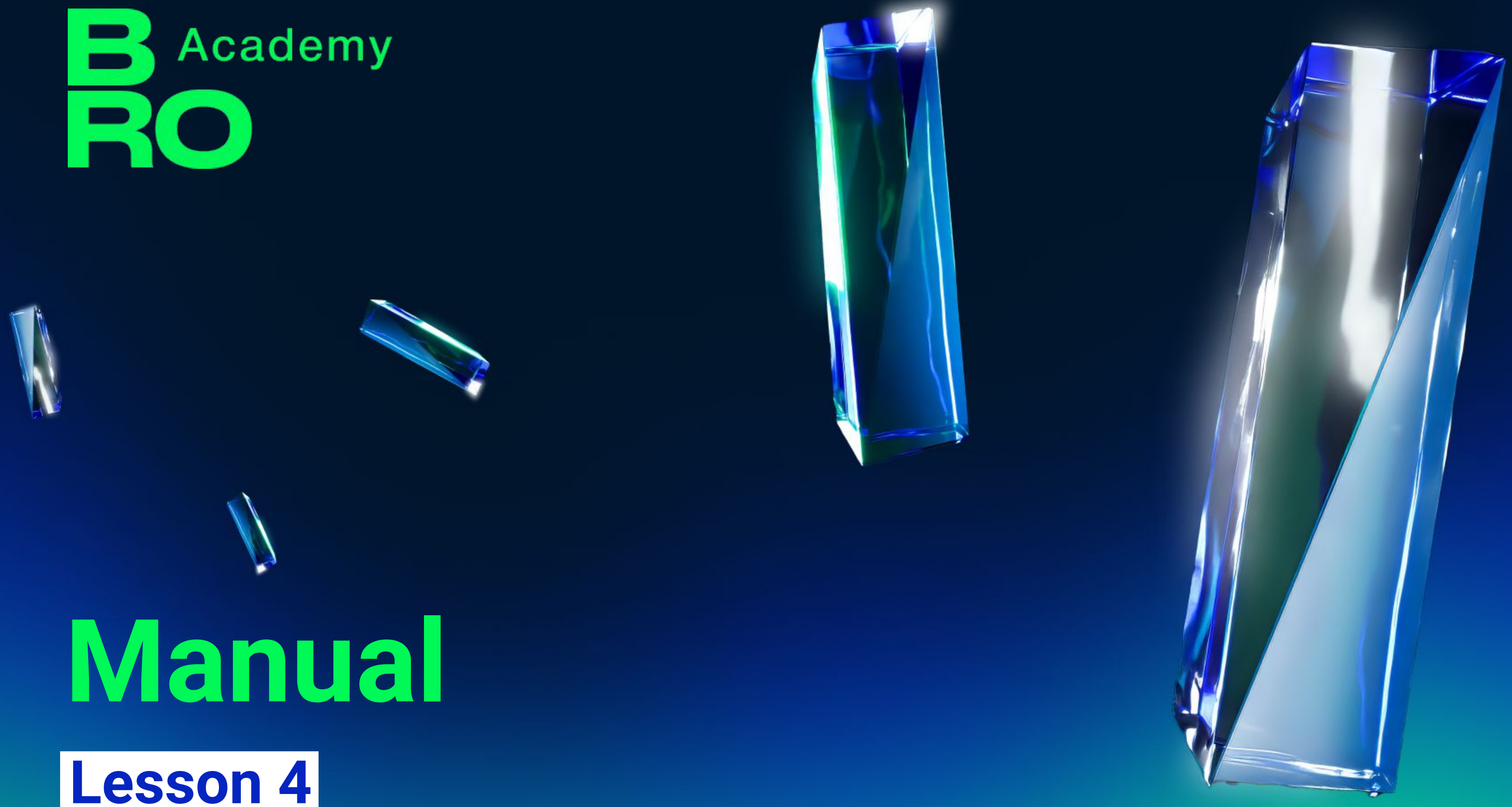
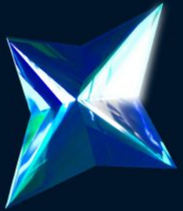
BRO Academy



Manual

Lesson 4



CSS is a language that works in tandem with HTML to style web pages.

It helps control how elements on a page will look and be positioned.

CSS describes how an element should be displayed on the screen, on paper, by voice, or using other media.

CSS rules consist of five main parts:

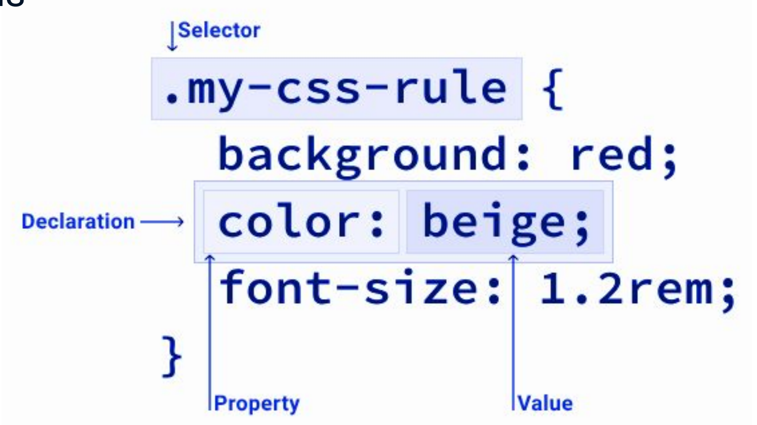
Selector: The selector helps choose elements in an HTML document that you want to style. For example, the p selector will select all paragraphs in the document.

Declaration block: This is a part of the code enclosed in curly braces {}. It includes everything between the opening and closing braces.

Declaration: It includes a property and a value. This tells the browser how the selected element should look.

Property: The property defines which aspect of the element will be styled. For example, in the following example, the color property will be styled.

Value: The value specifies exactly what style will be applied to the property.



```
.my-css-rule {
  background: red;
  color: beige;
  font-size: 1.2rem;
}
```

Types of

Selectors

- Tag Selector

  `h1 {}`

- Class Selector

  `.main-heading {}`

- ID Selector

  `#reasons-dection {}`

- Attribute Selector

  `a[href] {}`

- Pseudo-classes

  `a:hover {}`

- Pseudo-elements

  `p::first-line {}`

- Universal Selector

  `* {}`

- Grouped Selectors

  `h1, .heading {}`

Attribute

Selectors

Attribute with Value	`input[type=”text”] { }`	`<input type="text">`
		`<input type="checkbox">`

Boolean Attribute:	`input[required] { }`	`<input type="text" required>`
		`<input type="text">`

Attribute with Space-separated Values	`p[class~="blue"] { }`	`<p class="blue"></p>`
		`<p class="blue red"></p>`
		`<p class="blue-new"></p>`

Attribute with Hyphen-separated Values	`p[class|="blue"] { }`	`<p class="blue"></p>`
		`<p class="blue red"></p>`
		`<p class="blue-new"></p>`

Attribute Starting with Value	`a[href^="http"] { }`	`<a href="http://abc.com"></a>`
Attribute Ending with Value	`a[href$=".pdf"] { }`	`<a href="form.pdf"></a>`
Attribute Containing Value	`a[href*="abc.com"] { }`	`<a href="http://www.abc.com"></a>`

Selector Combinators

Combinators allow individual selectors to be combined into new types of selectors.

descendant combinator	**space**	`body article p`	an elements are nested inside each other	`<body> <article> <p></p> </article> </body>`	`<body> <section> <p></p> </section> <p></p> </body>`
child combinator	**>**	`article > p`	an element is an immediate child	`<article> <p></p> <a></a> </article>`	`<article> <div> <p></p> </div> <p></p> </article>`
adjacent sibling combinator	**+**	`p + img`	an element that comes immediately after	`<article> <p></p> <img/> <a></a> </article>`	`<article> <p></p> <a></a> <img/> </article>`
general sibling combinator	**~**	`p ~ img`	an element located anywhere below the code, inside a common parent	`<article> <p></p> <a></a> <img/> </article>`	`<article> <p></p> <a> <img/> </a> </article>`

Selector

Combinations examples

`p {}`	`<p class="intro module"></p>`
	`<div class="intro"></div>`
	`<p class="default-text"></p>`

`p a { }`	`<p>` `<div>`
	`<a></a>` `<a></a>`
	`</p>` `</div>`

`.intro {}`	`<p class="intro"></p>`
	`<div class="intro"></div>`

`div.intro { }`	`<p class="intro"></p>`
	`<div class="intro"></div>`

`.intro.module { }`	`<p class="module"></p>`
	`<p class="module intro"></p>`
	`<p class="intro module"></p>`
	`<p class="intro"></p>`

`.intro .module { }`	`<p class="intro module"></p>`
	`<div class="intro ` ~~`module`~~ `">`
	`  <p class="module"></p>`
	`</div>`

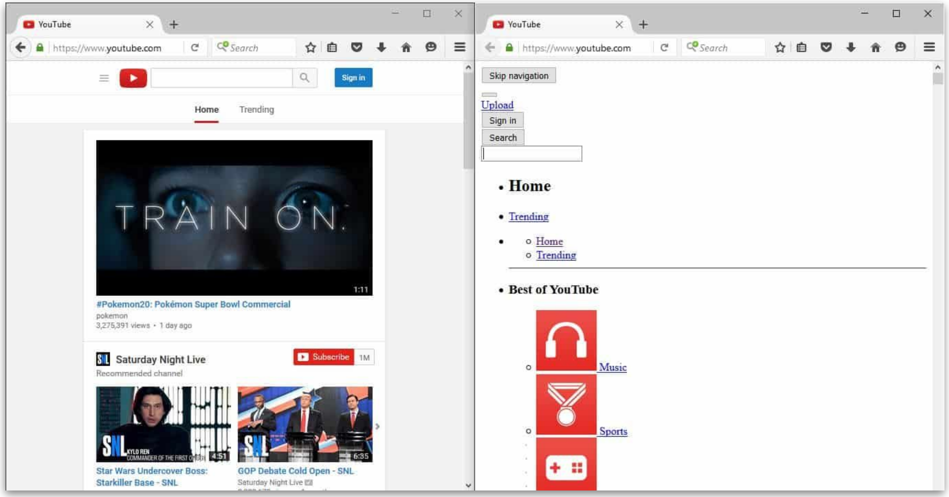
`.intro,`
`.module { }`
	`<p class="intro"></p>`
	`<div class="module"></div>`

`.intro {} ≠ .Intro {}`	`<p class="intro"></p>`
	`<div class="Intro"></div>`

`.nav ul li a { }`	`.nav a { }`
`/* comment */`	
Selectors are read from right to left **nav ul li {}**	The rule applies to the **li** element, which is part of the **ul** element, which is inside the **nav** element.

Type of CSS

properties



- Animation and transition properties
- Color and background properties
- Box model properties
- Dimension properties
- Generated content properties
- Layout properties

- **Typographical properties**
- **List properties**
- Table properties
- Transform properties
- Visual formatting properties

typographical properties

- font-family
- font-size
- font-weight
- font-style
- font-variant
- letter-spacing
- word-spacing
- line-height
- text-align
- text-decoration
- text-indent
- text-transform
- text-shadow
- vertical-align
- white-space
- direction

list properties

- list-style-type
- list-style-position
- list-style-image

color and background

- **color**
- background-color
- background-image
- background-repeat
- background-position
- background-attachment

Types of CSS values

Address	Addresses are used to specify the path to a file, such as setting a background image on a page.	`url(../image.jpg)`
String	A text string used as a property value.	`font-family: "Times New Roman", serif;`
Color	Sets the color of an element's background, text, or other parameters.	`#ff0000, rgb(255, 0, 0)`
Function	Allows you to calculate a value based on given conditions.	`calc(10px * 2)`
Number	The value can be an integer containing digits from 0 to 9 and a decimal number, where the integer and decimal parts are separated by a period.	`1.2, 0`
+ Time	Specifies time in seconds or milliseconds.	`3s, 3ms`
+ Percentages	Percentage notation is used when the value should be relative to the parent element or when sizes depend on external conditions.	`10%`
+ Size	Absolute units of measurement are used to specify the sizes of various elements in CSS.	`10px`
+ Angle	Specifies the angle of tilt or rotation.	`90deg`
CSS Directives	These are special constructs that start with the @ symbol. Directives most often affect the entire document but do not style anything themselves.	`@import`
Keywords	Predefined values for properties that need to be memorized.	`display: block;`

Applying Styles in Order

1
Inheritance

2
Cascading

3
Specificity

Inheritance of Styles

Inheritance in CSS allows some CSS properties to be passed from parent elements to child elements. Inheritance is designed to make developers' jobs easier. Otherwise, they would have to style all child elements as well as their parents. CSS files would be much larger and harder to create and maintain.

Many CSS properties are not inherited. If all CSS properties were inherited, it would create additional difficulties for developers, as they would have to disable unwanted CSS properties for child elements. Generally, only properties that make your job easier are inherited.

For example, you can set font-size and font-family on the <body> element. These properties will be inherited by all child elements. You can then override these properties as needed.

Style

Inherited

```
p { color: value; }
```

List related properties

```
p { list-style-image: value; }
p { list-style-position: value; }
p { list-style-type: value; }
p { list-style: value; }
```

Text related properties

```
p { font-family: value; }
p { font-size: value; }
p { font-style: value; }
p { font-variant: value; }
p { font-weight: value; }
p { font: value; }
p { letter-spacing: value; }
p { line-height: value; }
p { orphans: value; }
p { text-align: value; }
p { text-indent: value; }
p { text-transform: value; }
p { widows: value; }
p { word-spacing: value; }
```

* – styles that need to be set for body to be inherited by all page elements

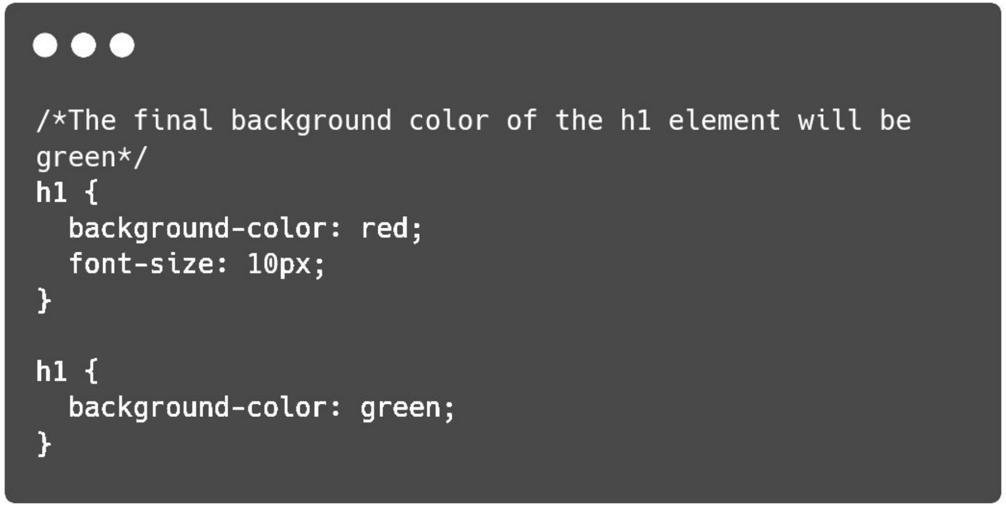
Cascading Styles

At some point, while working on a project, you may find that the CSS you expect to be applied to an element is not working.

Usually, the problem is that you have created two rules that can potentially apply to the same element.

The cascade and the closely related concept of specificity are the mechanisms that control which rule is applied when there is such a conflict.

The style of your element may be determined by a rule other than the one you expected, so you need to understand how these mechanisms work.

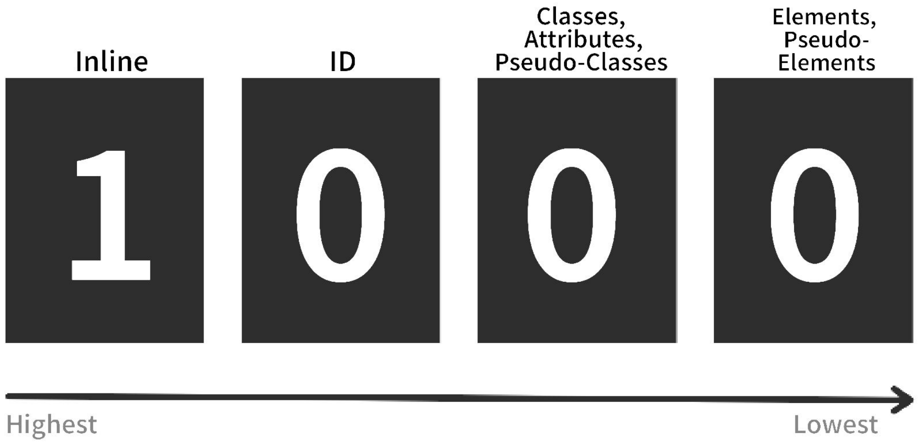


```
/*The final background color of the h1 element will be
green*/
h1 {
  background-color: red;
  font-size: 10px;
}

h1 {
  background-color: green;
}
```

Simply put, the cascade of style sheets means that the order of rules in CSS matters; when two rules with the same specificity are applicable, the one that comes last in the CSS is used.

Specificity of selectors

Specificity determines how the browser decides which rule to apply when multiple rules have different selectors but can still be applied to the same element.


Inline
ID
Classes, Attributes, Pseudo-Classes
Elements, Pseudo-Elements
1
0
0
0
Highest
Lowest

specificity = weight

Avoid overusing weight; try to create lightweight elements.

селектор * has no specificity weight (0)

Selector nesting should be no more than three levels.

Applying Styles in Order

1

Browser Styles

2

User Styles

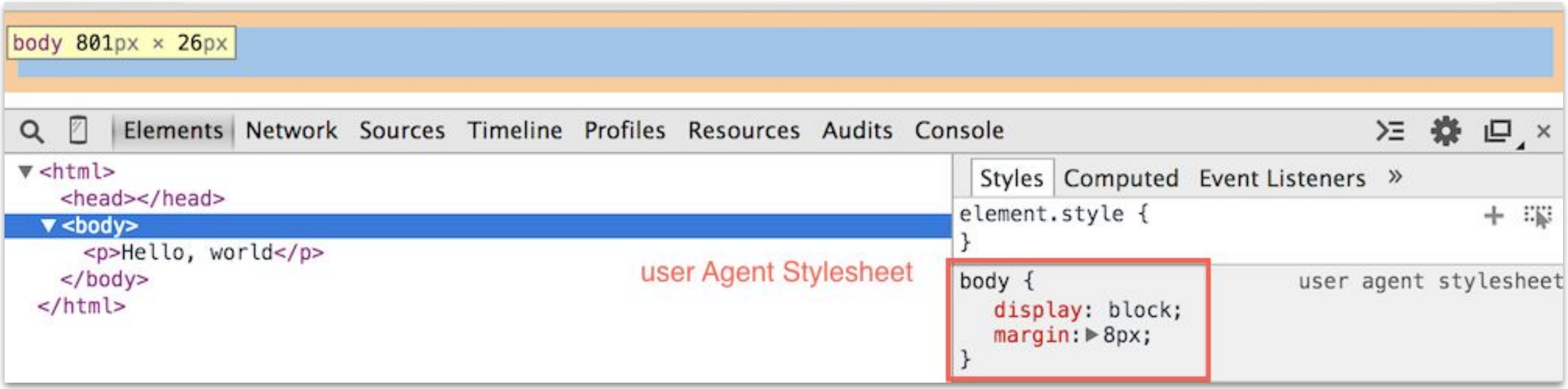
3

Author Styles

Default Browser Styles

User agent stylesheet

is the default set of CSS styles provided by the web browser to ensure that web pages are displayed in a readable and usable manner, even when no specific styles are defined by the author of the web page. These default styles are applied to various HTML elements to establish a consistent baseline across different web browsers.
Understanding how these default styles work helps developers create more consistent and controlled designs by overriding and customizing them with their own CSS.



```
▼<html>
   <head></head>
 ▼<body>
     <p>Hello, world</p>
   </body>
 </html>
```

```
element.style {
}
body {
   display: block;
   margin:▶8px;
}
```

Normalize CSS

normalize.css

- maintains useful browser settings rather than erasing them
- standardizes styles for many HTML elements
- corrects errors and inconsistencies
- improves usability with subtle improvements
- and includes comments and detailed documentation.

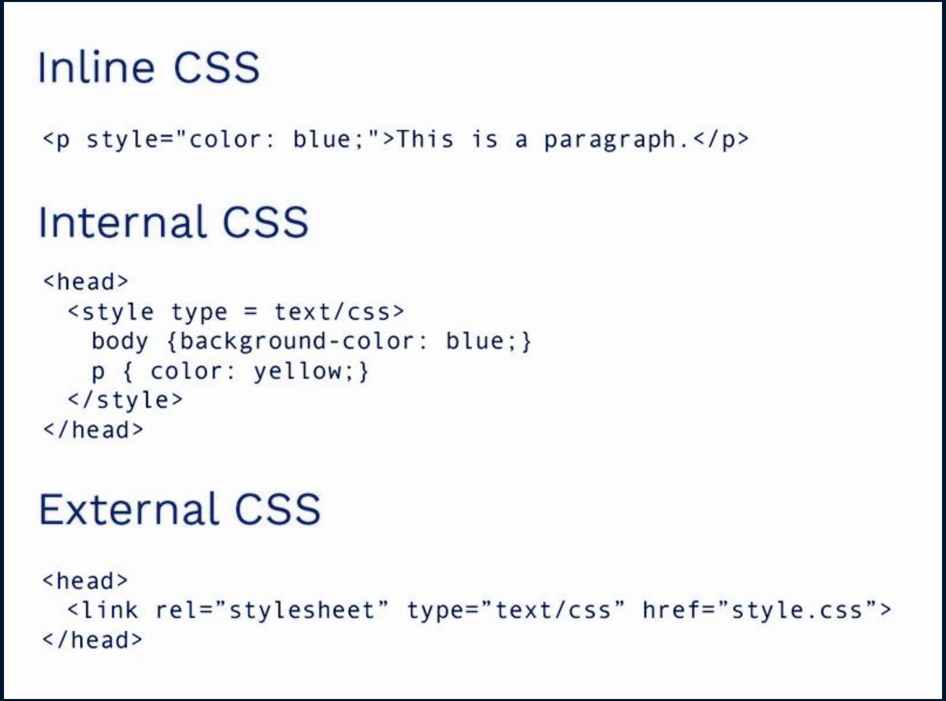
Connects to your style file:

```
<link  rel="stylesheet"  type="text/css"
 href="https://necolas.github.io/normalize.css">
```

Developers often use CSS normalize stylesheets to minimize the differences in User Agent Stylesheets between browsers.

Normalize stylesheet makes the default styling consistent across browsers without removing it entirely.

Connecting Styles


Inline CSS

```
<p style="color: blue;">This is a paragraph.</p>
```

Internal CSS

```
<head>
  <style type = text/css>
    body {background-color: blue;}
    p { color: yellow;}
  </style>
</head>
```

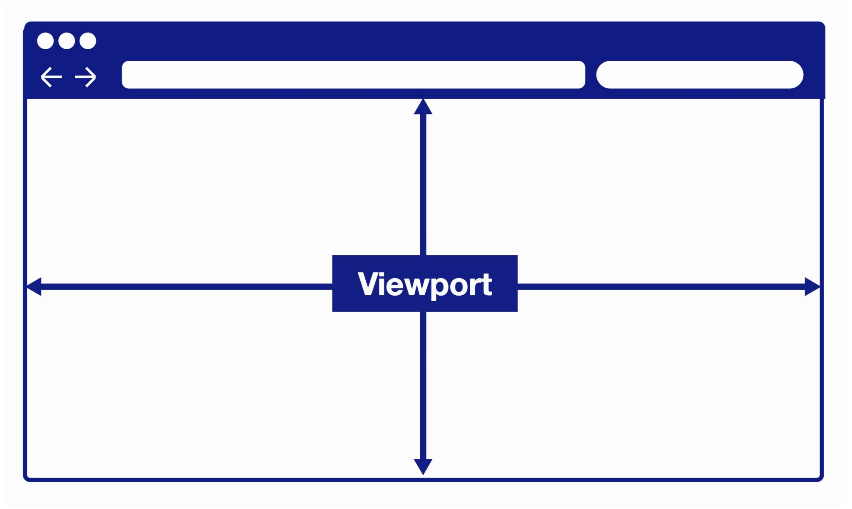
External CSS

```
<head>
  <link rel=”stylesheet” type=”text/css” href=”style.css”>
</head>
```

- inline styles
- embedded (tag style in head)
- linked (tag link)
  One stylesheet for all pages.

viewport

The viewport is the full viewing area within a browser window. On a computer, you can resize the browser window, making it larger or smaller, which automatically changes the viewport size. The viewport also varies with the device and will be smaller on a mobile phone and tablet devices than on a computer screen.
Users can adjust it, and it is measured in **px**



Units of Measurement Relative to the Viewport

vh / vw
(% of the viewport height/width)

vmin / vmax
(% of the smallest/largest height/width of the viewport)

svw, lvw, dvw – new!

Absolute

Units of Measurement

are tied to actual physical sizes.

A pixel is the basic, absolute, and final unit of measurement. A pixel in CSS is what you see on the screen. It sets the value for screen resolution. All values in web development are recalculated to pixels by the browser.

A physical pixel is a pixel on the device's matrix. For example, when we say that the screen width of a device is 480px, we mean that there are 480 indivisible cells across the entire width, where 1px equals one cell.

A CSS pixel is a CSS measurement unit equal to 1/96th of an inch. One pixel is not always equal to one cell on the screen. This depends on the screen density; the higher the number of physical pixels on the screen, the higher its density and the more detailed the image displayed on it.

`px`

- `width: 100px; height: 100px;`
- `margin: 20px; padding: 20px;`
- `font-size: 15px;`

Relative

Units of Measurement

describe values that depend on other values.

- **em (from parent) / rem (from html tag)**
    - `font-size: 1.5rem; font-size: 1.2em;`
    - `margin: 0.5rem; padding: 0.8em;`

- **% (relative to the same property of the parent):**
    - Element sizes relative to parent element sizes:
      `width: 50%; height: 75%;`
    - Margins and paddings relative to parent element sizes:
      `margin: 10%; padding: 5%;`

Colors in

CSS

- hex / hexa
- rgb / rgba
- hsl / hsla
- color keywords
- currentcolor
- transparent

Color properties:

- color
- background-color
- text-shadow
- text-decoration-color
- border-color
- box-shadow
- outline

Color HEX/HEXA

Hexadecimal symbols include digits (0-9) and letters (A-F), which can be written in both uppercase and lowercase. In six-digit notation, the first pair of digits represents the red channel. The second pair represents the green channel, and the last pair represents the blue channel. Three-digit notation is a shorthand version of six-digit notation. The first digit represents the red channel, the second represents the green, and the last represents the blue.

```
p {color: #FFAA00;}
p {color: #FA0;}
```

In eight-digit notation, the first six digits are interpreted the same as in six-digit notation. The last two digits, interpreted as a hexadecimal number, determine the alpha channel of the color. For example, 00 represents a fully transparent color, and ff represents a fully opaque color.

```
p {color: #FFAA0000;} <- full transparency
p {color: #FFAA00FF;} <- full opacity
```

Color

RGB/RGBA

```
p {color: rgb(0, 0, 0);}  <- black
p {color: rgb(100%, 100%, 100%);}  <- white
p {color: rgba(50%, 10%, 60%, 1);} <- full opacity
p {color: rgba(31, 250, 118, 0);} <- full transparency
```

RGBA stands for **Red**, **Green**, **Blue**, and Alpha.
It's a color model in CSS that allows you to define colors (each between 0 and 255), and the alpha value (between 0 and 1), which represents the color's opacity.

New version:

```
p { color: rgb(100% 100% 100%); }
p { color: rgb(255 255 255 / .5); }
p { color: rgb(0 0 0 / 50%); }
```

Color

HSL/HSLA

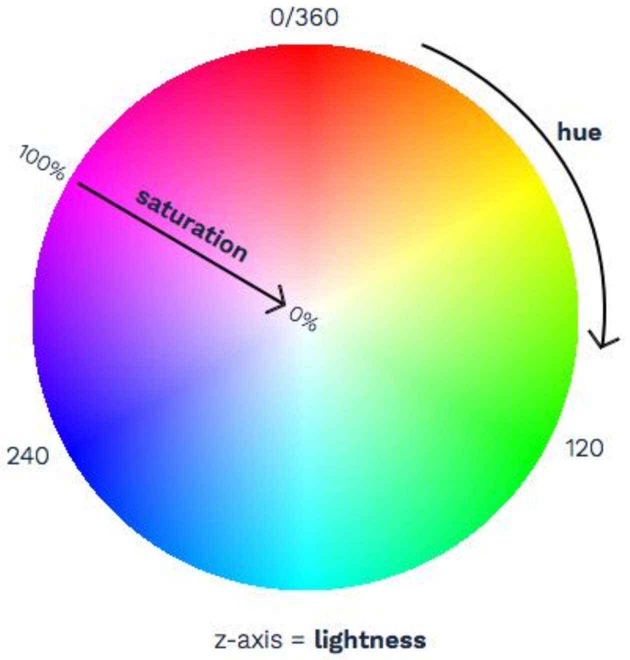
```
p {color: hsla(280,100%,50%,1);}
          hsl(hue, saturation, lightness)
```

It is a color model in CSS that describes colors in a way that is more intuitive for humans to understand compared to the traditional RGB model. **HSLA** is an extension of HSL that includes an alpha channel to specify the opacity of the color.

HSL and HSLA provide a more intuitive way to control colors, making it easier to create variations in hue, saturation, and lightness.

Using HSL and HSLA can make it easier to maintain a consistent color scheme across a web page or application.

- **Hue**
  - 0 – 359 (deg)
- **Saturation**
  - 0 – 100%
- **Lightness**
  - 0 – 100%
- **Alpha-transparency**
  - 0 – 1



Color	Name	Hex	RGB
🟦	aqua	#00FFFF	0,255,255
⬛	black	#000000	0,0,0
🟦	blue	#0000FF	0,0,255
🟪	fuchsia	#FF00FF	255,0,255
⬜	gray	#808080	128,128,128
🟩	green	#008000	0,128,0
🟩	lime	#00FF00	0,255,0
🟥	maroon	#800000	128,0,0
🟦	navy	#000080	0,0,128
🟫	olive	#808000	128,128,0
🟪	purple	#800080	128,0,128
🟥	red	#FF0000	255,0,0
⬜	silver	#C0C0C0	192,192,192
🟦	teal	#008080	0,128,128
⬜	white	#FFFFFF	255,255,255
🟨	yellow	#FFFF00	255,255,0

Colors keywords

CSS 1 introduced 16 color keywords, providing a simple way to specify standard colors. The CSS Color Module Level 3 added 131 new color keywords, significantly expanding the palette of colors available to web developers.

However, these are better used for personal purposes, such as in demos.

Keyword

currentColor

– Inherits the color value from the element or parent.

```
div {
  color: red;
  border: 5px solid currentcolor;
}
```

The border color will be red.

```
div {
  color: red;
  border: 5px solid currentcolor;
  color: yellow;
}
```

The border color will be yellow because of the cascade rule.

```
div {
  color: green;
}
div p {
  border: 5px solid currentcolor;
}
```

The border color will be green because the color is defined by the parent.

Useful for making the color of a vector icon match the text color.

Keyword transparent

It means full transparency.

In CSS3, the transparent keyword can be applied to any property that accepts a color value. This allows you to set transparency for various elements, including text, background, borders, etc.

```
div {
  color: transparent;
  border: 5px solid transparent;
}
```

The text will take up space on the page but will not be visible. Its outline will also not be visible.

This can be useful in various design scenarios, such as when you want to create interactive elements whose visibility changes based on user actions, or when you need to set transparency for background images or other elements to achieve a certain visual effect.

Other ways to set a fully transparent color:

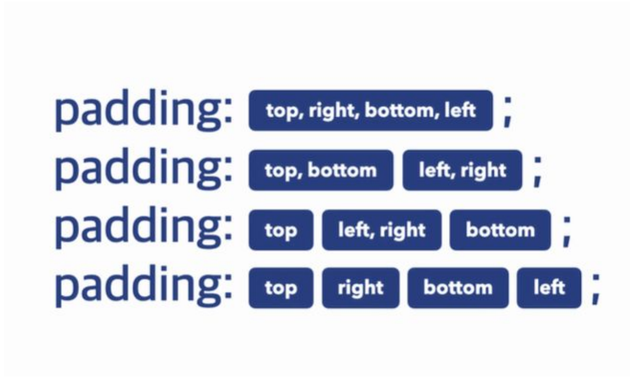
```
color: rgba(0, 255, 0, 0);
color: #00FFFFFF;
color: hsl(180, 100%, 50%);
```

`opacity: 0;` – the entire element will be transparent

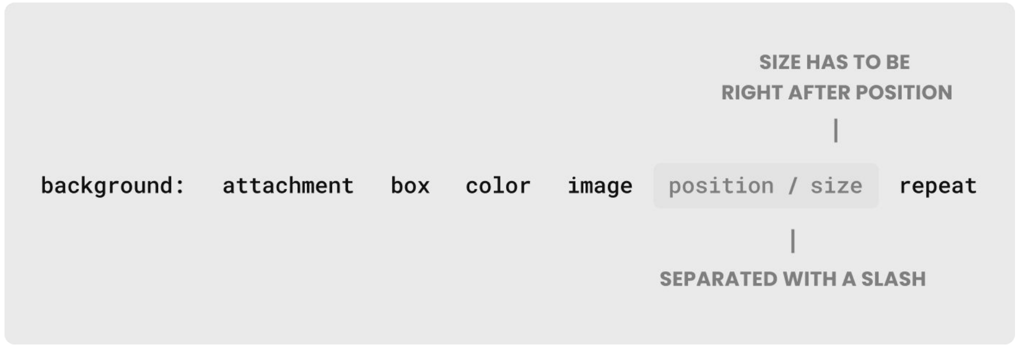
Types of

shortland properties

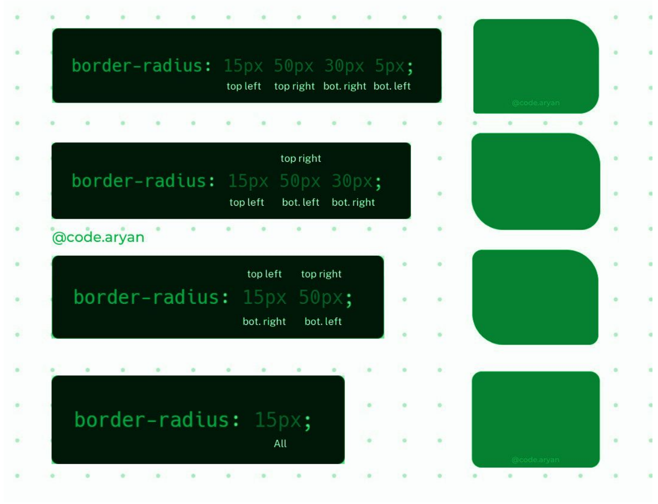
CSS shorthand properties allow developers to set multiple related CSS properties with a single declaration, simplifying the code and making it more concise and readable. Shorthand properties group together multiple properties that are commonly used together, reducing the amount of CSS needed and helping maintain consistency.



```
padding: top, right, bottom, left ;
padding: top, bottom  left, right ;
padding: top  left, right  bottom ;
padding: top  right  bottom  left ;
```



```
                                        SIZE HAS TO BE
                                     RIGHT AFTER POSITION
                                              |
background:  attachment  box  color  image  position / size  repeat
                                              |
                                   SEPARATED WITH A SLASH
```



```
border-radius: 15px 50px 30px 5px;
               top left  top right  bot. right  bot. left

                           top right
border-radius: 15px 50px 30px;
               top left  bot. left  bot. right

               top left  top right
border-radius: 15px 50px;
               bot. right  bot. left

border-radius: 15px;
               All
```

@code.aryan

Vendor Prefixes

WebKit `-webkit-`
(Safari, old versions of Chrome)

Mozilla Firefox `-moz-`

Opera `-o-`

Internet Explorer и **Microsoft Edge** `-ms-`
(old versions)

Use Autoprefixer [**Autoprefixer**]().

Vendor prefixes are one way browsers use to give us CSS developers access to new
not yet considered stable.

Before going on keep in mind this approach is declining in popularity though, in favo
experimental flags, which must be enabled explicitly in the user's browser.

Why? Because developers instead of considering vendor prefixes as a way to previe
they shipped them in production - something considered harmful by the CSS Workin

Mostly becau you add a flag and developers start using it in production, brov
bad position if they realise something must change. With flags, you can't ship a featu
can push all your visitors to enable that flag in their browser (just joking, don't try).

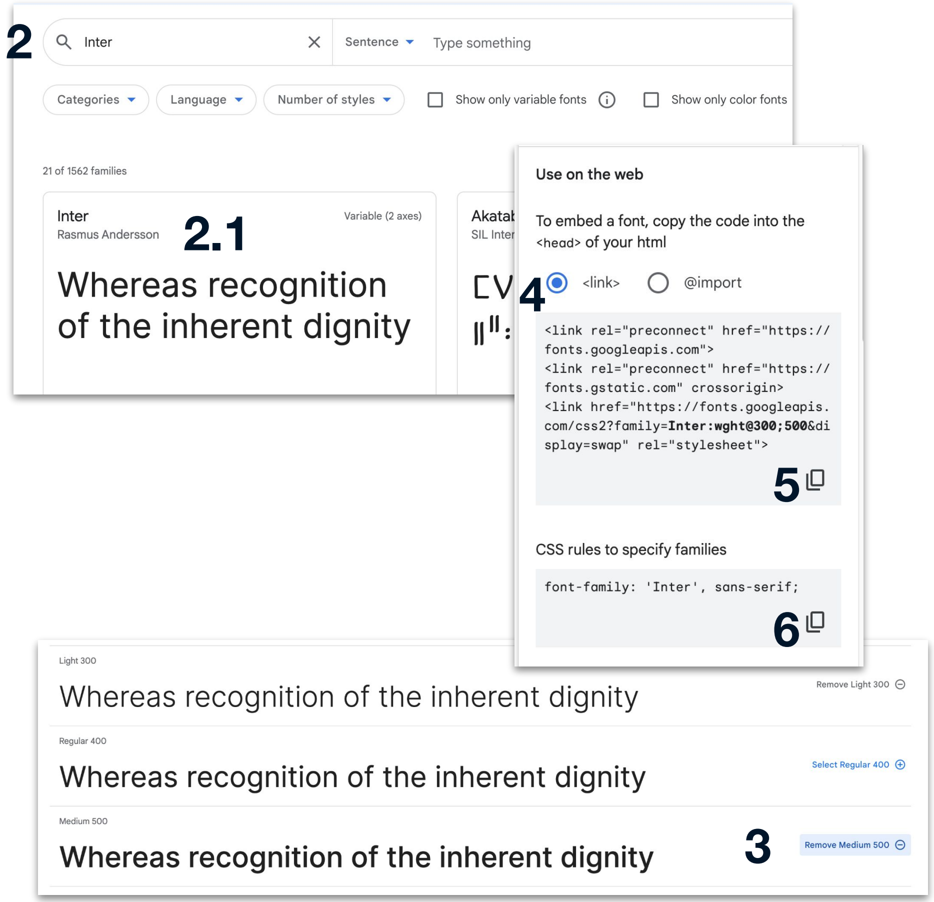
That said, let's see what vendor prefixes are.

Since Opera is Chromium-based and Edge will soon be too, -o- and -ms- will probab
out of fashion. But as we said, vendor prefixes as a whole are going out of fashion, to

Writing prefixes is hard, mostly because of uncertainty. Do you actually need a prefix
property? Several online resources are outdated, too, which makes it even harder to

Connecting a Font

1. Open **fonts.google.com**
2. **Search** for the desired font
3. Choose the desired styles
4. In the side panel, select the "**link**" option
5. Copy the **code and paste** it into the <head> tag in the HTML
6. Set the font for the page in the **CSS file**



To embed a font, copy the code into the <head> of your html

[✓ <link>] @import

```
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;500&display=swap" rel="stylesheet">
```

CSS rules to specify families

```
font-family: 'Inter', sans-serif;
```

font-family

Sets the font family that the element will use.

Why "family"? Because what we know as a font is actually composed of several sub-fonts.
which provide all the style (bold, italic, light..) we need.

You can set multiple values, so the second option will be used if the first cannot be used for some reason:

- if it's not found on the machine,
- or the network connection to download the font failed

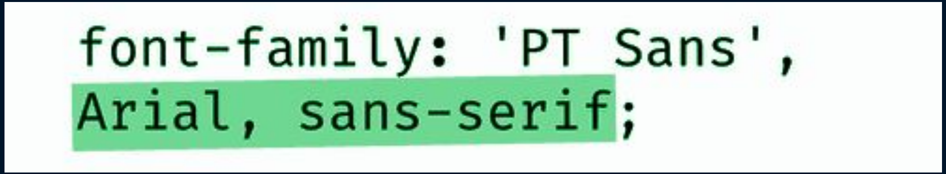
I used some specific fonts up to now, ones we call Web Safe Fonts, as they are pre-installed on different operating systems.

Example

```
body {
  font-family: “Intro”, Arial, sans-serif;
}
```

CSS

font-family



```
font-family: 'PT Sans',
Arial, sans-serif;
```

font family =

- A font from the layout
- A web-safe font
- Font type

This ensures the text looks as similar to the layout as possible, even if the preferred font isn't loaded.

Fonts types:

- Serif – Fonts with small lines at the end of characters
- Sans-serif - Fonts without those lines
- Cursive – Handwriting-like fonts, often with connected letters
- Monospace - Fonts where all characters have the same width
- Fantasy – Decorative fonts

Web-safe

Fonts

Sans-serif

- Arial
- Verdana
- Tahoma
- Trebuchet MS

Serif

- Times New Roman
- Georgia
- Garamond

- Courier New (**monospace**)
- Brush Script MT (**cursive**)

```
font-family: Inter, Arial, sans-serif;
```

```
font-family: Playfair Display, Georgia, serif;
```

Working with

Text

- ***font-style* –** allows you to apply an italic style to a font
- font-weight – sets the width of a font
- **font-size –** determine the size of fonts
- **line-height –** change the height of a line
- **color –** text color

Example

```
body {
  font-style: italic;
  font-weight: bold;
  font-size: 12px;
  line-height: 1.5;
  color: #000;
}
```

properties and values

font

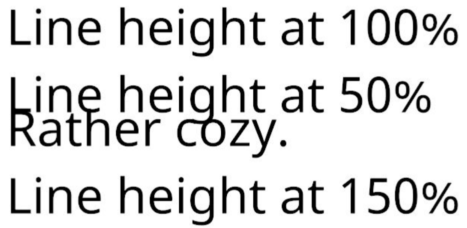
`font-style`	`normal`
	`italic` – cursive;	

`font-variant`	`SMALL-CAPS` – все маленькие буквы отображаются капителью `normal`	
`font-weight`	`100` – thin `200` – extra Light `300` – light `400` – **`normal`** `500` – medium `600` – semibold `700` – **`bold`** `800` – extra Bold `900` – heavy	**`lighter`** and **`bolder`** using when need to set a font weight thinner or thicker than that of its parent

properties and values

font

font-size	`font-size: 10px;`
	`font-size: 2em;`
	`font-size: 3.5rem;`	

line-height	`line-height: 3.5;`
	`line-height: 3em;`
	`line-height: 13px;`
		`line-height: 34%;`
		`line-height: normal;`
		
		`don't use` **`line-height < 1 or 100%`**
		
		Line height at 100%
		Line height at 50%
		Rather cozy.
		Line height at 150%

CSS Emmet
cheat sheet

Hex color	c	color: #000;
Rgb color	c:r	color: rgb(0, 0, 0);
Opacity	op	opacity:;
fonts	fw	font-weight:;
	fw:n	font-weight: normal;
	fw:b	font-weight: bold;
	fs	font-style: italic;
	fs:n	font-style: normal;
	fz	font-size:;
	ff	font-family:;
	ff:a	font-family: Arial, "Helvetica Neue", Helvetica, sans-serif;
	ff:t	font-family: "Times New Roman", Times, Baskerville, Georgia, serif;
line-height	lh	line-height: ;

`text-transform`

– transform the case of an element:

- **`Capitalize`** – uppercase the first letter of each word
- **`UPPERCASE`** – uppercase all the text
- **`lowercase`** – lowercase all the text
- **`none`** – disable transforming the text, used to avoid inheriting the property

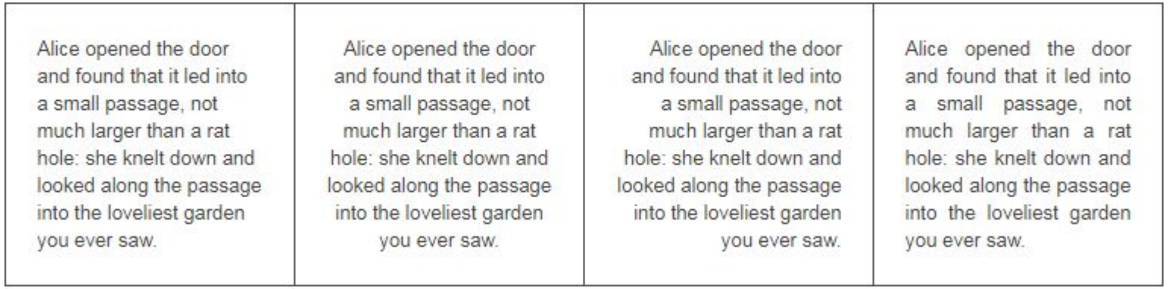
Example

```
body {
  text-transform: uppercase;
}
```

text-align

By default text align has the start value, meaning the text starts at the "start", origin 0, 0 of the box that contains it. This means top left in left-to-right languages, and top right in right-to-left languages.

- **`left`**
- **`right`**
- **`center`**
- **`justify`** (nice to have a consistent spacing at the line ends)



Alice opened the door
and found that it led into
a small passage, not
much larger than a rat
hole: she knelt down and
looked along the passage
into the loveliest garden
you ever saw.

left

Alice opened the door
and found that it led into
a small passage, not
much larger than a rat
hole: she knelt down and
looked along the passage
into the loveliest garden
you ever saw.

center

Alice opened the door
and found that it led into
a small passage, not
much larger than a rat
hole: she knelt down and
looked along the passage
into the loveliest garden
you ever saw.

right

Alice opened the door
and found that it led into
a small passage, not
much larger than a rat
hole: she knelt down and
looked along the passage
into the loveliest garden
you ever saw.

justify

Example

```
p {
  text-align: right;
}
```

text-decoration

– add decorations to the text

- **`text-decoration-line:`**
  - Overline
  - Underline
  - ~~Line-through~~
  - None
  - Underline Overline
- **`text-decoration-color`**
- **`text-decoration-style:`**
  - solid
  - double
  - dotted
  - dashed
  - dot dash
  - dot dot dash
  - wave

Example

```
p {
  text-decoration: underline;
  text-decoration: underline dashed yellow;
}

p {
  text-decoration-line: underline;
  text-decoration-color: yellow;
  text-decoration-style: dashed;
}
```

CSS Emmet

cheat sheet

text-transform	tt	text-transform:uppercase;
	tt:n	text-transform:none;
	tt:c	text-transform:capitalize;
	tt:u	text-transform:uppercase;
	tt:l	text-transform:lowercase;
text-align	ta	text-align: left;
	ta:c	text-align: center;
	ta:r	text-align: right;
	ta:j	text-align:justify;
text-decoration	td:n	text-decoration: none;
	td:u	text-decoration: underline;
	tt	text-transform: uppercase;
	td:o	text-decoration:overline;
	td:l	text-decoration:line-through;

vertical-align

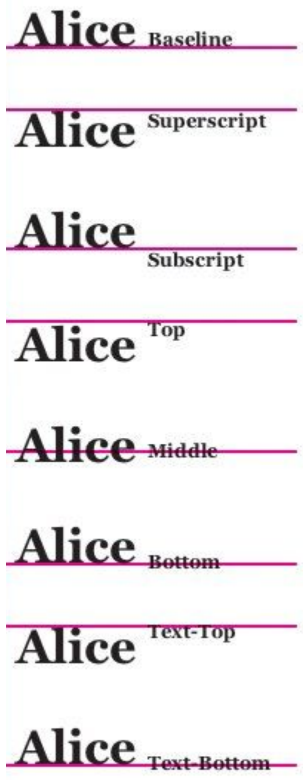
Example

```
p {
  vertical-align: middle;
}
```

Determines how inline elements are vertically aligned.

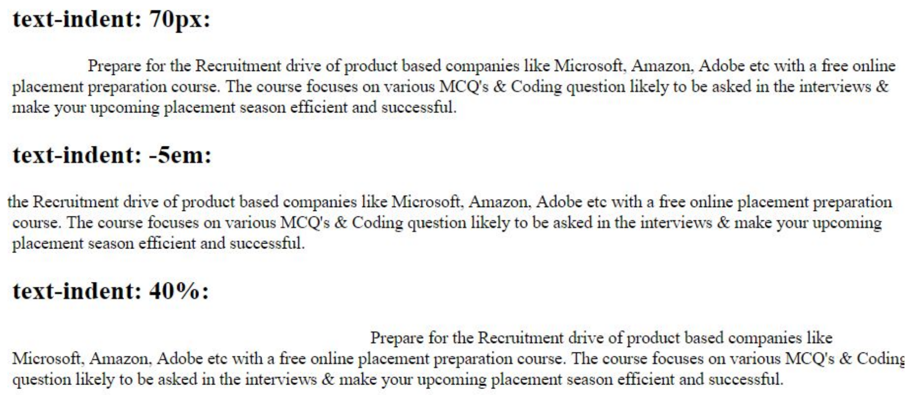
Those are used to align the text in a position higher or lower (using negative values) than the baseline of the parent element.

- **`baseline`** – aligns the baseline to the baseline of the parent element
- **`sub`** – makes an element subscripted, simulating the sub tag result
- **`super`** – makes an element superscripted, simulating the sup tag result
- **`top`** – align the top of the element to the top of the line
- **`text-top`** – align the top of the element to the top of the parent element font
- **`middle`** – align the middle of the element to the middle of the line of the parent
- **`bottom`** – align the bottom of the element to the bottom of the line
- **`text-bottom`** – align the bottom of the element to the bottom of the parent element font



text-indent

Indent the first line of a paragraph by a set length, or a percentage of the paragraph width.



text-indent: 70px:

Prepare for the Recruitment drive of product based companies like Microsoft, Amazon, Adobe etc with a free online placement preparation course. The course focuses on various MCQ's & Coding question likely to be asked in the interviews & make your upcoming placement season efficient and successful.

text-indent: -5em:

the Recruitment drive of product based companies like Microsoft, Amazon, Adobe etc with a free online placement preparation course. The course focuses on various MCQ's & Coding question likely to be asked in the interviews & make your upcoming placement season efficient and successful.

text-indent: 40%:

Prepare for the Recruitment drive of product based companies like Microsoft, Amazon, Adobe etc with a free online placement preparation course. The course focuses on various MCQ's & Coding question likely to be asked in the interviews & make your upcoming placement season efficient and successful.

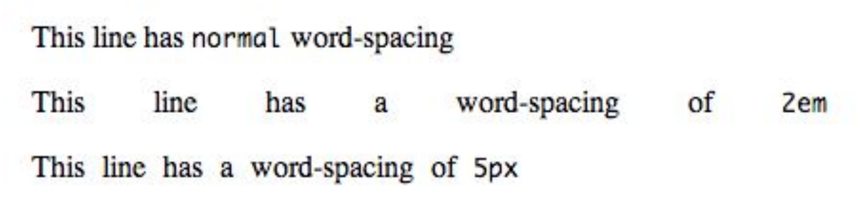
Example

```css
p {
  text-indent: -10px;
}
```

word-spacing

Modifies the spacing between each **word**.

You can use the **`normal`** keyword, to reset inherited values, or use a length value.



This line has normal word-spacing

This line has a word-spacing of 2em

This line has a word-spacing of 5px

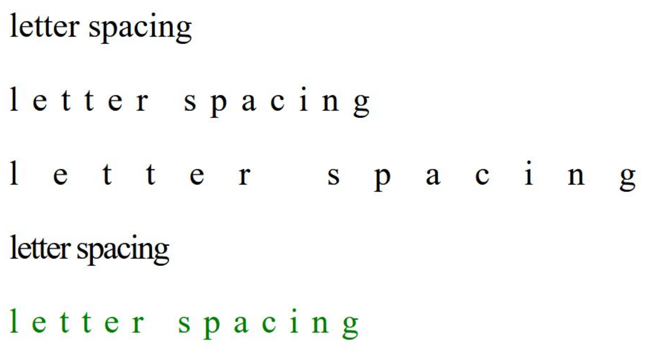
Example

```
p {
  word-spacing: 2px;
}

span {
  word-spacing: -0.2em;
}
```

`letter-spacing`

Modifies the spacing between each **letter**.

You can use the **`normal`** keyword, to reset inherited values, or use a length value.



letter spacing

l e t t e r   s p a c i n g

l   e   t   t   e   r   s   p   a   c   i   n   g

letter spacing

l e t t e r   s p a c i n g

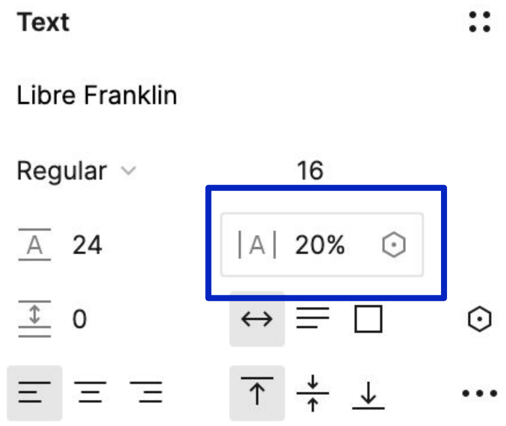
Example

```
p {
  letter-spacing: 0.2px;
}

span {
  letter-spacing: -0.2em;
}
```

letter-spacing

If you use a length value, use in px or em, not percentage like in Figma.

100% = 1em / 20% = 0.2em



Example

```
span {
  letter-spacing: 0.2em;
}
```

CSS Emmet

cheat sheet

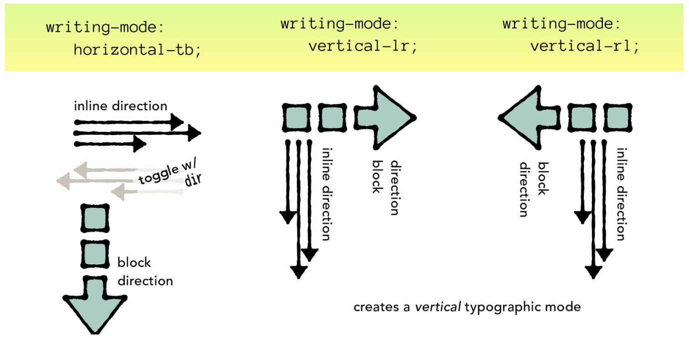
vertical-align	va	vertical-align:top;
	va:sup	vertical-align:super;
	va:t	vertical-align:text-top;
	va:tt	text-transform:uppercase;
	va:m	vertical-align:middle;
	va:bl	vertical-align:baseline;
	va:b	vertical-align:bottom;
	va:tb	vertical-align:text-bottom;
	va:sub	vertical-align:sub;
text-indent	ti	text-indent:;
	ti:-	text-indent:-9999px;
word-spacing	wos	word-spacing:;
letter-spacing	lts	letter-spacing:;
	lts-n	letter-spacing:normal;

`writing-mode`

Defines whether lines of text are laid out horizontally or vertically, and the direction in which blocks progress.

The values you can use are

- **`horizontal-tb`** (default)
- **`vertical-rl`** content is laid out vertically. New lines are put on the left of the previous
- **`vertical-lr`** content is laid out vertically. New lines are put on the right of the previous


writing-mode: horizontal-tb;
writing-mode: vertical-lr;
writing-mode: vertical-rl;
inline direction
toggle w/ dir
block direction
inline direction
block direction
block direction
inline direction
creates a *vertical* typographic mode

Example

```
p {
  writing-mode: vertical-rl;
}
```

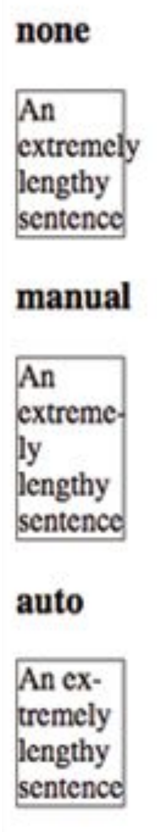
hyphens

Determines if hyphens should be automatically added when going to a new line.

- `none` (default)
- `manual` – only add an hyphen when there is already a visible hyphen or a hidden hyphen (a special character)
- `auto` – add hyphens when determined the text can have a hyphen.

Example

```
p {
  hyphens: manual;
}
```

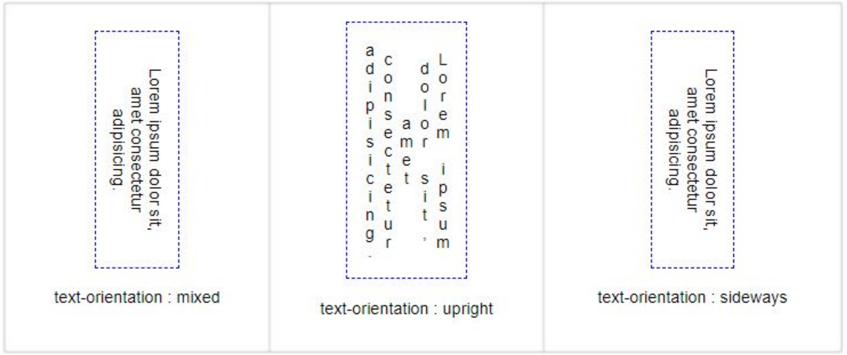


none
An extremely lengthy sentence
manual
An extreme-ly lengthy sentence
auto
An ex-tremely lengthy sentence

`text-orientation`

When **`writing-mode`** is in a vertical mode, determines the orientation of the text.

- **`mixed` –** is the default, and if a language is vertical (like Japanese) it preserves that orientation, while rotating text written in western languages
- **`upright` –** makes all text be vertically oriented
- **`sideways` –** makes all text horizontally oriented


Lorem ipsum dolor sit, amet consectetur adipisicing.
text-orientation : mixed
Lorem ipsum dolor sit, amet consectetur adipisicing.
text-orientation : upright
Lorem ipsum dolor sit, amet consectetur adipisicing.
text-orientation : sideways

Example

```
p {
  text-orientation: mixed;
}
```

CSS Emmet

cheat sheet

writing-mode	wm	writing-mode:lr-tb;
	wm:btl	writing-mode:bt-lr;
	wm:btr	writing-mode:bt-rl;
	wm:lrb	writing-mode:lr-bt;
	wm:lrt	writing-mode:lr-tb;
	wm:rlb	writing-mode:rl-bt;
	wm:rlt	writing-mode:rl-tb;
	wm:tbl	writing-mode:tb-lr;
	wm:tbr	writing-mode:tb-rl;

`white-space`

Sets how CSS handles the white space, new lines and tabs inside an element. Valid values that collapse white space are:

- **`normal`** – collapses white space. Adds new lines when necessary as the text reaches the container end
- **`nowrap`** – collapses white space. Does not add a new line when the text reaches the end of the container, and suppresses any line break added to the text
- **`pre-line`** – collapses white space. Adds new lines when necessary as the text reaches the container end

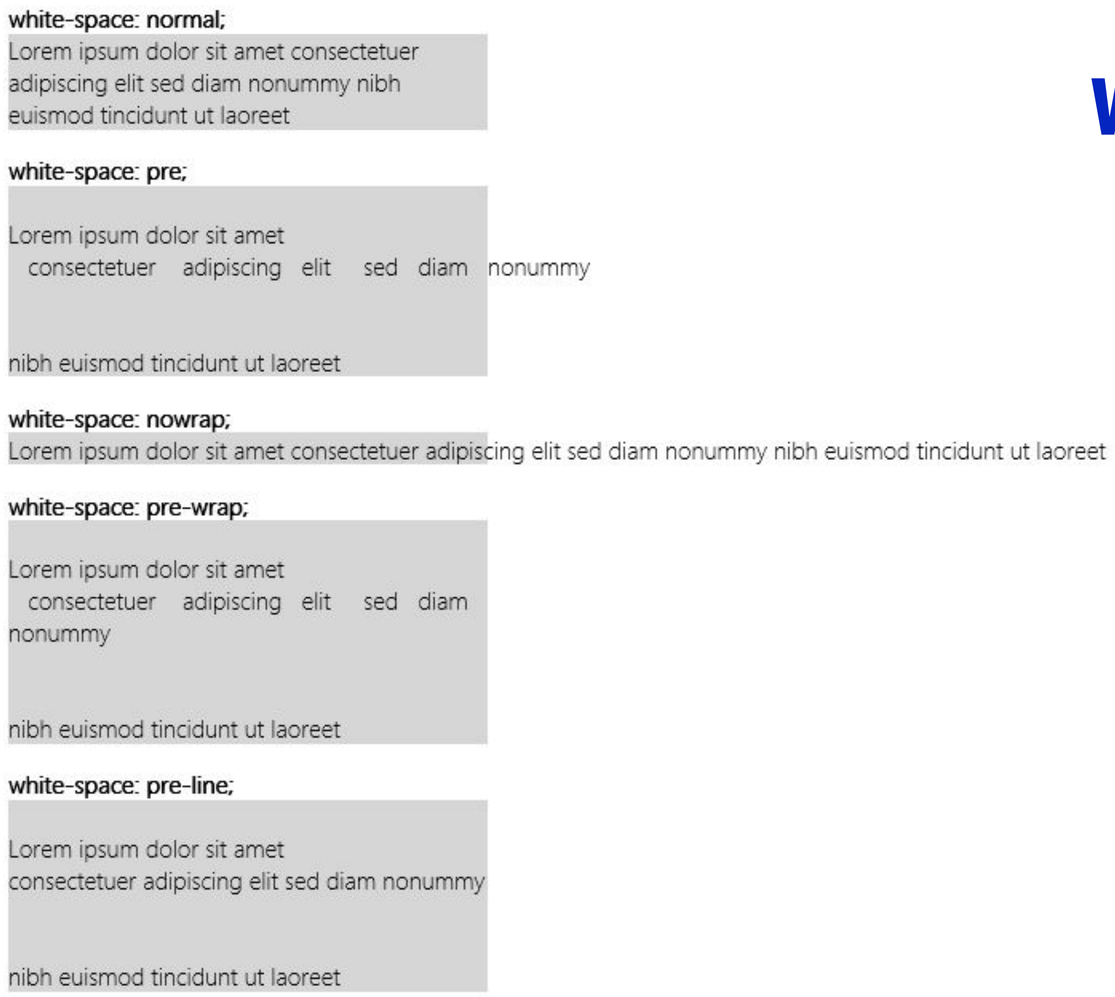
Valid values that preserve white space are:

- **`pre`** – preserves white space. Does not add a new line when the text reaches
- the end of the container, but preserves line break added to the text
- **`pre-wrap`** – preserves white space. Adds new lines when necessary
- as the text reaches the container end

Example

```
p {
  white-space: normal;
}
```

white-space



white-space: normal;

```
Lorem ipsum dolor sit amet consectetuer
adipiscing elit sed diam nonummy nibh
euismod tincidunt ut laoreet
```

white-space: pre;

```

Lorem ipsum dolor sit amet
   consectetuer    adipiscing   elit    sed   diam   nonummy



nibh euismod tincidunt ut laoreet
```

white-space: nowrap;

```
Lorem ipsum dolor sit amet consectetuer adipiscing elit sed diam nonummy nibh euismod tincidunt ut laoreet
```

white-space: pre-wrap;

```

Lorem ipsum dolor sit amet
   consectetuer    adipiscing   elit    sed   diam
nonummy



nibh euismod tincidunt ut laoreet
```

white-space: pre-line;

```

Lorem ipsum dolor sit amet
consectetuer adipiscing elit sed diam nonummy



nibh euismod tincidunt ut laoreet
```

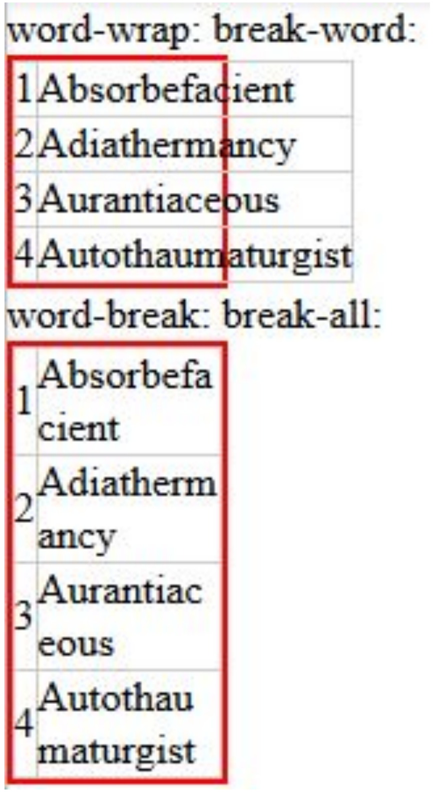
word-break

– property specifies how words should break when reaching the end of a line. It deals with the breaking of words themselves rather than preventing overflow.

- **`normal` –** (default) means the text is only broken between words, not inside a word
- **`break-all`** – the browser can break a word (but no hyphens are added)
- **`keep-all` –** suppress soft wrapping. Mostly used for CJK (Chinese/Japanese/Korean) text.

Example

```
p {
  word-break: right;
}
```


word-wrap: break-word:

1	Absorbefacient
2	Adiathermancy
3	Aurantiaceous
4	Autothaumaturgist

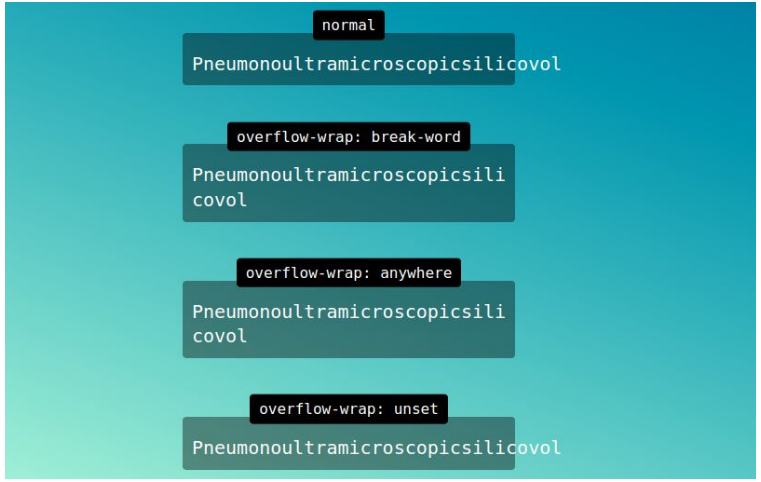
word-break: break-all:

1	Absorbefa cient
2	Adiatherm ancy
3	Aurantiac eous
4	Autothau maturgist

overflow-wrap

– property is used to prevent long words from overflowing their container. It allows the browser to break the word at a suitable point to fit within the container.

- **`normal`**
- **`break-word`** - to break it at the exact length of the line
- **`anywhere`** - if the browser sees there's a soft wrap opportunity somewhere earlier. No hyphens are added, in any case.



```
normal
Pneumonoultramicroscopicsilicovol

overflow-wrap: break-word
Pneumonoultramicroscopicsili
covol

overflow-wrap: anywhere
Pneumonoultramicroscopicsili
covol

overflow-wrap: unset
Pneumonoultramicroscopicsilicovol
```

Example

```
p {
  overflow-wrap: break-word;
}

p {
  overflow-wrap: anywhere;
}
```

`overflow-wrap vs word-break`

Behavior:

- **`overflow-wrap: break-word`:** Attempts to break words only when necessary to prevent overflow. It tries to break the word at a convenient point.
- **`word-break: break-all`:** Forces the word to break at any character, regardless of whether it fits within the container or not.

Use Cases:

- **`overflow-wrap: break-word`**:
    - Use when you want to break long words to fit within the container without overflowing.
    - Example: In responsive design, to ensure long words (like URLs or long technical terms) do not overflow their containers.
- **`word-break: break-all`**:
    - Use when you want to break words at any character to ensure they fit within the container.
    - Example: For languages with long continuous text without spaces (e.g., certain East Asian languages), or when dealing with very narrow columns of text.

Language Sensitivity:

- **`overflow-wrap`** is generally used for languages that use spaces between words.
- **`word-break: break-all`** is particularly useful for languages like Chinese, Japanese, and Korean, where breaking at any character is more common.

CSS Emmet

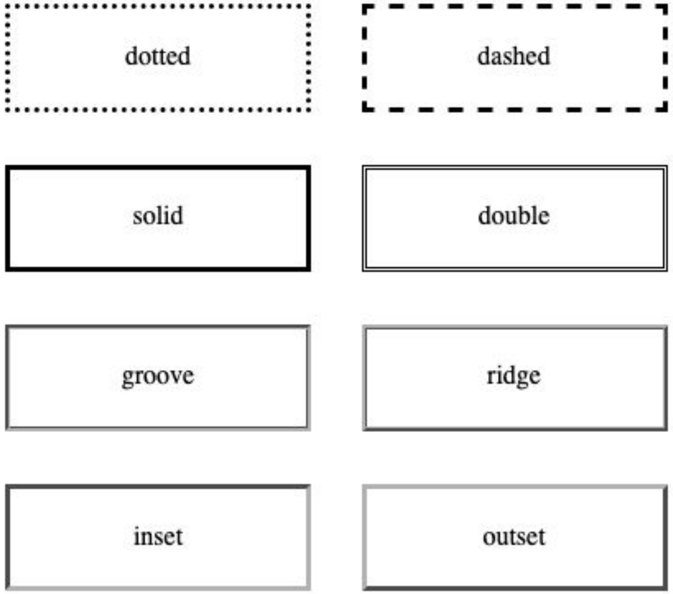
cheat sheet

white-space	whs	white-space:;
	whs:n	white-space:normal;
	whs:p	white-space:pre;
	whs:nw	white-space:nowrap;
	whs:pw	white-space:pre-wrap;
	whs:pl	white-space:pre-line;
	whsc	white-space-collapse:;
word-break	wob	word-break:;
	wob:n	word-break:normal;
	wob:k	word-break:keep-all;
	wob:ba	word-break:break-all;

border-style

lets you choose the style of the border.

- dotted
- dashed
- solid
- double
- groove
- ridge
- inset
- outset
- none
- hidden



Example

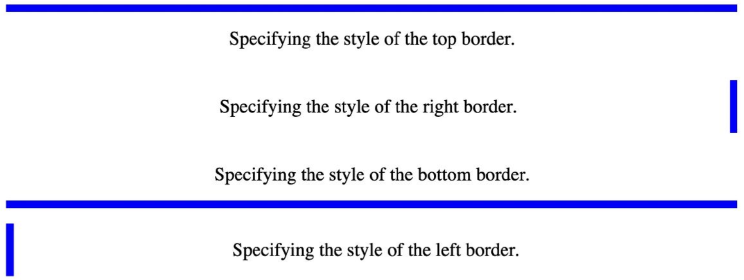
```
p {
  border-style: dashed;
}
```

border-style

You can set a different style for each edge using the properties

- **`border-top-style`**
- **`border-right-style`**
- **`border-bottom-style`**
- **`border-left-style`**

or you can use **`border-style`** with multiple values to define them



Specifying the style of the top border.

Specifying the style of the right border.

Specifying the style of the bottom border.

Specifying the style of the left border.

Example

```
p {
  border-style: solid dotted solid dotted;
}
```

border-width

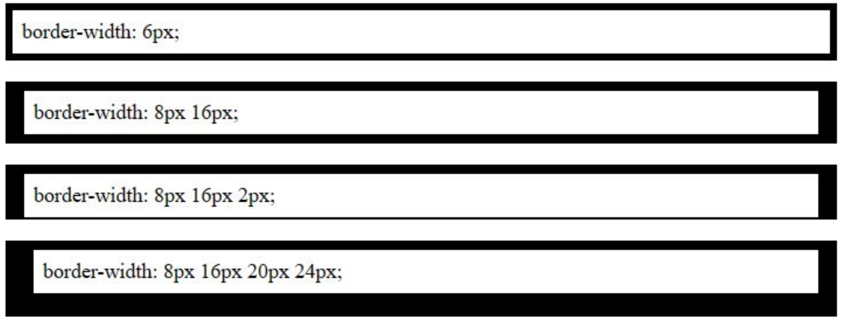
– is used to set the width of the border.

- `thin`
- `medium` (the default value)
- `thick`

or express a value in **pixels, em or rem** or any other valid length value.

You can set the width of each edge (**top-right-bottom-left)** separately by using 4 values:

- `border-top-width`
- `border-right-width`
- `border-bottom-width`
- `border-left-width`



```
border-width: 6px;
border-width: 8px 16px;
border-width: 8px 16px 2px;
border-width: 8px 16px 20px 24px;
```

Example

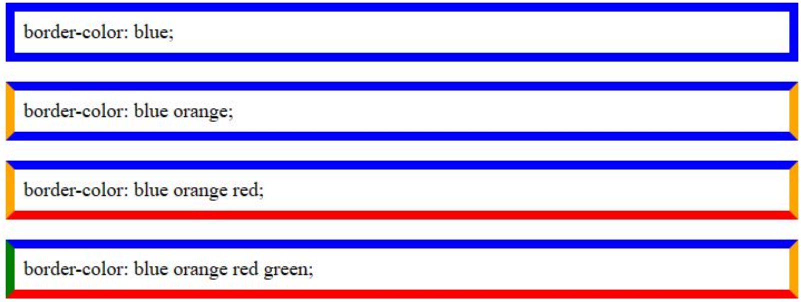
```
p {
  border-width: 2px;
}

p {
  border-width: 2px 1px 2px 1px;
}
```

border-color

– is used to set the color of the border.

If you don't set a color, the border by default is colored using the color of the text in the element.

You can set the color of each edge (top-right-bottom-left) separately by using 4 values.



border-color: blue;

border-color: blue orange;

border-color: blue orange red;

border-color: blue orange red green;

Example

```
p {
  border-color: yellow;
}

p {
  border-color: black red yellow blue;
}
```

`border shorthand`

Those 3 properties mentioned:

- **`border-width`**
- **`border-style`**
- **`border-color`**

can be set using the shorthand property border .

You can also use the edge-specific properties:

- **`border-top`**
- **`border-right`**
- **`border-bottom`**
- **`border-left`**

Example

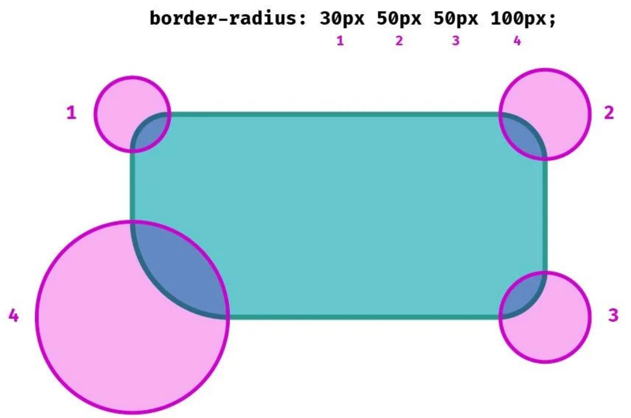
```
p {
  border: 2px black solid;
}

p {
  border-left: 2px black solid;
  border-right: 3px red dashed;
}
```

border-radius

is used to set rounded corners to the border. You need to pass a value that will be used as the radius of the circle that will be used to round the border.

You can also use the edge-specific properties:

- **`border-top-left-radius`**
- **`border-top-right-radius`**
- **`border-bottom-left-radius`**
- **`border-bottom-right-radius`**



Example

```
p {
  border-radius: 3px;
}
```

CSS Emmet
cheat sheet

borders	bd	border:;
	bd+	border: 1px solid #000;
	bd:n	border: none;
border-style	bs:n	border-style:none;
	bds:h	border-style:hidden;
	bds:dt	border-style:dotted;
	bds:ds	border-style:dashed;
	bds:s	border-style:solid;
	bds:db	border-style:double;
	bds:w	border-style:wave;
border-width	bdw	border-width:;
border-color	bdc	border-color:#000;
	bdc:t	border-color:transparent;
border-radius	bdrs	border-radius:;
	bdtrrs	border-top-right-radius:;
	bdtlrs	border-top-left-radius:;
	bdbrrs	border-bottom-right-radius:;
	bdblrs	border-bottom-left-radius:;

`list-style-type`

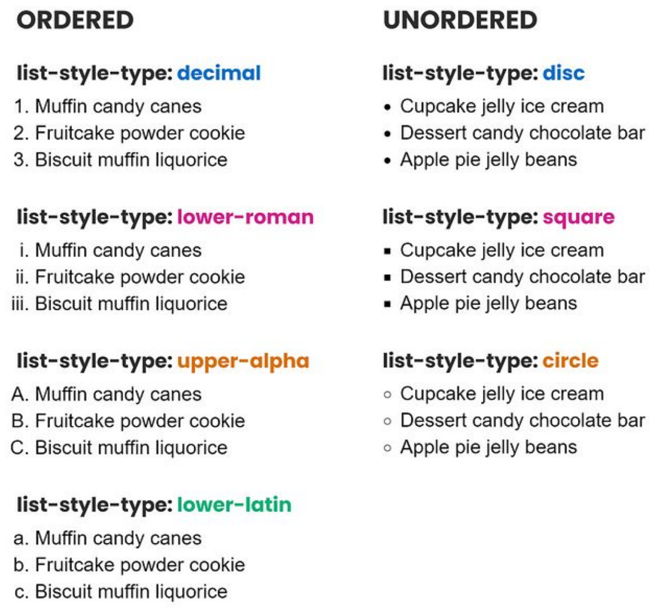
– is used to set a predefined marker to be used by the list:

We have lots of possible values, which you can see here https://developer.mozilla.org/en-US/docs/Web/CSS/list-style-type with examples of their appearance.

Some of the most popular ones are:

- `disc`
- `circle`
- `square`
- `none`

```
list-style-type: "🐣";
```



ORDERED

list-style-type: decimal

1. Muffin candy canes
2. Fruitcake powder cookie
3. Biscuit muffin liquorice

list-style-type: lower-roman

i. Muffin candy canes
ii. Fruitcake powder cookie
iii. Biscuit muffin liquorice

list-style-type: upper-alpha

A. Muffin candy canes
B. Fruitcake powder cookie
C. Biscuit muffin liquorice

list-style-type: lower-latin

a. Muffin candy canes
b. Fruitcake powder cookie
c. Biscuit muffin liquorice

UNORDERED

list-style-type: disc

• Cupcake jelly ice cream
• Dessert candy chocolate bar
• Apple pie jelly beans

list-style-type: square

▪ Cupcake jelly ice cream
▪ Dessert candy chocolate bar
▪ Apple pie jelly beans

list-style-type: circle

◦ Cupcake jelly ice cream
◦ Dessert candy chocolate bar
◦ Apple pie jelly beans

Example

```
li {
  list-style-type: square;
}
```

`list-style`

`list-style-image`

– is used to use a custom marker when a predefined marker is not appropriate.

`list-style-position`

– lets you add the marker:

- **`outside`** (the default) or
- **`inside`**

of the list content, in the flow of the page rather than outside of it.

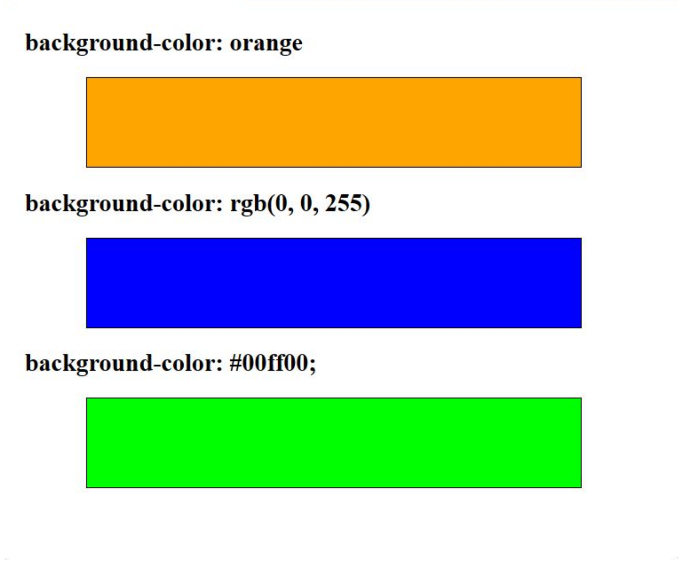
The **`list-style`** shorthand property lets us specify all those properties in the same line.

Example

```
li {
  list-style-image: url(list-image.png);
}

li {
  list-style-position: inside;
}

li {
  list-style: url(list-image.png) inside;
}
```

background-color

– accepts a color value, which can be one of the color keywords, or an rgb, a hex color and etc.


background-color: orange

background-color: rgb(0, 0, 255)

background-color: #00ff00;

Example

```
div {
  background-color: #333;
}
```

CSS Emmet

cheat sheet

list-style	lis	list-style:;
	lis:n	list-style: none;
list-style-position	lisp	list-style-position:;
	lisp:i	list-style-position:inside;
	lisp:o	list-style-position:outside;
list-style-type	list	list-style-type:;
	list:n	list-style-type:none;
	list:d	list-style-type:disc;
	list:c	list-style-type:circle;
	list:s	list-style-type:square;
	list:dc	list-style-type:decimal;
background-color	bgc	background-color:#fff;
	bgc:t	background-color:transparent;

outline

– is a line that goes around the element, outside of the border. In contrast to border, outlines do not take any space in the box model. So adding an outline to an element does not affect the position of the element or other elements.

In addition, outlines can be non-rectangular in some browsers. This can happen if outline is applied on a span element that has text with different font-size properties inside it. Unlike borders, outlines cannot have rounded corners.

The essential parts of outline are:

- `outline-color`
- `outline-style`
- `outline-width`

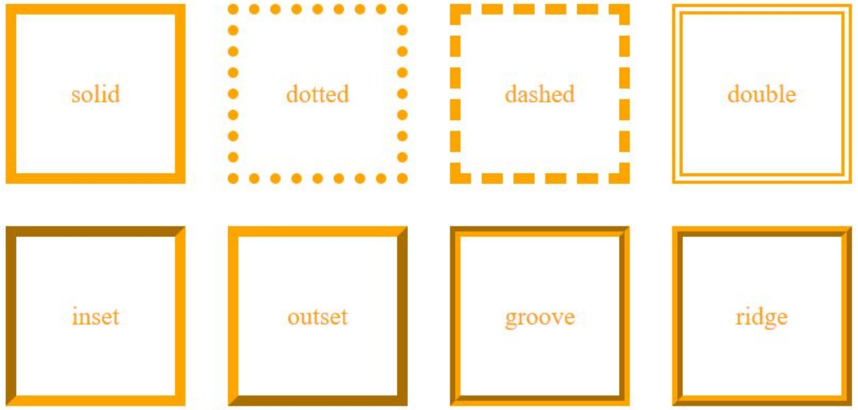
Example

```
p {
  outline: 1px solid blue;
}

p {
  outline-color:blue;
  outline-width: 3px;
}
```

`outline-style`

mostly like for **`border-style`**

- `dotted`
- `dashed`
- `solid`
- `double`
- `groove`
- `ridge`
- `inset`
- `outset`
- `none`
- `hidden`



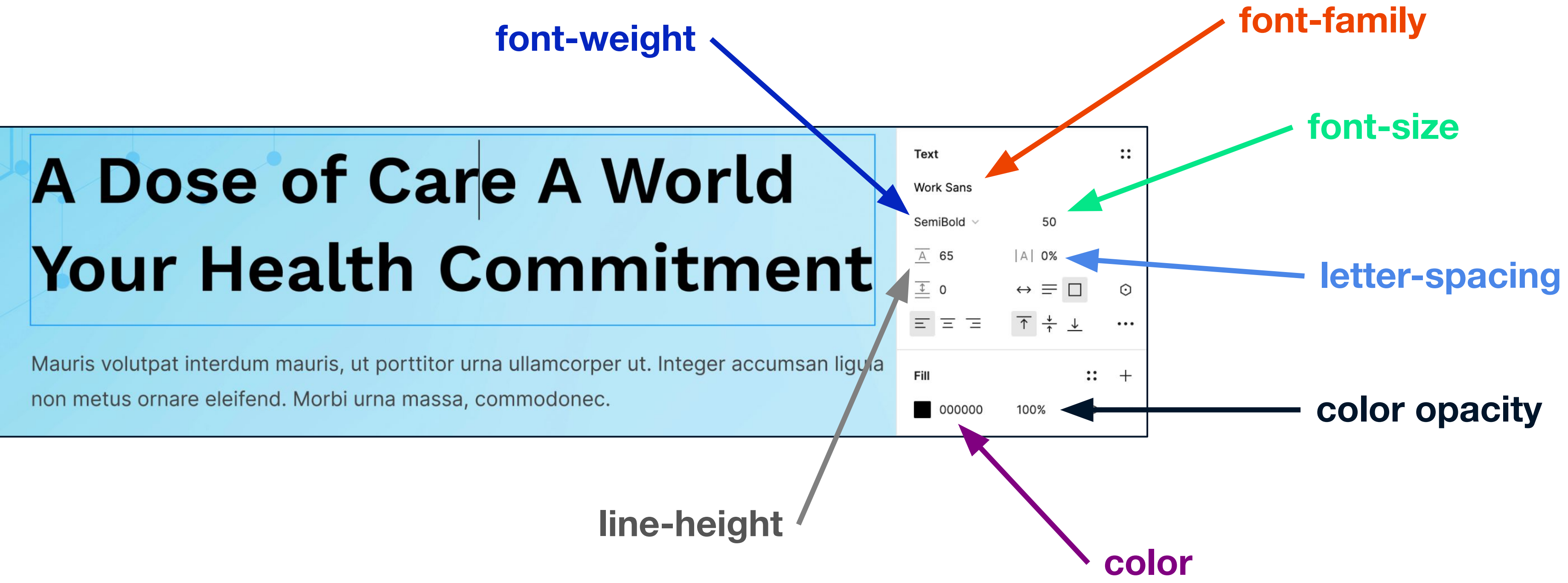
Example

```
.p {
  outline-style: dotted;
}
```

CSS Emmet
cheat sheet

outline	ol	outline:;
	ol:n	outline:none;
outline-width	olw	outline-width:;
outline-style	ols	outline-style:;
	ols:n	outline-style:none;
	ols:dt	outline-style:dotted;
	ols:ds	outline-style:dashed;
	ols:s	outline-style:solid;
	ols:db	outline-style:double;
	ols:g	outline-style:groove;
	ols:r	outline-style:ridge;
	ols:i	outline-style:inset;
	ols:o	outline-style:outset;
outline-color	olc	outline-color:#000;
	olc:i	outline-color:invert;

styles in Figma


A Dose of Care A World Your Health Commitment

Mauris volutpat interdum mauris, ut porttitor urna ullamcorper ut. Integer accumsan ligula non metus ornare eleifend. Morbi urna massa, commodonec.

Text

Work Sans

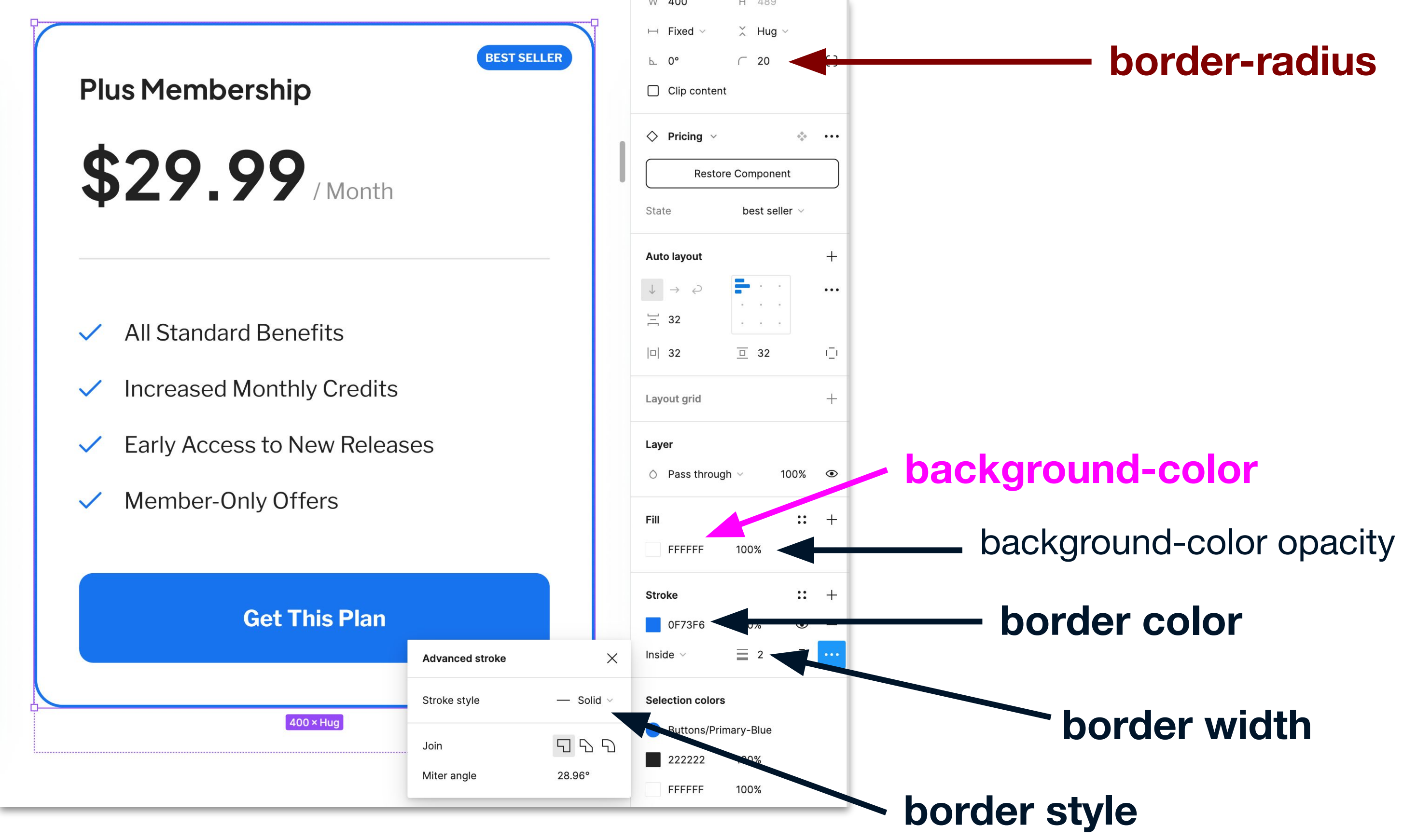
SemiBold 50

65 0%

0

Fill

000000 100%



BEST SELLER

Plus Membership

$29.99 / Month

✓ All Standard Benefits

✓ Increased Monthly Credits

✓ Early Access to New Releases

✓ Member-Only Offers

Get This Plan

400 × Hug

W 400 H 489

Fixed Hug

0° 20

Clip content

Pricing

Restore Component

State best seller

Auto layout

32

32 32

Layout grid

Layer

Pass through 100%

Fill

FFFFFF 100%

Stroke

0F73F6

Inside 2

Advanced stroke

Stroke style Solid

Join

Miter angle 28.96°

Selection colors

Buttons/Primary-Blue

222222 100%

FFFFFF 100%

styles in Figma