

Tommy Olsson & Paul O'Brien



The Ultimate CSS Reference (Chapter 8)

Thank you for downloading this sample chapter from *The Ultimate CSS Reference*, by Tommy Olsson and Paul O'Brian.

This excerpt encapsulates the Summary of Contents, Information about the Author and SitePoint, Table of Contents, and Chapter 8: *Layout Properties*. We hope you find this information useful in evaluating the book.

For more information, visit sitepoint.com

Summary of Contents of this Excerpt		
8. Layout Properties	263	
Summary of Additional Book Cont	ents	
1. What is CSS	1	
2. General Syntax and Nomenclature	23	
3. At-rules Reference	47	
4. Selector Reference	59	
5. The Cascade, Specificity, and Inheritance	117	
6. CSS Layout and Formatting	139	
7. Box Properties	187	
9. List Properties	285	
10. Table Properties	291	
11. Color and Backgrounds		
12. Typographical Properties		
13. Generated Content		
14. User Interface Properties	357	
15. Paged Media Properties	361	
16. Vendor Specific Properties		
17. Workarounds, Filters, and Hacks		
18. Differences Between HTML and XHTML		
Alphabetical Property Index		



THE ULTIMATE CSS REFERENCE

BY TOMMY OLSSON & PAUL O'BRIEN

The Ultimate CSS Reference

by Tommy Olsson and Paul O'Brien

Copyright © 2008 SitePoint Pty Ltd

Managing Editor: Simon Mackie Technical Director: Kevin Yank

Technical Editor: Andrew Tetlaw

Expert Reviewer: Natalie Downe

Cover Design: Simon Celen

Expert Reviewer: Roger Johansson Interior Design: Xavier Mathieu

Printing History:

First Edition: February 2008

Notice of Rights

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations included in critical articles or reviews.

Notice of Liability

The author and publisher have made every effort to ensure the accuracy of the information herein. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors and SitePoint Pty Ltd, nor its dealers or distributors will be held liable for any damages to be caused either directly or indirectly by the instructions contained in this book, or by the software or hardware products described herein.

Trademark Notice

Rather than indicating every occurrence of a trademarked name as such, this book uses the names only in an editorial fashion and to the benefit of the trademark owner with no intention of infringement of the trademark.



Published by SitePoint Pty Ltd

48 Cambridge Street Collingwood VIC Australia 3066 Web: www.sitepoint.com Email: business@sitepoint.com

ISBN 978–0–9802858–5–7 Printed and bound in the United States of America

About the Authors

Hailing from Hampshire in the UK, **Paul O'Brien** is a freelance web designer specializing in CSS layouts. After selling a successful packaging business back in 1998 he was all set for a quiet existence, dabbling with his hobby of web design. However, what started out as a hobby soon became a full-time occupation as the demand for well-coded CSS layouts started growing. Even when he's not working, he can be found giving out helpful advice in the SitePoint Forums where he has racked up nearly 20,000 posts, all of which are CSS-related.

Paul's other passion is karate, which he has studied continuously for 35 years. He currently holds the rank of Third Dan (Sandan) in Shotokan karate, so I wouldn't argue with him if I were you!

Tommy Olsson is a pragmatic evangelist for web standards and accessibility, who lives in the outback of central Sweden. Visit his blog at http://www.autisticcuckoo.net/.

About the Expert Reviewers

The always excitable **Natalie Downe** works for Clearleft, in Brighton, as a client-side web developer. An experienced usability consultant and project manager, her first loves remain front-end development and usability engineering. She enjoys Doing Things Right and occasionally dabbling in the dark art of Python and poking the odd API.

Roger Johansson is a web professional with a passion for web standards, accessibility, and usability. He spends his days developing web sites at Swedish web consultancy NetRelations, and his evenings and weekends writing articles for his personal sites, http://www.456bereastreet.com/ and http://www.kaffesnobben.com/.

About the Technical Editor

Andrew Tetlaw has been tinkering with web sites as a web developer since 1997 and has also worked as a high school English teacher, an English teacher in Japan, a window cleaner, a car washer, a kitchen hand, and a furniture salesman. At SitePoint he is dedicated to making the world a better place through the technical

editing of SitePoint books and kits. He is also a busy father of five, enjoys coffee, and often neglects his blog at http://tetlaw.id.au/.

About the Technical Director

As Technical Director for SitePoint, Kevin Yank oversees all of its technical publications—books, articles, newsletters, and blogs. He has written over 50 articles for SitePoint, but is best known for his book, *Build Your Own Database Driven Website Using PHP & MySQL*. Kevin lives in Melbourne, Australia, and enjoys performing improvised comedy theater and flying light aircraft.

About SitePoint

SitePoint specializes in publishing fun, practical, and easy-to-understand content for web professionals. Visit http://www.sitepoint.com/ to access our books, newsletters, articles, and community forums.

The Online Reference

The online version of this reference is located at http://reference.sitepoint.com/css. The online version contains everything in this book, fully hyperlinked and searchable. The site also allows you to add your own notes to the content and to view those added by others. You can use these user-contributed notes to help us to keep the reference up to date, to clarify ambiguities, or to correct any errors.

Your Feedback

If you wish to contact us, for whatever reason, please feel free to email us at books@sitepoint.com. We have a well-manned email support system set up to track your inquiries. Suggestions for improvement are especially welcome.

Reprint Permissions

Do you want to license parts of this book for photocopying, email distribution, Intranet or Extranet posting or for inclusing in a coursepack? Please go to Copyright.com and type in this book's name or ISBN number to purchase a reproduction license.

Table of Contents

Chapter 1	What Is CSS?	1
CSS Versio	ons	4
Linking CS	S to a Web Document	5
Media	a Queries	14
Standards	Mode, Quirks Mode, and Doctype Sniffing	17
Summary .		20
Chapter 2	General Syntax and	
	Nomenclature	23
Statement	s	25
At-rules		25
Rule Sets.		26
Selectors.		26
Declaration	n Blocks	28
Declaration	ns, Properties, and Values	28
Keyw	ords	29
Lengt	hs and Units	29
Perce	ntages	32
Colors	s	33
Numb	pers	37
String	gs	37
URIs .		38
Initia	l Values	39
Short	hand Properties	39
CSS Comm	nents	42
CSS Identi	fiers	43
CSS Escap	e Notation	43

	CSS Syntax Errors	44
	Summary	45
Ch	apter 3 At-rules Reference	47
	@charset	
	@import	
	@media	
	@page	
	@font-face	
	@namespace	
Ch	apter 4 Selector Reference	59
	Universal Selector	60
	Element Type Selector	62
	Class Selector	63
	ID Selector	65
	Attribute Selector	67
	CSS3 Attribute Selectors	71
	Selector Grouping	72
	Combinators	73
	Descendant Selector	74
	Child Selector	76
	Adjacent Sibling Selector	77
	General Sibling Selector	79
	Pseudo-classes	80
	:link	83
	:visited	84
	:active	85
	:hover	86

:foc	us	87
:firs	t-child	88
:lan	g(C)	89
CSS	3 Pseudo-classes	90
Pseudo-e	elements	106
:firs	t-letter	107
:firs	t-line	110
:bef	ore	113
:afte	er	114
::sel	ection	115
Chapter 5	The Cascade, Specificity, and	l
	Inheritance	117
The Cases	ade	118
!importa	nt Declarations	124
	ty	
Inheritan	ce	133
The	CSS Property Value inherit	135
Summary	/	137
Chapter 6	CSS Layout and Formatting.	120
_	•	
	port, the Page Box, and the Canvas	
	Box Model	
	taining Block	
	apsing Margins	
	Internet Explorer 5 Box Model	
	Internet Explorer hasLayout Property	
	ng Concepts	
Bloc	ck Formatting	164

Inline Formatting	166
List Formatting	168
Table Formatting	168
Replaced Elements	175
Positioning	176
Relative Positioning	176
Absolute Positioning	178
Fixed Positioning	178
Stacking Contexts	179
Floating and Clearing	180
The Relationship Between display, position, and float.	184
Summary	185
Chapter 7 Box Properties	187
Dimensions	187
height	188
min–height	190
max-height	192
width	194
min-width	196
max-width	198
Margins	200
margin-top	200
margin-right	202
margin-bottom	205
margin-left	207
margin	209
Padding	211
padding-top	212
padding-right	213

	padding-bottom	215
	padding-left	216
	padding	218
Bord	lers and Outlines	220
	border-top-color	220
	border-top-style	222
	border-top-width	224
	border-top	226
	border-right-color	228
	border-right-style	229
	border-right-width	232
	border-right	233
	border-bottom-color	235
	border-bottom-style	236
	border-bottom-width	239
	border-bottom	240
	border-left-color	242
	border-left-style	243
	border-left-width	246
	border-left	247
	border-color	249
	border-style	251
	border-width	254
	border	255
	outline-color	258
	outline-style	259
	outline-width	260
	outline	261

Chapter 8	Layout Properties	 263
display		 264
position .		 267
float		 269
clear		 271
visibility		 273
top		 275
right		 276
bottom .		 277
left		 278
z-index .		 279
overflow		 280
clip		 283
Chapter 9	List Properties	 285
list-style-t	type	 286
list-style-	position	 288
list-style-i	image	 289
list-style		 290
Chapter 10	Table Properties	 291
table-layo	out	 292
border-co	llapse	 293
border-spa	acing	 294
empty-cel	lls	 295
caption-si	de	 297

Chapter 11 Color and Backgrounds	299
background-color	299
background-image	301
background-repeat	303
background-position	305
background-attachment	309
background	312
color	315
Chapter 12 Typographical Properties	317
font-family	318
font-size	320
font-weight	321
font-style	323
font-variant	324
font	325
letter-spacing	326
word-spacing	327
line-height	328
text-align	330
text-decoration	332
text-indent	334
text-transform	335
text-shadow	337
vertical-align	338
white-space	341
direction	343
unicode-bidi	344

Chapter 13	Generated Content	347
content		348
counter-inc	erement	352
counter-res	et	354
quotes		355
Chapter 14	User Interface Properties	357
cursor		358
Chapter 15	Paged Media Properties	361
page-break	-before	362
page-break	-inside	363
page-break	-after	364
orphans		365
widows		366
Chapter 16	Vendor-specific Properties	367
Mozilla Ext	ensions	371
-moz-	border-radius	372
-moz-	box-sizing	375
The di	splay Property Value: -moz-inline-box	377
Internet Ex	plorer Extensions	379
zoom	·	380
filter		381
behavi	or	387
The ex	pression Property Value	388
Summary.		390

Chapter 17	Workarounds, Filters, and
	Hacks
Internet Ex	plorer Conditional Comments
Workaroun	ds and Filters400
CSS Hacks	404
Summary .	
Chapter 18	Differences Between HTML and XHTML
NAINAE T	
	s
Case Sensit	ivity
Optional Ta	ags
Root Eleme	ent Properties
Appendix A	Alphabetic Property Index 417

Layout Properties

Layout properties allow authors to control the visibility, position, and behavior of the generated boxes for document elements. CSS layout is a complex topic and further information can be found in CSS Layout and Formatting (p. 139).



264

display

```
display: { block | inline | inline-block |
inline-table | list-item | run-in | table |
table-caption | table-cell | table-column |
table-column-group | table-footer-group |
table-header-group | table-row | table-row-group |
none | inherit };
```

This property controls the type of box an element generates.

The computed value may differ from the specified value for the root element and for floated or absolutely positioned elements; see The Relationship Between display, position, and float (p. 184) for details about the relationship between the display, float (p. 269), and position (p. 267) properties.

Example

The following rule will cause a elements that are descendants of the .menu element to render as block elements instead of inline elements:

```
.menu a {
   display: block;
}
```

block makes the element generate a block box.

Note that a **user agent style sheet** may override the initial value of inline for some elements.

Value

block

inline	inline makes the element generate one or more inline boxes.
inline-block	inline-block makes the element generate a block box that's laid out as if it were an inline box.

 $\textbf{inline-table} \qquad \qquad \textbf{inline-table} \ \ \, \textbf{makes} \ the \ element \ behave \ like \ a \ table \ that \emph{'s}$

laid out as if it were an inline box.

list-item makes the element generate a principal block

box and a list-item inline box for the list marker.

run-in A value of run-in makes the element generate either a

block box or an inline box, depending on the context. If the run-in box doesn't contain a block box, and is followed by a sibling block box (except a table caption) in the normal flow that isn't, and doesn't contain, a run-in box, the run-in box becomes the first inline box of the sibling block box. Otherwise, the run-in box becomes a block

box.

table table makes the element behave like a table.

table-caption table-caption makes the element behave like a table

caption.

table-cell makes the element behave like a table cell.

table-column makes the element behave like a table

column.

table-column-group table-column-group makes the element behave like a

table column group.

table-footer-group makes the element behave like a

table footer row group.

table-header-group makes the element behave like a

table header row group.

table-row table-row makes the element behave like a table row.

table-row-group makes the element behave like a table

body row group.

none

A value of none makes the element generate no box at all. Descendant boxes cannot generate boxes either, even if their display property is set to something other than none.

Compatibility

Internet Explorer			Firefox				Opera		
5.5	6.0	7.0	1.0	1.5	2.0	1.3	2.0	3.0	9.2
Partial	Partial	Partial	Partial	Partial	Partial	Full	Full	Full	Full

Internet Explorer versions up to and including 7:

- don't support the values inline-table, run-in, table, table-caption, table-cell, table-column, table-column-group, table-row, and table-row-group
- only support the values table-footer-group and table-header-group for thead and tfoot elements in HTML
- only support the value inline-block for elements that are naturally inline or have been set to inline outside the declaration block
- treat block as list-item on li elements in HTML
- will apply a layout (p. 158) to inline-block elements
- don't support the value inherit

Firefox versions up to and including 2.0, and Opera 9.2 and prior versions:

- only support the value table-column-group for colgroup elements in HTML
- only support the value table-column for col elements in HTML

Firefox versions up to and including 2.0 don't support the values inline-block, inline-table, or run-in.



position

```
position: { absolute \mid fixed \mid relative \mid static \mid inherit \} ;
```

		SP	EC		
inherite	d	ini	tial	١	version
NO		sta	tic	CSS2	
BRO		OWSER	SUPPO	ORT	
IE5.5+	F	F1+	Saf1.3	3+	Op9.2+
BUGGY	F	ULL	FULI		FULL

The position property, together with the float property, controls the way in which the position of the element's generated box is computed. See Positioning (p. 176) for details about element positioning.

Boxes with a position value other than static are said to be positioned. Their vertical placement in the stacking context is determined by the z-index (p. 279) property.

Example

This style rule makes the element with ID "sidebar" absolutely positioned at the top right-hand corner of its containing block:

```
#sidebar {
  position: absolute;
  top: 0;
  right: 0;
}
```

Value

absolute

The value absolute generates an absolutely positioned box that's positioned relative to its containing block. The position can be specified using one or more of the properties top (p. 275), right (p. 276), bottom (p. 277), and left (p. 278). Absolutely positioned boxes are removed from the flow and have no effect on later siblings. Margins on absolutely positioned boxes never collapse with margins on other boxes.

fixed

The value fixed generates an absolutely positioned box that's positioned relative to the initial containing block (normally the viewport). The position can be specified using one or more of the properties top (p. 275), right (p. 276), bottom (p. 277), and left (p. 278). In the print media type, the element is rendered on every page.

relative

The value relative generates a positioned box whose position is first computed as for the normal flow. The generated box is then offset from this position according to the properties top (p. 275) or bottom (p. 277)

and/or left (p. 278) or right (p. 276). The position of the following box is computed as if the relatively positioned box occupied the position that was computed before the box was offset. This value cannot be used for table cells, columns, column groups, rows, row groups, or captions.

The value static generates a box that isn't positioned, but occurs in the normal flow. The properties top (p. 275), right (p. 276), bottom (p. 277), left (p. 278), and z-index (p. 279) are ignored for static boxes.

Compatibility

Internet Explorer			Firefox			Safari			Opera
5.5	6.0	7.0	1.0	1.5	2.0	1.3	2.0	3.0	9.2
Buggy	Buggy	Buggy	Full	Full	Full	Full	Full	Full	Full

Internet Explorer for Windows versions up to and including 6 don't support the value fixed.

Internet Explorer for Windows versions up to and including 6 have problems with margin calculations for absolutely positioned boxes. Percentages for dimensions are computed relative to the parent block, rather than the containing block. Consider this example:

```
<div id="containing">
    <div id="parent">
        <div id="child"></div>
        </div>
    </div>
```

```
#containing {
  position: relative;
  width: 200px;
  height:200px;
}
#parent {
  width: 100px;
  height: 100px;
}
#child {
  position: absolute;
```

```
top: 10px;
left: 10px;
width: 50%;
height: 50%;
}
```

Here, the element with ID "child" is absolutely positioned, and therefore its containing block is the one generated by the element with the (convenient) ID "containing"—the "child" element's nearest positioned ancestor. IE6 and earlier versions will make the "child" element 50 pixels square—50% of the element with the ID "parent"—instead of the expected 100 pixels, since they base the calculation on the dimensions of the *parent* block rather than the *containing* block.

Internet Explorer versions up to and including 7:

- always generate a new stacking context (p. 179) for positioned boxes, even if z-index is auto
- don't support the value inherit

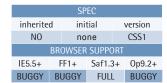
In Internet Explorer for Windows versions up to and including 7, a position value of absolute will cause an element to gain a layout (p. 158), as will a value of fixed in version 7.



float

```
float: { left | right | none | inherit } ;
```

This property specifies whether or not a box should float and, if so, if it should float to the left or to the right. A floating box is shifted to the left or to the right as far as it can go, and non-floating content in the normal flow will flow around it on the opposite side. The float property is ignored for elements



Example

This style rule makes the box generated by the element with ID "nav" float to the left:

```
#nav {
  float: left;
}
```

that are absolutely positioned. User agents are also allowed to ignore it when it's applied to the root element.

See Floating and Clearing (p. 180) for more information about the behavior of floated elements.

Value

270

left makes the element generate a block box that is floated to the left

right makes the element generate a block box that is floated to the right

none makes the element generate a box that is not floated

Compatibility

	Internet Explorer			Firefox				Opera		
	5.5	6.0	7.0	1.0	1.5	2.0	1.3	2.0	3.0	9.2
В	uggy	Buggy	Buggy	Buggy	Buggy	Buggy	Full	Full	Full	Buggy

Internet Explorer versions up to and including 6 add three pixels of padding (in the floated direction) to adjacent line boxes.

In Internet Explorer versions up to and including 6, the left or right margins are doubled on floated elements that touch their parents' side edges. The margin value is doubled on the side that touches the parent. A simple fix for this problem is to set display to inline for the floated element.

Internet Explorer for Windows versions up to and including 7:

- will place a floated box below an immediately preceding line box
- will expand a left-floated box to the width of the containing block if it has a right-floated child and a computed width of auto
- will apply a layout (p. 158) to a floated element
- don't support the value inherit

In Firefox versions up to and including 2.0, a floated box appears below an immediately preceding line box. A left-floated box with a right-floated child and a computed width of auto expands to the width of the containing block.

In Opera up to and including version 9.2, if the computed width of the floated box is auto and it has floated children, its width is computed as if the floats don't wrap and aren't cleared.

Other Relevant Stuff

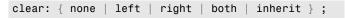


clear (p. 271)

prevents a box from being adjacent to floated boxes



clear



		SP	EC		
inherite	d	ini	tial	,	version
NO	NO		none		CSS1
	BR		SUPPO	RT	
IE5.5+	F	F1+	Saf1.3	3+	Op9.2+
BUGGY	FULL		FULL		FULL

This property specifies which sides of an element's box (or boxes) can't be adjacent to any floated boxes. This property can clear an element only from floated boxes within the same block formatting context (p. 164). It doesn't clear the element from floated child boxes within the element itself.

The clearance is achieved by adding space above the top margin of the

Example

This style rule prevents all pre elements in an HTML document from being adjacent to a previously floated box:

```
pre {
   clear: both;
}
```

element, if necessary, until the top of the element's border edge is below the bottom edge of any boxes floated in the specified direction or directions.

When clear is specified for a run-in box, it applies to the block box to which the run-in box eventually belongs.

See Floating and Clearing (p. 180) for more information about the behavior of cleared elements.

Value

1eft The value left adds space above the element's generated box, if necessary, to put its top border edge below the bottom edge of any left-floating boxes previously generated by elements in the same block formatting context.

right The value right adds space above the element's generated box, if necessary, to put its top border edge below the bottom edge of any right-floating boxes previously generated by elements in the same block formatting context.

The value both adds space above the element's generated box, if necessary, to put its top border edge below the bottom edge of any floating boxes that were previously generated by elements in the same block formatting context.

none The value **none** doesn't clear any previously floated boxes.

Compatibility

Int	ternet Explo	rer	Firefox			Safari			Opera
5.5	6.0	7.0	1.0	1.5	2.0	1.3	2.0	3.0	9.2
Buggy	Buggy	Buggy	Full	Full	Full	Full	Full	Full	Full

Internet Explorer for Windows versions up to and including 6 exhibit a bug known as the peekaboo bug, wherein a cleared element that touches the floating box(es) it clears may become invisible.

Internet Explorer for Windows version 7:

- doesn't clear elements with an unshared ancestor whose height value is anything other than auto
- doesn't clear floated elements if the clear property is specified for an element floating in the opposite direction

Other Relevant Stuff



float (p. 269)

 $specifies\ whether\ a\ box\ should\ float\ to\ the\ left\ or\ right,\ or\ not\ float\ at\ all$



visibility

```
visibility: { visible | hidden | collapse |
inherit };
```

 SPEC

 inherited
 initial
 version

 YES
 visible
 CSS2, 2.1

 BROWSER SUPPORT

 IE5.5+
 FF1+
 Saf1.3+
 Op9.2+

 PARTIAL
 FULL
 PARTIAL
 BUGGY

This property specifies whether an element is visible—that is, whether the box(es) that are generated by an element are rendered.

Note that even if a box in the normal flow is hidden, it still affects the layout of other elements, unlike the behavior that occurs when we suppress box generation altogether by setting display

Example

This style rule makes the element with ID "dynamic" generate an invisible box. It can be made visible using client-side scripting:

```
#dynamic {
  visibility: hidden;
}
```

to none. Descendant boxes of a hidden box will be visible if their visibility is set to visible, whereas descendants of an element for which display is set to none can never generate boxes of their own.

The initial value and the inheritability were changed in CSS2.1 to address the previously undefined state for the root element.

Value

visible The value visible makes the generated boxes visible.

hidden The value hidden makes the generated boxes invisible without removing them from the layout. Descendant boxes can be made visible.

collapse The value **collapse** is only meaningful for certain internal table objects: rows, row groups, columns, and column groups. It causes the object to

be removed from the display; the space it occupied will be filled by subsequent siblings. This doesn't affect the table layout in any other way, so the user agent doesn't have to recompute the layout of the table. If a spanned row or column intersects the collapsed object, it is clipped. When it's specified for any other element than these internal table objects, collapse causes the same behavior as hidden.

Compatibility

Internet Explorer			Firefox			Safari			Opera
5.5	6.0	7.0	1.0	1.5	2.0	1.3	2.0	3.0	9.2
Partial	Partial	Partial	Full	Full	Full	Partial	Partial	Partial	Buggy

Internet Explorer for Windows versions up to and including 7:

- don't support the value collapse
- don't support the value inherit
- don't allow descendant boxes of an element whose visibility value is hidden to be made visible if the ancestor has a layout (p. 158)

Opera 9.2 and prior versions treat the value collapsed as hidden for all elements.

Safari versions up to and including 2.0 don't support the value collapse.



top

```
top: { length | percentage | auto | inherit } ;
```

SPEC								
inherite	d	initial			version			
NO	NO		auto		CSS2			
	BROWSE			ORT				
IE7+	F	F1+	Saf1.3+		Op9.2+			
FULL	F	ULL	FULL		FULL			

For absolutely positioned boxes, this property specifies how far the top margin edge of the box is offset below the top padding edge of its containing block. However, should the value for top be auto (the initial value), the top margin edge of the box will be positioned at the top content edge of its containing block.

Example

This style rule makes the element with ID "logo" generate a relatively positioned box that's shifted down by ten pixels:

```
#logo {
  position: relative;
  top: 10px;
}
```

For relatively positioned boxes, this property specifies how far the top edge of the box is offset below the position it would have had in the normal flow.

Compatibility

Internet Explorer			Firefox				Opera		
5.5	6.0	7.0	1.0	1.5	2.0	1.3	2.0	3.0	9.2
Buggy	Buggy	Full	Full	Full	Full	Full	Full	Full	Full

Internet Explorer for Windows versions up to and including 6:

- compute percentage values on the basis of the height of the parent block, rather than of the containing block
- don't support the specification of both the position and the dimensions of an absolutely positioned element using top, right, bottom, and left together; they'll use the last vertical and horizontal position specified, and need the dimensions to be specified using width and height



right

```
right: { length | percentage | auto | inherit } ;
```

For absolutely positioned boxes, this property specifies how far the right margin edge of the box is offset from the left of the right padding edge of its containing block.

For relatively positioned boxes, this property specifies how far the right edge of the box is offset from the left of the position it would have had in the normal flow.

Example

This style rule makes the element with ID "sidebar" generate an absolutely positioned box at the top right-hand corner of its containing block:

```
#sidebar {
  position: absolute;
  top: 0;
  right: 0;
}
```

If both right and left have a value other than auto, the offset is over-constrained. If the direction property is ltr, right will be ignored. If direction is rtl, left will be ignored.

Compatibility

Int	Internet Explorer			Firefox			Safari		
5.5	6.0	7.0	1.0	1.5	2.0	1.3	2.0	3.0	9.2
Buggy	Buggy	Full	Full	Full	Full	Full	Full	Full	Full

Internet Explorer for Windows versions up to and including 6:

- compute percentage values on the basis of the width of the parent block, rather than that of the containing block
- don't support the specification of both the position and the dimensions of an absolutely positioned element using top, right, bottom, and left together; they'll use the last vertical and horizontal position specified, and need the dimensions to be specified using width and height



bottom

bottom: { length percentage auto	inherit	}	;
--------------------------------------	---------	---	---

		SP	EC		
inherite	d	ini	tial		version
NO	NO		auto		CSS2
	BROW			ORT	
IE7+	F	F1+	Saf1.3	3+	Op9.2+
FULL	FULL		FULI	L	FULL

For absolutely positioned boxes, this property specifies how far the bottom margin edge of the box is offset above the bottom padding edge of its containing block.

For relatively positioned boxes, this property specifies how far the bottom edge of the box is offset above the position it would have had in the normal flow.

Example

This style rule makes the element with ID "logo" generate a relatively positioned box that's shifted ten pixels upward:

```
#logo {
  position: relative;
  bottom: 10px;
}
```

If both top and bottom have a value other than auto, bottom is ignored.

Compatibility

Int	ernet Explo	rer		Firefox			Opera		
5.5	6.0	7.0	1.0	1.5	2.0	1.3	2.0	3.0	9.2
Buggy	Buggy	Full	Full	Full	Full	Full	Full	Full	Full

Internet Explorer for Windows versions up to and including 6:

- compute percentage values on the basis of the height of the parent block, rather than of the containing block
- are one pixel off when bottom and right are used to specify the position, and the offset is an odd number of pixels
- don't support the specification of both the position and the dimensions of an absolutely positioned element using top, right, bottom, and left together; they'll use the last vertical and horizontal position specified, and need the dimensions to be specified using width and height



left

left: { length | percentage | auto | inherit } ;

		SP	EC			
inherited init		initial		version		
NO au		auto		CSS2		
BROWSER SUPPORT						
IE7+	F	F1+	Saf1.3+		Op9.2+	
FULL	- 1	FULL	FULL		FULL	

For absolutely positioned boxes, this property specifies how far the left margin edge of the box is offset to the right of the left padding edge of its containing block. However, should the value for left be auto (the initial value), the left margin edge of the box is positioned at the left content edge of its containing block.

For relatively positioned boxes, this property specifies how far the left edge of the box is offset to the right of the

Example

This style rule makes the element with ID "nav" generate an absolutely positioned box at the top left-hand corner of its containing block:

```
#nav {
  position: absolute;
  top: 0;
  left: 0;
}
```

position it would have had in the normal flow. If both right and left have a value other than auto, the offset is over-constrained. If the direction property is ltr, right will be ignored. If direction is rtl, left will be ignored.

Compatibility

Int	ernet Explo	rer	Firefox				Opera		
5.5	6.0	7.0	1.0	1.5	2.0	1.3	2.0	3.0	9.2
Buggy	Buggy	Full	Full	Full	Full	Full	Full	Full	Full

Internet Explorer for Windows versions up to and including 6:

- compute percentage values on the basis of the width of the parent block, rather than that of the containing block
- don't support the specification of both the position and the dimensions of an absolutely positioned element using top, right, bottom, and left together; they'll use the last vertical and horizontal position specified, and need the dimensions to be specified using width and height

Internet Explorer for Windows versions up to and including 7 don't support the value inherit.

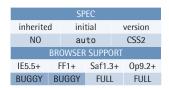


z-index

```
z-index: { integer | auto | inherit } ;
```

This property specifies the stack level of a box whose position value is one of absolute, fixed, or relative.

The **stack level** refers to the position of the box along the z axis, which runs perpendicular to the display. The higher the value, the closer the box is to the user; in other words, a box with a high z-index will obscure a box with a lower z-index occupying the same location along the x and y axes.



Example

This style rule makes the element with ID "warning" absolutely positioned and assigns it a higher stack level than its siblings:

```
#warning {
  position: absolute;
  z-index: 1;
}
```

See Stacking Contexts (p. 179) for more information about stacking contexts.

Value

An integer value—which can be negative—sets the stack level of the box in the current stacking context, and also establishes a new stacking context. The box itself has stack level 0 (zero) in the new context.

The value auto gives the box the same stack level as its parent, and doesn't establish a new stacking context.

Compatibility

	Inte	ernet Explo	rer	Firefox				Opera		
	5.5	6.0	7.0	1.0	1.5	2.0	1.3	2.0	3.0	9.2
В	Buggy	Buggy	Buggy	Buggy	Buggy	Buggy	Full	Full	Full	Full

In Internet Explorer for Windows versions up to and including 6, select elements always appear on top of everything else; their stack level can't be changed.

Internet Explorer for Windows versions up to and including 7 always use the nearest positioned ancestor to determine the stacking context for the element in question.

Internet Explorer for Windows version 7 treats the value auto as if it were 0 (zero).

Internet Explorer for Windows versions up to and including 7 don't support the value inherit.

In Firefox versions up to and including 2, a negative stack level positions the box behind the stacking context, rather than above the context's background and borders and below block-level descendants in the normal flow.

Other Relevant Stuff



position (p. 267)

specifies the positioning scheme used to position an element



overflow

```
overflow: { auto | hidden | scroll | visible |
inherit };
```

This property specifies the behavior that occurs when an element's content overflows the element's box.

The default behavior is to make the overflowing content visible, but it can be changed so that the content is clipped to the confines of the element's box, optionally providing a mechanism for scrolling the content.

Example

This style rule makes the pre element type in HTML generate a fixed-sized box with visible scrollbars:

inherited

FULL

initial

visible

FULL

Saf1.3+

FULL

version

0p9.2+

FULL

```
pre {
  width: 40em;
  height: 20em;
  overflow: scroll;
}
```

If the overflow property is applied to the body or html elements in an HTML document, the user agent may apply it to the viewport. This does not apply to XHTML, though.

If a scrollbar needs to be provided, the user agent should insert it between the element's outer padding edge and its inner border edge. The space occupied by the scrollbar should be subtracted (by the user agent) from the computed width or height, so that the inner border edge is preserved.

Boxes with an overflow value other than visible will expand vertically to enclose any floated descendant boxes.

Margins will never collapse for a box with an overflow value other than visible.

Value

auto

The behavior of auto isn't specified in any detail in the CSS2.1 specification. In existing implementations it provides scrollbar(s) when necessary, but it doesn't show scrollbars unless the content overflows the element's box.

hidden

hidden causes content that overflows the element's box to be clipped. No scrolling mechanism will be provided, so the overflow will be invisible and inaccessible.

scroll

scroll clips overflowing content, just like hidden, but provides a scrolling mechanism so that the overflow can be accessed. This scrolling mechanism is present whether the content overflows the element's box or not, to prevent it from appearing and disappearing in a dynamic layout. When the output medium is print, this value allows overflowing content to be printed (as if the value were visible).

visible

visible allows overflowing content to be visible. It will be rendered *outside* the element's box, and may thus overlap other content.

Compatibility

In	ternet Explo	rer	Firefox				Opera		
5.5	6.0	7.0	1.0	1.5	2.0	1.3	2.0	3.0	9.2
Buggy	Buggy	Full	Full	Full	Full	Full	Full	Full	Full

Internet Explorer for Windows versions up to and including 6:

- will not apply a value specified for the body element to the viewport, if the computed value for the html element is visible
- will increase the width and height of the element when the value is specified as visible, instead of rendering the overflow outside the element's box; if the value is auto, hidden, or scroll, and the element's width is specified as auto, the width will increase to avoid overflow

In Internet Explorer for Windows versions up to and including 7:

- a relatively positioned child of an element whose overflow value is auto or scroll will behave as if the position were specified as fixed; if overflow is hidden, a relatively positioned element will be visible if the generated box lies outside the parent's box
- the value inherit is unsupported

In Internet Explorer for Windows version 7, the values auto, hidden, and scroll cause an element to gain a layout (p. 158).

Firefox versions up to and including 2 apply ${\tt overflow}$ to table row groups.



clip

```
clip: { shape | auto | inherit } ;
```

This property sets the clipping region for an absolutely positioned element.

Any part of an element that would render outside the clipping region will be invisible. This includes the content of the element and its children, backgrounds, borders, outlines, and even any visible scrolling mechanism.

Clipping may be further influenced by any clipping regions that are set for the element's ancestors, and whether or not those have a visibility property whose value is something other than

inherited initial version NO auto CSS2, 2.1 BROWSER SUPPORT IE5.5+ FF1+ Saf1.3+ Op9.2+ PARTIAL FULL FULL FULL

Example

This style rule assigns a clipping region of 200×100 pixels for the element with ID "tunnel-vision". The upper left-hand corner of the clipping region is at position (50,50) with respect to the element's box:

visible. Clipping may also occur at the edges of the browser window, or the margins of the paper (when printing).

The default clipping region is a rectangle with the same dimensions as the element's border box.

Value

If the value is specified as auto, no clipping will be applied.

The only shape value that's allowed in CSS2.1 is a rectangle, which must be specified using the rect() functional notation. The function takes four comma-separated arguments—top, right, bottom, and left—in the usual TRouBLe order. Each argument is either auto or a length, and negative length values are allowed. The top and bottom positions are relative to the top border edge of the element's box. The left and right positions are relative to the left border edge in a left-to-right environment, or to the right border edge in a right-to-left environment. When specified as auto, the position is that of the corresponding border edge.

Note that the interpretation of positions specified in the rect() functional notation changed between CSS2 and CSS2.1. In CSS2, each value specified the offset from the corresponding border edge.

Compatibility

	Int	ernet Explo	rer	Firefox				Opera		
Ī	5.5	6.0	7.0	1.0	1.5	2.0	1.3	2.0	3.0	9.2
	Partial	Partial	Partial	Full	Full	Full	Full	Full	Full	Full

Internet Explorer for Windows versions up to and including 7 do not support the recommended syntax for the rect() notation. However, they do support a deprecated syntax where the arguments are separated by whitespace rather than commas.



What's Next?

If like what you've seen in this chapter from *The Ultimate CSS Reference*, why not order yourself a copy?

This book is perfect anyone wanting an in-depth, accurate, and beautifully presented CSS reference at their fingertips. You'll not find a more up-to-date CSS reference that includes browser compatibility information, working examples and easy-to-read descriptions. Covering the entire CSS language *The Ultimate CSS Reference* contains all the CSS knowledge you'll ever need.

Purchase Options

Depending on your preference, you've got the convenience, durability, and usability of a hard cover printed version, a transportable off-line version (PDF), as well as our freely accessible <u>online version</u>. Choose one or choose them all.

Order a copy today.



Appendix

Alphabetic Property Index

This is a complete, alphabetical fist of the C55 properties contained in this reference.
background
background-attachment
background-color
background-image
background-position305
background-repeat303
behavior
border
border-bottom
border-bottom-color
border-bottom-style236
border-bottom-width239
border-color 249

418 The Ultimate CSS Reference

border-collapse
border-left
border-left-color
border-left-style
border-left-width
border-right
border-right-color
border-right-style229
border-right-width232
border-spacing
border-style
border-top
border-top-color
border-top-style
border-top-width
border-width
bottom
caption-side
clear 271
clip
color
content
counter-increment
counter-reset
cursor
direction 343
display
empty-cells
The expression Property Value
filter

float	269
font	325
font-family	318
font-size	320
font-style	323
font-variant	324
font-weight	321
height	188
left	278
letter-spacing	326
line-height	328
list-style	290
list-style-image	289
list-style-position	288
list-style-type	286
margin	209
margin-bottom	205
margin-left	207
margin-right	202
margin-top	200
max-height	192
max-width	198
min-height	190
min-width	196
-moz-border-radius	372
-moz-box-sizing	375
The display Property Value: -moz-inline-box	377
orphans	365
outline	261
outline-color	258

outline-style259
outline-width
overflow 280
padding
padding-bottom 215
padding-left216
padding-right213
padding-top212
page-break-after
page-break-before
page-break-inside
position 267
quotes
right
table-layout
text-align
text-decoration
text-indent
text-shadow
text-transform
top
unicode-bidi
vertical-align 338
visibility 273
white-space 341
widows
width
word-spacing
z-index
zoom