

# Some notes on CSS

John R Hudson\*

2nd October 2024

<b>Contents</b>		<b>8 Properties</b>	<b>21</b>
<b>1 Background</b>	<b>3</b>	8.1 align-content . . . . .	21
<b>2 Using styles</b>	<b>3</b>	8.2 align-items . . . . .	21
<b>3 Syntax</b>	<b>3</b>	8.3 align-self . . . . .	21
3.1 @ rules . . . . .	3	8.4 all . . . . .	22
3.2 Backslash . . . . .	8	8.5 azimuth . . . . .	22
3.3 Symbols . . . . .	8	8.6 background . . . . .	24
<b>4 Layout</b>	<b>9</b>	8.7 border . . . . .	26
4.1 The containing block . . . . .	9	8.8 border-radius . . . . .	27
4.2 Formatting context . . . . .	9	8.9 border-image . . . . .	28
4.3 Margins, borders and padding . . . . .	9	8.10 border-collapse . . . . .	28
4.4 Aural box model . . . . .	10	8.11 border-spacing . . . . .	29
4.5 Page boxes . . . . .	10	8.12 bottom . . . . .	29
4.6 Tables . . . . .	10	8.13 box-decoration-break . . . . .	29
<b>5 Units</b>	<b>11</b>	8.14 box-sizing . . . . .	29
5.1 Angle . . . . .	11	8.15 box-shadow . . . . .	30
5.2 Colour . . . . .	11	8.16 break-after . . . . .	30
5.3 Counters . . . . .	15	8.17 break-before . . . . .	31
5.4 Fonts . . . . .	15	8.18 break-inside . . . . .	32
5.5 Frequency . . . . .	17	8.19 caption-side . . . . .	33
5.6 Length . . . . .	17	8.20 caret-color . . . . .	33
5.7 Resolution . . . . .	17	8.21 clear . . . . .	33
5.8 Time . . . . .	18	8.22 clip-path . . . . .	33
5.9 calc( ) . . . . .	18	8.23 clip-rule . . . . .	34
<b>6 The structure of a rule</b>	<b>18</b>	8.24 color . . . . .	34
<b>7 Selectors</b>	<b>18</b>	8.25 column-fill . . . . .	34
		8.26 column-gap . . . . .	35
		8.27 column-rule . . . . .	35
		8.28 column-span . . . . .	35
		8.29 columns . . . . .	35
		8.30 contain . . . . .	36
		8.31 content . . . . .	36
		8.32 counter-increment . . . . .	37

---

\*The author would welcome notification of any errors or possible misunderstandings.

8.33	counter-reset	37	8.79	position	63
8.34	cue	37	8.80	quotes	63
8.35	cursor	38	8.81	resize	64
8.36	direction	39	8.82	rest	64
8.37	display	39	8.83	right	64
8.38	elevation	39	8.84	row-gap	65
8.39	empty-cells	40	8.85	speak	65
8.40	flex	40	8.86	speak-as	65
8.41	flex-flow	40	8.87	table-layout	65
8.42	float	41	8.88	text-align	65
8.43	font	41	8.89	text-combine-upright	66
8.44	font-variant	44	8.90	text-decoration	66
8.45	gap	46	8.91	text-indent	66
8.46	grid	47	8.92	text-orientation	66
8.47	grid-area	48	8.93	text-overflow	67
8.48	grid-column	49	8.94	text-transform	67
8.49	grid-row	49	8.95	top	67
8.50	height	50	8.96	transform	67
8.51	image-orientation	50	8.97	transform-box	68
8.52	isolation	50	8.98	transform-origin	68
8.53	justify-content	50	8.99	unicode-bidi	68
8.54	justify-items	51	8.100	vertical-align	69
8.55	justify-self	51	8.101	visibility	70
8.56	left	53	8.102	voice-balance	70
8.57	letter-spacing	53	8.103	voice-duration	70
8.58	line-height	53	8.104	voice-family	71
8.59	list-style	53	8.105	voice-pitch	71
8.60	margin	54	8.106	voice-range	71
8.61	mask	55	8.107	voice-rate	72
8.62	mix-blend-mode	58	8.108	voice-stress	72
8.63	object-fit	59	8.109	voice-volume	72
8.64	object-position	59	8.110	white-space	73
8.65	opacity	59	8.111	widows	73
8.66	order	59	8.112	width	73
8.67	orphans	60	8.113	will-change	74
8.68	outline	60	8.114	word-spacing	74
8.69	overflow	60	8.115	writing-mode	74
8.70	padding	61	8.116	z-index	74
8.71	page-break-after	61			
8.72	page-break-before	61	<b>9</b>	<b>Values</b>	<b>75</b>
8.73	page-break-inside	61	9.1	Image	75
8.74	pause	61			
8.75	place-content	62	<b>10</b>	<b>Inheritance</b>	<b>76</b>
8.76	place-items	62			
8.77	place-self	62	<b>A</b>	<b>Sources for this document</b>	<b>78</b>
8.78	play-during	62	<b>B</b>	<b>SVG 1.0 colours</b>	<b>79</b>

# 1 Background

During the 1990s there were several attempts to develop style sheets which separated the presentation of material on a website from its content and structure as defined by HTML. CSS, or Cascading Style Sheets, first published in 1996, differed from other attempts in that it allowed styles to be inherited, thereby avoiding the need to create separate style sheets for every page.

CSS2 was published in 1998 but adoption was slow and there were a number of errors in the specification which were remedied through a series of updates between 2007 and 2011 known as CSS2.1. Since then CSS development, sometimes known as ‘Level 3,’ has moved to rolling updates.

The most recent updates consulted for these notes are listed in [appendix A on page 78](#). Please note that these notes are intended as a quick reference and not as a replacement for reading the source documentation which is regularly updated (see [CSS Current work](#)).

## 2 Using styles

A style is defined by a rule which, unless otherwise specified, is assumed to be encoded in `utf-8` and which may be embedded

- within the `style=" "` attribute of an HTML element
- within the HTML `<style>... </style>` element

or included in a separate style sheet called from within

- the `<style>... </style>` element with the `@import "style-sheet.css";` directive (see section [3.1](#))
- the `<link . />` element with, for example,  
`<link rel="stylesheet" href="style-sheet.css" type="text/css"/>`

The last option is the preferred option with HTML.

## 3 Syntax

### 3.1 @ rules

@ rules (@ followed by a keyword) are terminated by a semi-colon and cannot occur in a block.

**@counter-style** allows the definition of counter styles which can then be used in [list-style-type](#) properties; a new counter style is defined by

```
@counter-style <counter-style-name> {system: <counter-system>;
symbols: <list of symbols>; negative: " "; prefix: " ";
suffix: " "; range: " "; pad: "<integer> && <symbol>"; speak-as: " ";
fallback: " "};
```

`<counter-style-name>` must not be "none" or "disc".

The following styles are predefined for system:

```
system: cyclic | numeric | alphabetic | symbolic | additive |
[fixed <integer>?] | [ extends <counter-style-name> ]
```

specifies which algorithm will be used to construct the counter's representation based on the counter value where:

**cyclic** cycles repeatedly through its provided symbols

**numeric** interprets the list of counter symbols as digits to a 'place-value' numbering system, similar to the default decimal counter style

**alphabetic** interprets the list of counter symbols as digits to an alphabetic numbering system, similar to the default lower-alpha counter style, which wraps from **a**, **b**, **c**, to **aa**, **ab**, **ac**.

**symbolic** cycles repeatedly through its provided symbols, doubling, tripling, etc. the symbols on each successive pass through the list

**additive** represents 'sign-value' numbering systems, which define additional digits with much larger values, so that the value of the number can be obtained by adding all the digits together, as in Roman numerals

**fixed** runs through its list of counter symbols once

**extends** system allows an author to use the algorithm of another counter style, but alter other aspects, such as the negative sign or the suffix.

In addition, the keyword **extends** can be used to extend an existing **list-style-type**; for example:

```
@counter-style decimal-paren { system: extends decimal; suffix: ") "; }
```

will create a decimal style terminated by a parenthesis rather than a full stop.

**symbols( cyclic | numeric | alphabetic | symbolic | fixed? [ <string> | <image> ]+ )** ; specifies the symbols to be used except where the **additive** algorithm is being used when **additive-symbols: [ <integer [0,∞]> && <symbol> ]#** performs this task

**negative: " "**; specifies how negative numbers are to be displayed

**prefix: " "**; specifies one or more symbols preceding the counter

**suffix: " "**; specifies one or more symbols following the counter; the default value is "\2E\20", that is, full stop followed by space

**range: [[ <integer> | infinite ]{2}]# | auto** ; may take a comma separated list of ranges in which **infinite** may be the first or second value in the range or **auto** which takes the range of the specified system

**pad: <integer [0,∞]> && <symbol>**; takes an integer specifying the amount of padding and a symbol to be used for padding where necessary

**fallback: <system>**; specifies the system to use if the specified system runs out of values

**speak-as: [auto | bullets | numbers | words | spell-out | <counter-style-name>]**; specifies how speech synthesisers are to interpret the specified system where:

**auto** if the counter style's system is **alphabetic**, = **spell-out**; if it is **cyclic**, = **bullets**; if it is **extends**, = **auto** for the extended style. Otherwise, it has the same effect as **numbers**.

**bullets** the UA speaks a cue that represents an unordered list item being read out.

**numbers** the counter's value is spoken as a number in the content language.

**words** the counter's value is spoken as normal text in the content language.

**spell-out** the counter's value is spoken letter-by-letter in the content language.

<**counter-style-name**> the counter's value is spoken out in the specified style or is treated as auto.

The following counter-styles are predefined:

Numeric:

decimal, decimal-leading-zero, arabic-indic, armenian,  
upper-armenian, lower-armenian, bengali, cambodian, khmer, cjk-decimal,  
devanagari, georgian, gujarati, gurmukhi, hebrew, kannada, lao,  
malayalam, mongolian, myanmar, oriya, persian, lower-roman, upper-roman,  
tamil, telugu, thai, tibetan

Alphabetic:

lower-alpha, lower-latin, upper-alpha, upper-latin, lower-greek,  
hiragana, hiragana-iroha, katakana, katakana-iroha

Symbolic:

disc, circle, square, disclosure-open, disclosure-closed

Fixed:

cjk-earthly-branch, cjk-heavenly-stem

**@font-face** allows for linking to fonts that are automatically fetched and activated when needed, allowing authors to select a font that closely matches the design goals for a given page rather than limiting the font choice to a set of fonts available on a given platform.

```
@font-face { font-family: <family-name>; src: url(<url>)  
[format (<format>)]; }
```

where <format> may be "woff", "woff2", "truetype", "opentype",  
"embedded-opentype" or "svg".

**@import** allows the importation of style sheets with, for example,

```
@import [ <url> | <string> ] <media-query-list>? ;  
@import "mystyle.css";  
@import url("mystyle.css");
```

Where an @import rule is only intended to apply to one media group, this may also be specified

```
@import url("fineprint.css") print;  
@import url("narrow.css") screen and (max-width: 400px);
```

@import rules may also use the syntax of @media rules.

**@media** introduces conditional rules intended for all, print and screen media:

```
@media screen and (<property>: <value>), print and  
(<property>: <value>) {...}
```

Note the alternative syntax for the DECLARATION (section 6), for example,

```
@media print {body {font-size: 10pt }}:  
@media screen {body {font-size: 13px }};  
@media screen, print {body {line-height: 1.2 }};  
width: <length> ;
```

describes the width of the targeted display area of the output device where `<length>` is a positive integer.

```
height: <length> ;
```

describes the height of the targeted display area of the output device where `<length>` is a positive integer.

```
aspect-ratio: <width>/<height> ;
```

specifies the aspect ratio of a media type where `<width>/<height>` are two positive integers.

```
orientation: portrait | landscape ;
```

indicates whether the value of `<height>` is greater than the value of `<width>`.

```
resolution: <resolution> | infinite ;
```

describes the resolution of the output device, i.e. the density of the pixels. When querying media with non-square pixels, resolution queries the density in the vertical dimension.

For output mediums that have no physical constraints on resolution (such as outputting to vector graphics), this feature must match the `infinite` value where `infinite` must be treated as larger than any possible `<resolution>`.

```
scan: interlace | progressive ;
```

describes the scanning process of some output devices:

**interlace** where video frames alternate between specifying only the ‘even’ lines on the screen and only the ‘odd’ lines. When displaying on interlaced screens, authors should avoid very fast movement across the screen to avoid ‘combing’, and should ensure that details on the screen are wider than 1px to avoid ‘twitter’.

**progressive** where rendering displays each screen fully, and needs no special treatment.

```
update: none | slow | fast ;
```

queries the ability of the output device to modify the appearance of content once it has been rendered. It accepts the following values:

**none** the layout can no longer be updated.

**slow** the layout may change dynamically according to the usual rules of CSS, but the output device is not able to render or display changes quickly enough for them to be perceived as a smooth animation.

**fast** the layout may change dynamically according to the usual rules of CSS, and the output device is not unusually constrained in speed.

```
overflow-block: none | scroll | paged ;
```

describes the behaviour of the device when content overflows the initial containing block:

**none** any overflowing content is simply not displayed

**scroll** overflowing content is exposed by allowing users to scroll to it

**paged** content that overflows one page is displayed on the following page.

```
overflow-inline: none | scroll ;
```

**none** any overflowing content is simply not displayed

**scroll** overflowing content is exposed by allowing users to scroll to it.

`color: <integer> ;`

describes the number of bits per colour component of the output device.

`color-index: <integer> ;`

describes the number of the number of entries in the colour lookup table of the output device.

`monochrome: <integer> ;`

describes the number of bits per pixel in a monochrome frame buffer.

`color-gamut: srgb | p3 | rec2020 ;`

describes the approximate range of colours that are supported by the UA and output device where

**srgb** means the UA and output device can support approximately the sRGB gamut or more.

**p3** means the UA and output device can support approximately the gamut specified by the DCI P3 Color Space or more.

**rec2020** means the UA and output device can support approximately the gamut specified by the ITU-R Recommendation BT.2020 Color Space or more.

**Note:** The **p3** gamut is larger than and includes the **srgb** gamut and the **rec2020** gamut is larger than and includes the **p3** gamut.

`pointer: none | coarse | fine ;`

queries the presence and accuracy of a pointing device such as a mouse where

**none** there is no pointing device

**coarse** the pointing device is of limited accuracy.

**fine** the device includes an accurate pointing device.

`any-pointer: none | coarse | fine ;`

queries the presence and accuracy of all pointing devices

`hover: none | hover ;`

queries the user's ability to hover over elements on the page with the primary pointing device.

**none** indicates that the primary pointing device can't hover, or that there is no pointing device.

**hover** indicates that the primary pointing device can easily hover over parts of the page.

`any-hover: none | hover ;`

queries the user's ability to hover over elements on the page with all pointing devices.

`device-width: <length> ;`

describes the width of the rendering surface of the output device.

`device-height: <length> ;`

describes the height of the rendering surface of the output device.

`device-aspect-ratio: <device-width>/<device-height> ;`

specifies the aspect ratio of a media type where `<device-width>/<device-height>` are two positive integers.

**@namespace** declares a default namespace, for example,

```
@namespace url("<url>");
```

while a namespace prefix may be declared with

```
@namespace prefix url("<url>");
```

**@page** introduces rules intended only for use with printed media; it takes the standard form for declarations (section 6):

```
@page: [first | left | right ] {<declaration>; ...;}
```

These cannot be used within `<style>` elements with the `scoped` attribute.

**@supports** interrogates the user agent to determine what features it supports:

```
@supports (<property>: <value> [!important]) { ... }
```

enabling alternative styling where a feature is or is not supported.

Multiple conditions must be linked with `and` or `or` and may be preceded by `not` but parentheses must be used to avoid any confusion.

A single declaration may apply to a comma separated list of elements and there can be space separated declarations for different elements but the second and all subsequent elements in the declaration must be preceded by `#` as in this example:

```
@supports ( display: flex ) {  
  body, #navigation, #content { display: flex; }  
  #navigation { background: blue; color: white; }  
  #article { background: white; color: black; } }
```

## 3.2 Backslash

The backslash may appear:

- in a comment
- outside a string followed by a newline character
- to cancel the meaning of a special CSS character, eg. `"\"`
- to introduce a hexadecimal string of up to six digits which represents a character.

## 3.3 Symbols

**[...]** whatever is enclosed in the brackets must occur in that order (the brackets may be omitted if this is obvious)

**&&** whatever is either side of the double ampersand must occur but not necessarily in that order

**||** one or more of what is either side of the double bar must occur but not necessarily in that order

**|** only one of the two things either side of the bar must occur

\* the preceding item occurs 0 or many times  
+ the preceding item occurs 1 or many times  
{*n*} the preceding item occurs *n* times  
{*n,o*} the preceding item occurs at least *n* times up to *o* times; omitting *o* means there is no maximum  
#{*n,o*} the preceding item occurs *n* times up to *o* times exactly  
! the preceding group is required.  
? the preceding item is optional  
/\* ... \*/ comment: cannot be nested

## 4 Layout

### 4.1 The containing block

Each element has a containing block; the initial containing block of the root element is the size of the viewport or page area. The positions of the containing blocks of all other elements are determined by the positions of the padding edges (Figure 1) of their ancestor, preceding or adjacent blocks unless their `position` is `fixed`, in which case it is determined by the viewport or page area.

### 4.2 Formatting context

Elements may be laid out within a block or an inline formatting context; block boxes, floats, absolutely positioned elements, block containers and most block boxes with overhang create a new block formatting context in which boxes are laid out vertically within their containing box with their left (or right in RTL contexts) edge against the left (or right) edge of the containing box. In an inline formatting context boxes are laid out horizontally between the edges of the containing box and, if their combined length exceeds the width of the containing box, split over more than one line.

*Flex boxes* have a flexible layout such that the children of a flex container can be laid out in any direction, and can “flex” their sizes, either growing to fill unused space or shrinking to avoid overflowing the parent. Both horizontal and vertical alignment of the children can be easily manipulated. Nesting of these boxes (horizontal inside vertical, or vertical inside horizontal) can be used to build layouts in two dimensions.

*Floats* are boxes that are shifted left (or right) from the containing box edge; content may flow down their right (or left) side if there is space and inline boxes are either shortened to fit the available space to the right (or left) or placed below the float. Immediately following floats are placed to the right (or left) of the float if there is space or below it otherwise.

*Grid layout boxes* are like flex boxes except that they are inherently two dimensional.

### 4.3 Margins, borders and padding

The area occupied by any content is defined by its width and height, plus the width of any margin, border or padding (figure 1). Where the width of all three is zero, the width and height of the area occupied by the content is the same as its content area.

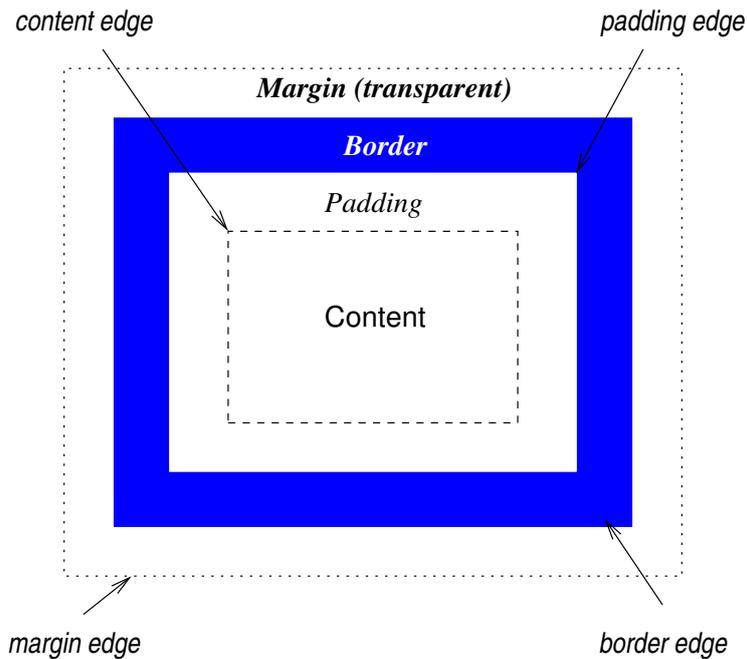


Figure 1: Box model (N.B. the default value of the margin and padding width is 0.)

The [box-sizing](#) property permits calculating the area to be occupied by any content by deducting the width of any margin, border or padding from a total size for the box.

The background property of the margin is always transparent; otherwise, the properties of the content box, padding, borders and margins may be defined in a rule.

In the case of inline boxes, the left and right margins, borders and padding are split between the first and last inline boxes in the element; thus assigning values to the left and right margins, borders and padding of inline boxes should be avoided.

#### 4.4 Aural box model

The aural box model (figure 2) illustrates the relationships between the [pause](#), [cue](#) and [rest](#) properties and the [voice-\\*](#) properties they interact with ([CSS Speech Module Level 1](#)).

#### 4.5 Page boxes

A page box contains the *page area* and the *margin*. The margins of a page box can be specified within an `@page` rule. Only percentage values can be used within a page box.

#### 4.6 Tables

Tables should only be used for the representation of data relationships, not for layout or presentation purposes.

Tables may be contained in block or inline elements.

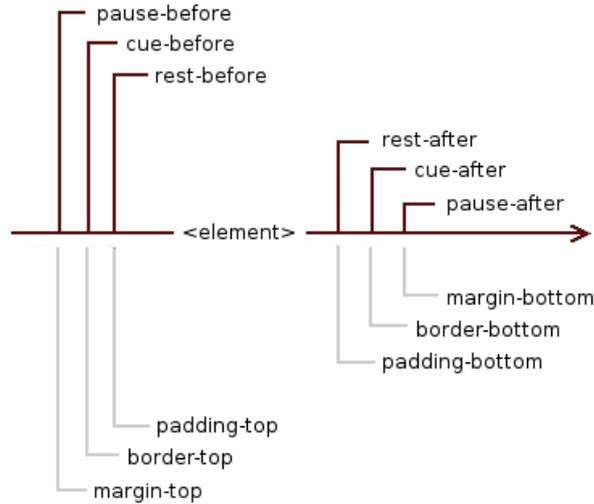


Figure 2: Aural box model

Tables are defined by their rows (header and data) so that columns may be defined implicitly from the cells in the rows.

Column and column group elements take the values of the [border](#), [background](#), [visibility](#) and [width](#) properties of their ancestor elements.

## 5 Units

### 5.1 Angle

The units of angles are:

**deg** degrees; there are 360 degrees in a full circle.

**grad** grads; there are 400 gradians in a full circle.

**rad** radians; there are  $2\pi$  radians in a full circle.

**turn** turns; there is 1 turn in a full circle.

where a negative value is always equivalent to a positive value.

### 5.2 Colour

Colour values can be specified in a variety of ways, of which four are now considered [Legacy syntax](#):

- using `<named-color>` from the SVG 1.0 colour keywords (see [appendix B on page 79](#)):

aliceblue antiquewhite aqua aquamarine azure beige bisque  
black blanchetdmondblue blueviolet brown burlywood cadetblue  
chartreuse chocolate coral cornflowerblue cornsilk crimson cyan  
darkblue darkcyan darkgoldenrod darkgray darkgreen darkgrey  
darkkhaki darkmagenta darkolivegreen darkorange darkorchid

darkred darksalmon darkseagreen darkslateblue darkslategray  
 darkslategrey darkturquoise darkviolet deeppink deepskyblue  
 dimgray dimgrey dodgerblue firebrick floralwhite forestgreen  
 fuchsia gainsboro ghostwhite gold goldenrod gray green  
 greenyellow grey honeydew hotpink indianred indigo ivory khaki  
 lavender lavenderblush lawngreen lemonchiffon lightblue  
 lightcoral lightcyan lightgoldenrodyellow lightgray lightgreen  
 lightgrey lightpink lightsalmon lightseagreen lightskyblue  
 lightslategray lightslategrey lightsteelblue lightyellow lime  
 limegreen linen magenta maroon mediumaquamarine mediumblue  
 mediumorchid mediumpurple mediumseagreen mediumslateblue  
 mediumspringgreen mediumturquoise mediumvioletred midnightblue  
 mintcream mistyrose moccasin navajowhite navy oldlace olive  
 olivedrab orange orangered orchid palegoldenrod palegreen  
 paleturquoise palevioletred papayawhip peachpuff peru pink plum  
 powderblue purple rebeccapurple red rosybrown royalblue  
 saddlebrown salmon sandybrown seagreen seashell sienna silver  
 skyblue slateblue slategray slategrey snow springgreen  
 steelblue tan teal thistle tomato turquoise violet wheat  
 white whitesmoke yellow yellowgreen

- using `<hex-color>`

```
color: #hex;
```

Where both digits are the same as in `#ff0000`, the hex string can be shortened to `#f00`.

- using RGB values

```
rgb(r g b / <alphavalue>);
rgb([0 - 255] [0 - 255] [0 - 255]);
rgb([0-100%] [0-100%] [0-100%]);
```

- using HSL values

```
hsl(h s l / <alphavalue>);
hsl(<hue> [0-100%] [0-100%]);
```

The `<hsl()>`, `<hsla()>`, `<hwb()>`, `<lch()>`, and `<oklch()>` colour functions use

```
<hue> = <number> | <angle>
```

where `<number>` is a number of degrees.

- using the HWB cylindrical coordinate model:

```
hwb(h s l / <alphavalue>);
hwb(<hue> [0-100%] [0-100%]);
```

### 5.2.1 Device-independent colours

- using the CIE LAB rectangular coordinate model:

```
lab(L a b / <alphavalue>);
lab([0 - 100] [-125 - 125] [-125 - 125]);
lab([0-100%] [-100%-100%] [-100%-100%]);
```

- using the CIE LCH cylindrical coordinate model:

```
lch(L C H / <alphavalue>);
lch([0 - 100] [0 - 150] <hue>);
lch([0-100%] [0-100%] <hue>);
```

- using the Oklab rectangular coordinate model:

```
oklab(L a b / <alphavalue>);
oklab([0 - 1] [-0.4 - 0.4] [-0.4 - 0.4]);
oklab([0-100%] [-100%-100%] [-100%-100%]);
```

- using the Oklch cylindrical coordinate model:

```
oklch(L C H / <alphavalue>);
oklch([0 - 1] [0 - 0.4] <hue>);
oklch([0-100%] [0-100%] <hue>);
```

### 5.2.2 Predefined colours

- using sRGB values:

```
color: color(srgb r g b);
color: color(srgb [0.0 - 1.0] [0.0 - 1.0] [0.0 - 1.0]);
color: color(srgb [0-100%] [0-100%] [0-100%]);
```

- using Linear-light sRGB values:

```
color: color(srgb-linear r g b);
color: color(srgb-linear [0.0 - 1.0] [0.0 - 1.0] [0.0 - 1.0]);
color: color(srgb-linear [0-100%] [0-100%] [0-100%]);
```

- using Display P3 values:

```
color: color(display-p3 r g b);
color: color(display-p3 [0.0 - 1.0] [0.0 - 1.0] [0.0 - 1.0]);
color: color(display-p3 [0-100%] [0-100%] [0-100%]);
```

- using a98 RGB values:

```
color: color(a98-rgb r g b);
color: color(a98-rgb [0.0 - 1.0] [0.0 - 1.0] [0.0 - 1.0]);
color: color(a98-rgb [0-100%] [0-100%] [0-100%]);
```

- using ProPhoto RGB values:

```
color: color(prophoto-rgb r g b);
color: color(prophoto-rgb [0.0 - 1.0] [0.0 - 1.0] [0.0 - 1.0]);
color: color(prophoto-rgb [0-100%] [0-100%] [0-100%]);
```

- using ITU-R BT.2020-2 values:

```
color: color(rec2020 r g b);
color: color(rec2020 [0.0 – 1.0] [0.0 – 1.0] [0.0 – 1.0]);
color: color(rec2020 [0-100%] [0-100%] [0-100%]);
```

- using CIE XYZ values:

```
color: color(xyz-d50/xyz-d65/xyz x y z);
color: color(xyz-d50/xyz-d65/xyz [0.0 – 1.0] [0.0 – 1.0] [0.0 – 1.0]);
color: color(xyz-d50/xyz-d65/xyz [0-100%] [0-100%] [0-100%]);
```

### 5.2.3 Legacy syntax

- using RGB values

```
color: rgb(r%,g%,b%);
color: rgb([0-255], [0-255], [0-255]);
color: rgb([0-100%], [0-100%], [0-100%]);
```

- using RGBA values

```
color: rgba(r%,g%,b%,0.5);
color: rgba([0-255], [0-255], [0-255], 0.5);
color: rgba([0-100%], [0-100%], [0-100%], 0.5);
```

- using HSL values

```
color: hsl(h,s%,l%);
hsl(<hue>, [0-100%], [0-100%]);
```

- using HSLA values

```
color: hsla(h,s%,l%,1);
hsla(<hue>, [0-100%], [0-100%], 100%);
```

### 5.2.4 Transparent

- using `transparent` which is interpreted as:

```
color: rgb(0 0 0 0);
```

### 5.2.5 Current colour

- using `currentcolor` to represent the value of the `color` property on the same element.

### 5.2.6 System colour

The `<system-color>` keywords are defined as follows:

**Canvas** Background of application content or documents.

**CanvasText** Text in application content or documents.

**LinkText** Text in non-active, non-visited links. For light backgrounds, traditionally blue.

**VisitedText** Text in visited links. For light backgrounds, traditionally purple.

**ActiveText** Text in active links. For light backgrounds, traditionally red.

**ButtonFace** The face background colour for push buttons.

**ButtonText** Text on push buttons.

**ButtonBorder** The base border colour for push buttons.

**Field** Background of input fields.

**FieldText** Text in input fields.

**Highlight** Background of selected text, for example from `::selection`.

**HighlightText** Text of selected text.

**SelectedItem** Background of selected items, for example a selected checkbox.

**SelectedItemText** Text of selected items.

**Mark** Background of text that has been specially marked (such as by the HTML mark element).

**MarkText** Text that has been specially marked (such as by the HTML mark element).

**GrayText** Disabled text. (Often, but not necessarily, gray.)

**AccentColor** Background of accented user interface controls.

**AccentColorText** Text of accented user interface controls.

### 5.3 Counters

Counters are referred to by case-sensitive identifiers;

```
counter(<identifier>)  
counter(<identifier>,<list-style-type>)
```

See [list-style](#) on page 53.

The default style is `decimal`.

To refer to a sequence of nested counters of the same name, use:

```
counters(<identifier>,<string>)  
counters(<identifier>,<string>,<list-style-type>)
```

The `identifier` must not be `none`, `inherit` or `initial`.

### 5.4 Fonts

Fonts<sup>1</sup> are selected on the basis that they support the required glyphs; a user agent should substitute a different font from the same generic font family or use a font from a different font family if a particular glyph is not supported by the specified font or fonts available within that font family.

The generic font families are:

**serif** for example, Times New Roman, Bodoni, Garamond, Minion Web, ITC Stone Serif, MS Georgia, Bitstream Cyberbit

---

<sup>1</sup>A font is a particular set of glyphs in a particular style with a particular weight; a typeface is a set of related fonts for which the term 'font family' is used in the CSS guidelines.

Table 1: Font weight

Weight		<b>bolder</b>	lighter
100	thin	400	100
200	extra light	400	100
300	light	400	100
400	normal	700	100
500	medium (where there is a normal weight font)	700	100
600	semi bold	900	400
700	bold	900	400
800	extra bold	900	700
900	black	900	700



Figure 3: Text baselines

**sans-serif** for example, MS Trebuchet, ITC Avant Garde Gothic, MS Arial, MS Verdana, Univers, Futura, ITC Stone Sans, Gill Sans, Akzidenz Grotesk, Helvetica

**cursive** for example, Caffisch Script, Adobe Poetica, Sanvito, Ex Ponto, Snell Round-hand, Zapf-Chancery

**fantasy** for example, Alpha Geometrique, Critter, Cottonwood, FB Reactor, Studz

**monospace** Courier, MS Courier New, Prestige, Everson Mono

Font weights take the values described in Table 1.

If a font with a weight below 400 is missing, each unassigned lower value in descending order is checked followed by each higher value and the closest reasonable value is assigned.

If the weights 400 or 500 are missing, the weights 500 or 400 are checked and then the preceding rule is followed.

If a weight above 500 is missing, each unassigned higher value in ascending order is checked followed by each lower value and the closest reasonable value is assigned.

Font position is calculated relative to the text baselines (figure 3).

## 5.5 Frequency

The units of frequency are:

**Hz:** Hertz

**kHz:** kilohertz

which may not be negative.

## 5.6 Length

Units may be relative or absolute; percentages or relative units are preferred. The relative units of font-size are:

**em** the em width of the current font

**ex** the x-height of the current font

**ch** the character advance of a narrow glyph, such as 0, in the current font

**rem** the font size of the root element

The relative units of the viewport are:

**vw** 1% of the viewport's width

**vh** 1% of the viewport's height

**vmin** 1% of viewport's smaller dimension

**vmax** 1% of viewport's larger dimension

Note that an inherited value (see section [10 on page 76](#)) will be absolute, even if computed from a percentage or relative value in the parent document.

The absolute units are:

**cm** centimetre

**mm** millimetre

**Q** quarter of a millimetre

**in** inch

**pc** pica =  $1/6$  in

**pt** point =  $1/72$ in

**px** pixel =  $1/96$  in

In section [8, Properties](#), where a property takes an absolute value for `<length>`, this value will use one or other of the above units.

## 5.7 Resolution

The units of resolution are:

**dpi** dots per CSS inch

**dpcm** dots per CSS centimetre

**dppx** (or **x**) dots per px unit

## 5.8 Time

The units of time are:

**ms:** milliseconds

**s:** seconds

They may not be negative.

## 5.9 calc( )

The `calc()` function allows a numeric CSS component value to be written as a mathematical expression using addition (+), subtraction (-), multiplication (\*) and/or division (/). It is evaluated using standard operator precedence rules (\* and / bind tighter than + and -, and operators are otherwise evaluated left-to-right).

It can be used wherever [Length](#), [Frequency](#), [Angle](#), [Time](#), `<percentage>`, `<number>`, or `<integer>` values are allowed. Components of a `calc()` expression can be literal values or `calc()` expressions.

## 6 The structure of a rule

A CSS rule consists of a `SELECTOR` containing one or more comma separated HTML elements<sup>2</sup> to which the rule applies followed by any number of declarations; each `DECLARATION` consists of a property followed by a colon and one or more values terminated by a semi-colon with all the declarations enclosed in braces

```
SELECTOR          {DECLARATION;          DECLARATION;          ...}
element, element, ... {property: value value ...; property: value value ...; ...;}
```

## 7 Selectors

In addition to HTML elements, selectors may contain various CSS elements: attribute and value elements, substring matching elements or pseudo-class elements,<sup>3</sup> optionally ending in a pseudo-element. If any of the elements in a `SELECTOR` are invalid, the entire rule is ignored.

The following are valid members of a `SELECTOR`:

\* any element

**e** element **e**

The specificity of a `SELECTOR` is calculated by concatenating:

- the number of ID selectors (**e#myid**),
- the number of class (**e.value**), attribute and pseudo-class selectors
- the number of type and pseudo-element selectors.

---

<sup>2</sup>Space separated elements take on a different meaning; see below **e f**.

<sup>3</sup>or HTML elements, not recommended as yet.

### Attribute and value elements

**e[attribute]** element *e* when attribute is set

**e[attribute="value"]** element *e* when the value of attribute exactly equals "value"

**e[attribute~="value"]** element *e* when value "value" appears in a list of values for the attribute

**e.value** element *e* when the `class=" "` attribute is set to "value" — equivalent to **e[class~="value"]**. — for example:

**e.warning** element *e* when the attribute `class="warning"`

**e.value1.value2** element *e* when the `class=" "` attribute contains both "value1" and "value2"

Note that `*.value` specifies all elements whose `class=" "` attribute is set to "value"

Such an element may also belong to a pseudo-element, e.g. `e.value:visited`.

### Substring matching elements

**e[attribute^="val"]** element *e* the value of whose attribute begins with the string "val"

**e[attribute\$="lue"]** element *e* the value of whose attribute ends with the string "lue"

**e[attribute\*="alu"]** element *e* the value of whose attribute contains the string "alu"

**e[attribute|="en"]** element *e* the value of whose attribute contains a hyphen separated list of values beginning with the string "en"

### Pseudo-class elements

**e:root** element *e* where it is the root of the document

**e:nth-child(n)** element *e* where it is the *n*th child of its parent

**e:nth-last-child(n)** element *e* where it is the *n*th child counting from the last child of its parent

**e:nth-of-type(n)** element *e* where it is the *n*th sibling of its type

**e:nth-last-of-type(n)** element *e* where it is the *n*th sibling counting from the last sibling of its type

**e:first-child** element *e* where it is the first child of its parent

**e:last-child** element *e* where it is the last child of its parent

**e:first-of-type** element *e* where it is the first sibling of its type

**e:last-of-type** element *e* where it is the last sibling of its type

**e:only-child** element *e* where it is the only child of its parent

**e:only-of-type** element *e* where it is the only sibling of its type

**e:empty** element *e* where it has no children

**e:target** element *e* where it is the target of a referring URL

**e:lang(la)** element *e* if it is in language *la*

**e:not(s)** element *e* when it does not match simple selector *s*

### Pseudo-class selectors in HTML

**e:active** element *e* where it is a `<link>`, `<a>` or `<area>` element with the `href=" "` attribute, a `<button>` or `<input>` element whose type is in the button state or a `<menuitem>` element that is not disabled

**e:checked** element *e* where it is an `<input>` or `<menuitem>` element whose `type="checkbox"` or `"radio"` or an `<option>` element that has been selected

**e:disabled** element *e* where it is disabled

**e:enabled** element *e* where it is a `<button>`, `<input>`, `<select>`, `<textarea>` or `<option>` element or an `<optgroup>` or `<menuitem>` element that it is not disabled

**e:focus** element *e* when it is in focus

(Use the [outline](#) property to define any outlining of the four pseudo-class elements above.)

**e:hover** element *e* when the pointer is hovering over it

**e:link** element *e* where it is a `<link>`, `<a>` or `<area>` element with an `href=" "` attribute before the link has been visited

**e:visited** element *e* where it is a `<link>`, `<a>` or `<area>` element with an `href=" "` attribute that has been visited

**e:warning** element *e* whose class is "warning" (the document language specifies how class is determined).

### Pseudo-elements

**e::first-line** applies a style to the first line of element *e*; NB. in a left-justified first line, the background does not necessarily extend all the way to the right margin.

**e::first-letter** applies a style to the first letter of element *e*

**e::before** places content before element *e*; the content inherits any inheritable properties of its element

**e::after** places content after an element *e*; the content inherits any inheritable properties of its element

Only one of the pseudo elements may be appended to a list of selectors.

### The id selector

**e#myid** element *e* when the attribute `id="myid"`. Note that no two `id=" "` attributes in a document can have the same value.

### Combination elements

**e f** element *f* when it is a descendant of element *e*

**e>f** element *f* when it is a child of element *e*

**e+f** element *f* when it immediately follows a sibling element *e*

**e~f** element *f* when it follows a sibling element *e*

Combination elements may be combined.

### Namespace elements

Elements which have been declared in a namespace (as defined in XML 1.0) may be used as follows:

**ns|e** elements with name *e* in namespace *ns*

**\*|e** elements with name *e* in any namespace, including those without a namespace

**|e** elements with name *e* without a namespace

**e** if no default namespace has been declared for selectors, this is equivalent to `*|e`. Otherwise it is equivalent to `ns|e` where `ns` is the default namespace.

In the above examples **e** may be `*`.

See section [3.1](#) for namespace declarations.

## 8 Properties

### 8.1 align-content

```
align-content: flex-start | flex-end | center |  
space-between | space-around | stretch ;
```

aligns the contents of the box as a whole within the box itself along the block/column/cross axis of the box where

**flex-start** aligns the `flex-item` to be flush with the edge of the `flex-container` corresponding to the `flex-container`'s cross-start side.

**flex-end** aligns the `flex-item` to be flush with the edge of the `flex-container` corresponding to the `flex-container`'s cross-end side. `space-between` which distributes items evenly along the vertical axis; if there is insufficient space, this is the equivalent of **start**

**center** centres the item within its container.

**space-between** distributes lines evenly in the flex container. If the leftover free-space is negative or there is only a single flex line in the flex container, this value is identical to **flex-start**. Otherwise, the cross-start edge of the first line in the flex container is placed flush with the cross-start content edge of the flex container, the cross-end edge of the last line in the flex container is placed flush with the cross-end content edge of the flex container, and the remaining lines in the flex container are distributed so that the spacing between any two adjacent lines is the same.

**space-around** distributes items evenly along the vertical axis with a half space at either end; if there is insufficient space, this is the equivalent of **center**

**stretch** stretches lines to take up the remaining space. If the leftover free-space is negative, this value is identical to **flex-start**. Otherwise, the free-space is split equally between all of the lines, increasing their cross size (see [figure 4 on page 23](#)).

It is not inherited and the default value is **stretch**.

### 8.2 align-items

```
align-items: flex-start | flex-end | center | baseline | stretch ;
```

specifies the default **align-self** for all of the child boxes in a box. See **align-self** for an explanation of the permissible values.

It is not inherited and the default value is **stretch**.

### 8.3 align-self

```
align-self: auto | flex-start | flex-end | center | baseline |  
stretch ;
```

specifies the vertical alignment of items in a container along the block/column/cross axis of the container where

`<baseline-position>` values may be

**auto** defers cross-axis alignment control to the value of `align-items` on the parent box.

**flex-start** aligns the `flex-item` to be flush with the edge of the `flex-container` corresponding to the `flex-container`'s cross-start side.

**flex-end** aligns the `flex-item` to be flush with the edge of the `flex-container` corresponding to the `flex-container`'s cross-end side. `baseline` uses the first box's baseline

**center** centres the flex item's margin box in the cross axis within the line

**baseline** aligns all participating flex items on the line such that their baselines align, and the item with the largest distance between its baseline and its cross-start margin edge is placed flush against the cross-start edge of the line.

**stretch** stretches the flex item using the length necessary to make the cross size of the item's margin box as close to the same size as the line as possible (figure 5 on the following page).

The default value is `auto`.

## 8.4 all

```
all: initial | inherit | unset ;
```

is a shorthand that resets all CSS properties except `direction` and `unicode-bidi`.

**initial** sets the property's specified value is its initial value.

**inherit** sets the property's specified and computed values to their inherited values.

**unset** if it is an inherited property, this is treated as `inherit`, and if it is not, this is treated as `initial`. It effectively erases all declared values occurring earlier in the cascade, correctly inheriting or not as appropriate for the property.

## 8.5 azimuth

```
azimuth: <angle> | [[ left-side | far-left | left |  
center-left | center | center-right | right | far-right |  
right-side ] || behind ] | leftwards | rightwards | inherit ;
```

specifies the spatial audio properties of an aural<sup>4</sup> element.

The azimuth values are:

`<angle>` refers to the position of the sound source with 0deg referring to the front, 90deg to the right, 180deg behind and 270deg to the left; negative values represent the equivalent positive value.

**left-side** 270deg. With `behind`, 270deg

**far-left** 300deg. With `behind`, 240deg

**left** 320deg. With `behind`, 220deg

**center-left** 340deg. With `behind`, 200deg

---

<sup>4</sup>The aural category will be replaced by the audio and speech categories in due course.

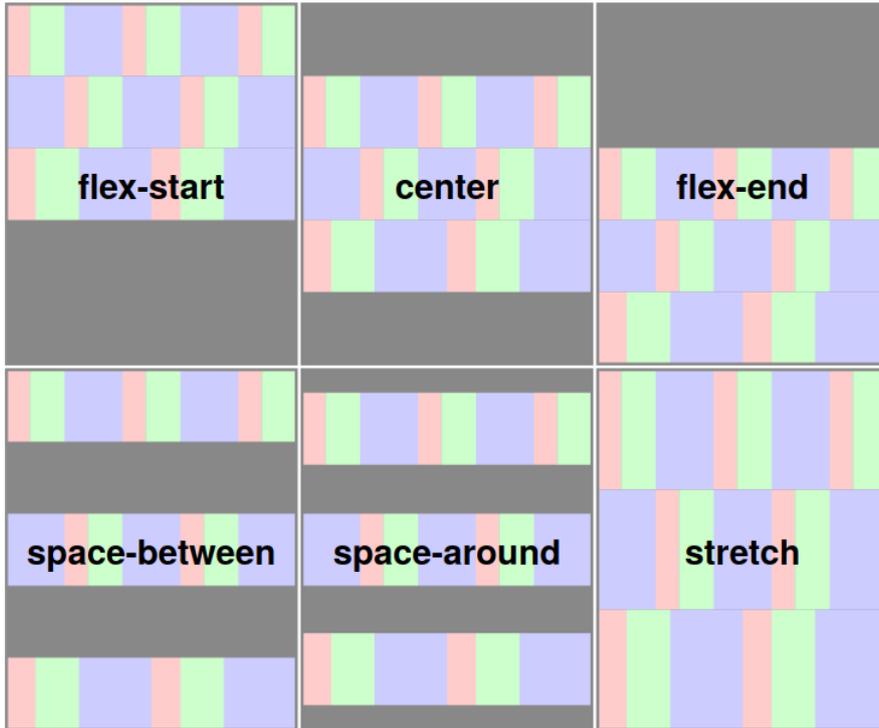


Figure 4: *align-content*

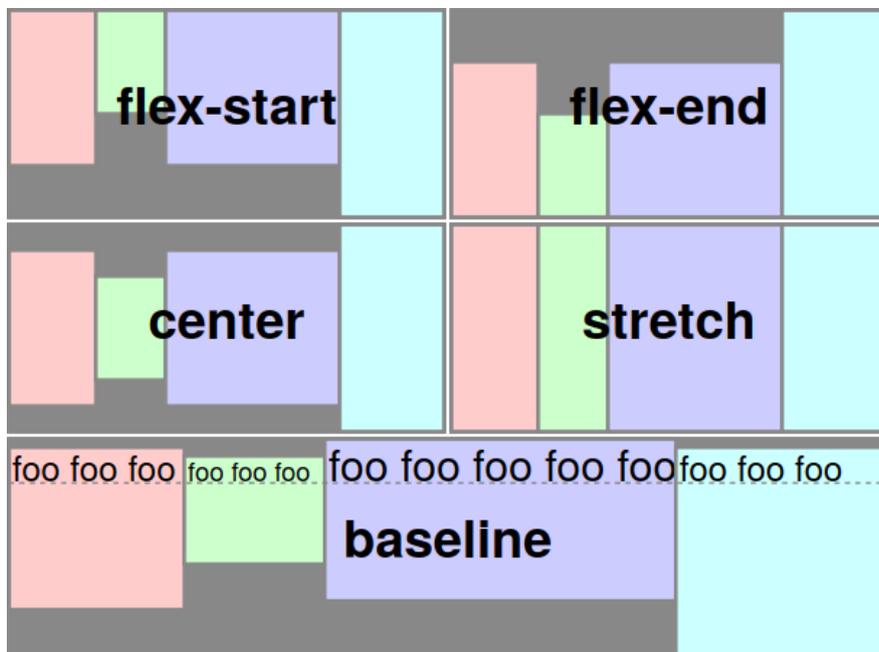


Figure 5: *align-items*

**center** 0deg. With **behind**, 180deg  
**center-right** 20deg. With **behind**, 160deg  
**right** 40deg. With **behind**, 140deg.  
**far-right** 60deg. With **behind**, 120deg  
**right-side** 90deg. With **behind**, 90deg  
**behind** 180deg  
**leftwards** subtracts 20 degrees  
**rightwards** adds 20 degrees  
The default value is **center**.

## 8.6 background

```
background: [<background-image> || <background-position>
 [ / <background-size> ]? || <repeat-style> || <attachment> ||
 <box> || <box># ,] ? <background-color> || <background-image> ||
 <background-position> [ / <background-size> ]? || <repeat-style> ||
 <attachment> || <box> || <box> ;
```

offers a shorthand for setting the background properties.

```
background-color: <color> | transparent ;
```

specifies the background colour (see [Colour](#) on page 11).

Setting `background-color: transparent` ensures that links display correctly. The value of `background-color` only applies to table columns if its value is `transparent` for both table rows and table cells.

It is not inherited by default and its default value is `transparent`.

```
background-image: url("<url>") | none ;
```

specifies the background image for an element. A `background-color` should also be specified; this will take the place of the image if it is not available and will show through any transparent regions of the image. The dimensions of the image will be calculated with reference to the values of the `background-position` property.

It is not inherited and the default value is `none`.

```
background-repeat: repeat | repeat-x | repeat-y | no-repeat |
 space | round | inherit ;
```

specifies whether and how a `background-image` should be repeated (tiled).

The repeat values may be:

**repeat** repeat horizontally and vertically

**repeat-x** repeat horizontally

**repeat-y** repeat vertically

**no-repeat** do not repeat

**space** the image is repeated as often as will fit within the background positioning area without being clipped and then the images are spaced out to fill the area.

**round** the image is repeated as often as will fit within the background positioning area. If it doesn't fit a whole number of times, it is rescaled so that it does.

It is not inherited and the default value is `repeat`.

```
background-attachment: scroll | fixed | local ;
```

specifies whether a `background-image` should scroll, remain fixed to the border area of the content or scroll if the contents of the element scroll; n.b. the value `scroll` will make an image appear fixed in a scrolling mechanism.

It is not inherited and the default value is `scroll`.

```
background-position: [ left | center | right | top | bottom |
<length-percentage> ] |
[ left | center | right | <length-percentage> ] |
[ top | center | bottom | <length-percentage> ] |
[ center | [ left | right ] <length-percentage>? ] &&
[ center | [ top | bottom ] <length-percentage>? ] ;
```

specifies the initial position of a `background-image`; the first value specifies its horizontal and the second its vertical position. In the absence of a second value, this is assumed to be `center`.

The position values are:

`<percentage>` refers to the width and height of the padding block of the image.

`<length>` refers to the distance horizontally or vertically from the top left corner of the padding box

**left** = 0% for the horizontal position

**center** = 50% for the horizontal or vertical position

**right** = 100% for the horizontal position

**top** = 0% for the vertical position

**bottom** = 100% for the vertical position

It is not inherited and the default value is `0% 0%`.

```
background-clip: [border-box | padding-box | content-box];
```

determines the background painting area. It is not inherited and the default value is `border-box`.

```
background-origin: [padding-box | border-box | content-box];
```

specifies the background positioning area. It is not inherited and the default value is `padding-box`.

`background-origin` has no effect if `background-attachment: fixed`.

```
background-size: [ <length-percentage [0,∞]> | auto ]{1,2} |
cover | contain ] ;
```

specifies the size of the background image.

**contain** scales the image as far as possible while retaining its natural aspect ratio to fit inside the background positioning area

**cover** scales the image while retaining its natural aspect ratio so that it completely covers the background positioning area

`[<length-percentage [0,∞]> | auto ]{1,2}` the first value gives the width of the corresponding image, the second value its height. If only one value is given the second is assumed to be `auto`. A percentage is relative to the background positioning area.

## 8.7 border

```
border: <border-width> || <border-style> || <border-color> ;
```

offers a shorthand for setting the same values for all four borders.

```
border-top: <border-width> || <border-style> ||  
<border-color> ;  
border-right: <border-width> || <border-style> ||  
<border-color> ;  
border-bottom: <border-width> || <border-style> ||  
<border-color> ;  
border-top: <border-width> || <border-style> ||  
<border-color> ;
```

offer shorthands for setting the values of individual borders.

### 8.7.1 border-color

```
border-color: <color> | transparent ;
```

offers a shorthand for setting the same colour for all four borders.

```
border-top-color:<color> | transparent ;  
border-right-color:<color> | transparent ;  
border-bottom-color:<color> | transparent ;  
border-left-color:<color> | transparent ;
```

specify the colours of individual borders.

Values are not inherited and the default value is the value of the `color` property.

### 8.7.2 border-style

```
border-style: none | hidden | dotted | dashed | solid |  
double | groove | ridge | inset | outset ;
```

offers a shorthand for setting the same style for all four borders.

The style values may be (figure 6):

**none** sets border-width to zero

**hidden** same as **none** when used in tables where it takes precedence if the value of [border-collapse](#) is collapse

**dotted** a sequence of dots

**dashed** a sequence of dashes

**solid** a single solid line

**double** two solid lines on either side of the border

**groove** appears to be carved in the surface

**ridge** appears to come up from the surface

**inset** makes the box appear embedded in the surface, the same as **groove** if the value of [border-collapse](#) is collapse

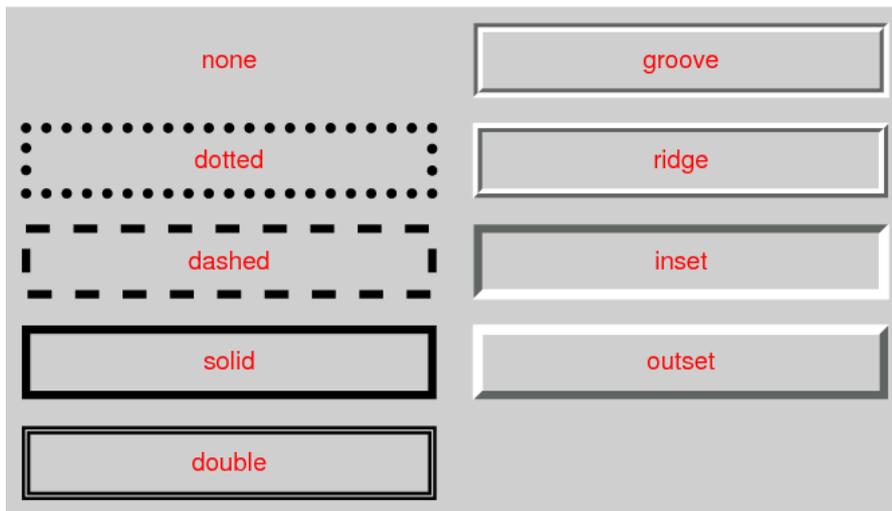


Figure 6: Border styles

**outset** makes the box appear proud of the surface, the same as **ridge** if the value of **border-collapse** is **collapse**

```
border-top-style: none | hidden | dotted | dashed | solid |
double | groove | ridge | inset | outset ;
border-right-style: none | hidden | dotted | dashed | solid |
double | groove | ridge | inset | outset ;
border-bottom-style: none | hidden | dotted | dashed | solid |
double | groove | ridge | inset | outset ;
border-left-style: none | hidden | dotted | dashed | solid |
double | groove | ridge | inset | outset ;
```

specify the styles of individual borders.

Values are not inherited and *there is no default value; so a border style must be set.*

### 8.7.3 border-width

```
border-width: <length [0,∞]> | thin | medium | thick ;
```

offers a shorthand for setting the same border width for all four borders.

```
border-top-width: <length [0,∞]> | thin | medium | thick ;
border-right-width: <length [0,∞]> | thin | medium | thick ;
border-bottom-width: <length [0,∞]> | thin | medium | thick ;
border-left-width: <length [0,∞]> | thin | medium | thick ;
```

specify the widths of individual borders. The default value is **medium**.

### 8.8 border-radius

```
border-radius: <length-percentage [0,∞]>{1,4}
[ / <length-percentage [0,∞]>{1,4} ]? ;
```

is shorthand for

```
border-top-left-radius: <length-percentage [0,∞]>{1,2} ;
border-top-right-radius: <length-percentage [0,∞]>{1,2} ;
border-bottom-right-radius: <length-percentage [0,∞]>{1,2} ;
border-bottom-left-radius: <length-percentage [0,∞]>{1,2} ;
```

which sets the radii of a quarter ellipse that defines the shape of the corner of the outer border edge.

## 8.9 border-image

```
border-image: <border-image-source> || <border-image-slice>
[ / <border-image-width> | / <border-image-width>? /
<border-image-outset> ]? || <border-image-repeat> ;
```

is shorthand for setting `border-image-source`, `border-image-slice`, `border-image-width`, `border-image-outset` and `border-image-repeat`. Omitted values are set to their initial values.

```
border-image-source: none | <image> ;

border-image-slice: [<number [0,∞]> | <percentage [0,∞]>]{1,4} &&
fill? ;
```

The default value is 100%.

**fill** causes the middle part of the border-image to be preserved.

```
border-image-width: [ <length-percentage [0,∞]> |
<number [0,∞]> | auto ]{1,4} ;
```

**auto** is the natural width or height (whichever is applicable) of the corresponding image slice.

```
border-image-outset: [ <length [0,∞]> | <number [0,∞]> ]{1,4} ;
```

The default value is 0.

```
border-image-repeat: [ stretch | repeat | round | space ]{1,2} ;
```

**stretch** the image is stretched to fill the area.

**repeat** the image is tiled (repeated) to fill the area.

**round** the image is tiled (repeated) to fill the area. If it does not fill the area with a whole number of tiles, the image is rescaled so that it does.

**space** the image is tiled (repeated) to fill the area. If it does not fill the area with a whole number of tiles, the extra space is distributed around the tiles.

The default value is **stretch**.

## 8.10 border-collapse

```
border-collapse: collapse | separate | inherit ;
```

specifies whether and how much space there is borders between table rows, columns and cells. If the value of `border-collapse` is `collapse`, and the value of `border-style` is `hidden`, these borders take precedence over all other borders; if not, wider borders take precedence over narrower ones, and cell border colours take precedence over row, column and table colours. The default value is `separate`.

### 8.11 border-spacing

```
border-spacing: <length> <length>? | inherit ;
```

specifies the spacing where the value of `border-collapse` is `separate`; if there are two values, the first applies to the horizontal and the second to the vertical borders.

The default value is 0.

```
empty-cells: show | hide | inherit ;
```

specifies whether borders or backgrounds are to be drawn for empty cells where the value of `border-collapse` is `separate`.

The default value is `show`.

### 8.12 bottom

See [position](#) on page 63

### 8.13 box-decoration-break

```
box-decoration-break: slice | clone ;
```

specifies, when a break splits a box,

- whether the box's margins, borders, padding, and other decorations wrap the broken edges of the box fragments
- how the background positioning area is derived from or duplicated across the box fragments and how the element's background is drawn within them.

The values are:

**clone** each box fragment is independently wrapped with the border, padding, and margin.

**slice** the element is rendered as if with no breaks present, and then sliced by the breaks afterwards; no border and no padding are inserted at a break; no box-shadow is drawn at a broken edge; backgrounds, border-radius, and the border-image are applied to the geometry of the whole box as if it were unbroken.

### 8.14 box-sizing

```
box-sizing: content-box | border-box ;
```

specifies how the size of a box is to be calculated (table 2 on the next page):

**content-box** as specified by CSS2.1, width and height apply to the width and height respectively of the content box of the element; the padding and border of the element are laid out and drawn outside the specified width and height.

Table 2: *box-sizing*

	<code>box-sizing:</code> <code>content-box</code>	<code>box-sizing:</code> <code>border-box</code>
MIN INNER WIDTH	<code>min-width</code>	<code>max(0, min-width, padding-left, padding-right, border-left-width, border-right-width)</code>
MAX INNER WIDTH	<code>max-width</code>	<code>max(0, max-width, padding-left, padding-right, border-left-width, border-right-width)</code>
MIN INNER HEIGHT	<code>min-height</code>	<code>max(0, min-height, padding-top, padding-bottom, border-top-width, border-bottom-width)</code>
MAX INNER HEIGHT	<code>max-height</code>	<code>max(0, max-height, padding-top, padding-bottom, border-top-width, border-bottom-width)</code>

**border-box** length and percentages values for width and height on this element determine the border box of the element, that is, any padding or border specified on the element is laid out and drawn inside this specified width and height so that the content width and height are calculated by subtracting the border and padding widths of the respective sides from the specified width and height properties but with a minimum value of 0.

It is not inherited and default is `content-box`.

### 8.15 box-shadow

```
box-shadow: <color>? && [<length>{2} <length [0,∞]>? <length>?] && inset? ;
```

**<color>** specifies the colour of the shadow. If the colour is absent, it defaults to `currentColor`.

**1st <length>** Specifies the horizontal offset of the shadow. A positive value draws a shadow that is offset to the right of the box, a negative length to the left.

**2nd <length>** Specifies the vertical offset of the shadow. A positive value offsets the shadow down, a negative one up.

**3rd <length [0,∞]>** Specifies the blur radius. Negative values are invalid. If the blur value is zero, the shadow's edge is sharp. Otherwise, the larger the value, the more the shadow's edge is blurred.

**4th <length>** Specifies the spread distance. Positive values cause the shadow to expand in all directions by the specified radius. Negative values cause the shadow to contract.

**inset** changes the drop shadow from an outer box-shadow (one that shadows the box onto the canvas, as if it were lifted above the canvas) to an inner box-shadow (one that shadows the canvas onto the box, as if the box were cut out of the canvas and shifted behind it).

### 8.16 break-after

```
break-after: auto | avoid | avoid-page | page | left | right | recto | verso | avoid-column | column | avoid-region | region ;
```

specifies page/column/region break behaviour after the generated box. The forced break values `left`, `right`, `recto`, `verso`, `page`, `column` and `region` create a forced break in the flow while

the avoid break values `avoid`, `avoid-page`, `avoid-column` and `avoid-region` indicate that content should be kept together.

Since breaks are only allowed between siblings, not between a box and its container, a `break-after` value on a last-child box is propagated to its container.

Values may be:

**auto** neither force nor forbid a break after the principal box.

**avoid** avoid a break after the principal box.

### 8.16.1 Page Break Values

**avoid-page** avoid a page break after the principal box.

**page** always force a page break after the principal box.

**left** force one or two page breaks after the principal box so that the next page is formatted as a left page.

**right** force one or two page breaks after the principal box so that the next page is formatted as a right page.

**recto** force one or two page breaks after the principal box so that the next page is formatted as either a left page or a right page, whichever is second in a page spread.

**verso** force one or two page breaks after the principal box so that the next page is formatted as either a left page or a right page, whichever is first in a page spread.

### 8.16.2 Column Break Values

**avoid-column** avoid a column break after the principal box.

**column** always force a column break after the principal box.

### 8.16.3 Region Break Values

**avoid-region** avoid a region break after the principal box.

**region** always force a region break after the principal box.

It is inherited and the default is `auto`.

## 8.17 `break-before`

```
break-before: auto | avoid | avoid-page | page | left | right |  
recto | verso | avoid-column | column | avoid-region | region ;
```

specifies page/column/region break behaviour before the generated box. The forced break values `left`, `right`, `recto`, `verso`, `page`, `column` and `region` create a forced break in the flow while the avoid break values `avoid`, `avoid-page`, `avoid-column` and `avoid-region` indicate that content should be kept together.

Since breaks are only allowed between siblings, not between a box and its container, a `break-before` value on a first-child box is propagated to its container.

Values may be:

**auto** neither force nor forbid a break before the principal box.

**avoid** avoid a break before the principal box.

### 8.17.1 Page Break Values

**avoid-page** avoid a page break before the principal box.

**page** always force a page break before the principal box.

**left** force one or two page breaks before the principal box so that the next page is formatted as a left page.

**right** force one or two page breaks before the principal box so that the next page is formatted as a right page.

**recto** force one or two page breaks before the principal box so that the next page is formatted as either a left page or a right page, whichever is second in a page spread.

**verso** force one or two page breaks before the principal box so that the next page is formatted as either a left page or a right page, whichever is first in a page spread.

### 8.17.2 Column Break Values

**avoid-column** avoid a column break before the principal box.

**column** always force a column break before the principal box.

### 8.17.3 Region Break Values

**avoid-region** avoid a region break before the principal box.

**region** always force a region break before the principal box.

It is inherited and the default is **auto**.

## 8.18 break-inside

```
break-inside: auto | avoid | avoid-page | avoid-column |  
avoid-region ;
```

specifies page/column/region break behaviour within the element's principal box.

Values may be:

**auto** neither force nor forbid a break within the box.

**avoid** avoid a break within the box.

**avoid-page** avoid a page break within the box.

**avoid-column** avoid a column break within the box.

**avoid-region** avoid a region break within the box.

It is not inherited and the default is **auto**.

## 8.19 caption-side

```
caption-side: top | bottom | inherit ;
```

specifies the position of the caption in a table element. Use [text-align](#) to adjust the horizontal position of a caption.

The default value is `top`.

## 8.20 caret-color

```
caret-color: auto | <color> ;
```

specifies the colour of the caret or visible indicator of the insertion point in an element where text (and potentially other content) is inserted by the user

**auto** use `currentColor`

**<color>** colour with the specified colour (see [5.2](#) on page 11).

It is inherited and the default is `auto`.

## 8.21 clear

```
clear: left | right | both | none ;
```

specifies whether the area(s) to the left and right of the float should remain clear of content.

It is not inherited and the default value is `none`.

## 8.22 clip-path

```
clip-path: <clip-source> | [ <basic-shape> || <geometry-box> ] | none ;
```

specifies a basic shape or references a `clipPath` element to create a clipping path.

The value for `<clip-source>` is a URL.

The value for `<basic-shape>` is a basic shape function as defined in the CSS Shapes module

The default value is `border-box`.

The values for `<geometry-box>` may be:

```
<shape-box> | fill-box | stroke-box | view-box ;
```

where

**<shape-box>** in the absence of a value for `<basic-shape>`, use the edges of the specified box, including any corner shapping as the clipping path.

**fill-box** use the object bounding box.

**stroke-box** use the stroke bounding box.

**view-box** use the nearest SVG viewport.

**none** do not create a clipping path.

## 8.23 clip-rule

```
clip-rule: nonzero | evenodd ;
```

indicates the algorithm which is to be used to determine whether a given point is inside a shape for a clipping region.

The values are:

**nonzero** determines the ‘insideness’ of a point on the canvas by drawing a ray from that point to infinity in any direction and then examining the places where a segment of the shape crosses the ray. Starting with a count of zero, add one each time a path segment crosses the ray from left to right and subtract one each time a path segment crosses the ray from right to left. After counting the crossings, if the result is zero then the point is outside the path. Otherwise, it is inside.

**evenodd** determines the ‘insideness’ of a point on the canvas by drawing a ray from that point to infinity in any direction and counting the number of path segments from the given shape that the ray crosses. If this number is odd, the point is inside; if even, the point is outside.

## 8.24 color

```
color: <absolute-color-base> | currentcolor | <system-color> ;
```

specifies the foreground colour where:

```
<absolute-color-base> = <hex-color> | <absolute-color-function> |  
<named-color> | transparent
```

and

```
<absolute-color-function> = <rgb()> | <rgba()> |  
                           <hsl()> | <hsla()> | <hwb()> |  
                           <lab()> | <lch()> | <oklab()> | <oklch()> |  
                           <color()>
```

See [Colour](#) on page 11 for the available units.

## 8.25 column-fill

```
column-fill: auto | balance | balance-all ;
```

specifies whether content in a multi-column line that does not immediately precede columns in a `column-span` is balanced across columns or not. It is not inherited and the default is `balance`.

**auto** fill columns sequentially

**balance** content equally between columns, as far as possible. In fragmented contexts, only the last fragment is balanced.

**balance-all** content equally between columns, as far as possible. In fragmented contexts, all fragments are balanced.

## 8.26 column-gap

```
column-gap: normal | <length-percentage> ;
```

specifies the gutters between columns.

## 8.27 column-rule

```
column-rule: <column-rule-width> || <column-rule-style> ||  
<column-rule-color> ;
```

is a shorthand for setting `column-rule-width`, `column-rule-style`, and `column-rule-color`.

### 8.27.1 column-rule-color

```
column-rule-color: color ;
```

specifies the colour of the column rule. It is not inherited and the default is `currentcolor`.

### 8.27.2 column-rule-style

```
column-rule-style: <line-style> | none ;
```

specifies the line style of the column rule. It is not inherited and the default is `none` where `none` defines a column rule width of 0.

### 8.27.3 column-rule-width

```
column-rule-width: <line-width> ;
```

sets the width of the column rule. It is not inherited and the default is `medium`. See [border-width](#) on page 27.

## 8.28 column-span

```
column-span: none | all ;
```

describes how many columns an element spans across.

## 8.29 columns

```
columns: <column-width> || <column-count> ;
```

is a shorthand property for setting `column-width` and `column-count`.

### 8.29.1 column-count

```
column-count: auto | <length [0,∞]> ;
```

describes the number of columns of a multicol container. It is not inherited and default is `auto`.

**auto** the number of columns will be determined by other properties (e.g., `column-width`, if it has a non-`auto` value).

`<integer [1,∞]>` describes the optimal number of columns into which the content of the element will be flowed. Values must be greater than 0. If both `column-width` and `column-count` have non-auto values, the integer value describes the maximum number of columns.

### 8.29.2 column-width

```
column-width: auto | <length [0,∞]> ;
```

describes the width of columns in multicol containers. It is not inherited and default is `auto`. **auto** the column width will be determined by other properties (e.g., `column-count`, if it has a non-auto value).

`<length [0,∞]>` describes the optimal column width. The actual column width may be wider (to fill the available space), or narrower (only if the available space is smaller than the specified column width). Negative values are not allowed. Used values will be clamped to a minimum of 1px.

### 8.30 contain

```
contain: none | strict | content | [ size || layout || paint ] ;
```

allows an author to indicate that an element and its contents are, as much as possible, *independent* of the rest of the document tree.

**none** no effect

**strict** turns on all forms of containment for the element

**content** turns on all forms of containment except size containment for the element

**size** turns on size containment for the element

**paint** turns on paint containment for the element. This ensures that the descendants of the containment box don't display outside its bounds; so, if an element is off-screen or otherwise not visible, its descendants are also guaranteed to be not visible.

It is not inherited and the default is `none`.

### 8.31 content

```
content: normal | none | [ <string> | url("<url>") | <counter> |  
attr(<identifier>) | open-quote | close-quote | no-open-quote |  
no-close-quote ]+ | inherit normal ;
```

specifies the content of a `:before` or `:after` pseudo-element.

The content values may be:

**none** no content

**normal** the same as `none` for the `:before` or `:after` pseudo-elements.

`<string>` text content

`<counter>` see [Counters](#) on page 15

`attr(<identifier>)` returns the value of the attribute of the selector which matches `<identifier>`

**open-quote** introduces an opening quote

**close-quote** introduces a closing quote

**no-open-quote** increments the nesting level for quotes

**no-close-quote** decrements the nesting level for quotes

The style of quotation mark is specified by the [quotes](#) property; see page 63.

It is not inherited and the default value is **normal**.

### 8.32 counter-increment

```
counter-increment: [<identifier> <integer>?]+ | none |
inherit ;
```

specifies the counter (section [5.3 on page 15](#)) to be increment and, optionally, the amount by which it is to be incremented.

It is not inherited and the default value is **none**.

### 8.33 counter-reset

```
counter-reset: [<identifier> <integer>?]+ | none | inherit ;
```

specifies the counter (section [5.3 on page 15](#)) to be set and optionally, the amount by which it is to be incremented.

It is not inherited and the default value is **none**.

### 8.34 cue

```
cue: <cue-before> | <cue-after> ? ;
```

offers a shorthand for setting [cue-before](#) and [cue-after](#). If one value is given, it applies to both properties.

#### 8.34.1 cue-after

```
cue-after: <uri> <decibel>? | none ;
```

specifies whether and how a cue is to be given after an element.

The values are:

**<uri>** designates an auditory icon resource. Use an alternative cue, such as a bell sound, if it is not available.

**none** no auditory icon is used.

**<decibel>** is a positive or negative number followed by **dB** indicating a change from the computed value of the [voice-volume](#) property within the selected element (as a result, the volume level of an audio cue changes when the [voice-volume](#) property changes).

When omitted, or where the value of [voice-volume](#) is **silent**, the value is 0dB. Note that  $-6.0\text{dB}$  to  $+6.0\text{dB}$  normally covers the entire audio range.

It is not inherited and the default in **none**.

### 8.34.2 cue-before

```
cue-before: <uri> <decibel>? | none ;
```

specifies whether and how a cue is to be given before an element.

The values are the same as for **cue-after**.

It is not inherited and the default is **none**.

### 8.35 cursor

```
cursor: [ [<url> [<x> <y>]?,]* [ auto | default | none |
context-menu | help | pointer | progress | wait | cell |
crosshair | text | vertical-text | alias | copy | move |
no-drop | not-allowed | grab | grabbing | e-resize |
n-resize | ne-resize | nw-resize | s-resize | se-resize |
sw-resize | w-resize | ew-resize | ns-resize | nesw-resize |
nwse-resize | col-resize | row-resize | all-scroll | zoom-in |
zoom-out ] ] ;
```

specifies the appearance of any pointer.

The cursor values are

**url("<url>")** if there is a list of URLs, the first readable image is selected

**<x> <y>** the x-coordinate and y-coordinate of the position in the cursor's coordinate system (left/top relative) which represents the precise position that is being pointed to

**auto** determined by the user agent

**default** platform dependant, normally an arrow

**none** no cursor is rendered

**context-menu** a context menu is available for the object under the cursor

**help** help is available for the object under the cursor

**pointer** the cursor is a pointer that indicates a link.

**progress** a progress indicator

**wait** indicates that the program is busy and the user should wait; often rendered as a watch or hourglass

**cell** indicates that a cell or set of cells may be selected.

**crosshair** a simple crosshair

**text** indicates text that may be selected.

**vertical-text** indicates vertical-text that may be selected

**alias** a shortcut to something to be created

**copy** indicates something to be copied

**move** indicates something to be moved

**no-drop** indicates that the dragged item cannot be dropped at the current cursor location

**not-allowed** indicates that the requested action will not be carried out

**grab** indicates that something can be grabbed

**grabbing** indicates that something is being grabbed

**e-resize, n-resize, ne-resize, nw-resize, s-resize, se-resize, sw-resize, w-resize** indicates that some edge is to be moved from a cardinal or intercardinal direction on the box.

**ew-resize, ns-resize, nesw-resize, nwse-resize** indicates a bidirectional resize cursor

**col-resize** indicates that the item/column can be resized horizontally

**row-resize** indicates that the item/row can be resized vertically

**all-scroll** indicates that the something can be scrolled in any direction

**zoom-in, zoom-out** indicates that something can be zoomed in or out.

It is inherited and the default value is `auto`.

### 8.36 direction

```
direction: ltr | rtl | inherit ;
```

specifies the direction of text.

It is inherited and the default value is `ltr`.

### 8.37 display

```
display: flex | inline-flex | grid | inline-grid ;
```

specifies the type of container within which to display content.

The values are:

**flex** generate a block-level flex container box.

**inline-flex** generate an inline-level flex container box.

**grid** generate a block-level grid container box.

**inline-grid** generate an inline-level grid container box.

It is not inherited.

### 8.38 elevation

```
elevation: <angle> | below | level | above | higher | lower |  
inherit ;
```

specifies the elevation properties of an aural<sup>5</sup> element.

The elevation values are:

**<angle>** refers to the position of the sound source with 0deg being level, 90deg directly above and -90deg directly below.

**below** -90deg

**level** 0deg

**above** 90deg

**higher** increases the angle by 10

**lower** decreases the angle by 10

The default value is `level`.

---

<sup>5</sup>The aural category will be replaced by the audio and speech categories in due course.

## 8.39 empty-cells

See [border-collapse](#) on page 28.

## 8.40 flex

```
flex : none | [ <flex-grow> <flex-shrink>? || <flex-basis> ] ;
```

offers a shorthand for [flex-flow](#) or [flex-shrink](#) and [flex-basis](#).

**auto** specifies the values 1 1 auto.

**none** specifies the values 0 0 auto.

It is not inherited and the default values are 0 1 auto.

### 8.40.1 flex-basis

```
flex-basis: content | <width> ;
```

sets the flex basis for a [flex-item](#).

The values are:

**auto**

**content** based on the [flex-item](#)'s content

**<width>** any value permitted for the [width](#) property.

**Note:**

setting these values as part of [flex](#) is recommended as [flex](#) resets unspecified components.

It is not inherited and the default value is **auto**.

### 8.40.2 flex-grow

```
flex-grow: <number> ;
```

specifies the proportion by which a [flex-item](#) will grow.

**Note:**

setting this value as part of [flex](#) is recommended as [flex](#) resets unspecified components.

It is not inherited and the default value is 0

### 8.40.3 flex-shrink

```
flex-shrink: <number> ;
```

specifies the proportion by which a [flex-item](#) will shrink.

**Note:**

setting this value as part of [flex](#) is recommended as [flex](#) resets unspecified components.

It is not inherited and the default value is 1

## 8.41 flex-flow

```
flex-flow: flex-direction || flex-wrap ;
```

offers a shorthand for [flex-direction](#) and [flex-wrap](#).

### 8.41.1 flex-direction

```
flex-direction: row | row-reverse | column | column-reverse ;
```

specifies how `flex-items` are placed in the flex container, by setting the direction of the flex container's main axis.

The values are:

**row** the flex container's main axis has the same orientation as the inline axis of the current writing mode. The `main-start` and `main-end` directions are equivalent to the `inline-start` and `inline-end` directions.

**row-reverse** same as `row`, except the `main-start` and `main-end` directions are swapped.

**column** the flex container's main axis has the same orientation as the block axis of the current writing mode. The `main-start` and `main-end` directions are equivalent to the `block-start` and `block-end` directions.

**column-reverse** same as `column`, except the `main-start` and `main-end` directions are swapped.

It is not inherited and the default value is `row`.

### 8.41.2 flex-wrap

```
flex-wrap: nowrap | wrap | wrap-reverse ;
```

controls whether the flex container is single-line or multi-line, and the direction of the cross-axis, which determines the direction new lines are stacked in.

The values are:

**nowrap** the flex container is single-line.

**wrap** the flex container is multi-line.

**wrap-reverse** same as `wrap`.

It is not inherited and the default value is `nowrap`.

### 8.42 float

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

specifies the position of a float.

It is not inherited and the default value is `none`.

See also [clear](#) which specifies whether the area(s) to the left and right of the float should remain clear of content.

### 8.43 font

```
[ [ <'font-style'> || <font-variant-css21> || <'font-weight'> ||  
<'font-stretch'> ]? <'font-size'> [ / <'line-height'> ]?  
<'font-family'> ] | caption | icon | menu | message-box |  
small-caption | status-bar | inherit ;
```

offers a shorthand for specifying font properties. For [line-height](#), see page 53,

The font property values are:

**caption** the font used for captioned controls  
**icon** the font used for icons  
**menu** the font used for menus  
**message-box** the font used in dialogue boxes  
**small-caption** the font used for labelling small controls  
**status-bar** the font used in window status bars

### 8.43.1 font-family

```
font-family: [ <family-name> | <generic-family> ] #  
| inherit ;
```

specifies a comma separated list of font families to be used.

The font family values are

**family-name** for example "Times Roman", "Helvetica"

**generic-family** serif | sans-serif | cursive | fantasy | monospace

It is inherited and the default value depends on the user agent.

See also [Fonts](#) on page 15.

### 8.43.2 font-kerning

```
font-kerning auto | normal | none ;
```

**auto** kerning is applied at the discretion of the user agent

**normal** kerning is applied

**none** kerning is not applied.

It is inherited and default is auto.

### 8.43.3 font-size

```
font-size: <absolute-size> | <relative-size> |  
<length-percentage> inherit ;
```

specifies the size of the font to be used.

The font-size values are:

**absolute-size** [ xx-small | x-small | small | medium | large | x-large |  
xx-large ] ; n.b. x-small is not mapped to the HTML sizes 1–6 (figure 7).

**relative-size** [larger | smaller]; that is, increase/decrease the HTML size

**<length-percentage>** a length value in **ems** specifies an absolute font size; a percentage value specifies a font size relative to the parent element's font size.

It is inherited and the default value is **medium**.

ABSOLUTE-SIZE VALUES	SCALING FACTOR	HTML HEAD- INGS	HTML FONT SIZES
xx-small	3/5	h6	1
x-small	3/4		
small	8/9	h5	2
medium	1	h4	3
large	6/5	h3	4
x-large	3/2	h2	5
xx-large	2	h1	6
	3		7

Figure 7: Font sizes

#### 8.43.4 font-size-adjust

`font-size-adjust: none | <number> ;`

specifies the aspect value of any fallback font, so that it is the same height as a font specified by a family-name.

**<number>** specifies the aspect value used in the calculation below to calculate the adjusted font size:  $c = (a/a')s$  where:

$s$  = font-size value

$a$  = aspect value as specified **<number>**

$a'$  = aspect value of actual font

$c$  = adjusted font-size to use

**Note:** this does not affect the `line-height` property which is computed on `font-size`.

#### 8.43.5 font-stretch

`font-stretch: normal | ultra-condensed | extra-condensed | condensed | semi-condensed | semi-expanded | expanded | extra-expanded | ultra-expanded | inherit ;`

specifies whether a normal, condensed, or expanded face from a font family is to be used. The default value is `normal`.

**Note:** provide two entries, one without `font-stretch` and one with in the CSS file, so that older browsers will use the first entry and ignore the second while newer browsers will let the second entry supersede the first.

#### 8.43.6 font-style

`font-style: normal | italic | oblique | inherit ;`

specifies which font style from a particular typeface should be used.

The font style values are:

**normal** upright

**italic** italic or cursive

**oblique** slanted, strictly speaking a slanted upright font but often regarded as the same as *italic*

It is inherited and the default value is **normal**.

#### 8.43.7 font-synthesis

```
font-synthesis: none | [ weight || style ] ;
```

specifies whether browsers are allowed to synthesise bold or italic styles when a font lacks them.

#### 8.43.8 font-weight

```
font-weight: normal | bold | bolder | lighter | 100 | 200 |  
300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit ;
```

specifies the font weight to be used (see [Fonts](#) on page 15).

It is inherited and the default value is **normal**.

#### 8.43.9 font-feature-settings

```
font-feature-settings: normal | <string> [ <integer> | on | off ] ;
```

provides low-level control over OpenType font features where <string> is an optionally comma separated list of case-sensitive [OpenType feature tags](#).

#### 8.44 font-variant

```
font-variant: normal | none | [ <common-lig-values> ||  
<discretionary-lig-values> || <historical-lig-values> ||  
<contextual-alt-values> || [ small-caps | all-small-caps |  
petite-caps | all-petite-caps | unicase | titling-caps ] ||  
<numeric-figure-values> || <numeric-spacing-values> ||  
<numeric-fraction-values> || ordinal || slashed-zero ||  
<east-asian-variant-values> || <east-asian-width-values> ||  
ruby || [ sub | super ] ] ;
```

provides a shorthand for all font-variant subproperties. It is inherited and the default is **normal**.

##### 8.44.1 font-variant-caps

```
font-variant-caps: normal | small-caps | all-small-caps |  
petite-caps | all-petite-caps | unicase | titling-caps ;
```

specifies the use of alternate glyphs for capitalisation when available in a typeface:

**normal** not enabled.

**small-caps** display small capitals  
**all-small-caps** display small capitals for both upper and lowercase letters  
**petite-caps** display petite capitals  
**all-petite-caps** display petite capitals for both upper and lowercase letters  
**unicase** display small capitals for uppercase letters with normal lowercase letters  
**titling-caps** display titling capitals, where all characters are in uppercase.  
 It is inherited and the default is **normal**.

#### 8.44.2 font-variant-east-asian

```
font-variant-east-asian: normal |
[ <east-asian-variant-values> || <east-asian-width-values> || ruby ] ;
```

specifies glyph substitution and sizing in East Asian text. It is inherited and the default is **normal**.

**normal** not enabled  
**jis78** render JIS78 forms  
**jis83** render JIS83 forms  
**jis90** render JIS90 forms  
**jis04** render JIS2004 forms  
**simplified** render simplified forms  
**traditional** render traditional forms  
**full-width** render full-width variants  
**proportional-width** render proportionally-spaced variants  
**ruby** display ruby variant glyphs.

#### 8.44.3 font-variant-ligatures

```
font-variant-ligatures: normal | none | [ <common-lig-values> ||
<discretionary-lig-values> || <historical-lig-values> ||
<contextual-alt-values> ] ;
```

specifies the use of ligatures when available in a typeface:

**normal** common default features are enabled; for OpenType fonts, common ligatures and contextual forms are on by default, discretionary and historical ligatures are not.  
**none** ligatures are disabled; may improve the speed of text rendering.  
**common-ligatures** display common ligatures; n.b. for OpenType fonts, common ligatures are enabled by default.  
**no-common-ligatures** disable display of common ligatures.  
**discretionary-ligatures** display discretionary ligatures; which ligatures are discretionary or optional is decided by the type designer.  
**no-discretionary-ligatures** disable display of discretionary ligatures.  
**historical-ligatures** display historical ligatures.

**no-historical-ligatures** disable display of historical ligatures.

**contextual** display contextual alternates; n.b. for OpenType fonts, this feature is on by default.

**no-contextual** disable display of contextual alternates.

**Note:** required ligatures, needed for correctly rendering complex scripts, are not affected by the settings above, including ‘none’.

#### 8.44.4 font-variant-numeric

```
font-variant-numeric: normal | [ <numeric-figure-values> ||  
<numeric-spacing-values> || <numeric-fraction-values> || ordinal ||  
slashed-zero ] ;
```

specifies the use of numerals when available in a typeface:

**normal** not enabled

**lining-nums** display lining numerals

**oldstyle-nums** display old-style numerals

**proportional-nums** display proportional numerals

**tabular-nums** display tabular numerals

**diagonal-fractions** display lining diagonal fractions

**stacked-fractions** display lining stacked fractions

**ordinal** display letter forms used with ordinal numbers

**slashed-zero** display slashed zeros.

It is inherited and the default is **normal**.

#### 8.44.5 font-variant-position

```
font-variant-position: normal | sub | super ;
```

specifies the use of subscript and superscript variants when available in a typeface:

**normal** not enabled

**sub** display subscript variants

**super** display superscript variants

It is inherited and the default is **normal**.

#### 8.45 gap

```
gap: <row-gap> <column-gap>? ;
```

offers a shorthand for [row-gap](#) and [column-gap](#).

## 8.46 grid

```
grid: <grid-template> | <grid-template-rows> /  
[ auto-flow && dense? ] <grid-auto-columns>? |  
[ auto-flow && dense? ] <grid-auto-rows>? / <grid-template-columns> ;
```

offers a shorthand for setting the explicit grid properties ([grid-template-rows](#), [grid-template-columns](#), and [grid-template-areas](#)) and the implicit grid properties ([grid-auto-rows](#), [grid-auto-columns](#), and [grid-auto-flow](#)) in a single declaration.

### 8.46.1 grid-template

```
grid-template: none | [ <grid-template-rows> /  
<grid-template-columns> ] | [ <line-names>? <string> <track-size>?  
<line-names>? ]+ [ / <explicit-track-list> ]? ;
```

offers a shorthand for setting [grid-template-columns](#), [grid-template-rows](#), and [grid-template-areas](#) in a single declaration.

**none** sets all three properties to their initial values (**none**).

### 8.46.2 grid-template-areas

```
grid-template-areas: none | <string>+ ;
```

specifies named grid areas, which are not associated with any particular grid item, but can be referenced from the grid-placement properties.

The values are:

**none** no named grid areas, and likewise no explicit grid tracks, are defined.

**<string>+** a row is created for every separate string listed.

The default value is **none**.

### 8.46.3 grid-template-columns

```
grid-template-columns: none | <track-list> | <auto-track-list> |  
subgrid <line-name-list>? ;
```

specifies, as a space-separated track list for the grid's columns, the line names and track sizing functions of the grid.

The values are:

**none** no explicit grid tracks are created; any rows/columns will be implicitly generated, and their size will be determined by the [grid-auto-rows](#) and [grid-auto-columns](#) properties.

**<track-list> | <auto-track-list>** specifies the track list as a series of track sizing functions and line names. Each track sizing function can be specified as a length, a percentage of the grid container's size, a measurement of the contents occupying the column or row, a range using the `minmax()` notation or a fraction of the free space in the grid.

**subgrid <line-name-list>?** The subgrid value indicates that the grid will adopt the spanned portion of its parent grid in that axis (the subgridded axis). Rather than being specified explicitly, the sizes of the grid rows/columns will be taken from the parent grid's definition.

It is not inherited and the default is `none`.

#### 8.46.4 `grid-auto-columns`

```
grid-auto-columns: <track-size>+ ;
```

specifies the size of implicitly-created tracks where `grid-template-columns` does not create them explicitly.

#### 8.46.5 `grid-template-rows`

```
grid-template-rows: none | <track-list> | <auto-track-list> |  
subgrid <line-name-list>? ;
```

specifies, as a space-separated track list for the grid's rows, the line names and track sizing functions of the grid.

The values are the same as `grid-template-columns`.

#### 8.46.6 `grid-auto-rows`

```
grid-auto-rows: <track-size>+ ;
```

specifies the size of implicitly-created tracks where `grid-template-rows` does not create them explicitly.

#### 8.46.7 `grid-auto-flow`

```
grid-auto-flow: [ row | column ] || dense ;
```

specifying exactly how auto-placed items get flowed into the grid.

The values are:

**row** fill each row in turn, adding new rows as necessary.

**column** fill each column in turn, adding new columns as necessary.

**dense** use the dense packing algorithm, which fills in holes earlier in the grid if smaller items come up later. If omitted, a sparse algorithm is used, where placement is forward, never backtracking to fill holes.

It is not inherited and the default value is `row`.

#### 8.47 `grid-area`

```
grid-area: <grid-line> [ / <grid-line> ]{0,3} ;
```

is a shorthand for `grid-row-start`, `grid-column-start`, `grid-row-end` and `grid-column-end`.

If four `<grid-line>` values are specified, `grid-row-start` is set to the first value, `grid-column-start` is set to the second value, `grid-row-end` is set to the third value, and `grid-column-end` is set to the fourth value.

When `grid-column-end` is omitted, if `grid-column-start` is a `<custom-ident>`, `grid-column-end` is set to that `<custom-ident>`; otherwise, it is set to `auto`.

When `grid-row-end` is omitted, if `grid-row-start` is a `<custom-ident>`, `grid-row-end` is set to that `<custom-ident>`; otherwise, it is set to `auto`.

When `grid-column-start` is omitted, if `grid-row-start` is a `<custom-ident>`, all four longhands are set to that value. Otherwise, it is set to `auto`.

## 8.48 `grid-column`

```
grid-column: <grid-line> [ / <grid-line> ]? ;
```

offers a shorthand for [grid-column-start/grid-column-end](#).

### 8.48.1 `grid-column-end`

```
grid-column-end: auto | <custom-ident> |  
[ <integer> && <custom-ident>? ] | [ span && [ <integer>  
|| <custom-ident> ] ] ;
```

contributes to determining a grid item's size and location within the grid.

The values are:

#### **auto**

`<custom-ident>` attempt to match the grid area's edge to a named grid area.

`<integer> && <custom-ident>?`  contributes the  $n$ th grid line to the grid item's placement.

`span && [ <integer> || <custom-ident> ]` contributes a grid span to the grid item's placement such that the corresponding edge of the grid item's grid area is  $n$  lines from its opposite edge in the corresponding direction.

### 8.48.2 `grid-column-start`

```
grid-column-start: auto | <custom-ident> |  
[ <integer> && <custom-ident>? ] | [ span && [ <integer>  
|| <custom-ident> ] ] ;
```

contributes to determining a grid item's size and location within the grid.

The values are as for `grid-column-end`.

## 8.49 `grid-row`

```
grid-row: <grid-line> [ / <grid-line> ]? ;
```

offers a shorthand for [grid-row-start/grid-row-end](#).

### 8.49.1 `grid-row-end`

```
grid-row-end: auto | <custom-ident> |  
[ <integer> && <custom-ident>? ] | [ span && [ <integer>  
|| <custom-ident> ] ] ;
```

contributes to determining a grid item's size and location within the grid.

The values are as for `grid-column-end`.

### 8.49.2 grid-row-start

```
grid-row-start: auto | <custom-ident> |  
[ <integer> && <custom-ident>? ] | [ span && [ <integer>  
|| <custom-ident> ] ] ;
```

contributes to determining a grid item's size and location within the grid.

The values are as for `grid-column-end`.

### 8.50 height

```
height: <length> | <percentage> | auto | inherit ;
```

specifies the height of a block level containing box or media type or the minimum height of a table row.

**<percentage>** refers to the height of the containing block; for a block whose [position](#) is **absolute** this is defined by the box's padding edge (Figure 1 on page 10).

**auto** depends on context

It is not inherited and the default value is **auto**.

### 8.51 image-orientation

```
image-orientation: from-image | none ;
```

specifies an orthogonal rotation to be applied to the element's images before they are used in the document. It is inherited and default is **from-image**.

**from-image** if the image has an orientation specified in its metadata, such as EXIF, this value computes to the angle that the metadata specifies is necessary to correctly orient the image. If there is no orientation specified in its metadata, this value computes to **none**.

**none** no additional rotation is applied

### 8.52 isolation

```
isolation: auto | isolate ;
```

**auto** elements are not isolated

**isolate** turns the element into a stacking context.

It is not inherited and the default is **auto**.

### 8.53 justify-content

```
justify-content: flex-start | flex-end | center |  
space-between | space-around ;
```

aligns the contents of the box as a whole) within the box itself along the inline/row/main axis of the box where

**flex-start** aligns the `flex-item` to be flush with the edge of the `flex-container` corresponding to the `flex-container`'s main-start side.

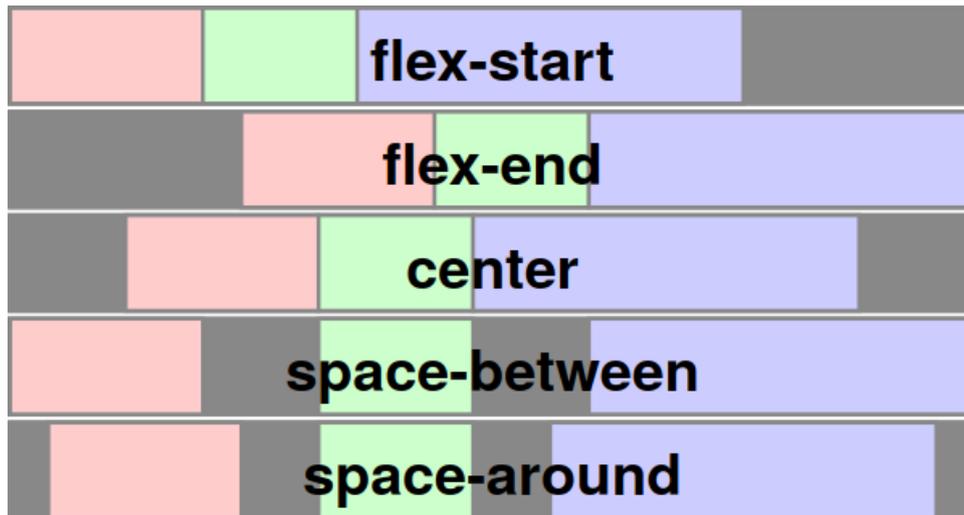


Figure 8: *justify-content*

**flex-end** aligns the **flex-item** to be flush with the edge of the **flex-container** corresponding to the **flex-container**'s main-end side. **space-between** which distributes items evenly along the horizontal axis; if there is insufficient space, this is the equivalent of **start**

**center** centres the items within its container.

**space-around** distributes items evenly along the horizontal axis with a half space at either end; if there is insufficient space, this is the equivalent of **center**

**space-around** distributes items evenly in the line, with half-size spaces on either end. If the leftover free-space is negative or there is only a single flex item on the line, this value is identical to **center**. Otherwise, the flex items on the line are distributed such that the spacing between any two adjacent flex items on the line is the same, and the spacing between the first/last flex items and the flex container edges is half the size of the spacing between flex items (see figure 8 on this page).

It is not inherited and the default value is **flex-start**.

### 8.54 justify-items

```
justify-items: normal | stretch | <baseline-position> |
[ <overflow-position>? <self-position> ] | [ legacy || [ left | right |
center ] ] ;
```

specifies the default **justify-self** for all of the child boxes in a box where

**legacy** causes the value to effectively inherit into descendants. It effectively replicates the behaviour of the obsolete HTML **<center>** element.

See **justify-self** for an explanation of the other permissible values.

### 8.55 justify-self

```
justify-self: auto | normal | stretch | <baseline-position> |
<overflow-position>? [ <self-position> | left | right ] ;
```

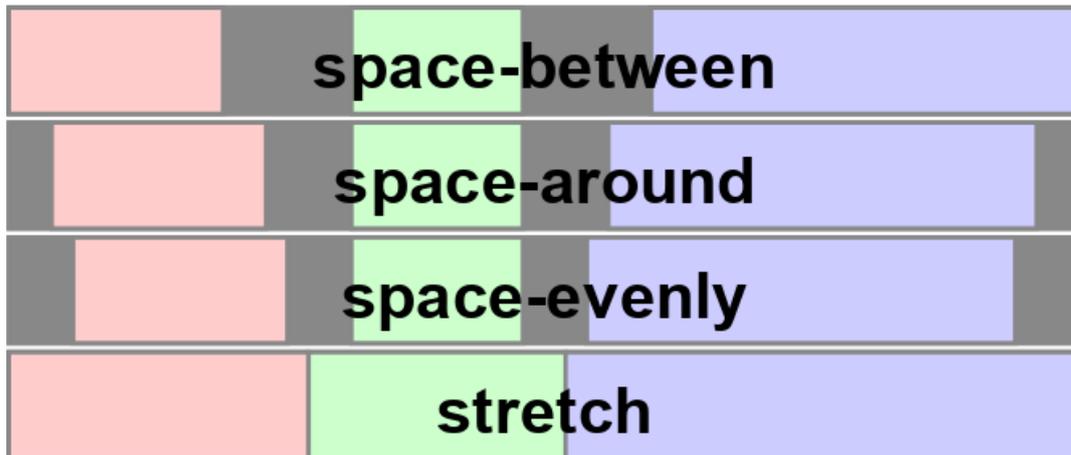


Figure 9: Distributed-alignment

justifies a box within its containing block along the horizontal axis of the container where `<baseline-position>` values may be

**baseline** which is the same as **first**

**first** which uses the first box's baseline

**last** which uses the last box's baseline

`<content-distribution>` values may be

**space-between** which distributes items evenly along the horizontal axis; if there is insufficient space, this is the equivalent of **start**

**space-around** which distributes items evenly along the horizontal axis with a half space at either end; if there is insufficient space, this is the equivalent of **center**

**space-evenly** which distributes items evenly along the horizontal axis with a full space at either end

**stretch** which gives all the items the same width within any existing max-height and max-width constraints. See table 2 on page 30 and figure 9.

`<overflow-position>?` may be

**safe** where, in the case of an overflow, defaults to **start**

**unsafe** where the existing alignment is honoured regardless of its impact

In the absence of a value, the UA works out the best way of handling the positioning.

`<self-position>` may be

**center** which centres the item within its container.

**start** which aligns the item to be flush with the container's start edge in the horizontal axis.

**end** which aligns the item to be flush with the container's end edge in the horizontal axis.

**self-start** which aligns the item to be flush with the edge of the container corresponding to the start side in the horizontal axis.

**self-end** which aligns the item to be flush with the edge of the container corresponding to the end side in the horizontal axis.

**flex-start** which aligns the `flex-item` to be flush with the edge of the `flex-container` corresponding to the `flex-container`'s main-start side.

**flex-end** which aligns the `flex-item` to be flush with the edge of the `flex-container` corresponding to the `flex-container`'s main-end side.

**left** which aligns the item to be flush with the alignment container's line-left or physical left edge, whichever is in the appropriate axis. If the property's axis is not parallel with either left↔right axis, this value behaves as `start`.

**right** which aligns the item to be flush with the alignment container's line-right or physical right edge, whichever is in the appropriate axis. If the property's axis is not parallel with either left↔right axis, this value behaves as `start`.

## 8.56 left

See [position](#) on page 63.

## 8.57 letter-spacing

```
letter-spacing: normal | <length> | inherit ;
```

specifies the inter-character spacing.

**<length>** refers to additional inter-character spacing and its value may be negative

The default value is `normal`.

## 8.58 line-height

```
line-height: normal | <number> | <length> | <percentage> |  
inherit ;
```

specifies the line height of an inline box.

**normal** is a reasonable value based on the font size.

**<number>** is a positive factor applied to the font size

**<percentage>** refers to the font size.

The default value is `normal`.

## 8.59 list-style

```
list-style: <list-style-type | symbols(system? <symbol list>) ||  
list-style-position || list-style-image ;
```

offers a shorthand for setting [list-style-type](#), [list-style-position](#) and [list-style-image](#).

### 8.59.1 list-style-image

```
list-style-image: uri | none | inherit ;
```

specifies the image to be used as the marker

The default value is `none`.

### 8.59.2 list-style-position

```
list-style-position: inside | outside | inherit ;
```

specifies whether the list marker is placed outside the left edge of the element or in an inline box as the first item of the element's content.

The default value is `outside`.

### 8.59.3 list-style-type

```
list-style-type: disc | circle | square | decimal |  
decimal-leading-zero | lower-roman | upper-roman |  
lower-greek | lower-latin | upper-latin | armenian |  
georgian | lower-alpha | upper-alpha | none | <symbols()> ;
```

specifies the style of the marker where

- the value of the property `list-style-image` is `none` or
- it is `<url>` but no image is available to that URL.

**decimal** displays decimal numbers beginning with 1

**decimal-leading-zero** displays decimal numbers beginning with 01 to 09 and then 10, 11 ...

The default value is `disc`.

```
list-style: symbols(system ? <symbol list>) ;
```

optionally specifies a system defined by an `@counter-style` rule (section 3.1).

## 8.60 margin

```
margin: <length> | <percentage> | auto | inherit ;
```

offers a shorthand for specifying the widths of all the margins of non-table elements.

The margin values are:

**<percentage>** refers to the width of the containing block (not to the height in the case of `margin-top` and `margin-bottom`)

**auto** is 0

Up to four values may be specified representing `margin-top`, `margin-right`, `margin-bottom` and `margin-left`. Values may be negative. Omitted values are co-opted as follows:

- 2nd from 1st
- 3rd from 1st
- 4th from 2nd (i.e. from 1st if no 2nd)

Additionally the properties:

```
margin-top: <length> | <percentage> | auto | inherit ;  
margin-right: <length> | <percentage> | auto | inherit ;  
margin-bottom: <length> | <percentage> | auto | inherit ;  
margin-left: <length> | <percentage> | auto | inherit ;
```

may be set separately.

Values are not inherited and the default value is 0.

Vertically adjoining margins of block content collapse to the greater of the two margin widths (or the positive minus the negative margin width) unless they are the bottom margin or the top margin of a root element or the margins of an absolutely positioned box.

Use `margin-right: auto;` and `margin-left: auto;` to centre an element with a defined width in its containing block; Note that, if the element's width is undefined, `margin-right: auto;` and `margin-left: auto;` default to 0.

## 8.61 mask

```
mask: <mask-reference> || <position> [ / <bg-size> ]? ||  
<repeat-style> || <geometry-box> || [ <geometry-box> | no-clip ] ||  
<compositing-operator> || <masking-mode> ;
```

offers a shorthand for setting the mask properties. It also resets `mask-border` to its initial setting.

### 8.61.1 mask-border

```
mask-border: <mask-border-source> || <mask-border-slice>  
[ / <mask-border-width>? [ / <mask-border-outset> ]? ]? ||  
<mask-border-repeat> || <mask-border-mode> ;
```

offers a shorthand for setting the `mask-border` properties.

### 8.61.2 mask-border-mode

```
mask-border-mode: luminance | alpha ;
```

indicates whether the `<image>` value for `mask-border-source` is treated as luminance mask or alpha mask.

**alpha** alpha values of the mask border image should be used as the mask values.

**luminance** luminance values of the mask border image should be used as the mask values.

### 8.61.3 mask-border-outset

```
mask-border-outset: [ <length> | <number> ]{1,4} ;
```

specify the amount by which the mask border image area extends beyond the border box. If it has four values, they set the outsets on the top, right, bottom and left sides in that order. If the left is missing, it is the same as the right; if the bottom is missing, it is the same as the top; if the right is missing, it is the same as the top.

### 8.61.4 mask-border-repeat

```
mask-border-repeat: [ stretch | repeat | round | space ]{1,2} ;
```

specifies how the images for the sides and the middle part of the mask border image are scaled and tiled. The first keyword applies to the horizontal sides, the second to the vertical ones. If the second keyword is absent, it is assumed to be the same as the first.

The values are as for `border-image-repeat`.

#### 8.61.5 `mask-border-slice`

```
mask-border-slice: <number-percentage>{1,4} fill? ;
```

specifies inward offsets from the top, right, bottom, and left edges of the `mask-border-image`, dividing it into nine regions: four corners, four edges and a middle. The middle image part is discarded and treated as fully opaque white (the content covered by the middle part is not masked and shines through) unless the `fill` keyword is present.

#### 8.61.6 `mask-border-source`

```
mask-border-source: none | <image> ;
```

specifies an image to be used as mask border image.

#### 8.61.7 `mask-border-width`

```
mask-border-width: [ <length-percentage> | <number> | auto ]{1,4} ;
```

specifies the border width of the mask border image area.

The values are as for `border-width`.

#### 8.61.8 `mask-clip`

```
mask-clip : [ <geometry-box> | no-clip ]# ;
```

determines the area that is affected by the mask.

`<geometry-box>` may have the value:

**content-box** the content is restricted to the content box.

**padding-box** the content is restricted to the padding box.

**border-box** the content is restricted to the border box.

**margin-box** the content is restricted to the margin box.

**fill-box** the content is restricted to the object bounding box.

**stroke-box** the content is restricted to the stroke bounding box.

**view-box** uses the nearest SVG viewport as reference box positioned at the origin of the coordinate system established by the `viewBox` attribute. The dimension of the reference box is set to the width and height values of the `viewBox` attribute.

**no-clip** the content is not restricted.

### 8.61.9 mask-composite

`mask-composite: add | subtract | intersect | exclude ;`

specifies the Porter-Duff compositing operator to be used.

The values are:

**add** the source is placed over the destination.

**subtract** the source is placed where it falls outside of the destination.

**intersect** the parts of source that overlap the destination replace the destination.

**exclude** the non-overlapping regions of source and destination are combined.

### 8.61.10 mask-image

`mask-image: none | <image> | <mask-source> ;`

sets the mask layer image of an element where

**mask-source** is a URL

**none** counts as a transparent black image layer.

### 8.61.11 mask-mode

`mask-mode: alpha | luminance | match-source ;`

indicates whether to use a luminance mask or alpha mask.

The values are:

**alpha** use the alpha values of the mask layer image as the mask values.

**luminance** use the luminance values of the mask layer image as the mask values.

**match-source** if the value of the `mask-image` property is `mask-source`, use the luminance values of the mask layer image as the mask values.

If the value of the `mask-image` property is `image`, use the alpha values of the mask layer image as the mask values.

### 8.61.12 mask-origin

`mask-origin: <geometry-box># ;`

specifies the mask positioning area for elements rendered as a single box; for elements rendered as multiple boxes specifies which boxes `box-decoration-break` operates on to determine the mask positioning area.

`<geometry-box>` may have the value:

**content-box** use the position relative to the content box.

**padding-box** use the position relative to the padding box. (For single boxes 00 is the upper left corner of the padding edge, 100% 100% is the lower right corner.)

**border-box** use the position relative to the border box.

**margin-box** use the position is relative to the margin box.

**fill-box** use the position relative to the object bounding box.

**stroke-box** use the position relative to the stroke bounding box.

**view-box** use the nearest SVG viewport as reference box positioned at the origin of the coordinate system established by the `viewBox` attribute. The dimension of the reference box is set to the width and height values of the `viewBox` attribute.

For SVG elements without an associated CSS layout box, the values `content-box`, `padding-box`, `BORDER-BOX` and `margin-box` compute to `fill-box`.

For elements with an associated CSS layout box, the values `fill-box`, `stroke-box` and `view-box` compute to the initial value of `mask-origin`.

### 8.61.13 mask-position

```
mask-position: <position> ;
```

specifies how mask layer images are positioned.

The values are as for `background-position`.

### 8.61.14 mask-repeat

```
mask-repeat: repeat-style ;
```

specifies how mask layer images are tiled after they have been sized and positioned.

The values are as for `background-repeat`

### 8.61.15 mask-size

```
mask-size: <background-size># ;
```

specifies how mask layer images are sized.

The values are as for `background-size`.

### 8.61.16 mask-type

```
mask-type: luminance | alpha ;
```

defines whether the content of the mask element is treated as as luminance mask or alpha mask.

The values are:

**alpha** alpha values of the mask should be used.

**luminance** luminance values of the mask should be used.

## 8.62 mix-blend-mode

```
mix-blend-mode: normal | multiply | screen | overlay | darken |  
lighten | color-dodge |color-burn | hard-light | soft-light |  
difference | exclusion | hue | saturation | color | luminosity ;
```

defines the formula that must be used to mix the colours with the backdrop.

It is not inherited and the default is `normal`.

### 8.63 object-fit

`object-fit: fill | contain | cover | none | scale-down ;`

specifies how the contents of a replaced element should be fitted to the box established by its used height and width. It is not inherited and the default value is `fill`.

**fill** the replaced content is sized to fill the element's content box: the object's concrete object size is the element's used width and height.

**contain** the replaced content is sized to maintain its natural aspect ratio while fitting within the element's content box: its concrete object size is resolved as a contain constraint against the element's used width and height.

**cover** the replaced content is sized to maintain its natural aspect ratio while filling the element's entire content box: its concrete object size is resolved as a cover constraint against the element's used width and height.

**none** the replaced content is not resized to fit inside the element's content box: determine the object's concrete object size using the default sizing algorithm with no specified size, and a default object size equal to the replaced element's used width and height.

**scale-down** size the content as if **none** or **contain** were specified, whichever would result in a smaller concrete object size.

### 8.64 object-position

This is computed as for `background-position`; see [background](#) on page 24. It is not inherited and the default value is `50% 50%`.

### 8.65 opacity

`opacity: <alphavalue> | inherit ;`

specifies the opacity of a colour of an element other than of an SVG element where:

`<alphavalue> = <number> | <percentage>`

where `<number>` is in the range 0.0 (transparent) to 1.0 (opaque) and `<percentage>` is in the range 0.0 to 100.

The default value is 1.0. (number) and 100 (percentage).

An element whose opacity is less than 1.0. (number) and 100 (percentage) is rendered in a new stacking context at `z-index:0`;

### 8.66 order

`order: <integer>`

specifies the order in which a flex item is displayed where this is not the order in which it appears in HTML. `-1` puts the item before any others.

It is not inherited and the default value is 0.

## 8.67 orphans

`orphans: <integer> | inherit ;`

specifies the minimum number of lines to be left at the bottom of a page.

The default value is 2.

## 8.68 outline

`outline: [ <outline-color> || <outline-style> || <outline-width> ] ;`

offers a shorthand for setting the outline properties of an element. Outlines differ from borders in that they can be drawn round any element, they are the same all round the element and they take up no space.

### 8.68.1 outline-color

`outline-color: <color> | invert ;`

specifies the `outline-color` if an `outline-style` is defined (see [Colour](#) on page 11).

It is not inherited and the default value is `invert`.

### 8.68.2 outline-style

`outline-style: auto | <border-style> ;`

specifies the `outline-style` of an element (see [border](#) on page 26).

It is not inherited and *there is no default value; so an outline style must be set.*

### 8.68.3 outline-width

`outline-width: <line-width> ;`

specifies the `outline-width` if an `outline-style` is defined (see [border](#) on page 26)..

It is not inherited and the default value is `medium`.

### 8.68.4 outline-offset

`outline-offset: <length> ;`

specifies the amount of any offset if the outline is not drawn just outside the border edge. It is not inherited and the default is 0.

## 8.69 overflow

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

specifies how any overflow from a block should be handled.

The overflow values are:

**visible** no clipping

**hidden** the content is clipped and no scrolling interface is provided

**scroll** the content is clipped and a scrolling mechanism is present.

**auto** a scrolling mechanism is provided if necessary.

It is not inherited and the default value is **visible**.

## 8.70 padding

```
padding: <length> | <percentage> | inherit ;
```

offers a shorthand for setting the values for the padding width.

<**percentage**> refers to the the width of the containing block (not to the height in the case of **padding-top** and **padding-bottom**)

Up to four values may be specified representing **padding-top**, **padding-right**, **padding-bottom** and **padding-left**. Values must be 0 or positive. Omitted values are co-opted as follows:

- 2nd from 1st
- 3rd from 1st
- 4th from 2nd (i.e. from 1st if no 2nd)

Alternatively, the width of each padding may be specified separately with:

```
padding-top: <length> | <percentage> | inherit ;  
padding-right: <length> | <percentage> | inherit ;  
padding-bottom: <length> | <percentage> | inherit ;  
padding-left: <length> | <percentage> | inherit ;
```

Values are not inherited and The default value is 0.

The colour of padding is specified by the **background** property.

## 8.71 page-break-after

An alias for [break-after](#). the value **always** defaults to **page**.

It is not inherited and the default value is **auto**.

## 8.72 page-break-before

An alias for [break-before](#). the value **always** defaults to **page**.

It is not inherited and the default value is **auto**.

## 8.73 page-break-inside

An alias for [break-inside](#).

It is not inherited and the default value is **auto**.

## 8.74 pause

```
pause: <pause-before> <pause-after>? ;
```

offers a shorthand for [pause-before](#) and [pause-after](#).

### 8.74.1 pause-after

```
pause-after: <time> | none | x-weak | weak | medium |
strong | x-strong ;
```

specifies the length of a pause after an element.

It is not inherited and the default is `none`.

The values are:

**<time>** non-negative absolute time units (seconds and milliseconds, e.g. "+3s", "250ms").

**none** 0ms.

**x-weak** | **weak** | **medium** | **strong** | **x-strong** monotonically non-decreasing implementation-dependent values indicating the break strength between elements.

It is not inherited and the default is `none`.

### 8.74.2 pause-before

```
pause-before: <time> | none | x-weak | weak | medium |
strong | x-strong ;
```

specifies the length of a pause before an element.

The values are the same as for `pause-after`.

It is not inherited and the default is `none`.

### 8.75 place-content

```
place-content: <align-content> <justify-content>? ;
```

offers a shorthand for [align-content](#) and [justify-content](#).

### 8.76 place-items

```
place-items: [ normal | stretch | <baseline-position> |
<self-position> ] [ normal | stretch | <baseline-position> |
<self-position> ]? ;
```

offers a shorthand for [align-items](#) and [justify-items](#).

### 8.77 place-self

```
place-self: <align-self> <justify-self>? ;
```

offers a shorthand for [align-self](#) and [justify-self](#).

### 8.78 play-during

```
play-during: url("<url>") [ mix || repeat ]? | auto | none |
inherit ;
```

specifies whether and how a background sound is to be played during an aural<sup>6</sup> element.

The play-during values are:

**url("<url>")** the link to an audio file to be played during the spoken content

**mix** play the sound from the parent element's **play-during** property along with the current background sound

**repeat** repeat the audio file if it is too short to fill the time to the end of the spoken content

**auto** play the parent element's **play-during** background sound (n.b. **inherit** restarts rather than continues the parent element's sound)

**none** play no sound during the spoken content (unless suppressed the parent element's sound will continue behind the next spoken element)

It is not inherited and the default value is **auto**.

## 8.79 position

```
position: static | relative | absolute | fixed | inherit ;
```

The position values are:

**static** the normal position within normal flow

**relative** the position within normal flow offset by some factor; n.b. this does not affect the position of subsequent elements in normal flow

**absolute** the position specified is outside normal flow and does not affect the positioning of subsequent elements in normal flow

**fixed** the position specified is outside the normal flow and fixed by reference to some point; it does not move when scrolled.

it is not inherited and the default value is **static**.

```
top: <length> | <percentage> | auto | inherit ;
right: <length> | <percentage> | auto | inherit ;
bottom: <length> | <percentage> | auto | inherit ;
left: <length> | <percentage> | auto | inherit ;
```

specify the absolute position of a box.

**<percentage>** refers to the height or width of the containing block as appropriate

**auto** depends on context.

Values are not inherited and the default value is **auto**.

## 8.80 quotes

```
quotes: [<string> <string>]+ | none | inherit ;
```

specifies the opening and closing quotation marks to be used.

The first pair of **<string>** values specify the outermost quotation mark style, subsequent pairs the style for each subsequent level of nesting.

The default value depends on the user agent.

---

<sup>6</sup>The aural category will be replaced by the audio and speech categories in due course.

## 8.81 **resize**

```
resize: none | both | horizontal | vertical ;
```

specifies whether the user can resize an element independently of any scrolling or zooming:

**none** the user cannot resize the element

**both** the user can adjust both the height and the width of the element

**horizontal** the user can only adjust the width of the element

**vertical** the user can only adjust the height of the element.

It is not inherited and the default is **none**.

## 8.82 **rest**

```
rest: <rest-before> <rest-after>? ;
```

offers a shorthand for [rest-before](#) and [rest-after](#).

### 8.82.1 **rest-after**

```
rest-after: <time> | none | x-weak | weak | medium |  
strong | x-strong ;
```

specifies a silence with a specific duration that occurs after the speech synthesis rendition of an element.

The values are:

**<time>** non-negative absolute time units (seconds and milliseconds, e.g. "+3s", "250ms")

**none** 0ms.

**x-weak** | **weak** | **medium** | **strong** | **x-strong** monotonically non-decreasing prosodic breaks in speech output between elements. The exact time is implementation-dependent.

It is not inherited and the default is **none**.

As opposed to the [pause](#) properties, the rest is inserted between the element's content and any [cue-before](#) or [cue-after](#) content. Adjoining rests are treated additively.

### 8.82.2 **rest-before**

```
rest-before: <time> | none | x-weak | weak | medium |  
strong | x-strong ;
```

specifies a silence with a specific duration that occurs before the speech synthesis rendition of an element.

The values are as for [rest-after](#).

It is not inherited and the default is **none**.

## 8.83 **right**

See [position](#).

## 8.84 row-gap

```
row-gap: normal | <length-percentage> ;
```

specifies the gutters between rows.

## 8.85 speak

```
speak: auto | never | always ;
```

specifies determines whether or not to render text audibly.

The values are:

**auto** resolves to **never** when **display** is **none**; otherwise it is equivalent to **always** if **visibility** is **visible** and to **never** otherwise.

**never** causes an element not be rendered.

**always** the element is rendered audibly.

It is inherited and the default value is **auto**.

## 8.86 speak-as

```
speak-as: normal | spell-out || digits || [ literal-punctuation |  
no-punctuation ] ;
```

determines in what manner text gets rendered audibly.

The values are:

**normal** use the language-dependent pronunciation rules for rendering the element's content.

**spell-out** spell the text one letter at a time (useful for acronyms and abbreviations).

**digits** speak numbers one digit at a time, for instance, "twelve" would be spoken as "one two", and "31" as "three one".

**literal-punctuation** name punctuation such as semicolons, braces, and so on aloud rather than rendering it naturally as appropriate pauses.

**no-punctuation** do not render punctuation whether spoken or rendered as pauses.

It is inherited and the default in **normal**.

## 8.87 table-layout

```
table-layout: auto | fixed | inherit ;
```

specifies whether a table layout is defined by the fixed widths, if any, of the table, its border, columns and cells or by an algorithm.

The default value is **auto**.

## 8.88 text-align

```
text-align: left | right | center | justify | inherit ;
```

specifies the alignment of a block of text.

`justify` should be avoided because there are no standard hyphenation rules and user agents use different algorithms to represent it.

The default value is `left` in `ltr` and `right` in `rtl` blocks

### 8.89 `text-combine-upright`

```
text-combine-upright: none | all ;
```

in vertical writing modes specifies the combination of multiple typographic character units into the space of a single typographic character unit. If the combined text is wider than `1em`, the UA must fit the contents within `1em`. The resulting composition is treated as a single upright glyph for the purposes of layout and decoration.

It is inherited and the default is `none`.

**none** no special processing.

**all** attempt to typeset horizontally all consecutive typographic character units within the box such that they take up the space of a single typographic character unit within the vertical line box.

### 8.90 `text-decoration`

```
text-decoration: none | [ underline || overline ||  
line-through || blink ] | inherit ;
```

specifies whether the text is to be decorated.

**underline** draw a line with the value of the `color` property under the line

**overline** draw a line with the value of the `color` property over the line

**line-through** draw a line with the value of the `color` property at the x-height of the line

**blink** alternative between `visibility: visible` and `visibility: hidden`

Only the value of the `color` property of a decoration is inherited by an element's descendants.

The default value is `none`.

### 8.91 `text-indent`

```
text-indent: <length> | <percentage> | inherit ;
```

specifies the indentation of the first line of a block.

`<percentage>` refers to the width of the containing block

A negative value creates a hanging indent which should be supported by specifying sufficient `padding`.

It is inherited if it is an inline block, The default value is 0.

### 8.92 `text-orientation`

```
text-orientation: mixed | upright | sideways ;
```

specifies the orientation of text within a line in vertical writing mode. It is inherited and the default is `mixed`.

**mixed** typographic character units from horizontal-only scripts are typeset sideways, i.e. 90° clockwise from their standard orientation in horizontal text. Typographic character units from vertical scripts are typeset with their intrinsic orientation. This value is typical for the layout of dominantly vertical-script text.

**upright** typographic character units from horizontal-only scripts are typeset upright, i.e. in their standard horizontal orientation. Typographic character units from vertical scripts are typeset with their intrinsic orientation and shaped normally.

**sideways** causes all text to be typeset sideways, as if in a horizontal layout, but rotated 90° clockwise.

### 8.93 text-overflow

```
text-overflow: clip | ellipsis ;
```

specifies rendering when inline content overflows:

**clip** clip inline content that overflows its block container element

**ellipsis** render an ellipsis character (... U+2026) to represent clipped inline content.

It is not inherited and the default is `clip`.

### 8.94 text-transform

```
text-transform: capitalize | uppercase | lowercase | none ;
```

specifies whether text is to be transformed.

The text transform values are:

**capitalize** makes the first letter of each word `uppercase`

**uppercase** makes all letters `uppercase`

**lowercase** makes all letters `lowercase`

**none** makes no effects

The default value is `none`.

### 8.95 top

See [position](#) on page 63.

### 8.96 transform

```
transform: none | <transform-list> ;
```

It is not inherited and the default is `none`.

## 8.97 transform-box

```
transform-box: content-box | border-box | fill-box | stroke-box  
| view-box ;
```

specifies the reference box for [transform](#) and [transform-origin](#) which may be

**content-box** the content box. The reference box of a table is the border box of its table wrapper box, not its table box.

**border-box** the border box. The reference box of a table is the border box of its table wrapper box, not its table box.

**fill-box** the object bounding box

**stroke-box** the stroke bounding box

**view-box** the nearest SVG viewport.

It is not inherited and the default is **view-box**.

## 8.98 transform-origin

```
transform-origin: [ left | center | right | top | bottom |  
<length-percentage> ] | [ left | center | right |  
<length-percentage> ] [ top | center | bottom |  
<length-percentage> ] <length>? | [[ center | left | right ] &&  
[ center | top | bottom ]] <length>? ;
```

If only one value is specified, the second value is assumed to be **center**. If one or two values are specified, the third value is assumed to be 0px.

If two or more values are defined and either no value is a keyword, or the only used keyword is **center**, then the first value represents the horizontal position (or offset) and the second represents the vertical position (or offset). A third value always represents the Z position (or offset) and must be of type **<length>**.

**<length-percentage>** a percentage for the horizontal offset relative to the width of the reference box or a percentage for the vertical offset relative to the height of the reference box representing an offset from the top left corner of the reference box

**<length>** a fixed length representing an offset from the top left corner of the reference box

**top** 0% for the vertical position

**right** 100% for the horizontal position

**bottom** 100% for the vertical position

**left** 0% for the horizontal position

**center** 50% (left 50%) for the horizontal position if the horizontal position is not otherwise specified, or 50% (top 50%) for the vertical position if it is.

It is not inherited and the default is 50% 50%.

## 8.99 unicode-bidi

```
unicode-bidi: normal | embed | isolate | bidi-override |  
isolate-override | plaintext ;
```

The unicode-bidi values are:

**normal** follow the Unicode `bidi` algorithm

**embed** embed an additional inline box that follows the `direction` property in an inline box

**isolate** in an inline box, isolates its contents. Neither is the content inside the box affected by the content surrounding the box, nor is the content surrounding the box affected by the content or specified directionality of the box. However, forced paragraph breaks within the box still create a corresponding break in the containing paragraph.

**bidi-override** override the algorithm where it conflicts with the `direction` property.

**isolate-override** combines the isolation behaviour of `isolate` with the directional override behaviour of `bidi-override` to surrounding content. It is equivalent to `isolate`, but within the box content is ordered as if `bidi-override` were specified.

**plaintext** behaves as `isolate` except that, for the purposes of the Unicode bidirectional algorithm, the base directionality of each of the box's bidi paragraphs (if a block container) or isolated sequences (if an inline) is determined by following the heuristic in rules P2 and P3 of the Unicode bidirectional algorithm (rather than by using the `direction` property of the box).

It is not inherited and the default value is `normal`.

## 8.100 vertical-align

```
vertical-align: baseline | sub | super | top | text-top |
middle | bottom | text-bottom | <percentage> | <length> | inherit ;
```

specifies the position of the content of inline boxes and table cells..

The vertical align values are:

**baseline** of the parent box or the first row that a cell spans

**sub** the subscript position of the parent box (not applicable to table elements); note that this value does not affect font size

**super** the superscript position of the parent box (not applicable to table elements); note that this value does not affect font size

**top** the top of the highest content area of the subtree of the parent element and all child elements

**text-top** the top of the parent element's content area (not applicable to table elements)

**middle** the x-height of the parent box or the middle of the table row

**bottom** the bottom of the lowest content area of the subtree of the parent element and all child elements

**text-bottom** the bottom of the parent element's content area (not applicable to table elements)

**<percentage>** refers to the line height; a positive value raises the content above the baseline, a negative value lowers it below the baseline (not applicable to table elements)

**<length>** a positive value raises the content this distance above the baseline, a negative value lowers it this distance below the baseline (not applicable to table elements)

It is not inherited and the default value is `baseline`.

### 8.101 visibility

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

specifies how boxes generated within an element will appear.

The visibility values are

**visible** boxes are visible

**hidden** boxes are generated but not visible, that is, they affect the normal flow; to prevent any effects on normal flow use `display: none`

**collapse** when applied to table rows, row groups columns or columns groups, has the same effect as **hide** in a spreadsheet (applies only to table elements)

The default value is **visible**.

### 8.102 voice-balance

```
voice-balance: <number> | left | center | right | leftwards |  
rightwards ;
```

controls the spatial distribution of audio output across a lateral sound stage.

The values are:

**<number>** -100 represents the left side, and 100 represents the right side. 0 represents the centre point

**left** -100

**center** 0

**right** 100

**leftwards** Moves the sound to the left, by subtracting 20 from the inherited **voice-balance** value up to -100

**rightwards** Moves the sound to the right, by adding 20 to the inherited **voice-balance** value up to 100

It is inherited and the default value is **center**.

### 8.103 voice-duration

```
voice-duration: auto | <time> ;
```

specifies how long it should take to render the selected element's content (not including audio cues, pauses and rests).

The values are:

**auto** uses the [voice-rate](#) to calculate duration.

**<time>** specifies a value in non-negative time units (seconds and milliseconds, e.g. "+3s", "250ms").

It is not inherited and the default value is **auto**.

## 8.104 voice-family

```
voice-family: [[<family-name> | <generic-voice>],]* [<family-name> |
<generic-voice>] | preserve ;
```

specifies a prioritized list of component values that are separated by commas to indicate that they are alternatives. Each component value potentially designates a speech synthesis voice instance.

The comma separated voice family values are:

<**family-name**> specific voice instances (e.g., Mike, comedian, mary, carlos2, 'valley girl'). Voice names must either be given quoted as strings, or unquoted as a sequence of one or more identifiers.

<**generic-voice**> a space separated list of values covering:

- <age> child, young and old
- <gender> male, female, or neutral
- <integer> indicating the preferred variant (e.g. "the second male child voice"). Only positive integers (i.e. excluding zero) are allowed. The value 1 refers to the first of all matching voices.

**preserve** inherit the voice-family value.

## 8.105 voice-pitch

```
voice-pitch: <frequency> && absolute | [[x-low | low | medium |
high | x-high] || [<frequency> | <semitones> | <percentage>]] ;
```

specifies the baseline pitch of the generated speech output which depends on the [voice-family](#).

The values are:

<**frequency**> specifies a value in frequency units (Hertz or kiloHertz, e.g. "100Hz", "+2kHz"). If **absolute** is not specified, a negative value represents a decrement, and a positive value represents an increment, relative to the inherited value.

**absolute** If specified, **frequency** is an absolute value which must be positive.

<**semitones**> specifies a relative change (decrement or increment) of *nst*.

<**percentage**> non-negative percentage value representing a change relative to the given keyword value or to the default value.

**x-low** | **low** | **medium** | **high** | **x-high** implementation and voice specific monotonically non-decreasing pitch levels.

It is inherited and the default value is **medium**.

## 8.106 voice-range

```
voice-range: <frequency> && absolute | [[x-low | low | medium |
high | x-high] || [<frequency> | <semitones> | <percentage>]] ;
```

specifies the variability in the 'baseline' pitch from the average pitch of the speech output.

The values are:

<**frequency**> specifies a value in frequency units (Hertz or kiloHertz, e.g. "100Hz", "+2kHz").  
If **absolute** is not specified, a negative value represents a decrement, and a positive value represents an increment, relative to the inherited value.

**absolute** If specified, **frequency** is an absolute value which must be positive.

<**semitones**> specifies a relative change (decrement or increment) of *nst*.

<**percentage**> non-negative percentage value representing a change relative to the given keyword value or to the default value.

**x-low** | **low** | **medium** | **high** | **x-high** implementation and voice specific monotonically non-decreasing pitch levels.

It is inherited and the default value is **medium**.

### 8.107 voice-rate

```
voice-rate: [normal | x-slow | slow | medium | fast |  
x-fast] || <percentage> ;
```

manipulates the rate of generated synthetic speech in terms of words per minute.

The values are

**normal** the default rate for the currently active voice.

**x-slow** | **slow** | **medium** | **fast** | **x-fast** implementation and voice-specific monotonically non-decreasing speaking rates

<**percentage**> non-negative percentage value representing a change relative to the given keyword value or to the default value.

It is inherited and the default value is **medium**.

### 8.108 voice-stress

```
voice-stress: normal | strong | moderate | none | reduced ;
```

manipulates the strength of emphasis.

The values are:

**normal** the default emphasis produced by the speech synthesizer.

**none** no emphasis.

**moderate** | **strong** monotonically increasing emphasis which is more than **normal**.

**reduced** Effectively the opposite of emphasizing a word.

It is inherited and the default value is **normal**.

### 8.109 voice-volume

```
voice-volume: silent | [[x-soft | soft | medium | loud | x-loud] ||  
<decibel>] ;
```

allows authors to control the amplitude of the audio waveform generated by the speech synthesiser, and to adjust the relative volume level of audio cues within the selected element.

The values are:

**silent** no sound is generated

**x-soft** minimum audible level

**soft** intermediate level between **x-soft** and **medium**

**medium** user's preferred volume level

**loud** intermediate level between **medium** and **x-loud**

**x-loud** maximum tolerable volume level

<**decibel**> is a positive or negative number followed by **dB** indicating a change from the value specified before it. Note that  $-6.0\text{dB}$  to  $+6.0\text{dB}$  normally covers the entire audio range. It is inherited and the default value is **medium**.

### 8.110 white-space

```
white-space: normal | pre | nowrap | pre-wrap | pre-line |
inherit ;
```

specifies whether and how white space is to be added to text.

The property values are

**normal** collapse sequences of white space; remove existing newline characters, converting linefeed characters as appropriate; break lines to fill boxes

**pre** do not collapse sequences of white space; break lines only where there is a newline.

**nowrap** collapse sequences of white space; convert linefeed characters as appropriate; suppress line breaks

**pre-wrap** do not collapse sequences of white space; break lines where there is a newline.and to fill boxes

**pre-line** collapse sequences of white space; remove existing newline characters; break lines where there is a newline.and to fill boxes

The default value is **normal**.

### 8.111 widows

```
widows: <integer> | inherit ;
```

specifies that minimum number of lines to be left at the top of a page.

The default value is 2.

### 8.112 width

```
width: <length> | <percentage> | auto | inherit ;
```

specifies the width of a block level containing box or media type or the minimum width of a table column.

<**percentage**> refers to the containing block; for a block whose **position** is **absolute** this is defined by the box's padding edge (Figure 1 on page 10).

**auto** depends on context

It is not inherited and the default value is **auto**.

### 8.113 will-change

`will-change: auto | <animateable-feature># ;`

provides a rendering hint to the user agent, stating what kinds of changes the author expects to perform on the element.

The values are:

**auto** no particular intent

**contents** animate or change something about the element's contents in the near future. Best used on elements near the bottom of the document tree, containing as little of the document as possible.

**<custom-ident>** animate or change the property with the given name on the element in the near future.

The default value is `auto`.

### 8.114 word-spacing

`word-spacing: normal | <length> | inherit ;`

specifies the inter-word spacing. Note that `text-align: justify` normally adds inter-word space.

**<length>** refers to the additional inter-word spacing and may be negative.

The default value is `normal`.

### 8.115 writing-mode

`writing-mode: horizontal-tb | vertical-rl | vertical-lr ;`

specifies whether lines of text are laid out horizontally or vertically and the direction in which blocks progress. Possible values:

**horizontal-tb** top-to-bottom block flow direction. Both the writing mode and the typographic mode are horizontal.

**vertical-rl** Right-to-left block flow direction. Both the writing mode and the typographic mode are vertical.

**vertical-lr** Left-to-right block flow direction. Both the writing mode and the typographic mode are vertical.

It is inherited and the default is `horizontal-tb`.

### 8.116 z-index

`z-index: auto | <integer> | inherit ;`

specifies the stack level of an element. Stack levels with a higher integer value are deemed to be closer to the user. Where two or more elements have the same stack level, they are stacked back to front according to their order in the document tree.

The background and borders always have a lower stack level than the lowest element in a stacking context.

**auto** is 0

It is not inherited and the default value is **auto**.

## 9 Values

### 9.1 Image

The value of any image property may be

`<url> | <gradient>`

where `<gradient>` may be

`<linear-gradient()> | <repeating-linear-gradient()> |  
<radial-gradient()> | <repeating-radial-gradient()>`

where

`linear-gradient([ <angle> | to <side-or-corner> ]? ,  
<color-stop-list>`

**angle** 0° is upwards and positive angles go clockwise

**side-or-corner** may be

`[left | right] || [top | bottom]`

**color-stop-list** comma separated list of colours; the definition of each colour may be extended as in:

`<color> && <length-percentage>? , [ <length-percentage>? ,  
<color> && <length-percentage>?> ]#`

and

`radial-gradient([ [ circle || <length> ] [ at <position> ]? , |  
[ ellipse || <length-percentage>{2} ] [ at <position> ]? , |  
[ [ circle | ellipse ] || <extent-keyword> ] [ at <position> ]? , |  
at <position> , ]?`

If `circle` is specified or is omitted, the `<size>` may be given explicitly as:

**<length>** Gives the radius of the circle explicitly. Negative values are invalid. Note: Percentages are not allowed here; they can only be used to specify the size of an elliptical gradient, not a circular one.

If `ellipse` is specified or is omitted, `<size>` may instead be given explicitly as:

**<length-percentage>{2}** the first value represents the horizontal radius, the second the vertical radius. Percentage values are relative to the corresponding dimension of the gradient box. Negative values are invalid.

**extent-keyword**

**closest-side** The ending shape is sized so that it exactly meets the side of the gradient box closest to the gradient's centre. If the shape is an `ellipse`, it exactly meets the closest side in each dimension.

**farthest-side** Same as `closest-side`, except the ending shape is sized based on the farthest side(s).

**closest-corner** The ending shape is sized so that it passes through the corner of the gradient box closest to the gradient's centre. If the shape is an `ellipse`, the ending shape is given the same aspect-ratio it would have if `closest-side` were specified.

**farthest-corner** Same as `closest-corner`, except the ending shape is sized based on the farthest corner. If the shape is an `ellipse`, the ending shape is given the same aspect ratio it would have if `farthest-side` were specified.

**position** determines the centre of the gradient; the default is `center`

**Note:** `repeating-linear-gradient()` and `repeating-radial-gradient()` are interpreted in the same way as `linear-gradient()` and `radial-gradient()` except that the color-stops are repeated infinitely.

## 10 Inheritance

The default values for all attributes are held in the *user-agent (browser) style sheet*; these may be overridden by the

**user style sheet** which is in turn overridden by the

**author style sheet** unless a rule in the user style sheet has the value `!important` as the final value in a declaration

This is to enable users with specific accessibility needs to control the presentation of content.

Unless the value is inherited, an element with an `id=" "` attribute takes precedence over an element with a `class=" "` attribute which takes precedence over any other element.

Where two or more rules might apply to the same element, the one with the highest specificity gains precedence; where two or more rules have the same specificity, the last one gains precedence.

The values of all attributes, whether applying to the root element or inherited from a parent element, are then resolved to

**computed values** which are absolute values; except where a computed value includes, for example, a percentage, this value is then turned into a

**used value** or an absolute value derived after ascertaining the dimensions of other elements and/or from applying the percentage; it is then possible that the user agent may not be able to apply this value but only an approximation of it, known as the

**actual value**

Applying the attribute `inherit` to a property of the root element causes it to be inherited when it would not otherwise have been inherited.

Where property values would not normally be inherited, this has been stated in section [8 Properties](#).

## References

Meyer, E. A. (2000). *Cascading style sheets: the definitive guide*. Sebastopol CA/Cambridge: O'Reilly.

## **A Sources for this document**

These notes are based on Meyer (2000)

[CSS Snapshot 2017](#)

[CSS Snapshot 2018](#)

[CSS Snapshot 2023](#)

## B SVG 1.0 colours

	HEX		HEX
aliceblue	#F0F8FF	antiquewhite	#FAEBD7
aqua (=cyan)	#00FFFF	aquamarine	#7FFFD4
azure	#F0FFFF	beige	#F5F5DC
bisque	#FFE4C4	black	#000000
blanchedalmond	#FFEBCD	blue	#0000FF
blueviolet	#8A2BE2	brown	#A52A2A
burlywood	#DEB887	cadetblue	#5F9EA0
chartreuse	#7FFF00	chocolate	#D2691E
coral	#FF7F50	cornflowerblue	#6495ED
cornsilk	#FFF8DC	crimson	#DC143C
cyan (=aqua)	#00FFFF	darkblue	#00008B
darkcyan	#008B8B	darkgoldenrod	#B8860B
darkgray (=darkgrey)	#A9A9A9	darkgreen	#006400
darkgrey (=darkgray)	#A9A9A9	darkkhaki	#BDB76B
darkmagenta	#8B008B	darkolivegreen	#556B2F
darkorange	#FF8C00	darkorchid	#9932CC
darkred	#8B0000	darksalmon	#E9967A
darkseagreen	#8FBC8F	darkslateblue	#483D8B
darkslategray (=darkslategrey)	#2F4F4F	darkslategrey (=darkslategray)	#2F4F4F
darkturquoise	#00CED1	darkviolet	#9400D3
deeppink	#FF1493	deepskyblue	#00BFFF
dimgray (=dimgrey)	#696969	dimgrey (=dimgray)	#696969
dodgerblue	#1E90FF	firebrick	#B22222
floralwhite	#FFFAF0	forestgreen	#228B22
fuchsia (=magenta)	#FF00FF	gainsboro	#DCDCDC
ghostwhite	#F8F8FF	gold	#FFD700
goldenrod	#DAA520	gray	#808080
green	#008000	greenyellow	#ADFF2F
grey	#808080	honeydew	#F0FFFO
hotpink	#FF69B4	indianred	#CD5C5C
indigo	#4B0082	ivory	#FFFFFF
khaki	#F0E68C	lavender	#E6E6FA
lavenderblush	#FFF0F5	lawngreen	#7CFC00
lemonchiffon	#FFFACD	lightblue	#ADD8E6
lightcoral	#F08080	lightcyan	#E0FFFF
lightgoldenrodyellow	#FAFAD2	lightgray (=lightgrey)	#D3D3D3
lightgreen	#90EE90	lightgrey (=lightgray)	#D3D3D3
lightpink	#FFB6C1	lightsalmon	#FFA07A
lightseagreen	#20B2AA	lightskyblue	#87CEFA
lightslategray (=lightslategrey)	#778899	lightslategrey (=lightslategray)	#778899
lightsteelblue	#B0C4DE	lightyellow	#FFFFE0
lime (= Unix green)	#00FF00	limegreen	#32CD32
linen	#FAF0E6	magenta (=fuchsia)	#FF00FF

	HEX		HEX
maroon	#800000	mediumaquamarine	#66CDAA
mediumblue	#0000CD	mediumorchid	#BA55D3
mediumpurple	#9370DB	mediumseagreen	#3CB371
mediumslateblue	#7B68EE	mediumspringgreen	#00FA9A
mediumturquoise	#48D1CC	mediumvioletred	#C71585
midnightblue	#191970	mintcream	#F5FFFA
mistyrose	#FFE4E1	moccasin	#FFE4B5
navajowhite	#FFDEAD	navy	#000080
oldlace	#FDF5E6	olive	#808000
olivedrab	#6B8E23	orange	#FFA500
orangered	#FF4500	orchid	#DA70D6
palegoldenrod	#EEE8AA	palegreen	#98FB98
paleturquoise	#AFEEEE	palevioletred	#DB7093
papayawhip	#FFEFD5	peachpuff	#FFDAB9
peru	#CD853F	pink	#FFC0CB
plum	#DDA0DD	powderblue	#B0E0E6
purple	#800080	rebeccapurple	#663399
red	#FF0000	rosybrown	#BC8F8F
royalblue	#4169E1	saddlebrown	#8B4513
salmon	#FA8072	sandybrown	#F4A460
seagreen	#2E8B57	seashell	#FFF5EE
sienna	#A0522D	silver	#C0C0C0
skyblue	#87CEEB	slateblue	#6A5ACD
slategray (=slategrey)	#708090	slategrey (=slategray)	#708090
snow	#FFFAFA	springgreen	#00FF7F
steelblue	#4682B4	tan	#D2B48C
teal	#008080	thistle	#D8BFD8
tomato	#FF6347	turquoise	#40E0D0
violet	#EE82EE	wheat	#F5DEB3
white	#FFFFFF	whitesmoke	#F5F5F5
yellow	#FFFF00	yellowgreen	#9ACD32

The document is licensed under the  
 Creative Commons Attribution-NonCommercial-ShareAlike 4.0  
 International

