![mdn web docs]

Fundamental text and font styling

In this article we'll start you on your journey towards mastering text styling with CSS. Here we'll go through all the basic fundamentals of text/font styling in detail, including setting font weight, family and style, font shorthand, text alignment and other effects, and line and letter spacing.

Prerequisites:	HTML basics (study Introduction to HTML), CSS basics (study Introduction to CSS).
Objective:	To learn the fundamental properties and techniques needed to style text on web pages.

What is involved in styling text in CSS?

Text inside an element is laid out inside the element's content box. It starts at the top left of the content area (or the top right, in the case of RTL language content), and flows towards the end of the line. Once it reaches the end, it goes down to the next line and flows to the end again. This pattern repeats until all the content has been placed in the box. Text content effectively behaves like a series of inline elements, being laid out on lines adjacent to one another, and not creating line breaks until the end of the line is reached, or unless you force a line break manually using the `<br>` element.

> **Note:** If the above paragraph leaves you feeling confused, then no matter — go back and review our Box model article to brush up on the box model theory before carrying on.

The CSS properties used to style text generally fall into two categories, which we'll look at separately in this article:

- **Font styles**: Properties that affect a text's font, e.g., which font gets applied, its size, and whether it's bold, italic, etc.
- **Text layout styles**: Properties that affect the spacing and other layout features of the text, allowing manipulation of, for example, the space between lines and letters, and how the text is aligned within the content box.

> **Note:** Bear in mind that the text inside an element is all affected as one single entity. You can't select and style subsections of text unless you wrap them in an appropriate element (such as a `<span>` or `<strong>`), or use a text-specific pseudo-element like ::first-letter (selects the first letter of an element's text), ::first-line (selects the first line of an element's text), or ::selection (selects the text currently highlighted by the cursor).

Fonts

Let's move straight on to look at properties for styling fonts. In this example, we'll apply some CSS properties to the following HTML sample:

HTML

```html
<h1>Tommy the cat</h1>

<p>Well I remember it as though it were a meal ago…</p>

<p>
  Said Tommy the Cat as he reeled back to clear whatever foreign matter may
have
  nestled its way into his mighty throat. Many a fat alley rat had met its
  demise while staring point blank down the cavernous barrel of this awesome
  prowling machine. Truly a wonder of nature this urban predator — Tommy the
cat
  had many a story to tell. But it was a rare occasion such as this that he
```

```
  did.
</p>
```

You can find the [finished example on GitHub](#)   (see also [the source code](#)   ).

Color

The [color](#) property sets the color of the foreground content of the selected elements, which is usually the text, but can also include a couple of other things, such as an underline or overline placed on text using the [text-decoration](#) property.

 color can accept any [CSS color unit](#), for example:

CSS	Play

```css
p {
  color: red;
}
```

This will cause the paragraphs to become red, rather than the standard browser default of black, like so:

                                                                              Play

Tommy the cat

Well I remember it as though it were a meal ago…

Said Tommy the Cat as he reeled back to clear whatever foreign matter may have nestled its way into his mighty throat. Many a fat alley rat had met its demise while staring point blank down the cavernous barrel of this awesome prowling machine. Truly a wonder of nature this urban predator — Tommy the cat had many a story to tell. But it was a rare occasion such as this that he did.

Font families

To set a different font for your text, you use the `font-family` property — this allows you to specify a font (or list of fonts) for the browser to apply to the selected elements. The browser will only apply a font if it is available on the machine the website is being accessed on; if not, it will just use a browser [default font](). A simple example looks like so:

```css
p {
  font-family: Arial;
}
```

This would make all paragraphs on a page adopt the arial font, which is found on any computer.

Web safe fonts

Speaking of font availability, there are only a certain number of fonts that are generally available across all systems and can therefore be used without much worry. These are the so-called **web safe fonts**.

Most of the time, as web developers we want to have more specific control over the fonts used to display our text content. The problem is to find a way to know which font is available on the computer used to see our web pages. There is no way to know this in every case, but the web safe fonts are known to be available on nearly all instances of the most used operating systems (Windows, macOS, the most common Linux distributions, Android, and iOS).

The list of actual web safe fonts will change as operating systems evolve, but it's reasonable to consider the following fonts web safe, at least for now (many of them have been popularized thanks to the Microsoft *Core fonts for the Web* initiative in the late 90s and early 2000s):

Name	Generic type	Notes
Arial	sans-serif	It's often considered best practice to also add *Helvetica* as a preferred alternative to *Arial* as, although their font faces are almost identical, *Helvetica* is considered to have a nicer shape, even if *Arial* is more broadly available.
Courier New	monospace	Some OSes have an alternative (possibly older) version of the *Courier New* font called *Courier*. It's considered best practice to use both with *Courier New* as the preferred alternative.
Georgia	serif	
Times New Roman	serif	Some OSes have an alternative (possibly older) version of the *Times New Roman* font called *Times*. It's considered best practice to use both with *Times New Roman* as the preferred alternative.
Trebuchet MS	sans-serif	You should be careful with using this font — it isn't widely available on mobile OSes.
Verdana	sans-serif	

> **Note:** Among various resources, the cssfontstack.com website maintains a list of web safe fonts available on Windows and macOS operating systems, which can help you make your decision about what you consider safe for your usage.
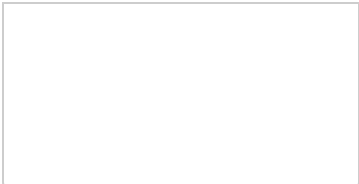
> **Note:** There is a way to download a custom font along with a webpage, to allow you to customize your font usage in any way you want: **web fonts**. This is a little bit more complex, and we will discuss it in a separate article later on in the module.

Default fonts

CSS defines five generic names for fonts: `serif`, `sans-serif`, `monospace`, `cursive`, and `fantasy`. These are very generic and the exact font face used from these generic names can vary between each browser and each operating system that they are displayed on. It represents a *worst case scenario* where the browser will try its best to provide a font that looks appropriate. `serif`, `sans-serif`, and `monospace` are quite predictable and should provide something reasonable. On the other hand, `cursive` and `fantasy` are less predictable and we recommend using them very carefully, testing as you go.

The five names are defined as follows:

Term	Definition	Example
`serif`	Fonts that have serifs (the flourishes and other small details you see at the ends of the strokes in some typefaces).	Play My big red elephant
`sans-serif`	Fonts that don't have serifs.	Play
`monospace`	Fonts where every character has the same width, typically used in code listings.	Play

Term	Definition	Example
`cursive`	Fonts that are intended to emulate handwriting, with flowing, connected strokes.	Play
`fantasy`	Fonts that are intended to be decorative.	Play

Font stacks

Since you can't guarantee the availability of the fonts you want to use on your webpages (even a web font *could* fail for some reason), you can supply a **font stack** so that the browser has multiple fonts it can choose from. This involves a `font-family` value consisting of multiple font names separated by commas, e.g.,

```css
p {
  font-family: "Trebuchet MS", Verdana, sans-serif;
}
```

In such a case, the browser starts at the beginning of the list and looks to see if that font is available on the machine. If it is, it applies that font to the selected elements. If not, it moves on to the next font, and so on.

It is a good idea to provide a suitable generic font name at the end of the stack so that if none of the listed fonts are available, the browser can at least provide something approximately suitable. To emphasize this point, paragraphs are given the browser's default serif font if no other option is available — which is usually Times New Roman — this is no good for a sans-serif font!

> **Note:** While you can use font family names that contain a space, such as `Trebuchet MS`, without quoting the name, to avoid mistakes in escaping, it is recommended to quote font family names that contain white space, digits, or punctuation characters other than hyphens.

> **Warning:** Any font family name which could be misinterpreted as a generic family name or a CSS-wide keyword must be quoted. While the font-family names can be included as a <custom-ident> or a <string>, font family names that happen to be the same as a CSS-wide property value, like `initial`, or `inherit`, or CSS have the same name as one to the generic font family names, like `sans-serif` or `fantasy`, must be included as a quoted string. Otherwise, the font family name will be interpreted as being the equivalent CSS keyword or generic family name. When used as keywords, the generic font family names — `serif`, `sans-serif`, `monospace`, `cursive`, and `fantasy` — and the global CSS keywords MUST NOT be quoted, as strings are not interpreted as CSS keywords.

A font-family example

Let's add to our previous example, giving the paragraphs a sans-serif font:
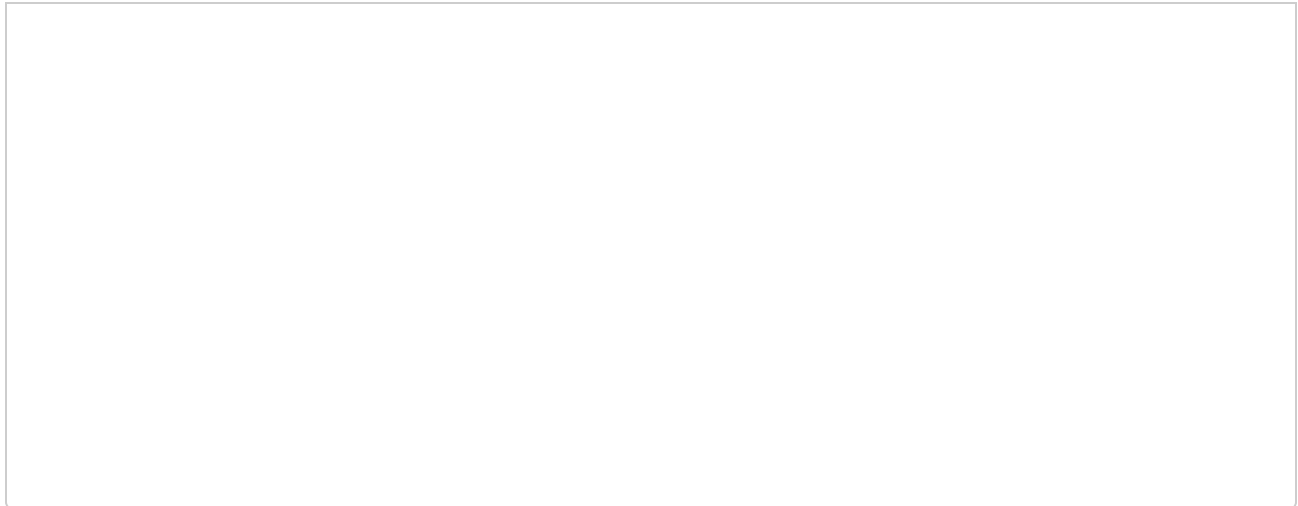
CSS	Play

```
p {
  color: red;
  font-family: Helvetica, Arial, sans-serif;
}
```

This gives us the following result:

Play

Font size

In our previous module's [CSS values and units](#) article, we reviewed length and size units. Font size (set with the `font-size` property) can take values measured in most of these units (and others, such as [percentages](#)); however, the most common units you'll use to size text are:

- `px` (pixels): The number of pixels high you want the text to be. This is an absolute unit — it results in the same final computed value for the font on the page in pretty much any situation.

- `em` s: 1 `em` is equal to the font size set on the parent element of the current element we are styling (more specifically, the width of a capital letter M contained inside the parent element). This can become tricky to work out if you have a lot of nested elements with different font sizes set, but it is doable, as you'll see below. Why bother? It is quite natural once you get used to it, and you

can use `em` to size everything, not just text. You can have an entire website sized using `em`, which makes maintenance easy.

- `rem`s: These work just like `em`, except that 1 `rem` is equal to the font size set on the root element of the document (i.e. <ins>`<html>`</ins>), not the parent element. This makes doing the maths to work out your font sizes much easier.

The `font-size` of an element is inherited from that element's parent element. This all starts with the root element of the entire document — <ins>`<html>`</ins> — the standard `font-size` of which is set to `16px` across browsers. Any paragraph (or another element that doesn't have a different size set by the browser) inside the root element will have a final size of `16px`. Other elements may have different default sizes. For example, an <ins>h1</ins> element has a size of `2em` set by default, so it will have a final size of `32px`.

Things become more tricky when you start altering the font size of nested elements. For example, if you had an <ins>`<article>`</ins> element in your page, and set its `font-size` to 1.5 `em` (which would compute to 24 `px` final size), and then wanted the paragraphs inside the `<article>` elements to have a computed font size of 20 `px`, what `em` value would you use?

HTML

```
<!-- document base font-size is 16px -->
<article>
  <!-- If my font-size is 1.5em -->
  <p>My paragraph</p>
  <!-- How do I compute to 20px font-size? -->
</article>
```

You would need to set its `em` value to 20/24, or 0.83333333 `em`. The maths can be complicated, so you need to be careful about how you style things. It is best to use `rem` where you can to keep things simple, and avoid setting the `font-size` of container elements where possible.

Font style, font weight, text transform, and text decoration

CSS provides four common properties to alter the visual weight/emphasis of text:

- `font-style` : Used to turn italic text on or off. Possible values are as follows (you'll rarely use this, unless you want to turn some italic styling off for some reason):

  - `normal` : Sets the text to the normal font (turns existing italics off).

  - `italic` : Sets the text to use the italic version of the font, if available; if not, it will simulate italics with oblique instead.

  - `oblique` : Sets the text to use a simulated version of an italic font, created by slanting the normal version.

- `font-weight` : Sets how bold the text is. This has many values available in case you have many font variants available (such as *-light*, *-normal*, *-bold*, *-extrabold*, *-black*, etc.), but realistically you'll rarely use any of them except for `normal` and `bold` :

  - `normal` , `bold` : Normal and bold font weight.

  - `lighter` , `bolder` : Sets the current element's boldness to be one step lighter or heavier than its parent element's boldness.

  - `100` – `900` : Numeric boldness values that provide finer grained control than the above keywords, if needed.

- `text-transform` : Allows you to set your font to be transformed. Values include:

  - `none` : Prevents any transformation.

  - `uppercase` : Transforms all text to capitals.

  - `lowercase` : Transforms all text to lower case.

  - `capitalize` : Transforms all words to have the first letter capitalized.

  - `full-width` : Transforms all glyphs to be written inside a fixed-width square, similar to a monospace font, allowing aligning of, e.g., Latin characters along with Asian language glyphs (like Chinese, Japanese, Korean).

- `text-decoration` : Sets/unsets text decorations on fonts (you'll mainly use this to unset the default underline on links when styling them). Available values are:

  - `none` : Unsets any text decorations already present.

- `underline` : Underlines the text.

- `overline` : Gives the text an overline.

- `line-through` : Puts a strikethrough over the text.

  You should note that [text-decoration](#) can accept multiple values at once if you want to add multiple decorations simultaneously, for example, `text-decoration:` `underline overline` . Also note that [text-decoration](#) is a shorthand property for [text-decoration-line](#) , [text-decoration-style](#) , and [text-decoration-color](#) . You can use combinations of these property values to create interesting effects, for example: `text-decoration: line-through red wavy` .

Let's look at adding a couple of these properties to our example:

Our new result is like so:

CSS	Play

```css
html {
  font-size: 10px;
}


h1 {
  font-size: 5rem;
  text-transform: capitalize;
}


h1 + p {
  font-weight: bold;
}


p {
  font-size: 1.5rem;
  color: red;
  font-family: Helvetica, Arial, sans-serif;
}
```

Play

Text drop shadows

You can apply drop shadows to your text using the `text-shadow` property. This takes up to four values, as shown in the example below:

```
CSS
text-shadow: 4px 4px 5px red;
```

The four properties are as follows:

1. The horizontal offset of the shadow from the original text — this can take most available CSS [length and size units](#), but you'll most commonly use `px`; positive values move the shadow right, and negative values left. This value has to be included.

2. The vertical offset of the shadow from the original text. This behaves similarly to the horizontal offset, except that it moves the shadow up/down, not left/right. This value has to be included.

3. The blur radius: a higher value means the shadow is dispersed more widely. If this value is not included, it defaults to 0, which means no blur. This can take most available CSS [length and size units](#).

4. The base color of the shadow, which can take any [CSS color unit](#). If not included, it defaults to `currentcolor`, i.e. the shadow's color is taken from the element's

`color` property.

Multiple shadows

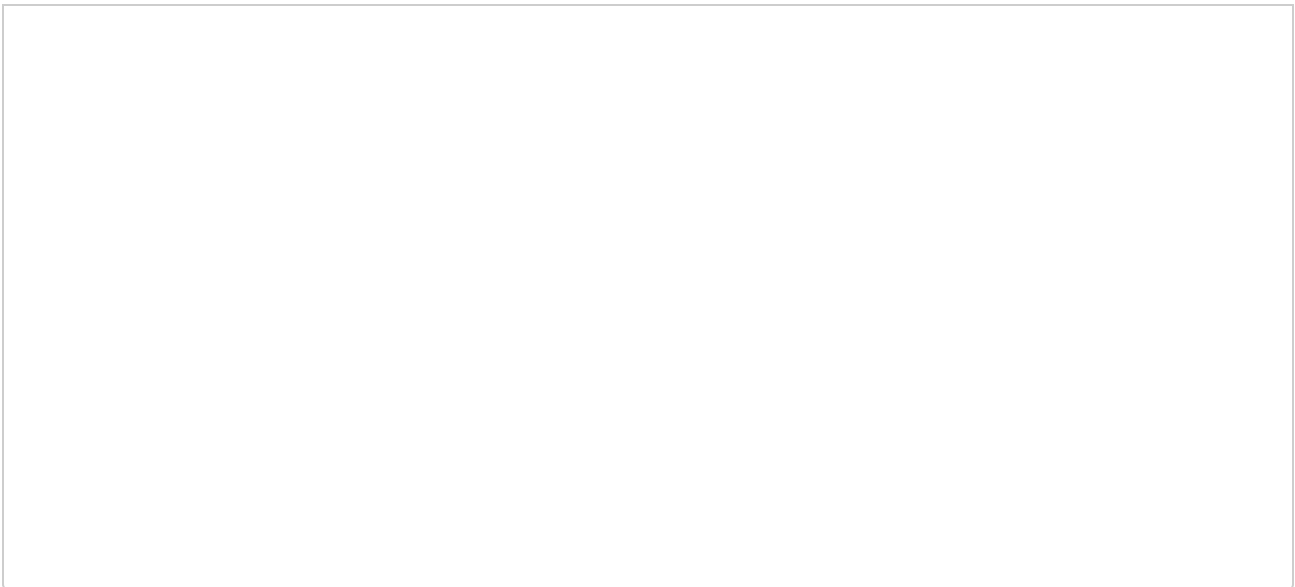You can apply multiple shadows to the same text by including multiple shadow values separated by commas, for example:

CSS	Play

```css
h1 {
  text-shadow:
    1px 1px 1px red,
    2px 2px 1px red;
}
```

If we applied this to the [h1](#) element in our Tommy The Cat example, we'd end up with this:

Play

> **Note:** You can see more interesting examples of `text-shadow` usage in the Sitepoint article [Moonlighting with CSS text-shadow](#).

Text layout

With basic font properties out of the way, let's have a look at properties we can use to affect text layout.

Text alignment

The `text-align` property is used to control how text is aligned within its containing content box. The available values are listed below. They work in pretty much the same way as they do in a regular word processor application:

- `left` : Left-justifies the text.

- `right` : Right-justifies the text.

- `center` : Centers the text.

- `justify` : Makes the text spread out, varying the gaps in between the words so that all lines of text are the same width. You need to use this carefully — it can look terrible, especially when applied to a paragraph with lots of long words in it. If you are going to use this, you should also think about using something else along with it, such as `hyphens` , to break some of the longer words across lines.

If we applied `text-align: center;` to the h1 in our example, we'd end up with this:

CSS	Play

```css
html {
  font-size: 10px;
}

h1 {
  font-size: 5rem;
  text-transform: capitalize;
  text-shadow:
    1px 1px 1px red,
    2px 2px 1px red;
  text-align: center;
}

h1 + p {
  font-weight: bold;
```

```
}

p {
  font-size: 1.5rem;
  color: red;
  font-family: Helvetica, Arial, sans-serif;
}
```

Play

Line height

The `line-height` property sets the height of each line of text. This property can not only take most [length and size units](), but can also take a unitless value, which acts as a multiplier and is generally considered the best option. With a unitless value, the `font-size` gets multiplied and results in the `line-height`. Body text generally looks nicer and is easier to read when the lines are spaced apart. The recommended line height is around 1.5 – 2 (double spaced). To set our lines of text to 1.6 times the height of the font, we'd use:

CSS   Play

```
p {
  line-height: 1.6;
}
```

Applying this to the <p> elements in our example would give us this result:

CSS   Play

```css
html {
  font-size: 10px;
}

h1 {
  font-size: 5rem;
  text-transform: capitalize;
  text-shadow:
    1px 1px 1px red,
    2px 2px 1px red;
  text-align: center;
}

h1 + p {
  font-weight: bold;
}

p {
  font-size: 1.5rem;
  color: red;
  font-family: Helvetica, Arial, sans-serif;
  line-height: 1.6;
}
```

                                                                    Play

---

Letter and word spacing

The `letter-spacing` and `word-spacing` properties allow you to set the spacing between letters and words in your text. You won't use these very often, but might find a use for them to obtain a specific look, or to improve the legibility of a particularly dense font. They can take most [length and size units](#).

To illustrate, we could apply some word- and letter-spacing to the first line of each `<p>` element in our HTML sample with:

CSS	Play

```css
p::first-line {
  letter-spacing: 4px;
  word-spacing: 4px;
}
```

This renders our HTML as:

CSS	Play

```css
html {
  font-size: 10px;
}
```

```css
h1 {
  font-size: 5rem;
  text-transform: capitalize;
  text-shadow:
    1px 1px 1px red,
    2px 2px 1px red;
  text-align: center;
  letter-spacing: 2px;
}

h1 + p {
  font-weight: bold;
}

p {
  font-size: 1.5rem;
  color: red;
  font-family: Helvetica, Arial, sans-serif;
  line-height: 1.6;
  letter-spacing: 1px;
}
```

Play

Other properties worth looking at

The above properties give you an idea of how to start styling text on a webpage, but there are many more properties you could use. We just wanted to cover the most important ones here. Once you've become used to using the above, you should also explore the following:

Font styles:

- `font-variant` : Switch between small caps and normal font alternatives.

- `font-kerning` : Switch font kerning options on and off.

- `font-feature-settings` : Switch various [OpenType](#) font features on and off.

- `font-variant-alternates` : Control the use of alternate glyphs for a given font-face.

- `font-variant-caps` : Control the use of alternate capital glyphs.

- `font-variant-east-asian` : Control the usage of alternate glyphs for East Asian scripts, like Japanese and Chinese.

- `font-variant-ligatures` : Control which ligatures and contextual forms are used in text.

- `font-variant-numeric` : Control the usage of alternate glyphs for numbers, fractions, and ordinal markers.

- `font-variant-position` : Control the usage of alternate glyphs of smaller sizes positioned as superscript or subscript.

- `font-size-adjust` : Adjust the visual size of the font independently of its actual font size.

- `font-stretch` : Switch between possible alternative stretched versions of a given font.

- `text-underline-position` : Specify the position of underlines set using the `text-decoration-line` property `underline` value.

- `text-rendering` : Try to perform some text rendering optimization.

Text layout styles:

- `text-indent` : Specify how much horizontal space should be left before the beginning of the first line of the text content.

- `text-overflow` : Define how overflowed content that is not displayed is signaled to users.

- `white-space` : Define how whitespace and associated line breaks inside the element are handled.

- `word-break` : Specify whether to break lines within words.

- `direction` : Define the text direction. (This depends on the language and usually it's better to let HTML handle that part as it is tied to the text content.)

- `hyphens` : Switch on and off hyphenation for supported languages.

- `line-break` : Relax or strengthen line breaking for Asian languages.

- `text-align-last` : Define how the last line of a block or a line, right before a forced line break, is aligned.

- `text-orientation` : Define the orientation of the text in a line.

- `overflow-wrap` : Specify whether or not the browser may break lines within words in order to prevent overflow.

- `writing-mode` : Define whether lines of text are laid out horizontally or vertically and the direction in which subsequent lines flow.

Font shorthand

Many font properties can also be set through the shorthand property `font` . These are written in the following order: `font-style` , `font-variant` , `font-weight` , `font-stretch` , `font-size` , `line-height` , and `font-family` .

Among all those properties, only `font-size` and `font-family` are required when using the `font` shorthand property.

A forward slash has to be put in between the `font-size` and `line-height` properties.

A full example would look like this:

```css
CSS
```

```css
font:
  italic normal bold normal 3em/1.5 Helvetica,
  Arial,
  sans-serif;
```

Active learning: Playing with styling text

In this active learning session we don't have any specific exercises for you to do. We'd just like you to have a good play with some font/text layout properties. See for yourself what you can come up with! You can either do this using offline HTML/CSS files, or enter your code into the live editable example below.

If you make a mistake, you can always reset it using the *Reset* button.

Play

Summary

We hope you enjoyed playing with text in this article! The next article will provide you with all you need to know about styling HTML lists.

Help improve MDN

Was this page helpful to you?

Yes   No

Learn how to contribute.

This page was last modified on Mar 27, 2024 by MDN contributors.