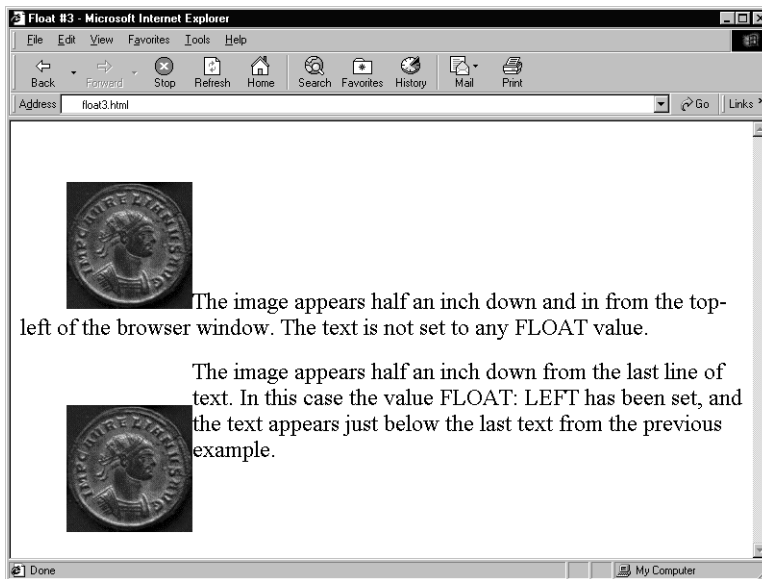
For example, when FLOAT is used when margins are set, the resulting text or other elements that follow a floated element should float in relationship to the margins that are set. In the following code, you can see the difference between non-floated and floated text appearing after an image floats in a margin. In the first example, the text appears flush with the bottom of the text, but in the second, the text appears flush to the top of the margin area, not with the image, which is as it should be. You can see the results of this code correctly displayed in Internet Explorer in Figure 12-19.

### Listing 12-16 Float #3

```

<HTML>
<HEAD>
<TITLE>Float #3</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE">
<IMG SRC="aurelian.jpg" STYLE="MARGIN-TOP: 0.5IN; MARGIN-
LEFT: 0.5IN;">The image appears half an inch down and in
from the top-left of the browser window. The text is not
set to any FLOAT value.<BR CLEAR="ALL">
<P>
<IMG SRC="aurelian.jpg" STYLE="MARGIN-TOP: 0.5IN; MARGIN-
LEFT: 0.5IN; FLOAT: LEFT">The image appears half an inch
down from the last line of text. In this case the value
FLOAT: LEFT has been set, and the text appears just below
the last text from the previous example.<BR CLEAR="ALL">
</BODY>
</HTML>

```



**Figure 12-19** FLOAT property used in conjunction with margins set to an image as displayed in Internet Explorer.

The FLOAT image should also be able to operate in series without any problem. In other words, you should be able to set several elements on a line without having them interfere with each other. You can see this in the following code, which displays in turn three images that should appear on the same line, one after the other, then a sentence whose words should appear on a single line, then a sentence created using the value FLOAT: RIGHT — which was written in reverse. You can see the results of this code correctly displayed in Internet Explorer in Figure 12–20:

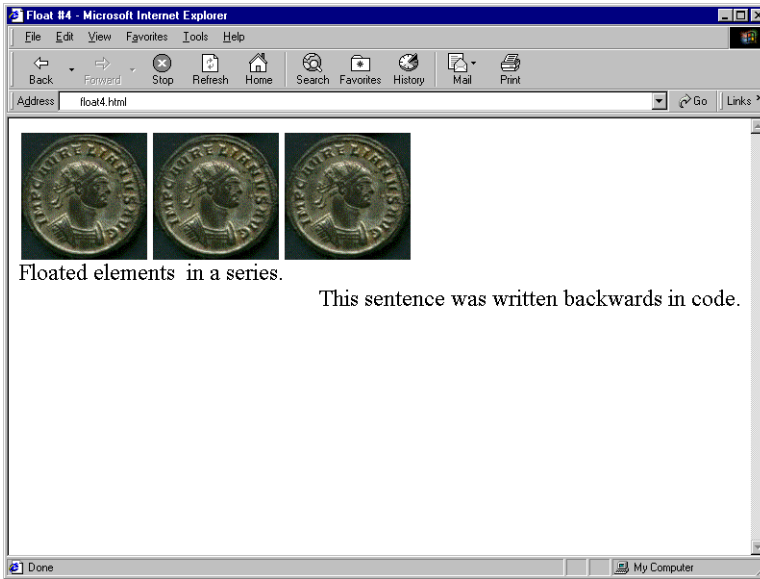
#### Listing 12-17 Float #4

```
<HTML>
<HEAD>
<TITLE>Float #4</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE">
<IMG SRC="aurelian.jpg" STYLE="FLOAT: LEFT">
<IMG SRC="aurelian.jpg" STYLE="FLOAT: LEFT">
<IMG SRC="aurelian.jpg" STYLE="FLOAT: LEFT">
<BR CLEAR="ALL">
<P STYLE="FLOAT: LEFT">Float ed </P>
<P STYLE="FLOAT: LEFT">&nbsp;elements </P>
<P STYLE="FLOAT: LEFT">&nbsp;in a series.</P>
<BR CLEAR="ALL">
<P STYLE="FLOAT: RIGHT">&nbsp;code.</P>
<P STYLE="FLOAT: RIGHT">&nbsp;in</P>
<P STYLE="FLOAT: RIGHT">&nbsp;backwards</P>
<P STYLE="FLOAT: RIGHT">&nbsp;written</P>
<P STYLE="FLOAT: RIGHT">&nbsp;was</P>
<P STYLE="FLOAT: RIGHT">&nbsp;sentence</P>
<P STYLE="FLOAT: RIGHT">This</P>
</BODY>
</HTML>
```

---

Notice that a non-breaking space character (“&nbsp;”) has been added before each word in either sentence. This is because the FLOAT property would otherwise put each word flush with each other, turning the sentence into one long one-word nonsense sentence.

While the FLOAT property appears to work well in Internet Explorer, it often fails when used within a table. Due to this circumstance, plus the minimal support for this property under Netscape Navigator, and given the fact



**Figure 12-20** Several examples of the FLOAT property displayed in a series in Internet Explorer.

that whenever you use `FLOAT`, you are affecting how the page will be laid out in the browser, its use cannot be recommended at this time.

## The HEIGHT Property

The `HEIGHT` property is used to scale an element (most typically an image) to the desired vertical length on a Web page. It can take one of three different properties: a unit of measurement value, a percentage value, and the name value `AUTO`, which sets the element to its default size. It has a sibling property, `WIDTH`, which does the exact same thing to horizontal length.

### ***HEIGHT***

**Description:** This property scales an element (typically an image) to the specified height.

**CSS Family Type:** Boxes

Values:

- **AUTO** - Specifies that the browser default value should be used.
- **%** - Sets a percentage value of the element's width.
- **length (units)** - Specifies the length value as a unit of measurement.

Sample code:

```
<IMG SRC="philco_jnr.jpg" STYLE="HEIGHT: 2.5in">
```

**Table 12-19 HEIGHT Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Safe

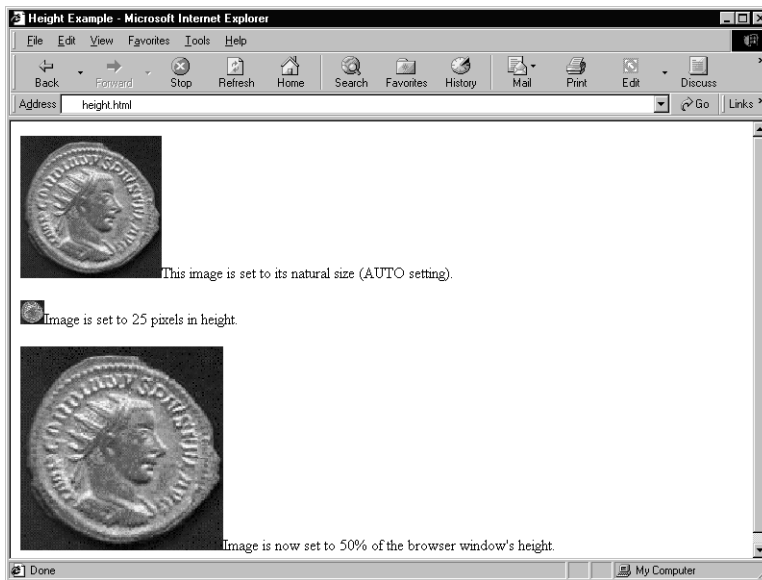
The **AUTO** setting is used to set an image to its normal size. **HEIGHT** can also take a specific measurement value, which sizes the height of the image to the desired value. It can also take a percentage value that sizes the height of the image in relation to the overall height of the browser window. You can see all of these values for **HEIGHT** exhibited in the following code, displayed in Figure 12-21.

**Listing 12-18 Height Example**

```
<HTML>
<HEAD>
<TITLE>Height Example</TITLE>
</HEAD>
<BODY>
<IMG STYLE="HEIGHT: AUTO" SRC="gordian3.jpg">This image is
set to its natural size (AUTO setting).
<P>
<IMG STYLE="HEIGHT: 25PX" SRC="gordian3.jpg">Image is set
to 25 pixels in height.
```

**Listing 12-18 Height Example (continued)**

```
<P>  
<IMG STYLE="HEIGHT: 50%" SRC="gordian3.jpg">Image is now  
set to 50% of the browser window's height.  
</BODY>  
</HTML>
```



**Figure 12-21** Sample HEIGHT code examples as seen in Internet Explorer.

The HEIGHT property is really meant for non-text elements, such as pictures or tables. It is not designed for use with text. When HEIGHT is used in conjunction with text, it does not appreciably affect the display of text on a Web page. There are many other CSS properties more suited to sizing text size (such as the FONT-SIZE or LINE-HEIGHT properties), so this particular use for the HEIGHT property is not recommended.

HEIGHT is fully supported under Internet Explorer, but not at all under Netscape Navigator, so its use cannot be recommended at this time, unfortunately.

# The WIDTH Property

The WIDTH property is used to scale an element (typically an image) to the desired vertical length on a Web page. It can take one of three different properties: a unit of measurement value, a percentage value, and the name value AUTO, which sets the element to its default size. It has a sibling property HEIGHT, which does the exact same thing, but to vertical length.

## WIDTH

Description: Scales an element (most typically an image) to the width specified.

CSS Family Type: Boxes

Values:

- AUTO - Specifies that the browser default value should be used.
- % - Sets a percentage value with respect to the browser window's width.
- *length (units)* - Specifies the length value as a unit of measurement.

Sample code:

```
<IMG STYLE="WIDTH: AUTO" SRC="gordian3.jpg">
Regular size (AUTO setting)
<P>
<IMG STYLE="WIDTH: 25px" SRC="gordian3.jpg"> Set
to 25 pixels
<P>
<IMG STYLE="WIDTH: 25%" SRC="gordian3.jpg"> Set to
25% of the browser window's width
```

**Table 12-20 WIDTH Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Safe

The AUTO setting is used to set an image to its normal size. WIDTH can also take a specific measurement value, which sizes the height of the image to the desired value. It can also take a percentage value that sizes the width of the image in relation to the overall width of the browser window. You can see all of these values for WIDTH exhibited in the following code, displayed in Figure 12-22:

### Listing 12-19 Width

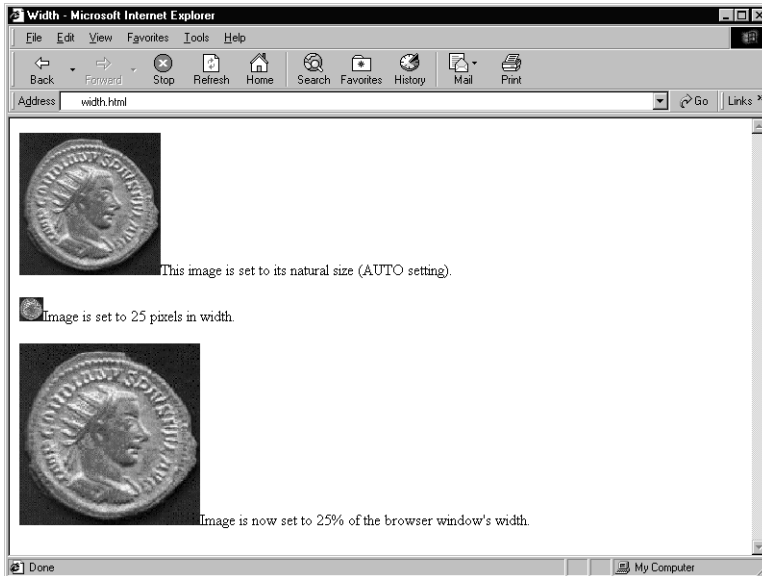
```
<HTML>
<HEAD>
<TITLE>Width</TITLE>
</HEAD>
<BODY>
<IMG STYLE="WIDTH: AUTO" SRC="gordian3.jpg">This image is
set to its natural size (AUTO setting).
<P>
<IMG STYLE="WIDTH: 25PX" SRC="gordian3.jpg">Image is set to
25 pixels in width.
<P>
<IMG STYLE="WIDTH: 25%" SRC="gordian3.jpg">Image is now set
to 25% of the browser window's width.
</BODY>
</HTML>
```

---

The WIDTH property is really meant for non-text elements, such as pictures or tables, although unlike HEIGHT it will display an appreciable effect on text in a Web browser. There are other CSS properties more suited to sizing text horizontally within a Web browser (such as the margin or padding sub-family of properties, which are also examined in this chapter), so this particular use for the WIDTH property is not recommended.

WIDTH is fully supported under Internet Explorer, but only partially under Netscape Navigator. Netscape Navigator does not support the use of WIDTH with such elements as images, but it will with text (as will Internet Explorer). If you set a paragraph to a value of WIDTH: 50%, the text will be displayed from the left margin to the center of the browser window. Its use for formatting images cannot be recommended at this time.





**Figure 12-22** Sample WIDTH code examples as seen in Internet Explorer.

## The PADDING-BOTTOM Property

The PADDING-BOTTOM property is used to add space between one element and another on screen. It can be used to add a specific measurement or percentage value between elements. It shares the same values as its sibling properties PADDING-LEFT, PADDING-RIGHT and PADDING-TOP.

### ***PADDING-BOTTOM***

Sets the amount of padding to the bottom of an element.

CSS Family Type: Boxes (Sub-Family: Padding)

Values:

- **%** - Sets a percentage value of the element's width.
- ***length (units)*** - Specifies the length value as a unit of measurement.

Sample code:

```
<B STYLE="PADDING-BOTTOM: 2IN">This is padded from
the bottom (i.e., from the element below it) by 2
inches.</B>
<P>
Some extra text to demonstrate the effect.
```

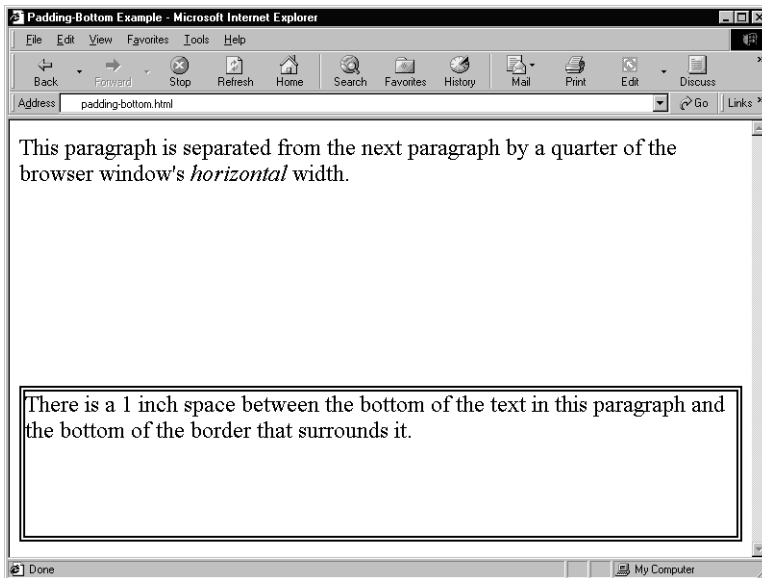
**Table 12-21 PADDING-BOTTOM Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Partial

PADDING-BOTTOM can be used to help space out objects on a Web page — a block-level element, such as a paragraph, surrounded by a border, for example. In this case, the PADDING-BOTTOM property can be used to add space between the text and the bottom of the border. You can see the effects of PADDING-BOTTOM in the following code (seen in Figure 12–23), which separates the top paragraph from the second paragraph, and adds space between the text and the border that surrounds it.

**Listing 12-20 Padding-Bottom Example**

```
<HTML>
<HEAD>
<TITLE>Padding-Bottom Example</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE">
<P STYLE="PADDING-BOTTOM: 25%;">This paragraph is separated
from the next paragraph by a quarter of the browser
window's <I>horizontal</I> width.</P>
<P STYLE="PADDING-BOTTOM: 1IN; BORDER-STYLE: DOUBLE;
BORDER-WIDTH: THICK">There is a 1 inch space between the
bottom of the text in this paragraph and the bottom of the
border that surrounds it.</P>
</BODY>
</HTML>
```



**Figure 12-23** Code example for PADDING-BOTTOM displayed.

The AUTO value sets the padding to its default.

This property is fully supported in both Netscape Navigator and Internet Explorer.

## The PADDING-LEFT Property

The PADDING-LEFT property is used to add space between one element and another on screen. It can be used to add a specific measurement or percentage value between elements. It shares the same values as its sibling properties PADDING-BOTTOM, PADDING-RIGHT and PADDING-TOP.

### **PADDING-LEFT**

Description: Sets the amount of padding to the left of an element.

CSS Family Type: Boxes (Sub-Family: Padding)

Values:

- % - Sets a percentage value of the element's width.

- *length (units)* - Specifies the length value as a unit of measurement.

Sample code:

```
<STRONG STYLE="PADDING-LEFT: 15PT">This is padded
from the left by 15 points.</STRONG>
```

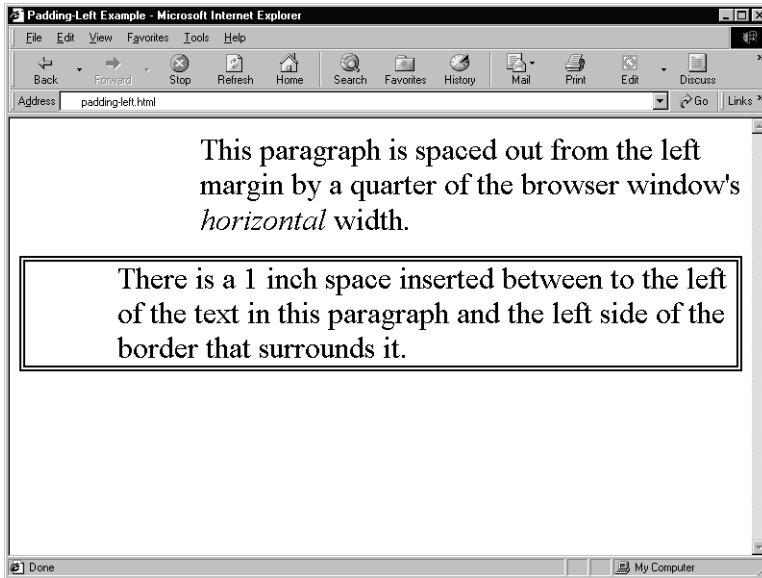
**Table 12-22 PADDING-LEFT Support in Major Browsers**

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> <i>(Mac)</i>	<i>IE 4.5</i> <i>(Mac)</i>	<i>(Win. &amp;</i> <i>UNIX)</i>	<i>NN 4.x</i>	<i>(Mac &amp;</i> <i>UNIX)</i>	<i>Opera</i> <i>3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Partial

PADDING-LEFT can be used to help space out objects on a Web page — a block-level element, such as a paragraph, surrounded by a border, for example. In this case, the PADDING-LEFT property can be used to add space between the text and the left of the border. You can see the effects of PADDING-LEFT in the following code (seen in Figure 12–24), which separates the top paragraph from the left margin, and the text in the second paragraph from the left side of the border in which it is enclosed.

**Listing 12-21 Padding-Left Example**

```
<HTML>
<HEAD>
<TITLE>Padding-Left Example</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: X-LARGE;">
<P STYLE="PADDING-LEFT: 25%;">This paragraph is spaced out
from the left margin by a quarter of the browser window's
<I>horizontal</I> width.</P>
<P STYLE="PADDING-LEFT: 1IN; BORDER-STYLE: DOUBLE; BORDER-
WIDTH: THICK">There is a 1 inch space inserted between to
the left of the text in this paragraph and the left side of
the border that surrounds it.</P>
</BODY>
</HTML>
```



**Figure 12-24** Code example for PADDING-LEFT displayed.

The AUTO value sets the padding to its default.

This property is fully supported in both Netscape Navigator and Internet Explorer.

## The PADDING-RIGHT Property

The PADDING-RIGHT property is used to add space between one element and another on screen. It can be used to add a specific measurement or percentage value between elements. It shares the same values as its sibling properties PADDING-LEFT, PADDING-RIGHT and PADDING-TOP.

### **PADDING-RIGHT**

Description: Sets the amount of padding to the right of an element.

CSS Family Type: Boxes (Sub-Family: Padding)

Values:

- % - Sets a percentage value of the element's width.

- *length (units)* - Specifies the length value as a unit of measurement.

Sample code:

```
<STRONG STYLE="PADDING-RIGHT: 200PX">This text has
a padding of 200 pixels from the right. To
demonstrate the effect properly, you need a
another, long, rambling sentence that just goes on
and on and on and on...</STRONG>
```

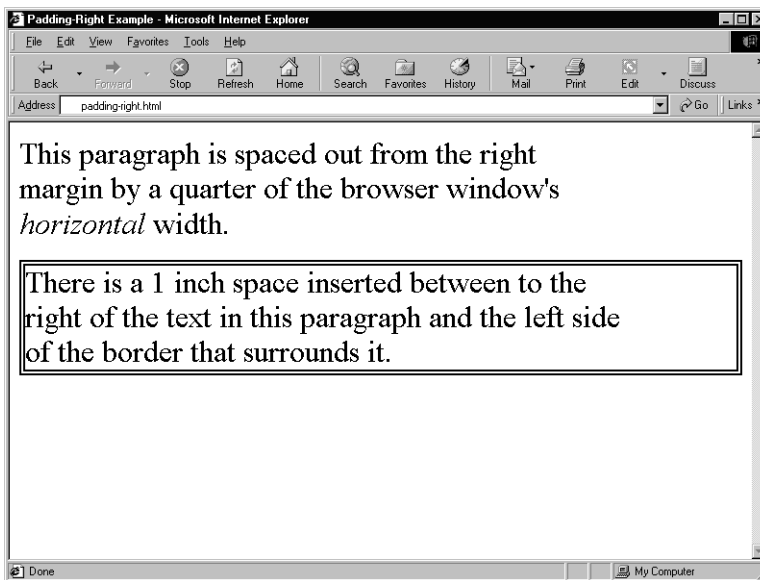
**Table 12-23 PADDING-RIGHT Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Partial

PADDING-RIGHT can be used to help space out objects on a Web page — for example, a block-level element, such as a paragraph, surrounded by a border. In this case, the PADDING-RIGHT property can be used to add space between the text and the right of the border. You can see the effects of PADDING-RIGHT in the following code (seen in Figure 12-25), which separates the top paragraph from the right margin, and the text in the second paragraph from the right side border in which it is enclosed.

**Listing 12-22 Padding-Right**

```
<HTML>
<HEAD>
<TITLE>Padding-Right</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: X-LARGE;">
<P STYLE="PADDING-RIGHT: 25%;">This paragraph is spaced out
from the right margin by a quarter of the browser window's
horizontal width.</P>
<P STYLE="PADDING-RIGHT: 1IN; BORDER-STYLE: DOUBLE; BORDER-
WIDTH: THICK">There is a 1 inch space inserted between to
the right of the text in this paragraph and the left side
of the border that surrounds it.</P>
</BODY>
</HTML>
```



**Figure 12-25** Code example for PADDING-RIGHT displayed.

The AUTO value sets the padding to its default.

This property is fully supported in both Netscape Navigator and Internet Explorer.

## The PADDING-TOP Property

The `PADDING-TOP` property is used to add space between one element and another on screen. It can be used to add a specific measurement or percentage value between elements. It shares the same values as its sibling properties `PADDING-LEFT`, `PADDING-RIGHT` and `PADDING-BOTTOM`.

### ***PADDING-TOP***

Description: Sets the amount of padding to the top of an element.

CSS Family Type: Boxes (Sub-Family: Padding)

Values:

- `%` - Sets a percentage value of the element's width.
- *length (units)* - Specifies the length value as a unit of measurement.

Sample code:

```
<B STYLE="PADDING-TOP: 50MM">This text is padded
from the top by 50 millimeters.</B>
```

**Table 12-24 PADDING-TOP Support in Major Browsers**

				<i>IE 5.0</i>		<i>NN 4.0</i>	
		<i>IE 4.01</i>	<i>IE 4.5</i>	<i>(Win. &amp;</i>		<i>(Mac &amp;</i>	<i>Opera</i>
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>(Mac)</i>	<i>(Mac)</i>	<i>UNIX)</i>	<i>NN 4.x</i>	<i>UNIX)</i>	<i>3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Partial

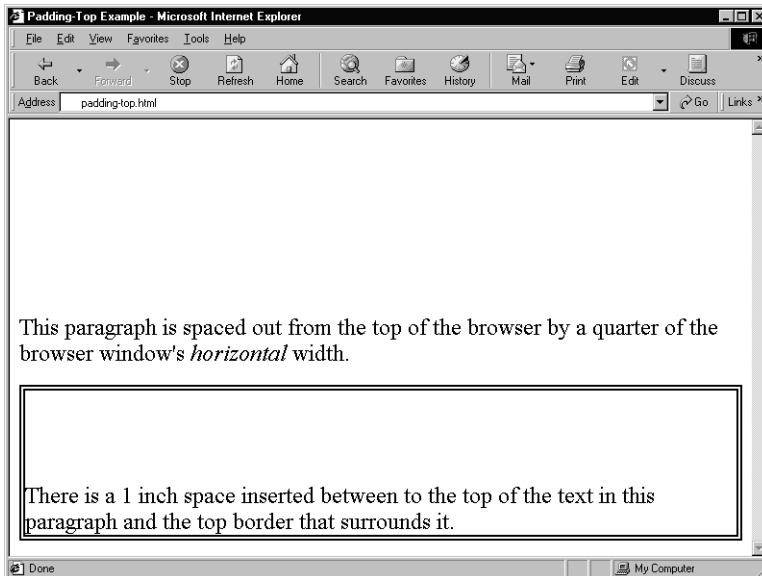
`PADDING-TOP` can be used to help space out objects on a Web page — for example, a block-level element, such as a paragraph, surrounded by a border. In this case, the `PADDING-TOP` property can be used to add space between the text and the top of the border. You can see the effects of `PADDING-TOP` in the following code (seen in Figure 12-26), which separates the top paragraph



from the top of the browser window. The second paragraph uses the property to add space between the text and the top of the border that surrounds it.

### Listing 12-23 Padding-Top Example

```
<HTML>
<HEAD>
<TITLE>Padding-Top Example</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE;">
<P STYLE="PADDING-TOP: 25%;">This paragraph is spaced out
from the top of the browser by a quarter of the browser
window's <I>horizontal</I> width.</P>
<P STYLE="PADDING-TOP: 1IN; BORDER-STYLE: DOUBLE; BORDER-
WIDTH: THICK">There is a 1 inch space inserted between to
the top of the text in this paragraph and the top border
that surrounds it.</P>
</BODY>
</HTML>
```



**Figure 12-26** Code example for PADDING-TOP displayed.

The AUTO value sets the padding to its default.

This property is fully supported in both Netscape Navigator and Internet Explorer.

## The PADDING Property

The PADDING property exists as a type of “shortcut” property that allows Web authors to set the values that belong to PADDING-TOP, PADDING-BOTTOM, PADDING-LEFT and PADDING-RIGHT quickly and easily. As with all of the other properties in its sub-family, PADDING is used to add spacing between elements on a Web page.

### **PADDING**

**Description:** This is an abbreviated element that enables a Web author to set margin values such as PADDING-TOP, PADDING-RIGHT, PADDING-LEFT or PADDING-BOTTOM quickly.

**CSS Family Type:** Boxes (Sub-Family: Padding)

**Values:**

- AUTO - Specifies that the browser default value should be used.
- % - Sets a percentage value of the element's width.
- *n [ n n n ]* measurement units
- If 1 value is present, all borders are set to the numerical value specified
- If 2 values are present, the top and bottom, and then side borders take the numerical values specified
- If 3 values are present, the top, right *and* left, then bottom takes the numerical values specified
- If 4 values are present, the top, right, bottom, then left borders take the numerical values specified

**Sample code:**

```
<IMG SRC="ne_bullet.jpg" STYLE="PADDING: 1cm">
This text is offset 1 cm right and below the
image, and the image is itself offset from the
margins of the browser window by the same amount.
```

**Table 12-25 PADDING Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Partial

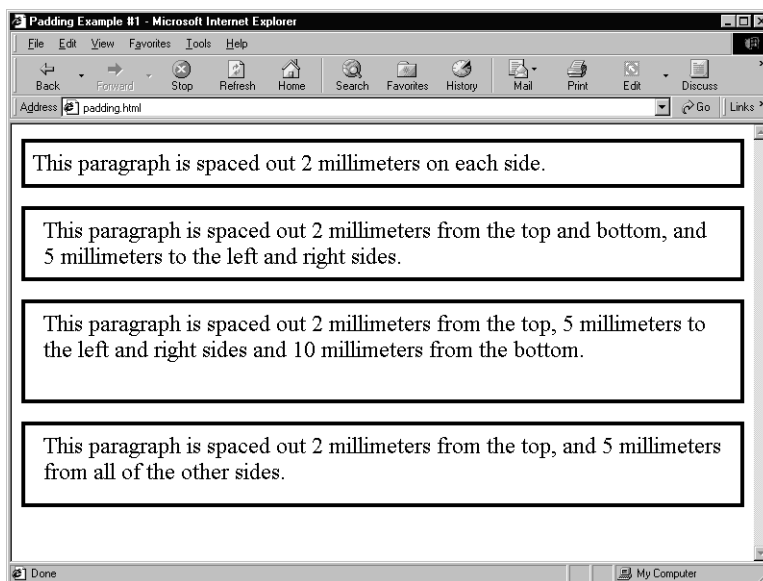
This property can take from one to four separate numerical values, which set the width of the padding. The default value is pixels, so if no measurement unit is specified, the value given is assumed to be a pixel value. If a single numerical value is present, all padding values are set to that value. If two numerical values are present, these padding values are passed on to the top and bottom, and the sides, respectively. If three values are present, the top, right and left, and then bottom take the numerical padding values specified. Finally, if four values are present, the top, right, bottom, then left sides take the numerical padding values specified. When specifying these values, they must be separated by a space, as can be seen in the following code example.

**Listing 12-24 Padding Example #1**

```

<HTML>
<HEAD>
<TITLE>Padding Example #1</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE;">
<P STYLE="PADDING: 2MM; BORDER-STYLE: SOLID;">This
paragraph is spaced out 2 millimeters on each side.</P>
<P STYLE="PADDING: 2MM 5MM; BORDER-STYLE: SOLID;">This
paragraph is spaced out 2 millimeters from the top and
bottom, and 5 millimeters to the left and right sides.</P>
<P STYLE="PADDING: 2MM 5MM 10MM; BORDER-STYLE: SOLID;">This
paragraph is spaced out 2 millimeters from the top, 5
millimeters to the left and right sides and 10 millimeters
from the bottom.</P>
<P STYLE="PADDING: 2MM 5MM 5MM 5MM; BORDER-STYLE:
SOLID;">This paragraph is spaced out 2 millimeters from the
top, and 5 millimeters from all of the other sides.</P>
</BODY>
</HTML>

```



**Figure 12-27** Code example using measurement values for PADDING displayed.

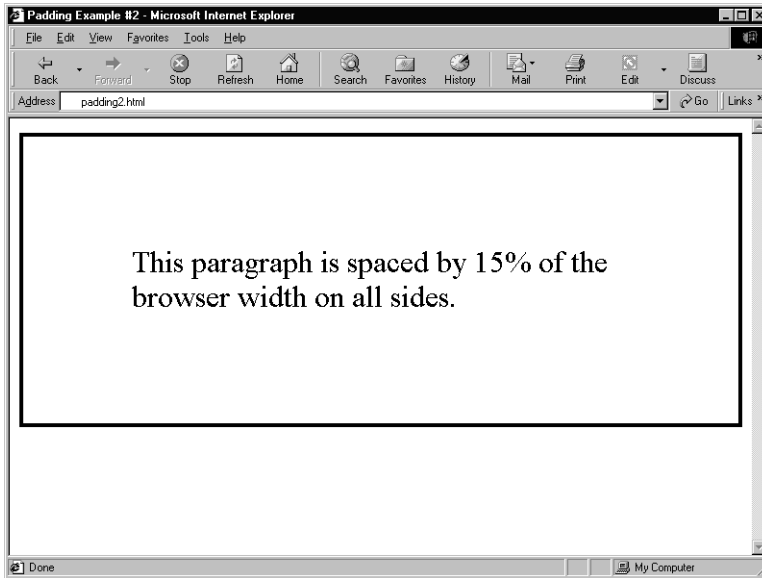
The borders added to each paragraph simply make it easier to see how and where the padding values are taking effect. If you are playing around with the PADDING property, using BORDER-STYLE will help you place your PADDING values correctly.

PADDING can also take a percentage value. The following code shows how this can be done.

```
<HTML>
<HEAD>
<TITLE>Padding Example #2</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: X-LARGE;">
<P STYLE="PADDING: 15%; BORDER-STYLE:
SOLID;">This paragraph is spaced by 15% of the
browser width on all sides.</P>
</BODY>
</HTML>
```

The AUTO value sets the padding to its default.

This property is fully supported in both Netscape Navigator and Internet Explorer.



**Figure 12-28** Code example using a percentage value with PADDING displayed.

## The MARGIN-BOTTOM Property

The MARGIN-BOTTOM property is used to add space between one element and another on screen. In terms of functionality, it is equivalent to the padding sub-family of properties, but the padding properties are meant to set the spacing between the border and the content of a box-level element. The margin sub-family of properties instead sets the space between the browser window and the border of the box property. In other words, the margin set of properties sets the spacing value outside box-level elements, while the padding set of properties sets the spacing value inside the boundaries of box-level elements. The two sets of properties produce the same sorts of effects.

MARGIN-BOTTOM can be used to add a specific measurement or percentage value between elements. It shares the same values as its sibling properties MARGIN-LEFT, MARGIN-RIGHT and MARGIN-TOP.

## MARGIN-BOTTOM

Description: Specifies the distance of an element from the bottom-edge of the browser window or from the next element that appears below it.

CSS Family Type: Boxes (Sub-Family: Margins)

Values:

- **AUTO** - Specifies that the browser default value should be used.
- **%** - Sets a percentage value of the element's *width*.
- **length (units)** - Specifies the length value as a unit of measurement.

Sample code:

```
<TT STYLE="MARGIN-BOTTOM: 1IN">This text is
indented 1 inch from the element below it.</TT>
```

**Table 12-26 MARGIN-BOTTOM Support in Major Browsers**

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i>	<i>IE 4.5</i>	<i>(Win. &amp;</i>	<i>NN 4.x</i>	<i>(Mac &amp;</i>	<i>Opera</i>
		<i>(Mac)</i>	<i>(Mac)</i>	<i>UNIX)</i>		<i>UNIX)</i>	<i>3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Safe

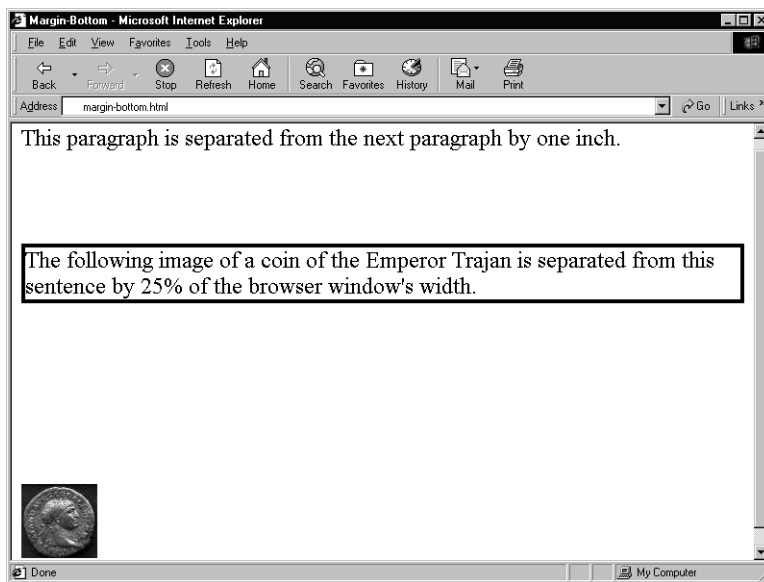
MARGIN-BOTTOM can take one of three different types of values: an AUTO value that sets things to the default spacing value between elements, a precise unit of measurement value or a percentage value. You can see the effects of a measurement value and a percentage value applied to the following code, as seen in Figure 12-29:

**Listing 12-25 Margin-Bottom**

```
<HTML>
<HEAD>
<TITLE>Margin-Bottom</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE">
<P STYLE="MARGIN-BOTTOM: 1IN;">This paragraph is separated
from the next paragraph by one inch. </P>
```

**Listing 12-25 Margin-Bottom (continued)**

```
<P STYLE="MARGIN-BOTTOM: 25%; BORDER-STYLE: SOLID;">The
following image of a coin of the Emperor Trajan is
separated from this sentence by 25% of the browser window's
width.</P>
<IMG SRC="trajan.jpg">
</BODY>
</HTML>
```



**Figure 12-29** Code example using a set measurement value and a percentage value with the MARGIN-BOTTOM property.

Note that in the previous code example, the image and the second sentence are separated by a distance equivalent to a quarter of the browser window's *width*. This is not what you may expect; it would seem to make more sense that the browser's windows *height* be used, instead. In fact, when a percentage value is used with MARGIN-BOTTOM, it takes as the base value the width of the element in which it is set. In this case, it takes as its value the width of the sentence, which stretches across the width of the browser's

window. It is important to keep this in mind whenever you try to set a percentage value with `MARGIN-BOTTOM` or any of its sibling properties.

This property is fully supported in the recent versions of Internet Explorer and Netscape Navigator, and so is considered safe to use.

## The `MARGIN-LEFT` Property

The `MARGIN-LEFT` property is used to add space between one element and another on screen. In terms of functionality, it is equivalent to the padding sub-family of properties, but the padding properties are meant to set the spacing between the border and the content of a box-level element. The margin sub-family of properties instead sets the space between the browser window and the border of the box property. In other words, the margin set of properties sets the spacing value outside box-level elements, while the padding set of properties sets the spacing value inside the boundaries of box-level elements. The two sets of properties produce the same sorts of effects.

`MARGIN-LEFT` can be used to add a specific measurement or percentage value between elements. It shares the same values as its sibling properties `MARGIN-BOTTOM`, `MARGIN-RIGHT` and `MARGIN-TOP`.

### **`MARGIN-LEFT`**

Description: Specifies the distance of an element from the left-edge of the browser window or from the element that appears immediately to the left.

CSS Family Type: Boxes (Sub-Family: Margins)

Values:

- `AUTO` - Specifies that the browser default value should be used.
- `%` - Sets a percentage value of the element's width.
- ***length (units)*** - Specifies the length value as a unit of measurement.

Sample code:

```
<H1 STYLE="MARGIN-LEFT: 2.5IN">This header is  
indented 2 and a half inches from the left  
margin.</H1>
```



**Table 12-27 MARGIN-LEFT Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Partial	Safe	Safe	Safe	Safe	Safe	Safe	Partial

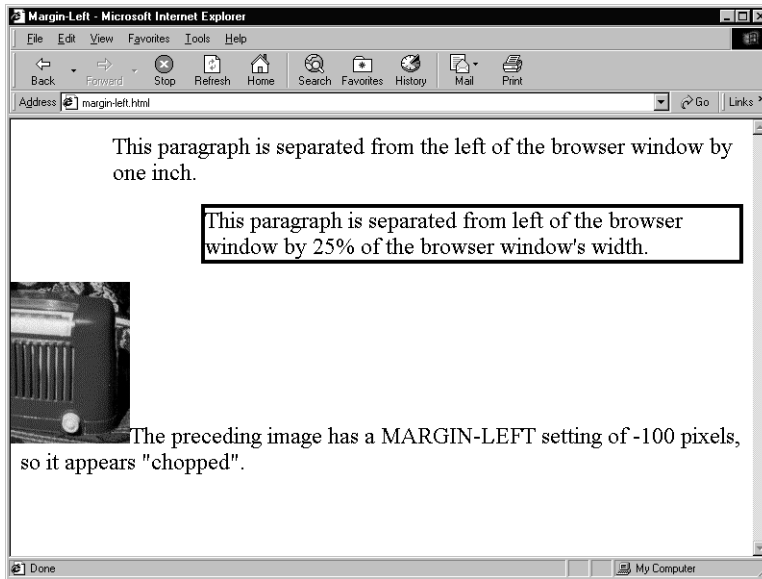
MARGIN-LEFT can take one of three different types of values: an AUTO value that sets things to the default spacing value between elements, a precise unit of measurement value or a percentage value. You can see the effects of a measurement value, a percentage value and a negative value applied to the following code, seen in Figure 12-30:

**Listing 12-26 Margin-Left**

```

<HTML>
<HEAD>
<TITLE>Margin-Left</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE">
<P STYLE="MARGIN-LEFT: 1IN;">This paragraph is separated
from the left of the browser window by one inch. </P>
<P STYLE="MARGIN-LEFT: 25%; BORDER-STYLE: SOLID;">This
paragraph is separated from left of the browser window by
25% of the browser window's width.</P>
<IMG SRC="addison55.jpg" STYLE="MARGIN-LEFT: -100PX;">The
preceding image has a MARGIN-LEFT setting of -100 pixels,
so it appears "chopped".<P>
</BODY>
</HTML>

```



**Figure 12-30** Code example using a set measurement value, a percentage value and a negative value with the MARGIN-LEFT property.

This property is fully supported in the recent versions of Internet Explorer and Netscape Navigator, and so is considered safe to use.

## The MARGIN-RIGHT Property

The MARGIN-RIGHT property is used to add space between one element and another on screen. In terms of functionality, it is equivalent to the padding sub-family of properties, but the padding properties are meant to set the spacing between the border and the content of a box-level element. The margin sub-family of properties instead sets the space between the browser window and the border of the box property. In other words, the margin set of properties sets the spacing value outside box-level elements, while the padding set of properties sets the spacing value inside the boundaries of box-level elements. The two sets of properties produce the same sorts of effects.

MARGIN-RIGHT can be used to add a specific measurement or percentage value between elements. It shares the same values as its sibling properties MARGIN-BOTTOM, MARGIN-LEFT and MARGIN-TOP.

## MARGIN-RIGHT

Description:

Specifies the distance of an element from the right-edge of the browser window or from the next element that appears to its right.

CSS Family Type: Boxes (Sub-Family: Margins)

Values:

- **AUTO** - Specifies that the browser default value should be used.
- **%** - Sets a percentage value of the element's width.
- ***length (units)*** - Specifies the length value as a unit of measurement.

Sample code:

```
<IMG SRC="trajan.jpg" STYLE="MARGIN-RIGHT:
1IN">This text is set 1 inch from the adjacent
image.
```

**Table 12-28 MARGIN-RIGHT Support in Major Browsers**

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> (Mac)	<i>IE 4.5</i> (Mac)	(Win. & UNIX)	<i>NN 4.x</i>	(Mac & UNIX)	<i>Opera</i> 3.6
Partial	Safe	Safe	Safe	Safe	Safe	Safe	Partial

MARGIN-RIGHT can take one of three different types of values: an AUTO value that sets things to the default spacing value between elements, a precise unit of measurement value or a percentage value. You can see the effects of a measurement value, a percentage value and a negative value applied to the following code, seen in Figure 12-31:

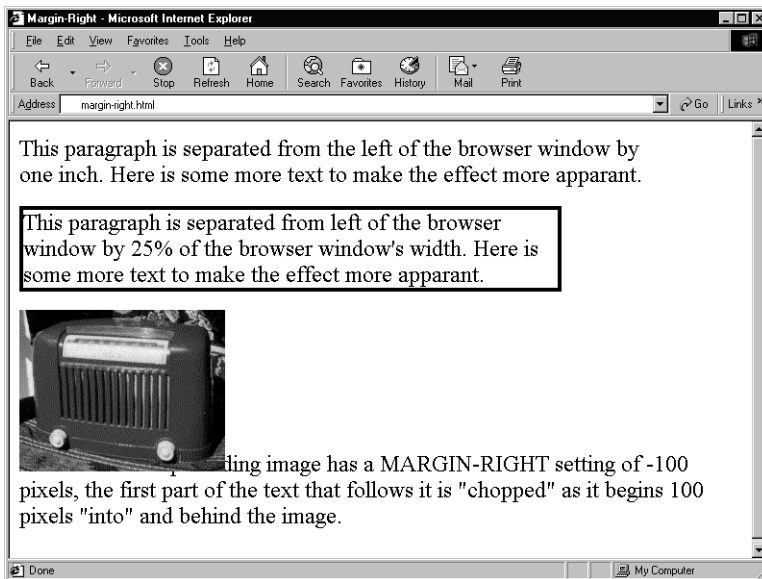
**Listing 12-27 Margin-Right**

```
<HTML>
<HEAD>
<TITLE>Margin-Right</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE">
<P STYLE="MARGIN-RIGHT: 1in;">This paragraph is separated
from the left of the browser window by one inch. Here is
some more text to make the effect more apparent.</P>
<P STYLE="MARGIN-RIGHT: 25%; BORDER-STYLE: SOLID;">This
paragraph is separated from left of the browser window by
25% of the browser window's width. Here is some more text
to make the effect more apparent.</P>
<IMG SRC="addison55.jpg" STYLE="MARGIN-RIGHT: -100PX;">The
preceding image has a MARGIN-RIGHT setting of -100 pixels,
so the first part of the text that follows it is "chopped"
as it begins 100 pixels "into" and behind the image.<P>
</BODY>
</HTML>
```

---

Notice that the text is all indented from the right margin of the browser window by the amounts specified in the code. The only effect that may surprise some people is that of a negative value applied to the image, and its effects on the text that follows it, with the first couple of words are covered or partially obscured by the image. This effect occurs because the negative value applied to `MARGIN-RIGHT` is applied within the boundary of the image itself. Therefore the text begins 100 pixels “within” the right margin of the image itself.

This property is fully supported in the recent versions of Internet Explorer and Netscape Navigator, and so is considered safe to use.



**Figure 12-31** Code example using a set measurement value, a percentage value and a negative value with the MARGIN-RIGHT property.

## The MARGIN-TOP Property

The MARGIN-TOP property is used to add space between one element and another on screen. In terms of functionality, it is equivalent to the padding sub-family of properties, but the padding properties are meant to set the spacing between the border and the content of a box-level element. The margin sub-family of properties instead sets the space between the browser window and the border of the box property. In other words, the margin set of properties sets the spacing value outside box-level elements, while the padding set of properties sets the spacing value inside the boundaries of box-level elements. The two sets of properties produce the same sorts of effects.

MARGIN-TOP can be used to add a specific measurement or percentage value between elements. It shares the same values as its sibling properties MARGIN-BOTTOM, MARGIN-LEFT and MARGIN-RIGHT.

## MARGIN-TOP

Description:

Specifies the distance of an element from the top-edge of the browser window or from the element that appears above it.

CSS Family Type: Boxes (Sub-Family: Margins)

Values:

- **AUTO** - Specifies that the browser default value should be used.
- **%** - Sets a percentage value of the element's width.
- ***length (units)*** - Specifies the length value as a unit of measurement.

Sample code:

```
<H1 STYLE="MARGIN-TOP: 150PX">Header situated 150
pixels from the top margin.</H1>
```

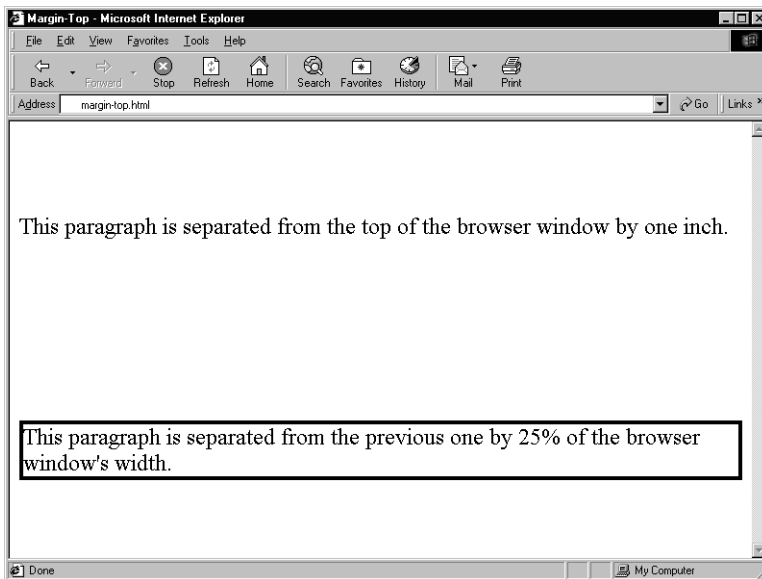
**Table 12-29 MARGIN-TOP Support in Major Browsers**

<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01 (Mac)</b></i>	<i><b>IE 4.5 (Mac)</b></i>	<i><b>IE 5.0 (Win. &amp; UNIX)</b></i>	<i><b>NN 4.x</b></i>	<i><b>NN 4.0 (Mac &amp; UNIX)</b></i>	<i><b>Opera 3.6</b></i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Safe

MARGIN-TOP can take one of three different types of values: an AUTO value that sets things to the default spacing value between elements, a precise unit of measurement value or a percentage value. You can see the effects of a measurement value and a percentage value applied to the following code, seen in Figure 12-32:

**Listing 12-28 Margin-Top**

```
<HTML>
<HEAD>
<TITLE>Margin-Top</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE">
<P STYLE="MARGIN-TOP: 1IN;">This paragraph is separated
from the top of the browser window by one inch. </P>
<P STYLE="MARGIN-TOP: 25%; BORDER-STYLE: SOLID;">This
paragraph is separated from the previous one by 25% of the
browser window's width.</P>
</BODY>
</HTML>
```



**Figure 12-32** Code example using a set measurement value and a percentage value with the MARGIN-TOP property.

Note that in the previous code example, the image and the second sentence are separated by a distance equivalent to a quarter of the browser

window's *width*. This is not what you may expect; it would seem to make more sense to take a percentage value relative to the browser's window *height* instead. In fact, when a percentage value is used with `MARGIN-TOP`, it takes as the base value the width of the element in which it is set. In this case, it takes as its value the width of the sentence, which stretches across the width of the browser's window. It is important to keep this in mind whenever you try to set a percentage value with `MARGIN-TOP` or any of its sibling properties.

This property is fully supported in the recent versions of Internet Explorer and Netscape Navigator, and so is considered safe to use.

## The MARGIN Property

This is a “shortcut” property that allows the Web author to quickly set margins that can combine the values of the rest of the margin sub-family of properties: `MARGIN-TOP`, `MARGIN-BOTTOM`, `MARGIN-LEFT` and `MARGIN-RIGHT`. In fact, this property makes things easier if you intend to set margin values for more than one side of a Web element.

### MARGIN

Description: This is an abbreviated property that enables a Web author to set margin values such as `MARGIN-TOP`, `MARGIN-RIGHT`, `MARGIN-LEFT` or `MARGIN-BOTTOM` quickly.

CSS Family Type: Boxes (Sub-Family: Margins)

Values:

- `AUTO` - Specifies that the browser default value should be used.
- `%` - Sets a percentage value of the element's width.
- `n [ n n n ] measurement_units`
- If 1 value is present, all borders are set to the numerical value specified
- If 2 values are present, the top and bottom, and then side borders take the numerical values specified
- If 3 values are present, the top, right *and* left, then bottom takes the numerical values specified
- If 4 values are present, the top, right, bottom, then left borders take the numerical values specified

Sample code:



```
<H1 STYLE="MARGIN: 2in">This text has a margin set
to two inches.</H1>
```

**Table 12-30 MARGIN Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Partial	Partial	Partial	Partial	Partial	Partial	Partial	Partial

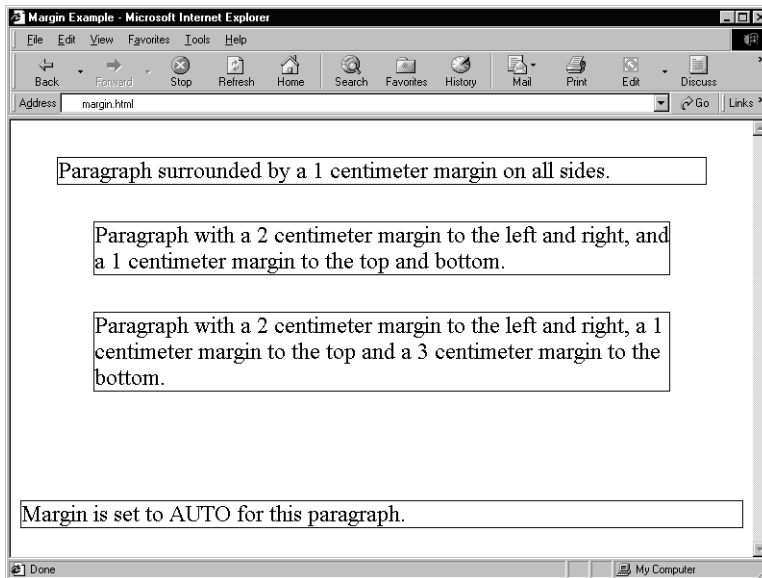
This property can take from one to four separate numerical values, which work in exactly the same manner as with the `PADDING` and `BORDER` properties. In this case, it sets the width of the selected margin. The default value is pixels, so if no measurement unit is specified, the value given is assumed to be a pixel value. If a single numerical value is present, all borders are set to that value. If two numerical values are present, the top and bottom, and then the side borders take on those values, respectively. If three values are present, the top, right and left, and then the bottom take the numerical values specified. Finally, if four values are present, the top, right, bottom, then left borders take the numerical values specified. When specifying these values, they must be accompanied by the measurement value to be used and they must all be separated by a space, as can be seen in the following code example.

**Listing 12-29 Margin Example #1**

```
<HTML>
<HEAD>
<TITLE>Margin Example #1</TITLE>
</HEAD>
</HTML>
<BODY STYLE="FONT-SIZE: LARGE;">
<P STYLE="MARGIN: 1CM; BORDER-STYLE: SOLID; BORDER-WIDTH:
1PX;">Paragraph surrounded by a 1 centimeter margin on all
sides.</P>
```

**Listing 12-29 Margin Example #1 (continued)**

```
<P STYLE="MARGIN: 1CM 2CM; BORDER-STYLE: SOLID; BORDER-  
WIDTH: 1PX;">Paragraph with a 2 centimeter margin to the  
left and right, and a 1 centimeter margin to the top and  
bottom.</P>  
<P STYLE="MARGIN: 1CM 2CM 3CM; BORDER-STYLE: SOLID; BORDER-  
WIDTH: 1PX;">Paragraph with a 2 centimeter margin to the  
left and right, a 1 centimeter margin to the top and a 3  
centimeter margin to the bottom.</P>  
<P STYLE="MARGIN: AUTO; BORDER-STYLE: SOLID; BORDER-WIDTH:  
1PX;">Margin is set to AUTO for this paragraph.</P>  
</BODY>  
</HTML>
```

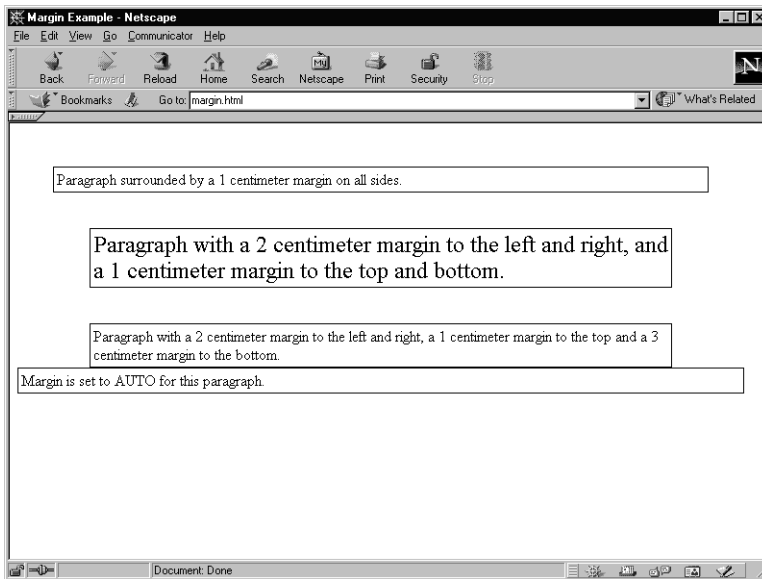


**Figure 12-33** Code example for the MARGIN property as seen in Internet Explorer.

As you can see in Figure 12-33, in the first example a 1 centimeter margin is set around the paragraph on all sides. In the second, a 2 centimeter margin is set on the left and right sides while 1 centimeter margin is set between to

its top and bottom sides. The third example is like the second, except that there is a 3 centimeter margin between it and the fourth paragraph.

The fourth paragraph is a problematic example, however, as it uses the MARGIN property value AUTO, which is not well defined in the official CSS specification. In Internet Explorer, it simply accepts whatever previous margin values may have been set, but does not take any special margin values of its own (in Figure 12–33, note how the border for the paragraph spans the browser window's entire width). Netscape Navigator uses a different definition of AUTO, which ignores any previously set margin values, and places the Web element immediately underneath the last element to appear on screen. You can see how Netscape Navigator renders the same code in Figure 12–34.



**Figure 12–34** Code example for the MARGIN property as seen in Netscape Navigator.

As you can see, Netscape Navigator is somewhat flaky when it comes to interpreting some CSS elements that have been properly set in the code (such as the FONT-SIZE property set to the <BODY> tag in the page), but otherwise it displays the margin code just fine. Note how it places the fourth paragraph directly underneath the third paragraph, even though the third paragraph supposedly has a 3 centimeter margin set between it and the fourth paragraph.

So, which interpretation is right? Since the official CSS specification does not provide thorough details on this point, arguably both interpretations are correct. The main thing to remember is that when you are using the `MARGIN` property with the value `AUTO`, expect different results in the two major browsers.

`MARGIN` can also take a percentage value, which works in exactly the same way as with the other margin sub-family of properties. When a percentage value is used with `MARGIN`, it takes as the base value the width of the element in which it is set, since this value is meant to be applied to box-level elements that stretch across the width of the sentence.



# THE CLASSIFICATION FAMILY OF PROPERTIES

## Topics in this Chapter

- The DISPLAY Property
- The WHITE-SPACE Property
- The LIST-STYLE-TYPE Property
- The LIST-STYLE-IMAGE Property
- The LIST-STYLE-POSITION Property
- The LIST-STYLE Property



# *Chapter* 13

The classification family of properties is a powerful group of elements that can fundamentally change the way a particular CSS property functions. These properties are designed to classify elements into categories, and not just to set a specific on-screen display.

This chapter looks at the following CSS properties:

- DISPLAY
- WHITE-SPACE
- LIST-STYLE-TYPE
- LIST-STYLE-IMAGE
- LIST-STYLE-POSITION
- LIST-STYLE

## The DISPLAY Property

The DISPLAY property belongs to the classification group of CSS elements. This is a particularly powerful property to play with. For example, using the DISPLAY property, you can make a bulleted item appear as if it were a regular block of text, make a hyperlink appear as regular text, and more.

## DISPLAY

Description: Controls the fundamental nature of the specified HTML tag.

CSS Family Type: Classification

Value:

- **BLOCK** - Sets the specified element as a block-type element.
- **INLINE** - Sets the specified element as an inline-type element.
- **LIST-ITEM** - Sets the specified element as a type of list-item.
- **NONE** - Switches off the display of the specified element.

Sample code:

```
<I STYLE="DISPLAY: LIST-ITEM">This italicized text
is indented in the same way a list-item would be.</I>
```

**Table 13-1 DISPLAY Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Unsafe	Unsafe	Partial	Partial	Partial	Partial	Partial

The **DISPLAY** property has four values: **BLOCK**, **INLINE**, **LIST-ITEM** and **NONE**. **BLOCK** sets the specified element to appear as a block-type element, which includes those tags that always appear on a full line by themselves (such as `<P>`, `<H1>` or `<BLOCKQUOTE>`). Contrast this to the **INLINE** setting, which deals with inline elements, including those tags that alter the appearance of text within a line and do not always begin and end a line (such as `<I>`, `<B>` or `<EM>`). The **LIST-ITEM** value makes the associated tag appear offset as if it were a bulleted item in a list. Finally, a value of **NONE** *removes* the typical formatting normally associated with a given HTML tag.

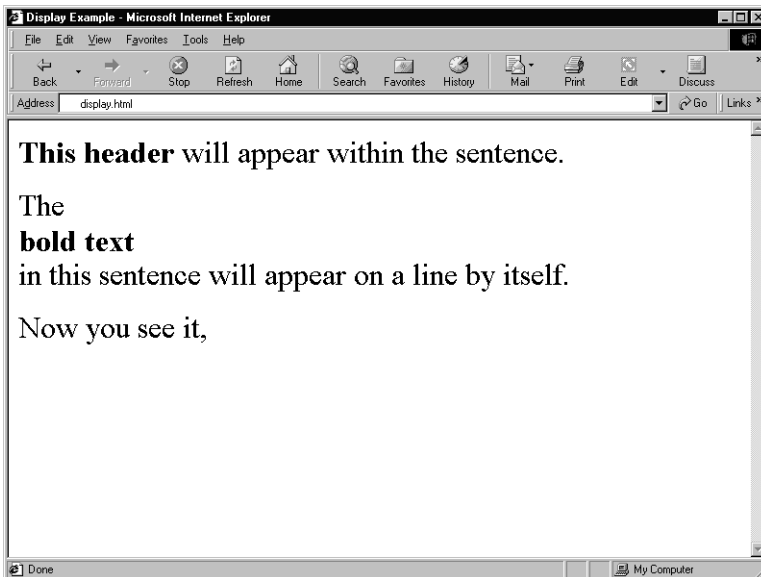
You can create some interesting effects with this property. For example, since list elements are always indented from the left margin, you could use that to your benefit. You can also take a header and stick it within a normal sentence. You can even make text disappear entirely from the page. You can



see several of these vales displayed in the following code, which is depicted in Figure 13-1:

### Listing 13-1 Display Example

```
<HTML>
<HEAD>
<TITLE>Display Example</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: X-LARGE;">
<P>
<H1 STYLE="DISPLAY: INLINE">This header</H1> &nbsp;  will
appear within the sentence.
<P>
The <B STYLE="DISPLAY: BLOCK">bold text</B> in this
sentence will appear on a line by itself.
<P>
Now you see it, <B STYLE="DISPLAY: NONE">now you don't.</B>
<P>
</BODY>
</HTML>
```



**Figure 13-1** Code example for the DISPLAY property as seen in Internet Explorer.

DISPLAY is rather an odd CSS property, since there are arguably easier and more precise ways to accomplish many of the same things using other CSS properties. For example, CSS provides better ways of setting margins for text than that used the first example. But at least this example gives you an idea as to what is possible with the DISPLAY property.

There is partial support for this element in the most recent versions of Netscape Navigator; it does not fully support the INLINE value. DISPLAY is fully supported within the most recent versions of Internet Explorer.

## The WHITE-SPACE Property

The WHITE-SPACE property is designed to add extra spaces (i.e., “white space”) between words. It does not function in the same way as WORD-SPACING or LETTER-SPACING do, by setting a fixed value that is used to separate the individual letters or words. Instead, it more closely resembles the <PRE> HTML tag. The WHITE-SPACE property can take one of three different values: NORMAL, NOWRAP and, tellingly, PRE.

### WHITE-SPACE

Description:

This property is used to set white space and the manner in which carriage returns are handled within a Web page.

CSS Family Type: Classification

Values:

- NORMAL - The default browser behavior for the HTML tag with which this element is associated.
- NOWRAP - When this value is set, carriage returns and linefeeds are rendered as single space. Line breaks are not effected, however.
- PRE - When this value is set, all spaces, carriage returns and linefeeds in the document are displayed “as is”.

Sample code:

```
<P STYLE="WHITE-SPACE: PRE">
The   extra white   spaces and carriage
returns in this sentence
are displayed.
</P>
```

**Table 13-2 WHITE-SPACE Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Unsafe	Unsafe	Unsafe	Unsafe	Partial	Partial	Unsafe

One of the tenets of Web formatting is that you are not normally allowed to have more than one horizontal space separating words in a line. In the days prior to the advent of the word processor, most typists added a couple of spaces between the end of one sentence and the beginning of another. The common form these days is to only have a single space between sentences. Accordingly, whenever somebody adds more than one space between the words in a sentence, the Web browser automatically subtracts them so that only a single space remains. Before the WHITE-SPACE property, Web authors either had the choice of adding multiple non-breaking spaces (“&nbsp;”) between words, or using the <PRE> (“preformatted text”) HTML tag. Unfortunately, this property gets little support from the two major browsers.

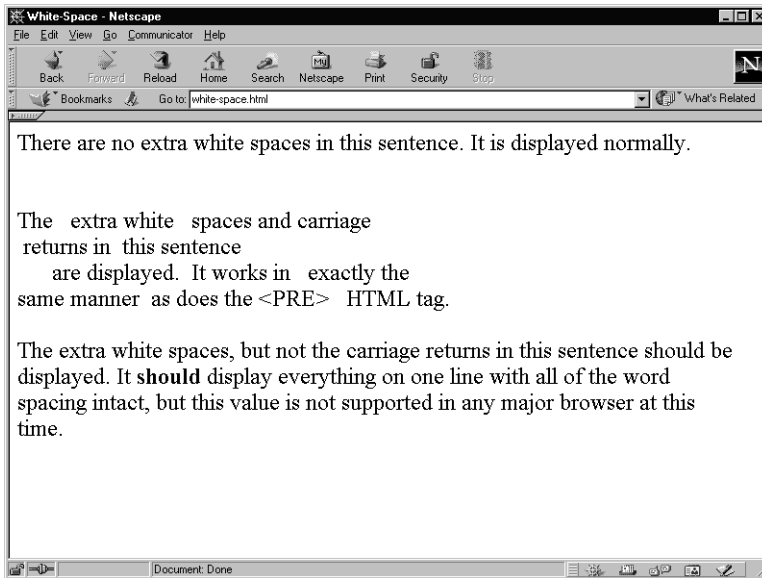
The NORMAL value for WHITE-SPACE simply tells the browser to format a line of text as it normally would, by removing excess spaces that appear between words. PRE works exactly like the <PRE> HTML tag, retaining the original spacing and carriage returns between letters and words on a page. The NOWRAP value also retains the original spacing between letters and words, but ignores any carriage returns. The intended effect is to display everything on one line, with all of the spacing between letters and words retained; using this property, you would have to insert a line break (“<BR>”) to insert a carriage return. You can see all three properties put to use in the following code sample.

**Listing 13-2 White-Space**

```
<HTML>
<HEAD>
<TITLE>White-Space</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE;">
<P STYLE="WHITE-SPACE: NORMAL">
There   are no   extra white spaces   in
      this sentence.   It is   displayed normally.
</P>
<P STYLE="WHITE-SPACE: PRE">
The   extra white   spaces and carriage
      returns in   this sentence
          are displayed. It works in   exactly the
same manner   as does the &lt;PRE> HTML tag.
</P>
<P STYLE="WHITE-SPACE: NOWRAP">
The   extra white   spaces, but not the   carriage
      returns in   this sentence
          should be   displayed.
It <B>should</B> display everything on one
      line with all   of the word spacing intact,
but this value is not   supported in   any
      major browser at   this time.
</P>
</BODY>
</HTML>
```

---

Unfortunately, `WHITE-SPACE` is not supported within Internet Explorer, and only partially under Netscape Navigator, which supports the `PRE` property, in addition to the default `NORMAL` value. Its use cannot be recommended at this time.



**Figure 13-2** Code example for the WHITE-SPACE property as seen in Netscape Navigator (note that the NOWRAP property is not supported).

## The LIST-STYLE-TYPE Property

The LIST-STYLE-TYPE property is one of a sub-family of list properties, including LIST-STYLE-IMAGE, LIST-STYLE-POSITION and LIST-STYLE, that are designed to alter the way the list marker at the beginning of a list element appears. Normally, when you create an unordered list (“<UL>”), a simple black circle precedes every list element (“<LI>”) in the list. If you create an ordered list (“<OL>”), each list element is preceded by a number. The LIST-STYLE-TYPE property provides the Web author with greater flexibility in setting the type of list marker that appears before a list element, ranging from different types of “dots” to upper- or lower-case roman numerals.

### **LIST-STYLE-TYPE**

Description: This property sets the type of list-markers that appear before list items.

## CSS Family Type: Classification

## Values:

- **CIRCLE | DISC | SQUARE** - Sets the type of symbol to appear before a list element.
- **DECIMAL | LOWER-ALPHA | LOWER-ROMAN | UPPER-ALPHA | UPPER-ROMAN** - Sets the type of alphanumeric symbol that appears before a list marker.
- **NONE** - No list-marker is displayed before a list item.

## Sample code:

```
<UL STYLE="LIST-STYLE-TYPE: LOWER-ALPHA">
<LI>List items
<LI>preceded by
<LI>lower-case letters
</UL>
```

**Table 13-3 LIST-STYLE-TYPE Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Safe

There are three different values for setting the type of “dot” list marker that can appear: **DISC**, **CIRCLE** and **SQUARE**. **DISC** produces the default filled-in black “dot” most commonly associated with individual list elements appearing in an unordered list. **CIRCLE** displays an unfilled, white-centered circle on screen, and **SQUARE** displays a black-filled square. Here’s a list showing what you should expect to see for each type:

- list item preceded by a Disc
- list item preceded by a Circle
- list item preceded by a Square.

There are five different values that can be set for the type of alphanumeric list marker that most commonly appears before ordered list elements: **DECIMAL**, **LOWER-ALPHA**, **UPPER-ALPHA**, **LOWER-ROMAN** and **UPPER-ROMAN**. **DECIMAL** displays a regular number, which is the default value

for all ordered list elements. LOWER-ALPHA and UPPER-ALPHA display lower- and upper-case letters in alphabetical order for each list element. Similarly, LOWER-ROMAN and UPPER-ROMAN display lower- and upper-case roman numerals in numerical order for each list element, as demonstrated in the following list:

- Decimal: 1, 2, 3...
- Lower-Alpha: a, b, c...
- Upper-Alpha: A, B, C...
- Lower-Roman: i, ii, iii...
- Upper-Roman: I, II, III...

In addition to these values, there is the additional value NONE, which displays no list marker before each list element, but still indents each list item as usual. You can see all of these values used in the following code, which is displayed in Figure 13-3:

### Listing 13-3 List-Style-Type

```
<HTML>
<HEAD>
<TITLE>List-Style-Type</TITLE>
</HEAD>
<BODY>
<TABLE STYLE="FONT-SIZE: LARGE;">
<TR>
<TD VALIGN="TOP">
<UL STYLE="LIST-STYLE-TYPE: DISC">
<LI>List items
<LI>preceded by
<LI>discs (the default value)
</UL>
<UL STYLE="LIST-STYLE-TYPE: CIRCLE">
<LI>List items
<LI>preceded by
<LI>hollow circles
</UL>
```

### Listing 13-3 List-Style-Type (continued)

```

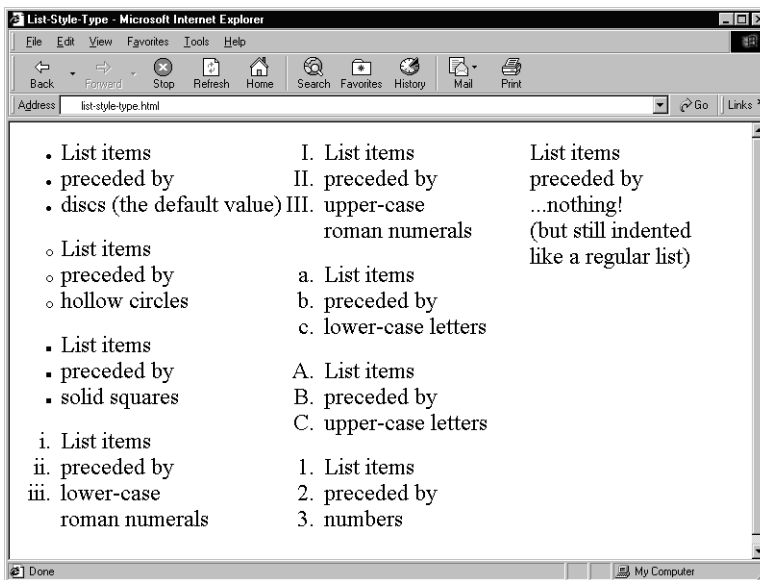
<UL STYLE="LIST-STYLE-TYPE: SQUARE">
<LI>List items
<LI>preceded by
<LI>solid squares
</UL>
<OL STYLE="LIST-STYLE-TYPE: LOWER-ROMAN">
<LI>List items
<LI>preceded by
<LI>lower-case <BR>roman numerals
</OL>
</TD>
<TD VALIGN="TOP">
<OL STYLE="LIST-STYLE-TYPE: UPPER-ROMAN">
<LI>List items
<LI>preceded by
<LI>upper-case <BR>roman numerals
</OL>
<OL STYLE="LIST-STYLE-TYPE: LOWER-ALPHA">
<LI>List items
<LI>preceded by
<LI>lower-case letters
</OL>
<OL STYLE="LIST-STYLE-TYPE: UPPER-ALPHA">
<LI>List items
<LI>preceded by
<LI>upper-case letters
</OL>
<UL STYLE="LIST-STYLE-TYPE: DECIMAL">
<LI>List items
<LI>preceded by
<LI>numbers
</OL>
</TD>
<TD VALIGN="TOP">
<UL STYLE="LIST-STYLE-TYPE: NONE">
<LI>List items
<LI>preceded by
<LI>...nothing!
<LI>(but still indented <BR>like a regular list)
</UL>

```



### Listing 13-3 List-Style-Type (continued)

```
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```



**Figure 13-3** All of the LIST-STYLE-TYPE property values displayed within Internet Explorer.

You can actually achieve these same effects within HTML, by using the TYPE attribute with either the unordered or ordered list tags. For example, the following two sets of code are equivalent:

```
<H1>Lists Elements Altered Using CSS</H1>
<UL STYLE="LIST-STYLE-TYPE: SQUARE">
  <LI>List items
  <LI>preceded by
  <LI>solid squares
</UL>
<OL STYLE="LIST-STYLE-TYPE: UPPER-ROMAN">
```

```

<LI>List items
<LI>preceded by
<LI>upper-case <BR>roman numerals
</OL>

<H1>Lists Elements Altered Using HTML</H1>
<UL TYPE="SQUARE">
<LI>List items
<LI>preceded by
<LI>solid squares
</UL>
<OL TYPE="I">
<LI>List items
<LI>preceded by
<LI>upper-case <BR>roman numerals
</OL>

```

For all of the unordered list markers, you would use the same names in CSS as you would in HTML (i.e., DISC, CIRCLE and SQUARE). For the ordered list markers in HTML, however, you use the first “number” to appear in the form you want it to take; so, as in the preceding code example, to begin a list consisting of upper-case roman numerals, you use the first roman numeral (“I”) to appear. Similarly, for lower-case roman you’d specify “i”, for upper-case alpha you’d use “A”, and for lower-case alpha you’d use “a”.

The LIST-STYLE-TYPE property is supported equally well in both Netscape Navigator and Internet Explorer, so it is safe to use.

## The LIST-STYLE-IMAGE Property

Along with the LIST-STYLE-TYPE, LIST-STYLE-POSITION and LIST-STYLE, LIST-STYLE-IMAGE is designed to alter the way the list marker at the beginning of a list element appears. The idea behind LIST-STYLE-IMAGE is that sometimes you may want something other than a standard, plain-old list marker appear in front of each element in a list. If you want to put a colorful 3D button, a picture of the Mona Lisa or any other image in front of a list item, LIST-STYLE-TYPE is the property for you.

## LIST-STYLE-IMAGE

Description: Specifies an image to be used as the marker before a list element.

CSS Family Type: Classification

Value:

- **NONE** - No list-marker is displayed before a list item.
- **URL(image)**- Specifies the URL of the graphic to be used as a list marker.

Sample code:

```
<UL>
<LI STYLE="LIST-STYLE-IMAGE:
url(trajan.jpg)">Emperor Trajan, 98-117 AD
<LI STYLE="LIST-STYLE-IMAGE:
url(hadrian.jpg)">Emperor Hadrian, 117-158 AD
</UL>
```

**Table 13-4 LIST-STYLE-IMAGE Support in Major Browsers**

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i>	<i>IE 4.5</i>	<i>(Win. &amp;</i>	<i>NN 4.x</i>	<i>(Mac &amp;</i>	<i>Opera</i>
		<i>(Mac)</i>	<i>(Mac)</i>	<i>UNIX)</i>		<i>UNIX)</i>	<i>3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Safe

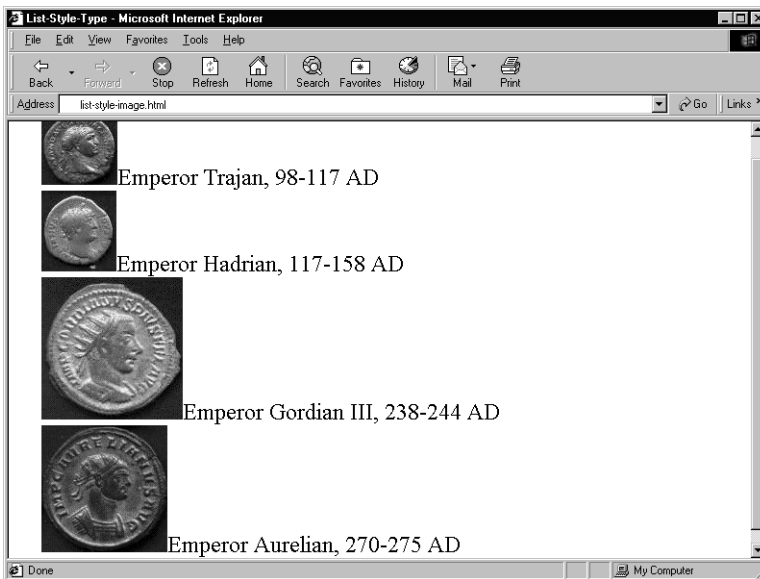
LIST-STYLE-TYPE has two possible values: NONE and a URL that links to the desired image. NONE is the default value, and, not surprisingly, it doesn't add a picture before a list element (it appears as a regular list item). By adding an URL value that links to an image, that image can become the "list marker" at the beginning of the list. The URL must be specified in the CSS manner (see the URL section in the "CSS Units" chapter for more detailed information on specifying URLs in CSS). The following sample code adds a different image before each list item.

**Listing 13-4 List-Style-Type**

```

<HTML>
<HEAD>
<TITLE>List-Style-Type</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE;">
<UL>
<LI STYLE="LIST-STYLE-IMAGE: URL(trajan.jpg);">Emperor
Trajan, 98-117 AD
<LI STYLE="LIST-STYLE-IMAGE: URL(hadrian.jpg);">Emperor
Hadrian, 117-158 AD
<LI STYLE="LIST-STYLE-IMAGE: URL(gordian3.jpg);">Emperor
Gordian III, 238-244 AD
<LI STYLE="LIST-STYLE-IMAGE: URL(aurelian.jpg);">Emperor
Aurelian, 270-275 AD
</UL>
</BODY>
</HTML>

```



**Figure 13-4** Sample LIST-STYLE-IMAGE code displayed within Internet Explorer.

This code example is extreme, as the images are too large to be practical, but it gets the idea across. One other thing that this demonstrates is that the images are displayed at their natural size. This is important to know if you intend to mix images using the LIST-STYLE-IMAGE property, as such CSS sizing properties as HEIGHT and WIDTH do *not* work in conjunction with the URL value.

This property is fully supported in Internet Explorer, but not at all in Netscape Navigator.

## The LIST-STYLE-POSITION Property

The LIST-STYLE-POSITION property, like all of the other sub-family of list style properties, is designed to alter the way the list marker at the beginning of a list element appears. This property does not cover where a list or the list element markers appear on screen as its name might lead you to believe. Instead, it positions the multiple lines of text after the list marker.

### ***LIST-STYLE-POSITION***

Description: Specifies how a list-marker is rendered relative to the content of the list item itself.

CSS Family Type: Classification

Values:

- **INSIDE** - Displays the text of a list item at a similar indentation level to the list-marker.
- **OUTSIDE** - Displays the text of a list item indented from the list-marker.

Sample code:

```
<UL STYLE="LIST-STYLE-POSITION: INSIDE">  
<LI>Notice how the text in the second line of this  
bullet item begins  
underneath the bullet rather than directly under  
the first word.  
</UL>
```

**Table 13-5 LIST-STYLE-POSITION Support in Major Browsers**

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Safe

The LIST-STYLE-POSITION property has two values: INSIDE and OUTSIDE. The effects of both of these values can only be seen when there are multiple lines of text appearing after a list element. OUTSIDE is the default value, and it essentially sets a margin value flush with the beginning of first word that appears in the first line after the list marker. All subsequent lines of text that appear after it are flush against that value. INSIDE removes this margin, and subsequent lines of text appear *under* the list marker. In addition, the INSIDE value moves the margin of the list item in slightly.

The effects are slightly different when used with the unordered list (“<UL>”) tag than with the ordered list (“<OL>”) tag. When the INSIDE value is applied to the unordered list tag, the next line of text appears directly underneath the list element marker. When applied to the ordered list tag, it appears under the period following the number — it does not appear flush underneath the number.

You can see both values used with the unordered and ordered list tags in the following code, which is displayed in Figure 13-5:

**Listing 13-5 List-Style-Position**

```
<HTML>
<HEAD>
<TITLE>List-Style-Position</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE;">
<UL STYLE="LIST-STYLE-POSITION: OUTSIDE">
<LI>The text for this bullet item will be displayed
normally, the second line of text beginning in a position
directly underneath the first word.
</UL>
```

**Listing 13-5 List-Style-Position (continued)**

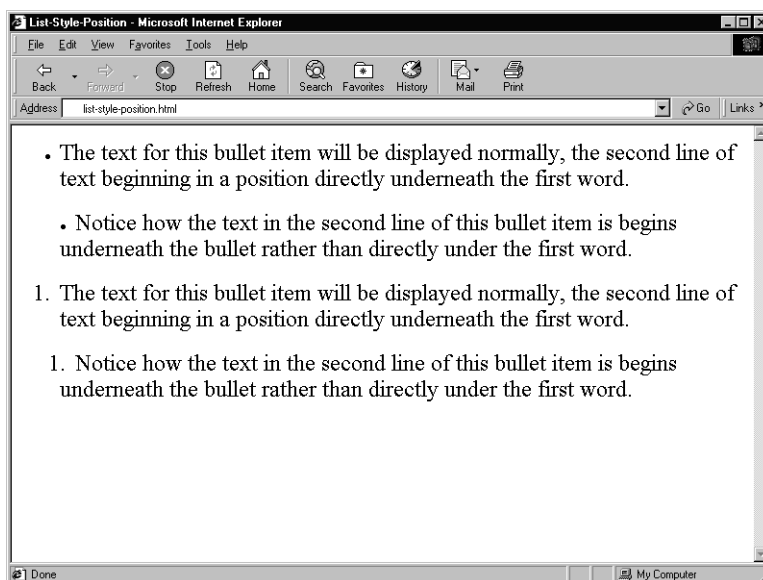
```
<UL STYLE="LIST-STYLE-POSITION: INSIDE">
<LI>Notice how the text in the second line of this bullet
item is begins
underneath the bullet rather than directly under the first
word.
</UL>

<OL STYLE="LIST-STYLE-POSITION: OUTSIDE">
<LI>The text for this bullet item will be displayed
normally, the
second line of text beginning in a position directly
underneath the first word.
</OL>
<OL STYLE="LIST-STYLE-POSITION: INSIDE">
<LI>Notice how the text in the second line of this bullet
item is begins
underneath the period that appears after the number rather
than directly under the first word.
</OL>
</BODY>
</HTML>
```

---

This particular CSS property is really a typographic nicety — it is an effect sometimes used in desktop publishing, primarily to save space. On the whole, though, the consensus is that the alternate `INSIDE` display value is not as “clean” as the default `OUTSIDE` value. Just because a property exists doesn’t mean you have to use it — but it is a nice option if you are tight on space.

The `LIST-STYLE-POSITION` property is fully supported within Internet Explorer, but not at all within Netscape Navigator. You should be able to get away with using the `INNER` value as a special effect for Internet Explorer users, but only when the needed effect is not critical (in other words, don’t design a page counting on the effect to appear to all users).



**Figure 13-5** Sample LIST-STYLE-POSITION code displayed within Internet Explorer.

## The LIST-STYLE Property

The LIST-STYLE property is a “shortcut” property that incorporates all of the features contained within its sibling properties LIST-STYLE-TYPE, LIST-STYLE-POSITION and LIST-STYLE-IMAGE. It is a convenient way to either set list display values that belong to its sibling properties, or to mix and match some — but not all — of them.

### **LIST-STYLE**

**Description:** This is an abbreviated property that enables the Web author to quickly set list item markers in the manner possible under LIST-STYLE-POSITION, LIST-STYLE-IMAGE and LIST-STYLE-TYPE.

**CSS Family Type:** Classification



Values:

- CIRCLE | DISC | SQUARE - Sets the symbol specified to appear before each list item.
- DECIMAL | LOWER-ALPHA | LOWER-ROMAN | UPPER-ALPHA | UPPER-ROMAN - Sets the type of alphanumeric symbol to appear before each list item.
- NONE - No list-marker is to be displayed before a list item.
- *URL(image)*- Specifies the URL of the graphic to be used as a list marker.
- INSIDE - Displays the text of a list item at a similar indentation level to the list-marker.
- OUTSIDE - Displays the text of a list item indented from the list-marker.

Sample code:

```
<UL STYLE="LIST-STYLE: SQUARE">
<LI>List items
<LI>preceded by
<LI>solid squares
</UL>
```

Table 13-6 LIST-STYLE Support in Major Browsers

				IE 5.0		NN 4.0	
IE 3.02	IE 4.x	IE 4.01 (Mac)	IE 4.5 (Mac)	(Win. & UNIX)	NN 4.x	(Mac & UNIX)	Opera 3.6
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Safe

LIST-STYLE can take the six values that belong to the LIST-STYLE-TYPE property: DECIMAL, LOWER-ALPHA, UPPER-ALPHA, LOWER-ROMAN and UPPER-ROMAN. DECIMAL displays a regular number, which is the default value for all ordered list elements. LOWER-ALPHA and UPPER-ALPHA display lower- and upper-case letters in alphabetical order for each list element. Similarly, LOWER-ROMAN and UPPER-ROMAN display lower- and upper-case roman numerals in numerical order for each list element. The value NONE (shared with LIST-STYLE-IMAGE) is the default value, and it doesn't do anything (in other words, the list appears the same as a regular, non-CSS altered list).

LIST-STYLE can take on the URL value that belongs to LIST-STYLE-TYPE. By adding an URL value that links to an image, that image can become

the “list marker” at the beginning of the list. The URL must be specified in the CSS manner (see the URL section in the “CSS Units” chapter for more detailed information on specifying URLs in CSS.)

LIST-STYLE can also take the two values that belong to LIST-STYLE-POSITION: INSIDE and OUTSIDE. OUTSIDE is the default value, and it essentially sets a margin value flush with the beginning of first word that appears in the first line after the list marker. All subsequent lines of text that appear after it are flush against that value. INSIDE removes this margin, and subsequent lines of text appear *under* the list marker.

You can see a mix of all of these values displayed in the following code (displayed in Figure 13–6).

### Listing 13-6 List-Style

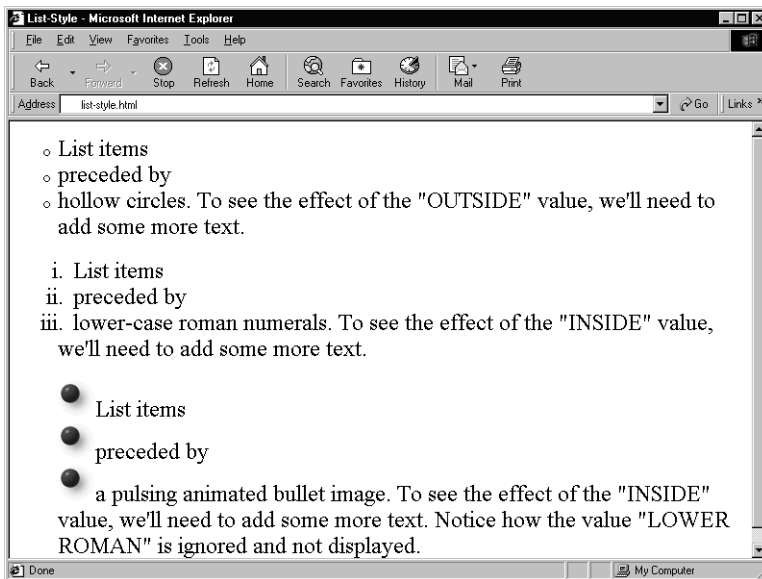
```
<HTML>
<HEAD>
<TITLE>List-Style</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: LARGE;">

<UL STYLE="LIST-STYLE: CIRCLE OUTSIDE;">
<LI>List items
<LI>preceded by
<LI>hollow circles
</UL>

<OL STYLE="LIST-STYLE: LOWER-ROMAN INSIDE;">
<LI>List items
<LI>preceded by
<LI>lower-case roman numerals. To see the effect of the
"INSIDE" value, we'll need to add some more text.
</OL>
```

**Listing 13-6 List-Style (continued)**

```
<UL STYLE="LIST-STYLE: URL(pulsing_bullet.gif) LOWER-roman
INSIDE;">
<LI>List items
<LI>preceded by
<LI>a pulsing animated bullet image. To see the effect of
the "INSIDE" value, we'll need to add some more text.
Notice how the value "LOWER ROMAN" is ignored and not
displayed.
</UL>
</BODY>
</HTML>
```



**Figure 13-6** Sample LIST-STYLE code displayed within Internet Explorer.

You will have noticed from Figure 13-6 that not all of the values can exist with each other. It is impossible to mix the values belonging to the `LIST-STYLE-TYPE` property when an image has already been set for the list marker using the `URL` value associated with the `LIST-STYLE-IMAGE` property. Specifically, notice how, in the third group of list items, the value

“URL(pulsing\_bullet.gif)” overrules the “LOWER-ROMAN” value. Neither of these sets of values conflict with the values associated with the `LIST-STYLE: INSIDE` or `OUTSIDE`. You can see from the first two groups of lists where the `INSIDE` and `OUTSIDE` values are mixed with `CIRCLE` and `LOWER-ROMAN`.

Internet Explorer fully supports the `LIST-STYLE` property. Netscape Navigator supports only the `LIST-STYLE-TYPE` values associated with the `LIST-STYLE` property.



# *Part* 3



**CSS2**

<i>Chapter 14</i>	<i>Overview, <b>286</b></i>
<i>Chapter 15</i>	<i>Selectors, Pseudo-Elements and Pseudo-Classes, <b>292</b></i>
<i>Chapter 16</i>	<i>New Media Types, <b>300</b></i>
<i>Chapter 17</i>	<i>The Box Family of Properties, <b>306</b></i>
<i>Chapter 18</i>	<i>Visual Formatting Family of Properties, <b>318</b></i>
<i>Chapter 19</i>	<i>Detailed Visual Formatting Family of Properties, <b>340</b></i>
<i>Chapter 20</i>	<i>Visual Effects Properties, <b>348</b></i>
<i>Chapter 21</i>	<i>Generated Content, Automatic Numbering and Lists, <b>360</b></i>
<i>Chapter 22</i>	<i>Paged Media Family of Properties, <b>378</b></i>
<i>Chapter 23</i>	<i>Font Family Properties, <b>394</b></i>
<i>Chapter 24</i>	<i>Text Family Properties, <b>406</b></i>
<i>Chapter 25</i>	<i>Table Family Properties, <b>412</b></i>
<i>Chapter 26</i>	<i>User Interface Properties, <b>428</b></i>
<i>Chapter 27</i>	<i>Aural Style Sheet Properties, <b>440</b></i>

# OVERVIEW

## Topics in this Chapter

- Overview of CSS2





# *Chapter* 14

This part provides an overview of the properties comprising the Cascading Style Sheet 2 (CSS2) specification. CSS2 is not designed to succeed the CSS1 specification, but rather to extend it with new functionality, and to better define many of the existing properties covered in the previous CSS1 specification. While the adoption of CSS2 in current browsers is minimal at best, if and when it is fully adopted by the major browsers, it will open up whole new realms of possibility for Web masters everywhere — particularly with regard to multimedia and the presentation of Web pages on devices other than a standard computer monitor. Among other things, CSS2 includes support for the following:

- definitions that support new media display types, such as print
- “aural” style sheets
- special features designed to better support Web pages across different languages
- much more precise control over font selection, including better font-matching mechanisms, “creating” fonts and support for downloadable fonts
- properties that will make table displays more versatile
- more control over the positioning of elements on a Web page
- an extended box family of properties

- greater ability to control the visibility of content, including controlling overflowing content and clipping sections of it from display
- the ability to set minimum and maximum widths and heights for on-screen elements
- automated functions that help generate content for such things as counters and automatic numbering
- greater control over user-interface elements, such as the display of cursors
- the ability for Web authors to specify exactly where elements should be positioned onscreen

The CSS2 specification actually incorporates many of the properties already covered in the CSS1 specification. In order not to repeat material already described in the CSS1 part, this part will describe only those CSS2 properties that are wholly new, or have greatly expanded functionality under CSS2.

This part looks in detail at the media types and properties of the following new families of functions under CSS2:

- Selectors, Pseudo-Elements and Pseudo-Classes
- New Media Types
- Box Family of Properties
- Visual Formatting Family of Properties
- Detailed Visual Formatting Family of Properties
- Visual Effects Properties
- Generated Content, Automatic Numbering and Lists
- Paged Media Family of Properties
- Font Family Properties
- Text Family Properties
- Table Family Properties
- User Interface Properties
- Aural Style Sheet Properties

It is worth noting here that there is a Color and Background family of properties under CSS2, but it incorporates properties fully covered in the CSS1 part of this book.

Probably the single most innovative concept introduced in the CSS2 specification is the introduction of aural style sheets — in other words, information that tells the Web browser how a page should *sound* when read. Aural style sheet information embedded in the code on a Web page is designed to

tell a browser how the text on a Web page should be interpreted and read by speaking devices. In large part, aural design is there to aid the visually impaired, but it has other, far-reaching uses; it will likely be important when you can tap into the Web while on the go. Aural style sheets foreshadow the day when a computer reads Web pages aloud while you are driving in your car, or in situations where your hands are otherwise occupied.

While a Web page may look great on your computer monitor, it doesn't always look so marvelous when the same Web page is printed. The introduction of the "paged media" concept in CSS2 tries to overcome aspects of the Web that do not lend themselves well to other media types, such as printed text. The paged media components of CSS2 allow Web authors to determine where such things as page breaks should occur when a page is printed, and where they want to set divisions both on the Web (on screen) and off (printed). This also means that you can set such things as headers and footers to the printed page, whether a page should be displayed in landscape or portrait styles, even where such things as crop marks should be displayed. CSS2 extends this control even further, to the extent that you determine, for example, how a left-facing page should be printed as opposed to a right-facing page, setting different margin values for each side. In addition to making Web pages more "print friendly", these new properties enable Web authors to better render Web pages for other specific media devices, including Braille readers, WebTV devices, small handheld computers and more.

Font characteristics are also expanded in CSS2. Using CSS1, you can set the size, type and color of a font. CSS2 allows Web authors to specify more precisely the type of font to be displayed, by providing information a browser can use in order to find a closer match to the fonts present on the user's system. CSS2 goes further still by providing browsers with downloadable font types if a font specified on a Web page does not exist on a user's computer, so Web authors can specify the exact font to be displayed. This new mechanism also supports Unicode, making it much easier for Web authors to add characters from non-European languages.


These and other such CSS2 features will provide Web authors with many wonderful new ways to shape and craft their Web pages. At the time of this writing, however, none of these features can be found in the popular Web browsers. It may take a while before CSS2 catches on in a big way, since many of these changes rely on the browser manufacturers to add significant new features to their products. In short, CSS2 provides Web authors with many new and useful tools, but we will all have to wait a while before we can use them.

Only a handful of CSS2 properties are currently supported by either Internet Explorer or Netscape Navigator. Many of the functions in the new CSS2 properties are available for use in the Mozilla browser, however, which is a “precursor” browser for Netscape Navigator 5.0. While there is no guarantee that these functions will be incorporated into the final build of Netscape Navigator 5.0, as decisions could be made to limit CSS2 support in an effort to reduce the file size or improve the speed of the browser, there is a fairly good chance that the examples provided here that are displayed in Mozilla will work as well or better in Netscape Navigator 5.0 when it is available.



# SELECTORS, PSEUDO- ELEMENTS AND PSEUDO-CLASSES

## Topics in this Chapter

- 
- Universal Selector
  - Child Selectors
  - Adjacent Sibling Selectors
  - Attribute Selectors
  - The BEFORE and AFTER Pseudo-Elements
  - The LANG Pseudo-Class
  - The LEFT, RIGHT and FIRST Pseudo-Classes
  - The FIRST-CHILD Pseudo-Class
  - The FOCUS Pseudo-Class
  - The HOVER Pseudo-Classes

# *Chapter* 15

CSS2 comes with a number of new selectors. These selectors allow Web authors much greater flexibility over how to associate CSS properties to Web elements. Only those wholly new selector types and those relevant to standard HTML are covered in this chapter. As of yet, none of these new selectors are recognized in any of the major browsers.

## Universal Selector

The universal selector is specified with the standard Windows/Unix “wild-card” character, which is an asterisk (“\*”). It allows the Web author to select all cases of a certain selector in a Web document. When an “\*” is added in front of a given selector, it formats all of the instances for that selector (without the “\*”) throughout the document, so that `*[LANG="EN"]` is equivalent to `[LANG="EN"]` wherever it appears in a Web page.

## Child Selectors

A child selector is applied whenever an element is considered to be the “child” of some “parent” element. It uses a “less-than” symbol (“>”) between at least two Web elements.

List elements can be considered the “children” of the “parent” ordered or unordered list tags. In a similar way, all displayed content on a Web page can be considered to be the child of the body tag. The child selector enables the Web author to specify that when a given child element appears after a certain parent element, it should be formatted in a way described. The following code sample provides two examples of this:

```
OL > LI {FONT-SIZE: 12PT}  
UL > LI {FONT-SIZE: 10PT}
```

In this case, all of the list items that are displayed in an ordered list will be rendered in a 12 point font, and those in an unordered list will be rendered in a 10 point font.

## Adjacent Sibling Selectors

Adjacent sibling selectors allow Web authors to decide how one Web element should appear if it appears immediately after another specified element that shares the same parent. In such a case, you could define how different header tags should interact with each other (because their immediate parent is the body tag), but not, say, a header and a cell in a table, which fall under different parent elements (the header to the body tag; the cell to the table tag).

Adjacent sibling selectors use the plus symbol “+” between two or more elements, and specify that if a certain condition occurs between these two elements, formatting should occur according to the rule stated by the selector. In the following case, whenever a second-level header immediately follows a top-level header, the second-level header will be indented by 1 inch:

```
H1 + H2 {TEXT-INDENT: 1IN}
```

If there are any second-level headers on the Web page that do not appear immediately after a top-level header, they will not be indented.



# Attribute Selectors

There are four attribute selectors introduced under CSS2, which are designed to match the defined attributes in the Web document. Here are quick descriptions for their use and purpose:

- `[attribute]` - Matches the name of the attribute contained within square brackets.
- `[attribute=value]` - A match is made when the attribute equals the value of “value”.
- `[attribute~=value]` - A match is made when the attribute roughly matches the value of “value”, in cases where the text “value” may be part of a larger word.
- `[attribute|=value]` - A match is made whenever the attribute matches the first few letters of a value whose first few letters match the text “value”.

These selectors allow Web masters greater control over selecting attributes with similar names.

In the following case, the attribute selector matches all second-level headers that use the “header” attribute, whatever its value:

```
H2[header] {COLOR: RED;}
```

This selector is more useful in cases where you want to combine selected attributes. For example, in the following case, all paragraphs that exactly match the “begin” attribute of “Goldilocks” and the “end” attribute of “ThreeBears” will be colored yellow:

```
P[begin="Goldilocks"][end="ThreeBears"]  
{COLOR: YELLOW;}
```

The “[~=]” value is useful in those cases where you want to match the attributes of a number of selectors that all begin with the same few letters in the chosen selector.

```
A[REL~= "prev"]
```

This will match all anchor tags that have the selector “pre” as part of its value, so it would match those that have the values “preview” and “previous”. It would also match a value like “prehensile”. For a more specific match, you’d want to use the “[|=]” attribute selector instead. The following example is more specific than the previous one:

```
A[REL|="prev"]
```

In this case, it would match all anchors that have values like “preview” and “previous”, but it would *not* match a value like “prehensile”.

## Pseudo-Elements and Pseudo-Classes

CSS2 also introduces a number of new pseudo-elements and pseudo-classes. Pseudo-elements are abstractions that enable functions not specified in the document language. For example, there is nothing in HTML that enables a Web author to specifically format the first letter or first line in a paragraph, but the CSS1 pseudo-elements `FIRST-LETTER` and `FIRST-LINE` make this possible. Similarly, pseudo-classes classify elements in ways other than their name, attributes or content, which lie outside typical document structure. This section looks briefly at the new CSS2 pseudo-elements and pseudo-classes.

The `BEFORE` and `AFTER` pseudo-elements insert a string and/or otherwise modify the display at the beginning and end, respectively, of the selected Web element. They are always used in conjunction with the `CONTENT` property. The following is a code snippet that demonstrates how `BEFORE` and `AFTER` are typically used:

```
<HTML>
<TITLE>Quotes</TITLE>
<STYLE>
Q:LANG(EN) {QUOTES: ' ' ' ' ' ' ' ' ' '}
Q:BEFORE {CONTENT: OPEN-QUOTE}
Q:AFTER {CONTENT: CLOSE-QUOTE}
</STYLE>
<BODY>
In the lecture the professor said <Q>Shakespeare
was the one who said the immortal words <Q>to
thine own self be true</Q></Q>
</BODY>
</HTML>
```

In this case the `BEFORE` and `AFTER` pseudo-classes set where the open and close quotes should appear on the Web page. For more information on how these two pseudo-elements can be used in context, see Chapter 18, “Visual Formatting Family of Properties”.

This sample code also shows the `LANG` pseudo-class in action, which determines how specific elements should appear given a specific language context. In this case “`LANG(EN)`” is specified for the type of quote marks to be displayed — in this case for the English language — but different languages require different types of quote marks. For example, it would be possible to set the type of quote marks to be displayed for English, French and German languages respectively:

```
Q:LANG(EN) {QUOTES: ' ' ' ' }
Q:LANG(FR) {QUOTES: '«' '»' }
Q:LANG(DE) {QUOTES: '»' '«' }
```

The `LEFT`, `RIGHT` and `FIRST` pseudo-classes are all used with the `@PAGE` media type. They are used to stipulate specific styles for the left, right and initial page of a Web document that is formatted for print. Here is an example of these pseudo-classes in action:

```
@PAGE :FIRST {MARGIN-LEFT: 1.5IN;
              MARGIN-RIGHT: 1.5IN;}
@PAGE :LEFT {MARGIN-LEFT: 1.5IN;
             MARGIN-RIGHT: 1IN;}
@PAGE :RIGHT {MARGIN-LEFT: 1IN;
              MARGIN-RIGHT: 1.5IN;}
```

In this case, the margin formats are set for how the initial page should be printed, and then all subsequent left-hand and right-hand pages. For more information on using these pseudo-classes, see Chapter 22, “Paged Media Family of Properties”.

The `FIRST-CHILD` pseudo-class matches a Web element that is the first child (i.e., it immediately follows the parent property) in a Web document. It should be used in conjunction with the child selector, and it selects only the initial child that appears after the parent element. The following sample code helps to explain how it should be used:

### Listing 15-1 First-Child Example

```
<HTML>
<HEAD>
<TITLE>First-Child Example</TITLE>
<STYLE>
DIV > P:FIRST-CHILD {COLOR: NAVY;}
</STYLE>
</HEAD>
```

**Listing 15-1 First-Child Example (continued)**

```
<BODY>
<DIV>
<P>
First sample sentence.
</P>
</DIV>
<DIV>
<H2>Note:</H2>
<P>
Second sample sentence. </P>
</DIV>
</BODY>
</HTML>
```

In this case, the first sentence would be colored blue, because it matches the conditions stated — it is contained within a paragraph immediately following a division tag. The second sentence is not colored blue, due to the intervening second-level header, which invalidates the conditions laid out in the selector in the header of the page.

The `FOCUS` and `HOVER` pseudo-classes are meant to convey information to the user whenever a cursor passes over a selected Web element, and are always associated with the anchor tag (“`<A>`”). With the `FOCUS` pseudo-class, the cursor will change to indicate that the Web element it is associated with has “focus” (meaning that it is available to take user input in some form). The `HOVER` pseudo-class indicates to the user that the element the cursor has passed over can be “activated” in some way. These new pseudo-classes are closely associated with the `LINK`, `VISITED` and `ACTIVE` pseudo-elements introduced under CSS1. The following code provides an example as to how these can be used:

```
A:LINK {COLOR: RED} /* Unvisited links */
A:VISITED {COLOR: BLUE} /* Visited links */
A:FOCUS {BACKGROUND: YELLOW} /* indicates a link */
A:HOVER {COLOR: YELLOW} /* User Hovers over link */
A:ACTIVE {COLOR: LIME} /* Active links */
```



# NEW MEDIA TYPES

## Topics in this Chapter

- 
- The ALL Media Type
  - The AURAL Media Type
  - The BRAILLE Media Type
  - The EMBOSSED Media Type
  - The HANDHELD Media Type
  - The PRINT Media Type
  - The PROJECTION Media Type
  - The SCREEN Media Type
  - The TTY Media Type
  - The TV Media Type

# *Chapter* 16

Media types introduce the concept that devices other than standard computer screens may be expected, now or in the future, to display Web content. Under the CSS2 specification, there are ten CSS2 media types: ALL, AURAL, BRAILLE, EMBOSSED, HANDHELD, PRINT, PROJECTION, SCREEN, TTY and TV. Each has particular characteristics, and is designed to provide Web authors with the option of specifying how a Web page should appear on very different display devices.

These properties are to be used with the new “at rule” @MEDIA, which is always specified in the header of a Web document. It selects the type of display device upon which a page can be displayed. Each of these different media types is defined in terms of media groupings. A continuous medium means that the medium is not broken up into discrete chunks, like the pages of a book; a paged medium, on the other hand, is. A visual medium is one that relies upon the viewer to see the page displayed, whereas an aural medium is one that speaks the page aloud, and a tactile medium means the Web page is intended to be felt (usually through a “display” mechanism that can be read by the visually impaired). A medium that relies upon grids can handle character grids, whereas a bitmap medium represents all displays (text and images) as bitmaps. Some media are interactive, which enable the user to work with and affect the flow of information, while others are static, meaning that the flow of information is fixed. In some cases, media types are capable of handling seemingly contradictory media groups, but this is wholly dependant on

the type of media groups that may be used with a specific type of display device.

The ALL media type is generally suitable for all display devices. The AURAL media type is intended for use with “talking browsers” (a computer equipped with a speech synthesizer, for example), and is associated with the continuous, aural, and interactive/static media groups. With the BRAILLE media type, Web content is intended for use on a Braille tactile interactive feedback device. It is associated with the continuous, tactile, grid and interactive/static media groups. EMBOSSSED is similar to BRAILLE, except the Web content is to be printed in Braille instead of being read through a mechanical Braille device. The HANDHELD media group is designed for people who use PalmPilots and similar PDA (personal data assistant) devices, which typically have relatively small screens — often with monochrome displays — and are likely limited in the bandwidth available to them. Its associated media groups include continuous/paged, visual, grid/bitmap and interactive/static.

The PRINT media group is designed to print Web content, or to be viewed on screen in a print-preview form. Its associated media groups are paged, visual, grid/bitmap and static. PROJECTION is a media group designed for projecting Web content on a remote screen, either directly or from printed transparencies. It is associated with the paged, visual, bitmap and static media groups. The SCREEN media type is designed for standard color computer displays, and is associated with the continuous, visual, grid and interactive/static media groups. The TTY media type is meant for displaying Web pages on such devices as teletype displays or terminals that use a fixed-pitch character grid. It is associated with the continuous, visual, grid and interactive/static media groups.

Finally, there is the TV media type, which allows for Web content to be displayed on a television screen, which implies relatively low screen resolution (compared to a computer screen), color display, sound and limited scrolling capabilities. It is associated with the continuous/paged, visual/aural, bitmap and interactive/static media groups. The following table provides a simple breakdown of all of the media groups to which the various media types belong.



**Table 16-1 Media Types and their Associated Media Groups**

<i>Media Types</i>	<i>Media Groups</i>								
	CONTINUOUS	PAGED	VISUAL	AURAL	TACTILE	GRID	BITMAP	INTERACTIVE	STATIC
AURAL	✓			✓				✓	✓
BRAILLE	✓				✓	✓		✓	✓
EMBOSS		✓			✓	✓		✓	✓
HANDHELD	✓	✓	✓			✓	✓	✓	✓
PRINT		✓	✓				✓		✓
PROJECTION		✓	✓				✓		✓
SCREEN	✓		✓				✓	✓	✓
TTY	✓		✓			✓		✓	✓
TV	✓	✓	✓	✓			✓	✓	✓

These media types enable Web authors to specify how things should “appear” when seen through various display devices. As a result, they are always set to apply globally to the Web document as a whole, as there is no point in trying to only specify how particular Web elements should appear.

The following code provides an example of how these media type values can be used:

**Listing 16-1 Media Types Example**

```
<HTML>
<HEAD>
<TITLE>Media Types Example</TITLE>
<STYLE>
@MEDIA SCREEN {BODY {FONT-SIZE: MEDIUM}}
@MEDIA PRINT {BODY {FONT-SIZE: SMALL}}
@MEDIA HANDHELD {BODY {FONT-SIZE: X-SMALL}}
@MEDIA TV {BODY {FONT-SIZE: LARGE}}
```

**Listing 16-1 Media Types Example (continued)**

```
</STYLE>
</HEAD>
<BODY>
The quick brown fox jumped over the lazy dog.
</BODY>
</HTML>
```

---

As you can see from this example, the size of the displayed font will change depending on the device being used to view the Web page. It is set to **MEDIUM** for a computer screen, **SMALL** for print (to minimize the number of pages to be printed), **X-SMALL** for handheld devices (to cram more text onto the screen), and **LARGE** for a TV screen, which is likely to be viewed from a distance.

As of yet, none of these media types are recognized by the major browsers.



# THE BOX FAMILY OF PROPERTIES

## Topics in this Chapter

- 
- The BORDER-TOP-COLOR Property
  - The BORDER-RIGHT-COLOR Property
  - The BORDER-LEFT-COLOR Property
  - The BORDER-BOTTOM-COLOR Property
  - The BORDER-TOP-STYLE Property
  - The BORDER-RIGHT-STYLE Property
  - The BORDER-LEFT-STYLE Property
  - The BORDER-BOTTOM-STYLE Property

# Chapter 17

CSS2 introduces several changes to the box family of properties. For example, several properties that formerly belonged to this family of properties under CSS1, such as `HEIGHT`, `FLOAT`, `CLEAR`, `HEIGHT`, and `WIDTH`, now belong to the visual formatting or visual formatting details categories of properties. In the new CSS specification, the existence of top, bottom, left and right borders are finally recognized in a number of new, wholly border properties. They are:

- `BORDER-TOP-COLOR`
- `BORDER-RIGHT-COLOR`
- `BORDER-LEFT-COLOR`
- `BORDER-BOTTOM-COLOR`
- `BORDER-TOP-STYLE`
- `BORDER-RIGHT-STYLE`
- `BORDER-LEFT-STYLE`
- `BORDER-BOTTOM-STYLE`

These new properties give Web authors greater control over the style and color of specific border sides.

## The Border “Side” Color Properties

There is a sub-family of new border values that could be called the “border side color” set of properties, as they set the color value for a particular side of a box within which an element is contained. This sub-family of properties includes `BORDER-TOP-COLOR`, `BORDER-RIGHT-COLOR`, `BORDER-LEFT-COLOR` and `BORDER-BOTTOM-COLOR`. These properties are designed to provide finer control over the display of border color properties than was available in the CSS1 specification.

### ***BORDER-TOP-COLOR***

Description: Specifies the color for the top border.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *color name* or *numerical color value* - Sets the color for the border.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<H1 STYLE="BORDER-TOP-COLOR: MAROON">Top-Level  
Header with a Maroon Colored Top Border</H1>
```

### ***BORDER-RIGHT-COLOR***

Description: Specifies the color for the right border.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *color name* or *numerical color value* - Sets the color for the border.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<H2 STYLE="BORDER-RIGHT-COLOR: TEAL">Top-Level  
Header with a Teal Colored Bottom Border</H1>
```

## ***BORDER-LEFT-COLOR***

Description: Specifies the color for the left border.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *color name* or *numerical color value* - Sets the color for the border.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<H2 STYLE="BORDER-LEFT-COLOR: SILVER">Second-Level  
Header with a Silver Colored Left Border</H2>
```

## ***BORDER-BOTTOM-COLOR***

Description: Specifies the color for the bottom border.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *color name* or *numerical color value* - Sets the color for the border.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

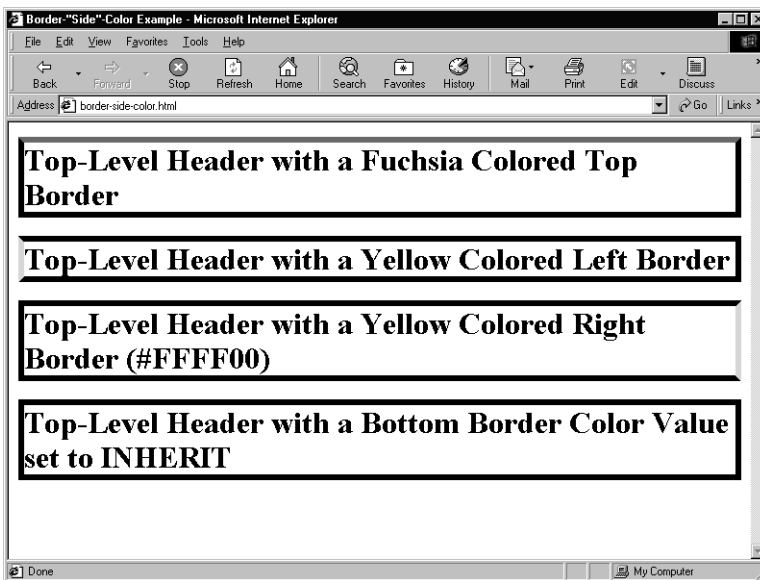
Sample code:

```
<H3 STYLE="BORDER-BOTTOM-COLOR: LIME">Third-Level  
Header with a Lime Colored Bottom Border</H3>
```

These properties can take either a standard CSS color name (such as “BLUE”) or a color value (such as the hexadecimal value “#00FF00”, equivalent to “BLUE”). When a color is specified, only the border specified will be set to that color. The following code examples show how it should be used, with the results displayed in Figure 17–1:

### Listing 17-1 Border-"Side"-Color Example

```
<HTML>
<HEAD>
<TITLE>Border-"Side"-Color Example</TITLE>
</HEAD>
<BODY>
<H1 STYLE="BORDER-TOP-COLOR: FUCHSIA; BORDER-STYLE: SOLID;
BORDER-WIDTH: THICK;">Top-Level Header with a Fuchsia
Colored Top Border</H1>
<H1 STYLE="BORDER-LEFT-COLOR: YELLOW; BORDER-STYLE: SOLID;
BORDER-WIDTH: THICK;">Top-Level Header with a Yellow
Colored Left Border</H1>
<H1 STYLE="BORDER-RIGHT-COLOR: #FFFF00; BORDER-STYLE:
SOLID; BORDER-WIDTH: THICK;">Top-Level Header with a Yellow
Colored Right Border (#FFFF00)</H1>
<H1 STYLE="BORDER-BOTTOM-COLOR: INHERIT; BORDER-STYLE:
SOLID; BORDER-WIDTH: THICK;">Top-Level Header with a Bottom
Border Color Value set to INHERIT</H1>
</BODY>
</HTML>
```



**Figure 17-1** The effect of all of the Border-"Side"-Color properties displayed using different color value types.



Note that the INHERIT value displays a black border, since it is deriving its value from its “parent”, which in this case is the color used by the other three sides of the border.

Interestingly enough, these properties are all supported within Internet Explorer 5.0. Although it is not supported in the equivalent version of Netscape, it is within the Mozilla browser (a precursor of the next Netscape browser), which means that these properties will most likely be fully supported Netscape Navigator 5.0.

## The Border “Side” Style Properties

There is a second new sub-family of border properties under CSS2, best described as the set of “border ‘side’ styles”. This sub-family is comprised of the following properties: BORDER-TOP-STYLE, BORDER-RIGHT-STYLE, BORDER-LEFT-STYLE and BORDER-BOTTOM-STYLE. These properties function in essentially the same way as the BORDER-STYLE property in CSS1, but are more specific in that their values apply only to a particular border side.

### ***BORDER-TOP-STYLE***

Description: Sets the type of top border to be displayed.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.
- HIDDEN - Same effect as NONE. It is designed to suppress border-displays when used in tables.

- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<P STYLE="BORDER-TOP-STYLE: DOTTED">Paragraph with  
a dotted top border.</P>
```

## ***BORDER-RIGHT-STYLE***

Description: Sets the type of right border to be displayed.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.
- HIDDEN - Same effect as NONE. It is designed to suppress border-displays when used in tables.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<P STYLE="BORDER-RIGHT-STYLE: RIDGE">Paragraph  
with a ridged right border.</P>
```

## ***BORDER-BOTTOM-STYLE***

Description: Sets the type of bottom border to be displayed.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.

- **INSET** - A 3D inset line is displayed.
- **GROOVE** - A 3D grooved line is displayed.
- **NONE** - No border is displayed, no matter what **BORDER-WIDTH** value is present.
- **OUTSET** - A 3D outset line is displayed.
- **RIDGE** - A 3D ridged line is displayed.
- **SOLID** - A solid border line is displayed.
- **HIDDEN** - Same effect as **NONE**. It is designed to suppress border-displays when used in tables.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<P STYLE="BORDER-BOTTOM-STYLE: SOLID">Paragraph  
with a solid bottom border.</P>
```

These properties take on all of the values found in their CSS1 cousin, the **BORDER-STYLE** property. There are a number of 2D and 3D border styles available. The 2D borders are created by the values **DASHED**, **DOTTED**, **DOUBLE** and **SOLID**, while the 3D borders are created by the **INSET**, **OUTSET**, **GROOVE** and **RIDGED** values. The **DASHED** value displays a border made up for dashes; the **DOTTED** value displays a border made from small dots; the **DOUBLE** value creates a border made from a double-line; and **SOLID** displays a solid border style. **INSET** creates a border that appears recessed, while **OUTSET** creates a border that appears raised. **GROOVE** displays a 3D grooved border line, and **RIDGE** create a 3D ridged border line. The value **NONE** ensures that no border is displayed.

In addition to these values are two new ones: **HIDDEN** and the ubiquitous CSS2 value **INHERIT**. The **HIDDEN** value is meant to be used to suppress the display of any border when used in conjunction with a table display. It works in the same way as **NONE**, but it should work as a conditional statement, active only when used within a table. The **INHERIT** value means that the border style is derived from whatever parent value may already be present.

When you set a border style for a specific side, it only appears for that side. You can combine it with the **BORDER-STYLE** property to set a different

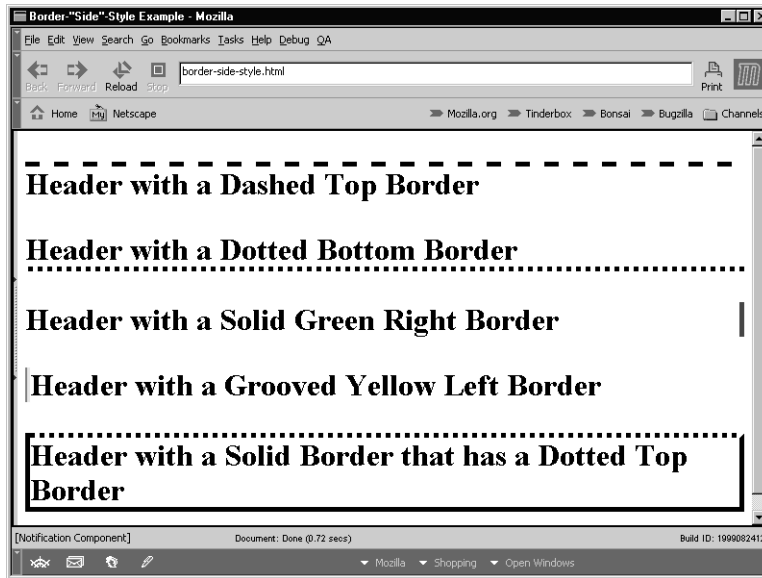
border display for a particular side. You can see several examples of the border “side” style properties in the following code example:

### Listing 17-2 Border-“Side”-Style Example

```
<HTML>
<HEAD>
<TITLE>Border-“Side”-Style Example</TITLE>
</HEAD>
<BODY>
<H1 STYLE="BORDER-TOP-STYLE: DASHED; BORDER-WIDTH:
THICK;">Header with a Dashed Top Border</H1>
<H1 STYLE="BORDER-BOTTOM-STYLE: DOTTED; BORDER-WIDTH:
THICK;">Header with a Dotted Bottom Border</H1>
<H1 STYLE="BORDER-RIGHT-STYLE: SOLID; BORDER-WIDTH: THICK;
BORDER-COLOR: GREEN;">Header with a Solid Green Right
Border</H1>
<H1 STYLE="BORDER-LEFT-STYLE: GROOVE; BORDER-WIDTH: THICK;
BORDER-COLOR: YELLOW;">Header with a Grooved Yellow Left
Border</H1>
<H1 STYLE="BORDER-STYLE: SOLID; BORDER-TOP-STYLE: DOTTED;
BORDER-WIDTH: THICK;">Header with a Solid Border that has a
Dotted Top Border</H1>
</BODY>
</HTML>
```

---

These properties are partially implemented in Internet Explorer 5.0, but not at all within Netscape Navigator 5.0. There is support from these properties under the Mozilla browser, however, which suggests that they will likely be fully supported in Netscape Navigator 5.0. Internet Explorer recognizes the “side” values for these properties, but it is unable to render certain border styles, dependant on the operating system (see Chapter 12 for more information).



**Figure 17-2** The Border “Side” Style Properties as Seen within the Mozilla Browser.

## Additional Values for the BORDER Sub-Family of Properties

Under the CSS2 specification, the BORDER properties and all of its sibling properties (BORDER-STYLE, BORDER-COLOR and BORDER-WIDTH) work in the same way as under the CSS1 specification: determining the style, color and width of the border displayed around a given Web element. CSS2 extends this property slightly by adding the named value INHERIT to all of these properties, the value HIDDEN to BORDER-STYLE and BORDER-WIDTH, and the value TRANSPARENT to BORDER-COLOR.

When the INHERIT value is used, these border properties take on the same value as that set by a parent element, if it exists (or is allowed under the circumstances).

The HIDDEN value, when applied to BORDER-STYLE and BORDER-WIDTH, is designed to suppress the display of any border when used in conjunction with a table display. It effectively has the same effect as NONE found under

CSS1, although it is meant to be a conditional statement, active only when used within a table.

The `TRANSPARENT` value, when applied to `BORDER-COLOR`, effectively renders the border invisible — it still has a width equal to what it would have if visible, but is simply not displayed. In this way, it can act almost like the `PADDING` property, possessing a spatial dimension, but otherwise no display characteristics.


These same values also apply to all of the “child” border properties, such as `BORDER-TOP`, `BORDER-LEFT`, `BORDER-RIGHT`, `BORDER-BOTTOM`, `BORDER-TOP-WIDTH`, `BORDER-LEFT-WIDTH`, `BORDER-RIGHT-WIDTH` and `BORDER-BOTTOM-WIDTH`.

None of these values are recognized as yet in the major browsers.



# VISUAL FORMATTING FAMILY OF PROPERTIES

## Topics in this Chapter

- 
- The DISPLAY Property
  - The POSITION Property
  - The TOP Property
  - The RIGHT Property
  - The BOTTOM Property
  - The CLEAR Property
  - The Z-INDEX Property
  - The DIRECTION Property
  - The UNICODE-BIDI Property



# *Chapter* 18

CSS2 introduces a new family of properties known as the visual formatting set of properties. This set of largely new or greatly expanded properties provide Web authors with much greater control over the display of elements on screen. The properties in this family include those that enable Web authors to exercise precise positioning of on-screen elements (also known as CSS positioning), to control the “level” in which a stack of 3D objects can appear on screen, and also to manage the direction in which text for other languages is displayed onscreen.

The visual formatting set of properties includes the following new or expanded properties:

- DISPLAY
- POSITION
- TOP
- RIGHT
- BOTTOM
- CLEAR
- Z-INDEX
- DIRECTION
- UNICODE-BIDI

The `FLOAT` and `CLEAR` properties have also been moved to this category from the Box Family in CSS1, but, on the whole, their roles have not changed much from those seen in CSS1.

## The `DISPLAY` Property

Under CSS1, the `DISPLAY` property falls under the set of classification properties. Under CSS2, it now falls under the set of visual formatting properties, but it retains its original purpose of being able to fundamentally alter or set the nature of a Web tag with which it is associated. Under CSS2, its role is expanded, and it has a number of new values.

### ***DISPLAY***

Description: Controls the fundamental nature of the specified HTML tag.

Media Group: All

CSS2 Family Type: Visual formatting

Value:

- `BLOCK` - Sets the specified element as a block-type element.
- `COMPACT` | `RUN-IN` - Creates either a block or inline box, depending on context.
- `INLINE` - Sets the specified element as an inline-type element.
- `LIST-ITEM` - Sets the specified element as a type of list-item.
- `MARKER` - This values sets the generated content before or after the box to be a marker. Should be used in conjunction with the `BEFORE` and `AFTER` pseudo-classes.
- `NONE` - Switches off the display of the specified element. It does not just create an invisible box; it creates no box at all.
- `TABLE` | `TABLE-CAPTION` | `TABLE-CELL` | `TABLE-COLUMN` | `TABLE-COLUMN-GROUP` | `TABLE-FOOTER-GROUP` | `TABLE-HEADER-GROUP` | `TABLE-ROW` | `TABLE-ROW-GROUP` | `INLINE-TABLE` - These values cause the selected element to behave like the specific table element specified.
- `INHERIT` - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<P STYLE="DISPLAY: NONE">This paragraph will not be  
displayed.</P>
```

Under CSS1, the `DISPLAY` property had the following four values: `BLOCK`, `INLINE`, `LIST-ITEM` and `NONE`. CSS2 adds another 14: `COMPACT`, `RUN-IN`, `MARKER`, `TABLE`, `TABLE-CAPTION`, `TABLE-CELL`, `TABLE-COLUMN`, `TABLE-COLUMN-GROUP`, `TABLE-FOOTER-GROUP`, `TABLE-HEADER-GROUP`, `TABLE-ROW`, `TABLE-ROW-GROUP`, `INLINE-TABLE` and `INHERIT`.

The `COMPACT` and `RUN-IN` values are designed to either set a block or inline box, depending on the context of the selected Web element. A block-level element set to `COMPACT` will, in effect, “compress” the size of the box under certain circumstances in an effort to conserve space. This is particularly handy with such things as lists, where you may not want the description of a list item to take up an extra line of space if it can “fit” in the same line as that of any content that might precede it. Similarly, `RUN-IN` will enable a block element to behave like an inline element, so that space can be saved. If the element so chosen doesn’t take the full width of the browser display, it is rendered as an inline element. The following code sample and illustration show how these two values are meant to work:

### Listing 18-1 Display Example #1

```
<HTML>  
<HEAD>  
<TITLE>Display Example #1</TITLE>  
<STYLE>  
DT {DISPLAY: COMPACT;}  
DD {MARGIN-LEFT: 2.5CM;}  
BODY {FONT-SIZE: X-LARGE;}  
H1 {DISPLAY: RUN-IN;}  
</STYLE>  
</HEAD>  
<BODY>  
<H1>Some parts of a car:</H1>  
<P>  
(A short list of parts and descriptions that go into making  
a car.)
```

### Listing 18-1 Display Example #1 (continued)

```
<DL>
<DT>Tires:
<DD><P>Made of rubber and is in contact with the ground.
<DT>Windshield:
<DD><P>Made of glass and keeps bugs and debris from hitting
the driver.
<DT>Seats:
<DD><P>Driver and passenger(s) sit in these.
</DL>
</BODY>
</HTML>
```

---

**Some parts of a car:** (A short list of parts and descriptions that go into making a car.)

Tires: Made of rubber and is in contact with the ground.

Windshield:

Made of glass and keeps bugs and debris from hitting the driver.

Seats: Driver and passenger(s) sit in these.

**Figure 18-1** An illustration of the intended effects of the COMPACT and RUN-IN values for the DISPLAY property.

As you can see in this example, the header at the top of the page is set to RUN-IN, so the text in the following paragraph is not displayed on the next line down as it usually would be, but is instead displayed immediately after the header, saving space on the page. The COMPACT value has been added to the <DT> (“Definition Term”) tag, and the <DD> (“Definition Description”)

tag has the value `MARGIN-LEFT: 2.5CM` attached to it. This means that if the definition term text is shorter than 2.5 centimeters in size, the definition description should follow immediately after the definition term, instead of on the next line down from it. You can see how the description follows the definition on the same line with the relatively short words “Tires:” and “Seats:”, and how they are compacted; “Windshield:” is too long, however, and cannot be compacted.

The desired effect of the `MARKER` value is in specifying how a generated marker should appear on screen, and setting the characteristics that distinguish it from the other content with which is associated. The `MARKER` value is meant to be used only with the `BEFORE` and `AFTER` pseudo-classes, and is designed to declare the selected content to be a marker in automatically generated content. `MARKER` content is formatted only on a single line, and is only created if the `CONTENT` property associated with the `BEFORE` or `AFTER` pseudo-classes actually generates content. Marker boxes can have padding and even borders, but cannot take margin values. The following example and illustration provide an idea as to how it should work:

### Listing 18-2 Display Example #2

```
<HTML>
<HEAD>
<TITLE>Display Example #2</TITLE>
<STYLE>
LI:BEFORE {DISPLAY: MARKER; CONTENT: "[" COUNTER(COUNTER)
          "]" ; COUNTER-INCREMENT: COUNTER; WIDTH: 2CM; TEXT-ALIGN:
          LEFT;}
BODY {FONT-SIZE: XX-LARGE;}
</STYLE>
</HEAD>
<BODY>
<OL>
<LI>Here is an item
<LI>Here is another item
<LI>Here is a third item
<LI>Here is the final, fourth item.
</OL>
</BODY>
</HTML>
```

---

- [1] Here is an item
- [2] Here is another item
- [3] Here is a third item
- [4] Here is the final, fourth item.

**Figure 18-2** Sample code using the MARKER value for the DISPLAY property.

As you can see from this example, the parameters set determine how the generated MARKER content should appear onscreen. Each of the list items is preceded by a number contained within square brackets, and each of these “markers” is left-aligned (most list markers are indented slightly); the text that follows them is spaced 2 centimeters from the marker.

Under CSS2, DISPLAY also carries a number of HTML table-tag related values: TABLE, TABLE-CAPTION, TABLE-CELL, TABLE-COLUMN, TABLE-COLUMN-GROUP, TABLE-FOOTER-GROUP, TABLE-HEADER-GROUP, TABLE-ROW, TABLE-ROW-GROUP and INLINE-TABLE. In each case, these values set the associated Web element to appear as an equivalent to the HTML table-tag value. Some of these values may be unfamiliar to you, as they correspond to rarely used but valid table tag values (such as <COL>, <COLGROUP>, <TFOOT> and <THEAD>, for example) under the official HTML 4.0 specification. There are one-to-one relationships between each of the HTML 4.0 table-tags and their DISPLAY property equivalents, as can be seen in Table 19-1.

These values are here are not supercede the existing HTML tags, but rather to be used by other document languages (such as the eXtended Markup Language, better known as “XML”) that do not have predefined table elements. This CSS2 scheme essentially allows XML applications (or

**Table 18-1 HTML 4.0 Table Tags and Their CSS2 DISPLAY Property Equivalents**

<CAPTION>	{DISPLAY: TABLE-CAPTION}
<COL>	{DISPLAY: TABLE-COLUMN}
<COLGROUP>	{DISPLAY: TABLE-COLUMN-GROUP}
<TABLE>	{DISPLAY: TABLE}
<TBODY>	{DISPLAY: TABLE-ROW-GROUP}
<TD>	{DISPLAY: TABLE-CELL}
<TFOOT>	{DISPLAY: TABLE-FOOTER-GROUP}
<TH>	{DISPLAY: TABLE-CELL}
<THEAD>	{DISPLAY: TABLE-HEADER-GROUP}
<TR>	{DISPLAY: TABLE-ROW}

similar languages) to incorporate tables by using what is effectively a “port” of CSS to whatever document language is being used.

Finally, there is the INHERIT named value for DISPLAY, which simply takes on whatever parent value has already been set for display.

For more information about the CSS1 values for DISPLAY, see Chapter 13.

## The Set of Positioning Properties

One of the longest-standing concerns in Web-page design is the positioning of elements on a Web page. How many times have you wanted to add some image or text to a *specific* point on a Web page, and found that this seemingly simple task tried your patience? Over time, people have attempted such tricks as transparent images, indented unordered list tags and maddeningly intricate sets of tables in order to get something exactly where they wanted it to be.

The set of positioning properties — consisting of POSITION, TOP, LEFT, RIGHT and BOTTOM — were among the earliest of the post-CSS1 properties to be suggested. They are part of a new sub-set of properties originally known as “CSS Positioning”, or simply CSS-P for short. Using these properties, you can now state exactly where you want a Web element to appear on screen. What

is more, these properties are among those few CSS2 properties that are at least partially implemented in Netscape Navigator 4.0 and Internet Explorer 5.0.

## POSITION

Description: Sets the positioning for a box-level element.

Media Group: Visual

CSS2 Family Type: Visual formatting

Values:

- **STATIC** - The box-level element positioning is normal, and is laid out to the normal flow.
- **RELATIVE** - The box-level element positioning is calculated with regard to the normal flow, and this element is offset relative to its normal position.
- **ABSOLUTE** - The box-level element is positioned using the **LEFT**, **RIGHT**, **TOP** and **BOTTOM** properties. Box-level elements positioned this way are separate from the normal flow.
- **FIXED** - The box-level element is positioned in the same manner as with **ABSOLUTE**, but is fixed with regard to some other element.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<STYLE>
@MEDIA SCREEN { {H1#first {POSITION: FIXED} }
@MEDIA PRINT { {H1#first {POSITION: STATIC} }
</STYLE>
```

## TOP

Description: Specifies the value that the current Web element is positioned below the containing box.

Media Group: Visual

CSS2 Family Type: Visual formatting

Values:

- ***n*** value - Sets a length unit value.
- **%** value - Sets a length percentage value relative to the width of the containing block.
- **AUTO** - The default value for the specific element.



- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="POSITION: ABSOLUTE; TOP: 1CM;">This
header is positioned 1 centimeter below the
containing box, which in this case is the bottom of
the browser window.</P>
```

## **RIGHT**

Description: Specifies the value that the current Web element is positioned to the right edge of its containing box.

Media Group: Visual

CSS2 Family Type: Visual formatting

Values:

- ***n*** value - Sets a length unit value.
- **%** value - Sets a length percentage value relative to the width of the containing block.
- **AUTO** - The default value for the specific element.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="POSITION: ABSOLUTE; RIGHT: 2CM;">This
header is positioned 2 centimeters from the
containing box, which in this case is the right
margin of the browser.</P>
```

## **BOTTOM**

Description: Specifies the value that the current Web element is positioned above the containing box.

Media Group: Visual

CSS2 Family Type: Visual formatting

Values:

- ***n*** value - Sets a length unit value.
- **%** value - Sets a length percentage value relative to the width of the containing block.
- **AUTO** - The default value for the specific element.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="POSITION: ABSOLUTE; BOTTOM: 1IN;">This  
header is positioned 1 inch from the containing  
box, which in this case is the bottom of the  
browser window.</P>
```

## LEFT

Description: Specifies the value that the current Web element is positioned to the left edge of its containing box.

Media Group: Visual

CSS2 Family Type: Visual formatting

Values:

- *n* value - Sets a length unit value.
- % value - Sets a length percentage value relative to the width of the containing block.
- AUTO - The default value for the specific element.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="POSITION: ABSOLUTE; LEFT: 20PX;">This  
header is positioned 20 pixels from the containing  
box, which in this case is the left margin of the  
browser.</P>
```

The `POSITION` property has five values: `ABSOLUTE`, `RELATIVE`, `STATIC`, `FIXED` and `INHERIT`. `POSITION` is meant to be associated with at least one (and usually more) of the other associated properties `LEFT`, `RIGHT`, `BOTTOM` and `TOP`, which take a unit of measurement value, a percentage value, the named value `AUTO` (which uses the default value normally used) or the ubiquitous `INHERIT` (which uses the parent value, if any, already set for any of these properties). Given the specific nature of the positioning element with reference to a computer screen, the default measurement value used is pixels.

The `ABSOLUTE` value for `POSITION` is the easiest to understand and is typically the most used. When you use the `ABSOLUTE` value, your element is positioned “absolutely” with reference to the top-left corner of the browser window. In other words, the top-left of your browser window represents the coordinates 0, 0, and you position your element with reference to that value. If you wanted to position an image to appear 25 pixels over and 100 down, you could write the following code:

```
<IMG SRC="fokker.gif" STYLE="POSITION: ABSOLUTE;  
LEFT: 250; TOP: 100;">
```

This means that the top-left corner of the image “fokker.gif” is positioned 250 pixels to the right and 100 pixels down from the top of the browser window. Absolutely positioned elements are removed from the normal flow of elements on screen, however, so any “normal” on-screen elements will flow underneath the absolutely positioned element. A good example of this can be seen in the illustration of the following code, as viewed within Internet Explorer 5.0:

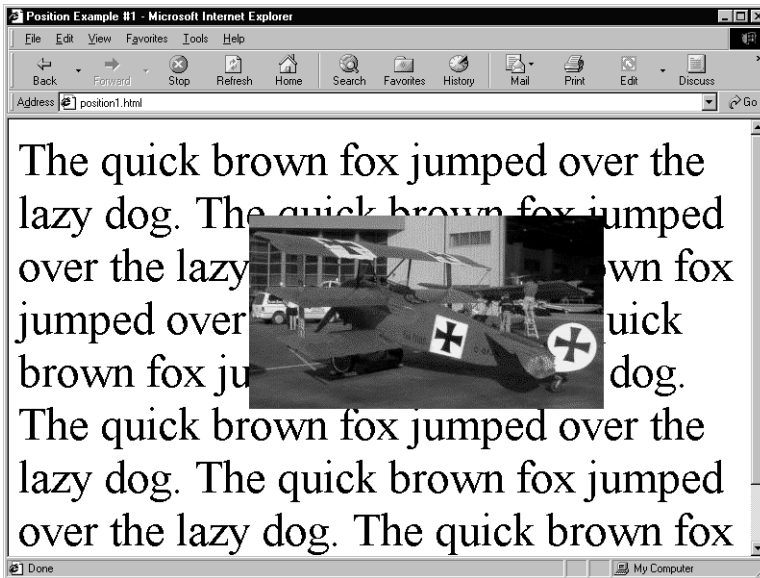
### Listing 18-3 Position Example #1

```
<HTML>  
<HEAD>  
<TITLE>Position Example #1</TITLE>  
</HEAD>  
<BODY style="FONT-SIZE: XX-LARGE;">  
<IMG SRC="fokker.gif" STYLE="POSITION: ABSOLUTE; LEFT: 250;  
TOP: 100;">  
The quick brown fox jumped over the lazy dog.  
The quick brown fox jumped over the lazy dog.  
The quick brown fox jumped over the lazy dog.  
The quick brown fox jumped over the lazy dog.  
The quick brown fox jumped over the lazy dog.  
The quick brown fox jumped over the lazy dog.  
The quick brown fox jumped over the lazy dog.  
</BODY>  
</HTML>
```

---

As you can see, the flow of the text on the page goes on as if the image of the old warplane is not there.

With the `RELATIVE` value, you can specify where you want an onscreen element to go *in relation to* other on-screen elements. Whereas an element positioned with `ABSOLUTE` is set apart from other elements on a page, `RELATIVE` is with respect to these other elements. Elements that are positioned using `RELATIVE` have their own box, and can influence subsequent elements. The following code shows `RELATIVE` being used to offset a

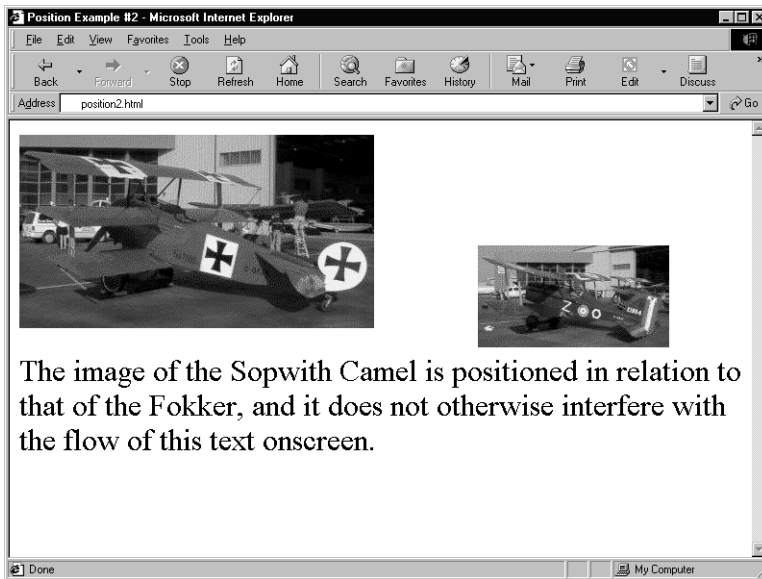


**Figure 18-3** An example of POSITION: ABSOLUTE as seen within Internet Explorer 5.0.

picture of a Sopwith Camel from that of the Fokker (which, in this case, does not use any special positioning values):

### Listing 18-4 Position Example #2

```
<HTML>
<HEAD>
<TITLE>Position Example #2</TITLE>
</HEAD>
<BODY style="FONT-SIZE: X-LARGE;">
<IMG SRC="fokker.gif">
<IMG SRC="sopwith.gif" STYLE="POSITION: RELATIVE; TOP: 20;
LEFT: 100;">
<P>
The image of the Sopwith Camel is positioned in relation to
that of the Fokker, and it does not otherwise interfere
with the flow of this text onscreen.
</BODY>
</HTML>
```



**Figure 18-4** Image of a Sopwith Camel airplane positioned relative to that of the Fokker airplane, as seen within Internet Explorer 5.0.

As you can see from this example, the image of the Sopwith Camel airplane is positioned 100 pixels to the left of, and 20 pixels down from, the Fokker airplane. Note that the 20 pixels down value is with respect to the image of the Fokker as a whole, rather than from the top of the Fokker image; also, without the `<P>` tag that appears between the second, relatively positioned image (or some other way of specifying a line break), the text that appears would, in fact, flow underneath the image as if it wasn't there.

The `STATIC` value is the default value for the `POSITION` property, and it essentially sets things to return to the normal flow on a page. Occasions for wanting to use this particular value would be rare. As an example, the following two lines of code would be displayed in an identical manner:

```
<I STYLE="POSITION: STATIC">Back to the normal
flow!</I>

<I>Back to the normal flow!</I>
```

The `FIXED` value behaves in much the same way as the `ABSOLUTE` value, but while the item will scroll with the page with the `ABSOLUTE` value, it will not with the `FIXED` value. In the following code example

(similar to that seen for the ABSOLUTE value), you can see how the text “behind” the FIXED image scrolls, but the image itself does not.

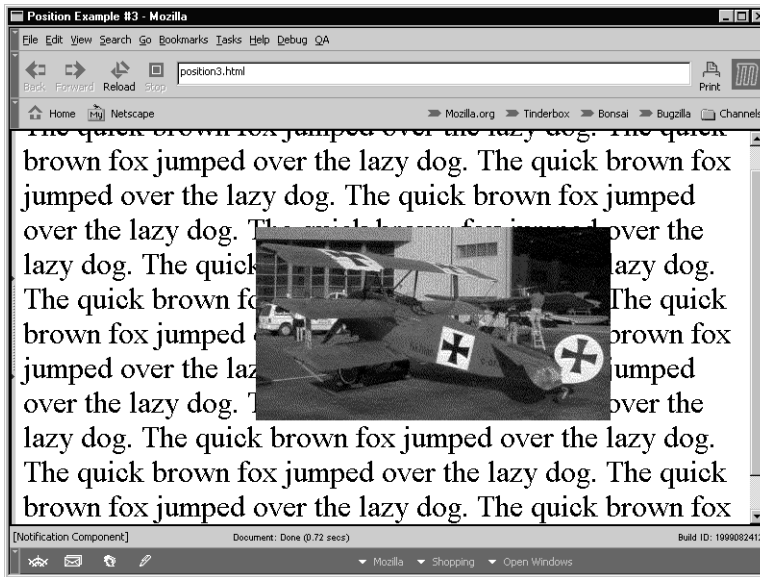
### Listing 18-5 Position Example #3

```
<HTML>
<HEAD>
<TITLE>Position Example #3</TITLE>
</HEAD>
<BODY style="FONT-SIZE: XX-LARGE;">
<IMG SRC="fokker.gif" STYLE="POSITION: FIXED; LEFT: 250;
TOP: 100;">
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
</BODY>
</HTML>
```

---

One note of caution when using the FIXED value — it is really designed for use on dynamic displays, and not for other media, such as print. With the introduction of media types under CSS2 (see Chapter 16, “New Media Types”), Web authors must be concerned with such things, so it is advisable to set things so that the FIXED value is only utilized in the “right” media. The following code snippet gives you an idea of how you can differentiate between different media while using the FIXED value effectively:

```
<STYLE>
@MEDIA SCREEN {H1#INITIAL {POSITION: FIXED}}
@MEDIA PRINT {H1#INITIAL {POSITION: STATIC}}
</STYLE>
```



**Figure 18-5** The FIXED value for the POSITION property displayed in the Mozilla browser.

Finally, the INHERIT value simply takes on whatever parent value may have already been set for POSITION.

These examples might imply that you always have to position items to the left and down from any previous element on the Web page. You can certainly use the LEFT and BOTTOM properties, but you can also use negative values for these as well as for RIGHT and TOP. Using negative values means that you can even position elements off the screen if you like. When you combine this feature with some JavaScript, it becomes possible to add dynamic CSS-based displays and to make text or images appear or disappear at will.

The POSITION property, along with LEFT, BOTTOM, RIGHT and TOP, have already been partially implemented within Internet Explorer 5.0. Internet Explorer 5.0 recognizes all but the FIXED value for POSITION, and all of the values for the other associated properties. These properties all appear to be supported in the Mozilla browser, which suggests that they will most likely be available in Netscape Navigator 5.0.

## The Z-INDEX Property

On a Web page, Web elements exist in only two dimension: the X (horizontal) and Y (vertical) axes. On a regular Web page, elements are either above or below, or to the right or left of, another element. With the advent of the `POSITION` property, a third dimension is added, so that things can appear either “in front of” or “behind” other elements. This third axis is known as the Z-axis, and elements can be controlled in this fashion by using the `Z-INDEX` property.

### Z-INDEX

Description: Sets the stacking level of the specified box in relation to the current stacking context. It also determines whether the specified box sets up a localized stacking context.

Media Group: Visual

CSS2 Family Type: Visual formatting

Values:

- ***n*** - An integer value representing the stacking level of the specified element's box. The specified element's box also sets up a local stacking value.
- **AUTO** - The stack value is the same as its parent's. The specified element's box does not set up a local stacking value.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

#### Listing 18-6 Z-Index Example

```
<HTML>
<HEAD>
<TITLE>Z-Index Example</TITLE>
<STYLE>
.stack {POSITION: ABSOLUTE; RIGHT: 1CM; BOTTOM: 0CM;}
</STYLE>
</HEAD>
```



**Listing 18-6 Z-Index Example (continued)**

```
<BODY STYLE="FONT-SIZE: XX-LARGE; COLOR: YELLOW;">
<P>
<IMG SRC="fokker.gif" CLASS="stack" STYLE="Z-INDEX: 1"
width=369 height=201 border=0 alt="">
<DIV ID="text1" CLASS="stack" STYLE="Z-INDEX: 3"> This text
overlays imagel.
</DIV>
<DIV ID="text2">This text has not been assigned a Z-INDEX
value, so it lies beneath everything else. </DIV>
<DIV ID="text3" CLASS="stack" STYLE="Z-INDEX: 2"> This text
will underlay text1, but overlay imagel.
</DIV>
</BODY>
</HTML>
```

Z-INDEX can take one of three values: an integer value that corresponds to the “stacking” level of the specified element’s box; AUTO, which makes stack value the same as its parent’s; and INHERIT, which in this case ought to work the same way as does AUTO. Z-INDEX is always used in conjunction with the POSITION property.

The integer value is the value that is most often used. It sets the “stacking” level, which determines where elements fall within the stack. Lower numbers are at the “bottom” of the stack; those with higher numbers appear “above” those below them. Any element that is not assigned a stack level falls below the lowest stacked item, and either does not appear (if the object on top is large enough) or flows underneath it. A good example of how this work can be seen in the sample code for Z-INDEX, as seen within the Mozilla browser in Figure 18-6.

As you can see from this example, the picture of the airplane lies at the “bottom” of the stack with a value of “1”. The line of text with the highest stack value (“3”) lies over top of the image. The line of text that has a stack level of “2” is also positioned on top of the image, but vertically below the line with the higher stack value. Note also how the line of text that has no assigned stack value flows underneath the image.

One might wonder from this example as to how Z-INDEX can be put to good use. While it can be used for adding captions over images (as is the case in the preceding example), it is possible to dynamically change the Z-INDEX stack value of an element using JavaScript, making dynamic displays possible.



**Figure 18-6** Z-INDEX sample code, as seen in Mozilla.

This value is not implemented in either Internet Explorer 5.0 or Netscape Navigator 4.x. It is implemented within the Mozilla browser (as can be seen from the illustration), however, so it will likely be available for use in Netscape Navigator 5.0.

## Text Direction Properties: DIRECTION and UNICODE-BIDI

There are many languages that read from right-to-left, such as Arabic and Hebrew, instead of the left-to-right more typical of European languages. When characters from both types of writing systems are present in a single document, it is known as bi-directional text: some text is to be read in one direction, and some in the other. The `DIRECTION` and `UNICODE-BIDI` properties are designed to help Web authors render this type of display on screen.

### ***DIRECTION***

Description: Sets the direction for the letters in the text to follow.

Media Group: Visual

CSS2 Family Type: Visual formatting

Values:

- LTR - Text is displayed from left to right.
- RTL - Text is displayed from right to left.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="DIRECTION: LTR;">This sentence is
displayed normally.</P>
<P STYLE="DIRECTION: RTL;">.yllamron deyalpsid si
ecnetnes siht</P>
```

The DIRECTION property has three values: LTR (“left-to-right”), RTL (“right-to-left”) and INHERIT. In essence, this property is used to flip around the ordering of the characters it contains. If you used the English proper name “Keith Schengili-Roberts” and used the RTL value on it, it would come out as: “streboR-ilignehcS htieK”. Although this may appear flashy-looking (or just silly) on a Web page, this is not the type of situation for which the property was designed. Here’s a better example — let’s say you had the following text on a Web page:

```
EnglishWord1 ArabicWord2 EnglishWord3
```

The bi-directional algorithm within most browsers already knows that it should display characters from a given character set in a particular direction, but it does not always know how to place such things as numerals correctly in conjunction with these characters. In this case, the text would actually be displayed in this way:

```
EnglishWord1 wroWcibarA2 EnglishWord3
```

The Arabic word is displayed from left-to-right, as it should be, but the number “2” now appears at the beginning of the word, so that it now reads “2ArabicWord” instead of “ArabicWord2”. This is where the bi-directional override tag should be used to explicitly set the direction of the Arabic text, as in the following example:

```
EnglishWord1 <SPAN STYLE="DIRECTION:
RTL">ArabicWord2</SPAN> EnglishWord3
```

This would display the text as it was intended, which would be like this:

```
EnglishWord1 2wroWcibarA EnglishWord3
```

The `DIRECTION` property is also designed for use with the `UNICODE-BIDI` property, helping to set the direction for the text display of other languages.

## UNICODE-BIDI

**Description:** Defines the Unicode text as having bi-directional characteristics, meaning that the text may be rendered from right to left and left to right in different circumstances.

**Media Group:** Visual

**CSS2 Family Type:** Visual formatting

**Values:**

- **NORMAL** - The selected Web element does not alter the current bi-directional setting.
- **EMBED** - If the selected Web element is inline, this value sets up an additional level of embedding with respect to the bi-directional algorithm.
- **BIDI-OVERRIDE** - If the selected Web element is inline or block-level, this value overrides that in place, keeping with the values set by the `DIRECTION` property.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

**Sample Code:**

```
<STYLE>
ARABIC, {DIRECTION: RTL; UNICODE-BIDI: EMBED}
ENGLISH {DIRECTION: LTR; UNICODE-BIDI: EMBED}
</STYLE>
```

Every browser has an implicit understanding of how text from various languages should appear on screen. This is handled by the `UNICODE` specification. The `UNICODE-BIDI` property enables Web authors to explicitly set the direction in which text and associated content should appear in Web pages that use languages meant to be read in different directions. Whereas `DIRECTION` can be used within a Web document, `UNICODE-BIDI` essentially has the same purpose, but applies instead to the whole of a Web document (or Web documents).

The `UNICODE-BIDI` property has four named properties: `NORMAL`, `EMBED`, `BIDI-OVERRIDE` and `INHERIT`. The `NORMAL` value does not alter the default bi-directional setting for the display of text. The `EMBED` value is meant for use with inline Web elements, in which it sets up an additional level of embedding with respect to the bi-directional algorithm. If the `BIDI-OVERRIDE` value is used, this value overrides the value already in place,

keeping with the values set by the DIRECTION property. It is meant for use in inline and block-level elements. Finally, the INHERIT value takes the same value for UNICODE-BIDI as that set for the property of the parent element. Of these values, EMBED is the one most likely to see use, as it is the most versatile.

The following code provides an example of how the UNICODE-BIDI property can be used:

### Listing 18-7 Unicode-Bidi and Direction Example

```
<HTML>
<HEAD>
<TITLE>Unicode-Bidi and Direction Example</TITLE>
<STYLE>
ARABIC {DIRECTION: RTL; UNICODE-BIDI: EMBED}
ENGLISH {DIRECTION: LTR; UNICODE-BIDI: EMBED}
HEBREW {DIRECTION: RTL; UNICODE-BIDI: EMBED}
</STYLE>
</HEAD>
<BODY>
<H1 CLASS="ENGLISH">English1 <SPAN
CLASS="ARABIC">Arabic1</SPAN> <SPAN
CLASS="HEBREW">Hebrew1</SPAN><H1>
<P CLASS="ENGLISH">English1 English2, English3.</P>
<P CLASS="ARABIC">Arabic1, Arabic2, Arabic3.</P>
<P CLASS="HEBREW">Hebrew1, Hebrew2, Hebrew3.</P>
</BODY>
</HTML>
```

---

The values for the direction of text will be applied to the text according to the rules and direction information laid out in the header of this Web page.

# DETAILED VISUAL FORMATTING FAMILY OF PROPERTIES

## Topics in this Chapter

- The MIN-WIDTH Property
- The MAX-WIDTH Property
- The MIN-HEIGHT Property
- The MAX-HEIGHT Property



# *Chapter* 19

In addition to the visual formatting properties covered in the previous chapter, there is another new category of properties designed to deal with the finer details of visual formatting. This collection of properties is there for Web authors to set such things as the minimum and maximum widths and heights of various displays, whether or not text that may overflow a given space is cut, and even whether or not a given Web element is visible.

The detailed visual formatting properties consist of the following:

- MIN-WIDTH
- MAX-WIDTH
- MIN-HEIGHT
- MAX-HEIGHT

## **The MIN-WIDTH and MAX-WIDTH Properties**

The MIN-WIDTH and MAX-WIDTH properties are designed to enable Web authors to set the smallest and largest possible width values for box elements. You now have control over the minimum and maximum width for any given

box element, and can better set how a particular Web element should appear on screen.

## MIN-WIDTH

Description: Sets the minimum width for a block element.

Media Group: Visual

CSS2 Family Type: Visual formatting (Details)

Values:

- *n* value - Sets a length unit value.
- % value - Sets a length percentage value relative to the width of the containing block.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="MIN-WIDTH: 2CM;">This Header Must Be No  
Smaller than 2CM in Width</H1>
```

## MAX-WIDTH

Description: Sets the maximum width for a block element.

Media Group: Visual

CSS2 Family Type: Visual formatting (Details)

Values:

- *n* value - Sets a length unit value.
- % value - Sets a length percentage value relative to the width of the containing block.
- NONE - Default value, sets no maximum value for the width of the box.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="MAX-WIDTH: 5CM;">This Header Must Be No  
Wider than 5CM</H1>
```

Both MIN-WIDTH and MAX-WIDTH take the same values: either a numeric unit of measurement value, a percentage value relative to the width of the containing block, NONE (which does not set a specific width value for the box), and INHERIT (which takes its value from that of the parent).



The MIN-WIDTH property is handy when you want to ensure that a Web element runs to at least a given minimum on the Web page; if, for example, you set the contents of a header to a size of 2 centimeters, no matter how much text is contained in the header, it will be at least 2 centimeters in width. Of course, it can be bigger than 2 centimeters. It is important to note that while you can set the minimum width of the box within which an element is contained, the elements contained within the box are not made to “fit” this space.

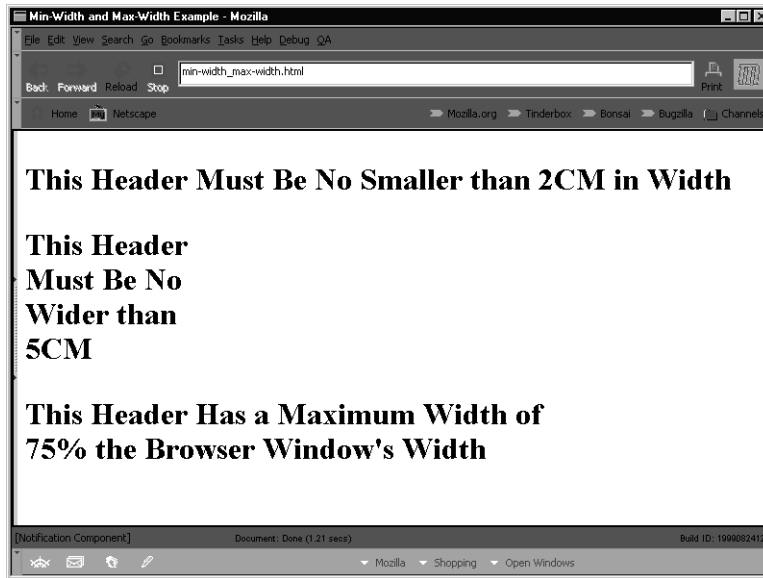
MAX-WIDTH is more likely to see common usage than MIN-WIDTH, as it constrains the box element to a certain maximum width on the Web page. The following code example shows two examples of MAX-WIDTH at work:

### Listing 19-1 Min-Width and Max-Width Example

```
<HTML>
<HEAD>
<TITLE>Min-Width and Max-Width Example</TITLE>
</HEAD>
<BODY>
<H1 STYLE="MIN-WIDTH: 2CM;">This Header Must Be No Smaller
than 2CM in Width</H1>
<H1 STYLE="MAX-WIDTH: 5CM;">This Header Must Be No Wider
than 5CM</H1>
<H1 STYLE="MAX-WIDTH: 75%;">This Header Has a Maximum Width
of 75% the Browser Window's Width</H1>
</BODY>
</HTML>
```

---

Neither the MIN-WIDTH nor MAX-WIDTH property is as yet supported in the currently released versions of Internet Explorer or Netscape Navigator. The illustration used in this section was derived from the Mozilla browser, however, which has functions that are expected to be incorporated into Netscape Navigator 5.0 when it is released.



**Figure 19-1** Effects of the MIN-WIDTH and MAX-WIDTH properties displayed within the Mozilla browser.

## The MIN-HEIGHT and MAX-HEIGHT Properties

The MIN-HEIGHT and MAX-HEIGHT properties work in much the same way as the MIN-WIDTH and MAX-WIDTH properties, but instead of setting the width (horizontal) value for a block element, they instead control the height (vertical) value. It is important to note that while you can set the height of the box within which an element is contained, the elements contained within the box are not stretched or condensed to fit.

### **MIN-HEIGHT**

Description: Sets the minimum height for a block element.

Media Group: Visual

CSS2 Family Type: Visual formatting (Details)

Values:

- *n* value - Sets a length unit value.
- % value - Sets a length percentage value relative to the height of the containing block.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="MIN-HEIGHT: 5CM;">This Header Must Be At  
Least 5CM in Height</H1>
```

## MAX-HEIGHT

Description: Sets the maximum height for a block element.

Media Group: Visual

CSS2 Family Type: Visual formatting (Details)

Values:

- *n* value - Sets a length unit value.
- % value - Sets a length percentage value relative to the height of the containing block.
- NONE - Default value, sets no maximum value for the height of the box.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="MAX-HEIGHT: 1CM;">This Header Must Be No  
Bigger than 1CM</H1>
```

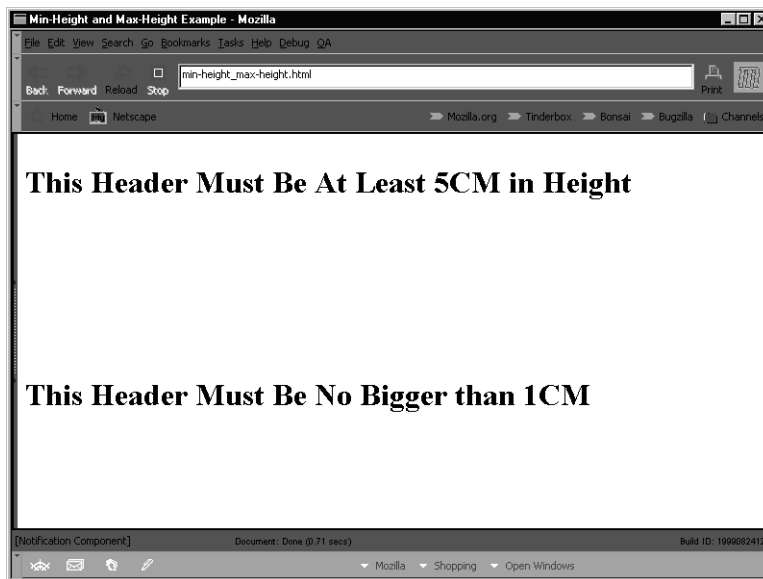
Both MIN-HEIGHT and MAX-HEIGHT take the same values: a numeric unit of measurement value, a percentage value relative to the width of the containing block, plus the two named values NONE (the default, which only applies to MAX-HEIGHT — the default value for MIN-HEIGHT is “0”) and INHERIT (which takes its value from that of the parent).

The MIN-HEIGHT property is handy when you want to ensure that a Web element runs to at least a given height on the Web page. If, for example, you set the contents of a header to a size of 5 centimeters in height, the box containing the header will be at least that size on the Web page.

MAX-HEIGHT instead constrains the box element to a certain maximum width on the Web page. The following code example shows both MIN-HEIGHT and MAX-WIDTH at work:

### Listing 19-2 Min-Height and Max-Height Example

```
<HTML>
<HEAD>
<TITLE>Min-Height and Max-Height Example</TITLE>
</HEAD>
<BODY>
<H1 STYLE="MIN-HEIGHT: 5CM;">This Header Must Be At Least
5CM in Height</H1>
<H1 STYLE="MAX-HEIGHT: 1CM;">This Header Must Be No Bigger
than 1CM</H1>
</BODY>
</HTML>
```



**Figure 19-2** Effects of the MIN-HEIGHT and MAX-HEIGHT properties displayed within the Mozilla browser.

Notice how, in the illustration (Figure 19–2), the top header is contained within a box that is contained within a box that is 5 centimeters in height. The text in the box is not “stretched to fit”; it is instead rendered normally, but the next Web element appears 5 centimeters underneath the box containing the top header. Note how the second header is set to only 1 centimeter in height, and as the header is no larger than this, the browser displays it properly. However, if the header contained more text and flowed onto another line, that subsequent text should appear “clipped”. (Unfortunately, this feature is not as yet supported in the Mozilla browser, so it is not possible to demonstrate this effect).

Neither the MIN-HEIGHT nor MAX-HEIGHT property is supported in the currently released versions of Internet Explorer or Netscape Navigator. The Mozilla browser does support these properties, however, so they are likely to be incorporated into Netscape Navigator 5.0 when it is released.

# VISUAL EFFECTS PROPERTIES

## Topics in this Chapter

- The OVERFLOW Property
- The CLIP Property
- The VISIBILITY Property



# Chapter 20

The set of visual effects properties is designed to enable Web authors to control how text or other Web elements are displayed when they exceed the dimensions of the box within which they are contained. With the introduction of absolute positioning and of fixed heights and width, it becomes necessary to tell the browser what to do on those occasions when content ends up exceeding its box. The following three new CSS2 properties are designed to aid Web authors in determining how material should be displayed under such circumstances:

- OVERFLOW
- CLIP
- VISIBILITY

## OVERFLOW

The `OVERFLOW` property tells the browser what to do whenever the content contained within the bounding box of a block-level element extends beyond the dimensions of the box. Using this property, you can tell the browser to simply not display the content that “overflows” the boundary of the box, dis-

play a scroll-bar so that the user can see the full content of the box if they desire, and more.

## OVERFLOW

Description: Determines how user can access content that would otherwise be hidden from view when it takes up more space than is allotted to it.

Media Group: Visual

CSS2 Family Type: Visual effects

Values:

- **VISIBLE** - The content is not clipped. It is displayed.
- **HIDDEN** - The content is clipped. No scrollbars are displayed to allow scrolling to view the rest of the content.
- **SCROLL** - The content is clipped, but a scrollbar or equivalent mechanism is made available to allow the viewer to scroll through the content.
- **AUTO** - A scrolling mechanism is automatically provided should it be needed.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<B>The author then said:</B>
<P>
<BLOCKQUOTE STYLE="WIDTH: 1.5IN; HEIGHT: 1IN;
OVERFLOW: AUTO">
If this quote runs too long, a scrollbar will
appear so that the viewer can continue to read the
somewhat lengthy and definitely meandering
quotation.
</BLOCKQUOTE>
<P STYLE="WIDTH: 1.5IN; HEIGHT: 1IN;
OVERFLOW: AUTO">
<IMG SRC="vanessa.jpg">
</P>
<P STYLE="WIDTH: 1.5IN; HEIGHT: 1IN;
OVERFLOW: HIDDEN">
<IMG SRC="vanessa.jpg">
</P>
```

The **OVERFLOW** property can take any one of five different named properties: **VISIBLE**, **HIDDEN**, **SCROLL**, **AUTO** and **INHERIT**. With the **VISIBLE** value, the content contained within the box is not clipped, but is instead



displayed, thereby forcing the box to fit the size of the content it contains. With the **HIDDEN** value, the content is simply clipped and there is no way for the viewer to see the remaining content. **SCROLL** and **AUTO** both display scrollbars for the user to see any content that would otherwise have been clipped; the difference between the two is that with **SCROLL** the scrollbars always appear, and with **AUTO** the scrollbars only appear if the situation requires. The **INHERIT** property simply takes on whatever parent value already exists for **OVERFLOW**.

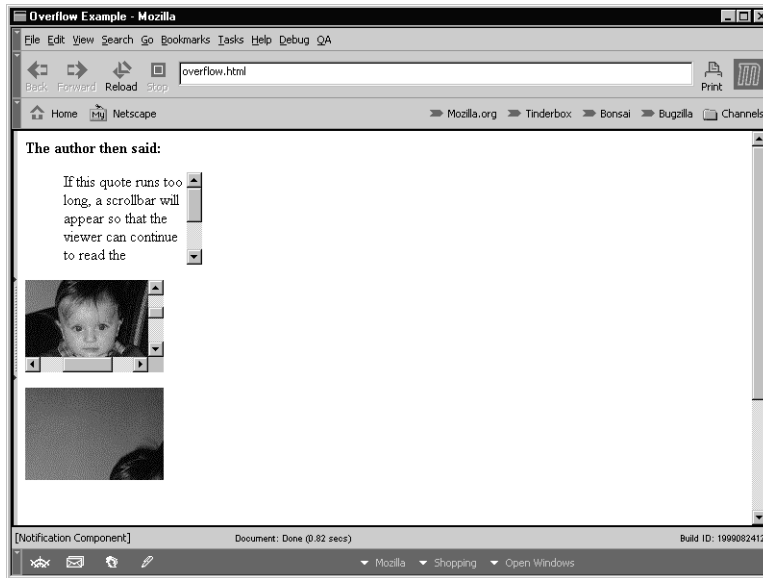
The value most likely to see use is **AUTO**, which automatically adds a scrollbar to the content so that people can view the content within the restraining box. It has the advantage over **SCROLL** of adding a scrollbar only if it is needed. The following code example shows the **OVERFLOW** property set to work:

### Listing 20-1 Overflow Example

```
<HTML>
<HEAD>
<TITLE>Overflow Example</TITLE>
</HEAD>
<BODY>
<B>The author then said:</B>
<P>
<BLOCKQUOTE STYLE="WIDTH: 1.5IN; HEIGHT: 1IN; OVERFLOW:
AUTO">
If this quote runs too long, a scrollbar will appear so
that the viewer can continue to read the somewhat lengthy
and definitely meandering quotation.
</BLOCKQUOTE>
<P STYLE="WIDTH: 1.5IN; HEIGHT: 1IN; OVERFLOW: AUTO">
<IMG SRC="vanessa.jpg">
</P>
<P STYLE="WIDTH: 1.5IN; HEIGHT: 1IN; OVERFLOW: HIDDEN">
<IMG SRC="vanessa.jpg"> </P>
</BODY>
</HTML>
```

---

Note in Figure 20–1 how a single scrollbar appears with the text example, but two scrollbars appear in the example associated with the picture. This is because the text is not constrained to a certain width, and it is only the overflow text at the bottom that needs to be scrolled. Since the image has a set



**Figure 20-1** Effects of the OVERFLOW property displayed in Mozilla.

width and height and is not clipped, scrollbars appear at the side and the bottom so the viewer can observe the full image.

There is full support for OVERFLOW in Internet Explorer 5.0. It is not supported at all in Netscape Navigator 4.0, but it is well supported in the Mozilla browser (as seen in Figure 20-1); since Mozilla's features will likely be integrated into Netscape Navigator 5.0, it is probable that OVERFLOW will be supported in that version of the browser.

## CLIP

By default, the bounding box of any given block-level element displays everything contained within it. Using the CLIP property, it is possible to define what portion of the content contained within the box is actually visible to the viewer.

## CLIP

Description: Defines how much of a Web element is made visible.

Media Group: Visual

CSS2 Family Type: Visual effects

Values:

- **RECT[*n n n n*]** - A rectangle defined by units of length in the following order: top, right, bottom and left. Negative values are permitted. Values are relative to the size of the Web element.
- **AUTO** - Sets the element to the regular, default visible value for that element. Can be used in conjunction with RECT.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1>The letters in this header are not "clipped"</H1>
<P>
<H1 STYLE="CLIP: RECT(10PX 5PX 10PX 5PX);">The
letters in this header will appear "clipped".</H1>
```

There are three named values that apply to CLIP: RECT, which defines a rectangle outlined by units of measurement; AUTO, which sets the clipping value to the default visible value; and INHERIT, which adopts its value from the parent. Of these three, it is really the RECT value that defines the CLIP property.

RECT takes a string of up to four separate measurement values that set the TOP, RIGHT, BOTTOM and LEFT clipping values, respectively. When a unit of measure value is set, it sets the clipping value “in” from the bounding box, so a value of 5PX clips content from the inside of the box by 5 pixels. You can see the clipping effect in the following code:

**Listing 20-2 Clip Example**

```
<HTML>
<HEAD>
<TITLE>Clip Example</TITLE>
</HEAD>
<BODY>
<H1>The letters in this header are not "clipped".</H1>
<P>
<H1 STYLE="CLIP: RECT(10PX 5PX 10PX 5PX);">The letters in
this header will appear "clipped".</H1>
</BODY>
</HTML>
```

---

**The letters in this header are  
not "clipped".**

**The letters in this header will  
appear "clipped".**

**Figure 20-2** Effects of the CLIP property as it should be displayed.

Notice how the text of the second header is clipped with respect to the text contained in the initial header. What you are effectively doing when you use the `CLIP` property is setting the size of the “viewport” contained within the bounding box of the block-level element, showing only what you intend to show. While the last example shows how `CLIP` can be used with text, it is

more likely to be used with such things as images, where you can effectively “crop” those sections of the image that you do not want rendered visible. Note that the measurement values in the code example are contained within round brackets, and that a space separates the individual values. It is also worth pointing out that the `CLIP` property can also take on negative values.

This property is not supported under either Netscape Navigator or Internet Explorer.

## VISIBILITY

The `VISIBILITY` property is there for Web authors to determine which Web elements should and should not be rendered visible on screen. It can be applied to all on-screen elements.

### *VISIBILITY*

Description: Sets whether or not the box associated with an element is rendered.

Media Group: Visual

CSS2 Family Type: Visual effects

Values:

- **VISIBLE** - The box generated by the element is displayed.
- **HIDDEN** - The box generated by the element is not displayed.
- **COLLAPSE** - When applied to a table element (i.e., a row, row group, column or column group), causes the entire selected row or column to not be displayed.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="VISIBILITY: HIDDEN;">Header in an  
Invisible Box</H1>
```

There are four named values for `VISIBILITY`: **VISIBLE**, the default value that displays the element onscreen; **HIDDEN**, which prevents the element from being displayed; **COLLAPSE**, which is to be used with tables and prevents a column or row from being displayed; and **INHERIT**, which simply takes on the parent value set for `VISIBILITY`. The **VISIBLE** and **HIDDEN**

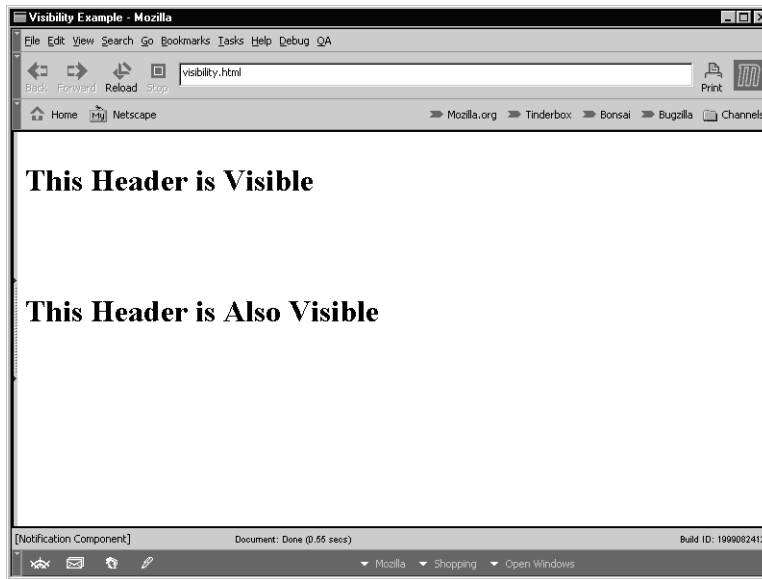
values are self-explanatory, and their effects can be seen in the following code example:

### Listing 20-3 Visibility Example

```
<HTML>
<HEAD>
<TITLE>Visibility Example</TITLE>
</HEAD>
<BODY>
<H1 STYLE="VISIBILITY: VISIBLE;">This Header is Visible</
H1>
<P>
<H1 STYLE="VISIBILITY: HIDDEN;">This Header is Invisible</
H1>
<P>
<H1 STYLE="VISIBILITY: VISIBLE;">This Header is Also
Visible</H1>
<P STYLE="FONT-SIZE: LARGE;">
This part of the sentence is visible, <B STYLE="VISIBILITY:
HIDDEN;">this is not visible</B>, and this is also visible.
</P>
</BODY>
</HTML>
```

---

You should note that when using the `VISIBILITY` property, the box surrounding the element (be it a in-line or block-level element) does not disappear, it simply “hides” the content from being displayed. The result is a blank space where the content would normally appear, as you can see in Figure 20–3.



**Figure 20-3** Effects of the VISIBILITY property as seen in the Mozilla browser.

The COLLAPSE value is meant to be used with table rows and columns, and is designed to have much the same effect as HIDDEN. Instead of leaving the space where an effectively invisible row or column would be displayed, however, it “collapses” the adjoining rows or columns so that it is as if the selected row or column did not exist. You can see an example of this in the following code, whose results are displayed in Figure 20-4:

### Listing 20-4 Visibility Example #2

```
<HTML>
<HEAD>
<TITLE>Visibility Example #2</TITLE>
</HEAD>
<BODY>
<TABLE>
<TR>
<TD VALIGN="TOP">
```

### Listing 20-4 Visibility Example #2 (continued)

```

<!-- left-hand table -->

<TABLE CELLPADDING="15">
<TR STYLE="FONT-SIZE: XX-LARGE;">
<TD>This will be visible</TD>
<TD>So will this</TD>
</TR>
<TR STYLE="VISIBILITY: COLLAPSE; FONT-SIZE: XX-LARGE;">
<TD>This should not be visible</TD>
<TD>Neither will this</TD>
</TR>
<TR STYLE="FONT-SIZE: XX-LARGE;">
<TD>This will be visible</TD>
<TD>So will this</TD>
</TR>
</TABLE>

</TD>
<TD VALIGN="TOP">
<!-- right-hand table -->

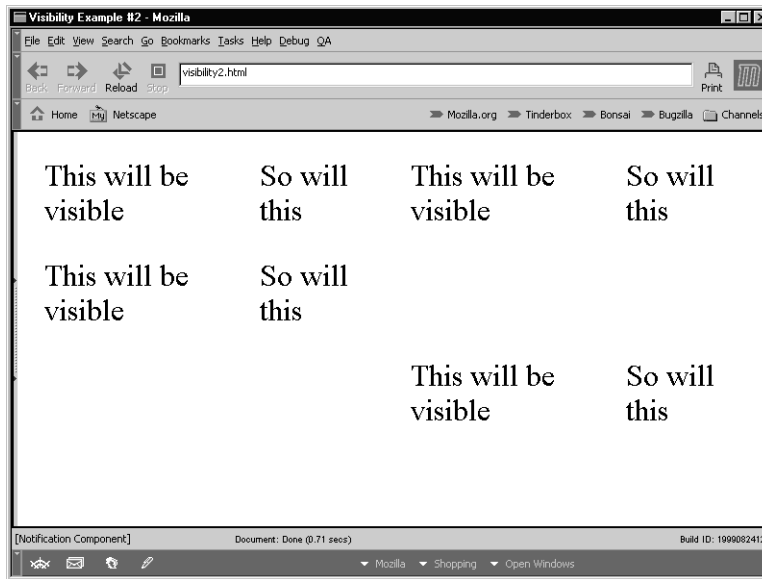
<TABLE CELLPADDING="15">
<TR STYLE="FONT-SIZE: XX-LARGE;">
<TD>This will be visible</TD>
<TD>So will this</TD>
</TR>
<TR STYLE="VISIBILITY: HIDDEN; FONT-SIZE: XX-LARGE;">
<TD>This should not be visible</TD>
<TD>Neither will this</TD>
</TR>
<TR STYLE="FONT-SIZE: XX-LARGE;">
<TD>This will be visible</TD>
<TD>So will this</TD>
</TR>
</TABLE>

</TD>
</TR>
</TABLE>
</BODY>
</HTML>

```

---





**Figure 20–4** The effects of the VISIBILITY property on two adjacent tables, one with COLLAPSE set to the middle row, and the second with HIDDEN to the middle row.

You can see the effects of the values COLLAPSE and HIDDEN as applied within two nested tables seen in Figure 20–4. In the first table, it is as if the row never existed (it has been “collapsed”), whereas in the second table, it is clear that there is an invisible row present in the table. (It is worth noting that while this displays properly when no table borders are added, in the build of the Mozilla browser being used for this illustration, adding table borders in fact “shows” where the otherwise invisible rows should appear, which is not according to the specification).

The VISIBILITY property is not supported in either Internet Explorer or Netscape Navigator. The screenshot used in Figure 20–4 is derived from the Mozilla browser, whose features are expected to be incorporated in Netscape Navigator 5.0, so you can presume VISIBILITY will appear in that version of the browser.

# GENERATED CONTENT, AUTOMATIC NUMBERING AND LISTS

## Topics in this Chapter

- The CONTENT Property
- The QUOTES Property
- The COUNTER-INCREMENT Property
- The COUNTER-RESET Property
- The MARKER-OFFSET Property

# Chapter 21

There is a new family of properties under CSS2 — generated content, automatic numbering and lists — that is responsible for generating content within a browser. The key word here is really “generated content”. What this means is that a browser is responsible for automatically generating content, such as the numbering added to list items, the correct type of quotation mark (depending on context), and much more.

This family brings together a number of new properties, greatly extends some others, and even comes with its own pseudo-elements.

These are the wholly new properties:

- CONTENT
- QUOTES
- COUNTER-INCREMENT
- COUNTER-RESET
- MARKER-OFFSET

The CSS2 specification also greatly extends the values that can be used with the following property introduced under CSS1:

- LIST-STYLE-TYPE (and, by extension, LIST-STYLE)

The following two pseudo-elements also fall under this family:

- BEFORE
- AFTER

## The CONTENT Property plus the BEFORE and AFTER Pseudo-Elements

There is little point in talking about the `CONTENT` property without also talking about the `BEFORE` and `AFTER` pseudo-elements, since their functions are so inextricably intertwined. The `CONTENT` property is used to set the type of content (i.e., an HTML tag) with which the `BEFORE` and `AFTER` pseudo-elements are associated. The `CONTENT` property is *always* used in conjunction with the `BEFORE` and `AFTER` pseudo-classes. This collection of CSS2 functions enables Web authors to set such things as audio cues to be spoken immediately before or after certain elements, or to add formatting to specific on-screen elements.

These functions are also used extensively by all of the other properties contained within the generated content, automatic numbering and lists family.

### CONTENT

**Description:** Determines the type of content and, subsequently, how it is to be displayed. Is always used in conjunction with the `BEFORE` and `AFTER` pseudo-classes.

**Media Group:** All

**CSS2 Family Type:** Generated content, automatic numbering, and lists

**Values:**

- **STRING** - Refers to regular textual content.
- **URL("filename")** - Points to a specific Web content (most typically an image or sound file) in a format supported by the browser.
- **COUNTER-INCREMENT | COUNTER-RESET** - These two counter properties can be used in conjunction with `CONTENT` to set a counter value for the associated element.
- **ATTR(X)** - Returns the string of the value of attribute X for the subject of the selector.
- **OPEN-QUOTE | CLOSE-QUOTE** - These two values retrieve and insert the string previously set by the `QUOTES` property.

- NO-OPEN-QUOTE | NO-CLOSE-QUOTE - These two values insert nothing, but do increment (or decrement) the nesting level for quotes.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

### Listing 21-1 Content Example

```
<HTML>
<HEAD>
<STYLE>
IMG:before {CONTENT: URL("This_Is_An_Image.wav")}
</STYLE>
</HEAD>
<BODY>
All of the images on this Web page are preceded by an audio
introduction, announcing them to be images, primarily for
the benefit of those using audio browsers.
<IMG SRC="image1.gif">
<IMG SRC="image2.gif">
</BODY>
</HTML>
```

---

## BEFORE

Inserts a string and/or otherwise modifies the display at the beginning of the selected Web element.

## AFTER

Inserts a string and/or otherwise modifies the display at the end of the selected Web element.

The CONTENT property is capable of taking a wide range of values, in addition to its common usage with the BEFORE and AFTER pseudo-classes. It can take a string value (i.e., text) that is set by the Web author and is to be used as an identifier. It can also take a URL value, which is to point to Web content such as an image or sound file. It can also take on the values COUNTER-INCREMENT and COUNTER-RESET, which set a counter value for the associated element (much like an ordered list).

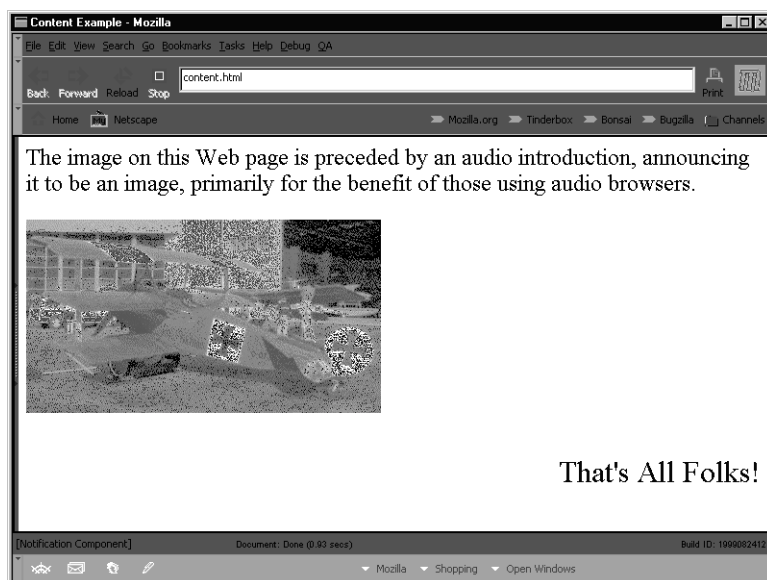
CONTENT can return a text string value by using the value ATTR along with the name of the string. It is also capable of being used with a number of quotation-mark related properties — OPEN-QUOTE, CLOSE-QUOTE, NO-OPEN-QUOTE and NO-CLOSE-QUOTE — which insert the correct type of quotation mark, depending on context. Since counter values and the quotation-mark values are closely tied to other properties in this family, they will be examined in detail later in this section. CONTENT can also take on the ubiquitous CSS2 value INHERIT, so that it will take on whatever parent value that had already been set for CONTENT.

- Since the BEFORE and AFTER pseudo-classes are meant to be used in a global context, CONTENT, BEFORE and AFTER are designed for use in the header of a Web page, or in a separate Web page containing CSS information meant to be applied to more than one Web page.
- With the following code sample, whose results are displayed in the Mozilla browser in Figure 21–1, we can see a good basic example of the BEFORE and AFTER pseudo-classes at work, along with examples of CONTENT taking a URL and a string value.

### Listing 21-2 Content Example #1

```
<HTML>
<HEAD>
<TITLE>Content Example #1</TITLE>
<STYLE>
  IMG:BEFORE {CONTENT: URL("This_Is_An_Image.wav")}
  BODY:AFTER {CONTENT: "That's All Folks!"; DISPLAY: BLOCK;
  MARGIN-TOP: 1CM; TEXT-ALIGN: RIGHT; COLOR: NAVY; FONT-SIZE:
  XX-LARGE;}
</STYLE>
</HEAD>
<BODY STYLE="FONT-SIZE: X-LARGE">
  The image on this Web page is preceded by an audio
  introduction, announcing it to be an image, primarily for
  the benefit of those using audio browsers.
  <P>
  <IMG SRC="fokker.jpg">
</BODY>
</HTML>
```

---



**Figure 21-1** CONTENT example as displayed within the Mozilla browser.

While the initial example can't be properly "displayed" within a book, the sound file ("This\_Is\_An\_Image.wav") would be played immediately prior to the image when the text on the Web browser is read aloud. In this way, an audio cue can be presented before a specific element to notify those who may be "viewing" the page using only an audio browser about the type of content appearing on the screen. The `AFTER` pseudo-element is associated with the `<BODY>` tag, so at the conclusion of the Web page, the phrase "That's all folks!" appears. In this latter case, you can see that the `CONTENT` tag is very flexible, and that it is possible to set a number of different display attributes for the text string.

This property is not supported in any of the major browsers available at this time, although it is worth pointing out that at least the non-audio segment seems to work well in the Mozilla browser. Since Mozilla's features are expected to appear in the final release version of Netscape Navigator 5.0, it is likely that this property will be supported in that version of the browser when it becomes available.

## The QUOTES Property

The `QUOTES` property is always used in conjunction with the `CONTENT` property and its associated pseudo-classes `BEFORE` and `AFTER`. Using this property, it is possible to set different types of quotation marks to appear — primarily to set the appropriate quotation symbol for a given language.

### QUOTES

Description: Determines the type of quotation mark to be displayed within embedded quotations.

Media Group: Visual

CSS2 Family Type: Generated content, automatic numbering and lists

Values:

- ***string string*** - The open-quote and close-quote values to be used by the `CONTENT` property.
- **NONE** - No quotation marks are displayed.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

#### Listing 21-3 Quotes

```
<HTML>
<TITLE>Quotes</TITLE>
<STYLE>
Q:LANG(EN) {QUOTES: ' ' ' ' " " " "}
Q:BEFORE {CONTENT: OPEN-QUOTE}
Q:AFTER {CONTENT: CLOSE-QUOTE}
</STYLE>
<BODY>
In the lecture the professor said <Q>Shakespeare was the
one who said the immortal words <Q>to thine own self be
true</Q></Q>
</BODY>
</HTML>
```

---

The sole purpose of the `QUOTES` property is to set the type of quotation marks to be displayed in relation to generated content. `QUOTES` can take three different values: a text string, the named value `NONE` (which tells the



browser not to display any quotation marks) and INHERIT, which takes on any parent value for QUOTES.

The most interesting value here is the string value, which is set to the type of quotation marks you wish to have appear. Two sets of strings must be specified: the open and close quotation marks for two different levels, with the second set denoting the quotation marks to be displayed when a quote is nested within a quote. The following code snippet shows how you can do this for a typical set of English quotation marks:

```
Q:LANG(EN) {QUOTES: ' ' ' ' " " " " ;}
```

A property such as this is crucial when you are generating quotation marks using automatically generated content, as you need to be able to tell the browser what to display when one quote is contained within another. It also provides Web authors with the possibility of setting alternate quotation marks, such as those used in other languages. For example, the following code snippet shows how you could set this up for French quotation marks:

```
Q:LANG(FR) {QUOTES: '«' '»' "<" ">" ;}
```

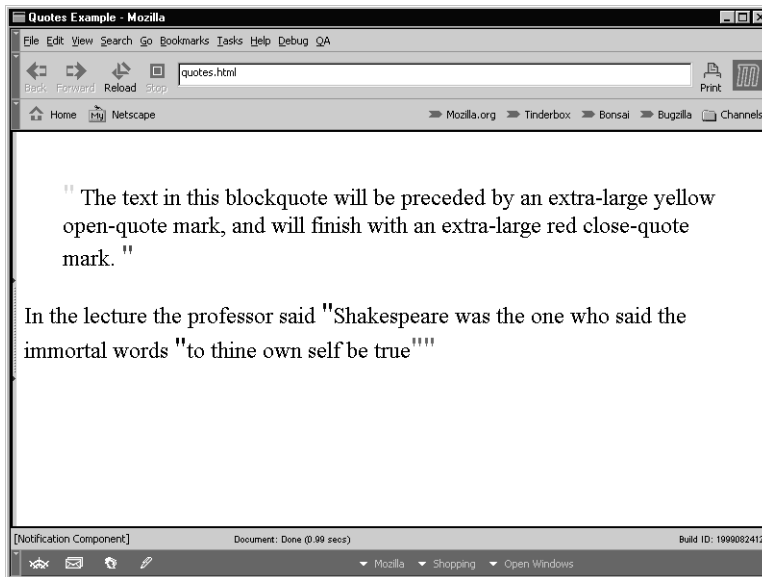
You can see how this can be applied to generated content in the following Web page code sample, displayed in the Mozilla browser in Figure 21–2.

#### Listing 21-4 Quotes Example

```
<HTML LANG="EN">
<HEAD>
<TITLE>Quotes Example</TITLE>
<STYLE>
BLOCKQUOTE:BEFORE {CONTENT: OPEN-QUOTE; COLOR: YELLOW;
FONT-SIZE: XX-LARGE;)}
BLOCKQUOTE:AFTER {CONTENT: CLOSE-QUOTE; COLOR: RED; FONT-
SIZE: XX-LARGE;)}
Q:LANG(EN) {QUOTES: ' ' ' ' " " " " ;}
Q:BEFORE {CONTENT: OPEN-QUOTE; COLOR: NAVY; FONT-SIZE: XX-
LARGE;)}
Q:AFTER {CONTENT: CLOSE-QUOTE; COLOR: FUCHSIA; FONT-SIZE:
XX-LARGE;)}
</STYLE>
</HEAD>
```

### Listing 21-4 Quotes Example (continued)

```
<BODY STYLE="FONT-SIZE: X-LARGE">
<P>
<BLOCKQUOTE>
The text in this blockquote will be preceded by an extra-
large yellow open-quote mark, and will finish with an
extra-large red close-quote mark.
</BLOCKQUOTE>
<P>
In the lecture the professor said <Q>Shakespeare was the
one who said the immortal words <Q>to thine own self be
true</Q></Q>
</BODY>
</HTML>
```



**Figure 21-2** QUOTES example as displayed within the Mozilla browser.

Unfortunately, this code is not properly displayed within the Mozilla browser. While the browser is capable of making the distinction between the types of quotation marks to be displayed around the `<BLOCKQUOTE>` and `<Q>`

tags, it does not do a proper job of setting the type of nested quotation marks to be displayed within the <Q> section.

This example does show how the OPEN-QUOTE and CLOSE-QUOTE value can be used with the CONTENT tag. Two other quotation-mark values are available for CONTENT: NO-OPEN-QUOTE and NO-CLOSE-QUOTE. These two values are to be used when you want quoted text to be nested within a quote, but do not want to have the quote marks displayed.

Neither of the popular browsers is capable of utilizing the QUOTES property.

## The COUNTER-INCREMENT and COUNTER-RESET Properties

The COUNTER-INCREMENT and COUNTER-RESET properties are both used to increase the range of automatic numbering options available within browsers. Using these properties, you can now automatically set such things as chapter headings with sub-sections. The COUNTER-INCREMENT property is used to establish which Web tag begins a sequence and when a value of “1” should be added to subsequent instances of that tag within the page. The COUNTER-RESET property determines which element should reset the counting for properties on the Web page (i.e., turn the increment value back to “0”).

### COUNTER-INCREMENT

Description: Defines the property to be automatically incremented, and the amount by which it is to be incremented.

Media Group: All

CSS2 Family Type: Generated content, automatic numbering and lists

Values:

- **<IDENTIFIER> <INTEGER>** - An identifier that selects the element to be incremented, and by how much.
- **NONE** - The default value.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

### Listing 21-5 Counter-Increment and Counter-Reset

```
<HTML>
<HEAD>
<TITLE>Counter-Increment and Counter-Reset</TITLE>
<STYLE>
H1:BEFORE {CONTENT: "Floor " counter(floor) ". "; COUNTER-
INCREMENT: floor; COUNTER-RESET: room;}
H2:BEFORE {CONTENT: counter(floor) "." counter(room) " ";
COUNTER-INCREMENT: room; }
</STYLE>
</HEAD>
<BODY>
<H1>Basement</H1>
Storage space
<H1>Main Floor</H1>
Reception
<H2>Restaurant</H2>
Frank's Fish & Steak House
</BODY>
</HTML>
```

---

## COUNTER-RESET

Description: Resets the numerical value set by the COUNTER-INCREMENT property.

Media Group: All

CSS2 Family Type: Generated content, automatic numbering and lists

Values:

- **<IDENTIFIER> <INTEGER>** - An identifier that selects the element to be incremented, and by how much.
- **NONE** - The default value.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

### Listing 21-6 Counter-Increment and Counter-Rest

```
<HTML>
<HEAD>
<TITLE>Counter-Increment and Counter-Rest</TITLE>
<STYLE>
H1:BEFORE {CONTENT: "Floor " counter(floor) " ";
COUNTER-INCREMENT: floor; COUNTER-RESET: room;}
H2:BEFORE {CONTENT: counter(floor) " " counter(room) " ";
COUNTER-INCREMENT: room; }
</STYLE>
</HEAD>
<BODY>
<H1>Basement</H1>
Storage space
<H1>Main Floor</H1>
Reception
<H2>Restaurant</H2>
Frank's Fish & Steak House
</BODY>
</HTML>
```

Both the COUNTER-INCREMENT and COUNTER-RESET properties take the same values: a text string that sets the identifier for the element to be incremented, as well as the value by which it should be incremented, plus the named values NONE (which tells the browser not to display any quote marks) and INHERIT (which takes on any parent value for these properties).

The string set for the COUNTER-INCREMENT and COUNTER-RESET properties works similarly for both properties. The text string works as an “identifier” to be associated with the property. These values are then defined. The following code snippet should make things clear:

```
H1:BEFORE {CONTENT: "Floor " counter(floor) " ";
COUNTER-INCREMENT: floor; COUNTER-RESET: room;}
```

Again, the CONTENT property plays a crucial role in how all of this works. It takes a text value and a special algorithm defined by two text values. In this case, whenever an <H1> appears on the page, the word “Floor ” (note the extra space) should appear, followed by the number as set by the formula “counter(floor)”. An extra space (“ ”) separates this in turn from whatever text is contained in this header. This is how the ATTR(X) value for COUNTER

works. You can actually use the named value ATTR with a string value representing the arbitrary value “X”. The value for counter is set by COUNTER-INCREMENT, and it should be reset whenever it meets the value set by the word “room”, as defined by COUNTER-RESET.

In the following code, the result should be an incremented list of the floors of a building along with the sections contained on each floor:

### Listing 21-7 Counter-Increment and Counter-Reset Example

```
<HTML>
<HEAD>
<TITLE>Counter-Increment and Counter-Reset Example</TITLE>
<STYLE>
H1:BEFORE {CONTENT: "Floor " counter(floor) " "; COUNTER-
INCREMENT: floor; COUNTER-RESET: room;}
H2:BEFORE {CONTENT: counter(floor) " " counter(room) " ";
COUNTER-INCREMENT: room; }
</STYLE>
</HEAD>
<BODY>
<H1>Basement</H1>
<H2>Storage space</H2>
<H1>Main Floor</H1>
<H2>Reception</H2>
<H2>Restaurant</H2>
<H2>Frank's Fish & Steak House</H2>
<H1>Second Floor</H1>
<H2>Men's Wear</H2>
<H1>Third Floor</H1>
<H2>Ladies' Wear</H2>
<H1>Fourth Floor</H1>
<H2>Children's Toys</H2>
<H1>Fifth Floor</H1>
<H2>TV and Stereos</H2>
<H2>Furniture</H2>
</BODY>
</HTML>
```

---

What should be displayed on screen appears in Figure 21–3:

As you can see, as we go up a floor, each of the floor numbers are incremented, but the areas on each floor are reset after every floor.

These properties are not yet supported in any major browser.

**Floor 1 Basement**

1 Storage space

**Floor 2 Main Floor**

1 Reception

2 Restaurant

3 Frank's Fish & Steak House

**Floor 3 Second Floor**

1 Men's Wear

**Floor 4 Third Floor**

1 Ladies' Wear

**Floor 5 Fourth Floor**

1 Children's Toys

**Figure 21-3** The COUNTER-INCREMENT and COUNTER-RESET sample code as it ought to be displayed.

## MARKER-OFFSET

The `MARKER-OFFSET` property is designed to create some extra “space” at the beginning of a list item. Some generated content may go outside of the normal bounds of the markers for most list items, so the `MARKER-OFFSET` property gives Web authors the leeway of adding extra room to accommodate the expanded marker.

### ***MARKER-OFFSET***

Description: Sets the distance between the border of a marker box in relation to the principal box with which it is associated.

Media Group: Visual

CSS2 Family Type: Generated content, automatic numbering and lists

Values:

- *n* value - Sets a length unit value.
- `AUTO` - The default length value.

- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

### Listing 21-8 Marker-Offset

```
<HTML>
<HEAD>
<TITLE>Marker-Offset</TITLE>
<STYLE>
P {MARGIN-LEFT: 10EM} /* Make space for counters */
LI:BEFORE {DISPLAY: MARKER; MARKER-OFFSET: 5EM; CONTENT:
counter(mycounter, UPPER-ALPHA) "."; COUNTER-INCREMENT:
mycounter;}
</STYLE>
</HEAD>
<BODY>
<P>
<OL>
<LI> Here is the first item.
<LI> Here is the second item.
<LI> Here is the third item.
</OL>
<P>
</BODY>
</HTML>
```

---

MARKER-OFFSET takes three different values: a numerical length value, the named value AUTO (which uses the browser's default value for spacing markers from the content of the list-item), and INHERIT (which assigns the parent value for MARKER-OFFSET).

The sample code from the property listing provides a good example of MARKER-OFFSET at work. It is set to a value of 10EM, which means that it is offset by a space equal to 10 em-spaces in the font that is being used. This should provide ample room in which generated content can appear due to the associated CONTENT property.

This property is not supported Netscape Navigator 4.0 or Internet Explorer 5.0.



## LIST-STYLE-TYPE

Under CSS2, a number of extra values have been added to the `LIST-STYLE-TYPE` attribute, originally introduced under CSS1. The majority of the new values enable Web authors to make list markers that use characters from other languages.

**Description:** This element sets the type of list-markers that appear before list items.

**Media Group:** Visual

**CSS2 Family Type:** Generated content, automatic numbering and lists

**Values:**

- `CIRCLE` | `DISC` | `SQUARE` - Sets the type of symbol to appear before a list element.
- `DECIMAL` | `DECIMAL-LEADING-ZERO` | `LOWER-ALPHA` | `LOWER-GREEK` | `LOWER-LATIN` | `LOWER-ROMAN` | `UPPER-ALPHA` | `UPPER-LATIN` | `UPPER-ROMAN` - Sets the type of alphanumeric symbol that appears before a list marker.
- `HEBREW` | `ARMENIAN` | `GEORGIAN` | `CJK-IDEOGRAPHIC` | `HIRAGANA` | `KATAKANA` | `HIRAGANA-IROHA` | `KATAKANA-IROHA` - Traditional numbering based upon the language type selected.
- `NONE` - No list-marker is displayed before a list item.
- `INHERIT` - Takes the same value as that set for the property of the parent element, if allowed.

**Sample code:**

```
<UL STYLE="LIST-STYLE-TYPE: LOWER-ALPHA">
<LI>List items
<LI>preceded by
<LI>lower-case letters
</UL>
```

`LIST-STYLE-TYPE` takes a wider variety of values under CSS2 than CSS1. Other than `INHERIT`, `LIST-STYLE-TYPE` also takes the following new values: `DECIMAL-LEADING-ZERO`, `LOWER-GREEK`, `HEBREW`, `GEORGIAN`, `ARMENIAN`, `CJK-IDEOGRAPHIC`, `HIRAGANA`, `KATAKANA`, `HIRAGANA-IROHA` and `KATAKANA-IROHA`. The `DECIMAL-LEADING-ZERO` value adds a numerical value that always displays at least two digits, with “0” leading the number if it is lower than the value “10”. `LOWER-GREEK` adds Greek letters in lower-case prior to each list-item,

functioning in much the same way that the CSS1 values LOWER-ALPHA and LOWER-roman do. The rest of the new values all add alphabetical listing in the native letter type of the specified language.

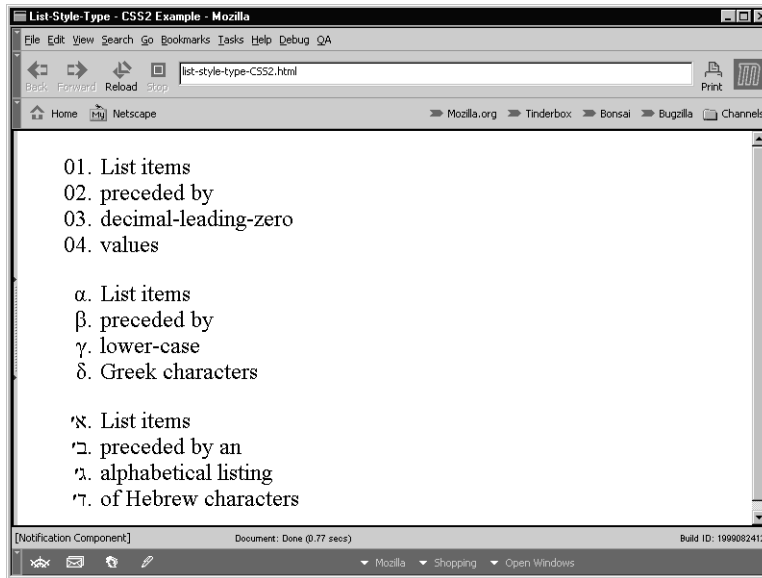
Neither of the major browsers support any of these additional values, although the Mozilla browser, which forecasts the types of features we can expect in Netscape Navigator 5.0, supports the DECIMAL-LEADING-ZERO, LOWER-GREEK and HEBREW.

The following code example can be seen displayed in the Mozilla browser in Figure 21-4:

### Listing 21-9 List-Style-Type - CSS2 Example

```
<HTML>
<HEAD>
<TITLE>List-Style-Type - CSS2 Example</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: X-LARGE; PADDING-LEFT: 1CM;">
<OL STYLE="LIST-STYLE-TYPE: DECIMAL-LEADING-ZERO;">
<LI>List items
<LI>preceded by
<LI>decimal-leading-zero
<LI>values
</OL>
<P>
<UL STYLE="LIST-STYLE-TYPE: LOWER-GREEK;">
<LI>List items
<LI>preceded by
<LI>lower-case
<LI>Greek characters
</UL>
<P>
<UL STYLE="LIST-STYLE-TYPE: HEBREW;">
<LI>List items
<LI>preceded by an
<LI>alphabetical listing
<LI>of Hebrew characters
</UL>
</BODY>
</HTML>
```

---




**Figure 21-4** Three new CSS2 values as seen in the Mozilla browser.

Because these values are now supported in `LIST-STYLE-TYPE`, they are also supported in the shortcut property `LIST-STYLE` under CSS2. For more information on the other values for `LIST-STYLE-TYPE` under CSS1, see Chapter 13, “The Classification Family of Properties”.

# PAGED MEDIA FAMILY OF PROPERTIES

## Topics in this Chapter

- 
- The SIZE Property
  - The MARKS Property
  - The PAGE-BREAK-BEFORE Property
  - The PAGE-BREAK-AFTER Property
  - The PAGE-BREAK-INSIDE Property
  - The PAGE Property
  - The ORPHANS Property
  - The WIDOWS Property

# Chapter 22

CSS2 introduces the idea of rendering Web pages as “paged media” — in other words, rendering a Web page specifically for print instead of the computer screen. The properties in the paged media family are designed to extend greater control over how Web pages are to be printed. Web authors now have control over how a Web page should look when it is printed.

The paged media properties are intended to solve the long-standing problem of rendering Web pages in a non-continuous medium, such as print. You now have greater control over such things as page breaks, the size of the printed page, setting the side (i.e., left or right) of a page, and more.

CSS2 does this by essentially extending the box model to the width of the printed page, making it a page box as defined by the various paged media properties. It is then up to the browser to take this information and produce printed output as designed by the Web author.

The paged media family consists of the following CSS2 properties:

- SIZE
- MARKS
- PAGE-BREAK-BEFORE
- PAGE-BREAK-AFTER
- PAGE-BREAK-INSIDE
- PAGE

- ORPHANS
- WIDOWS

## SIZE

The **SIZE** property is designed to set the orientation and size of the printed page box. With this property, Web authors can precisely control the dimensions for the printed output of their Web pages, and can even set whether or not the pages should be printed in portrait (the default) or landscape mode.

### SIZE

Description: Sets the size and orientation of the displayed or printed page.

Media Group: Visual, Paged

CSS2 Family Type: Paged media

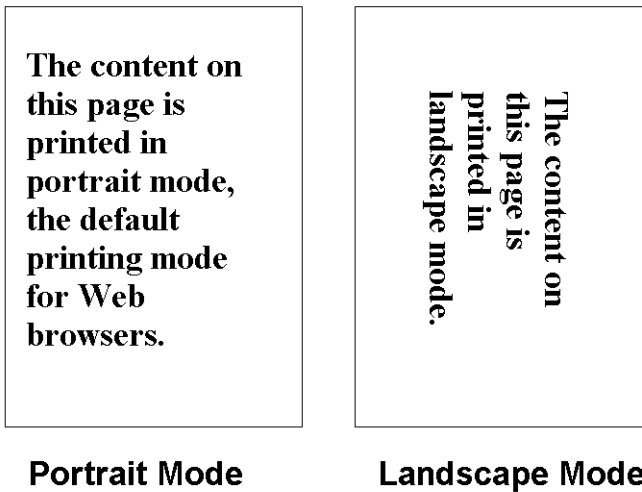
Values:

- ***nn*** - Sets the size for the page box in length units.
- **AUTO** - The page box is set to the size and orientation of the target sheet.
- **PORTRAIT** - Overrides the target's orientation and the shorter sides are horizontal.
- **LANDSCAPE** - Overrides the target's orientation and the longer sides are horizontal.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
@PAGE {SIZE: 8.5in 11in LANDSCAPE; MARGIN: 0.5IN}
```

The **SIZE** property can take any of the following values: a twin set of numeric values that set the width and height of the page, respectively, and **AUTO**, which sets the page box to the size and orientation of the target paper-size. There are also two orientation values: **PORTRAIT**, the default, which prints the content so that the content of the page flows down the long side of the page, and **LANDSCAPE**, which flows the content along the short side of the page. The following diagram should make the distinction between **PORTRAIT** and **LANDSCAPE** orientation clear:



**Figure 22-1** Portrait and Landscape printing modes displayed.

The `SIZE` property is used exclusively with the `@PAGE` media type (media types are covered in more depth in Chapter 16). It is used to set a value for the size and orientation of the page within the header of the page, which is the obvious place to set this sort of information for a Web page designed to be printed.

The following example code sets the size of the Web page to a typical 8.5 x 11 inch “letter” format, tells the browser to print the output in landscape mode, and sets a 0.5 inch margin on each side of the page:

#### **Listing 22-1 Size Example**

```
<HEAD>
<TITLE>Size Example</TITLE>
<STYLE>
@PAGE {SIZE: 8.5in 11in LANDSCAPE; MARGIN: 0.5IN}
</STYLE>
</HEAD>
```

**Listing 22-1 Size Example (continued)**

```
<BODY>
The content on this page will be printed in landscape mode,
have an indented margin of 0.5 inches on all side, and be
set for printing on regular 8.5 x 11 inch paper.
</BODY>
</HTML>
```

---

SIZE is not yet supported in either Netscape Navigator or Internet Explorer.

## MARKS

The MARKS property is designed to set the type of sign (or “mark”) to appear defining the boundaries of the printed page. In a print shop, these marks define where the pages would be trimmed before binding, or where the pages should be aligned. In this case, it really just defines the printable area of the Web page.

### MARKS

Description: This property sets the type of mark to be displayed, defining the page box. It can be used to add crop marks or cross marks defining the extent of the page.

Media Group: Visual, Paged

CSS2 Family Type: Paged media

Values:

- CROP - Adds crop marks to a page, used to determine where the page should be cut.
- CROSS - Adds cross marks to a page, used to align sheets.
- NONE - No mark is made visible.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

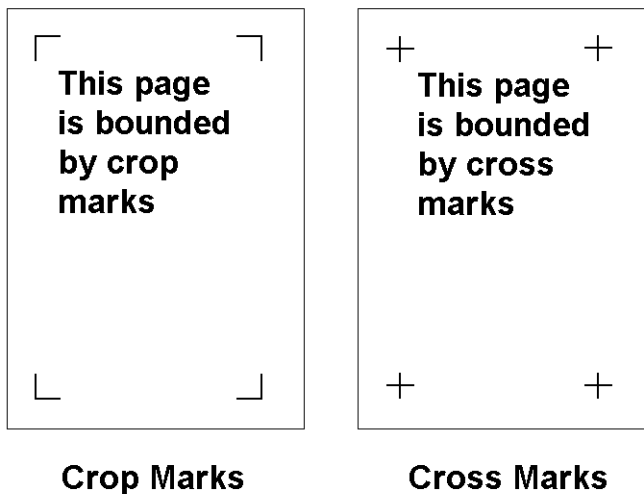
Sample Code:

```
<STYLE>
@PAGE {SIZE: 8.5IN 11IN; MARKS: CROP;}
</STYLE>
```



The MARKS property is used in conjunction with the @PAGE media type. The MARKS property can take one of four different named values: CROP, which adds a crop mark a page; CROSS, which adds a cross mark used to align sheets; NONE (the default), where no mark is visible; and the ubiquitous INHERIT, which takes on whatever parent value may have already been set for the property.

The CROP and CROSS values are handy for determining the boundary of printing margins for the page, giving Web authors a better idea as to where the margins on the printed page appear. The difference between the two types of marks can be seen in the following illustration:



**Figure 22-2** Crop and Cross marks displayed.

As of yet, the MARKS property has not been implemented in either Internet Explorer or Netscape Navigator.

## The Page Break Properties

The paged media family comes with three properties specifically designed to set where page breaks should (or should not) occur within Web pages when they are printed: `PAGE-BREAK-BEFORE`, `PAGE-BREAK-AFTER` and `PAGE-BREAK-INSIDE`. The first two accomplish the task of inserting a page break. `PAGE-BREAK-INSIDE` operates using a sub-set of values assigned to both `PAGE-BREAK-BEFORE` and `PAGE-BREAK-AFTER` and is used to prevent a page-break within a particular part of the Web page. All three are used in conjunction with block-level properties, so page breaks are not set from within a line of text.

### ***PAGE-BREAK-BEFORE***

Description: Tells the browser to insert a page break before the selected block-level element.

Media Group: Visual, Paged

CSS2 Family Type: Paged media

Values:

- **AUTO** - Neither forces nor prohibits a page break from occurring.
- **ALWAYS** - Tells the browser to always force a page break to occur before the block-level element.
- **AVOID** - Tells the browser to avoid forcing a page break before the block-level element.
- **LEFT** - Forces one or more page breaks to occur before the block-level element so that it appears on a left-hand page.
- **RIGHT** - Forces one or more page breaks to occur before the block-level element so that it appears on a right-hand page.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

**Listing 22-2 Page-Break-Before**

```
<HTML>
<HEAD>
<TITLE>Page-Break-Before</TITLE>
<STYLE>
H1 {PAGE-BREAK-BEFORE: ALWAYS;}
</STYLE>
</HEAD>
<BODY>
Here is some text.
<H1>This header will appear on a new page</H1>
</BODY>
</HTML>
```

---

## ***PAGE-BREAK-AFTER***

Description: Tells the browser to insert a page break after the selected block-level element.

Media Group: Visual, Paged

CSS2 Family Type: Paged media

Values:

- **AUTO** - Neither forces nor prohibits a page break from occurring.
- **ALWAYS** - Tells the browser to always force a page break to occur after the block-level element.
- **AVOID** - Tells the browser to avoid forcing a page break after the block-level element.
- **LEFT** - Forces one or more page breaks to occur after the block-level element so that it appears on a left-hand page.
- **RIGHT** - Forces one or more page breaks to occur after the block-level element so that it appears on a right-hand page.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

### Listing 22-3 Page-Break-After

```
<HTML>
<HEAD>
<TITLE>Page-Break-After</TITLE>
<STYLE>
P {PAGE-BREAK-AFTER: ALWAYS;}
</STYLE>
</HEAD>
<BODY>
<P>
Each subsequent paragraph will be forced onto a new page.
</P>
<P>
See, this is on a second page.
</P>
<P>
This is on yet another page.
</P>
</BODY>
</HTML>
```

---

## ***PAGE-BREAK-INSIDE***

Description: Tells the browser to insert a page break within the selected block-level element.

Media Group: Visual, Paged

CSS2 Family Type: Paged media

Values:

- **AUTO** - Neither forces nor prohibits a page break from occurring.
- **AVOID** - Tells the browser to avoid forcing a page break within the block-level element.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

**Listing 22-4 Page-Break-Inside**

```
<HTML>
<HEAD>
<TITLE>Page-Break-Inside</TITLE>
<STYLE>
BLOCKQUOTE {PAGE-BREAK-INSIDE: AVOID;}
</STYLE>
</HEAD>
<BODY>
<BLOCKQUOTE>No matter how long this quote may be, the PAGE-
BREAK-INSIDE property will ensure that a page break does
not occur inside of it, breaking it across more than one
page.
</BLOCKQUOTE>
</BODY>
</HTML>
```

Both `PAGE-BREAK-BEFORE` and `PAGE-BREAK-AFTER` share the following values: `AUTO`, `AVOID`, `LEFT`, `RIGHT`, `ALWAYS` and `INHERIT`. The `AUTO` value is the default setting, and neither forces nor prohibits a page break from occurring at a specific point. `AVOID` tells the browser to specifically avoid forcing a page break before a given block-level element. The `LEFT` value forces a page break to occur so that subsequent text appears on a left-facing page, and the `RIGHT` value forces a page break to occur so that the subsequent text appears on a right-facing page. (When using the `LEFT` and `RIGHT` values, be aware that in order for the browser to produce a subsequent page that is either to be printed as a left- or right-facing page, a blank page may need to be inserted. For example, if the previous page is left-facing and `LEFT` is used, then a blank right page would have to be present before the next left-facing page could appear). The `ALWAYS` value sets the page break after a given block-level element when used in `PAGE-BREAK-AFTER`, and before it in `PAGE-BREAK-BEFORE`. The `INHERIT` value takes on whatever page-break value has already been set by the parent. While `PAGE-BREAK-BEFORE` and `PAGE-BREAK-AFTER` share all of these values, `PAGE-BREAK-INSIDE` only takes on `AUTO`, `AVOID` and `INHERIT`.

The `PAGE-BREAK-BEFORE` property is used when you want to set a page-break immediately before a given block level element, as the following example code shows:

**Listing 22-5 Page-Break-Before Example**

```
<HTML>
<HEAD>
<TITLE>Page-Break-Before Example</TITLE>
<STYLE>
H1 {PAGE-BREAK-BEFORE: ALWAYS;}
</STYLE>
</HEAD>
<BODY>
Here is some text.
<H1>This header will appear on a new page</H1>
</BODY>
</HTML>
```

---

In this particular case, the browser has been instructed to put in a page break immediately before any top-level header that appears on the Web page. This scenario makes a lot of sense if you want each top-level header to be displayed on a new page. You could also set things so that you could have every paragraph on a Web page printed on a different page, as the following PAGE-BREAK-AFTER code example shows:

**Listing 22-6 Page-Break-After Example**

```
<HTML>
<HEAD>
<TITLE>Page-Break-After Example</TITLE>
<STYLE>
P {PAGE-BREAK-AFTER: ALWAYS;}
</STYLE>
</HEAD>
<BODY>
<P>
Each paragraph will be forced onto a new page.
</P>
<P>
See, this is on a second page.
</P>
```

**Listing 22-6 Page-Break-After Example (continued)**

```
<P>
This is on yet another page.
</P>
</BODY>
</HTML>
```

---

Use the `PAGE-BREAK-INSIDE` property when you deliberately want to make sure that a page-break does not occur *within* a given block-level element. Using this property ensures that whatever Web element you select remains on the same printed page, as the following code example shows:

**Listing 22-7 Page-Break-Inside Example**

```
<HTML>
<HEAD>
<TITLE>Page-Break-Inside Example</TITLE>
<STYLE>
BLOCKQUOTE {PAGE-BREAK-INSIDE: AVOID;}
</STYLE>
</HEAD>
<BODY>
<BLOCKQUOTE>No matter how long this quote may be, the PAGE-
BREAK-INSIDE property will ensure that a page break does
not occur inside of it, breaking it across more than one
page.
</BLOCKQUOTE>
</BODY>
</HTML>
```

---

Together, all three properties present Web authors with the opportunity to explicitly set where printed page-breaks should or should not occur.

None of these properties are currently implemented within either Netscape Navigator or Internet Explorer.

## PAGE

The `PAGE` property is used to set a specific printed property to elements contained within a printed Web page. In other words, it allows Web authors to specify how certain elements within a page should appear. If, for example, you wanted all tables to be printed in landscape mode, you simply attach a name to the `PAGE` property and then associate that name with the specific tables you want printed in landscape mode.

### ***PAGE***

Description: Used to set a particular page type to be associated with a particular type of Web element.

Media Group: Visual, Paged

CSS2 Family Type: Paged media

Values:

- ***name*** - An identifier that is associated with the element.
- **AUTO** - Follows the automatic formatting for the element. This is the default value.

Sample Code:

#### Listing 22-8 Page Example

```
<HTML>
<HEAD>
<TITLE>Page</TITLE>
<STYLE>
@PAGE regular {SIZE 8.5IN 11IN; MARGIN: 1.5IN;}
@PAGE landscape {SIZE: LANDSCAPE;}
DIV {PAGE: regular;}
TABLE {PAGE: landscape;}
</STYLE>
</HEAD>
<BODY>
<DIV>
The idea behind the PAGE property is to create a page break
<I>and</I> enable the Web author to format the content of
the subsequent page in a different manner than that which
went before it.
</DIV>
```



**Listing 22-8 Page Example (continued)**

```
<TABLE>
<TR>
<TD>The content in this table will be set in a new page,
and will appear in a landscape format.</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

There are two possible values for `PAGE`: a name value set by the Web author (which could be anything), and `AUTO` (the default), which follows the automatic formatting for the given element. A good example of how to use this property can be seen in the extensive example code provided in the definition listing for the property. In this case, the content on the page is printed as usual, but when the printer meets the table contained on the page, the `PAGE` property associated with it tells the printer to print the table on a separate page and in landscape mode rather than portrait mode.

This property is not yet supported in either Netscape Navigator or Internet Explorer.

## WIDOWS and ORPHANS

“Widows” and “orphans” are old printing terms used to specify the minimum amount of lines of text that should appear at the top and bottom of a printed page, respectively. The CSS2 properties accomplish the same thing, ensuring that a minimum amount of text appears at the top and bottom of each page.

### WIDOWS

Description: Sets the minimum number of lines of a paragraph that must be left at the top of a page.

Media Group: Visual, Paged

CSS2 Family Type: Paged media

Values:

- *n* - An integer value that assigns the number of lines that must appear in a paragraph; otherwise, it will be forced to appear on the previous page. Default value is 2.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<STYLE>
@PAGE {WIDOWS: 3;}
</STYLE>
```

## ORPHANS

Description: Sets the minimum number of lines of a paragraph that must be left at the bottom of a page.

Media Group: Visual, Paged

CSS2 Family Type: Paged media

Values:

- *n* - An integer value that assigns the number of lines that must appear in a paragraph; otherwise, it will be forced to appear on the next page. Default value is 2.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<STYLE>
@PAGE {ORPHANS: 3;}
</STYLE>
```

Both the WIDOWS and ORPHANS properties share the same values: an integer value that assigns the number of lines that must appear in a paragraph before it is forced to move to another page (one way or the other), and the named value INHERIT, which takes on whatever parent value may have already been set. The default numeric value for both WIDOWS and ORPHANS is 2, ensuring that at least two lines of text in a paragraph are either at the top or bottom of a page before a page-break will occur.

The usage for these two properties is not fully qualified in the CSS2 specification, although it would make sense that they would be used in conjunction with the @PAGE media type. Be careful when assigning numeric values

for either WIDOWS or ORPHANS, as too large a value will only add to the amount of white-space added to each page.

Neither WIDOWS nor ORPHANS are supported in the two major browsers at the moment.

# FONT FAMILY PROPERTIES

## Topics in this Chapter

- The FONT-STRETCH Property
- The FONT-SIZE-ADJUST Property
- @FONT-FACE



# Chapter 23

Under CSS1, the font family consists of a number of properties designed to specify and alter the way fonts are displayed on screen. While they have many capabilities, it is still possible in many instances for the CSS1 properties to “fall through the cracks”, with the intended font display coming out incorrectly.

To prevent this, CSS2 goes further, filling in the “gap” by adding properties that can describe exactly the type of font to be displayed, improve font-matching, and enable fonts to be “created” by providing a means for the browser to download the appropriate font from an online source.

To this extensive family of properties, CSS2 adds two more:

- FONT-STRETCH
- FONT-SIZE-ADJUST

It also adds an “at-rule” for specifying the name of a font descriptor:

- @FONT-FACE

Taken together, these new elements round out the family of font properties under CSS2.

## FONT-STRETCH

The `FONT-STRETCH` property is designed to provide a normal, condensed or expanded font face on demand from the Web author. This property differs from the CSS1 property `FONT-WEIGHT` property; `FONT-WEIGHT` deals with the “boldness” assigned to a font, while `FONT-STRETCH` is capable of actually changing the relative width of the individual characters that comprise the displayed font.

### ***FONT-STRETCH***

Description: Sets how condensed or expanded a selected font element should be.

Media Group: Visual

CSS2 Family Type: Font

Values:

- `ULTRA-CONDENSED | EXTRA-CONDENSED | CONDENSED | SEMI-CONDENSED | NORMAL | SEMI-EXPANDED | EXPANDED | EXTRA-EXPANDED | ULTRA-EXPANDED` - Range of fonts from most condensed to most expanded.
- `WIDER` - Sets the selected font to a more expanded value.
- `NARROWER` - Sets the selected font to a more condensed value.
- `INHERIT` - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="FONT-STRETCH: ULTRA-CONDENSED;">An  
Ultra-Condensed Header</H1>
```

`FONT-STRETCH` does exactly what it says: it either compresses or stretches the appearance of a displayed font. `FONT-STRETCH` comes with ten different named values. Ranging from most condensed to most expanded, these values are `ULTRA-CONDENSED`, `EXTRA-CONDENSED`, `CONDENSED`, `SEMI-CONDENSED`, `NORMAL`, `SEMI-EXPANDED`, `EXPANDED`, `EXTRA-EXPANDED` and `ULTRA-EXPANDED`. The universally applicable CSS2 property `INHERIT`, which takes the parent value set for `FONT-STRETCH`, is also available.

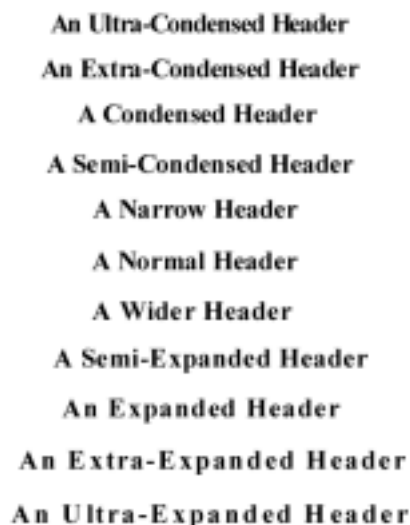
The following sample code, depicted in Figure 23–1, shows how you can use these values and the effects they have on the displayed font:

### Listing 23-1 Font-Stretch Example

```
<HTML>
<HEAD>
<TITLE>Font-Stretch Example</TITLE>
</HEAD>
<BODY>
<H3 STYLE="FONT-STRETCH: ULTRA-CONDENSED;">An Ultra-
Condensed Header</H3>
<H3 STYLE="FONT-STRETCH: EXTRA-CONDENSED;">An Extra-
Condensed Header</H3>
<H3 STYLE="FONT-STRETCH: CONDENSED;">A Condensed Header
</H3>
<H3 STYLE="FONT-STRETCH: SEMI-CONDENSED;">An Semi-
Condensed Header</H3>
<H3 STYLE="FONT-STRETCH: NARROWER;">A Narrower Header</H3>
<H3 STYLE="FONT-STRETCH: NORMAL;">A Normal Header</H3>
<H3 STYLE="FONT-STRETCH: WIDER;">An Wider Header</H3>
<H3 STYLE="FONT-STRETCH: SEMI-EXPANDED;">A Semi-Expanded
Header</H3>
<H3 STYLE="FONT-STRETCH: EXPANDED;">An Expanded Header</H3>
<H3 STYLE="FONT-STRETCH: EXTRA-EXPANDED;">An Extra-
Expanded Header</H3>
<H3 STYLE="FONT-STRETCH: ULTRA-EXPANDED;">An Ultra-
Expanded Header</H3>
</BODY>
</HTML>
```

---

As you can see in this case, the WIDER value is equal to the SEMI-EXPANDED value, and the NARROWER value is equal to the SEMI-CONDENSED value. That is because these values are relative to whatever FONT-STRETCH value has been set. In this case, as no FONT-STRETCH value has been set, they are set to the default, which is NORMAL, so WIDER is one size larger, and NARROWER is one size smaller.



An Ultra-Condensed Header  
An Extra-Condensed Header  
A Condensed Header  
A Semi-Condensed Header  
A Narrow Header  
A Normal Header  
A Wider Header  
A Semi-Expanded Header  
An Expanded Header  
An Extra-Expanded Header  
An Ultra-Expanded Header

**Figure 23-1** The effects of the various FONT-STRETCH values illustrated.

## FONT-SIZE-ADJUST

The FONT-SIZE-ADJUST property gives Web authors control over the legibility of the font displayed. It effectively extends the functionality of the FONT-SIZE property, by addressing an inherent problem with specifying font sizes: not all font sizes are equal. A courier font set to 12 points does not take up the same space on screen as 12 point Times Roman, 12 point Helvetica or just about any other font. You can see this illustrated in Figure 23-2, which uses a number of different fonts all set to the same point size.

This is primarily due to the different aspect values of various fonts, which is the value of the size of the font divided by its x-height. While fonts of the same point size may have a different display size, fonts of the same aspect value appear to have the same size. FONT-SIZE-ADJUST enables Web authors to set the aspect value for the fonts on a Web page, so that they all appear with the same legibility, as Figure 23-3 depicts.



**Shelby Valante 12 Point**

*The quick brown fox jumped over the lazy dog.*

**Verdana 12 Point**

The quick brown fox jumped over the lazy dog.

**Garamond 12 Point**

The quick brown fox jumped over the lazy dog.

**Arial 12 Point**

The quick brown fox jumped over the lazy dog.

**Comic Sans MS 12 Point**

The quick brown fox jumped over the lazy dog.

**Century Gothic 12 Point**

The quick brown fox jumped over the lazy dog.

**Figure 23-2** Different fonts set to the same point size (12 points).

**Shelby Valante**

*The quick brown fox jumped over the lazy dog.*

**Verdana**

The quick brown fox jumped over the lazy dog.

**Garamond**

The quick brown fox jumped over the lazy dog.

**Arial**

The quick brown fox jumped over the lazy dog.

**Comic Sans MS**

The quick brown fox jumped over the lazy dog.

**Century Gothic**

The quick brown fox jumped over the lazy dog.

**Figure 23-3** The same fonts with similar legibility values.

## FONT-SIZE-ADJUST

Description: Sets the aspect value for a selected font element. Useful in keeping the “look” of the size of a substituted font the same or close to that of the intended font.

Media Group: Visual

CSS2 Family Type: Font

Values:

- *n* - Sets the aspect value.
- NONE - Browser does not preserve the aspect value between fonts.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="FONT-SIZE: 12pt; FONT-FAMILY: COURIER;
FONT-SIZE-ADJUST: 1.5;">Font-Size-Adjust-ed
Header</H1>
```

FONT-SIZE-ADJUST takes one of three values: a numerical value (which is equivalent to the aspect value), plus the named values NONE (which does not preserve the aspect value between fonts) and INHERIT (which takes on whatever parent value exists for FONT-SIZE-ADJUST).

The general rule is, the larger the aspect value, the more legible the font will appear. FONT-SIZE-ADJUST uses the following formula to calculate the aspect value:

$$y(a/a') = c$$

where:

*y* = the “font-size” of first font chosen

*a'* = the aspect value of the *available* font

*c* = the proper “font-size” to apply to the available font

In the event that a font match can’t be made, the FONT-SIZE-ADJUST property takes the next available font, and applies the aspect value of the chosen font to the font that is available. This ensures that the font that is displayed is as legible as the one the Web author had originally intended to appear.

Examples of how this property is to be specified in code are not provided in the CSS2 specification. It takes a value that sets the aspect value, and it must be associated with at least one font name, so the following code is an educated guess as to how it might be implemented:

```
<H1 STYLE="FONT-SIZE: 12pt; FONT-FAMILY: COURIER;  
FONT-SIZE-ADJUST: 1.5;">Font-Size-Adjust-ed  
Header</H1>
```

In this case, the chosen font family is courier. If this is not available on the user's system, then the closest available font is chosen, and then the formula from the FONT-SIZE-ADJUST property is applied to the value "1.5", and the resulting font is set to that value.

## The @FONT-FACE Pseudo-Element

The new @FONT-FACE pseudo-element provides the mechanism by which Web authors can specify fonts that can be downloaded by the user and then displayed directly within the Web page. It can take a number of values, many of them derived from the many CSS2 font specific properties, and it works in exactly the same way.

The following is a sample Web page that will attempt to download the font "Mildly-funky" and then display that font when requested in a Web page:

### Listing 23-2 @Font-Face Example

```
<HTML>  
<HEAD>  
<TITLE>@Font-Face Example</TITLE>  
<STYLE>  
@FONT-FACE {FONT-FAMILY: "Mildly-Funky"; SRC:URL("http://  
www.FunkyFontsAreUs.com/newfonts/mild-funky"); FONT-STYLE:  
NORMAL, ITALIC;}  
BODY {FONT-FAMILY: "Mildly-funky", NORMAL}  
H1 {FONT-FAMILY: "Mildly-funky", ITALIC}  
</STYLE>  
</HEAD>
```

**Listing 23-2 @Font-Face Example (continued)**

```
<BODY>
<H1>This heading uses an Italic Mildly Funky Downloaded
Font</H1>
The text in the remaining body of the Web page uses the
same font as in the header, but uses the normal (i.e., un-
italicized) version of it.
</BODY>
</HTML>
```

---

In this case, the `@FONT-FACE` pseudo-element looks up a reference for the “Mildly-Funky” font on the fictitious Web site “<http://www.FunkyFontsAreUs.com/newfonts/mild-funky>”. Subsequently, style sheet information is provided that tells the browser to use that font for both the body of the document and any top-level headers, rendering the first in a normal, non-italic way, and the latter in an italic version of the “Mildly-Funky” font.

As you can see from this example, the `@FONT-FACE` pseudo-element relies upon other values to help “shape” the font that gets used on screen. It can use any of the following CSS2 properties as values, all of which work in exactly the same way as their property equivalents: `FONT-FAMILY`, `FONT-STYLE`, `FONT-VARIANT`, `FONT-STRETCH`, `FONT-SIZE`, `FONT-WEIGHT`, `FONT-STYLE`, `FONT-VARIANT`, `FONT-WEIGHT`. As these are already covered in other parts of this book, there is no need to repeat the same information here. It also uses a number of new values: `UNICODE-RANGE`, `UNITS-PER-EM`, `SRC`, `PANOSE-1`, `STEMV`, `STEMH`, `SLOPE`, `CAP-HEIGHT`, `X-HEIGHT`, `ASCENT`, `DESCENT`, `WIDTHS`, `BBOX`, `DEFINITION-SRC`, `BASELINE`, `CENTERLINE`, `TOPLINE` and `MATHLINE`. These need some additional explanation.

The `UNICODE-RANGE` value is an optional value used by Web authors who want to make sure that a font being downloaded is capable of displaying certain characters. There are many fonts that can only provide a specific number of characters corresponding to a certain range within the Unicode font specification. `UNICODE-RANGE` takes a hexadecimal number (which can include wildcard characters) that refers to a specific Unicode character range. For example, if you wanted to make sure that the font you are downloading has all of the characters in the standard Latin-1 font (used by most Western languages) you would write the following code:

```
UNICODE-RANGE: U+00??
```

This covers the Unicode range of 0000 to 00FF, which contains all of characters in the Latin-1 font. If the selected downloadable font does not cover this Unicode range of characters, it will not be downloaded or displayed. If you wanted to make sure your font included all fractional characters, you'd specify "U+215?", and if you wanted to ensure that your font covered all Tibetan characters, you'd specify "U+OF??". The Table 24-1 provides a sample of the languages supported in the first fractional part (ranging from the hexadecimal range 0000-0FFF) of the entire Unicode specification.

**Table 23-1 The hexadecimal range 0000-0FFF for the Unicode specification, and the languages covered**

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
0																
1																
2																
3																
4																
5																
6																
7																
8																
9																
0A																
0B																
0C																
0D																
0E																
0F																

Not all sections in the Unicode specification have been confirmed or allotted, and these spaces either contain question marks or are blank. For a complete listing of all of the names and ranges for existing Unicode characters, visit <http://www.indigo.ie/egt/standards/iso10646/bmp-roadmap-table.html>, from which the information in Table 24-1 is largely derived.

The `UNITS-PER-EM` value takes an integer that sets the width of the letter “m” (a standard square measure used for defining font character size) for the font being downloaded. Common values for this are 250 under the Intellifont standard, 1000 for Type 1, and 2048 for TrueType, OpenType and TrueType GX.

`SRC` takes a URL value and determines the source from which the font file should be downloaded. An example of how this can be used can be seen in the code sample provided earlier.

`PANOSE-1` is used to match the downloadable font to a standard TrueType font classification. It takes a 10-digit value that categorizes the key display features of the Latin typeface (used by most Western languages).

`STEMV` and `STEMH` take a numeric value and set the vertical and horizontal stem width (meaning the width for the “stroke” that makes up a character) for the downloaded font. If these are used, the `UNITS-PER-EM` value must be as well. The `SLOPE` value also takes an integer value, and it sets the vertical stroke angle for the font being downloaded.

`CAP-HEIGHT` takes an integer value and is used to set the height for a font’s uppercase characters. `X-HEIGHT` works in the same way, but is used to set the height of the lowercase characters instead. If these are used, the `UNITS-PER-EM` value must be as well.

`ASCENT` and `DESCENT` set the maximum unaccented height and depth for the font. Essentially, they set, respectively, the height for the highest displayed part of a font (such as the top of a “T” or “I”) and the depth of the lowest displayed part of a font (such as the lowest part of the letters “g” or “j”), in their unaccented forms. Characters that are accented in some way (such as “Ñ” or “Ç”) are excluded from this measure. If these values are used, the `UNITS-PER-EM` value must be as well.

When a font has to be synthesized, the `WIDTHS`, `BBOX` and `DEFINITION-SRC` values come into play. These descriptors are intended to help provide a description for the font to be synthesized. `WIDTHS` sets the value for the width of individual characters (also known as “glyphs”), and it takes a Unicode range followed by a numeric value that sets the width for each glyph represented in the Unicode range. In the follow code snippet, `WIDTHS` is used to set the widths of all Latin-1 fonts to a value of 1500.

```
WIDTHS: U+00?? 1500
```

The `BBOX` value sets the maximum-bounding box (in other words, the maximum size of the “box” in which all glyphs for a font appear). It takes four numeric values, which set the dimensions for the lower left x, lower left y, upper right x, and upper right y for the bounding box for the font.

The `DEFINITION-SRC` value works in the same way as `SRC`, and points to a specific font resource.

Finally, there are four descriptors that define the alignment values for the font. They are `BASELINE`, `CENTERLINE`, `TOPLINE` and `MATHLINE`. `BASELINE`, `CENTERLINE` and `TOPLINE` sets the lower-baseline, central baseline and top baseline values for the font. `MATHLINE` sets the mathematical baseline for the font. All of these values take a numeric value, and, if specified, all of the descriptors must be used in conjunction with the `DEFINITION-SRC` value.

As of yet, none of these font values are supported in either Internet Explorer or Netscape Navigator.

# TEXT FAMILY PROPERTIES

## Topics in this Chapter

- The TEXT-SHADOW Property





# Chapter 24

There are a number of different properties contained within the text family under CSS1: `TEXT-INDENT`, `TEXT-ALIGN`, `TEXT-DECORATION`, `LETTER-SPACING`, `WORD-SPACING`, `TEXT-TRANSFORM`, `WHITE-SPACE`. There is but one wholly new additional property introduced under the CSS2 specification: `TEXT-SHADOW`.

## TEXT-SHADOW

The `TEXT-SHADOW` property does exactly what it describes — it adds a background shadow to the selected text. This shadow can appear behind and to the left or right of the text, or even in front of the text. Since it is a “shadow”, the shadow never actually overlays the text itself, although the shadow may overlay itself, depending on how sharp or how blurry you make the ends of the shadow. According to the CSS specification, the shadow does not extend the box of its Web element, and it can therefore overlay any text that may lie immediately adjacent to the effected text.

## TEXT-SHADOW

Description: Adds a shadow effect to text.

Media Group: Visual

CSS2 Family Type: Text

Values:

- **NONE** - No shadow value is set.
- ***n n [n]***, - Sets a color value plus the unit length of the shadow. At least two length unit values must be present. A comma can separate a chain of overlaying values.
- ***color name*** or ***numerical color value*** - Sets the color value for the shadow. Always associated with a length unit value.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="TEXT-SHADOW: GRAY 2PX 3PX, LIME -4PX 3PX  
4PX;">Header with Drop Shadow Effect</H1>
```

There are several different combinations of values that can be set for **TEXT-SHADOW**. There are two different comma separated values that set how much the shadow is offset from the text. The first number sets the horizontal length of the shadow from the text, and the second number does the same for the vertical length. By default, the first numeric value sets the shadow to the right of the text, and the second numeric value sets the shadow below the text. **TEXT-SHADOW** can accept negative values for these as well, so it is possible to set the shadow the left and above the text by using negative numerical values. **TEXT-SHADOW** can take a third numeric length value, which sets the amount of “blur” at the boundaries of the shadow. How this should work is not explicitly stated in the CSS2 specification, so results may vary from browser to browser if and when this property is implemented.

The shadow can also take a specific color value, which can either be set as a standard color name or a numerical color value. If no color value is set, the shadow takes on the default value set by the **COLOR** property — and if that isn’t explicitly set, it is whatever color value is associated with the text (the default is therefore black). You can also set more than one shadow to a given selection of text.

The following code example displays several different values for the TEXT-SHADOW property:

### Listing 24-1 Text-Shadow Example

```
<HTML>
<HEAD>
<TITLE>Text-Shadow Example</TITLE>
</HEAD>
<BODY>
<P>
<DIV ALIGN="center">
<H1 STYLE="TEXT-SHADOW: 5PX 5PX;">Drop Shadow</H1>
<P>
<H1 STYLE="TEXT-SHADOW: -5PX -5PX RED;">Drop Shadow</H1>
<P>
<H1 STYLE="TEXT-SHADOW: -5PX 5PX 5PX SILVER;">Drop
Shadow</H1>
<P>
<H1 STYLE="TEXT-SHADOW: GRAY 10PX 10PX, LIME -5PX 5PX;">
Drop Shadow</H1>
</DIV>
</BODY>
</HTML>
```

---

You can see how this code ought to be displayed in Figure 24–1.

This property is not yet supported in any major browser.

**Drop Shadow**  
**Drop Shadow**  
**Drop Shadow**  
**Drop Shadow**

**Figure 24-1** Sample TEXT-SHADOW code as is should be displayed in a compliant browser.



# TABLE FAMILY PROPERTIES

## Topics in this Chapter

- The CAPTION-SIDE Property
- The TABLE-LAYOUT Property
- The BORDER-COLLAPSE Property
- The BORDER-SPACING Property
- The EMPTY-CELLS Property
- The SPEAK-HEADER Property

# Chapter 25

Another of the new sets of properties that CSS2 introduces is the set of table properties. This set of wholly new properties is designed to provide Web authors with much greater control over the appearance of tables and table elements — even how the table contents are spoken aloud.

The table family consists of the following six new properties:

- `CAPTION-SIDE`
- `TABLE-LAYOUT`
- `BORDER-COLLAPSE`
- `BORDER-SPACING`
- `EMPTY-CELLS`
- `SPEAK-HEADER`

## CAPTION-SIDE

The `CAPTION-SIDE` property enables Web authors to place the caption box in relation to the rest of the table. It works in much the same fashion to the little-used `<CAPTION>` HTML tag, but allows for greater flexibility in the placement of table captions.

## CAPTION-SIDE

Description: Specifies where the caption will be in relation to the table.

Media Group: Visual

CSS2 Family Type: Tables

Values:

- TOP - Positions the caption above the table.
- BOTTOM - Positions the caption below the table.
- LEFT - Positions the caption to the left side of the table.
- RIGHT - Positions the caption to the right side of the table.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<TABLE BORDER>

<CAPTION STYLE="CAPTION-SIDE: LEFT;">A Simple
Table</CAPTION>

<TR>

<TD>This text is in the table</TD>

</TR>

</TABLE>
```

CAPTION-SIDE can take any one of up to five different named values: TOP, BOTTOM, LEFT, RIGHT and INHERIT. The <CAPTION> tag can actually take on a similar set of values (VALIGN=TOP or BOTTOM, ALIGN=TOP, BOTTOM, LEFT, RIGHT, or CENTER), but the LEFT and RIGHT values for CAPTION-SIDE produce different results.

When you add a caption to a table, it appears outside the boundaries of the table's contents, but is bound to appear in a space alongside the table itself (in other words, if you set a visible BORDER value to the table, the caption will be located outside of the table's border).

When you use the ALIGN=LEFT or ALIGN=RIGHT value with the <CAPTION> tag, it aligns the text in the caption; it does not alter where the caption appears in relation to the table. That is where the LEFT and RIGHT values for CAPTION-SIDE come into effect: they place the caption to the left or right, respectively, of the table itself. You can see examples of all four



native values for CAPTION-SIDE displayed in the following code, with its intended effects seen in Figure 25–1.

### Listing 25-1 Caption-Side Example

```
<HTML>
<HEAD>
<TITLE>Caption-Side Example</TITLE>
<STYLE>
TD {FONT-SIZE: X-LARGE}
</STYLE>
</HEAD>
<BODY>
<DIV ALIGN="CENTER">

<TABLE BORDER>
<H1 STYLE="CAPTION-SIDE: LEFT;">Table Caption to the Left</
H1>
<TR>
<TD>This text is contained within the table</TD>
</TR>
</TABLE>

<P>

<TABLE BORDER>
<H1 STYLE="CAPTION-SIDE: RIGHT;">Table Caption to the
Right</H1>
<TR>
<TD>This text is contained within the table</TD>
</TR>
</TABLE>

<P>
<TABLE BORDER>
<H1 STYLE="CAPTION-SIDE: TOP;">Table Caption to the Top</
H1>
<TR>
<TD>This text is contained within the table</TD>
</TR>
</TABLE>
```

**Listing 25-1 Caption-Side Example (continued)**

```

<P>
<TABLE BORDER>
<H1 STYLE="CAPTION-SIDE: BOTTOM;">Table Caption to the
Bottom</H1>
<TR>
<TD>This text is contained within the table</TD>
</TR>
</TABLE>

</BODY>
</HTML>

```

---

**Table Caption to the  
Left**

This text is contained within the table
--

This text is contained within the table
--

**Table Caption to the  
Right**

**Table Caption to the Top**

This text is contained within the table
---

This text is contained within the table
---

**Table Caption to the Bottom**

**Figure 25-1** The CAPTION-SIDE code example as it should be displayed.

The CAPTION-SIDE property provides greater flexibility to Web authors looking to create more sophisticated table displays.

As of yet, the CAPTION-SIDE property is not supported in any major browser.

# TABLE-LAYOUT

The introduction of the `TABLE-LAYOUT` property is designed to enable Web authors to explicitly lay out their tables according to whether or not the width should be dependant on the size of the content of the individual cells comprising the table. By expressing this property, Web authors gain more explicit control over how the table (and its contents) should appear on screen.

## TABLE-LAYOUT

Description: Controls how a table's width is displayed, using either a fixed or automatic value.

Media Group: Visual

CSS2 Family Type: Tables

Values:

- **AUTO** - Uses automatic table layout.
- **FIXED** - Uses a fixed table layout.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<TABLE STYLE="TABLE-LAYOUT: AUTO;">
<TR>
<TD>This text is in the table</TD>
</TR>
</TABLE>
```

The `TABLE-LAYOUT` property has three named values: **AUTO**, **FIXED** and **INHERIT**. With the **AUTO** property, the browser must first determine the size of contents of each cell and the resulting column in which they reside. Once that is calculated, the browser then proceeds to lay out the resulting table. This is the default value. This method is not terribly efficient, since the browser has to do a lot of calculation beforehand in order to present anything.

The **FIXED** value enables Web authors to set a maximum size for the table to be displayed, which necessarily sets an upper limit on what can be contained within the individual cells and columns of cells within the resulting table. In the case that some of the resulting content in the cells cannot be displayed fully, Web authors can use the `OVERFLOW` property to determine whether or not a viewer is able to see it in another fashion, perhaps by being able to scroll the contents into view.

The INHERIT value simply takes whatever parent value may have already been set for TABLE-LAYOUT.

The following code gives an idea to how TABLE-LAYOUT can be used:

### Listing 25-2 Table-Layout Example

```
<HTML>
<HEAD>
<TITLE>Table-Layout Example</TITLE>
</HEAD>
<BODY>

<TABLE STYLE="TABLE-LAYOUT: AUTO;" BORDER>
<TR>
<TD><IMG SRC="fokker.gif"></TD>
</TR>
</TABLE>

<P>

<TABLE STYLE="WIDTH: 1.5IN; HEIGHT: 1IN; OVERFLOW: AUTO;
TABLE-LAYOUT: FIXED;" BORDER>
<TR>
<TD><IMG SRC="fokker.gif"></TD>
</TR>
</TABLE>

</BODY>
</HTML>
```

---

In this example there are two tables displayed, both containing an image. In the first case, where the value AUTO is used, the table is sized to fit the size of the image. In the second case, the value FIXED has been applied to the table, in conjunction with specific HEIGHT, WIDTH and OVERFLOW property values. If the image is too big to fit in the table, a scrollbar will appear, courtesy of the AUTO value set to OVERFLOW.

As of yet, the TABLE-LAYOUT property is not supported in any of the major browsers.

# Controlling Table Borders

CSS2 provides two properties that give Web authors greater control over how the borders for tables are displayed: `BORDER-COLLAPSE` and `BORDER-SPACING`. Together, they allow Web authors to choose whether a border should appear “collapsed” or “separated”. A collapsed border is attached only to the outer border of the table. With a separated border, each cell can have a border value all of its own.

## ***BORDER-COLLAPSE***

Description: Selects the table border to be used.

Media Group: Visual

CSS2 Family Type: Tables

Values:

- `COLLAPSE` - Table is set to the collapsing borders model.
- `SEPARATE` - Table is set to the separated borders model.
- `INHERIT` - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

### **Listing 25-3 Border-Collapse**

```
<HTML>
<HEAD>
<TITLE>Border-Collapse</TITLE>
<STYLE>
TABLE {BORDER: OUTSET 15PT; BORDER-COLLAPSE: SEPARATE;
BORDER-SPACING: 15PT;}
TD {BORDER: INSET 15PT; FONT-SIZE: XX-LARGE;}
</STYLE>
</HEAD>
<BODY>
<TABLE>
<TR>
```

**Listing 25-3 Border-Collapse (continued)**

```
<TD>Cell 1</TD>
<TD>Cell 2</TD>
<TD>Cell 3</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

---

## ***BORDER-SPACING***

Description: Sets the distance separating adjacent cell borders. If one value is set, it is used for both height and width. If two values are used, the first is taken for horizontal spacing, while the second is used for vertical spacing.

Media Group: Visual

CSS2 Family Type: Tables

Values:

- *n [n]* - units measurement
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

**Listing 25-4 Border-Spacing**

```
<HTML>
<HEAD>
<TITLE>Border-Spacing</TITLE>
<STYLE>
TABLE {BORDER: OUTSET 5PT; BORDER-COLLAPSE: SEPARATE;
BORDER-SPACING: 10PT;}
TD {BORDER: INSET 5PT;}
</STYLE>
</HEAD>
<BODY>
<TABLE>
<TR>
```

**Listing 25-4 Border-Spacing (continued)**

```
<TD>Cell 1</TD>
<TD>Cell 2</TD>
<TD>Cell 3</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

`BORDER-COLLAPSE` can take any one of three named values: `COLLAPSE`, which sets the table to the standard collapsing borders model; `SEPARATE`, which sets the table to the separated borders model; and the `INHERIT` value, which takes on the same value as that set for the parent for `BORDER-COLLAPSE`. The most interesting value here is the `SEPARATE` value. When you choose the `SEPARATE` value, you must add a `BORDER-SPACING` value to your code. The `BORDER-SPACING` property most typically takes a basic standard of unit measurement value. If a single value is specified, the value is set to all borders, and if two are specified, the first value is taken for horizontal spacing, while the second is used as the vertical spacing value. It can also take the named value `INHERIT`, which takes on whatever value has been set to the parent for `BORDER-SPACING`.

The following example code shows both `BORDER-COLLAPSE` and `BORDER-SPACING` in action, the results of which can be seen in Figure 25–2.

**Listing 25-5 Border-Collapse and Border-Spacing Example**

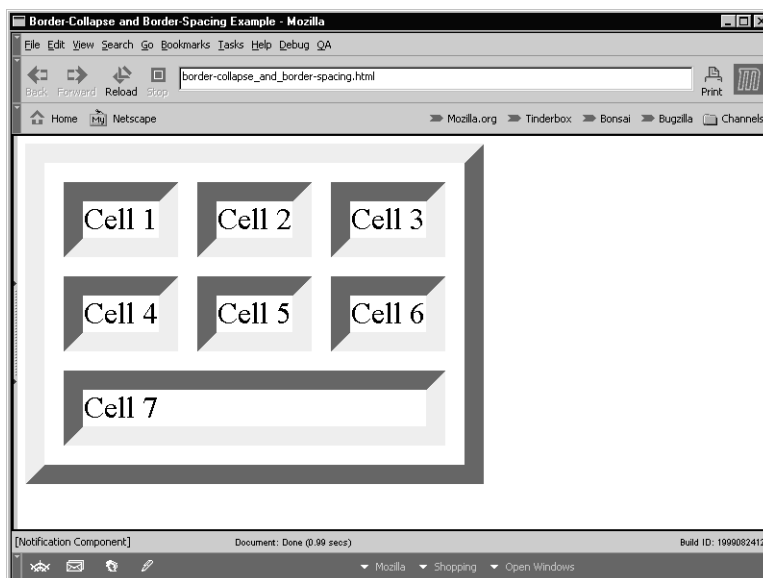
```
<HTML>
<HEAD>
<TITLE>Border-Collapse and Border-Spacing Example</TITLE>
<STYLE>
TABLE {BORDER: OUTSET 15PT; BORDER-COLLAPSE: SEPARATE;
BORDER-SPACING: 15PT;}
TD {BORDER: INSET 15PT; FONT-SIZE: XX-LARGE;}
</STYLE>
</HEAD>
<BODY>
```

### Listing 25-5 Border-Collapse and Border-Spacing Example

```

<TABLE>
<TR>
<TD>Cell 1</TD>
<TD>Cell 2</TD>
<TD>Cell 3</TD>
</TR>
<TR>
<TD>Cell 4</TD>
<TD>Cell 5</TD>
<TD>Cell 6</TD>
</TR>
<TR>
<TD COLSPAN="3">Cell 7</TD>
</TR>
</TABLE>
</BODY>
</HTML>

```



**Figure 25-2** Example code for the BORDER-COLLAPSE and BORDER-SPACING properties displayed.

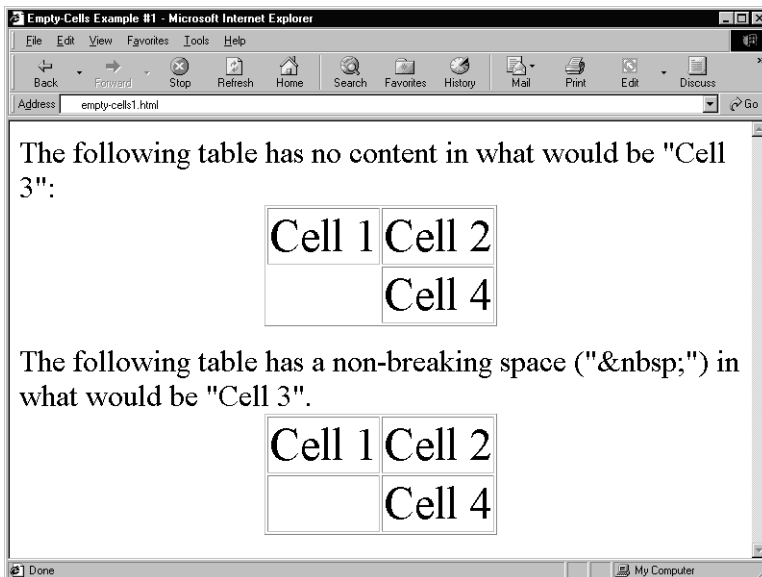


Note that each cell has its own border, and that the table has a border separate from those of its cells.

These properties are not yet supported in the major browsers.

## EMPTY-CELLS

There are often cases when a one or more table cells may contain no content whatsoever, but are still needed so that the table display remains intact (i.e., that the number of cells in a row or column equals the number assigned to the table as the whole). In a regular table, cells that contain no content appear “empty” and the border for the table (assuming one exists) seems to fill in and overlay the empty cell. Some Web authors have gotten around this by inserting a non-breaking space (“&nbsp;”) into the otherwise empty cell, so that it would display an actual cell containing no visible content. You can see the effects of both of these examples in the following illustration.



**Figure 25-3** The “empty-cell” effect illustrated.

The `EMPTY-CELLS` property is designed to let Web authors automatically specify how empty cells should appear in a given table.

## EMPTY-CELLS

Description: This property is used to control how table cells containing no content are to be displayed.

Media Group: Visual

CSS2 Family Type: Tables

Values:

- **SHOW** - A border is drawn around the empty cell (the default table value).
- **HIDE** - No border is drawn around the empty cell.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<TABLE STYLE="EMPTY-CELLS: HIDE;" BORDER>
<TR>
<TD>No border should be drawn around the adjacent
cell.</TD>
<TD></TD>
</TABLE>
```

The **EMPTY-CELLS** property can take any one of three named values: **SHOW**, which displays a full border around the empty cell; **HIDE**, which does not draw a border around the cell; and **INHERIT**, which takes whatever parent value has been set for **EMPTY-CELLS**.

The **SHOW** value is equivalent to adding the non-breaking space character within a table cell, and the **HIDE** is like having a completely empty cell, as displayed in the previous illustration. Using **EMPTY-CELLS**, Web authors no longer have to “fill in” each empty cell with a non-breaking space in order for the border around the cell to appear. The following code sample should produce table displays equivalent to those seen in the previous illustration.

### Listing 25-6 Empty-Cells Example #2

```
<HTML>
<HEAD>
<TITLE>Empty-Cells Example #2</TITLE>
<STYLE>
TD {FONT-SIZE: XX-LARGE;}
BODY {FONT-SIZE: X-LARGE;}
</STYLE>
</HEAD>
```

**Listing 25-6 Empty-Cells Example #2 (continued)**

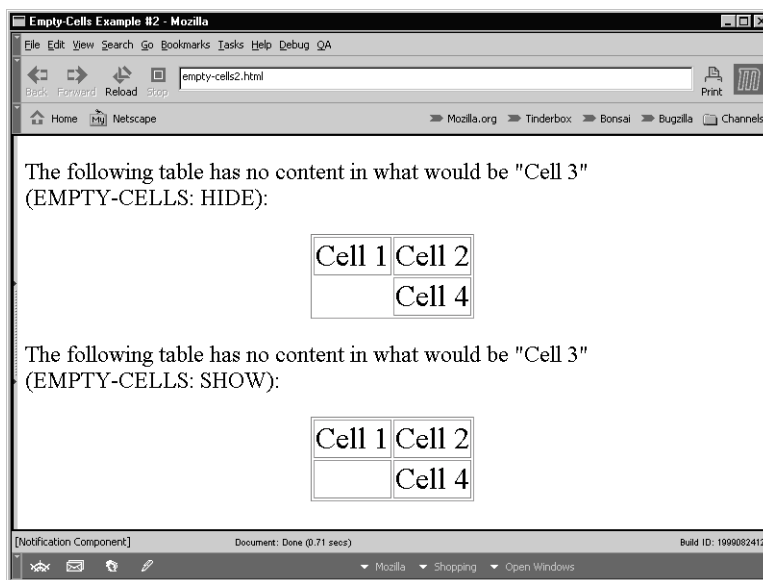
```
<BODY>
<P>
The following table has no content in what would be "Cell
3" (EMPTY-CELLS: HIDE):
<DIV ALIGN="CENTER">
<TABLE STYLE="EMPTY-CELLS: HIDE;" BORDER>
<TR>
<TD>Cell 1</TD>
<TD>Cell 2</TD>
</TR>
<TR>
<TD></TD>
<TD>Cell 4</TD>
</TR>
</TABLE>
</DIV>

<P>

The following table has no content in what would be "Cell
3" (EMPTY-CELLS: SHOW):
<DIV ALIGN="CENTER">
<TABLE STYLE="EMPTY-CELLS: SHOW;" BORDER>
<TR>
<TD>Cell 1</TD>
<TD>Cell 2</TD>
</TR>
<TR>
<TD></TD>
<TD>Cell 4</TD>
</TR>
</TABLE>
</DIV>
</BODY>
</HTML>
```

---

The `EMPTY-CELLS` property is not yet supported within any of the major browsers. It is supported in the Mozilla browser, however, so it will likely be available in Netscape Navigator 5.0 when it released.



**Figure 25-4** The effects of the EMPTY-CELLS property as seen within the Mozilla browser.

## SPEAK-HEADER

The **SPEAK-HEADER** property is actually an aural property specific to the table family. Its job is to speak aloud the headers in a table in the manner prescribed by the Web author. The Web can either specify that table headers be read out only once at the beginning of a list of associated cell values, or repeated along with each cell value with which they are associated.

### ***SPEAK-HEADER***

**Description:** Determines how a table header is spoken before individual cell values.

**Media Group:** Aural

**CSS2 Family Type:** Tables

**Values:**

- **ONCE** - The header value is only spoken once, at the beginning of a listing of cell values.

- **ALWAYS** - The header value is spoken before each associated cell value.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<HTML>
<STYLE>
TH {SPEAK-HEADER: ALWAYS}
</STYLE>
<BODY>
<TABLE>
<TR>
<TH>Fruits</TH>
<TH>Vegetables</TH>
</TR>
<TR>
<TD>Apple</TD>
<TD>Carrot</TD>
</TR>
</TABLE>
```

The **SPEAK-HEADER** property can take one of three named values: **ONCE**, in which case the header value is only said aloud one time at the beginning of a list of cell values; **ALWAYS**, which reads the header value before each associated cell value; and **INHERIT**, which takes the same value as that set for the property of the parent value set for **SPEAK-HEADER**.

This value is not supported in any major browser as yet.

# USER INTERFACE PROPERTIES

## Topics in this Chapter

- The CURSOR Property
- The OUTLINE Property
- The OUTLINE-WIDTH Property
- The OUTLINE-STYLE Property
- The OUTLINE-COLOR Property



# Chapter 26

The user interface properties are newly introduced under CSS2. These properties are designed to enable Web authors greater control over such things as the display of cursors on screen, and the “focus” outline that appears around such elements as buttons and form fields, to indicate that they are selected and ready for input.

There are a total of five new properties under the family of user interface properties:

- CURSOR
- OUTLINE
- OUTLINE-WIDTH
- OUTLINE-STYLE
- OUTLINE-COLOR

## The CURSOR Property

The CURSOR property is used to define the type of cursor to be displayed over a given Web element. The purpose of this property is to draw the attention of the user to the element. It can be applied to any element on a Web page: text, images or any other objects.

## CURSOR

Description: Defines the type of cursor to be displayed.

Media Group: Visual, Interactive

CSS2 Family Type: User interface

Values:

- *URL* - Points to a specific Web page's content. As the user passes over the link it will change to the selected cursor type.
- *AUTO* - Browser displays the type of cursor normally associated with the Web element.
- *CROSSHAIR* - A symbol resembling a "+" is displayed.
- *DEFAULT* - The default cursor type is displayed.
- *POINTER* - A pointer cursor (typically a small, upper-left pointing arrow) is displayed.
- *MOVE* - Cursor indicating that a Web element is to be moved.
- *E-RESIZE* | *NE-RESIZE* | *NW-RESIZE* | *N-RESIZE* | *SE-RESIZE* |
- *SW-RESIZE* | *S-RESIZE* | *W-RESIZE* - Cursor associated when an edge or corner can be moved.
- *TEXT* - Indicates that text can be selected. Typically rendered as an I-bar.
- *WAIT* - Indicates that the program is busy and that the user should be patient while the operation is completed.
- *HELP* - Online help is available for the object under the cursor. Typically rendered as a question mark with pointer or as a balloon.
- *INHERIT* - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
Take aim and fire! <A HREF="link.html"><IMG  
SRC="shooting_gallery.gif" STYLE="CURSOR:  
CROSSHAIR;"></A>
```

There are plenty of values available for the *CURSOR* property, enabling Web authors to choose from just about the whole range of cursor types available.

There are several *CURSOR* values designed to produce very specific cursor types. For example, the *CROSSHAIR* value is designed to display a symbol that resembles a "+". The *POINTER* value shows a typical "pointing" cursor, which is typically a small, upper-left pointing arrow.



Then there are a number of cursors normally used to indicate that an edge or corner can be moved. These cursor values are: E-RESIZE, NE-RESIZE, NW-RESIZE, N-RESIZE, SE-RESIZE, SW-RESIZE, S-RESIZE and W-RESIZE. The letter at the beginning of each of these values designates the cardinal direction of the type of edge or corner cursor to be displayed (“E” for east, “NW” for north west and so on).

Several cursor values are more dependent on the type of operating system being used. The MOVE value should display the type of cursor indicating a Web element can be moved (regardless of whether it can be or not). This type of cursor is commonly displayed as a thick double-line, which can be either vertically or horizontally aligned. The TEXT value should display the type of cursor that indicates text can be selected. This type of cursor is most typically rendered as an I-bar. The WAIT value should display the type of cursor commonly associated with a computer’s “busy-mode” cursor, indicating that the user should be patient while the operation the computer is working on is completed. A good example of this is the standard hourglass cursor. The HELP value should display the type of cursor associated with online help, most typically rendered as either a question mark with pointer or in a separate balloon.

If the AUTO value is used, the browser displays whatever type of cursor would normally be associated with the Web element. Similar to this is the DEFAULT value, which displays the default “pointing” cursor for the browser (most often some form of arrow).

A URL value can also be used with CURSOR, which must point to some sort of cursor resource (presumably a bitmap image of the cursor), which is then loaded by the browser and displayed. The URL value can be cascaded, so that if the first cursor image can’t be found, a back-up cursor can be. You can see an example of this in the following code snippet:

```
<H1 STYLE="CURSOR: URL("funkyhand.csr"),  
  URL("otherhand.cur"), TEXT;">Try Scrolling Over  
This Header and See What Happens!</H1>
```

Finally, there is the INHERIT value, which simply takes on whatever parent value that may have already been set for CURSOR.

When using this property, it is important to keep in mind that the type of cursor displayed depends entirely on the operating system on the computer being used by the viewer. The values that have been selected in the CSS2 specification are deliberately generic, though, so you can be assured that your viewers will definitely see something close to your intent.

The CURSOR property has not been implemented yet in either Internet Explorer or Netscape Navigator.

## The OUTLINE Sub-Family of Properties

Under CSS2, it is now possible for Web authors to create visual outlines around objects on a Web page, such as buttons, active form fields, image-maps and other objects, in order to make them stand out to viewers. The sub-family of OUTLINE properties is designed to add a border to these types of objects. These outlines do not take up any space on the page (in other words, they do not expand the box within which the element is contained), and they can be non-rectangular in shape (which is typical of most image-maps).

This is simply an extension of what is already present in most operating systems, in signaling to the viewer what is known as “focus” (i.e., that a particular element is currently selected). For example, if you are presented with a Web-based form while using a typical MS Windows computer, you will notice that a cursor will typically appear in the field in which you can currently type, and if you hit the TAB button on your keyboard a few times, you will notice that the cursor jumps to the next field down. If you keep doing this and a button, checkbox or radio button is selected, you will discover that the selected object is usually surrounded by a thin border. This is exactly the type of border that can be explicitly set using the set of OUTLINE properties.

The sub-family consists of four different properties: OUTLINE-WIDTH, OUTLINE-STYLE, OUTLINE-COLOR and the shortcut property OUTLINE. These properties are meant to be set globally across a Web page, not to a specific element, so these properties should be set within the header of a Web page.

## The OUTLINE-WIDTH Property

The OUTLINE-WIDTH property is used to set a width for the border that appears around an active object.

### **OUTLINE-WIDTH**

Description: Sets a value for the border width indicating focus for such things as buttons or form fields.

Media Group: Visual, Interactive

CSS2 Family Type: User interface

Values:

- `n [ n n n ] measurement_units`
- Sets the thickness of the border.
- If 1 value is present, all borders are set to the numerical value specified
- If 2 values are present, the top/bottom and side borders take the numerical values specified
- If 3 values are present, the top, right and left, then bottom takes the numerical values specified
- If 4 values are present, the top, right, bottom then left borders take the numerical values specified
- `THIN | MEDIUM | THICK` - Sets the relative thickness of the outline.
- `INHERIT` - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<STYLE>
BUTTONSTYLE {OUTLINE-WIDTH: MEDIUM;}
</STYLE>
```

The `OUTLINE-WIDTH` property can take a variety of values, both numeric and named. Its numeric values mirror those that can be set for such properties as `MARGIN` or `BORDER-WIDTH`, in that it can take from one to four separate numerical values that set the width of the outline. The default value is pixels, so if no measurement unit is specified, the value given is assumed to be a pixel value. If multiple values are to be set using other units of measurement, each value, in turn, must be set with a specific unit of measurement — one measurement unit value *does not* automatically apply to all. If a single numerical value is present, all borders are set to that value. If two numerical values are present, the top/bottom and side borders take on those values, respectively. If three values are present, the top, right and left, and then the bottom take the numerical values specified. Finally, if four values are present, the top, right, bottom then left borders (i.e., clockwise from the top) take the numerical values specified. When you specify these values, they must all be separated by a space, as can be seen in the following code snippet:

```
<HEAD>
<STYLE>
BUTTONSTYLE {OUTLINE-WIDTH: 10 15 5 20;}
</STYLE>
</HEAD>
```

OUTLINE-WIDTH can also take a number of named values, such as the following pre-set values: THIN, MEDIUM and THICK. These set relative thickness values for the outline. It is also possible to set a different named value for all four sides of the outline by arranging them like their numeric equivalents, following the same ordering structure, as you can see from the following snippet of code:

```
<HEAD>
<STYLE>
BUTTONSTYLE {OUTLINE-WIDTH: THIN MEDIUM THIN
THICK; }
</STYLE>
</HEAD>
```

OUTLINE-WIDTH can also take named value INHERIT, which takes the same value as that set for the parent value for OUTLINE-WIDTH, if it has already been set.

This property is not yet supported within either Internet Explorer or Netscape Navigator.

## The OUTLINE-STYLE Property

The OUTLINE-STYLE property sets the type of outline to be displayed around a given object on a Web page. This property copies the values used for the BORDER-STYLE property.

### **OUTLINE-STYLE**

Description: Sets a value for the border style indicating focus for such things as buttons or form fields.

Media Group: Visual, Interactive

CSS2 Family Type: User interface

Values:

- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.

- **OUTSET** - A 3D outset line is displayed.
- **RIDGE** - A 3D ridged line is displayed.
- **SOLID** - A solid border line is displayed.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<STYLE>
BUTTONSTYLE {OUTLINE-STYLE: DOUBLE;}
</STYLE>
```

There are a number of 2D and 3D outline types available for **OUTLINE-STYLE**. The 2D outlines are created by the values **DASHED**, **DOTTED**, **DOUBLE** and **SOLID**, while the 3D outlines are created by the **INSET**, **OUTSET**, **GROOVE** and **RIDGED** values. For the 2D outline types, the **DASHED** value displays an outline made up for dashes, the **DOTTED** value displays an outline made from small dots, the **DOUBLE** value creates an outline made from a double-line, and **SOLID** displays a solid outline style. For the 3D outline types, **INSET** creates a outline that appears recessed, while **OUTSET** creates a outline that appears the opposite of **INSET**. **GROOVE** displays a 3D grooved outline, and **RIDGE** creates a 3D ridged outline. The value **NONE** ensures that no outline is displayed. The following code snippet would display a prominent 3D groove that would appear around whatever button is currently selected on the Web page:

```
<HEAD>
<STYLE>
BUTTONSTYLE {OUTLINE-STYLE: GROOVE;}
</STYLE>
</HEAD>
```

**OUTLINE-STYLE** can also take the named value **INHERIT**, which takes the same value as that set for the parent value for **OUTLINE-STYLE**, if it has already been set.

This property is not yet supported within either Netscape Navigator or Internet Explorer.

## The OUTLINE-COLOR Property

The `OUTLINE-COLOR` property sets the color for the outline to be displayed around a given object on a Web page that can currently accept input from the user. This property copies the values used for the `BORDER-COLOR` property.

### OUTLINE-COLOR

Description: Sets a color value for the border indicating focus for such things as buttons or form fields.

Media Group: Visual, Interactive

CSS2 Family Type: User interface

Values:

- *color name* or *numerical color value* - Sets the color for the border.
- `INVERT` - Reverses the color currently being used in the background behind the user interface element.
- `INHERIT` - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<STYLE>
BUTTONSTYLE {OUTLINE-COLOR: INVERT;}
</STYLE>
```

The `OUTLINE-COLOR` property, like all other color CSS properties, allows Web authors to set a color value in a variety of ways. For example, it can take a standard color name (such as “YELLOW”) or a hexadecimal color value (such as “#FFFF00”, which also produces yellow), both of which are standard ways of setting color values in HTML. In CSS, you can also specify color using a “compressed” hexadecimal color value (such as “#00F”, which is blue). You can also use a three-digit RGB (“Red Green Blue”) color value (where `RGB(0,255,0)` is green), and a three-digit RGB percentage color value (where `RGB(100%,100%,100%)` is white). (For more information on these latter ways of specifying color, please see the “Color Formats” section in the CSS Units chapter).

The following code snippet shows how you could set a thick green border color for the selected buttons on a Web page:

```
<HEAD>
<STYLE>
```

```
BUTTONSTYLE {OUTLINE-COLOR: GREEN; OUTLINE-WIDTH:  
THICK; }  
</STYLE>  
</HEAD>
```

OUTLINE-COLOR can also take the named value INVERT, which uses the color opposite to that currently used by the background. So, if the background is white, a black outline will be displayed. This is a fairly common programmer's trick, designed to ensure that the outline will always be visible no matter the circumstance.

OUTLINE-COLOR can also take named value INHERIT, which takes the same value as that set for the parent value for OUTLINE-COLOR, if it has already been set.

This property is not yet supported within either Internet Explorer or Netscape Navigator.

## OUTLINE

The OUTLINE property is a “shortcut” property that allows Web authors to set all of the properties normally reserved for the other outline sub-family of properties. In this fashion, it is modeled on another shortcut property, BORDER.

### OUTLINE

Description: A shortcut property that sets a value for the border indicating focus for such things as buttons or form fields.

Media Group: Visual, Interactive

CSS2 Family Type: User interface

Values:

- `n [ n n n ] measurement_units`
- Sets the thickness of the outline.
- If 1 value is present, all outline are set to the numerical value specified
- If 2 values are present, the top/bottom and side outlines take the numerical values specified
- If 3 values are present, the top, right and left, then bottom outlines takes the numerical values specified

- If 4 values are present, the top, right, bottom then left outlines take the numerical values specified
- THIN | MEDIUM | THICK - Sets the relative thickness of the outline.
- *color name* or *numerical color value* - Sets the color for the outline.
- INVERT - Reverses the color currently being used in the background behind the user interface element.
- HIDDEN - Same effect as NONE. It is designed to suppress outline-displays when used in tables.
- DASHED - A dashed outline is displayed.
- DOTTED - A dotted outline is displayed.
- DOUBLE - A double-line outline is displayed.
- INSET - A 3D inset outline is displayed.
- GROOVE - A 3D grooved outline is displayed.
- NONE - No outline is displayed, no matter what OUTLINE-WIDTH value is present.
- OUTSET - A 3D outset outline is displayed.
- RIDGE - A 3D ridged outline is displayed.
- SOLID - A solid outline is displayed.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<STYLE>
BUTTONSTYLE {OUTLINE: THICK RED GROOVE;}
</STYLE>
```

Using OUTLINE, you can use all of the border styles you could normally set through the OUTLINE-STYLE property, set the width normally reserved for the OUTLINE-WIDTH property, and determine the color normally reserved for the OUTLINE-COLOR property (see these properties for a full explanation of the relevant values). Using OUTLINE, it is possible to set all of the outline values for the selected elements on a Web page quickly and easily, as the following code snippet, which combines values found in the OUTLINE-STYLE, OUTLINE-WIDTH and OUTLINE-COLOR properties, shows:

```
<HEAD>
<STYLE>
BUTTONSTYLE {OUTLINE: THICK RED GROOVE;}
</STYLE>
</HEAD>
```



This property is not yet supported within either Internet Explorer or Netscape Navigator.

# AURAL STYLE SHEET PROPERTIES

## Topics in this Chapter

- The VOLUME Property
- The SPEAK Property
- The PAUSE-BEFORE and PAUSE-AFTER Properties
- The PAUSE Property
- The CUE-BEFORE and CUE-AFTER Properties
- The CUE Property
- The PLAY-DURING Property
- The AZIMUTH Property
- The ELEVATION Property
- The SPEECH-RATE Property
- The VOICE-FAMILY Property
- The PITCH Property
- The PITCH-RANGE Property
- The STRESS Property
- The RICHNESS Property
- The SPEAK-PUNCTUATION Property
- The SPEAK-NUMERAL Property



# Chapter 27

One of the most significant innovations in the CSS2 specification is the introduction of aural style sheets. With aural style sheets, you can now set the way a Web page should *sound*.

More specifically, aural style sheet properties are designed to tell the Web browser how it should read aloud the text on a Web page. Using the various aural style sheet properties available under CSS2, Web masters can set the type of voice to be played, where the sound should be placed within the stereo sound field, set pauses and cues, add sound effects, move the sound around in the soundscape and even determine whether the voice should be female or male. Using the aural style sheet properties, you can easily make the Web browser your stage and your text the players upon it.

The intent of these properties is to turn your ordinary browser into a “talking browser”, allowing the “viewer” to hear the textual content of a Web page. This frees the user from having to necessarily be in front of a computer screen when surfing the Web, for example. Talking browsers offer future Web users the possibility of surfing from just about anywhere. A “Web-Walkman” may appear in the not-too-distant future. “Talking browsers” are also designed to aid the visually impaired, opening up the Web to a wider range of potential surfers.

There are 19 aural style sheet properties in the CSS2 specification:

- VOLUME
- SPEAK

- PAUSE-BEFORE
- PAUSE-AFTER
- PAUSE
- CUE-BEFORE
- CUE-AFTER
- CUE
- PLAY-DURING
- AZIMUTH
- ELEVATION
- SPEECH-RATE
- VOICE-FAMILY
- PITCH
- PITCH-RANGE
- STRESS
- RICHNESS
- SPEAK-PUNCTUATION
- SPEAK-NUMERAL

Whenever you are using sound of any sort, you should remember that not all computers are equipped with sound cards. While the majority of multimedia computers intended for the consumer market are so equipped, they are still something of a rarity in most business environments. It is important for Web authors not to rely upon spoken cues to get the point of a Web page across to the viewer. Having said that, it is likely that these aural style sheet properties will not adversely affect the function of the Web page for those Web browsers not equipped for sound.

None of these properties are as yet implemented in any major browser, but it is well worth understanding these properties for the day they are.

## The VOLUME Property

The `VOLUME` property behaves much like the “volume” knob on your radio or television — it regulates the “loudness” level of the sound being played back. The aural style sheet `VOLUME` property is designed to control the sound level of the speech being played over the computer’s speakers. It is a very versatile property, allowing Web authors complete control over the sound level.

## VOLUME

Description: Sets the level for the dynamic range of the sound being played (i.e., its “volume”).

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Volume properties)

Values:

- ***n*** - range is between 0 (near silence) and 100 (full volume)
- **% value** - range is between 0 (near silence) and 100 (full volume)
- **SILENT** - Volume is turned off
- **X-SOFT** - Very low volume level, equivalent to the numerical value “0” (which does not imply silence)
- **SOFT** - Low volume level, equivalent to the numerical value “25”
- **MEDIUM** - Medium volume level, equivalent to the numerical value “50”
- **LOUD** - Moderately loud volume level, equivalent to the numerical value “75”
- **X-LOUD** - Very loud volume level, equivalent to the numerical value “100”
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="VOLUME: MEDIUM;">
I'm the big gruff bear.
<IMG SRC="bear.gif">
<B STYLE="VOLUME: LOUD;">Roar!</B>
</P>
```

The VOLUME property can take a wide range of values, covering the complete spectrum from near-quiet to deafening loudness. It can take a numeric value that ranges from “0” to “100”, where “0” is equivalent to near-silence, and “100” is the loudest possible sound the Web browser can produce. The VOLUME property can also take a percentage value, where a value of “0%” is the same as near-silence (or “0”), and “100%” is full volume (or “100” in numerical terms).

You may ask yourself why the value “0” is equivalent to “near-quiet” instead of complete silence. This is because the CSS2 specification does not set the value of “0” to equal no sound at all — it merely means the lowest pos-

sible audible setting available that still produces sound. Instead, there is a named value, called **SILENT**, which takes care of that.

**VOLUME** also has six named properties that correspond to preset volume levels: **SILENT**, **X-SOFT**, **SOFT**, **MEDIUM**, **LOUD** and **X-LOUD**. These values work in very much the same way the **XX-SMALL** to **XX-LARGE** values work with the **FONT-SIZE** property, providing a range of set values applied to the sound level produced by the browser when reading the page aloud. The **SILENT** value is equivalent to no sound at all — for reasons already discussed, the value of **SILENT** is in fact “lower” than the value “0”. **X-SOFT** is the equivalent to a very low volume level, and it is this sound level that is equivalent to the numerical value “0”. **SOFT** is a low volume level equivalent to the numerical value “25”. **MEDIUM** is a medium volume level which is the same as the numerical value “50”. Note that the **MEDIUM** level means that it is exactly half as loud a sound as the Web browser is able to produce. The **LOUD** value is equivalent to the numerical value “75”, and, finally, the **X-LOUD** is the loudest sound volume level possible, equivalent to the numerical value “100”.

**VOLUME** can also take the named value **INHERIT**, which in this case means that the text takes on whatever the parent sound level is set.

Volume, when applied to speech, is a highly variable thing. The actual sound level in speech always varies (unless the voice is speaking in a monotone). In any speech, there will be peaks and valleys, which can be used for emphasis by the speaker. The specification takes this into account, and allows for some latitude in the interpretation of these values by the browser. Keep in mind also that there could also be difference in the actual sound level produced by different browsers on different computers. A value of “50” may not necessarily be equivalent to the same value on a different computer — especially since most computers that have sound cards often have different sound levels set by their users. If and when this particular CSS2 property is widely implemented by the major browsers, keep in mind that what may seem like a **MEDIUM** sound level to you may be **LOUD** on a different system. This is still a very useful property for setting the emphasis of the speaking voice. The following simple code provides an example of this:

### Listing 27-1 Volume Example

```
<HTML>
<HEAD>
<TITLE>Volume Example</TITLE>
</HEAD>
```

**Listing 27-1 Volume Example (continued)**

```
<BODY>
<P STYLE="VOLUME: MEDIUM;">
I'm the big gruff bear.
<IMG SRC="bear.gif">
<B STYLE="VOLUME: LOUD;">Roar!</B>
</P>
</BODY>
</HTML>
```

---

This property is not implemented in any of the major browsers as yet.

## The SPEAK Property

The SPEAK property is used to set those parts of the Web page whose text is to be spoken aloud by the browser. This particular property can be used to tell the browser not only what parts to speak, but also how it should speak them.

### ***SPEAK***

Description: Specifies whether text is to be read out loud by the browser.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Speaking Properties)

Values:

- NONE - Suppresses the text being read aloud.
- NORMAL - Reads text aloud using the default pronunciation rules.
- SPELL-OUT - Spells out text one letter at a time.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="SPEAK: NORMAL">
<IMG SRC="mom.gif">
Don't tell the baby that she's about to have a
<I STYLE="SPEAK: SPELL-OUT">bath</I>.
<IMG SRC="dad.gif">
</P>
```

The `SPEAK` property has four separate named values: `NONE`, `NORMAL`, `SPELL-OUT` and `INHERIT`. The value `NONE` does what you would expect — it tells the browser not to speak aloud the selected part of the Web site. You would use the `NONE` property for those sections of a Web page — for whatever reason — you would not want to have read aloud. An ideal way of using this value would be in conjunction with the `<DIV>` tag, making it easy to set up distinct areas of the Web page text that are “silent”. The `NORMAL` value is the default browser speak setting, which should be a normal speaking voice. The `SPELL-OUT` value is to be used in those situations where you want to spell out the individual letters that comprise a word. This is particularly useful for such acronyms as “U.S.A.”, which might otherwise be rendered aloud by the browser as “usa”. It could also be used to deliberately spell out the letters in a word for any another reason, as the following Web code example displays:

#### Listing 27-2 Speak Example

```
<HTML>
<HEAD>
<TITLE>Speak Example</TITLE>
</HEAD>
<BODY>
<P STYLE="SPEAK: NORMAL">
<IMG SRC="mom.gif">
Don't tell the baby that she's about to have a <I
STYLE="SPEAK: SPELL-OUT">bath</I>.
<IMG SRC="dad.gif">
</P>
</BODY>
</HTML>
```

---

The fourth property, `INHERIT`, takes on whatever the parent value is set to for the `SPEAK` property. For example, if the previous example code did not explicitly close off of the `SPELL-OUT` value, `INHERIT` could be used in a subsequent paragraph to return the following text to the parent `SPEAK: NORMAL` setting embedded in the `<BODY>` tag.

This property is not yet implemented in any of the major Web browsers.



## The Pause Sub-Set of Properties

The sub-set of pause-related properties is designed to stop speech briefly immediately before or after a specific Web element, lending emphasis to the sentence being spoken by the Web browser. There are three properties in this sub-set: PAUSE-BEFORE, PAUSE-AFTER and PAUSE. The PAUSE-BEFORE property is designed to stop speech briefly *before* a specific Web element, and the PAUSE-AFTER property does the same, but instead stops speech immediately *after* a given Web element. The PAUSE property is a shortcut property that enables Web authors to simply set a pause before and/or after a given Web element.

This particular set of aural sub-properties is meant to let Web authors better imitate the natural rhythm of human speech, and to lend further emphasis to the text being read by the browser.

### PAUSE-BEFORE

Description: Sets a value for a break in speech occurring immediately before an element.

Media Group: Aural

CSS2 Family Type: Aural style sheets

Values:

- *n* - Time value for a break in speech expressed in either milliseconds or seconds.
- % - Percentage value expressed in relation to the “speed” of the text being read as set by the SPEECH-RATE property.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
Need a dramatic pause?  
<P STYLE="PAUSE-BEFORE: 50MS;">You just got one</P>
```

### PAUSE-AFTER

Description: Sets a value for a break in speech occurring immediately after an element.

Media Group: Aural

CSS2 Family Type: Aural style sheets

Values:

- *n* - Time value for a break in speech expressed in either milliseconds or seconds.
- % - Percentage value expressed in relation to the “speed” of the text being read as set by the SPEECH-RATE property.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="PAUSE-AFTER: 50MS;">Need a dramatic
pause?</P> You just got one.
```

## PAUSE

Description: A shortcut property that sets a value for a break in speech. The value(s) set is applied to the beginning and end of an element.

Media Group: Aural

CSS2 Family Type: Paged media

Values:

- *n [n]* - Time value for a break in speech expressed in either milliseconds or seconds.
- % - Percentage value expressed in relation to the “speed” of the text being read as set by the SPEECH-RATE property.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P>The title of this play is:</P>
<H1 STYLE="PAUSE: 1S;">The Tempest</H1>
<P>by William Shakespeare</P>
```

All three properties can take any of three different values: a numerical value representing the time of the pause, a percentage value set relative to the rate at which the text is being read, and the ubiquitous CSS2 named value INHERIT.

The most straightforward setting for this is the numeric time value you can set with these properties. Simply set a numerical value, and then set whether or not this value should be expressed in milliseconds (MS) or seconds (S). In order to set a half-second pause value, you would use a value of 500 milliseconds (1,000 milliseconds go into 1 second), or you can simply state it a

decimal fraction of a second. The following code shows how you can use a numeric value with the PAUSE-BEFORE and PAUSE-AFTER properties:

### Listing 27-3 Pause Properties Example 1

```
<HTML>
<HEAD>
<TITLE>Pause Properties Example 1</TITLE>
</HEAD>
<BODY>
Need a dramatic pause?
<P STYLE="PAUSE-BEFORE: 500MS;">You just got one.</P>
<P STYLE="PAUSE-AFTER: 0.5S;">Need a dramatic pause?</P>
You just got one.
</BODY>
</HTML>
```

---

With the PAUSE-BEFORE example, a half-second pause is associated with the paragraph tag (“<P>”), so the pause occurs immediately before the presence of the tag. With the PAUSE-AFTER example, a half-second pause appears at the close of the paragraph tag it is associated with (“</P>”).

Things are slightly different with the PAUSE property. It can take two separate numeric values. If two values are specified, the first pause value is applied before the Web element, and the second value is applied after. If only one value is specified (as in the following code example), the pause value is applied both before *and* after the Web element.

### Listing 27-4 Pause Properties Example 2

```
<HTML>
<HEAD>
<TITLE>Pause Properties Example 2</TITLE>
</HEAD>
<BODY>
<P>The title of this play is:</P>
<H1 STYLE="PAUSE: 1S;">The Tempest</H1>
<P>by William Shakespeare</P>
</BODY>
</HTML>
```

---

In this example, a one second pause is inserted immediately before and after the text in the header (“The Tempest”) is read aloud.

The pause properties can also take a percentage value, which relates to the speed at which the text is being read aloud by the browser. This value is set in relation to the `SPEECH-RATE` property (covered later in this chapter), or the default value for it used by the browser if it is not explicitly set. When a pause value is expressed in percentage terms, the length of the pause is in fact determined by the length of time it takes for a single word to be spoken. So if `SPEECH-RATE` is set at a extremely pedestrian 60 words per second — which means that one word is spoken every second — then a pause value set to 100% means that you will end up with a one second pause. The following code shows how you can get different lengths of pauses depending on the value of the `SPEECH-RATE` property.

### Listing 27-5 Pause Properties Example 3

```
<HTML>
<HEAD>
<TITLE>Pause Properties Example 3</TITLE>
</HEAD>
<BODY STYLE="SPEECH-RATE: 120";>
In this section of the Web page the speech is rendered at
120 words per minute.
<P STYLE="PAUSE-BEFORE: 100%";>The pause before this text
lasts for half a second.</P>
<P STYLE="PAUSE-BEFORE: 200%";>The pause before this text
lasts for one second.</P>
<P STYLE="PAUSE-BEFORE: 50%";>The pause before this text
lasts for a quarter of a second.</P>
<DIV STYLE="SPEECH-RATE: 60";>
In this section of the Web page the speech is rendered at a
very slow 60 words per minute.
<P STYLE="PAUSE-BEFORE: 100%";>The pause before this text
lasts for one second.</P>
<P STYLE="PAUSE-BEFORE: 200%";>The pause before this text
lasts for two seconds.</P>
<P STYLE="PAUSE-BEFORE: 50%";>The pause before this text
lasts for half a second</P>
</DIV>
</BODY>
</HTML>
```

---

Notice how in this last example, despite having the same percentage values, the length of the pauses are quite different, depending wholly on the value to which `SPEECH-RATE` has been set.

Finally, the pause properties can also take the named value `INHERIT`, which derives its value from the parent value set for any of the pause properties. If the `PAUSE-BEFORE` value for `<BODY>` is set to one second, then the declaration `P {PAUSE-BEFORE: INHERIT}` would add a one second pause to the beginning of each paragraph.

As of yet, `PAUSE`, `PAUSE-BEFORE` and `PAUSE-AFTER` are not implemented by any major browser.

## The Cue Sub-Set of Properties

Just as there is a sub-set of pause-related properties, so too is there a sub-set of cue-related properties. In this case, however, instead of attaching a pause value to a Web element, the cue essentially adds the equivalent of a “sound effect”. If you wanted to add the sound of a creaking door, a doorbell, a ghostly wail or anything else you can think of, you can do so using the set of cue properties. There are three properties in this sub-set of properties: `CUE-BEFORE`, `CUE-AFTER` and `CUE`. The `CUE-BEFORE` property is designed to add a sound effect *before* a specific Web element, and the `CUE-AFTER` property adds a sound effect *after* a given Web element. The `CUE` property is a shortcut property that enables Web authors to simply set a sound effect before and/or after a given Web element. Note: when a cue property is used, the sound appears either before or after the text being read by the browser, not along with it.

### ***CUE-BEFORE***

Description: Adds a sound to be played immediately *before* a Web element. Good for aurally distinguishing significant parts on a Web page.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Cue properties)

Values:

- `URL(sound)` - Points to a sound file in a format supported by the browser.
- `NONE` - No sound is played.

- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="CUE-BEFORE: BONG.AU;">Hamlet: Act I,  
Scene I</H1>
```

## CUE-AFTER

Description: Adds a sound to be played immediately *after* a Web element. Good for aurally distinguishing significant parts on a Web page.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Cue properties)

Values:

- URL(sound) - Points to a sound file in a format supported by the browser.
- NONE - No sound is played.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
You can get there by <A HREF="link.html"  
STYLE="CUE-AFTER; CLICK.WAV">Clicking here</A>.
```

## CUE

Description: Adds a sound to a Web element. If a single value is present, the sound is played before and after the element; if two are present, the first sound is played before and the second sound after the element.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Cue Properties)

Values:

- URL(sound) - Points to a sound file in a format supported by the browser.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
A sound appears before this <A HREF="link.html"  
STYLE="CUE: BADA-BING.AIFF;">link</A>.  
<P>
```

```
A sound appears before and after this
<A HREF="link.html" STYLE="BADA-BING.AIFF BADA-
BOOM.AIFF;">link</A>.
```

The cue properties can take any one of two different values: a URL that points to a sound file the browser is capable of playing back, and the named property INHERIT.

The CSS2 specification do not standardize the types of sound files that any particular browser must be capable of playing back, therefore the implementation of valid sound file formats will likely depend on the browser being used.

The implementation of sound has been built into browsers for several years now (since version 3.0 of each respective browser), and through the use of various plug-ins, just about any particular type of sound file can be played back through a browser. Netscape Navigator recognizes the following sound file formats: .WAV, .AU, .MID/.MIDI and .AIFF/.AIF. Microsoft Internet Explorer's audio player supports the following: .AU, .SND, .MID/.MIDI, .WAV, .AIFF/.AIF and .AIFC sound file formats. The sound file formats the two have in common, therefore, are .AIFF, .AU, .MID/.MIDI, and .WAV. The majority of these sound file formats are digitally-recorded sound file formats that can be derived from any recorded source, while the .MID/.MIDI file format consists of coded information designed to play synthesized sound through a sound card capable of playing the file format. The following is a simple table that covers the native support for these sound file formats in the two major browsers:

**Table 27-1 "Native" Sound File Formats Supported by Netscape Navigator and Internet Explorer since Version 3.0**

	<i>Netscape Navigator</i>	<i>Internet Explorer</i>
.AU	✓	✓
.AIF/.AIFF	✓	✓
.AIFC		✓
.MID/.MIDI	✓	
.SND		✓
.WAV	✓	✓

These file formats can be considered reasonably safe to use in code, if only because they have been implemented for so long in the two major browsers. In some cases, these sound file formats go back a dozen or more years, and were originally designed for a specific computing platform. Here's a quick summary of the sound file formats commonly used on the Web:

- .AIFF/.AIF - (Audio Interchange File Format) is a popular sound file format common to several operating systems.
- .AIFC - a sound file format that is essentially the same as .AIFF/.AIF, but compressed.
- .AU - a sound file format first used on SUN/NeXT computers, but which has become popular on other, more common operating systems.
- .SND - a sound file format used primarily on Macintosh computers.
- .MIDI/.MID - a sound file format understood by sound applications for both Macintosh and MS-Windows computers.
- .WAV - a sound format used primarily by MS-Windows computers. Of all of the sound file formats listed here, it is the most-likely sound file format you will readily find on the Web.

All of these sound file formats have been around for a number of years, but they are rapidly being superceded by two popular sound file formats that have grown in popularity on the Web: the RealAudio sound file format (.RA or .RAM) and the MP3 (.MP3) sound file format. At the time of writing, neither were supported directly in a minimum install of either Netscape Navigator or Internet Explorer, although either or both formats are supported by various add-ons, often offered as optional install features for either browser. Despite the popularity of these relative newcomers, it is probably a wiser course of action for the moment to assume that not all users will be capable of playing these sound file types (although it is safe to say that these two major formats will likely become “native” sound file formats within the major browsers within a short period of time). All of the example code displayed here uses “safe” sound file formats.

If you wanted to add a sound immediately prior to a Web element, in order to lend the following text some emphasis or to “set a mood”, you would use the CUE-BEFORE property. To add a sound immediately after a Web element, you would use the CUE-AFTER property instead. The shortcut property CUE works in much the same way as the PAUSE property, in that it can take two URL values. If two values are specified, the first sound file is played before the Web element, and the second sound file reference is played after it. If



only one sound file is specified, the same sound is played before *and* after the Web element.

The following code shows all three properties in action:

### Listing 27-6 Cue Properties Example

```
<HTML>
<HEAD>
<TITLE>Cue Properties Example</TITLE>
</HEAD>
<BODY>
<H1 STYLE="CUE-BEFORE: BONG.AU;">Hamlet: Act I, Scene I
</H1>
You can get there by <A HREF="link.html" STYLE="CUE-AFTER;
CLICK.WAV">Clicking here</A>.
<P>
A single sound appears before and after this
<A HREF="link.html" STYLE="CUE: BADA-BING.AIFF;">link</A>.
<P>
Two different sounds appear before and after this
<A HREF="link.html" STYLE="CUE: BADA-BING.AIFF BADA-
BOOM.AIFF;">link</A>.
</BODY>
</HTML>
```

---

In the CUE-BEFORE example, the sound “BONG.AU” is played immediately before the text contained in the header. With the CUE-AFTER example, the sound is played at the end of the link. There are also two examples of the CUE property in action. With the first, only a single sound file (“BADA-BING.AIFF”) is specified, so it is played both before and after the link. In the second, two sound files are specified, so the first sound file (“BADA-BING”) is played before the link, and the second (“BADA-BOOM”) is played after it.

The cue properties can also take on the named INHERIT value, so that if a parent value is set for a particular sound, any Web element can call it by using the INHERIT value.

None of these properties have been implemented yet in either Netscape Navigator or Internet Explorer.

## The PLAY-DURING Property

The **PLAY-DURING** property is used to set a sound file to play “behind” specified text on a Web page. In this way, you can add a sound effect or music that is played continuously behind your text as it is being read by the browser. It differs from the cue sub-set of properties in that it does not interrupt the flow of the text as it is read, but instead is designed to be “background” sound.

### **PLAY-DURING**

Description: Sets a sound file to be “behind” a particular Web element as it is read.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Mixing Properties)

Values:

- **URL(sound)** - Points to a sound file in a format supported by the browser.
- **MIX** - Overlays the specified sound with any other sound values that may be present.
- **REPEAT** - Keeps repeating the sound for the duration that the Web element is present.
- **AUTO** - Any parent sound present continues to play (instead of simply being restarted) once the selected **PLAY-DURING** element has finished.
- **NONE** - No sound; if a parent sound exists, it too is turned off.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<BODY STYLE="PLAY-DURING: WINDYOUTDOORS.WAV;">
Hamlet: Whither wilt thou lead me? Speak! I'll go
no further.
<P STYLE="PLAY-DURING: GHOSTLY.WAV MIX REPEAT;">
Ghost: Mark me.
</P>
Hamlet: I will.
<P STYLE="PLAY-DURING: GHOSTLY.WAV MIX REPEAT;">
Ghost: My hour is almost come, when I to
sulph'rous and tormenting flames must render up
myself.
</BODY>
```

Like the cue sub-set of properties, the `PLAY-DURING` property takes a URL to specify the name of the sound file to be played. The same browser restrictions regarding the types of sound files that can be played under the cue properties apply to the `PLAY-DURING` property.

In addition to the URL used to specify the sound file, `PLAY-DURING` can also take the following named values: `MIX`, `REPEAT`, `AUTO`, `NONE` and `INHERIT`. The `MIX` value is used to “mix” or add-in any sound from a previously-set `PLAY-DURING` sound. Thus, you can have multiple background sounds playing at once. The `REPEAT` value is designed to play a sound file multiple times under a given selection of text. You would use this value when you have a relatively short sound file you want to use to “fill up” a given section of text, you want to add continuous background music or you simply wish to ensure that a “parent” background sound is played throughout the reading of the text on the page.

The `AUTO` value is designed to let any parent background sound continue to play under a given Web element. The `NONE` value effectively “turns off” any sound that is playing, creating silence. The `INHERIT` value explicitly sets the sound to whatever parent sound has previously been set.

The following code example, which stages a scene from Hamlet, shows several of these values put into action:

### Listing 27-7 Play-During Example

```
<HTML>
<HEAD>
<TITLE>Play-During Example</TITLE>
</HEAD>
<BODY STYLE="PLAY-DURING: WINDY-OUTDOORS.WAV REPEAT;">
Hamlet: Whither wilt thou lead me? Speak! I'll go no
further.
<P STYLE="PLAY-DURING: GHOSTLY.WAV MIX REPEAT;">
Ghost: Mark me.
</P>
Hamlet: I will.
<P STYLE="PLAY-DURING: GHOSTLY.WAV MIX REPEAT;">
Ghost: My hour is almost come, when I to sulph'rous and
tormenting flames must render up myself.
</BODY>
</HTML>
```

In this brief scene, two sound files are played: WINDY-OUTDOORS.WAV and GHOSTLY.WAV. The WINDY-OUTDOORS.WAV — combined with the REPEAT value — is set to the <BODY> tag, so that sound will be played throughout the reading of the Web page. The GHOSTLY.WAV file is set up so that it is associated only with those lines spoken by the ghost, designed to lend a special mood to his lines. The MIX value is used whenever the GHOSTLY.WAV file is played, so it is played over the WINDY-OUTDOORS.WAV, instead of simply displacing it.

The PLAY-DURING property is not yet supported in either Netscape Navigator or Internet Explorer.

## The AZIMUTH Property

The AZIMUTH property is used to describe the location of a sound in a flat circle surrounding the Web page “viewer” (or, more properly, “listener”). It, along with the ELEVATION property, are designed to allow the Web author to “place” where speech and sound will appear in a three-dimensional sound space. In simpler terms, the AZIMUTH property sets where the sound is placed “around” the listener, and the ELEVATION property sets where the sound is in terms of its “height” above or below the listener.

### AZIMUTH

Description: Sets the “angle” in which a sound is positioned in a sound space surrounding the viewer.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Spatial properties)

Values:

- *n* DEG - The angle in degrees in which a sound is located. The range of the angle is from 0DEG to 360DEG. In this scheme, a value of 0DEG is directly ahead of the viewer, 180DEG is behind the viewer, 90DEG is to the right, and 270DEG is to the left. These values can also be expressed as negative values, so 90DEG is equivalent to -270DEG and 270DEG is the same as -90DEG.
- LEFT-SIDE - Equivalent to 270DEG. If BEHIND is also set, its value does not change.

- FAR-LEFT - Equivalent to 300DEG. If BEHIND is also set, the value becomes 240DEG.
- LEFT - Same as 320DEG. If BEHIND is also set, the value becomes 220DEG.
- CENTER-LEFT - Equivalent to 340DEG. If BEHIND is also set, the value becomes 200DEG.
- CENTER - Equivalent to 0DEG. If BEHIND is also set, the value becomes 180DEG.
- CENTER-RIGHT - Equivalent to 20DEG. If BEHIND is also set, the value becomes 160DEG.
- RIGHT - Equivalent to 40DEG. If BEHIND is also set, the value becomes 140DEG.
- FAR-RIGHT - Equivalent to 60DEG. If BEHIND is also set, the value becomes 120DEG.
- RIGHT-SIDE - Equivalent to 90DEG. If BEHIND is also set, its value does not change.
- BEHIND - Equivalent to 180DEG. Can also be used to modify other named values, taking a position that was in front and “mirroring” it to the same location behind the viewer.
- LEFTWARDS - Moves the sound 20 degrees to the left (counterclockwise) to its current position.
- RIGHTWARDS - Moves the sound 20 degrees to the right (clockwise) to its current position.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

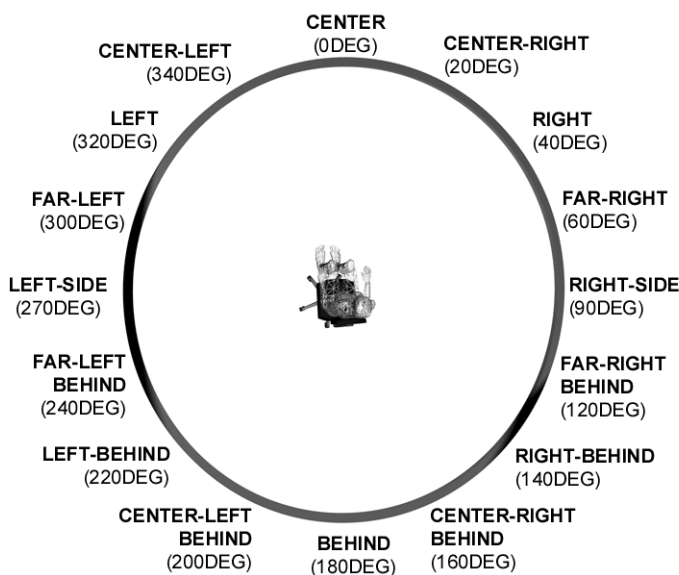
Sample Code:

```
<P AZIMUTH: LEFT>Hamlet: How does the Queen?<P>
<P AZIMUTH: RIGHT>King: She sounds to see them
bleed. </P>
<P AZIMUTH: FAR-RIGHT>Queen: No, no! the drink,
the drink! O my dear Hamlet!
The drink, the drink! I am poison'd. </P>
```

The AZIMUTH property can take both a numeric value and several preset named values. The numeric value is set in degrees (DEG) with a range in angle values from 0 to 360. In this scheme, a value of 0DEG places a sound directly in front of the listener, 180DEG places it behind the listener, 90DEG is to the right, and 270DEG is to the left. These numeric values can also be expressed in negatives, so that 90DEG is equivalent to -270DEG, 180DEG is the same as 0DEG, -270DEG is the same as 90DEG, and -360DEG is the same as 180DEG.

There are also a number of convenient named values that are preset equivalents to various numeric values. These named values are: **LEFT-SIDE**, **FAR-LEFT**, **LEFT**, **CENTER-LEFT**, **CENTER**, **CENTER-RIGHT**, **RIGHT**, **FAR-RIGHT**, **RIGHT-SIDE** and **BEHIND**. There are also three name values that can be used in conjunction with these other values: **BEHIND**, **LEFTWARDS** and **RIGHTWARDS**. Finally, there is the **INHERIT** value, which takes on the value of any parent value set for **AZIMUTH** on the Web page.

As seen from above, starting from the leftmost value and going clockwise, the values run in the following order: **LEFT-SIDE** (270DEG), **FAR-LEFT** (300DEG), **LEFT** (320DEG), **CENTER-LEFT** (340DEG), **CENTER** (0DEG), **CENTER-RIGHT** (20DEG), **RIGHT** (40DEG), **FAR-RIGHT** (60DEG), and **RIGHT-SIDE** (90DEG). If the value **BEHIND** is combined with any of these values — with the exception of **LEFT-SIDE** and **RIGHT-SIDE** values — it inverts or “mirrors” the value, placing it at the same angle *behind* the listener. Continuing the circle going clockwise, the values are **FAR-RIGHT BEHIND** (120DEG), **RIGHT-BEHIND** (140DEG), **CENTER-RIGHT BEHIND** (160DEG), **BEHIND** (180DEG), **CENTER-LEFT BEHIND** (200DEG), **LEFT BEHIND** (220DEG) and **FAR-LEFT BEHIND** (240DEG). The following illustration shows how the sound can be placed all around the listener using these named values.



**Figure 27-1** Using named values with **AZIMUTH** to place sound around a listener.

There are two other named values used to shift sound from the last location: LEFTWARDS and RIGHTWARDS. LEFTWARDS moves the sound 20 degrees to the left (counterclockwise) from its last position, and RIGHTWARDS moves the sound 20 degrees to the right (clockwise) from its last position.

### Listing 27-8 Azimuth Example

```
<HTML>
<HEAD>
<TITLE>Azimuth Example</TITLE>
</HEAD>
<BODY>
<P STYLE="AZIMUTH: LEFT;">Hamlet: How does the Queen?<P>
<P STYLE="AZIMUTH: RIGHT;">King: She sounds to see them
bleed. </P>
<P STYLE="AZIMUTH: FAR-RIGHT;">Queen: No, no! the drink,
the drink! O my dear Hamlet!
The drink, the drink! I am poison'd.</P>
<P STYLE="AZIMUTH: CENTER;">[Dies]</P>
<P STYLE="AZIMUTH: LEFT;">Hamlet: O villany! Ho! let the
door be lock'd.
Treachery! Seek it out.</P>
<P STYLE="FAR-LEFT;">Laertes: It is here, Hamlet. Hamlet,
thou art slain; No medicine in the world can do thee good.
In thee there is not half an hour of life. The treacherous
instrument is in thy hand, Unbated and envenom'd. The foul
practice Hath turn'd itself on me. Lo, here I lie, Never to
rise again. Thy mother's poison'd. I can no more. The King,
the King's to blame. </P>
<P STYLE="AZIMUTH: CENTER;">[Dies]</P>
<P STYLE="AZIMUTH: LEFT;">Hamlet: The point envenom'd
too?</P>
<P STYLE="AZIMUTH: RIGHTWARDS;">Then, venom, to thy work.</
P>
<P STYLE="AZIMUTH: CENTER;">[Hurts the King]</P>
<P STYLE="AZIMUTH: BEHIND;">All: Treason! treason!</P>
</BODY>
</HTML>
```

---

In this extensive snippet from near the end of Shakespeare's "Hamlet", you can see how the various players are placed around the listener. Hamlet and Laertes are positioned to the left-side, while the King and Queen are positioned to right-side, and the rest of the royal court are placed behind the

listener. All of the positions are static, indicating where particular players are. Movement is suggested by the use of the `RIGHTWARDS` value as Hamlet goes over to stab the King. Using all of these values with the `AZIMUTH` property, it is possible to create a moving, dynamic soundscape, very much like listening to a recorded play.

The creation of these effects is entirely up to the technology available to the listener's computer. While most multimedia systems are capable of stereo output, not all systems are necessarily up to a full 360-degree placement of sound. In these situations, the sound system may try to approximate the placement of sound, perhaps by making sounds placed "behind" sound more distant than those in front. Results are likely to vary, not only between computing systems, but also between individual computers, as well.

As of yet, the `AZIMUTH` property is not implemented by any of the major browsers.

## The `ELEVATION` Property

Where the `AZIMUTH` property is used to set the placement of sound around the listener, the `ELEVATION` property is used to set the relative "height" of the sound above or below the plane of the stereo soundscape.

### ***ELEVATION***

Description: Sets the "angle" in which a sound is played in a sound space that runs from below to above the viewer's position.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Spatial properties)

Values:

- `n DEG` - The angle in degrees in which a sound is located. The range of the angle is from `-90DEG` to `90DEG`. In this scheme, a value of `0DEG` is directly in front of the viewer, `90DEG` is directly above the viewer and `-90DEG` is directly below the viewer.
- `BELOW` - Equivalent to `-90DEG`.
- `LEVEL` - Equivalent to `0DEG`.
- `ABOVE` - Equivalent to `90DEG`.
- `HIGHER` - Adds 10 degrees to the current elevation, in effect "raising" it.



- LOWER - Subtracts 10 degrees to the current elevation, in effect “lowering” it.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

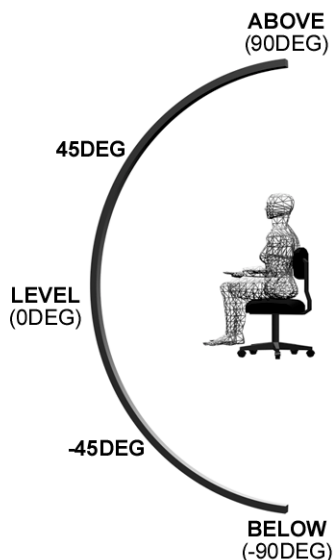
```
<P STYLE="ELEVATION: -20DEG;">
Gravedigger: A pestilence on him for a mad rogue!
'A pour'd a flagon of Rhenish on my head once.
This same skull, sir, was Yorick's skull, the
King's jester.
</P>
<P STYLE="ELEVATION: 20DEG;">
Hamlet: This?
</P>
<P STYLE="ELEVATION: -20DEG;">
Gravedigger: E'en that.
</P>
<P STYLE="ELEVATION: LOWER;">
Hamlet: Let me see.
</P>
<P STYLE="ELEVATION: CENTER;">
[Takes the skull.]
</P>
<P STYLE="ELEVATION: HIGHER;">
Alas, poor Yorick! I knew him, Horatio. A fellow
of infinite jest, of most excellent fancy.
</P>
```

Again, much like the AZIMUTH property, the ELEVATION property can take either a numeric value or a special named value that is the equivalent of a preset numeric value.

Users can set the height of the sound in a virtual semi-circle that goes from directly below the plane of the listener to directly above. In this scheme, the range in angle runs from -90DEG (directly below the listener) to 90DEG (directly above the listener).

There are six named values for ELEVATION: BELOW, LEVEL, ABOVE, HIGHER, LOWER and INHERIT. The preset values are: BELOW (equivalent to -90DEG), LEVEL (equivalent to 0DEG), and ABOVE (equivalent to 90DEG). You can see all of these represented in the following diagram.

The named value HIGHER adds 10 degrees to the current elevation, essentially “raising” the sound up; LOWER subtracts 10 degrees to the cur-



**Figure 27-2** Using named values with AZIMUTH to place sound above and below a listener.

rent elevation, in effect “lowering” it. The `INHERIT` value takes on whatever parent value may have been set for `ELEVATION`.

Both `AZIMUTH` and `ELEVATION` are great for placing different sound sources in a soundscape, allowing Web authors to transform the text on their Web pages to become the actors on a sound stage. The following code provides a practical demonstration as to how you might use `ELEVATION`.

#### Listing 27-9 Elevation Example

```
<HTML>
<HEAD>
<TITLE>Elevation Example</TITLE>
</HEAD>
<BODY>
<P STYLE="ELEVATION: -20DEG;">
Gravedigger: A pestilence on him for a mad rogue! 'A pour'd
a flagon of Rhenish on my head once. This same skull, sir,
```

**Listing 27-9 Elevation Example (continued)**

```
was Yorick's skull, the King's jester.
</P>
<P STYLE="ELEVATION: 20DEG;">
Hamlet: This?
</P>
<P STYLE="ELEVATION: -20DEG;">
Gravedigger: E'en that.
</P>
<P STYLE="ELEVATION: LOWER;">
Hamlet: Let me see.
</P>
<P STYLE="ELEVATION: CENTER;">
[Takes the skull.]
</P>
<P STYLE="ELEVATION: HIGHER;">
Alas, poor Yorick! I knew him, Horatio. A fellow of
infinite jest, of most excellent fancy.
</P>
</BODY>
</HTML>
```

---

In this example derived from “Hamlet”, the gravedigger is quite naturally placed below the plane of the listener, whereas Hamlet, who is standing on the ground, is placed just above it. Using the LOWER and HIGHER value, it would seem to the listener that Hamlet is stooping to pick up Yorick’s skull, then standing up again.

The ELEVATION property is not yet supported in any of the major browsers.

## The SPEECH-RATE Property

The SPEECH-RATE property is designed to allow the Web author to set the rate at which the speech is played back by the Web browser. It essentially sets the “speed” at which the text on the Web page is read.

## ***SPEECH-RATE***

Description: Sets the rate at which words on a Web page are spoken.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Voice characteristic properties)

Values:

- *n* - Sets the speaking rate in terms of words-per-minute read aloud.
- X-SLOW - Equivalent to 80 words-per-minute.
- SLOW - Equivalent to 120 words-per-minute.
- MEDIUM - Equivalent to 180-200 words-per-minute.
- FAST - Equivalent to 300 words-per-minute.
- X-FAST - Equivalent to 500 words-per-minute.
- FASTER - Increases the current speech rate by 40 words-per-minute.
- SLOWER - Decreases the current speech rate by 40 words-per-minute.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<BODY STYLE="SPEECH-RATE: MEDIUM;">
[Ghost beckons Hamlet.]
<P STYLE="SPEECH-RATE: SLOWER;">
Horatio: It beckons you to go away with it, as if
it some impartment did desire to you alone.
</P>
<P STYLE="SPEECH-RATE: FASTER;">
Marcellus: Look with what courteous action It
waves you to a more removed ground. But do not go
with it!
</P>
<P STYLE="SPEECH-RATE: FAST;">
Horatio: No, by no means!
</P>
<P STYLE="SPEECH-RATE: SLOW;">
Hamlet: It will not speak. Then will I follow it.
</P>
</BODY>
```

The **SPEECH-RATE** property can take either a numeric value or one of a series of named values that are the equivalent of preset numeric values. All of these values are designed to set the speaking rate in terms of words read

aloud over the course of a minute. Not only can this property be used to simply set the rate at which the text on the page is read, but it can also lend dramatic emphasis to text, if used properly.

A relaxed, typical English speaking voice runs to just under 200 words per minute. A relatively fast speaker talks at a rate of about 300 words per minute. Thanks to computer technology, you can have the computer read text extremely fast, with the limit really being set by the software driving the text-reading program.

There are eight different named values for SPEECH-RATE: X-SLOW, SLOW, MEDIUM, FAST, X-FAST, FASTER, SLOWER and INHERIT. X-SLOW is equivalent to 80 words-per-minute, SLOW to 120 words-per-minute, MEDIUM to 180-200 words-per-minute, FAST to 300 words-per-minute and finally X-FAST to 500 words-per-minute. The FASTER value increases the current speech rate by 40 words-per-minute, and SLOWER decreases the speech rate by 40 words-per-minute. The ubiquitous INHERIT property takes the same value as that set for the parent value for SPEECH-RATE.

It is safe to say that there will be an implicit SPEECH-RATE value set for any speech-enabled browser. For the most part, Web authors who intend to use this property are likely to either go with the implicit value used by the browser (likely equivalent to a MEDIUM value in the range of 180-200 words-per-minute) or to set an explicit value in the <BODY> tag of the Web page. SPEECH-RATE can also be used to lend dramatic emphasis to sections of text on a Web page, though, as the following code sample shows:

### Listing 27-10 Speech-Rate Example

```
<HTML>
<HEAD>
<TITLE>Speech-Rate Example</TITLE>
</HEAD>
<BODY STYLE="SPEECH-RATE: MEDIUM;">
[Ghost beckons Hamlet.]
<P STYLE="SPEECH-RATE: SLOWER;">
Horatio: It beckons you to go away with it, as if it some
impairment did desire to you alone.
</P>
<P STYLE="SPEECH-RATE: FASTER;">
Marcellus: Look with what courteous action It waves you to
a more removed ground. But do not go with it!
</P>
```

**Listing 27-10 Speech-Rate Example (continued)**

```
<P STYLE="SPEECH-RATE: FAST;">
Horatio: No, by no means!
</P>
<P STYLE="SPEECH-RATE: SLOW;">
Hamlet: It will not speak. Then will I follow it.
</P>
</BODY>
</HTML>
```

In this brief scene, a whole range of `SPEECH-RATE` values are used. The initial value is set to `MEDIUM`, but Horatio's initial lines are rendered using `SLOWER` to add weight to his apprehension. Marcellus' exclamation is rendered using `FASTER`, and Horatio's imploring of Hamlet uses the `FAST` setting. The contemplative Hamlet's concluding words are rendered using `SLOW`.

The `SPEECH-RATE` property is not yet supported in either Netscape Navigator or Internet Explorer.

## The VOICE-FAMILY Property

Using the `VOICE-FAMILY` property, you can actually set the "sex" of a speaking voice. Not only can you set a generic sex-value to the voice, but you can also add specific voices if they are supported by your browser.

### ***VOICE-FAMILY***

Description: Sets a priority list of voice family names.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Voice characteristic properties)

Values:

- **MALE | FEMALE | CHILD** - A "generic" voice value that sounds male, female and child-like, respectively. Can modify the value of a specific-voice.
- ***specific-voice name*** - A specific, predefined name for a voice value.

- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="VOICE-FAMILY: LAERTES, MALE;">
Laertes: Farewell, Ophelia, and remember well what
I have said to you.
</P>
<P STYLE="VOICE-FAMILY: OPHELIA, FEMALE;">
Ophelia: 'Tis in my memory lock'd, and you
yourself shall keep the key of it.
</P>
<P STYLE="VOICE-FAMILY: LAERTES, MALE;">
Laertes: Farewell.
</P>
<P STYLE="VOICE-FAMILY: MALE;">
Exit Laertes.
</P>
```

The **VOICE-FAMILY** property can take any of four specific named values: **MALE**, **FEMALE**, **CHILD** and **INHERIT**. It can also take a specific, pre-defined voice name whose vocal characteristics have previously been set. It is the vocal equivalent of the **FONT-FAMILY** property, allowing Web authors to set several values for a voice; the first value that matches what is available on the browser is the one that is used.

Although it is not explicitly stated in the CSS2 specification, it is assumed that the **MALE** voice has a relatively deep sound to it, a **FEMALE** voice would be rendered in a higher pitch, and a **CHILD**'s voice would be higher still. Since there are no specific values provided, chances are that the results will vary from browser to browser.

The **VOICE-FAMILY** property can also take any predefined voice name that may exist. There is no explanation in the specification as to how this might work in practice, but presumably various standard voice types would be made available to the browser. As a fall-back, you can always use the standard sex names as well, in case the specific voices are not universally available. With that in mind, it should be possible to write the following code:

**Listing 27-11 Voice-Family Example**

```
<HTML>
<HEAD>
<TITLE>Voice-Family Example</TITLE>
</HEAD>
<BODY>
<P STYLE="VOICE-FAMILY: LAERTES, MALE;">
Laertes: Farewell, Ophelia, and remember well what I have
said to you.
</P>
<P STYLE="VOICE-FAMILY: OPHELIA, FEMALE;">
Ophelia: 'Tis in my memory lock'd, and you yourself shall
keep the key of it.
</P>
<P STYLE="VOICE-FAMILY: LAERTES, MALE;">
Laertes: Farewell.
</P>
<P STYLE="VOICE-FAMILY: MALE;">
Exit Laertes.
</P>
</BODY>
</HTML>
```

---

In this code example, there are two distinct voices being used: LAERTES, a male voice, and OPHELIA, a female voice. In each case, the specific voice-name, such as “LAERTES”, is followed up by the name value “MALE”. In this way, in case the preset vocal characteristics for the character of LAERTES is not present on a user’s system, it will load up the default MALE voice instead.

The VOICE-FAMILY property is not yet supported in any of the major browsers.

## The PITCH Property

The PITCH property is designed to set the pitch level for a given speaking voice. Using this property, you can artificially set up different types of voices to be played back through your browser. The PITCH property helps Web authors better characterize voices reading text from the Web page.



## PITCH

Description: Sets the average pitch for a given speaking voice. The average pitch for a typical male voice is 120Hz, while for a female voice it is typically about 210Hz.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Voice Characteristic Properties)

Values:

- ***n* HZ** - Sets a numeric value for the speaking voice in Hertz.
- **X-LOW | LOW | MEDIUM | HIGH | X-HIGH** - There are no specific pitch settings for these values, as that depends on the voice family being used. The browser should interpret these values from a very low pitch (X-LOW) to a very high pitch (X-HIGH).
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="VOICE-FAMILY: MALE; PITCH: MEDIUM;">
Hamlet: What hour now?
</P>
<P STYLE="VOICE-FAMILY: MALE; PITCH: LOW;">
Horatio: I think it lacks of twelve.
</P>
<P STYLE="VOICE-FAMILY: MALE; PITCH: X-LOW;">
Marcellus: No, it is struck.
</P>
```

Another member of the voice characteristics grouping of sub-properties, the **PITCH** property enables you to set the current pitch of the voice reading the text aloud. Depending on how widely this property and the **VOICE-FAMILY** properties are eventually implemented in the major browsers, they both seem to be alternate ways of achieving much the same thing: lending character to a given speaking voice.

The average pitch of a given voice will vary depending on the **VOICE-FAMILY** value to which it belongs. For example, a typical male voice is a relatively low 120Hz, whereas a typical female voice is usually higher, in the range of 210Hz.

It is possible to either set a numeric value for the speaking voice in Hertz (Hz), or in one of five named values: **X-LOW**, **LOW**, **MEDIUM**, **HIGH**, **X-HIGH** and **INHERIT**. The CSS2 specification does not set specific Hertz values for these named values, as they can have different values depending

on the VOICE-FAMILY value they belong to (a value of “HIGH” for a male voice would necessarily be lower than the same value applied to a female voice). All the specification says is that the values must progress from low Hertz values (from “X-LOW”) to high Hertz values (to “X-HIGH”). The INHERIT value simply takes on whatever PITCH value has been assigned to the parent property.

Although not specifically referenced in the specification, it is implied that the values for PITCH are most likely to be used in conjunction with the VOICE-FAMILY property, making it possible to vary the standard VOICE-FAMILY properties of MALE, FEMALE and CHILD. In this way, it should be relatively easy for a “viewer” to distinguish between different reading voices on a given Web page, as can be seen in the following code example:

#### Listing 27-12 Pitch Example

```
<HTML>
<HEAD>
<TITLE>Pitch Example</TITLE>
</HEAD>
<BODY>
<P STYLE="VOICE-FAMILY: MALE; PITCH: MEDIUM;">
Hamlet: What hour now?
</P>
<P STYLE="VOICE-FAMILY: MALE; PITCH: LOW;">
Horatio: I think it lacks of twelve.
</P>
<P STYLE="VOICE-FAMILY: MALE; PITCH: X-LOW;">
Marcellus: No, it is struck.
</P>
</BODY>
</HTML>
```

---

In this case, we have three different male characters. Normally, it would be hard to distinguish between them, but by using different values for the PITCH property, it is possible to do so. The voice assigned to HAMLET has a PITCH value of MEDIUM, Horatio has a PITCH value of LOW, and Marcellus' PITCH is set to X-LOW.

This property is not as yet implemented in any of the major browsers.

# The PITCH-RANGE Property

Another member of the voice characteristics of aural sub-properties, the `PITCH-RANGE` property is meant to be used in conjunction with the `PITCH` property, and is designed to enable the Web author to set the variable range for a given speaking voice. While the `PITCH` property is used to set the average pitch of a given spoken voice, the `PITCH-RANGE` property is essentially used to add the amount of inflection that is added to a speaking voice. A relatively small value leads to a dry voice, whereas a high value sets a highly animated voice.

## ***PITCH-RANGE***

Description: Specifies variation of the pitch for a particular voice.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Voice characteristic properties)

Values:

- *n* - A numerical value in a range between 0 and 100. A value of 50 is the normal pitch-range for a given speaking voice, 0 produces a monotone speaking voice, while 100 produces a highly animated speaking voice.
- `INHERIT` - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="VOICE-FAMILY: MALE; PITCH: LOW; PITCH-  
RANGE: 25;">  
Ghost: Revenge his foul and most unnatural murder.  
</P>  
<P STYLE="VOICE-FAMILY: MALE; PITCH: MEDIUM;  
PITCH-RANGE: 80;">  
Hamlet: Murder?  
</P>  
<P STYLE="VOICE-FAMILY: MALE; PITCH: MEDIUM;  
PITCH-RANGE: 35;">  
Ghost: Murder most foul, as in the best it is; but  
this most foul, strange, and unnatural.  
</P>
```

The `PITCH-RANGE` property can either take a specific numerical value that falls anywhere in a range between 0 and 100, and the value `INHERIT`. A numerical value of 50 is the default value, and is the normal pitch-range for a

given English-speaking voice. A value of 0 produces a speaking voice that is a monotone, staying at the value fixed by `PITCH`. A value of 100 produces a highly animated speaking voice. If you think of things in terms of emotionally charged speaking voices, a 0 value for `PITCH-RANGE` produces a dry, emotionless voice, whereas a value of 100 produces a highly emotionally charged voice. The named value `INHERIT` simply takes on whatever parent value may have been set for `PITCH-RANGE`. The following code example tries to add some emotion to the voices speaking the lines from Shakespeare's "Hamlet":

### Listing 27-13 Pitch-Range Example

```
<HTML>
<HEAD>
<TITLE>Pitch-Range Example</TITLE>Pi</TITLE>
</HEAD>
<BODY>
<P STYLE="VOICE-FAMILY: MALE; PITCH: LOW; PITCH-RANGE:
25;">
Ghost: Revenge his foul and most unnatural murder.
</P>
<P STYLE="VOICE-FAMILY: MALE; PITCH: MEDIUM; PITCH-RANGE:
80;">
Hamlet: Murder?
</P>
<P STYLE="VOICE-FAMILY: MALE; PITCH: MEDIUM; PITCH-RANGE:
35;">
Ghost: Murder most foul, as in the best it is; but this
most foul, strange, and unnatural.
</P>
</BODY>
</HTML>
```

---

In this short scene, the Ghost speaks in a cold voice, drained of emotion, whereas Hamlet reacts to the news of his father's murder with a great deal of emotion.

Neither Internet Explorer nor Netscape Navigator support this property yet.

# The STRESS Property

The STRESS property is used to help “color” a voice, in order to add more inflection or emphasis. This property is a companion to the PITCH-RANGE property, aiding it in an effort to add more or less emotional inflection to a given voice.

## STRESS

Description: Sets the amount of stress (i.e., the emphasis) of the speaking voice.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Voice characteristic properties)

Values:

- ***n*** - Sets the value of stress in a given voice. Range is between 0 and 100, with a value of 50 being the normal stress for a given voice.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="VOICE-FAMILY: MALE; PITCH: LOW; PITCH-  
RANGE: 25; STRESS: 40">  
King: How do you, pretty lady?  
</P>  
<P STYLE="VOICE-FAMILY: FEMALE; PITCH: HIGH;  
PITCH-RANGE: 45; STRESS: 95">  
Ophelia: Well, God 'ild you! They say the owl was a  
baker's daughter. Lord, we know what we are, but  
know not what we may be. God be at your table!  
</P>
```

The STRESS property can either take a numeric value ranging from 0 to 100, or can take the named property INHERIT common to all CSS2 properties.

English is a language that relies upon stress values within words. The more a word is stressed, the more each syllable in the word is emphasized, and the more inflected it becomes. In the CSS2 scheme, a value of 50 is equal to normally inflected English speech, 0 is equal to very dry, uninflected speech, and 100 is a wildly inflected voice. The INHERIT property simply takes on whatever parent value may have already been set for the STRESS property.

You can see how this property can be put to use in the following excerpt from “Hamlet”:

### Listing 27-14 Stress Example

```
<HTML>
<HEAD>
<TITLE>Stress Example</TITLE>
</HEAD>
<BODY>
<P STYLE="VOICE-FAMILY: MALE; PITCH: LOW; PITCH-RANGE: 25;
STRESS: 40">
King: How do you, pretty lady?
</P>
<P STYLE="VOICE-FAMILY: FEMALE; PITCH: HIGH; PITCH-RANGE:
45; STRESS: 95">
Ophelia: Well, God 'ild you! They say the owl was a baker's
daughter. Lord, we know what we are, but know not what we
may be. God be at your table!
</P>
</BODY>
</HTML>
```

---

In this brief snippet, we see how the King's words are relatively uninflected, with a `STRESS` value of 40, reflecting how solemn and reserved he is in front of the clearly mad Ophelia. Ophelia, being insane, has highly inflected speech with a `STRESS` value of 95.

The `STRESS` property is not yet implemented in either Internet Explorer or Netscape Navigator.

## The RICHNESS Property

The `RICHNESS` property is used to set the “brightness” (also known as “richness”) of the speaking voice. This term really refers to the power behind the voice, and its ability to fill a room. A very rich voice fills a room, whereas a smooth voice does not.

## RICHNESS

Description: Sets the “brightness” (also known as “richness”) of the speaking voice being used. The brighter the voice, the better it can be heard from a distance.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Voice characteristic properties)

Values:

- *n* - Sets the value of brightness in a given voice. Range is between 0 and 100, with a value of 50 being the normal value for a given voice.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="VOICE-FAMILY: MALE; PITCH: LOW; PITCH-  
RANGE: 25; STRESS: 40"; RICHNESS: 80;">
```

```
King: O, yet defend me, friends! I am but hurt.  
</P>
```

```
<P STYLE="VOICE-FAMILY: MALE; PITCH: MEDIUM;  
PITCH-RANGE: 40; STRESS: 80"; RICHNESS: 70;">
```

```
Hamlet: Here, thou incestuous, murd'rous, damned  
Dane,  
Drink off this potion! Is thy union here?  
Follow my mother.
```

```
</P>
```

```
<P STYLE="VOICE-FAMILY: MALE; PITCH: HIGH; PITCH-  
RANGE: 20; STRESS: 50; RICHNESS: 40;">
```

```
[King dies.]
```

```
</P>
```

The RICHNESS property can either take a numeric value within a range of 0 to 100, or the named value INHERIT. The higher the numeric value, the more the voice will “carry” and resonate within the room. A lower value produces a softer, more mellifluous voice. The default value for RICHNESS is 50, equal to that of regular speech. Any Web element set to the INHERIT value will take on whatever value for RICHNESS that has already been set in the parent.

Like many of the other properties in the voice characteristics sub-group, **RICHNESS** lends itself well to adding further depth the characterization of the reading voice. This can be seen in the following code example:

### Listing 27-15 Richness Example

```
<HTML>
<HEAD>
<TITLE>Richness Example</TITLE>
</HEAD>
<BODY>
<P STYLE="VOICE-FAMILY: MALE; PITCH: LOW; PITCH-RANGE: 25;
STRESS: 40"; RICHNESS: 80;">
King: O, yet defend me, friends! I am but hurt.
</P>
<P STYLE="VOICE-FAMILY: MALE; PITCH: MEDIUM; PITCH-RANGE:
40; STRESS: 80"; RICHNESS: 70;">
Hamlet: Here, thou incestuous, murd'rous, damned Dane,
Drink off this potion! Is thy union here?
Follow my mother.
</P>
<P STYLE="VOICE-FAMILY: MALE; PITCH: HIGH; PITCH-RANGE: 20;
STRESS: 50; RICHNESS: 40;">
[King dies.]
</P>
</BODY>
</HTML>
```

---

In this scene from “Hamlet”, a dying Hamlet forces the King to drink the poison meant for him. In this emotionally charged scene, both Hamlet and the King’s voices have high **RICHNESS** values assigned to their voices. Contrast this to the narrator who mentions the King’s death, which is rendered in a relatively dry, unobtrusive tone with a low **RICHNESS** value.

The **RICHNESS** property is not yet implemented in either Internet Explorer or Netscape Navigator,



# The Speech Sub-Group of Properties

There is one final sub-grouping of like properties in the aural style sheets portion of the CSS2 specification. The oddly non-descriptive speech sub-group of properties contains two properties that tell the browser how it should handle reading things like numbers and dates aloud. These two properties are the `SPEAK-PUNCTUATION` and `SPEAK-NUMERAL` properties.

## ***SPEAK-PUNCTUATION***

Description: Determines how punctuation in text will be read aloud.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Speech properties)

Values:

- `CODE` - Punctuation marks such as periods, commas and brackets are spoken literally.
- `NONE` - Punctuation marks are not spoken, but are instead interpreted as natural pauses in speech.
- `INHERIT` - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
In the following sentence, all of the punctuation
(periods, exclamation marks and commas) are read
aloud.
<P STYLE="SPEAK-PUNCTUATION: CODE";>
Queen: No, no! the drink, the drink! O my dear
Hamlet!
The drink, the drink! I am poison'd.
</P>
```

## ***SPEAK-NUMERAL***

Description: Determines how numbers should be read aloud when encountered by the browser.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Speech properties)

Values:

- **DIGITS** - Pronounces numbers as individual digits. The number “123” would be spoken as “one two three”.
- **CONTINUOUS** - Pronounces numbers as a full number. The number “123” would be spoken as “one hundred and twenty-three”.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P>
Francisco: You come most carefully upon your hour.
</P>
<P STYLE="SPEAK-NUMERAL: CONTINUOUS;">
Bernardo: 'Tis now struck 12. Get thee to bed,
Francisco.
</P>
```

The **SPEAK-PUNCTUATION** and **SPEAK-NUMERAL** properties are similar in that they allow Web authors to specify exactly how certain parts of a Web page’s text should be played back. **SPEAK-PUNCTUATION** determines whether the punctuation marks that exist in the text should be named as the text is read aloud (“Hello there semi-colon”), or whether the text should flow normally according to the punctuation rules (“Hello there;”). Similarly, **SPEAK-NUMERAL** determines whether a string of numbers should be read as “one two three” or as “one hundred and twenty-three”. Both of the properties can be said to operate in a similar way to the **SPELL-OUT** value for the **SPEECH** property.

**SPEAK-PUNCTUATION** can take one of three possible named values: **CODE**, in which punctuation marks (such as periods, commas, brackets and so on) are spoken literally; **NONE**, in which the punctuation marks are not spoken, but are instead interpreted as natural pauses in speech, and the value **INHERIT**, which takes on the value for **SPEAK-PUNCTUATION** that has been previously set for the parent.

**SPEAK-NUMERAL** works in much the same way. It can also take one of three named properties: **DIGITS**, which pronounces numbers as individual digits; **CONTINUOUS**, which pronounces a string of numbers as one full number, and the value **INHERIT**, which simply takes on whatever value for **SPEAK-NUMERAL** has been set for the parent.

You can see both of these properties put into action in the following code sample:

**Listing 27-16 Speak-Punctuation and Speak-Numeral Example**

```
<HTML>
<HEAD>
<TITLE>Speak-Punctuation and Speak-Numeral Example</TITLE>
</HEAD>
<BODY>
In the following sentence, all of the punctuation (periods,
exclamation marks and commas) are read aloud.
<P STYLE="SPEAK-PUNCTUATION: CODE";>
Queen: No, no! the drink, the drink! O my dear Hamlet!
The drink, the drink! I am poison'd.
</P>
<P>
Francisco: You come most carefully upon your hour.
</P>
<P STYLE="SPEAK-NUMERAL: CONTINUOUS;">
Bernardo: 'Tis now struck 12. Get thee to bed, Francisco.
</P>
</BODY>
</HTML>
```

These properties may clearly be used for different purposes. There are occasions when you might want the punctuation to be read aloud, to set apart the text from the punctuation. In the course of normal speech, you would probably want to set the `SPEAK-NUMERAL` property to `CONTINUOUS` (as in the previous code example), but there are cases where it would be better to have the individual numerals spoken separately, such as in a table, so that the listener, who may be copying down the figures, has a better chance of recording the right number.

Neither the `SPEAK-PUNCTUATION` nor the `SPEAK-NUMERAL` properties are supported yet in either Netscape Navigator or in Internet Explorer.

# **CSS1 PROPERTY, PSEUDO-ELEMENT, PSEUDO-CLASS, AND CONCEPTS REFERENCE**



# *Appendix A*

## Alphabetical Listing of CSS Properties

### **BACKGROUND**

Description: Sets the background color or image. It can take on any value contained in the background family of properties.

CSS Family Type: Color and Background

Values:

- *color name* or *numerical color value*
- URL(image)
- SCROLL - Image moves when the browser window is scrolled.
- FIXED - Image does not move when the browser window is scrolled.
- TRANSPARENT – renders the specified element transparent
- X% Y% - Percentage is in reference to the dimensions of the browser window display.
- X Y - Represents absolute coordinate position of the image.

- (LEFT/CENTER/RIGHT) | (TOP/CENTER/BOTTOM) - Keywords representing screen positions. Left keyword is the X-position and the right keyword is the Y-position for the image.
- REPEAT - Image is horizontally and vertically tiled.
- REPEAT-X - Image is horizontally tiled.
- REPEAT-Y - Image is vertically tiled.
- NO-REPEAT - The image is not repeated.

Sample code:

```
<B STYLE="BACKGROUND: GREEN">This text is displayed
against a green background.</B>
```

Support in Major Browsers:

				<i><b>IE 5.0</b></i>		<i><b>NN 4.0</b></i>	
<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01</b></i> <i><b>(Mac)</b></i>	<i><b>IE 4.5</b></i> <i><b>(Mac)</b></i>	<i><b>(Win. &amp;</b></i> <i><b>UNIX)</b></i>	<i><b>NN 4.x</b></i>	<i><b>(Mac &amp;</b></i> <i><b>UNIX)</b></i>	<i><b>Opera</b></i> <i><b>3.6</b></i>
Partial	Partial	Partial	Safe	Safe	Partial	Partial	Safe

## BACKGROUND-ATTACHMENT

Description: If BACKGROUND-IMAGE element is set, this element specifies how the background image should move when the browser window is scrolled.

CSS Family Type: Color and Background

Values:

- SCROLL - Image moves when the browser window is scrolled.
- FIXED - Image does not move when the browser window is scrolled.

Sample code:

```
<BODY STYLE="BACKGROUND-IMAGE: URL(canvas.gif);
BACKGROUND-ATTACHMENT: FIXED">
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Safe

## BACKGROUND-COLOR

Description: Sets the background color of an element.

CSS Family Type: Color and Background

Values:

- *color name* or numerical color value
- TRANSPARENT – renders the specified element transparent

Sample code:

```
<I STYLE="BACKGROUND-COLOR: FFFF00">Italicized  
text on a yellow background.</I>
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Safe

## BACKGROUND-IMAGE

Description: Sets a graphic image as a background image.

CSS Family Type: Color and Background

If this element is specified, the CSS elements BACKGROUND-REPEAT, BACKGROUND-ATTACHMENT and BACKGROUND-POSITION can also be used with it. A color value can also be added.

Values:

- URL(image)

Sample code:

```
<BODY STYLE="BACKGROUND-IMAGE: URL(canvas.gif)">
```

Support in Major Browsers:

<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01 (Mac)</b></i>	<i><b>IE 4.5 (Mac)</b></i>	<i><b>IE 5.0 (Win. &amp; UNIX)</b></i>	<i><b>NN 4.x</b></i>	<i><b>NN 4.0 (Mac &amp; UNIX)</b></i>	<i><b>Opera 3.6</b></i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Safe

## BACKGROUND-POSITION

Description: If the BACKGROUND-IMAGE element is specified, this element specifies where the image first appears, and then describes how it should be tiled.

CSS Family Type: Color and Background

Values:

- X% Y% - Percentage is in reference to the dimensions of the browser window display.
- X Y - Represents the absolute coordinate position of the image.
- (LEFT/CENTER/RIGHT) | (TOP/CENTER/BOTTOM) - Keywords representing screen positions. Left keyword is the X-position and the right keyword is the Y-position for the image.

Sample code:

```
<BODY STYLE="BACKGROUND-IMAGE:
URL(victorinus.jpg); BACKGROUND-POSITION: 100 50">
```

Support in Major Browsers:

<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01 (Mac)</b></i>	<i><b>IE 4.5 (Mac)</b></i>	<i><b>IE 5.0 (Win. &amp; UNIX)</b></i>	<i><b>NN 4.x</b></i>	<i><b>NN 4.0 (Mac &amp; UNIX)</b></i>	<i><b>Opera 3.6</b></i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Safe



## BACKGROUND-REPEAT

Description: If the BACKGROUND-IMAGE element is set, this additional element specifies how the image is repeated on the Web page.

CSS Family Type: Color and Background

Values:

- REPEAT - Image is horizontally and vertically tiled.
- REPEAT-X - Image is horizontally tiled.
- REPEAT-Y - Image is vertically tiled.
- NO-REPEAT - The image is not repeated.

Sample code:

```
<BODY STYLE="BACKGROUND-IMAGE: URL(gordian3.jpg);
BACKGROUND-REPEAT: REPEAT-X">
```

Support in Major Browsers:

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> <i>(Mac)</i>	<i>IE 4.5</i> <i>(Mac)</i>	<i>(Win. &amp;</i> <i>UNIX)</i>	<i>NN 4.x</i>	<i>(Mac &amp;</i> <i>UNIX)</i>	<i>Opera</i> <i>3.6</i>
Unsafe	Partial	Partial	Safe	Safe	Partial	Partial	Safe

## BORDER

Description: This is an abbreviated element that allows the Web author to specify BORDER-TOP, BORDER-RIGHT, BORDER-BOTTOM and BORDER-LEFT elements easily.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units
- Sets the thickness of the border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.
- *color name* or *numerical color value* - Sets the color for the border.
- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.

- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.

Sample code:

```
<STRONG STYLE="BORDER: RIDGE RED">A strong
statement.</STRONG>
```

Support in Major Browsers:

				<i><b>IE 5.0</b></i>		<i><b>NN 4.0</b></i>	
<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01</b></i>	<i><b>IE 4.5</b></i>	<i><b>(Win. &amp;</b></i>	<i><b>NN 4.x</b></i>	<i><b>(Mac &amp;</b></i>	<i><b>Opera</b></i>
		<i><b>(Mac)</b></i>	<i><b>(Mac)</b></i>	<i><b>UNIX)</b></i>		<i><b>UNIX)</b></i>	<i><b>3.6</b></i>
Unsafe	Partial	Partial	Safe	Partial	Partial	Partial	Partial

## ***BORDER-BOTTOM***

Descriptions: Sets the display value for the bottom border of the specified HTML tag.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- ***n*** measurement units
- Sets the thickness of the border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.
- ***color name*** or ***numerical color value*** - Sets the color for the border.
- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.

- **SOLID** - A solid border line is displayed.

Sample code:

```
<EM STYLE="BORDER-BOTTOM: DOUBLE BLUE">Text with a
double blue underline.</EM>
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Partial

## ***BORDER-BOTTOM-WIDTH***

Description: Sets the thickness of the bottom border.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units
- Sets the thickness of the bottom border.
- **THIN** | **MEDIUM** | **THICK** - Sets the relative thickness of the border.

Sample code:

```
<BLOCKQUOTE STYLE="BORDER-BOTTOM-WIDTH:
THICK">"This is an important quote,"</BLOCKQUOTE>
he said.
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Partial

## BORDER-COLOR

Description: Sets the color of the border sides.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- *color name* or *numerical color value* - Sets the color for the border.

Sample code:

```
<H3 STYLE="BORDER-COLOR: OLIVE">Header with a  
Colored Border</H3>
```

Support in Major Browsers:

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> (Mac)	<i>IE 4.5</i> (Mac)	(Win. & UNIX)	<i>NN 4.x</i>	(Mac & UNIX)	<i>Opera</i> <i>3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Partial

## BORDER-LEFT

Description: Specifies how the left border associated with an HTML tag should be displayed.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units
- Sets the thickness of the border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.
- *color name* or *numerical color value* - Sets the color for the border.
- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.

- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.

Sample code:

```
<ADDRESS STYLE="BORDER-LEFT: 1.0cm AQUA">123
Underwater Drive<BR>An aqua-tic address.</ADDRESS>
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Partial

## BORDER-LEFT-WIDTH

Sets the thickness of the left border.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units
- Sets the thickness of the left border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.

Sample code:

```
<H4 STYLE="BORDER-LEFT-WIDTH THIN">Text with a thin
border to the left.</H4>
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Partial

## BORDER-RIGHT

Description: Specifies how the right border associated with an HTML tag should be displayed.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units
- Sets the thickness of the border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.
- *color name* or *numerical color value* - Sets the color for the border.
- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.

Sample code:

```
<H5 STYLE="BORDER-RIGHT: THICK DOUBLE
YELLOW">Header with Yellow Right Border</H5>
```

Support in Major Browsers:

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> <i>(Mac)</i>	<i>IE 4.5</i> <i>(Mac)</i>	<i>(Win. &amp;</i> <i>UNIX)</i>	<i>NN 4.x</i>	<i>(Mac &amp;</i> <i>UNIX)</i>	<i>Opera</i> <i>3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Partial

## BORDER-RIGHT-WIDTH

Description: Sets the thickness of the right border.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units
- Sets the thickness of the right border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.

Sample code:

```
<SUP STYLE="BORDER-RIGHT-WIDTH 0.2in">Superscript  
with border to the right.</SUP>
```

Support in Major Browsers:

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> (Mac)	<i>IE 4.5</i> (Mac)	<i>(Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>(Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Partial

## BORDER-STYLE

Description: Sets the type of border to be displayed.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.

Example:

```
<H1 STYLE="BORDER-STYLE: SOLID">Header with Inset  
Border.</H1>
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Partial	Safe	Safe	Partial	Partial	Partial	Partial

## BORDER-TOP

Description: Specifies how the top border associated with an HTML tag should be displayed.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units
- Sets the thickness of the border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.
- *color name* or *numerical color value* - Sets the color for the border.
- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.

Sample code:

```
<TT STYLE="BORDER-TOP: THIN DASHED
00FF00">Teletype text with a top-border</TT>
```



Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Partial

## BORDER-TOP-WIDTH

Description: Sets the thickness of the top border.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- n measurement units
- Sets the thickness of the top border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.

Sample code:

```
<SUB STYLE="BORDER-TOP-WIDTH 1mm">Subscript with  
very thin top border.</SUB>
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Partial

## BORDER-WIDTH

Description: Sets the thickness of the border for the specified element. One to four border sides can be set.

CSS Family Type: Boxes (Sub-family: Borders)

Values:

- n [ n n n ] measurement units
- Sets the thickness of the border.

- If 1 value is present, all borders are set to the numerical value specified
- If 2 values are present, the top/bottom and side borders take the numerical values specified
- If 3 values are present, the top, right and left, then bottom takes the numerical values specified
- If 4 values are present, the top, right, bottom then left borders take the numerical values specified
- **THIN | MEDIUM | THICK** - Sets the relative thickness of the border.

Sample code:

```
<H2 STYLE="BORDER-WIDTH: 5 5 15 5">Header with  
Surrounding Border</H2>
```

Support in Major Browsers:

				<i><b>IE 5.0</b></i>		<i><b>NN 4.0</b></i>	
		<i><b>IE 4.01</b></i>	<i><b>IE 4.5</b></i>	<i><b>(Win. &amp;</b></i>		<i><b>(Mac &amp;</b></i>	<i><b>Opera</b></i>
<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>(Mac)</b></i>	<i><b>(Mac)</b></i>	<i><b>UNIX)</b></i>	<i><b>NN 4.x</b></i>	<i><b>UNIX)</b></i>	<i><b>3.6</b></i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Partial

## CLEAR

Description: Sets whether the specified element will allow other elements to “float” along its sides.

CSS Family Type: Boxes

Values:

- **BOTH** - The specified element will be moved below floating elements on either side.
- **LEFT** - The specified element will be moved below floating elements on the left side only.
- **NONE** - Any floating elements are allowed on either side of the specified element.
- **RIGHT** - The specified element will be moved below floating elements on the right side only.

Example:

```
<B STYLE="CLEAR: BOTH">This text should appear  
below the image.</B>
```

## Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> (Mac)	<i>IE 4.5</i> (Mac)	<i>IE 5.0</i> (Win. & UNIX)	<i>NN 4.x</i>	<i>NN 4.0</i> (Mac & UNIX)	<i>Opera</i> <i>3.6</i>
Unsafe	Partial	Partial	Safe	Safe	Partial	Partial	Safe

Clear / Float variant:

		<i>IE 4.01</i>	<i>IE 4.5</i>	<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	(Mac)	(Mac)	(Win. & UNIX)	<i>NN 4.x</i>	(Mac & UNIX)	<i>Opera 3.6</i>
Unsafe	Partial	Unsafe	Partial	Partial	Unsafe	Unsafe	Safe

## COLOR

Description: Sets the color to be displayed.

## CSS Family Type: Color and Background

Values:

- *color name* or *numerical color value*

Sample code:

```
<UL>
<LI STYLE="COLOR: FF0000">This is displayed as red
text
<LI STYLE="COLOR: 00FF00">This is displayed as
green text
<LI STYLE="COLOR: 0000FF">This is displayed as
blue text
</UL>
```

## Support in Major Browsers:

[illegible]

## DISPLAY

Description: Controls the fundamental nature of the specified HTML tag.

CSS Family Type: Classification

Value:

- BLOCK - Sets the specified element as a block-type element.
- INLINE - Sets the specified element as an inline-type element.
- LIST-ITEM - Sets the specified element as a type of list-item.
- NONE - Switches off the display of the specified element.

Sample code:

```
<I STYLE="DISPLAY: LIST-ITEM">This italicized text
is indented in the same way a list-item would be.
</I>
```

Support in Major Browsers:

				<i><b>IE 5.0</b></i>		<i><b>NN 4.0</b></i>	
<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01</b></i> <i><b>(Mac)</b></i>	<i><b>IE 4.5</b></i> <i><b>(Mac)</b></i>	<i><b>(Win. &amp;</b></i> <i><b>UNIX)</b></i>	<i><b>NN 4.x</b></i>	<i><b>(Mac &amp;</b></i> <i><b>UNIX)</b></i>	<i><b>Opera</b></i> <i><b>3.6</b></i>
Unsafe	Unsafe	Unsafe	Partial	Partial	Partial	Partial	Partial

## FLOAT

Description: Sets how elements can wrap around another, “parent” element as it floats on a Web page.

CSS Family Type: Boxes

Values:

- NONE - Other elements cannot be displayed to float around the specified element.
- LEFT - Other elements can be displayed to the left margin of the specified element.
- RIGHT - Other elements can be displayed to the right margin of the specified element.

Example:

```
<IMG SRC="aurelian.jpg" STYLE="FLOAT: RIGHT">This
text floats to the left of the image.
```

Support in Major Browsers:

<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01 (Mac)</b></i>	<i><b>IE 4.5 (Mac)</b></i>	<i><b>IE 5.0 (Win. &amp; UNIX)</b></i>	<i><b>NN 4.x</b></i>	<i><b>NN 4.0 (Mac &amp; UNIX)</b></i>	<i><b>Opera 3.6</b></i>
Unsafe	Partial	Partial	Partial	Partial	Partial	Partial	Partial

Float / Margin Variant:

<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01 (Mac)</b></i>	<i><b>IE 4.5 (Mac)</b></i>	<i><b>IE 5.0 (Win. &amp; UNIX)</b></i>	<i><b>NN 4.x</b></i>	<i><b>NN 4.0 (Mac &amp; UNIX)</b></i>	<i><b>Opera 3.6</b></i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Safe

Float Elements in Series Variant:

<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01 (Mac)</b></i>	<i><b>IE 4.5 (Mac)</b></i>	<i><b>IE 5.0 (Win. &amp; UNIX)</b></i>	<i><b>NN 4.x</b></i>	<i><b>NN 4.0 (Mac &amp; UNIX)</b></i>	<i><b>Opera 3.6</b></i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Safe

Float on Text Elements:

<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01 (Mac)</b></i>	<i><b>IE 4.5 (Mac)</b></i>	<i><b>IE 5.0 (Win. &amp; UNIX)</b></i>	<i><b>NN 4.x</b></i>	<i><b>NN 4.0 (Mac &amp; UNIX)</b></i>	<i><b>Opera 3.6</b></i>
Unsafe	Partial	Partial	Safe	Safe	Partial	Partial	Partial

## FONT

Description: This is an abbreviated element that allows the Web author to quickly set fonts in the manner possible under the FONT-FAMILY,

FONT-SIZE, FONT-STYLE, FONT-VARIANT and FONT-WEIGHT CSS elements.

## CSS Family Type: Font

Values:

- CURSIVE | FANTASY | MONOSPACE | SANS-SERIF | SERIF - Sets the type of visual characteristics of font “family” name that can be specified.
- *font\_family* - The name of the font to be displayed
- LARGE | MEDIUM | SMALL | X-LARGE | X-SMALL | XX-LARGE | XX-SMALL - Specifies the size of the text. Values range from XX-SMALL (smallest) to XX-LARGE (largest).
- LARGER | SMALLER - Changes the size of the current specified element relative to a previously specified value.
- *n* value - Specific unit value for the specified element.
- % value – Sets a percentage for the size of a font relative to the base font size.
- ITALIC | NORMAL - Determines whether or not text is italicized
- OBLIQUE - Sets an italic-like property if the font used is sans-serif.
- 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 - Absolute font weight values.
- BOLD | NORMAL - Sets whether or not the specified text element uses a bold face.
- BOLDER | LIGHTER - Changes the weight of the current specified element relative to a previously specified value.

Sample code:

**Fantasy font**

## Support in Major Browsers:

[illegible]

## FONT-FAMILY

Description: Sets the type of font to be displayed with the specified element.

CSS Family Type: Font

Values:

- CURSIVE | FANTASY | MONOSPACE | SANS-SERIF | SERIF - Sets the type of visual characteristics of the font.

Sample code:

```
<B STYLE="FONT-FAMILY: SERIF">This sentence is
displayed in a serif font.</B>
```

Support in Major Browsers:

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i>	<i>IE 4.5</i>	<i>(Win. &amp;</i>	<i>NN 4.x</i>	<i>(Mac &amp;</i>	<i>Opera</i>
		<i>(Mac)</i>	<i>(Mac)</i>	<i>UNIX)</i>		<i>UNIX)</i>	<i>3.6</i>
Partial	Safe	Safe	Safe	Safe	Safe	Safe	Safe

## FONT-SIZE

Description: Sets the display size of text.

CSS Family Type: Font

Values:

- LARGE | MEDIUM | SMALL | X-LARGE | X-SMALL | XX-LARGE | XX-SMALL - Sets the size of the text. Values range from XX-SMALL (smallest) to XX-LARGE (largest).
- LARGER | SMALLER - Changes the size of the current specified element relative to a previously specified value.
- n* value - Specific unit value for the specified element.
- % value - Sets a percentage for the size of a font relative to the base font size.

Sample code:

```
<H1 STYLE="FONT-SIZE: 48pt">Monster-Sized
Heading</H1>
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Partial	Partial	Partial	Partial	Partial	Safe	Safe	Safe

## FONT-STYLE

Description: Sets whether or not the displayed font is italicized.

CSS Family Type: Font

Values:

- **ITALIC | NORMAL** - Determines whether or not text is italicized
- **OBLIQUE** – To be used to create italicized text (used for fonts do not understand the term “italic”).

Sample code:

```
<I STYLE="FONT: OBLIQUE">This text is set to  
oblique. </I>
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Partial	Safe	Safe	Safe	Safe	Partial	Partial	Safe

## FONT-VARIANT

Description: Determines how text is displayed using capital letters.

CSS Family Type: Font

Values:

- **NORMAL | SMALL-CAPS**
- These values are toggles to turn the small-caps effect on and off.



Sample code:

<H1 STYLE="FONT-VARIANT: SMALL-CAPS">this text is  
displayed in small capital letters</H1>

## Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> <i>(Mac)</i>	<i>IE 4.5</i> <i>(Mac)</i>	<i>IE 5.0</i> <i>(Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0</i> <i>(Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Safe

## FONT-WEIGHT

Description: Sets the thickness of the displayed font.

CSS Family Type: Font

Values:

- 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 - Absolute font weight values.
- BOLD | NORMAL - Sets whether or not the specified text element uses a bold face.
- BOLDER | LIGHTER - Changes the weight of the current specified element relative to that of a previously specified value.

Sample code:

***<I STYLE="FONT: BOLD">This italicized text is also  
set to bold using CSS.</I>***

## Support in Major Browsers:

[illegible]

## HEIGHT

Description: This property scales an element (typically an image) to the specified height.

CSS Family Type: Boxes (Sub-Family: Padding)

Values:

- **AUTO** - Specifies that the browser default value should be used.
- **%** - Sets a percentage value of the element's width.
- ***length (units)*** - Specifies the length value as a unit of measurement.

Sample code:

```
<IMG SRC="philco_jnr.jpg" STYLE="HEIGHT: 2.5in">
```

Support in Major Browsers:

				<b>IE 5.0</b>		<b>NN 4.0</b>	
<b>IE 3.02</b>	<b>IE 4.x</b>	<b>IE 4.01</b>	<b>IE 4.5</b>	<b>(Win. &amp; UNIX)</b>	<b>NN 4.x</b>	<b>(Mac &amp; UNIX)</b>	<b>Opera 3.6</b>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Safe

## LETTER-SPACING

Description: Specifies the spacing value between letters.

CSS Family Type: Text

Values:

- **NORMAL** - The browser's default letter-spacing value.
- ***length (units)*** - Sets the length value between letters as a unit of measurement.

Sample code:

```
<B STYLE="LETTER-SPACING: 1cm">Some wide letter  
spacing!</B>
```



## LIST-STYLE

Description: This is an abbreviated element that enables Web authors to quickly set list item markers in the manner possible under LIST-STYLE-POSITION, LIST-STYLE-IMAGE and LIST-STYLE-TYPE.

CSS Family Type: Classification

Values:

- CIRCLE | DISC | SQUARE - Sets the symbol specified to appear before each list item.
- DECIMAL | LOWER-ALPHA | LOWER-roman | UPPER-ALPHA | UPPER-roman - Sets the type of alphanumeric symbol to appear before each list item.
- NONE - No list-marker is to be displayed before a list item.
- *URL (image)*- Specifies the URL of the graphic to be used as a list marker.
- INSIDE - Displays the text of a list item at a similar indentation level to the list-marker.
- OUTSIDE - Displays the text of a list item indented from the list-marker.

Sample code:

```
<UL STYLE="LIST-STYLE: SQUARE">
<LI>List items
<LI>preceded by
<LI>solid squares
</UL>
```

Support in Major Browsers:

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> <i>(Mac)</i>	<i>IE 4.5</i> <i>(Mac)</i>	<i>(Win. &amp;</i> <i>UNIX)</i>	<i>NN 4.x</i>	<i>(Mac &amp;</i> <i>UNIX)</i>	<i>Opera</i> <i>3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Safe

## LIST-STYLE-IMAGE

Description: Specifies an image to be used as the marker before a list element.

CSS Family Type: Classification

Values:

- **NONE** - No list-marker is displayed before a list item.
- ***URL(image)***- Specifies the URL of the graphic to be used as a list marker.

Sample code:

```
<UL>
<LI STYLE="LIST-STYLE-IMAGE: url(trajan.jpg)">
Emperor Trajan, 98-117 AD
<LI STYLE="LIST-STYLE-IMAGE: url(hadrian.jpg)">
Emperor Hadrian, 117-158 AD
</UL>
```

Support in Major Browsers:

				<i><b>IE 5.0</b></i>		<i><b>NN 4.0</b></i>	
<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01</b></i>	<i><b>IE 4.5</b></i>	<i><b>(Win. &amp;</b></i>	<i><b>NN 4.x</b></i>	<i><b>(Mac &amp;</b></i>	<i><b>Opera</b></i>
		<i><b>(Mac)</b></i>	<i><b>(Mac)</b></i>	<i><b>UNIX)</b></i>		<i><b>UNIX)</b></i>	<i><b>3.6</b></i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Safe

## LIST-STYLE-POSITION

Description: Specifies how a list-marker is rendered relative to the content of the list item itself.

CSS Family Type: Classification

Values:

- **INSIDE** - Displays the text of a list item at a similar indentation level to the list-marker.
- **OUTSIDE** - Displays the text of a list item indented from the list-marker.

Sample code:

```
<UL STYLE="LIST-STYLE-POSITION: INSIDE">
<LI>Notice how the text in the second line of this
bullet item is begins underneath the bullet rather
than directly under the first word.
</UL>
```



## MARGIN

**Description:** This is an abbreviated element that enables Web authors to quickly set margin values such as MARGIN-TOP, MARGIN-RIGHT, MARGIN-LEFT or MARGIN-BOTTOM.

**CSS Family Type:** Boxes (Sub-Family: Margin)

**Values:**

- **AUTO** - Specifies that the browser default value should be used.
- **%** - Sets a percentage value of the element's width.
- **n [ n n n ] measurement\_units**
- If 1 value is present, all borders are set to the numerical value specified
- If 2 values are present, the top/bottom and side borders take the numerical values specified
- If 3 values are present, the top, right *and* left, then bottom takes the numerical values specified
- If 4 values are present, the top, right, bottom then left borders take the numerical values specified

**Sample code:**

```
<H1 STYLE="MARGIN: 2in">This text has a margin set
to two inches.</H1>
```

**Support in Major Browsers:**

		<i><b>IE 4.01</b></i>	<i><b>IE 4.5</b></i>	<i><b>IE 5.0</b></i>		<i><b>NN 4.0</b></i>	
<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>(Mac)</b></i>	<i><b>(Mac)</b></i>	<i><b>(Win. &amp; UNIX)</b></i>	<i><b>NN 4.x</b></i>	<i><b>(Mac &amp; UNIX)</b></i>	<i><b>Opera 3.6</b></i>
Partial	Partial	Partial	Partial	Partial	Partial	Partial	Partial

## MARGIN-BOTTOM

**Description:** Specifies the distance of an element from the bottom edge of the browser window or from the next element that appears below it.

**CSS Family Type:** Boxes (Sub-Family: Margin)

**Values:**

- **AUTO** - Specifies that the browser default value should be used.





## MARGIN-RIGHT

Description: Specifies the distance of an element from the right edge of the browser window or from the next element that appears to its right.

CSS Family Type: Boxes (Sub-Family: Margin)

Values:

- **AUTO** - Specifies that the browser default value should be used.
- **%** - Sets a percentage value of the element's width.
- **length (units)** - Specifies the length value as a unit of measurement.

Sample code:

```
<IMG SRC="trajan.jpg" STYLE="MARGIN-RIGHT: 1IN">
This text is set 1 inch from the adjacent image.
```

Support in Major Browsers:

				<i><b>IE 5.0</b></i>		<i><b>NN 4.0</b></i>	
<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01</b></i>	<i><b>IE 4.5</b></i>	<i><b>(Win. &amp;</b></i>	<i><b>NN 4.x</b></i>	<i><b>(Mac &amp;</b></i>	<i><b>Opera</b></i>
		<i><b>(Mac)</b></i>	<i><b>(Mac)</b></i>	<i><b>UNIX)</b></i>		<i><b>UNIX)</b></i>	<i><b>3.6</b></i>
Partial	Safe	Safe	Safe	Safe	Safe	Safe	Partial

## MARGIN-TOP

Description: Specifies the distance of an element from the top edge of the browser window or from the element that appears above it.

CSS Family Type: Boxes (Sub-Family: Margin)

Values:

- **AUTO** - Specifies that the browser default value should be used.
- **%** - Sets a percentage value of the element's width.
- **length (units)** - Specifies the length value as a unit of measurement.

Sample code:

```
<H1 STYLE="MARGIN-TOP: 150PX">Header situated 150
pixels from the top margin.</H1>
```

## Support in Major Browsers:

				<i>IE 5.0</i>		<i>NN 4.0</i>	
		<i>IE 4.01</i>	<i>IE 4.5</i>	(Win. & UN <sup>IX</sup> )	<i>NN 4.x</i>	(Mac & UN <sup>IX</sup> )	<i>Opera 3.6</i>
<i>IE 3.02</i>	<i>IE 4.x</i>	(Mac)	(Mac)				
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Safe

## PADDING

Description: This is an abbreviated element that enables Web authors to quickly set margin values such as PADDING-TOP, PADDING-RIGHT, PADDING-LEFT or PADDING-BOTTOM.

### CSS Family Type: Boxes (Sub-Family: Padding)

Values:

- **AUTO** - Specifies that the browser default value should be used.
- **%** - Sets a percentage value of the element's width.
- **n [ n n n ] measurement\_units**
- If 1 value is present, all borders are set to the numerical value specified
- If 2 values are present, the top/bottom and side borders take the numerical values specified
- If 3 values are present, the top, right **and** left, then bottom takes the numerical values specified
- If 4 values are present, the top, right, bottom then left borders take the numerical values specified

Sample code:

This text is offset 1 cm right and below the image, and the image is itself offset from the margins of the browser window by the same amount.

## Support in Major Browsers:

[illegible]

## PADDING-BOTTOM

Sets the amount of padding to the bottom of an element.

CSS Family Type: Boxes (Sub-Family: Padding)

Values:

- *%* - Sets a percentage value of the element's width.
- *length (units)* - Specifies the length value as a unit of measurement.

Sample code:

```
<B STYLE="PADDING-BOTTOM: 2IN">This is padded from
the bottom (i.e., from the element below it) by 2
inches.</B>

<P>
Some extra text to demonstrate the effect.
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Partial

## PADDING-LEFT

Description: Sets the amount of padding to the left of an element.

CSS Family Type: Boxes (Sub-Family: Padding)

Values:

- *%* - Sets a percentage value of the element's width.
- *length (units)* - Specifies the length value as a unit of measurement.

Sample code:

```
<STRONG STYLE="PADDING-LEFT: 15PT">This is padded
from the left by 15 points.</STRONG>
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Partial

## PADDING-RIGHT

Description: Sets the amount of padding to the right of an element.

CSS Family Type: Boxes (Sub-Family: Padding)

Values:

- % - Sets a percentage value of the element's width.
- *length (units)* - Specifies the length value as a unit of measurement.

Sample code:

```
<STRONG STYLE="PADDING-RIGHT: 200PX">This text has
a padding of 200 pixels from the right. To
demonstrate the effect properly, you need a
another, long, rambling sentence that just goes on
and on and on and on...</STRONG>
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Partial

## PADDING-TOP

Description: Sets the amount of padding to the top of an element.

CSS Family Type: Boxes (Sub-Family: Padding)

Values:

- % - Sets a percentage value of the element's width.

- *length (units)* - Specifies the length value as a unit of measurement.

Sample code:

```
<B STYLE="PADDING-TOP: 50MM">This text is padded
from the top by 50 millimeters.</B>
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Partial

## TEXT-ALIGN

Description: Sets the alignment of the text.

CSS Family Type: Text

Values:

- **CENTER | JUSTIFY | LEFT | RIGHT** - Aligns the text contained by the specified element.

Sample code:

```
<P STYLE="TEXT-ALIGN: CENTER">Centered text</P>
<P STYLE="TEXT-ALIGN: JUSTIFY">Justified text</P>
<P STYLE="TEXT-ALIGN: LEFT">Left-aligned text</P>
<P STYLE="TEXT-ALIGN: RIGHT">Right-aligned text</P>
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Partial	Safe	Partial	Partial	Safe	Safe	Safe	Safe

## TEXT-DECORATION

Description: Adds a “decorative” property to the HTML tag specified.

CSS Family Type: Text

Values:

- **BLINK** - Text blinks.
- **LINE-THROUGH** - Text contains a line through the middle (strikethrough text).
- **NONE** - No decoration is added.
- **OVERLINE** - Text is displayed with a line running above it.
- **UNDERLINE** - Text is underlined.

Sample code:

```
<A HREF="fred.html" STYLE="TEXT-DECORATION:
NONE">This link is not underlined</A>
<H1 STYLE="TEXT-DECORATION: OVERLINE">Header with
an Overline</H1>
<H1 STYLE="TEXT-DECORATION: UNDERLINE">Header
with an Underline</H1>
<P STYLE="TEXT-DECORATION: LINE-THROUGH">
Ignore this sentence
<P>
<I STYLE="TEXT-DECORATION: BLINK">This text
blinks</I>
```

Support in Major Browsers:

				<i><b>IE 5.0</b></i>		<i><b>NN 4.0</b></i>	
<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01</b></i> <i><b>(Mac)</b></i>	<i><b>IE 4.5</b></i> <i><b>(Mac)</b></i>	<i><b>(Win. &amp;</b></i> <i><b>UNIX)</b></i>	<i><b>NN 4.x</b></i>	<i><b>(Mac &amp;</b></i> <i><b>UNIX)</b></i>	<i><b>Opera</b></i> <i><b>3.6</b></i>
Partial	Partial	Partial	Partial	Partial	Partial	Partial	Partial

## TEXT-INDENT

Description: Specifies indents from the left margin or from a block-element.

CSS Family Type: Text

Values:

- ***length (units)*** - Specifies the length value as a unit of measurement.



## VERTICAL-ALIGN

Description: Sets the vertical positioning of the HTML tag with which it is associated.

CSS Family Type: Text

Values:

- **BASELINE | BOTTOM | MIDDLE | SUB | SUPER | TEXT-TOP | TEXT-BOTTOM | TOP** - Aligns the text element in relation to the specified value.

Sample code:

```
<B STYLE="VERTICAL-ALIGN: SUB">Subscript-aligned
text</B> <I STYLE="FONT-SIZE: 16pt">Large, itali-
cized text.</I>
<B STYLE="VERTICAL-ALIGN: SUPER">Superscript-
aligned text</B> <I STYLE="FONT-SIZE: 16pt">Large,
italicized text.</I>
```

Support in Major Browsers:

				<i><b>IE 5.0</b></i>		<i><b>NN 4.0</b></i>	
<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01</b></i>	<i><b>IE 4.5</b></i>	<i><b>(Win. &amp;</b></i>	<i><b>NN 4.x</b></i>	<i><b>(Mac &amp;</b></i>	<i><b>Opera</b></i>
		<i><b>(Mac)</b></i>	<i><b>(Mac)</b></i>	<i><b>UNIX)</b></i>		<i><b>UNIX)</b></i>	<i><b>3.6</b></i>
Unsafe	Partial	Partial	Partial	Partial	Unsafe	Unsafe	Partial

## WHITE-SPACE

Description: This element is used to set white space and the way carriage returns are handled within a Web page.

CSS Family Type: Classification

Values:

- **NORMAL** - The default browser behavior for the HTML tag this element is associated with.
- **NOWRAP** - When this value is set, carriage returns and linefeeds are rendered as single space. However, line breaks are not effected.
- **PRE** - When this value is set, all spaces, carriage returns, and linefeeds in the document are displayed “as is”.



Sample code:

```
<P STYLE="WHITE-SPACE: NORMAL">
There   is no  extra white space   in
      this sentence.   It is   displayed normally.
</P>
```

```
<P STYLE="WHITE-SPACE: PRE">
The   extra white   spaces and carriage
      returns in   this sentence
          are displayed.
</P>
```

```
<P STYLE="WHITE-SPACE: NOWRAP">
The   extra white   space and carriage
      returns in   this sentence
          are <B>not</B> displayed.
</P>
```

Support in Major Browsers:

				<i><b>IE 5.0</b></i>		<i><b>NN 4.0</b></i>	
<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01</b></i>	<i><b>IE 4.5</b></i>	<i><b>(Win. &amp;</b></i>		<i><b>(Mac &amp;</b></i>	<i><b>Opera</b></i>
		<i><b>(Mac)</b></i>	<i><b>(Mac)</b></i>	<i><b>UNIX)</b></i>	<i><b>NN 4.x</b></i>	<i><b>UNIX)</b></i>	<i><b>3.6</b></i>
Unsafe	Unsafe	Unsafe	Unsafe	Unsafe	Partial	Partial	Unsafe

## **WIDTH**

Description: Scales an element (most typically an image) to the width specified.

CSS Family Type: Boxes

Values:

- **AUTO** - Specifies that the browser default value should be used.
- **%** - Sets a percentage value with respect to the browser window's width.
- ***length (units)*** - Specifies the length value as a unit of measurement.

Sample code:

```
<IMG STYLE="WIDTH: AUTO" SRC="gordian3.jpg"> Regular
size (AUTO setting)
<P>
<IMG STYLE="WIDTH: 25px" SRC="gordian3.jpg"> Set
to 25 pixels
<P>
<IMG STYLE="WIDTH: 25%" SRC="gordian3.jpg"> Set to
25% of the browser window's width
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Partial	Partial	Safe

## WORD-SPACING

Description: Sets the spacing between words.

CSS Family Type: Text

Values:

- **NORMAL** - The browser default word spacing value is used.
- ***length (units)*** - Sets the length value between words as a unit of measurement.

Sample code:

```
<I STYLE="WORD-SPACING: 1cm">The words in this
sentence are all spaced 1cm apart.</I>
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Unsafe	Safe	Safe	Unsafe	Unsafe	Unsafe	Safe

# Pseudo-Elements and Pseudo-Classes

## ANCHOR

Description: Determines how links are displayed.

Values:

- A:LINK – Determines how a “resting” (i.e., unclicked) link appears on a Web page.
- A:ACTIVE – Determines how a link should appear when clicked (or otherwise selected) by the user.
- A:VISITED – Determines how the link should appear after it has been visited.

Sample code:

```
<STYLE>
A:LINK {COLOR: FUCHSIA;}
A:ACTIVE {COLOR: OLIVE;}
A:VISITED {COLOR: BLACK;}
</STYLE>
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Unsafe	Unsafe	Partial

## FIRST-LINE

Description: Sets how the first line of text in a paragraph should be displayed.

Sample code:

```
<STYLE>
P:FIRST-LINE {FONT-VARIANT: SMALL-CAPS;
COLOR: NAVY;}
</STYLE>
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Unsafe	Unsafe	Unsafe	Unsafe	Unsafe	Unsafe	Safe

## FIRST-LETTER

Description: Sets how the very first letter in the first line of text in a paragraph should be displayed.

Sample code:

```
<STYLE>
P {FONT-SIZE: 12pt; LINE-HEIGHT: 12pt;}
P:FIRST-LETTER {FONT-SIZE: 18pt; FLOAT: LEFT;
COLOR: NAVY; FONT-WEIGHT: BOLD;}
</STYLE>
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Unsafe	Unsafe	Unsafe	Unsafe	Unsafe	Unsafe	Safe

## Implementation of Basic CSS Concepts

### Containment in HTML

Description: In order for browsers to understand and implement CSS formatting, they must be aware of the ways in which CSS elements can be combined within the HTML contained on a Web page.

Sample code:

```
<HTML>
<HEAD>
<TITLE>Containment in HTML</TITLE>
<LINK REL=STYLESHEET TYPE="text/css" HREF="http://
/www.mondoville.com/funky.css">
<STYLE>
@IMPORT URL(http://www.mondoville.com/not-as-
funky.css);
H1 {COLOR: RED}
</STYLE>
</HEAD>
<BODY>
<H1>This Header is Red</H1>
<P STYLE="COLOR: BLUE">The text in this paragraph
is blue.
</BODY>
</HTML>
```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Partial	Partial	Partial	Partial	Partial	Partial	Partial	Safe

## Grouping

Description: Grouping is designed to shorten the overall length of style sheet code by allowing Web authors to add several formatting elements together in a single, compact statement.

Sample code:

```
<STYLE>
H1, H2, H3 {FONT-FAMILY: HELVETICA;}
P {FONT-SIZE: 12pt; FONT-FAMILY: COURIER; FONT-
VARIANT: NORMAL; COLOR: NAVY;}
</STYLE>
```

Support in Major Browsers:

<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01 (Mac)</b></i>	<i><b>IE 4.5 (Mac)</b></i>	<i><b>IE 5.0 (Win. &amp; UNIX)</b></i>	<i><b>NN 4.x</b></i>	<i><b>NN 4.0 (Mac &amp; UNIX)</b></i>	<i><b>Opera 3.6</b></i>
None	Safe	Safe	Safe	Safe	Safe	Safe	Safe

## Inheritance

Description: A browser should allow “child” elements to take on the properties of a “parent” element.

Sample Code:

```
<P STYLE="FONT-SIZE: 14pt; FONT-STYLE: ITALIC;
COLOR: BLUE;">
```

This text in this paragraph should appear as in an italic, blue-colored, 14 point font. The only exception to this is **<B>this bold section</B>**. Note how the bold section (the "child") retains the formatting belonging to the rest of the paragraph (the "parent").

```
</P>
```

Support in Major Browsers:

<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01 (Mac)</b></i>	<i><b>IE 4.5 (Mac)</b></i>	<i><b>IE 5.0 (Win. &amp; UNIX)</b></i>	<i><b>NN 4.x</b></i>	<i><b>NN 4.0 (Mac &amp; UNIX)</b></i>	<i><b>Opera 3.6</b></i>
Unsafe	Safe	Partial	Safe	Safe	Safe	Safe	Safe

## CLASS as Selector

Description: CLASS is an HTML attribute that enables Web authors to “name” a set of formatting properties, which can then be called upon later in a Web page.

Sample Code:

```
<HTML>
<HEAD>
<TITLE>CLASS as Selector</TITLE>
<STYLE>
H1.RED {COLOR: RED;}
H1.GREEN {COLOR: GREEN;}
H1.BLUE {COLOR: #0000FF;}
</STYLE>
</HEAD>
<BODY>

<H1 CLASS="RED">A Red Header</H1>

<H1 CLASS="GREEN">A Green Header</H1>

<H1 CLASS="BLUE">A Blue Header</H1>

</BODY>
</HTML>
```

Support in Major Browsers:

		<i>IE 4.01</i>	<i>IE 4.5</i>	<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>(Mac)</i>	<i>(Mac)</i>	<i>(Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>(Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Partial	Partial	Partial	Partial	Safe	Safe	Safe

## ID as Selector

Description: ID is an HTML attribute that enables Web authors to assign a common type of formatting element that can be called upon by different HTML tags.

Sample Code:

```
<HTML>
<HEAD>
<TITLE>ID as Selector</TITLE>
<STYLE>
#GREEN {COLOR: GREEN; FONT-WEIGHT: BOLD;}
H1#GREEN { letter-spacing: 0.5em }
```

```

</STYLE>
</HEAD>
<BODY>

<H1>A Header</H1>

<P ID="GREEN">Here is some green text.</P>

<H2 ID="GREEN">A Green Header</H2>

<P>This text is not green or bold.</P>

</BODY>
</HTML>

```

Support in Major Browsers:

				<i><b>IE 5.0</b></i>		<i><b>NN 4.0</b></i>	
<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01 (Mac)</b></i>	<i><b>IE 4.5 (Mac)</b></i>	<i><b>(Win. &amp; UNIX)</b></i>	<i><b>NN 4.x</b></i>	<i><b>(Mac &amp; UNIX)</b></i>	<i><b>Opera 3.6</b></i>
Unsafe	Partial	Partial	Partial	Partial	Partial	Partial	Partial

---

## Contextual Selectors

Description: Web browsers that understand CSS should be able to selectively apply CSS formatting rules based on the context in which they appear.

Sample Code:

```

<HTML>
<HEAD>
<TITLE>Contextual selectors</TITLE>
<STYLE>
UL LI {FONT-SIZE: X-LARGE; COLOR: BLACK;}
UL UL LI {FONT-SIZE: LARGE; COLOR: GRAY;}
</STYLE>
</HEAD>
<BODY>

<UL>
<LI>An extra-large list item in a black font
  <UL>
    <LI>A large sub-list item in a grey font

```



```

        <LI>Another large sub-list item in a grey font
    </UL>
    <LI>Another extra-large list item in a black font
</UL>

</BODY>
</HTML>

```

Support in Major Browsers:

				<i><b>IE 5.0</b></i>		<i><b>NN 4.0</b></i>	
<i><b>IE 3.02</b></i>	<i><b>IE 4.x</b></i>	<i><b>IE 4.01</b></i>	<i><b>IE 4.5</b></i>	<i><b>(Win. &amp;</b></i>	<i><b>NN 4.x</b></i>	<i><b>(Mac &amp;</b></i>	<i><b>Opera</b></i>
		<i><b>(Mac)</b></i>	<i><b>(Mac)</b></i>	<i><b>UNIX)</b></i>		<i><b>UNIX)</b></i>	<i><b>3.6</b></i>
Safe	Safe	Safe	Safe	Safe	Safe	Safe	Safe

## Comments

Description: In order to explain how or why certain CSS values have been applied, it is necessary for Web authors to be able to add comments to the CSS code.

Sample Code:

```

<HTML>
<HEAD>
<TITLE>Comments</TITLE>
<STYLE>
H1 {FONT-SIZE: X-LARGE; COLOR: BLACK;
BACKGROUND: GRAY;}
/* Sets the formatting for the top-level headers */
</STYLE>
</HEAD>
<BODY>
<H1>Large, Funky-Looking Header</H1>

<H1 STYLE="FONT-SIZE: LARGE; COLOR: #333333;
BACKGROUND: YELLOW;"
/* Regular top-level header formatting style has
been over-ruled in this case */>
Another, Smaller Header</H1>

</BODY>
</HTML>

```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Unsafe	Safe	Safe	Safe	Safe	Safe	Safe	Safe

## The Cascade

### *Cascading Order*

Description: The cascading order selects specific selectors over more general ones, and if there are two CSS rules with the same weight, the last one specified is the one that is used.

Sample Code:

```
<HTML>
<HEAD>
<TITLE>Cascading Order</TITLE>
<LINK REL="stylesheet" TYPE="text/css"
HREF="one.css">
<LINK REL="stylesheet" TYPE="text/css"
HREF="two.css">
<STYLE>
LI {COLOR: OLIVE;}
OL LI {COLOR: NAVY;}
UL LI {COLOR: YELLOW;}
LI.LIME {COLOR: LIME;}
OL LI#MAR {COLOR: MAROON;}
.ORDER {COLOR: BLUE;}
.ORDER {COLOR: SILVER;}
.DECORATION {TEXT-DECORATION: LINE-THROUGH;}
</STYLE>
</HEAD>

<BODY STYLE="FONT-SIZE: LARGE;
FONT-COLOR: BLACK">
```

### Support in Major Browsers:

[illegible]

## !!IMPORTANT

Description: Web authors can increase the weight of a particular CSS declaration by using “!IMPORTANT” in their code.

Sample Code:

```
<HTML>
<HEAD>
<TITLE>!IMPORTANT</TITLE>
<STYLE>
P {COLOR: BLUE !IMPORTANT;}
P {COLOR: RED;}
P.FUCHSIA {COLOR: FUCHSIA;}
</STYLE>
</HEAD>
<BODY>

<P>
This line of text should be in blue.

<P STYLE="COLOR: RED;">
This line of text should also be in blue, even
though it is set to red.

<P CLASS="FUCHSIA">
This line of text should also be in blue, even
though it is set to fuchsia.

</BODY>
</HTML>
```

Support in Major Browsers:

				<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01</i> <i>(Mac)</i>	<i>IE 4.5</i> <i>(Mac)</i>	<i>(Win. &amp;</i> <i>UNIX)</i>	<i>NN 4.x</i>	<i>(Mac &amp;</i> <i>UNIX)</i>	<i>Opera</i> <i>3.6</i>
Unsafe	Safe	Unsafe	Unsafe	Safe	Unsafe	Unsafe	Safe

---



# **CSS2 PROPERTY, PSEUDO-ELEMENT AND PSEUDO- CLASS REFERENCE**



# *Appendix B*

## Alphabetical Listing of CSS2 Elements

### ***AZIMUTH***

Description: Sets the “angle” in which a sound is positioned in a sound space surrounding the viewer.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Spatial properties)

Values:

- *n* DEG - The angle in degrees in which a sound is located. The range of the angle is from 0DEG to 360DEG. In this scheme, a value of 0DEG is directly ahead of the viewer, 180DEG is behind the viewer, 90DEG is to the right, and 270DEG is to the left. These values can also be expressed as negative values, so 90DEG is equivalent to -270DEG and 270DEG is the same as -90DEG.
- LEFT-SIDE - Equivalent to 270DEG. If BEHIND is also set, its value does not change.

- FAR-LEFT - Equivalent to 300DEG. If BEHIND is also set, the value becomes 240DEG.
- LEFT - Same as 320DEG. If BEHIND is also set, the value becomes 220DEG.
- CENTER-LEFT - Equivalent to 340DEG. If BEHIND is also set, the value becomes 200DEG.
- CENTER - Equivalent to 0DEG. If BEHIND is also set, the value becomes 180DEG.
- CENTER-RIGHT - Equivalent to 20DEG. If BEHIND is also set, the value becomes 160DEG.
- RIGHT - Equivalent to 40DEG. If BEHIND is also set, the value becomes 140DEG.
- FAR-RIGHT - Equivalent to 60DEG. If BEHIND is also set, the value becomes 120DEG.
- RIGHT-SIDE - Equivalent to 90DEG. If BEHIND is also set, its value does not change.
- BEHIND - Equivalent to 180DEG. Can also be used to modify other named values, taking a position that was in front and “mirroring” it to the same location behind the viewer.
- LEFTWARDS - Moves the sound 20 degrees to the left (counterclockwise) to its current position.
- RIGHTWARDS - Moves the sound 20 degrees to the right (clockwise) to its current position.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P AZIMUTH: LEFT>Hamlet: How does the Queen?<P>
<P AZIMUTH: RIGHT>King: She sounds to see them
bleed. </P>
<P AZIMUTH: FAR-RIGHT>Queen: No, no! the drink,
the drink! O my dear Hamlet!
The drink, the drink! I am poison'd. </P>
```

## BACKGROUND

Description: Sets the background color or image. It can take on any value contained in the background family of properties.

Media Group: Visual

CSS2 Family Type: Colors and backgrounds



Values:

- *color name* or *numerical color value*
- URL(image) - Points to an image file in a format supported by the browser.
- SCROLL - Image moves when the browser window is scrolled.
- FIXED - Image does not move when the browser window is scrolled.
- TRANSPARENT - Renders the specified element transparent.
- X% Y% - Represents a position as a percentage of the dimensions of the browser window display.
- X Y - Represents an absolute coordinate position of the image.
- (LEFT/CENTER/RIGHT) | (TOP/CENTER/BOTTOM) - Keywords representing screen positions. Left keyword is the X-position and the right keyword is the Y-position for the image.
- REPEAT - Image is horizontally and vertically tiled.
- REPEAT-X - Image is horizontally tiled.
- REPEAT-Y - Image is vertically tiled.
- NO-REPEAT - The image is not repeated.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<B STYLE="BACKGROUND: GREEN">This text is displayed  
against a green background.</B>
```

## BACKGROUND-ATTACHMENT

Description: If BACKGROUND-IMAGE element is set, this element specifies how the background image should move when the browser window is scrolled.

Media Group: Visual

CSS2 Family Type: Colors and backgrounds

Values:

- SCROLL - Image moves when the browser window is scrolled.
- FIXED - Image does not move when the browser window is scrolled.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<BODY STYLE="BACKGROUND-IMAGE: URL(canvas.gif);  
BACKGROUND-ATTACHMENT: FIXED">
```

## ***BACKGROUND-COLOR***

Description: Sets the background color of an element.

Media Group: Visual

CSS2 Family Type: Colors and backgrounds

Values:

- *color name* or numerical color value
- TRANSPARENT - Renders the specified element transparent.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<I STYLE="BACKGROUND-COLOR: FFFF00">Italicized  
text on a yellow background.</I>
```

## ***BACKGROUND-IMAGE***

Description: Sets a graphic image as a background image.

If this property is specified, the CSS elements BACKGROUND-REPEAT, BACKGROUND-ATTACHMENT and BACKGROUND-POSITION can also be used with it. A color value can also be added.

Media Group: Visual

CSS2 Family Type: Colors and backgrounds

Values:

- URL(image) - Points to an image file in a format supported by the browser.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<BODY STYLE="BACKGROUND-IMAGE: URL(canvas.gif)">
```

## ***BACKGROUND-POSITION***

Description: If BACKGROUND-IMAGE element is specified, this element specifies where the image first appears, and then describes how it should be tiled.

Media Group: Visual

CSS2 Family Type: Colors and backgrounds

Values:

- X% Y% - Represents a position as a percentage of the dimensions of the browser window display.
- X Y - Represents an absolute coordinate position of the image.
- (LEFT/CENTER/RIGHT) | (TOP/CENTER/BOTTOM) - Keywords representing screen positions. Left keyword is the X-position and the right keyword is the Y-position for the image.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<BODY STYLE="BACKGROUND-IMAGE:
URL(victorinus.jpg); BACKGROUND-POSITION: 100 50">
```

## ***BACKGROUND-REPEAT***

Description: If the BACKGROUND-IMAGE element is set, this additional property specifies how the image is repeated on the Web page.

Media Group: Visual

CSS2 Family Type: Colors and backgrounds

Values:

- REPEAT - Image is horizontally and vertically tiled.
- REPEAT-X - Image is horizontally tiled.
- REPEAT-Y - Image is vertically tiled.
- NO-REPEAT - The image is not repeated.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<BODY STYLE="BACKGROUND-IMAGE: URL(gordian3.jpg);
BACKGROUND-REPEAT: REPEAT-X">
```

## ***BORDER***

Description: An abbreviated element that allows Web authors to specify BORDER-TOP, BORDER-RIGHT, BORDER-BOTTOM and BORDER-LEFT elements easily.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units
- Sets the thickness of the border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.
- *color name* or *numerical color value* - Sets the color for the border.
- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.
- HIDDEN - Same effect as NONE. It is designed to suppress border displays when used in tables.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<H1 STYLE="BORDER: OUTSET BLUE">A prominent  
header</H1>
```

## ***BORDER-BOTTOM***

Descriptions: Sets the display value for the bottom border of the specified HTML tag.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units
- Sets the thickness of the border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.
- *color name* or *numerical color value* - Sets the color for the border.
- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.
- HIDDEN - Same effect as NONE. It is designed to suppress border displays when used in tables.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<P STYLE="BORDER-BOTTOM: GROOVE GREEN">Paragraph
surrounded by grooved green border.</P>
```

## ***BORDER-BOTTOM-COLOR***

Description: Specifies the color for the bottom border.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *color name* or *numerical color value* - Sets the color for the border.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<H3 STYLE="BORDER-BOTTOM-COLOR: LIME">Third-Level
Header with a Lime Colored Bottom Border</H3>
```

## ***BORDER-BOTTOM-STYLE***

Description: Sets the type of bottom border to be displayed.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- **DASHED** - A dashed border line is displayed.
- **DOTTED** - A dotted border line is displayed.
- **DOUBLE** - A double-line is displayed.
- **INSET** - A 3D inset line is displayed.
- **GROOVE** - A 3D grooved line is displayed.
- **NONE** - No border is displayed, no matter what **BORDER-WIDTH** value is present.
- **OUTSET** - A 3D outset line is displayed.
- **RIDGE** - A 3D ridged line is displayed.
- **SOLID** - A solid border line is displayed.
- **HIDDEN** - Same effect as **NONE**. It is designed to suppress border displays when used in tables.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<P STYLE="BORDER-BOTTOM-STYLE: SOLID">Paragraph  
with a solid bottom border.</P>
```

## ***BORDER-BOTTOM-WIDTH***

Description: Sets the thickness of the bottom border.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- ***n*** measurement-units
- Sets the thickness of the bottom border.
- **THIN** | **MEDIUM** | **THICK** - Sets the relative thickness of the border.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<H1 STYLE="BORDER-BOTTOM-WIDTH: THIN">Header with  
a thin bottom border</H1>
```

## ***BORDER-COLLAPSE***

Description: Selects the table border to be used.

Media Group: Visual

CSS2 Family Type: Tables

Values:

- COLLAPSE - Table is set to the collapsing borders model.
- SEPARATE - Table is set to the separated borders model.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<HTML>  
<HEAD>  
<TITLE>Border-Collapse</TITLE>  
<STYLE>  
TABLE {BORDER: OUTSET 15PT; BORDER-COLLAPSE:  
SEPARATE; BORDER-SPACING: 15PT;}  
TD {BORDER: INSET 15PT; FONT-SIZE: XX-LARGE;}  
</STYLE>  
</HEAD>  
<BODY>  
<TABLE>  
<TR>  
<TD>Cell 1</TD>  
<TD>Cell 2</TD>  
<TD>Cell 3</TD>  
</TR>  
</TABLE>
```

## ***BORDER-COLOR***

Description: Sets the color of the border sides.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *color name* or *numerical color value* - Sets the color for the border.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.
- TRANSPARENT - Renders the border transparent.

Sample code:

```
<H1 STYLE="BORDER-COLOR: MAROON">Top-Level Header
with a Maroon Colored Border</H1>
```

## BORDER-LEFT

Description: Specifies how the left border associated with an HTML tag should be displayed.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement-units
- Sets the thickness of the border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.
- *color name* or *numerical color value* - Sets the color for the border.
- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<H1 STYLE="BORDER-LEFT: 1IN TEAL">Check out the
thick teal border to the left</H1>
```



## BORDER-LEFT-COLOR

Description: Specifies the color for the left border.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *color name* or *numerical color value* - Sets the color for the border.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<H2 STYLE="BORDER-LEFT-COLOR: SILVER">Second-Level  
Header with a Silver Colored Left Border</H2>
```

## BORDER-LEFT-STYLE

Description: Sets the type of left border to be displayed.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.
- HIDDEN - Same effect as NONE. It is designed to suppress border displays when used in tables.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<P STYLE="BORDER-LEFT-STYLE: INSET">Paragraph with  
an inset left border.</P>
```

## BORDER-LEFT-WIDTH

Sets the thickness of the left border.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement-units
- Sets the thickness of the left border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<P STYLE="BORDER-LEFT-WIDTH MEDIUM">Paragraph with  
a medium border to the left.</P>
```

## BORDER-RIGHT

Description: Specifies how the right border associated with an HTML tag should be displayed.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units
- Sets the thickness of the border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.
- *color name* or *numerical color value* - Sets the color for the border.
- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.

- **HIDDEN** - Same effect as **NONE**. It is designed to suppress border displays when used in tables.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<BLOCKQUOTE STYLE="BORDER-RIGHT: THIN INSET  
YELLOW">"This quote stands out with its right  
border."</BLOCKQUOTE>
```

## ***BORDER-RIGHT-COLOR***

Description: Specifies the color for the right border.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *color name* or *numerical color value* - Sets the color for the border.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<H2 STYLE="BORDER-RIGHT-COLOR: TEAL">Top-Level  
Header with a Teal Colored Bottom Border</H1>
```

## ***BORDER-RIGHT-STYLE***

Description: Sets the type of right border to be displayed.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- **DASHED** - A dashed border line is displayed.
- **DOTTED** - A dotted border line is displayed.
- **DOUBLE** - A double-line is displayed.
- **INSET** - A 3D inset line is displayed.
- **GROOVE** - A 3D grooved line is displayed.
- **NONE** - No border is displayed, no matter what **BORDER-WIDTH** value is present.
- **OUTSET** - A 3D outset line is displayed.

- **RIDGE** - A 3D ridged line is displayed.
- **SOLID** - A solid border line is displayed.
- **HIDDEN** - Same effect as **NONE**. It is designed to suppress border displays when used in tables.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<P STYLE="BORDER-RIGHT-STYLE: RIDGE">Paragraph
with a ridged right border.</P>
```

## ***BORDER-RIGHT-WIDTH***

Description: Sets the thickness of the right border.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units
- Sets the thickness of the right border.
- **THIN** | **MEDIUM** | **THICK** - Sets the relative thickness of the border.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<TT STYLE="BORDER-RIGHT-WIDTH 1IN">Teletype script
with a 1 inch thick right border.</TT>
```

## ***BORDER-SPACING***

Description: Sets the distance separating adjacent cell borders. If one value is set, it is used for both height and width. If two values are set, the first is taken for horizontal spacing, while the second is used for vertical spacing.

Media Group: Visual

CSS2 Family Type: Tables

Values:

- *n* [*n*] - units measurement
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<HTML>
<HEAD>
<TITLE>Border-Spacing</TITLE>
<STYLE>
TABLE {BORDER: OUTSET 5PT; BORDER-COLLAPSE: SEPA-
RATE; BORDER-SPACING: 10PT;}
TD {BORDER: INSET 5PT;}
</STYLE>
</HEAD>
<BODY>
<TABLE>
<TR>
<TD>Cell 1</TD>
<TD>Cell 2</TD>
<TD>Cell 3</TD>
</TR>
</TABLE>
```

## ***BORDER-STYLE***

Description: Sets the type of border to be displayed.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.
- HIDDEN - Same effect as NONE. It is designed to suppress border displays when used in tables.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<H2 STYLE="BORDER-STYLE: SOLID">Header with Solid
Border.</H2>
```

## ***BORDER-TOP***

Description: Specifies how the top border associated with an HTML tag should be displayed.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units
- Sets the thickness of the border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.
- *color name* or *numerical color value* - Sets the color for the border.
- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.
- HIDDEN - Same effect as NONE. It is designed to suppress border displays when used in tables.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<H2 STYLE="BORDER-TOP: THIN GROOVE AQUA">Header
with a blue groovy border</H2>
```

## ***BORDER-TOP-COLOR***

Description: Specifies the color for the top border.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *color name* or *numerical color value* - Sets the color for the border.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<H1 STYLE="BORDER-TOP-COLOR: MAROON">Top-Level  
Header with a Maroon Colored Top Border</H1>
```

## ***BORDER-TOP-STYLE***

Description: Sets the type of top border to be displayed.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.
- HIDDEN - Same effect as NONE. It is designed to suppress border displays when used in tables.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<P STYLE="BORDER-TOP-STYLE: DOTTED">Paragraph with  
a dotted top border.</P>
```

## ***BORDER-TOP-WIDTH***

Description: Sets the thickness of the top border.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *n* measurement units
- Sets the thickness of the right border.
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<TT STYLE="BORDER-TOP-WIDTH 25PX">Teletype script
with a 25 pixel thick top border.</TT>
```

## ***BORDER-WIDTH***

Description: Sets the thickness of the border for the specified element. One to four border sides can be set.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- *n* [ *n n n n* ] measurement\_units
- Sets the thickness of the border.
- If 1 value is present, all borders are set to the numerical value specified
- If 2 values are present, the top/bottom and side borders take the numerical values specified
- If 3 values are present, the top, right and left, then bottom take the numerical values specified
- If 4 values are present, the top, right, bottom then left borders take the numerical values specified
- THIN | MEDIUM | THICK - Sets the relative thickness of the border.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.



Sample code:

```
<H1 STYLE="BORDER-WIDTH: 5MM 5MM 15MM 1CM">A Header  
with Surrounding Border of Different Widths</H1>
```

## ***BOTTOM***

Description: Specifies the value that the current Web element is positioned above the containing box.

Media Group: Visual

CSS2 Family Type: Visual formatting

Values:

- *n* value - Sets a length unit value.
- % value – Sets a length percentage value relative to the width of the containing block.
- AUTO - The default value for the specific element.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="POSITION: ABSOLUTE; BOTTOM: 1IN;">This  
header is positioned 1 inch from the containing  
box, which in this case is the bottom of the  
browser window.</P>
```

## ***CAPTION-SIDE***

Description: Specifies where a caption will be in relation to a table.

Media Group: Visual

CSS2 Family Type: Tables

Values:

- TOP - Positions the caption above the table.
- BOTTOM - Positions the caption below the table.
- LEFT - Positions the caption to the left side of the table.
- RIGHT - Positions the caption to the right side of the table.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<TABLE BORDER>
<CAPTION STYLE="CAPTION-SIDE: LEFT;">A Simple
Table</CAPTION>
<TR>
<TD>This text is in the table</TD>
</TR>
</TABLE>
```

## ***CLEAR***

Description: Sets whether the specified element will allow other elements to “float” along its sides.

Media Group: Visual

CSS2 Family Type: Visual formatting

Values:

- **BOTH** - The specified element will be moved below floating elements on either side.
- **LEFT** - The specified element will be moved below floating elements on the left side only.
- **NONE** - Any floating elements are allowed on either side of the specified element.
- **RIGHT** - The specified element will be moved below floating elements on the right side only.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<B STYLE="CLEAR: BOTH">This text should appear
below the image.</B>
```

## ***CLIP***

Description: Defines how much of a Web element is made visible.

Media Group: Visual

CSS2 Family Type: Visual Effects

Values:

- **RECT***[n n n n]* - A rectangle defined by units of length in the following order: top, right, bottom and left. Negative values are permitted. Values are relative to the size of the Web element.
- **AUTO** - Sets the element to the regular, default visible value for that element. Can be used in conjunction with **RECT**.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1>The letters in this header are not "clipped".</H1>
<P>
<H1 STYLE="CLIP: RECT(10PX 5PX 10PX 5PX);">The letters in this header will appear "clipped".</H1>
```

## COLOR

Description: Sets the color to be displayed.

Media Group: Visual

CSS2 Family Type: Colors and backgrounds

Values:

- *color name* or *numerical color value*
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<UL>
<LI STYLE="COLOR: FF0000">This is displayed as red text
<LI STYLE="COLOR: 00FF00">This is displayed as green text
<LI STYLE="COLOR: 0000FF">This is displayed as blue text
</UL>
```

## CONTENT

Description: Determines the type of content, and subsequently, how it is to be displayed. Is always used in conjunction with the BEFORE and AFTER pseudo-classes.

Media Group: All

CSS2 Family Type: Generated content, automatic numbering and lists

Values:

- **STRING** - Refers to regular textual content.
- **URL("filename")** - Points to specific Web content (most typically an image or sound file) in a format supported by the browser.
- **COUNTER-INCREMENT | COUNTER-RESET** - These two counter properties can be used in conjunction with CONTENT to set a counter value for the associated element.
- **ATTR(X)** - Returns the string of the value of attribute X for the subject of the selector.
- **OPEN-QUOTE | CLOSE-QUOTE** - These two values retrieve and insert the string previously set by the QUOTES property.
- **NO-OPEN-QUOTE | NO-CLOSE-QUOTE** - These two values insert nothing, but do increment (or decrement) the nesting level for quotes.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<HTML>
<HEAD>
<STYLE>
  IMG:before {CONTENT: URL("This_Is_An_Image.wav")}
</STYLE>
</HEAD>
<BODY>
  All of the images on this Web page are preceded by
  an audio introduction, announcing them to be
  images, primarily for the benefit of those using
  audio browsers.
  <IMG SRC="image1.gif">
  <IMG SRC="image2.gif">
</BODY>
</HTML>
```

## COUNTER-INCREMENT

Description: Defines the property to be automatically incremented, and the amount by which it is to be incremented.

Media Group: All

CSS2 Family Type: Generated Content, automatic numbering and lists

Values:

- **<IDENTIFIER> <INTEGER>** - An identifier that selects the element to be incremented, and by how much.
- **NONE** - The default value.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<HTML>
<HEAD>
<TITLE>Counter-Increment and Counter-Reset</
TITLE>
<STYLE>
H1:BEFORE {CONTENT: "Floor " counter(floor) ". ";
COUNTER-INCREMENT: floor; COUNTER-RESET: room;}
H2:BEFORE {CONTENT: counter(floor) "."
counter(room) " "; COUNTER-INCREMENT: room; }
</STYLE>
</HEAD>
<BODY>
<H1>Basement</H1>
Storage space
<H1>Main Floor</H1>
Reception
<H2>Restaurant</H2>
Frank's Fish & Steak House
</BODY>
</HTML>
```

## COUNTER-RESET

Description: Resets the numerical value set by the COUNTER-INCREMENT property.

Media Group: All

CSS2 Family Type: Generated content, automatic numbering and lists

Values:

- **<IDENTIFIER> <INTEGER>** - An identifier that selects the element to be incremented, and by how much.
- **NONE** - The default value.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<HTML>
<HEAD>
<STYLE>
<TITLE>Counter-Increment and Counter-Reset</
TITLE>
<STYLE>
H1:BEFORE {CONTENT: "Floor " counter(floor) ". ";
COUNTER-INCREMENT: floor; COUNTER-RESET: room;}
H2:BEFORE {CONTENT: counter(floor) "."
counter(room) " "; COUNTER-INCREMENT: room; }
</STYLE>
</HEAD>
<BODY>
<H1>Basement</H1>
Storage space
<H1>Main Floor</H1>
Reception
<H2>Restaurant</H2>
Frank's Fish & Steak House
</BODY>
</HTML>
```

## CUE

Description: Adds a sound to a Web element. If a single value is present, the sound is played before and after the element; if two are present, the first sound is played before and the second sound after the element.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Cue properties)

Values:

- **URL(sound)** - Points to a sound file in a format supported by the browser.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
A single sound appears before and after this <A
HREF="link.html" STYLE="CUE: BADA-
BING.AIFF;">link</A>.
<P>
Two different sounds appear before and after this
<A HREF="link.html" STYLE="CUE: BADA-BING.AIFF
BADA-BOOM.AIFF;">link</A>.
```

## CUE-AFTER

Description: Adds a sound to be played immediately *after* a Web element.  
Good for aurally distinguishing significant parts on a Web page.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Cue properties)

Values:

- URL(sound) - Points to a sound file in a format supported by the browser.
- NONE - No sound is played.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
You can get there by <A HREF="link.html"
STYLE="CUE-AFTER; CLICK.WAV">Clicking here</A>.
```

## CUE-BEFORE

Description: Adds a sound to be played immediately *before* a Web element.  
Good for aurally distinguishing significant parts on a Web page.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Cue properties)

Values:

- URL(sound) - Points to a sound file in a format supported by the browser.
- NONE - No sound is played.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="CUE-BEFORE: BONG.AU;">Hamlet: Act I,  
Scene I</H1>
```

## CURSOR

Description: Defines the type of cursor to be displayed.

Media Group: Visual, Interactive

CSS2 Family Type: User interface

Values:

- **URL** - Points to specific Web content. As the user passes over the link, it will change to the selected cursor type.
- **AUTO** - Browser displays the type of cursor normally associated with the Web element.
- **CROSSHAIR** - A symbol resembling a “+” is displayed.
- **DEFAULT** - The default cursor type is displayed.
- **POINTER** - A pointer cursor (typically a small, upper-left pointing arrow) is displayed.
- **MOVE** - Cursor indicating that a Web element is to be moved.
- **E-RESIZE | NE-RESIZE | NW-RESIZE | N-RESIZE | SE-RESIZE |**
- **SW-RESIZE | S-RESIZE | W-RESIZE** - Cursor associated when an edge or corner that can be moved.
- **TEXT** - Indicates that text can be selected. Typically rendered as an I-bar.
- **WAIT** - Indicates that the program is busy and that the user should be patient while the operation is completed. Often an hourglass.
- **HELP** - Online help is available for the object under the cursor. Typically rendered as a question mark with pointer or as a balloon.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
Take aim and fire! <A HREF="link.html"><IMG  
SRC="shooting_gallery.gif" STYLE="CURSOR:  
CROSSHAIR;"></A>
```



## ***DIRECTION***

Description: Sets the direction for the letters in the text to follow.

Media Group: Visual

CSS2 Family Type: Visual formatting

Values:

- LTR - Text is displayed from left to right.
- RTL - Text is displayed from right to left.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="DIRECTION: LTR;">This sentence is displayed normally.</P>  
<P STYLE="DIRECTION: RTL;">.yllamron deyalpsid si ecnetnes sihT</P>
```

## ***DISPLAY***

Description: Controls the fundamental nature of the specified HTML tag.

Media Group: All

CSS2 Family Type: Visual formatting

Value:

- BLOCK - Sets the specified element as a block-type element.
- COMPACT | RUN-IN - Creates either a block or inline box, depending on context.
- INLINE - Sets the specified element as an inline-type element.
- LIST-ITEM - Sets the specified element as a type of list item.
- MARKER - This value sets the generated content before or after the box to be a marker. Should be used in conjunction with the :BEFORE and :AFTER pseudo-classes.
- NONE - Switches off the display of the specified element. It does not just create an invisible box; it creates no box at all.
- TABLE | TABLE-CAPTION | TABLE-CELL | TABLE-COLUMN | TABLE-COLUMN-GROUP | TABLE-FOOTER-GROUP | TABLE-HEADER-GROUP | TABLE-ROW | TABLE-ROW-GROUP | INLINE-TABLE - These values cause the selected element to behave like the specific table element specified.

- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<P STYLE="DISPLAY: NONE">This paragraph will not be
displayed.</P>
```

## ELEVATION

Description: Sets the “angle” in which a sound is played in a sound space that runs from below to above the viewer’s position.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Spatial properties)

Values:

- *n* DEG - The angle in degrees in which a sound is located. The range of the angle is from -90DEG to 90DEG. In this scheme, a value of 0DEG is directly in front of the viewer, 90DEG is directly above the viewer and -90DEG is directly below the viewer.
- BELOW - Equivalent to -90DEG.
- LEVEL - Equivalent to 0DEG.
- ABOVE - Equivalent to 90DEG.
- HIGHER - Adds 10 degrees to the current elevation, in effect, “raising” it.
- LOWER - Subtracts 10 degrees to the current elevation, in effect, “lowering” it.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="ELEVATION: -20DEG;">
Gravedigger: A pestilence on him for a mad rogue!
'A pour'd a flagon of Rhenish on my head once.
This same skull, sir, was Yorick's skull, the
King's jester.
</P>
<P STYLE="ELEVATION: 20DEG;">
Hamlet: This?
</P>
<P STYLE="ELEVATION: -20DEG;">
Gravedigger: E'en that.
</P>
```

```
<P STYLE="ELEVATION: LOWER;">
Hamlet: Let me see.
</P>
<P STYLE="ELEVATION: CENTER;">
[Takes the skull.]
</P>
<P STYLE="ELEVATION: HIGHER;">
Alas, poor Yorick! I knew him, Horatio. A fellow
of infinite jest, of most excellent fancy.
</P>
```

## EMPTY-CELLS

Description: This property is used to control how table cells containing no content are to be displayed.

Media Group: Visual

CSS2 Family Type: Tables

Values:

- SHOW - A border is drawn around the empty cell (the default table value).
- HIDE - No border is drawn around the empty cell.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<TABLE STYLE="EMPTY-CELLS: HIDE;" BORDER>
<TR>
<TD>No border should be drawn around the adjacent
cell.</TD>
<TD></TD>
</TABLE>
```

## FLOAT

Description: Sets how elements can wrap around another, “parent” element as it floats on a Web page.

Media Group: Visual

CSS2 Family Type: Visual formatting

Values:

- NONE - Other elements cannot be displayed to float around the specified element.
- LEFT - Other elements can be displayed to the left margin of the specified element.
- RIGHT - Other elements can be displayed to the right margin of the specified element.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<IMG SRC="aurelian.jpg" STYLE="FLOAT: RIGHT">This
text floats to the left of the image.
```

## FONT

Description: An abbreviated element that allows Web authors to quickly set fonts in the manner possible under the FONT-FAMILY, FONT-SIZE, FONT-STYLE, FONT-VARIANT and FONT-WEIGHT CSS elements.

Media Group: Visual

CSS2 Family Type: Font

Values:

- CURSIVE | FANTASY | MONOSPACE | SANS-SERIF | SERIF - Sets the type of visual characteristics of font “family” name that can be specified.
- *font\_family* - The name of the font to be displayed.
- LARGE | MEDIUM | SMALL | X-LARGE | X-SMALL | XX-LARGE | XX-SMALL - Specifies the size of the text. Values range from XX-SMALL (smallest) to XX-LARGE (largest).
- LARGER | SMALLER - Changes the size of the current specified element relative to a previously specified value.
- *n* value - Specific unit value for the specified element.
- % value - Sets a percentage for the size of a font relative to the base font size.
- ITALIC | NORMAL - Determines whether or not text is italicized.
- OBLIQUE - Sets an italic-like property if the font used is sans-serif.
- 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 - Absolute font weight values.

- **BOLD | NORMAL** - Sets whether or not the specified text element uses a bold face.
- **BOLDER | LIGHTER** - Changes the weight of the current specified element relative to a previously specified value.
- **CAPTION** - Sets the system font used for captioned controls, such as buttons.
- **ICON** - Sets the system font used to label icons.
- **MENU** - Sets the system font used in menus, such as drop-down menus.
- **MESSAGE-BOX** - Sets the system font used for dialog boxes.
- **SMALL-CAPTION** - Sets the system font used for labeling small controls.
- **STATUS-BAR** - Sets the system font used for window status bars.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<FORM STYLE="FONT: MENU ITALIC;">
<SELECT NAME="emperors">
<OPTION>Augustus
<OPTION>Tiberius
<OPTION>Caligula
<OPTION>Claudius
<OPTION>Nero
</SELECT>
</FORM>
```

## FONT-FAMILY

Description: Sets the type of font to be displayed with the specified element.

Media Group: Visual

CSS2 Family Type: Font

Values:

- **CURSIVE | FANTASY | MONOSPACE | SANS-SERIF | SERIF** - Sets the type of visual characteristics of the font.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<B STYLE="FONT-FAMILY: SERIF">This sentence is
displayed in a serif font.</B>
```

## FONT-SIZE

Description: Sets the display size of text.

Media Group: Visual

CSS2 Family Type: Font

Values:

- LARGE | MEDIUM | SMALL | X-LARGE | X-SMALL | XX-LARGE | XX-SMALL - Sets the size of the text. Values range from XX-SMALL (smallest) to XX-LARGE (largest).
- LARGER | SMALLER - Changes the size of the current specified element relative to a previously specified value.
- *n* value - Specific unit value for the specified element.
- % value – Sets a percentage for the size of a font relative to the base font size.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<H1 STYLE="FONT-SIZE: 48 pt">Monster-Sized
Heading</H1>
```

## FONT-SIZE-ADJUST

Description: Sets the aspect value for a selected font element. Useful in keeping the “look” of the size of a substituted font the same or close to that of the intended font.

Media Group: Visual

CSS2 Family Type: Font

Values:

- *n* - Sets the aspect value.
- NONE - Browser does not preserve the aspect value between fonts.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="FONT-SIZE: 12pt; FONT-FAMILY: COURIER;  
FONT-SIZE-ADJUST: 1.5;">Font-Size-Adjust-ed  
Header</H1>
```

## FONT-STRETCH

Description: Sets how condensed or expanded a selected font element should be.

Media Group: Visual

CSS2 Family Type: Font

Values:

- ULTRA-CONDENSED | EXTRA-CONDENSED | CONDENSED | SEMI-CONDENSED | NORMAL | SEMI-EXPANDED | EXPANDED | EXTRA-EXPANDED | ULTRA-EXPANDED - Range of fonts from most condensed to most expanded.
- WIDER - Sets the selected font to a more expanded value.
- NARROWER - Sets the selected font to a more condensed value.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="FONT-STRETCH: ULTRA-CONDENSED;">An  
Ultra-Condensed Header</H1>
```

## FONT-STYLE

Description: Sets whether or not the displayed font is italicized.

Media Group: Visual

CSS2 Family Type: Font

Values:

- ITALIC | NORMAL - Determines whether or not text is italicized.
- OBLIQUE – Used to create italicized text (for fonts that do not understand the term “italic”).

- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<I STYLE="FONT: OBLIQUE">This text is set to  
oblique, </I>
```

## FONT-VARIANT

Description: Determines how text is displayed using capital letters.

Media Group: Visual

CSS2 Family Type: Font

Values:

- NORMAL | SMALL-CAPS - These values are toggles to turn the small-caps effect on and off.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<H1 STYLE="FONT-VARIANT: SMALL-CAPS">this text is  
displayed in small capital letters</H1>
```

## FONT-WEIGHT

Description: Sets the thickness of the displayed font.

Media Group: Visual

CSS2 Family Type: Font

Values:

- 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 - Absolute font weight values.
- BOLD | NORMAL - Sets whether or not the specified text element uses a bold face.
- BOLDER | LIGHTER - Changes the weight of the current specified element relative to that of a previously specified value.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<I STYLE="FONT: BOLD">This italicized text is set  
to bold using CSS.</I>
```



## HEIGHT

Description: This property scales an element (typically an image) to the specified height.

Media Group: Visual

CSS2 Family Type: Visual formatting (Details)

Values:

- **AUTO** - Specifies that the browser default value should be used.
- **%** - Sets a percentage value of the element's width.
- ***length (units)*** - Specifies the length value as a unit of measurement.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<IMG SRC="philco_jnr.jpg" STYLE="HEIGHT: 2.5 in">
```

## LEFT

Description: Specifies the value that the current Web element is positioned to the left edge of its containing box.

Media Group: Visual

CSS2 Family Type: Visual formatting

Values:

- ***n* value** - Sets a length unit value.
- **% value** - Sets a length percentage value relative to the width of the containing block.
- **AUTO** - The default value for the specific element.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="POSITION: ABSOLUTE; LEFT: 20PX;">This  
header is positioned 20 pixels from the containing  
box, which in this case is the left margin of the  
browser.</P>
```

## LETTER-SPACING

Description: Specifies the spacing value between letters.

Media Group: Visual

CSS2 Family Type: Text

Values:

- **NORMAL** - The browser's default letter-spacing value.
- ***length (units)*** - Sets the length value between letters as a unit of measurement.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<B STYLE="LETTER-SPACING: 1cm">Some wide letter  
spacing!</B>
```

## LINE-HEIGHT

Description: Sets the distance between lines of text on a Web page.

Media Group: Visual

CSS2 Family Type: Visual formatting (Details)

Values:

- **NORMAL** - Sets the line height to the default value used by the browser.
- ***n*** - Sets a multiple value for the current height of the font in use.
- **%** - Sets a percentage value in relation to the size of the current font in use.
- ***length (units)*** - Specifies the length value as a unit of measurement.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
Baseline<BR>  
<H1 STYLE="LINE-HEIGHT: 2in">This appears 2 inches  
down from the break element at the top of the  
page!</H1>
```

## LIST-STYLE

Description: An abbreviated element that enables Web authors to quickly set list-item markers in the manner possible under LIST-STYLE-POSITION, LIST-STYLE-IMAGE and LIST-STYLE-TYPE.

Media Group: Visual

CSS2 Family Type: Generated content, automatic numbering and lists

Values:

- CIRCLE | DISC | SQUARE - Sets the symbol specified to appear before each list item.
- DECIMAL | LOWER-ALPHA | LOWER-roman | UPPER-ALPHA | UPPER-roman | LOWER-GREEK - Sets the type of alphanumeric symbol that appears before each list item.
- HEBREW | ARMENIAN | GEORGIAN | CJK-IDEOGRAPHIC | HIRAGANA | KATAKANA | HIRAGANA-IROHA | KATAKANA-IROHA - Traditional numbering based upon the language type selected.
- NONE - No list marker is to be displayed before a list item.
- *URL(image)*- Specifies the URL of the graphic to be used as a list marker.
- INSIDE - Displays the text of a list item at a similar indentation level to the list marker.
- OUTSIDE - Displays the text of a list item indented from the list-marker.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<UL STYLE="LIST-STYLE: HEBREW">
  <LI>List items
  <LI>preceded by
  <LI>alphabetical hebrew letters
</UL>
```

## LIST-STYLE-IMAGE

Description: Specifies an image to be used as the marker before a list element.

Media Group: Visual

CSS2 Family Type: Generated content, automatic numbering and lists

Values:

- NONE - No list marker is displayed before a list item.
- *URL(image)*- Specifies the URL of the graphic to be used as a list marker.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<UL>
<LI STYLE="LIST-STYLE-IMAGE: url(trajan.jpg)">Emperor Trajan, 98-117 AD
<LI STYLE="LIST-STYLE-IMAGE:
url(hadrian.jpg)">Emperor Hadrian, 117-158 AD
</UL>
```

## LIST-STYLE-POSITION

Description: Specifies how a list marker is rendered relative to the content of the list item itself.

Media Group: Visual

CSS2 Family Type: Generated content, automatic numbering and lists

Values:

- INSIDE - Displays the text of a list item at a similar indentation level to the list marker.
- OUTSIDE - Displays the text of a list item indented from the list marker.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<UL STYLE="LIST-STYLE-POSITION: INSIDE">
<LI>Notice how the text in the second line of this
bullet item is begins
underneath the bullet rather than directly under
the first word.
</UL>
```

## LIST-STYLE-TYPE

Description: This element sets the type of list markers that appear before list items.

Media Group: Visual

CSS2 Family Type: Generated content, automatic numbering and lists

Values:

- CIRCLE | DISC | SQUARE - Sets the type of symbol to appear before a list element.
- DECIMAL | DECIMAL-LEADING-ZERO | LOWER-ALPHA | LOWER-GREEK | LOWER-LATIN | LOWER-ROMAN | UPPER-ALPHA | UPPER-LATIN | UPPER-ROMAN - Sets the type of alphanumeric symbol that appears before a list marker.
- HEBREW | ARMENIAN | GEORGIAN | CJK-IDEOGRAPHIC | HIRAGANA | KATAKANA | HIRAGANA-IROHA | KATAKANA-IROHA - Traditional numbering based upon the language type selected.
- NONE - No list marker is displayed before a list item.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<UL STYLE="LIST-STYLE-TYPE: LOWER-ALPHA">  
<LI>List items  
<LI>preceded by  
<LI>lower-case letters  
</UL>
```

## MARGIN

Description: An abbreviated element that enables Web authors to quickly set margin values such as MARGIN-TOP, MARGIN-RIGHT, MARGIN-LEFT or MARGIN-BOTTOM.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Borders)

Values:

- AUTO - Specifies that the browser default value should be used.
- % - Sets a percentage value of the element's width.

- `n [ n n n ] measurement_units`
- If 1 value is present, all borders are set to the numerical value specified
- If 2 values are present, the top/bottom and side borders take the numerical values specified
- If 3 values are present, the top, right ***and*** left, then bottom take the numerical values specified
- If 4 values are present, the top, right, bottom then left borders take the numerical values specified
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<P STYLE="MARGIN: 1 in">This text has a margin set  
to one inche.</P>
```

## MARGIN-BOTTOM

Description: Specifies the distance of an element from the bottom edge of the browser window or from the next element that appears below it.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Margins)

Values:

- AUTO - Specifies that the browser default value should be used.
- % - Sets a percentage value of the element's width.
- ***length (units)*** - Specifies the length value as a unit of measurement.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<P STYLE="MARGIN-BOTTOM: 2IN">This text is indented  
2 inches from the element below it.</P>
```

## MARGIN-LEFT

Description: Specifies the distance of an element from the left edge of the browser window or from the element that appears immediately to the left.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Margins)

Values:

- **AUTO** - Specifies that the browser default value should be used.
- **%** - Sets a percentage value of the element's width.
- ***length (units)*** - Specifies the length value as a unit of measurement.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<P STYLE="MARGIN-LEFT: 3IN">This paragraph is  
indented 3 inches from the left margin.</H1>
```

## MARGIN-RIGHT

Description: Specifies the distance of an element from the right edge of the browser window or from the next element that appears to its right.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Margins)

Values:

- **AUTO** - Specifies that the browser default value should be used.
- **%** - Sets a percentage value of the element's width.
- ***length (units)*** - Specifies the length value as a unit of measurement.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<IMG SRC="augustus.jpg" STYLE="MARGIN-RIGHT:  
0.5CM">This text is set half a centimeter from the  
image.
```

## MARGIN-TOP

Description:

Specifies the distance of an element from the top edge of the browser window or from the element that appears above it.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Margins)

Values:

- **AUTO** - Specifies that the browser default value should be used.
- **%** - Sets a percentage value of the element's width.
- ***length (units)*** - Specifies the length value as a unit of measurement.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<H1 STYLE="MARGIN-TOP: 50MM">Header positioned 50
millimeters from the top margin.</H1>
```

## MARKER-OFFSET

Description: Sets the distance between the border of a marker box in relation to the principle box with which it is associated.

Media Group: Visual

CSS2 Family Type: Generated Content, Automatic numbering and lists

Values:

- ***n* value** - Sets a length unit value.
- **AUTO** - The default length value.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<HTML>
<HEAD>
<TITLE>Marker-Offset</TITLE>
<STYLE>
P {MARGIN-LEFT: 10EM} /* Make space for counters */
LI:BEFORE {DISPLAY: MARKER; MARKER-OFFSET: 5EM;
CONTENT: counter(mycounter, UPPER-ALPHA) ".";
COUNTER-INCREMENT: mycounter;}
```



```
</STYLE>
</HEAD>
<BODY>
<P>
<OL>
<LI> Here is the first item.
<LI> Here is the second item.
<LI> Here is the third item.
</OL>
<P>
</BODY>
</HTML>
```

## MARKS

Description: This property sets the type of mark to be displayed, defining the page box. It can be used to add crop marks or cross marks defining the extent of the page. Media Group: Visual, Paged

CSS2 Family Type: Paged media

Values:

- CROP - Adds crop marks to a page, used to determine where the page should be cut.
- CROSS - Adds cross marks to a page, used to align sheets.
- NONE - No mark is made visible.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<STYLE>
@page {SIZE: 8.5IN 11IN; MARKS: CROP;}
</STYLE>
```

## MAX-HEIGHT

Description: Sets the maximum height for a block element.

Media Group: Visual

CSS2 Family Type: Visual formatting (Details)

Values:

- *n* value - Sets a length unit value.
- % value - Sets a length percentage value relative to the height of the containing block.

- **NONE** - Default value, sets no maximum value for the height of the box.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="MAX-HEIGHT: 1CM;">This Header Must Be No  
Bigger than 1CM</H1>
```

## MAX-WIDTH

Description: Sets the maximum width for a block element.

Media Group: Visual

CSS2 Family Type: Visual formatting (Details)

Values:

- **n** value - Sets a length unit value.
- **%** value – Sets a length percentage value relative to the width of the containing block.
- **NONE** - Default value, sets no maximum value for the width of the box.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="MAX-WIDTH: 5CM;">This Header Must Be No  
Wider than 5CM</H1>
```

## MIN-HEIGHT

Description: Sets the minimum height for a block element.

Media Group: Visual

CSS2 Family Type: Visual formatting (Details)

Values:

- **n** value - Sets a length unit value.
- **%** value – Sets a length percentage value relative to the height of the containing block.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="MIN-HEIGHT: 5CM;">This Header Must Be At  
Least 5CM in Height</H1>
```

## MIN-WIDTH

Description: Sets the minimum width for a block element.

Media Group: Visual

CSS2 Family Type: Visual formatting (Details)

Values:

- *n* value - Sets a length unit value.
- % value – Sets a length percentage value relative to the width of the containing block.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="MIN-WIDTH: 2CM;">This Header Must Be No  
Smaller than 2CM in Width</H1>
```

## ORPHANS

Description: Sets the minimum number of lines of a paragraph that must be left at the bottom of a page.

Media Group: Visual, Paged

CSS2 Family Type: Paged media

Values:

- *n* - An integer value that assigns the number of lines that must appear in a paragraph; otherwise, it will be forced to appear on the next page. Default value is 2.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<STYLE>  
@PAGE {ORPHANS: 3;}  
</STYLE>
```

## OUTLINE

Description: A shortcut property that sets a value for the border indicating focus for such things as buttons or form fields.

Media Group: Visual, Interactive

CSS2 Family Type: User interface

Values:

- `n [ n n n ] measurement_units`
- Sets the thickness of the border.
- If 1 value is present, all borders are set to the numerical value specified
- If 2 values are present, the top/bottom and side borders take the numerical values specified
- If 3 values are present, the top, right and left, then bottom take the numerical values specified
- If 4 values are present, the top, right, bottom then left borders take the numerical values specified
- **THIN** | **MEDIUM** | **THICK** - Sets the relative thickness of the border.
- *color name* or *numerical color value* - Sets the color for the border
- **INVERT** - Reverses the color currently being used in the background behind the user interface element.
- **HIDDEN** - Same effect as **NONE**. It is designed to suppress border displays when used in tables.
- **DASHED** - A dashed border line is displayed.
- **DOTTED** - A dotted border line is displayed.
- **DOUBLE** - A double-line is displayed.
- **INSET** - A 3D inset line is displayed.
- **GROOVE** - A 3D grooved line is displayed.
- **NONE** - No border is displayed, no matter what **BORDER-WIDTH** value is present.
- **OUTSET** - A 3D outset line is displayed.
- **RIDGE** - A 3D ridged line is displayed.
- **SOLID** - A solid border line is displayed.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<STYLE>
BUTTONSTYLE {OUTLINE: THICK RED GROOVE;}
</STYLE>
```

## OUTLINE-COLOR

Description: Sets a color value for the border indicating focus for such things as buttons or form fields.

Media Group: Visual, Interactive

CSS2 Family Type: User interface

Values:

- *color name* or *numerical color value* - Sets the color for the border.
- INVERT - Reverses the color currently being used in the background behind the user interface element.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<STYLE>
BUTTONSTYLE {OUTLINE-COLOR: INVERT;}
</STYLE>
```

## OUTLINE-STYLE

Description: Sets a value for the border style indicating focus for such things as buttons or form fields.

Media Group: Visual, Interactive

CSS2 Family Type: User interface

Values:

- DASHED - A dashed border line is displayed.
- DOTTED - A dotted border line is displayed.
- DOUBLE - A double-line is displayed.
- INSET - A 3D inset line is displayed.
- GROOVE - A 3D grooved line is displayed.
- NONE - No border is displayed, no matter what BORDER-WIDTH value is present.
- OUTSET - A 3D outset line is displayed.
- RIDGE - A 3D ridged line is displayed.
- SOLID - A solid border line is displayed.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<STYLE>
BUTTONSTYLE {OUTLINE-STYLE: DOUBLE;}
</STYLE>
```

## OUTLINE-WIDTH

Description: Sets a value for the border width indicating focus for such things as buttons or form fields.

Media Group: Visual, Interactive

CSS2 Family Type: User interface

Values:

- `n [ n n n ] measurement_units`
- Sets the thickness of the border.
- If 1 value is present, all borders are set to the numerical value specified
- If 2 values are present, the top/bottom and side borders take the numerical values specified
- If 3 values are present, the top, right and left, then bottom take the numerical values specified
- If 4 values are present, the top, right, bottom then left borders take the numerical values specified
- `THIN | MEDIUM | THICK` - Sets the relative thickness of the outline.
- `INHERIT` - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<STYLE>
BUTTONSTYLE {OUTLINE-WIDTH: MEDIUM;}
</STYLE>
```

## OVERFLOW

Description: Determines how user can access content that would otherwise be hidden from view when it takes up more space than is allotted to it.

Media Group: Visual

CSS2 Family Type: Visual effects

Values:

- **VISIBLE** - The content is not clipped. It is displayed.
- **HIDDEN** - The content is clipped. No scrollbars are displayed to allow scrolling to view the rest of the content.
- **SCROLL** - The content is clipped, but a scrollbar or equivalent mechanism is made available to allow the viewer to scroll through the content.
- **AUTO** - A scrolling mechanism is automatically provided should it be needed.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<BLOCKQUOTE STYLE="WIDTH: 1.5IN; HEIGHT: 1IN; OVER-
FLOW: AUTO">
If this quote runs too long, a scrollbar will
appear so that the viewer can continue to read the
somewhat lengthy and definitely meandering quota-
tion.
</BLOCKQUOTE>
<P STYLE="WIDTH: 1.5IN; HEIGHT: 1IN; OVERFLOW:
AUTO">
<IMG SRC="vanessa.jpg">
</P>
<P STYLE="WIDTH: 1.5IN; HEIGHT: 1IN; OVERFLOW:
HIDDEN">
<IMG SRC="vanessa.jpg">
</P>
```

## PADDING

Description: An abbreviated element that enables Web authors to quickly set margin values such as **PADDING-TOP**, **PADDING-RIGHT**, **PADDING-LEFT** or **PADDING-BOTTOM**.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Padding)

Values:

- **AUTO** - Specifies that the browser default value should be used.
- **%** - Sets a percentage value of the element's width.
- **n [ n n n ] measurement\_units**
- If 1 value is present, all borders are set to the numerical value specified

- If 2 values are present, the top/bottom and side borders take the numerical values specified
- If 3 values are present, the top, right *and* left, then bottom take the numerical values specified
- If 4 values are present, the top, right, bottom then left borders take the numerical values specified
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<IMG SRC="hadrian.jpg" STYLE="PADDING: 2CM">
```

## PADDING-BOTTOM

Description: Sets the amount of padding to the bottom of an element.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Padding)

Values:

- % - Sets a percentage value of the element's width.
- *length (units)* - Specifies the length value as a unit of measurement.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<P STYLE="PADDING-BOTTOM: 2CM">This paragraph is
padded from the bottom by 2 centimeters.</B>
<P>
Some extra text to demonstrate the effect.
```

## PADDING-LEFT

Description: Sets the amount of padding to the left of an element.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Padding)

Values:

- % - Sets a percentage value of the element's width.
- *length (units)* - Specifies the length value as a unit of measurement.



- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<P STYLE="PADDING-LEFT: 25MM">This is padded from  
the left by 25 millimeters.</P>
```

## ***PADDING-RIGHT***

Description: Sets the amount of padding to the right of an element.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Padding)

Values:

- % - Sets a percentage value of the element's width.
- ***length (units)*** - Specifies the length value as a unit of measurement.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<P STYLE="PADDING-RIGHT: 400PX">This text has a  
PADDING-RIGHT value of 400 pixels. Here is some  
more text to make the effect more apparent.</P>
```

## ***PADDING-TOP***

Description: Sets the amount of padding to the top of an element.

Media Group: Visual

CSS2 Family Type: Boxes (Sub-family: Padding)

Values:

- % - Sets a percentage value of the element's width.
- ***length (units)*** - Specifies the length value as a unit of measurement.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<I STYLE="PADDING-TOP: 5CM">This text is padded  
from the top by 5 centimeters.</I>
```

## PAGE

Description: Used to set a particular page type to be associated with a particular type of Web element.

Media Group: Visual, Paged

CSS2 Family Type: Paged media

Values:

- ***name*** - An identifier that is associated with the element.
- **AUTO** - Follows the automatic formatting for the element. This is the default value.

Sample Code:

```
<HTML>
<HEAD>
<TITLE>Page</TITLE>
<STYLE>
@page regular {SIZE 8.5IN 11IN; MARGIN: 1.5IN;}
@page landscape {SIZE: LANDSCAPE;}
DIV {PAGE: regular;}
TABLE {PAGE: landscape;}
</STYLE>
</HEAD>
<BODY>
<DIV>
The idea behind the PAGE property is to create a
page break <I>and</I> enable the Web author to
format the content of the subsequent page in a
different manner than that which went before it.
</DIV>
<TABLE>
<TR>
<TD>The content in this table will be set in a new
page, and will appear in a landscape format.</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

## ***PAGE-BREAK-AFTER***

Description: Tells the browser to insert a page break after the selected block-level element.

Media Group: Visual, Paged

CSS2 Family Type: Paged media

Values:

- **AUTO** - Neither forces nor prohibits a page break from occurring.
- **ALWAYS** - Tells the browser to always force a page break to occur after the block-level element.
- **AVOID** - Tells the browser to avoid forcing a page break after the block-level element.
- **LEFT** - Forces one or more page breaks to occur after the block-level element so that it appears on a left-hand page.
- **RIGHT** - Forces one or more page breaks to occur after the block-level element so that it appears on a right-hand page.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<HTML>
<HEAD>
<TITLE>Page-Break-After</TITLE>
<STYLE>
P {PAGE-BREAK-AFTER: ALWAYS;}
</STYLE>
</HEAD>
<BODY>
<P>
Each subsequent paragraph will be forced onto a
new page.
</P>
<P>
See, this is on a second page.
</P>
<P>
This is on yet another page.
</P>
</BODY>
</HTML>
```

## ***PAGE-BREAK-BEFORE***

Description: Tells the browser to insert a page break before the selected block-level element.

Media Group: Visual, Paged

CSS2 Family Type: Paged media

Values:

- **AUTO** - Neither forces nor prohibits a page break from occurring.
- **ALWAYS** - Tells the browser to always force a page break to occur before the block-level element.
- **AVOID** - Tells the browser to avoid forcing a page break before the block-level element.
- **LEFT** - Forces one or more page breaks to occur before the block-level element so that it appears on a left-hand page.
- **RIGHT** - Forces one or more page breaks to occur before the block-level element so that it appears on a right-hand page.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<HTML>
<HEAD>
<TITLE>Page-Break-Before</TITLE>
<STYLE>
H1 { PAGE-BREAK-BEFORE: ALWAYS; }
</STYLE>
</HEAD>
<BODY>
Here is some text.
<H1>This header will appear on a new page</H1>
</BODY>
</HTML>
```

## ***PAGE-BREAK-INSIDE***

Description: Tells the browser to insert a page break within the selected block-level element.

Media Group: Visual, Paged

CSS2 Family Type: Paged media

Values:

- **AUTO** - Neither forces nor prohibits a page break from occurring.
- **AVOID** - Tells the browser to avoid forcing a page break within the block-level element.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<HTML>
<HEAD>
<TITLE>Page-Break-Inside</TITLE>
<STYLE>
BLOCKQUOTE {PAGE-BREAK-INSIDE: AVOID;}
</STYLE>
</HEAD>
<BODY>
<BLOCKQUOTE>No matter how long this quote may be,
the PAGE-BREAK-INSIDE property will ensure that a
page break does not occur inside of it, breaking
it across more than one page.
</BLOCKQUOTE>
</BODY>
</HTML>
```

## PAUSE

Description: A shortcut property that sets a value for a break in speech. The value(s) set is applied to the beginning and end of an element.

Media Group: Aural

CSS2 Family Type: Paged media

Values:

- ***n [n]*** - Time value for a break in speech expressed in either milliseconds or seconds.
- **%** - Percentage value expressed in relation to the “speed” of the text being read as set by the **SPEECH-RATE** property.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P>The title of this play is:</P>
<H1 STYLE="PAUSE: 1S;">The Tempest</H1>
<P>by William Shakespeare</P>
```

## ***PAUSE-AFTER***

Description: Sets a value for a break in speech occurring immediately after an element.

Media Group: Aural

CSS2 Family Type: Aural style sheets

Values:

- ***n*** - Time value for a break in speech expressed in either milliseconds or seconds.
- **%** - Percentage value expressed in relation to the “speed” of the text being read as set by the **SPEECH-RATE** property.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="PAUSE-AFTER: 0.5S;">Need a dramatic  
pause?</P> You just got one.
```

## ***PAUSE-BEFORE***

Description: Sets a value for a break in speech occurring immediately before an element.

Media Group: Aural

CSS2 Family Type: Aural style sheets

Values:

- ***n*** - Time value for a break in speech expressed in either milliseconds or seconds.
- **%** - Percentage value expressed in relation to the “speed” of the text being read as set by the **SPEECH-RATE** property.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
Need a dramatic pause?  
<P STYLE="PAUSE-BEFORE: 500MS;">You just got one.  
</P>
```

## PITCH

Description: Sets the average pitch for a given speaking voice. The average pitch for a typical male voice is 120Hz, while for a female voice it is typically about 210Hz.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Voice characteristic properties)

Values:

- **n HZ** - Sets a numeric value for the speaking voice, in Hertz.
- **X-LOW | LOW | MEDIUM | HIGH | X-HIGH** - There are no specific pitch settings for these values, as that depends on the voice family being used. The browser should interpret these values from a very low pitch (X-LOW) to a very high pitch (X-HIGH).
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="VOICE-FAMILY: MALE; PITCH: MEDIUM;">
Hamlet: What hour now?
</P>
<P STYLE=" VOICE-FAMILY: MALE; PITCH: LOW;">
Horatio: I think it lacks of twelve.
</P>
<P STYLE=" VOICE-FAMILY: MALE; PITCH: X-LOW;">
Marcellus: No, it is struck.
</P>
```

## PITCH-RANGE

Description: Specifies variation of the pitch for a particular voice.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Voice characteristic properties)

Values:

- **n** - A numerical value in a range between 0 and 100. A value of 50 is the normal pitch range for a given speaking voice, 0 produces a monotone speaking voice, while 100 produces a highly animated speaking voice.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="PITCH-RANGE: 25;">
Ghost: Revenge his foul and most unnatural murder.
</P>
<P STYLE="PITCH-RANGE: 80;">
Hamlet: Murder?
</P>
<P STYLE="PITCH-RANGE: 35;">
Ghost: Murder most foul, as in the best it is; but
this most foul, strange, and unnatural.
</P>
```

## PLAY-DURING

Description: Sets a sound file to be “behind” a particular Web element as it is read.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Mixing properties)

Values:

- URL(sound) - Points to a sound file in a format supported by the browser.
- MIX - Overlays the specified sound with any other sound values that may be present.
- REPEAT - Keeps repeating the sound for the duration the Web element is present.
- AUTO - Any parent sound present continues to play (instead of simply being restarted) once the selected PLAY-DURING element has finished.
- NONE - No sound; if a parent sound exists, it, too, is turned off.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<BODY STYLE="PLAY-DURING: WINDY-OUTDOORS.WAV
REPEAT;">
Hamlet: Whither wilt thou lead me? Speak! I'll go
no further.
<P STYLE="PLAY-DURING: GHOSTLY.WAV MIX REPEAT;">
Ghost: Mark me.
</P>
Hamlet: I will.
```



```
<P STYLE="PLAY-DURING: GHOSTLY.WAV MIX REPEAT;">  
Ghost: My hour is almost come, when I to  
sulph'rous and tormenting flames must render up  
myself.  
</BODY>
```

## POSITION

Description: Sets the positioning for a box-level element.

Media Group: Visual

CSS2 Family Type: Visual formatting

Values:

- **STATIC** - The box-level element positioning is normal, and is laid out to the normal flow.
- **RELATIVE** - The box-level element positioning is calculated with regard to the normal flow, and this element is offset relative to its normal position.
- **ABSOLUTE** - The box-level element is positioned using the **LEFT**, **RIGHT**, **TOP** and **BOTTOM** properties. Box-level elements positioned this way are separate from the normal flow.
- **FIXED** - The box-level element is positioned in the same manner as with **ABSOLUTE**, but is fixed with regard to some other element.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<STYLE>  
@MEDIA SCREEN { {H1#first {POSITION: FIXED} }  
@MEDIA PRINT { {H1#first {POSITION: STATIC} }  
</STYLE>
```

## QUOTES

Description: Determines the type of quotation mark to be displayed within embedded quotations.

Media Group: Visual

CSS2 Family Type: Generated Content, automatic numbering and lists

Values:

- *string, string* - The open-quote and close-quote values to be used by the CONTENT property.
- NONE - No quotation marks are displayed.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<HTML>
<TITLE>Quotes</TITLE>
<STYLE>
Q:LANG(EN) {QUOTES: ' ' ' ' ' ' ' ' ' '}
Q:BEFORE {CONTENT: OPEN-QUOTE}
Q:AFTER {CONTENT: CLOSE-QUOTE}
</STYLE>
<BODY>
In the lecture the professor said <Q>Shakespeare
was the one who said the immortal words <Q>to
thine own self be true</Q></Q>
</BODY>
</HTML>
```

## RICHNESS

Description: Sets the “brightness” (also known as “richness”) of the speaking voice being used. The brighter the voice, the better it can be heard from a distance.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Voice characteristic properties)

Values:

- *n* - Sets the value of brightness in a given voice. Range is between 0 and 100, where a value of 50 is the normal value for a given voice.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="VOICE-FAMILY: MALE; PITCH: LOW; PITCH-
RANGE: 25; STRESS: 40"; RICHNESS: 80;">
King: O, yet defend me, friends! I am but hurt.
</P>
```

```
<P STYLE="VOICE-FAMILY: MALE; PITCH: MEDIUM;  
PITCH-RANGE: 40; STRESS: 80"; RICHNESS: 70;">  
Hamlet: Here, thou incestuous, murd'rous, damned  
Dane,  
Drink off this potion! Is thy union here?  
Follow my mother.  
</P>  
<P STYLE="VOICE-FAMILY: MALE; PITCH: HIGH; PITCH-  
RANGE: 20; STRESS: 50; RICHNESS: 40;">  
[King dies.]  
</P>
```

## RIGHT

Description: Specifies the value that the current Web element is positioned to the right edge of its containing box.

Media Group: Visual

CSS2 Family Type: Visual formatting

Values:

- *n* value - Sets a length unit value.
- % value - Sets a length percentage value relative to the width of the containing block.
- AUTO - The default value for the specific element.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="POSITION: ABSOLUTE; RIGHT: 2CM;">This  
header is positioned 2 centimeters from the  
containing box, which in this case is the right  
margin of the browser.</P>
```

## SIZE

Description: Sets the size and orientation of the displayed or printed page.

Media Group: Visual, Paged

CSS2 Family Type: Paged media

Values:

- *nn* - Sets the size for the page box in length units.
- AUTO - The page box is set to the size and orientation of the target sheet.

- **PORTRAIT** - Overrides the target's orientation and the shorter sides are horizontal.
- **LANDSCAPE** - Overrides the target's orientation and the longer sides are horizontal.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
@page {SIZE: 8.5in 11in LANDSCAPE; MARGIN: 0.5IN}
```

## ***SPEAK***

Description: Specifies whether text is to be read out loud by the browser.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Speaking properties)

Values:

- **NONE** - Suppresses the text being read aloud.
- **NORMAL** - Reads text aloud using the default pronunciation rules.
- **SPELL-OUT** - Spells out text one letter at a time.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="SPEAK: NORMAL">
<IMG SRC="mom.gif">
Don't tell the baby that she's about to have a <I
STYLE="SPEAK: SPELL-OUT">bath</I>.
<IMG SRC="dad.gif">
</P>
```

## ***SPEAK-HEADER***

Description: Determines how a table header is spoken before individual cell values.

Media Group: Aural

CSS2 Family Type: Tables

Values:

- **ONCE** - The header value is only spoken once, at the beginning of a listing of cell values.
- **ALWAYS** - The header value is spoken before each associated cell value.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<HTML>
<STYLE>
TH {SPEAK-HEADER: ALWAYS}
</STYLE>
<BODY>
<TABLE>
<TR>
<TH>Fruits</TH>
<TH>Vegetables</TH>
</TR>
<TR>
<TD>Apple</TD>
<TD>Carrot</TD>
</TR>
</TABLE>
```

## ***SPEAK-NUMERAL***

Description: Determines how numbers should be read aloud when encountered by the browser.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Speech properties)

Values:

- **DIGITS** - Pronounces numbers as individual digits. The number “123” would be spoken as “one two three”.
- **CONTINUOUS** - Pronounces numbers as a full number. The number “123” would be spoken as “one hundred twenty-three”.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P>
Francisco: You come most carefully upon your hour.
</P>
<P STYLE="SPEAK-NUMERAL: CONTINUOUS;">
Bernardo: 'Tis now struck 12. Get thee to bed,
Francisco.
</P>
```

## ***SPEAK-PUNCTUATION***

Description: Determines how punctuation in text will be read aloud.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Speech properties)

Values:

- **CODE** - Punctuation marks such as periods, commas and brackets are spoken literally.
- **NONE** - Punctuation marks are not spoken, but are instead interpreted as natural pauses in speech.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
In the following sentence, all of the punctuation
(periods, exclamation marks and commas) are read
aloud.
<P STYLE="SPEAK-PUNCTUATION: CODE";>
Queen: No, no! the drink, the drink! O my dear
Hamlet!
The drink, the drink! I am poison'd.
</P>
```

## ***SPEECH-RATE***

Description: Sets the rate at which words on a Web page are spoken.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Voice characteristic properties)

Values:

- ***n*** - Sets the speaking rate in terms of words-per-minute read aloud.
- **X-SLOW** - Equivalent to 80 words-per-minute.
- **SLOW** - Equivalent to 120 words-per-minute.
- **MEDIUM** - Equivalent to 180-200 words-per-minute.
- **FAST** - Equivalent to 300 words-per-minute.
- **X-FAST** - Equivalent to 500 words-per-minute.
- **FASTER** - Increases the current speech rate by 40 words-per-minute.
- **SLOWER** - Decreases the current speech rate by 40 words-per-minute.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<BODY STYLE="SPEECH-RATE: MEDIUM;">
[Ghost beckons Hamlet.]
<P STYLE="SPEECH-RATE: SLOWER;">
Horatio: It beckons you to go away with it, as if
it some impartment did desire to you alone.
</P>
<P STYLE="SPEECH-RATE: FASTER;">
Marcellus: Look with what courteous action It
waves you to a more removed ground. But do not go
with it!
</P>
<P STYLE="SPEECH-RATE: FAST;">
Horatio: No, by no means!
</P>
<P STYLE="SPEECH-RATE: SLOW;">
Hamlet: It will not speak. Then will I follow it.
</P>
</BODY>
```

## STRESS

Description: Sets the amount of stress (i.e., the emphasis) of the speaking voice.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Voice characteristic properties)

Values:

- *n* - Sets the value of stress in a given voice. Range is between 0 and 100, and a value of 50 is the normal stress for a given voice.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="VOICE-FAMILY: MALE; PITCH: LOW; PITCH-
RANGE: 25; STRESS: 40";>
King: How do you, pretty lady?
</P>

<P STYLE="VOICE-FAMILY: FEMALE; PITCH: HIGH;
PITCH-RANGE: 45; STRESS: 95";>
Ophelia: Well, God 'ild you! They say the owl was a
baker's daughter. Lord, we know what we are, but
know not what we may be. God be at your table!
</P>
```

## TABLE-LAYOUT

Description: Controls how a table's width is displayed using either a fixed or automatic value.

Media Group: Visual

CSS2 Family Type: Tables

Values:

- AUTO - Uses automatic table layout.
- FIXED - Uses a fixed table layout.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<TABLE STYLE="TABLE-LAYOUT: AUTO;">
<TR>
<TD>This text is in the table</TD>
</TR>
</TABLE>
```



## TEXT-ALIGN

Description: Sets the alignment of the text.

Media Group: Visual

CSS2 Family Type: Text

Values:

- **CENTER** | **JUSTIFY** | **LEFT** | **RIGHT** - Aligns the text contained by the specified element.
- ***string*** - An identifying string upon which cells in a table will align.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="TEXT-ALIGN: CENTER">Centered text</P>
<P STYLE="TEXT-ALIGN: JUSTIFY">Justified text</P>
<P STYLE="TEXT-ALIGN: LEFT">Left-aligned text</P>
<P STYLE="TEXT-ALIGN: RIGHT">Right-aligned text</P>
```

## TEXT-DECORATION

Description: Adds a “decorative” property to the HTML tag specified.

Media Group: Visual

CSS2 Family Type: Text

Values:

- **BLINK** - Text blinks.
- **LINE-THROUGH** - Text contains a line through the middle (strikethrough text).
- **NONE** - No decoration is added.
- **OVERLINE** - Text is displayed with a line running above it.
- **UNDERLINE** - Text is underlined.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample code:

```
<A HREF="fred.html" STYLE="TEXT-DECORATION:
NONE">This link is not underlined</A>
<H1 STYLE="TEXT-DECORATION: OVERLINE">Header with
an Overline</H1>
<H1 STYLE="TEXT-DECORATION: UNDERLINE">Header
with an Underline</H1>
```

```
<P STYLE="TEXT-DECORATION: LINE-THROUGH">
Ignore this sentence
<P>
<I STYLE="TEXT-DECORATION: BLINK">This text
blinks</I>
```

## TEXT-INDENT

Description: Specifies indents from the left margin or from a block-element.

Media Group: Visual

CSS2 Family Type: Text

- **length (units)** - Specifies the length value as a unit of measurement.
- **%** - Specifies the distance as a percentage of the browser window display.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="TEXT-INDENT: 2em">This text is
indented from the margins by 1 inch.</H1>
```

## TEXT-SHADOW

Description: Adds a shadow effect to text.

Media Group: Visual

CSS2 Family Type: Text

Values:

- **NONE** - No shadow value is set.
- **nn [n]**, - Sets a color value plus the unit length of the shadow. At least two length unit values must be present. A comma can separate a chain of overlaying values.
- **color name** or **numerical color value** - Sets the color value for the shadow. Always associated with a length unit value.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="TEXT-SHADOW: GRAY 2PX 3PX, LIME -4PX 3PX 4PX;">Header with Drop Shadow Effect</H1>
```

## TEXT-TRANSFORM

Description: Specifies the type of case to be used on the text.

Media Group: Visual

CSS2 Family Type: Text

Values:

- CAPITALIZE - Sets the initial letter of a word in capital letters.
- LOWERCASE - Sets all text to lower case.
- NONE - No transformation is performed.
- UPPERCASE - Sets all text to upper case.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="TEXT-TRANSFORM: CAPITALIZE">Header in initial caps</H1>  
<H2 STYLE="TEXT-TRANSFORM: LOWERCASE">HEADER ALL IN LOWERCASE LETTERS</H2>  
<H3 STYLE="TEXT-TRANSFORM: UPPERCASE">header all in uppercase letters</H3>  
<H4 STYLE="TEXT-TRANSFORM: NONE">Nothing special happens with this header</H4>
```

## TOP

Description: Specifies the value that the current Web element is positioned below the containing box.

Media Group: Visual

CSS2 Family Type: Visual formatting

Values:

- *n* value - Sets a length unit value.
- % value - Sets a length percentage value relative to the width of the containing block.
- AUTO - The default value for the specific element.

- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="POSITION: ABSOLUTE; TOP: 1CM;">This
header is positioned 1 centimeter below the
containing box, which in this case is the bottom of
the browser window.</P>
```

## UNICODE-BIDI

Description: Defines the Unicode text as having bi-directional characteristics, meaning that the text may be rendered from right to left and from left to right in different circumstances.

Media Group: Visual

CSS2 Family Type: Visual formatting

Values:

- **NORMAL** - The selected Web element does not alter the current bi-directional setting.
- **EMBED** - If the selected Web element is inline, this value sets up an additional level of embedding with respect to the bi-directional algorithm.
- **BIDI-OVERRIDE** - If the selected Web element is inline or block-level, this value overrides that in place, keeping with the values set by the **DIRECTION** property.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<STYLE>
ARABIC {DIRECTION: RTL; UNICODE-BIDI: EMBED}
ENGLISH {DIRECTION: LTR; UNICODE-BIDI: EMBED}
</STYLE>
```

## VERTICAL-ALIGN

Description: Sets the vertical positioning of the HTML tag with which it is associated.

Media Group: Visual

## CSS2 Family Type: Visual formatting (Details)

Values:

- **BASELINE | BOTTOM | MIDDLE | SUB | SUPER | TEXT-TOP | TEXT-BOTTOM | TOP** - Aligns the text element in relation to the specified value.
- ***n* value** - Sets a length unit value.
- **% value** - Sets a length percentage value relative to the line-height of the containing block.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<B STYLE="VERTICAL-ALIGN: SUB">Subscript-aligned
text</B> <I STYLE="FONT-SIZE: 16pt">Large, itla-
cized text.</I>
<B STYLE="VERTICAL-ALIGN: SUPER">Superscript-
aligned text</B> <I STYLE="FONT-SIZE: 16pt">Large,
itlacized text.</I>
```

## VISIBILITY

Description: Sets whether or not the box associated with an element is rendered.

Media Group: Visual

CSS2 Family Type:

Values:

- **VISIBLE** - The box generated by the element is displayed.
- **HIDDEN** - The box generated by the element is not displayed.
- **COLLAPSE** - When applied to a table element (i.e., a row, row group, column or column group) causes the entire selected row or column to not be displayed.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<H1 STYLE="VISIBILITY: HIDDEN;">Header in an
Invisible Box</H1>
```

## VOICE-FAMILY

Description: Sets a priority list of voice-family names.

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Voice characteristic properties)

Values:

- **MALE | FEMALE | CHILD** - A “generic” voice value that sounds male, female and child-like, respectively. Can modify the value of a specific-voice.
- ***specific-voice name*** - A specific, predefined name for a voice value.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="VOICE-FAMILY: LAERTES, MALE;">
Laertes: Farewell, Ophelia, and remember well what
I have said to you.
</P>
<P STYLE="VOICE-FAMILY: OPHELIA, FEMALE;">
Ophelia: 'Tis in my memory lock'd, and you your-
self shall keep the key of it.
</P>
<P STYLE="VOICE-FAMILY: LAERTES, MALE;">
Laertes: Farewell.
</P>
<P STYLE="VOICE-FAMILY: MALE;">
Exit Laertes.
</P>
```

## VOLUME

Description: Sets the level for the dynamic range of the sound being played (i.e., its “volume”).

Media Group: Aural

CSS2 Family Type: Aural (Sub-family: Volume properties)

Values:

- ***n*** - (range is between 0 (near silence) to 100 (full volume))
- **% value** - (range is between 0 (near silence) to 100 (full volume))
- **SILENT** - Volume is turned off

- **X-SOFT** - Very low volume level, equivalent to the numerical value “0” (which does not imply silence)
- **SOFT** - Low volume level, equivalent to the numerical value “25”
- **MEDIUM** - Medium volume level, equivalent to the numerical value “50”
- **LOUD** - Moderately loud volume level, equivalent to the numerical value “75”
- **X-LOUD** - Very loud volume level, equivalent to the numerical value “100”
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="VOLUME: MEDIUM;">
I'm the big gruff bear.
<IMG SRC="bear.gif">
<B STYLE="VOLUME: LOUD;">Roar!</B>
</P>
```

## WHITE-SPACE

Description: This element is used to set white space and the way carriage returns are handled within a Web page.

Media Group: Visual

CSS2 Family Type: Text

Values:

- **NORMAL** - The default browser behavior for the HTML tag with which this element is associated.
- **NOWRAP** - When this value is set, carriage returns and linefeeds are rendered as single space. Line breaks are not effected, however.
- **PRE** - When this value is set, all spaces, carriage returns, and linefeeds in the document are displayed “as is”.
- **INHERIT** - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<P STYLE="WHITE-SPACE: NORMAL">
There   are no extra white spaces in
      this sentence.  It is displayed normally.
</P>
```

```
<P STYLE="WHITE-SPACE: PRE">
The  extra white  spaces and carriage
returns in  this sentence
      are displayed.
</P>

<P STYLE="WHITE-SPACE: NOWRAP">
The  extra white  spaces and carriage
returns in  this sentence
      are <B>not</B> displayed.
</P>
```

## WIDOWS

Description: Sets the minimum number of lines of a paragraph that must be left at the top of a page.

Media Group: Visual, Paged

CSS2 Family Type: Paged media

Values:

- *n* - An integer value that assigns the number of lines that must appear in a paragraph; otherwise, it will be forced to appear on the previous page. Default value is 2.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<STYLE>
@page {WIDOWS: 3;}
</STYLE>
```

## WIDTH

Description: Scales an element (most typically an image) to the width specified.

Media Group: Visual

CSS2 Family Type: Visual formatting (Details)

Values:

- AUTO - Specifies that the browser default value should be used.



- `%` - Sets a percentage value with respect to the browser window's width.
- *length (units)* - Specifies the length value as a unit of measurement.
- `INHERIT` - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<IMG STYLE="WIDTH: AUTO" SRC="gordian3.jpg"> Regular size (AUTO setting)
<P>
<IMG STYLE="WIDTH: 25px" SRC="gordian3.jpg"> Set to 25 pixels
<P>
<IMG STYLE="WIDTH: 25%" SRC="gordian3.jpg"> Set to 25% of the browser window's width
```

## WORD-SPACING

Description: Sets the spacing between words.

Media Group: Visual

CSS2 Family Type: Text

Values:

- `NORMAL` - The browser default word spacing value is used.
- *length (units)* - Sets the length value between words as a unit of measurement.
- `INHERIT` - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<I STYLE="WORD-SPACING: 1cm">The words in this sentence are all spaced 1cm apart.</I>
```

## Z-INDEX

Description: Sets the stacking level of the specified box in relation to the current stacking context. It also determines whether the specified box sets up a localized stacking context.

Media Group: Visual

CSS2 Family Type: Visual formatting

Values:

- *n* - An integer value representing the stacking level of the specified element's box. The specified element's box also sets up a local stacking value.
- AUTO - The stack value is the same as its parent's. The specified element's box does not set up a local stacking value.
- INHERIT - Takes the same value as that set for the property of the parent element, if allowed.

Sample Code:

```
<HTML>
<HEAD>
<TITLE>Z-Index Example</TITLE>
<STYLE>
.stack {POSITION: ABSOLUTE; RIGHT: 1CM; BOTTOM:
0CM; }
</STYLE>
</HEAD>
<BODY STYLE="FONT-SIZE: XX-LARGE; COLOR: YEL-
LOW;">
<P>
<IMG SRC="fokker.gif" CLASS="stack" STYLE="Z-
INDEX: 1" width=369 height=201 border=0 alt="">
<DIV ID="text1" CLASS="stack" STYLE="Z-INDEX: 3">
This text overlays image1.
</DIV>
<DIV ID="text2">This text has not been assigned a
Z-INDEX value, so it lies beneath everything else.
</DIV>
<DIV ID="text3" CLASS="stack" STYLE="Z-INDEX: 2">
This text will underlay text1, but overlay image1.
</DIV>
</BODY>
</HTML>
```

# Pseudo-Classes and Pseudo-Elements

## ***ACTIVE***

Determines how a hypertext link should appear when clicked (or otherwise selected) by the user.

## ***AFTER***

Inserts a string and/or otherwise modifies the display at the end of the selected Web element.

## ***BEFORE***

Inserts a string and/or otherwise modifies the display at the beginning of the selected Web element.

## ***FIRST***

Used to specify the style of the first page of a Web page (used in conjunction with the @page rule).

## ***FIRST-CHILD***

Sets the first child Web element *only* to have the same display characteristics as that of the parent Web element.

## ***FIRST-LETTER***

Sets how the first letter in the first line of text in a paragraph should be displayed.

## ***FIRST-LINE***

Sets how the first line of text in a paragraph should be displayed.

## ***FOCUS***

This pseudo-element is set to take effect whenever the Web element with which it is associated has “focus” (i.e., is available to take user input in some form).

## ***HOVER***

This pseudo-element is set to take effect whenever the cursor lies immediately above the Web element with which it is associated.

## ***LANG***

Sets the language associated with a particular Web element.

## ***LEFT***

Used to specify the style of all left-handed Web pages (used in conjunction with the @page rule).

## ***LINK***

Determines how a “resting” (i.e., unclicked) hypertext link appears on a Web page.

## ***RIGHT***

Used to specify the style of all right-handed Web pages (used in conjunction with the @page rule).

## ***VISITED***

Determines how a hypertext link should appear after it has been visited (i.e., clicked).

# Media Types

## ***ALL***

Web content is suitable for all devices.

Includes all media types.

## ***AURAL***

Web content is intended for a “talking browser” (equipped with a speech synthesizer).

Associated Media Groups: Continuous, Aural, Interactive/Static.

## ***BRAILLE***

Web content is intended for use on a Braille tactile interactive feedback device.

Associated Media Groups: Continuous, Tactile, Grid, Interactive/Static.

## ***EMBOSSSED***

Web content is designed for printing in Braille.

Associated Media Groups: Paged, Tactile, Grid, Interactive/Static.

## ***HANDHELD***

Web content is meant to be displayed on a handheld device (typically with a small, monochrome screen and limited bandwidth).

Associated Media Groups: Continuous/Paged, Visual, Grid/Bitmap, Interactive/Static.

## ***PRINT***

Web content is specifically designed to be printed, or it can be viewed on screen in a print-preview form.

Associated Media Groups: Paged, Visual, Bitmap, Static.

## ***PROJECTION***

Web content is meant for projection, either directly or from printed transparencies.

Associated Media Groups: Paged, Visual, Bitmap, Static.

## ***SCREEN***

Web content is designed for standard color computer displays.

Associated Media Groups: Continuous, Visual, Grid, Interactive/Static.

## ***TTY***

Web content is designed for use on such things as teletype displays or terminals that use a fixed-pitch character grid.

Associated Media Groups: Continuous, Visual, Grid, Interactive/Static.

## ***TV***

Web content is meant for display on a television screen (typically relatively low resolution, color display, sound, and with limited scrolling capabilities).

Associated Media Groups: Continuous/Paged, Visual/Aural, Bitmap, Interactive/Static.



# **COLORS, UNITS OF MEASURE, PERCENTAGE UNITS AND URLS**





# *Appendix C*

## Colors

You can specify colors in CSS in any of five ways:

1. Using a predefined color name (e.g., “BLUE”)
2. Using a full hexadecimal color value preceded by a hash mark (e.g., “#00FF00”)
3. Using a compressed hexadecimal color value preceded by a hash mark, where each number is duplicated (e.g., “#0F0”)
4. Using a three-digit RGB (“Red Green Blue”) color value preceded by “RGB”, with each digit specifying the amount of color to be used on a scale from 0 to 255, and the digits contained in round brackets (e.g., RGB(0,255,0))
5. Using a three-value RGB percentage color value preceded by “RGB”, with each digit specifying the percentage of color to be used on a scale from 0% to 100%, and the percentage values contained within round brackets (e.g., RGB(0%,100%,0%)).

Sample code:

## Listing C-1 Color Formats

```

<HTML>
<HEAD>
<TITLE>Color Formats</TITLE>
</HEAD>

<BODY STYLE="FONT-SIZE: LARGE">
<P STYLE="COLOR: LIME;">
This text should be lime green <CODE>(STYLE="COLOR: LIME;"</CODE>.
</P>
<P STYLE="COLOR: #00FF00;">
This text should be lime green <CODE>(STYLE="COLOR: #00FF00;" )
</CODE>.
</P>
<P STYLE="COLOR: #0F0;">
This text should be lime green <CODE>(STYLE="COLOR: #0F0;"</CODE>.
</EM>
<P STYLE="COLOR: RGB(0,255,0);">
This text should be lime green <CODE>(STYLE="COLOR: RGB(0,255,0);" )
</CODE>.
</P>
<P STYLE="COLOR: RGB(0%,100%,0%;">
This text should be lime green <CODE>(STYLE="COLOR: RGB(0%,100%,0%;" )
</CODE>.
</P>

</BODY>
</HTML>

```

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Partial	Safe	Partial	Safe	Safe	Safe	Safe	Safe

## VGA Palette Color Names

According to the CSS specification, a browser should be able to recognize the 16 core color names that correspond to the Windows VGA palette of colors.

The following is a listing of those color names and their hexadecimal equivalents:

**Table C-1 The 16 Color Names that Comprise the VGA Palette**

<i>Color Name</i>	<i>Hexadecimal Value</i>
Aqua	00FFFF
Black	000000
Blue	0000FF
Fuchsia	FF00FF
Gray	808080
Green	008000
Lime	00FF00
Maroon	800000
Navy	000080
Olive	808000
Purple	800080
Red	FF0000
Silver	C0C0C0
Teal	008080
White	FFFFFF
Yellow	FFFF00

## The “Safe” 216-Color Palette

While many monitors are capable of rendering displays in up to 16.7 million colors, there are still many computer displays capable of only displaying 256 colors at a time. Of those colors, only 216 colors can be displayed without being dithered to a different color value. The following is a listing of “safe” color values that will be displayed uniformly and without dithering on systems capable of displaying at least 256 colors.

The “safe” values are those hexadecimal values containing any combination of the following hexadecimal values:

- 00
- 33
- 66
- 99
- CC
- FF

The following list is a reliable palette to choose from when using color values with CSS.

**Table C-2 Values for the “Safe” Color Palette**

### *“Safe” Hexadecimal Color Values*

	<i>0000</i>	<i>3300</i>	<i>6600</i>	<i>9900</i>	<i>CC00</i>	<i>FF00</i>
<b>00</b>	000000	003300	006600	009900	00CC00	00FF00
	000033	003333	006633	009933	00CC33	00FF33
	000066	003366	006666	009966	00CC66	00FF66
	000099	003399	006699	009999	00CC99	00FF99
	0000CC	0033CC	0066CC	0099CC	00CCCC	00FFCC
	0000FF	0033FF	0066FF	0099FF	00CCFF	00FFFF
<b>33</b>	330000	333300	336600	339900	33CC00	33FF00
	330033	333333	336633	339933	33CC33	33FF33
	330066	333366	336666	339966	33CC66	33FF66
	330099	333399	336699	339999	33CC99	33FF99

Table C-2 Values for the "Safe" Color Palette

	3300CC	3333CC	3366CC	3399CC	33CCCC	33FFCC
	3300FF	3333FF	3366FF	3399FF	33CCFF	33FFFF
<b>66</b>	660000	663300	666600	669900	66CC00	66FF00
	660033	663333	666633	669933	66CC33	66FF33
	660066	663366	666666	669966	66CC66	66FF66
	660099	663399	666699	669999	66CC99	66FF99
	6600CC	6633CC	6666CC	6699CC	66CCCC	66FFCC
	6600FF	6633FF	6666FF	6699FF	66CCFF	66FFFF
<b>99</b>	990000	993300	996600	999900	99CC00	99FF00
	990033	993333	996633	999933	99CC33	99FF33
	990066	993366	996666	999966	99CC66	99FF66
	990099	993399	996699	999999	99CC99	99FF99
	9900CC	9933CC	9966CC	9999CC	99CCCC	99FFCC
	9900FF	9933FF	9966FF	9999FF	99CCFF	99FFFF
<b>CC</b>	CC0000	CC3300	CC6600	CC9900	CCCC00	CCFF00
	CC0033	CC3333	CC6633	CC9933	CCCC33	CCFF33
	CC0066	CC3366	CC6666	CC9966	CCCC66	CCFF66
	CC0099	CC3399	CC6699	CC9999	CCCC99	CCFF99
	CC00CC	CC33CC	CC66CC	CC99CC	CCCCCC	CCFFCC
	CC00FF	CC33FF	CC66FF	CC99FF	CCCCFF	CCFFFF
<b>FF</b>	FF0000	FF3300	FF6600	FF9900	FFCC00	FFFF00
	FF0033	FF3333	FF6633	FF9933	FFCC33	FFFF33
	FF0066	FF3366	FF6666	FF9966	FFCC66	FFFF66
	FF0099	FF3399	FF6699	FF9999	FFCC99	FFFF99
	FF00CC	FF33CC	FF66CC	FF99CC	FFCCCC	FFFFCC
	FF00FF	FF33FF	FF66FF	FF99FF	FFCCFF	FFFFFF

## *A Spectrum of Colors and Their Hexadecimal Equivalents*

The following lists common colors with their hexadecimal equivalents. The list begins with the variations of white, then blacks and grays, and then a full spectrum ranging from reds to violets.

**Table C-3 Hexadecimal Values for Common Color Names**

<i>Color Name</i>	<i>Hexadecimal Value</i>	<i>Color Name</i>	<i>Hexadecimal Value</i>
Snow	FFFAFA	Lavender Blush	FFF0F5
Ghost White	F8F8FF	Misty Rose	FFE4E1
White Smoke	F5F5F5	White	FFFFFF
Gainsboro	0EDCDC	Black	000000
Floral White	FFFAF0	Dark Slate Gray	2F4F4F
Old Lace	FDF5E6	Dim Gray	696969
Linen	FAF0E6	Slate Gray	708090
Antique White	FAEBD7	Light Slate Gray	778899
Papaya Whip	FFEFD5	Gray	BEBEBE
Blanched Almond	FFEBCD	Light Gray	D3D3D3
Bisque	FFE4C4	Midnight Blue	191970
Peach Puff	FFDAB9	Navy Blue	000080
Navajo White	FFDEAD	Cornflower Blue	6495ED
Moccasin	FFE4B5	Dark Slate Blue	483D8B
Cornsilk	37F8DC	Slate Blue	6A5ACD
Ivory	FFFFF0	Medium Slate Blue	7B68EE
Lemon Chiffon	FFFACD	Light Slate Blue	8470FF
Seashell	FFF5EE	Medium Blue	0000CD
Honeydew	F0FFF0	Royal Blue	4169E1
Mint Cream	F5FFFA	Blue	0000FF
Azure	F0FFFF	Dodger Blue	1E90FF
Alice Blue	F0F8FF	Deep Sky Blue	0013FF
Lavender	E6E6FA	Sky Blue	87CEEB

**Table C-3 Hexadecimal Values for Common Color Names (continued)**

<i><b>Color Name</b></i>	<i><b>Hexadecimal Value</b></i>	<i><b>Color Name</b></i>	<i><b>Hexadecimal Value</b></i>
Light Sky Blue	87CEFA	Yellow Green	9ACD32
Steel Blue	4682B4	Forest Green	228B22
Light Steel Blue	B0C4DE	Olive Drab	6B8E23
Light Blue	ADD8E6	Dark Khaki	BDB76B
Powder Blue	B0E0E6	Khaki	F0E68C
Pale Turquoise	AFEEEE	Pale Goldenrod	EEE8AA
Dark Turquoise	00CED1	Light Goldenrod Yellow	FAFAD2
Medium Turquoise	48D1CC	Light Yellow	FFFFE0
Turquoise	40E0D0	Yellow	FFFF00
Cyan	00FFFF	Gold	FFD700
Light Cyan	E0FFFF	Light Goldenrod	EEDD82
Cadet Blue	5F9EA0	Goldenrod	DAA520
Medium Aquamarine	66CDAA	Dark Goldenrod	B8860B
Aquamarine	7FFFD4	Rosy Brown	BC8F8F
Dark Green	006400	Indian Red	CD5C5C
Dark Olive Green	556B2F	Saddle Brown	8B4513
Dark Sea Green	8FBC8F	Sienna	A0522D
Sea Green	2E8B57	Peru	CD853F
Medium Sea Green	3CB371	Burlywood	DEB887
Light Sea Green	20B2AA	Beige	F5F5DC
Pale Green	98FB98	Wheat	F5DEB3
Spring Green	00FF7F	Sandy Brown	F4A460
Lawn Green	7CFC00	Tan	D2B48C
Green	00FF00	Chocolate	D2691E
Chartreuse	7FFF00	Firebrick	B22222
Medium Spring Green	00FA9A	Brown	A52A2A
Green Yellow	ADFF2F	Dark Salmon	E9967A
Lime Green	32CD32	Salmon	FA8072

**Table C-3 Hexadecimal Values for Common Color Names (continued)**

<i><b>Color Name</b></i>	<i><b>Hexadecimal Value</b></i>	<i><b>Color Name</b></i>	<i><b>Hexadecimal Value</b></i>
Light Salmon	FFA07A	Medium Violet Red	C71585
Orange	FFA500	Violet Red	D02090
Dark Orange	FF8C00	Magenta	FF00FF
Coral	FF7F50	Violet	EE82EE
Light Coral	F08080	Plum	DDA0DD
Tomato	FF6347	Orchid	DA70D6
Orange Red	FF4500	Medium Orchid	BA55D3
Red	FF0000	Dark Orchid	15332CC
Hot Pink	FF69B4	Dark Violet	9400D3
Deep Pink	FF0E93	Blue Violet	8A2BE2
Pink	FFC0CB	Purple	A020F0
Light Pink	FFB6C1	Medium Purple	9370DB
Pale Violet Red	DB7093	Thistle	D8BFD8
Maroon	B03060		

## Units of Measure

There are eight units of measure that can be used in combination with CSS properties:

- inches – IN
- millimeters – MM
- centimeters – CM
- picas – PC
- points – PT
- pixels – PX
- m-lengths – EM
- x-heights – EX



Sample code:

### Listing C-2 Length Units

```
<HTML>
<HEAD>
<TITLE>Length Units</TITLE>
<STYLE TYPE="text/css">
.NOLEFTMARGIN {MARGIN-LEFT: 0;}
.INCH {MARGIN-LEFT: 1IN;}
.MM {MARGIN-LEFT: 25.4MM;}
.CM {MARGIN-LEFT: 2.54CM;}
.PICA {MARGIN-LEFT: 6PC;}
.POINT {MARGIN-LEFT: 72PT;}
.EX {MARGIN-LEFT: 12EX;}
.EM {MARGIN-LEFT: 6EM;}
.PIXEL {MARGIN-LEFT: 96PX;}
</STYLE>
</HEAD>

<BODY STYLE="font-family : monospace;">
<P CLASS="NOLEFTMARGIN">
This paragraph has no left margin. All of the other
paragraphs that follow have margins that should be
set the equivalent of one inch.
</P>
<P CLASS="INCH">
This paragraph has a left margin set to 1 inch.
</P>
<P CLASS="MM">
This paragraph has a left margin set to 25.4 millimeters.
</P>
<P CLASS="CM">
This paragraph has a left margin set to 2.54 millimeters.
</P>
<P CLASS="PICA">
This paragraph has a left margin set to 6 picas.
</P>
<P CLASS="POINT">
This paragraph has a left margin set to 72 points.
</P>
<P CLASS="EX">
This paragraph has a left margin set to 12 x-heights.
</P>
```

### Listing C-2 Length Units (continued)

```
<P CLASS="EM">
This paragraph has a left margin set to 6 em-spaces. The
major browsers calculate 1 ex = 1/2 em, which
is not technically correct, but at least it is consistent.
;-)
</P>
<P CLASS="PIXEL">
This paragraph has a left margin set to 96 pixels (which is
equivalent to 1 inch on the monitor
displaying this code).
</P>

</BODY>
</HTML>
```

---

Support in Major Browsers:

		<i>IE 4.01</i>	<i>IE 4.5</i>	<i>IE 5.0</i>		<i>NN 4.0</i>	
<i>IE 3.02</i>	<i>IE 4.x</i>	<i>(Mac)</i>	<i>(Mac)</i>	<i>(Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>(Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Partial	Partial	Partial	Partial	Partial	Partial	Partial	Partial

---

## Percentage Units

Using a number followed by a percentage sign (“%”) always specifies percentage values in CSS. Percentage values are always relative to another value, such as the width or the height of the browser window.

Sample code:

### Listing C-3 Percentages

```
<HTML>
<HEAD>
<TITLE>Percentages</TITLE>
</HEAD>
<BODY STYLE="FONT-SIZE: X-LARGE">
<P STYLE="MARGIN-LEFT: 25%;">
This text is indented 25% of the browser window to the
left. Here's some extra text to make this more
apparent.
</P>

<P STYLE="MARGIN-RIGHT: 25%;">
This text is indented 25% of the browser window to the
right. Here's some extra text to make this more
apparent.
</P>

</BODY>
</HTML>
```

## Support in Major Browsers:

[illegible]

## URLs

When a URL is used in a CSS rule, using the word “URL” and the link contained in round brackets specifies it.

Source code:

### Listing C-4 URLs

```
<HTML>
<HEAD>
<TITLE>URLs</TITLE>
<STYLE>
BODY {BACKGROUND: URL(http://www.klgroup.com/images/background.gif);
MARGIN-RIGHT: 100PX; MARGIN-LEFT: 100PX;}
</STYLE>
</HEAD>
<BODY STYLE="FONT-SIZE: X-LARGE";>
This Web page uses a background image imported from an outside
source, and has margins set to 100 pixels on both sides so that the
text does not stray over the dark yellow "sides" set by the back-
ground image.

</BODY>
</HTML>
```

---

Support in Major Browsers:

<i>IE 3.02</i>	<i>IE 4.x</i>	<i>IE 4.01 (Mac)</i>	<i>IE 4.5 (Mac)</i>	<i>IE 5.0 (Win. &amp; UNIX)</i>	<i>NN 4.x</i>	<i>NN 4.0 (Mac &amp; UNIX)</i>	<i>Opera 3.6</i>
Safe	Safe	Safe	Safe	Safe	Partial	Partial	Safe

---





# *Index*



## ***Symbols***

!IMPORTANT property 66

## ***A***

A:ACTIVE 86

A:LINK 86

A:VISITED 86

ANCHOR pseudo-element 86

## ***C***

CLASS HTML attribute 13

combining CSS rules 44

comments in styles 57

CSS comments 57

CSS grouping 44

## ***D***

<DIV> 13

## ***F***

FIRST-LETTER pseudo-element 92

FIRST-LINE pseudo-element 89

FONT-FAMILY property 100

## ***G***

grouping 44

## ***H***

HTML attributes for CSS 13

HTML tags for CSS 13

**I**

ID HTML attribute 13

**S**

<SPAN> 13

<STYLE> 13

STYLE HTML attribute 13





