Shorthand properties

English (US) ▾

Change language

▲ Jump to section

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

Even if they are very convenient to use, there are a few edge cases to keep in mind when using them:

1. A value which is not specified is set to its initial value. That sounds anecdotal, but it really means that it **overrides** previously set values. Therefore:

   ```
   background-color: red;
   background: url(images/bg.gif) no-repeat left top;
   ```

   will not set the color of the background to `red` but to `background-color`'s default, `transparent`, as the second rule has precedence.

2. 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`.

3. 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. Handling of these cases are grouped in several categories:

1. 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:

	The 1-value syntax: `border-width: 1em` — The unique value represents all edges
	The 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.
	The 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
	The 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).

2. 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:

	The 1-value syntax: `border-radius: 1em` — The unique value represents all corners
	The 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.
	The 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
	The 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

A background with the following properties ...

```
background-color: #000;
background-image: url(images/bg.gif);
background-repeat: no-repeat;
background-position: left top;
```

... can be shortened to just one declaration:

```
background: #000 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

The following declarations ...

```
font-style: italic;
font-weight: bold;
font-size: .8em;
line-height: 1.2;
font-family: Arial, sans-serif;
```

... can be shortened to the following:

```
font: italic bold .8em/1.2 Arial, sans-serif;
```

This shorthand declaration is actually equivalent to the longhand declarations above plus `font-variant: normal` and `font-size-adjust: none` (CSS2.0 / CSS3), `font-stretch: normal` (CSS3).

Border properties

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

```
border-width: 1px;
border-style: solid;
border-color: #000;
```

... can be simplified as:

```
border: 1px solid #000;
```

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. The following CSS declarations ...

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

... 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").

```
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).

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 to one of:

CSS provides four special universal property values for controlling inheritance. Every CSS property accepts these values.

`inherit`

Sets the property value applied to a selected element to be the same as that of its parent element. Effectively, this "turns on inheritance".

`initial`

Sets the property value applied to a selected element to the initial value of that property.

`unset`

Resets the property to its natural value, which means that if the property is naturally inherited it acts like `inherit`, otherwise it acts like `initial`.

> **Note**: There is also a newer value, `revert`, which has limited browser support.

> **Note**: See Origin of CSS declarations in Introducing the CSS Cascade for more information on each of these and how they work.

We can look at a list of links and explore how universal values work. The live example below allows you to play with the CSS and see what happens when you make changes. Playing with code really is the best way to get to grips with HTML and CSS.

For example:

1. The second list item has the class `my-class-1` applied. This sets the color of the `<a>` element nested inside to inherit. If you remove the rule how does it change the color of the link?
2. Do you understand why the third and fourth links are the color that they are? Check the description of the values above if not.
3. Which of the links will change color if you define a new color for the `<a>` element — for example `a { color: red; }`?

- Default [link](#) color
- Inherit the [link](#) color
- Reset the [link](#) color
- Unset the [link](#) color

```css
body {
    color: green;
}

.my-class-1 a {
    color: inherit;
}

.my-class-2 a {
    color: initial;
}

.my-class-3 a {
    color: unset;
}
```

```html
<ul>
    <li>Default <a href="#">link</a> color</li>
    <li class="my-class-1">Inherit the <a href="#">link</a> color</li>
    <li class="my-class-2">Reset the <a href="#">link</a> color</li>
    <li class="my-class-3">Unset the <a href="#">link</a> color</li>
</ul>
```

Reset

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

See also

- CSS Key Concepts: CSS syntax, at-rule, comments, specificity and inheritance, the box, layout modes and visual formatting models, and margin collapsing, or the initial, computed, resolved, specified, used, and actual values. Definitions of value syntax, shorthand properties and replaced elements.
- Shorthand properties: `all`, `animation`, `background`, `border`, `border-block-end`, `border-block-start`, `border-bottom`, `border-color`, `border-image`, `border-inline-end`, `border-inline-start`, `border-left`, `border-radius`, `border-right`, `border-style`, `border-top`, `border-width`, `column-rule`, `columns`, `flex`, `flex-flow`, `font`, `gap`, `grid`, `grid-area`, `grid-column`, `grid-row`, `grid-template`, `list-style`, `margin`, `mask`, `offset`, `outline`, `overflow`, `padding`, `place-content`, `place-items`, `place-self`, `scroll-margin`, `scroll-padding`, `text-decoration`, `text-emphasis`, `transition`

Related Topics

[CSS](#)

[CSS Reference](#)

[Shorthand properties](#)

Found a problem with this page?

- [Source on **GitHub**](#)
- [Report a problem with this content on**GitHub**](#)
- Want to fix the problem yourself? See[our Contribution guide](#).

Last modified: Dec 17, 2020, [by MDN contributors](#)