Text and typography

- Change the typeface
- Use italic and oblique fonts
- Make text bold
- Change the size of text
- Change the space between lines
- Change the space between characters
- Change the space between words
- font shorthand
- Change the case of text
- Add underlines, overlines, and through-lines to text
- Add an indent to your text
- Deal with overflowing or hidden content
- Control white-space
- Control how words break
- Change text alignment
- Change the direction of text
- Change the flow of text
- Change the orientation of text
- Add a shadow to text
- Variable fonts
- Pseudo-elements
- ::first-letter and ::first-line pseudo-elements
- ::selection pseudo-element
- font-variant
- Resources

Text is one of the core building blocks of the web.

When making a website, you don't necessarily need to style your text; HTML actually has some pretty reasonable default styling.

However, text will likely make up the majority of your website, so it's worthwhile to add some styling to spruce it up. By changing a few basic properties, you can significantly improve the reading experience for your users!

{% Codepen { user: 'web-dot-dev', id: 'WNEJWGy', height: 590 } %}

In this module, we'll first look at some fundamental CSS font properties like `font-family`, `font-style`, `font-weight`, and `font-size`. Then, we'll dive into properties that affect paragraphs of text, such as `text-indent` and `word-spacing`. The module finishes with some more advanced topics such as variable fonts and pseudo-elements.

Change the typeface

{% BrowserCompat 'css.properties.font-family' %}

Use `font-family` to change the typeface of your text.

The `font-family` property accepts a comma-separated list of strings, either referring to *specific* or *generic* font families. Specific font families are quoted strings, such as "Helvetica", "EB Garamond", or "Times New Roman". Generic font families are keywords such as `serif`, `sans-serif`, and `monospace` (find the full list of options on MDN). The browser will display the first available typeface from the provided list.

{% Aside %} When the browser chooses which font to display from your `font-family` declaration, it doesn't stop at the first available font in the list. Instead, it selects fonts one character at a time. If a particular character isn't available in the first font in the list, it moves on to the next, and so on. {% endAside %}

When using `font-family`, you should specify at least one generic font family in case the user's browser doesn't have your preferred fonts. Generally, the fallback generic font family should be similar to your preferred fonts: if using `font-family: "Helvetica"` (a sans-serif font family), your fallback should be `sans-serif` to match.

{% Codepen { user: 'web-dot-dev', id: 'yLojraG', height: 470, tab: 'css,result' } %}

Use italic and oblique fonts

{% BrowserCompat 'css.properties.font-style' %}

Use `font-style` to set whether text should be italic or not. `font-style` accepts one of the following keywords: `normal`, `italic`, and `oblique`.

{% Aside %} Q: What's the difference between `italic` and `oblique`? A: In fonts that support it, `font-style: italic` is typically a cursive version of the regular typeface. `font-style: oblique` displays a slanted version of regular typeface. {% endAside %}

{% Codepen { user: 'web-dot-dev', id: 'MWvGRjx', height: 280 } %}

Make text bold

{% BrowserCompat 'css.properties.font-weight' %}

Use `font-weight` to set the "boldness" of text. This property accepts keyword values (`normal`, `bold`), relative keyword values (`lighter`, `bolder`), and numeric values (`100` to `900`).

The keywords `normal` and `bold` are equivalent to the numeric values `400` and `700`, respectively.

The keywords `lighter` and `bolder` are calculated relative to the parent element. See MDN's Meaning of Relative Weights for a handy chart showing how this value is determined.

{% Aside %} Most fonts, especially the "web-safe" ones, only support the weights `400` (`normal`) and `700` (`bold`). When importing fonts using `@font-face` or `@import`, you can choose specific weights you want to pull in. Still, non-variable fonts only support numeric values for `font-weight` in the 100s, e.g. `100`, `200`, `300`, etc. If you want to use `font-weight: 321` (for example), you'll have to use a Variable Font. {% endAside %}

{% Codepen { user: 'web-dot-dev', id: 'gOxKxNz', height: 580 } %}

Change the size of text

{% BrowserCompat 'css.properties.font-size' %}

Use `font-size` to control the size of your text elements. This property accepts length values, percentages, and a handful of keyword values.

In addition to length and percentage values, `font-size` accepts some *absolute* keyword values (`xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`) and a couple of *relative* keyword values (`smaller`, `larger`). The relative values are relative to the parent element's `font-size`.

{% Aside %} Q: What's the difference between `em` and `rem`? A: In CSS, `em` represents the `font-size` inherited from the element's parent. For example, `font-size: 2em` is equivalent to the parent's `font-size` multiplied by two. `rem` is similar, but instead represents the `font-size` inherited from the *root element*, i.e. `<html>`. {% endAside %}

{% Codepen { user: 'web-dot-dev', id: 'eYEroda', height: 370 } %}

{% Codepen { user: 'web-dot-dev', id: 'vYJzYzw' } %}

Change the space between lines

{% BrowserCompat 'css.properties.line-height' %}

Use `line-height` to specify the height of each line in an element. This property accepts either a number, length, percentage, or the keyword `normal`. Generally, it's recommended to use a number instead of a length or percentage to avoid issues with inheritance.

{% Codepen { user: 'web-dot-dev', id: 'XWaqQjv', height: 600 } %}

{% Codepen { user: 'web-dot-dev', id: 'ExvLJNx', height: 820 } %}

Change the space between characters

{% BrowserCompat 'css.properties.letter-spacing' %}

Use `letter-spacing` to control the amount of horizontal space between characters in your text. This property accepts length values such as `em`, `px`, and `rem`.

Note that the specified value *increases* the amount of natural space between characters. In the demo below, try selecting an individual letter to see the size of its letterbox and how it changes with `letter-spacing`.

{% Codepen { user: 'web-dot-dev', id: 'abyGxBz', height: 560 } %}

Change the space between words

{% BrowserCompat 'css.properties.word-spacing' %}

Use `word-spacing` to increase or decrease the length of space between each word in your text. This property accepts length values such as `em`, `px`, and `rem`. Note that the length you specify is for *extra* space in addition to the normal spacing. This means that `word-spacing: 0` is equivalent to `word-spacing: normal`.

{% Codepen { user: 'web-dot-dev', id: 'xxLjeRG', height: 280 } %}

`font` shorthand

You can use the shorthand `font` property to set many font-related properties at once. The list of possible properties are `font-family`, `font-size`, `font-stretch`, `font-style`, `font-variant`, `font-weight`, and `line-height`.

Check out MDN's `font` article for the specifics of how to order these properties.

{% Codepen { user: 'web-dot-dev', id: 'JjyvVbY', height: 270 } %}

Change the case of text

{% BrowserCompat 'css.properties.text-transform' %}

Use `text-transform` to modify the capitalization of your text without needing to change the underlying HTML. This property accepts the following keyword values: `uppercase`, `lowercase`, and `capitalize`.

{% Codepen { user: 'web-dot-dev', id: 'qBXYwqb', height: 450 } %}

Add underlines, overlines, and through-lines to text

{% BrowserCompat 'css.properties.text-decoration' %}

Use `text-decoration` to add lines to your text. Underlines are most commonly used, but it's possible to add lines above your text or right through it!

The `text-decoration` property is shorthand for the more specific properties detailed below.

The `text-decoration-line` property accepts the keywords `underline`, `overline`, and `line-through`. You can also specify multiple keywords for multiple lines.

{% Codepen { user: 'web-dot-dev', id: 'LYjmvbN', height: 460, tab: 'css,result' } %}

The `text-decoration-color` property sets the color of all decorations from `text-decoration-line`.

{% Codepen { user: 'web-dot-dev', id: 'oNedOYL', height: 460, tab: 'css,result' } %}

The `text-decoration-style` property accepts the keywords `solid`, `double`, `dotted`, `dashed`, and `wavy`.

{% Codepen { user: 'web-dot-dev', id: 'dyzeLOp', height: 460, tab: 'css,result' } %}

The `text-decoration-thickness` property accepts any length values and sets the stroke width of all decorations from `text-decoration-line`.

{% Codepen { user: 'web-dot-dev', id: 'VwzxNmm', height: 460, tab: 'css,result' } %}

The `text-decoration` property is a shorthand for all the above properties.

{% Codepen { user: 'web-dot-dev', id: 'YzxLMpN', height: 460, tab: 'css,result' } %}

{% Aside %} Use `text-underline-position` to offset the underline of a `text-decoration: underline` by the specified amount. This property doesn't work for `overline` or `line-through`. {% endAside %}

Add an indent to your text

{% BrowserCompat 'css.properties.text-indent' %}

Use `text-indent` to add an indent to your blocks of text. This property takes either a length (for example, `10px`, `2em`) or a percentage of the containing block's width.

{% Codepen { user: 'web-dot-dev', id: 'RwZyOoV', height: 300 } %}

Deal with overflowing or hidden content

{% BrowserCompat 'css.properties.text-overflow' %}

Use `text-overflow` to specify how hidden content is represented. There are two options: `clip` (the default), which truncates the text at the point of overflow; and `ellipsis`, which displays an ellipsis (…) at the point of overflow.

{% Codepen { user: 'web-dot-dev', id: 'GRvdLNv', height: 250, tab: 'css,result' } %}

Control white-space

{% BrowserCompat 'css.properties.white-space' %}

The `white-space` property is used to specify how whitespace in an element should be handled. For more details, check out the `white-space` article on MDN.

{% Codepen { user: 'web-dot-dev', id: 'PoKegbO', height: 970 } %}

`white-space: pre` can be useful for rendering ASCII art or carefully indented code blocks.

{% Codepen { user: 'web-dot-dev', id: 'OJjZGbz', height: 440 } %}

Control how words break

{% BrowserCompat 'css.properties.word-break' %}

Use `word-break` to change how words should be “broken” when they would overflow the line. By default, the browser will not split words. Using the keyword value `break-all` for `word-break` will instruct the browser to break words at individual characters if necessary.

{% Codepen { user: 'web-dot-dev', id: 'BadxEQY' } %}

Change text alignment

{% BrowserCompat 'css.properties.text-align' %}

Use `text-align` to specify the horizontal alignment of text in a block or table-cell element. This property accepts the keyword values `left`, `right`, `start`, `end`, `center`, `justify`, and `match-parent` .

The values `left` and `right` align the text to the left and right sides of the block, respectively.

Use `start` and `end` to represent the location of the start and end of a line of text in the current writing mode. Therefore, `start` maps to `left` in English, and to `right` in Arabic script which is written right to left (RTL). They're logical alignments, learn more in our logical properties module.

Use `center` to align the text to the center of the block.

The value of `justify` organizes the text and changes word spacings automatically so that the text lines up with both the left and right edges of the block.

{% Codepen { user: 'web-dot-dev', id: 'bGrMJBM', height: 570 } %}

Change the direction of text

{% BrowserCompat 'css.properties.direction' %}

Use `direction` to set the direction of your text, either `ltr` (left to right, the default) or `rtl` (right to left). Some languages like Arabic, Hebrew, or Persian are written right to left, so `direction: rtl` should be used. For English and all other left-to-right languages, use `direction: ltr`.

{% Aside 'caution' %} Generally, you should favor using the HTML attribute `dir` instead of `direction`. Check out this StackOverflow discussion for more details. {% endAside %}

Change the flow of text

{% BrowserCompat 'css.properties.writing-mode' %}

Use `writing-mode` to change the way text flows and is arranged. The default is `horizontal-tb`, but you can also set `writing-mode` to `vertical-lr` or `vertical-rl` for text that you want to flow horizontally.

{% Codepen { user: 'web-dot-dev', id: 'WNEJWoK', height: 680 } %}

Change the orientation of text

{% BrowserCompat 'css.properties.text-orientation' %}

Use `text-orientation` to specify the orientation of characters in your text. The valid values for this property are `mixed` and `upright`. This property is only relevant when `writing-mode` is set to something other than `horizontal-tb`.

{% Codepen { user: 'web-dot-dev', id: 'QWMrPGV', height: 660 } %}

Add a shadow to text

{% BrowserCompat 'css.properties.text-shadow' %}

Use `text-shadow` to add a shadow to your text. This property expects three lengths (`x-offset`, `y-offset`, and `blur-radius`) and a color.

{% Codepen { user: 'web-dot-dev', id: 'jOLxRVe', height: 530 } %}

Check out the `text-shadow` section of our module on Shadows to learn more.

Variable fonts

Typically, “normal” fonts require importing different files for different versions of the typeface, e.g. bold, italic, or condensed. Variable fonts are fonts that can contain many different variants of a typeface in one file.



Check out our article on Variable Fonts for more details.

Pseudo-elements

{% Aside 'key-term' %} A *pseudo-element* is a part of an element that you can target via CSS keywords without having to add more HTML. Check out our module on pseudo-elements for a deep-dive into this subject! {% endAside %}

`::first-letter` and `::first-line` pseudo-elements

{% BrowserCompat 'css.selectors.first-letter' %}

The `::first-letter` and `::first-line` pseudo-elements target a text element's first letter and first line respectively.

{% Codepen { user: 'web-dot-dev', id: 'KKvRYNr', height: 270 } %}

`::selection` pseudo-element

{% BrowserCompat 'css.selectors.selection' %}

Use the `::selection` pseudo-element to change the appearance of user-selected text.

When using this pseudo-element, only certain CSS properties can be used: `color`, `background-color`, `text-decoration`, `text-shadow`, `stroke-color`, `fill-color`, `stroke-width`.

{% Codepen { user: 'web-dot-dev', id: 'MWvGRbx', height: 390, tab: 'css,result' } %}

font-variant

{% BrowserCompat 'css.properties.font-variant' %}

The `font-variant` property is a shorthand for a number of CSS properties that let you choose font variants like `small-caps` and `slashed-zero`. The CSS properties this shorthand includes are `font-variant-alternates`, `font-variant-caps`, `font-variant-east-asian`, `font-variant-ligatures`, and `font-variant-numeric`. Check out the links on each property for more details about its usage.

{% Codepen { user: 'web-dot-dev', id: 'eYEroBa', height: 260 } %}

{% Assessment 'typography' %}

Resources

- Font best practices discusses importing fonts, rendering fonts, and other best practices for using fonts on the web.
- MDN Fundamental text and font styling.

□□□ #2
□ □□□ □□□ 2026-10-31 22:46:46 UTC
□ □□□ □□□ 2026-10-31 22:46:49 UTC