Filter
- Box model
- Cascade
  - Introduction
  - Inheritance
  - Specificity
  - Property value processing
  - Shorthand properties
- Custom properties
- Colors
- Columns
- Conditional rules
- Containment
- CSSOM view
- Display
- Filter effects
- Flexbox
- Fonts
- Grid
- Images
- Lists and counters
- Logical properties
- Media queries
- Nesting style rules
- Overflow
- Positioning
- Scroll anchoring
- Scroll snap
- Selectors

Shorthand properties

In this article
- Tricky edge cases
- Background properties
- Font properties
- Border properties
- Margin and padding properties
- Position properties
- The universal shorthand property
- Shorthand properties
- See also

Shorthand properties are CSS properties that let you set the values of multiple other CSS properties simultaneously. Using a shorthand property, you can write more concise (and often more readable) style sheets, saving time and energy.

The CSS specification defines shorthand properties to group the definition of common properties acting on the same theme. For instance, the CSS `background` property is a shorthand property that's able to define the values of `background-color`, `background-image`, `background-repeat`, and `background-position`. Similarly, the most common font-related properties can be defined using the shorthand `font`, and the different margins around a box can be defined using the `margin` shorthand.

Tricky edge cases

There are a few edge cases to keep in mind when using shorthand properties.

Omitting properties

A value which is not specified is set to its initial value. That means that it **overrides** previously set values. For example:

CSS

```css
p {
  background-color: red;
  background: url("images/bg.gif") no-repeat left top;
}
```

This will not set the color of the background to `red` but to the default value for `background-color`, which is `transparent`.

Only the individual properties values can inherit. As missing values are replaced by their initial value, it is impossible to allow inheritance of individual properties by omitting them. The keyword `inherit` can be applied to a property, but only as a whole, not as a keyword for one value or another. That means that the only way to make some specific value to be inherited is to use the longhand property with the keyword `inherit`.

Ordering properties

Shorthand properties try not to force a specific order for the values of the properties they replace. This works well when these properties use values of different types, as the order has no importance, but this does not work as easily when several properties can have identical values.

Two important cases here are:

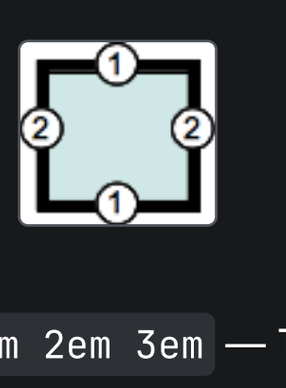
- properties related to the edges of a box, like `border-style`, `margin` or `padding`
- properties related to the corners of a box, like `border-radius`

EDGES OF A BOX

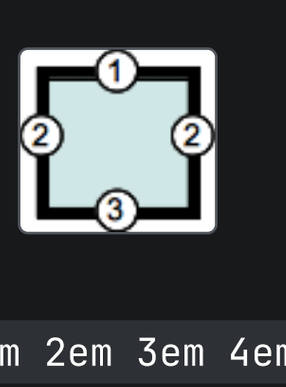
Shorthands handling properties related to edges of a box, like `border-style`, `margin` or `padding`, always use a consistent 1-to-4-value syntax representing those edges:

- **1-value syntax:** `border-width: 1em` — The single value represents all edges:

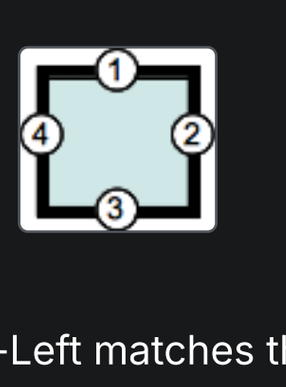
- **2-value syntax:** `border-width: 1em 2em` — The first value represents the vertical, that is top and bottom, edges, the second the horizontal ones, that is the left and right ones:



- **3-value syntax:** `border-width: 1em 2em 3em` — The first value represents the top edge, the second, the horizontal, that is left and right, ones, and the third value the bottom edge:



- **4-value syntax:** `border-width: 1em 2em 3em 4em` — The four values represent the top, right, bottom and left edges respectively, always in that order, that is clock-wise starting at the top:

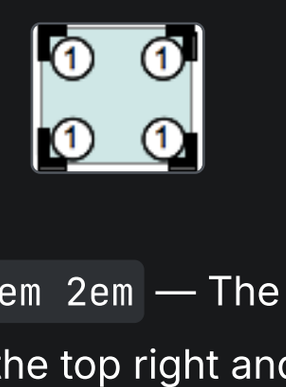


  The initial letter of Top-Right-Bottom-Left matches the order of the consonant of the word *trouble*: TRBL. You can also remember it as the order that the hands would rotate on a clock: `1em` starts in the 12 o'clock position, then `2em` in the 3 o'clock position, then `3em` in the 6 o'clock position, and `4em` in the 9 o'clock position.

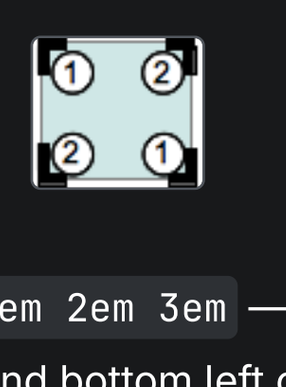
CORNERS OF A BOX

Similarly, shorthands handling properties related to corners of a box, like `border-radius`, always use a consistent 1-to-4-value syntax representing those corners:

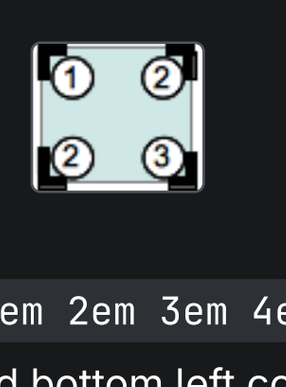
- **1-value syntax:** `border-radius: 1em` — The single value represents all corners:



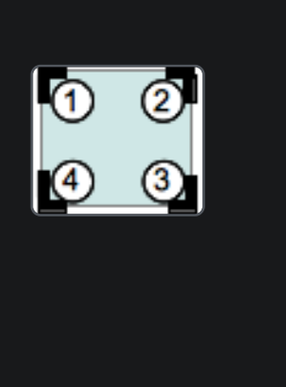
- **2-value syntax:** `border-radius: 1em 2em` — The first value represents the top left and bottom right corner, the second the top right and bottom left ones:



- **3-value syntax:** `border-radius: 1em 2em 3em` — The first value represents the top left corner, the second the top right and bottom left ones, and the third value the bottom right corner:



- **4-value syntax:** `border-radius: 1em 2em 3em 4em` — The four values represent the top left, top right, bottom right and bottom left corners respectively, always in that order, that is clock-wise starting at the top left:



Background properties

Consider a background with the following properties

CSS

```css
background-color: black;
background-image: url("images/bg.gif");
background-repeat: no-repeat;
background-position: left top;
```

These four declarations can be shortened to just one:

CSS

```css
background: black url("images/bg.gif") no-repeat left top;
```

(The shorthand form is actually the equivalent of the longhand properties above plus `background-attachment: scroll` and, in CSS3, some additional properties.)

See `background` for more detailed information, including CSS3 properties.

Font properties

Consider the following declarations:

CSS

```css
font-style: italic;
font-weight: bold;
font-size: 0.8em;
line-height: 1.2;
font-family: "Arial", sans-serif;
```

These 5 statements can be shortened to the following:

CSS

```css
font:
  italic bold 0.8em/1.2 "Arial",
  sans-serif;
```

This shorthand declaration is actually equivalent to the longhand declarations above plus `font-variant: normal`, `font-size-adjust: none`, and `font-stretch: normal`.

Border properties

With borders, the width, color, and style can be simplified into one declaration. For example, consider the following CSS:

CSS

```css
border-width: 1px;
border-style: solid;
border-color: black;
```

It can be simplified as:

CSS

```css
border: 1px solid black;
```

Margin and padding properties

Shorthand versions of margin and padding values work similarly; the margin property allows for shorthand values to be specified using one, two, three, or four values. Consider the following CSS declarations:

CSS

```css
margin-top: 10px;
margin-right: 5px;
margin-bottom: 10px;
margin-left: 5px;
```

They are the same as the following declaration using the four value shorthand. Note that the values are in clockwise order, beginning at the top: top, right, bottom, then left (TRBL, the consonants in "trouble").

CSS

```css
margin: 10px 5px 10px 5px;
```

Margin shorthand rules for one, two, three and four value declarations are:

- When **one** value is specified, it applies the same margin to **all four sides**.
- When **two** values are specified, the first margin applies to the **top and bottom**, the second to the **left and right**.
- When **three** values are specified, the first margin applies to the **top**, the second to the **left and right**, the third to the **bottom**.
- When **four** values are specified, the margins apply to the **top**, **right**, **bottom**, and **left** in that order (clockwise).

Position properties

With position, the shorthand versions of top, right, bottom and left can be simplified into one declaration. For example, consider the following CSS:

CSS

```css
top: 0;
right: 20px;
bottom: 0;
left: 20px;
```

It can be simplified as:

CSS

```css
inset: 0 20px 0 20px;
```

Just like margins and paddings, the inset values are ordered clockwise - top, right, bottom, then left (TRBL).

The universal shorthand property

CSS provides a universal shorthand property, `all`, which applies its value to every property in the document. Its purpose is to change the properties' inheritance model.

See Handling conflicts or Introducing the CSS Cascade for more information about how inheritance works in CSS.

Shorthand properties

- `all`
- `animation`
- `animation-range`
- `background`
- `border`
- `border-block`
- `border-block-end`
- `border-block-start`
- `border-bottom`
- `border-color`
- `border-image`
- `border-inline`
- `border-inline-end`
- `border-inline-start`
- `border-left`
- `border-radius`
- `border-right`
- `border-style`
- `border-top`
- `border-width`
- `column-rule`
- `columns`
- `contain-intrinsic-size`
- `container`
- `flex`
- `flex-flow`
- `font`
- `font-synthesis`
- `font-variant`
- `gap`
- `grid`
- `grid-area`
- `grid-column`
- `grid-row`
- `grid-template`
- `inset`
- `inset-block`
- `inset-inline`
- `list-style`
- `margin`
- `margin-block`
- `margin-inline`
- `mask`
- `mask-border`
- `offset`
- `outline`
- `overflow`
- `overscroll-behavior`
- `padding`
- `padding-block`
- `padding-inline`
- `place-content`
- `place-items`
- `place-self`
- `position-try`
- `scroll-margin`
- `scroll-margin-block`
- `scroll-margin-inline`
- `scroll-padding`
- `scroll-padding-block`
- `scroll-padding-inline`
- `scroll-timeline`
- `text-box`
- `text-decoration`
- `text-emphasis`
- `text-wrap`
- `transition`
- `view-timeline`
- `-webkit-text-stroke`
- `-webkit-border-before`
- `-webkit-mask-box-image`

See also

- CSS syntax
- At-rules
- Specificity
- Inheritance
- Introducing the CSS Cascade
- Learn: Handling conflicts
- Learn: Cascade layers
- CSS cascading and inheritance module
- Visual formatting models
- Initial, computed, used, and actual values
- Value definition syntax

This page was last modified on Oct 13, 2025 by MDN contributors.
View this page on GitHub • Report a problem with this content