CSS

CSS is the language we use to style an HTML document.

CSS describes how HTML tags should be displayed

Table of Contents

Backgrounds

Border

Box Model

Font

Text

Column

Colors

Table

Speech

List & Markers

Animations

Transitions

UI

Pseudo-Class

Pseudo-Tag

Absolute Measurement

Relative Measurement

Angles

Time

Frequency

Colors

Selector Types

Outline

3D / 2D Transform

Generated Content

Line Box

Hyperlink

Positioning

Ruby

Paged Media

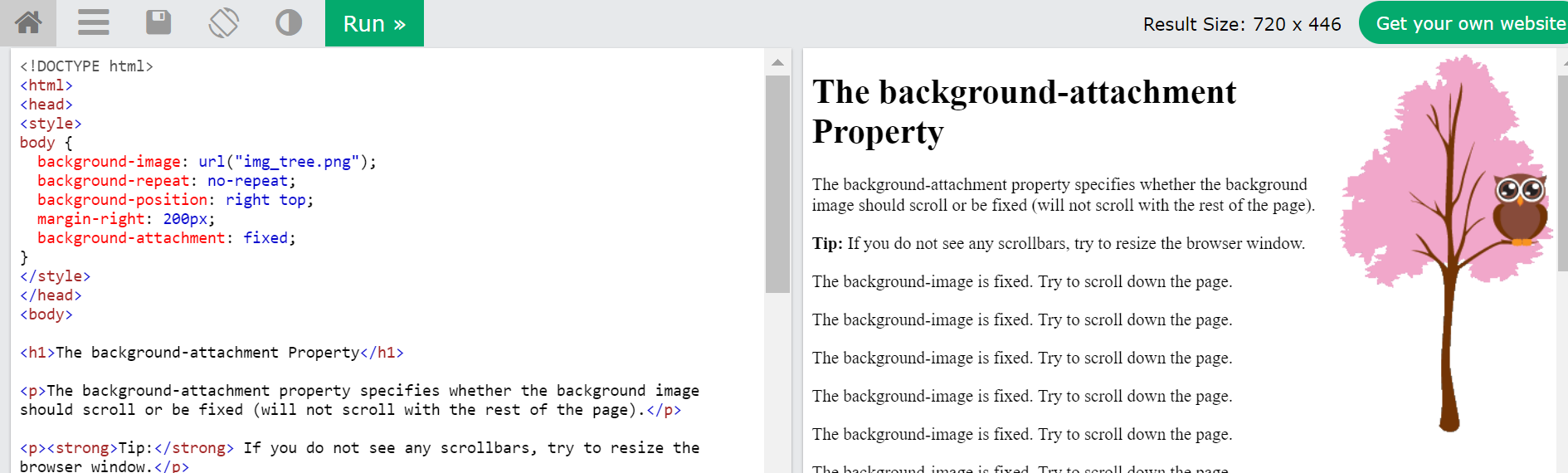
CSS selectors:

|  |  |  |
| --- | --- | --- |
| **Selector** | **Example** | **selects** |
| [.*class*](https://www.w3schools.com/cssref/sel_class.asp) | .intro | all tags with class="intro" |
| .class1.class2 | .name1.name2 | all tags with both name1 and name2 set within its class attribute |
| .class1 .class2 | .name1 .name2 | all tags with name2 that is a descendant of an tag with name1 |
| [#*id*](https://www.w3schools.com/cssref/sel_id.asp) | #firstname | the tag with id="firstname" |
| [\*](https://www.w3schools.com/cssref/sel_all.asp) | \* | all tags |
| [*tag*](https://www.w3schools.com/cssref/sel_element.asp) | p | all <p> tags |
| [*tag.class*](https://www.w3schools.com/cssref/sel_element_class.asp) | p.intro | all <p> tags with class="intro" |
| [*tag,tag*](https://www.w3schools.com/cssref/sel_element_comma.asp) | div, p | all <div> tags and all <p> tags |
| [*tag* *tag*](https://www.w3schools.com/cssref/sel_element_element.asp) | div p | all <p> tags inside <div> tags |
| [*tag*>*tag*](https://www.w3schools.com/cssref/sel_element_gt.asp) | div > p | all <p> tags where the parent is a <div> tag |
| [*tag*+*tag*](https://www.w3schools.com/cssref/sel_element_pluss.asp) | div + p | the first <p> tag that is placed immediately after <div> tags |
| [*tag1*~*tag2*](https://www.w3schools.com/cssref/sel_gen_sibling.asp) | p ~ ul | every <ul> tag that is preceded by a <p> tag |
| [[*attribute*]](https://www.w3schools.com/cssref/sel_attribute.asp) | [target] | all tags with a target attribute |
| [[*attribute*=*value*]](https://www.w3schools.com/cssref/sel_attribute_value.asp) | [target=\_blank] | all tags with target="\_blank" |
| [[*attribute*~=*value*]](https://www.w3schools.com/cssref/sel_attribute_value_contains.asp) | [title~=flower] | all tags with a title attribute containing the word "flower" |
| [[*attribute*|=*value*]](https://www.w3schools.com/cssref/sel_attribute_value_lang.asp) | [lang|=en] | all tags with a lang attribute value starting with "en" |
| [[*attribute*^=*value*]](https://www.w3schools.com/cssref/sel_attr_begin.asp) | a[href^="https"] | every <a> tag whose href attribute value begins with "https" |
| [[*attribute*$=*value*]](https://www.w3schools.com/cssref/sel_attr_end.asp) | a[href$=".pdf"] | every <a> tag whose href attribute value ends with ".pdf" |
| [[*attribute*\*=*value*]](https://www.w3schools.com/cssref/sel_attr_contain.asp) | a[href\*="w3schools"] | every <a> tag whose href attribute value contains the substring "w3schools" |
| [:active](https://www.w3schools.com/cssref/sel_active.asp) | a:active | the active link |
| [::after](https://www.w3schools.com/cssref/sel_after.asp) | p::after | Insert something after the content of each <p> tag |
| [::before](https://www.w3schools.com/cssref/sel_before.asp) | p::before | Insert something before the content of each <p> tag |
| [:checked](https://www.w3schools.com/cssref/sel_checked.asp) | input:checked | every checked <input> tag |
| [:default](https://www.w3schools.com/cssref/sel_default.asp) | input:default | the default <input> tag |
| [:disabled](https://www.w3schools.com/cssref/sel_disabled.asp) | input:disabled | every disabled <input> tag |
| [:empty](https://www.w3schools.com/cssref/sel_empty.asp) | p:empty | every <p> tag that has no children (including text nodes) |
| [:enabled](https://www.w3schools.com/cssref/sel_enabled.asp) | input:enabled | every enabled <input> tag |
| [:first-child](https://www.w3schools.com/cssref/sel_firstchild.asp) | p:first-child | every <p> tag that is the first child of its parent |
| [::first-letter](https://www.w3schools.com/cssref/sel_firstletter.asp) | p::first-letter | the first letter of every <p> tag |
| [::first-line](https://www.w3schools.com/cssref/sel_firstline.asp) | p::first-line | the first line of every <p> tag |
| [:first-of-type](https://www.w3schools.com/cssref/sel_first-of-type.asp) | p:first-of-type | every <p> tag that is the first <p> tag of its parent |
| [:focus](https://www.w3schools.com/cssref/sel_focus.asp) | input:focus | the input tag which has focus |
| [:fullscreen](https://www.w3schools.com/cssref/sel_fullscreen.asp) | :fullscreen | the tag that is in full-screen mode |
| [:hover](https://www.w3schools.com/cssref/sel_hover.asp) | a:hover | links on mouse over |
| [:in-range](https://www.w3schools.com/cssref/sel_in-range.asp) | input:in-range | input tags with a value within a specified range |
| [:indeterminate](https://www.w3schools.com/cssref/sel_indeterminate.asp) | input:indeterminate | input tags that are in an indeterminate state |
| [:invalid](https://www.w3schools.com/cssref/sel_invalid.asp) | input:invalid | all input tags with an invalid value |
| [:lang(*language*)](https://www.w3schools.com/cssref/sel_lang.asp) | p:lang(it) | every <p> tag with a lang attribute equal to "it" (Italian) |
| [:last-child](https://www.w3schools.com/cssref/sel_last-child.asp) | p:last-child | every <p> tag that is the last child of its parent |
| [:last-of-type](https://www.w3schools.com/cssref/sel_last-of-type.asp) | p:last-of-type | every <p> tag that is the last <p> tag of its parent |
| [:link](https://www.w3schools.com/cssref/sel_link.asp) | a:link | all unvisited links |
| [::marker](https://www.w3schools.com/cssref/sel_marker.asp) | ::marker | the markers of list items |
| [:not(*selector*)](https://www.w3schools.com/cssref/sel_not.asp) | :not(p) | every tag that is not a <p> tag |
| [:nth-child(*n*)](https://www.w3schools.com/cssref/sel_nth-child.asp) | p:nth-child(2) | every <p> tag that is the second child of its parent |
| [:nth-last-child(*n*)](https://www.w3schools.com/cssref/sel_nth-last-child.asp) | p:nth-last-child(2) | every <p> tag that is the second child of its parent, counting from the last child |
| [:nth-last-of-type(*n*)](https://www.w3schools.com/cssref/sel_nth-last-of-type.asp) | p:nth-last-of-type(2) | every <p> tag that is the second <p> tag of its parent, counting from the last child |
| [:nth-of-type(*n*)](https://www.w3schools.com/cssref/sel_nth-of-type.asp) | p:nth-of-type(2) | every <p> tag that is the second <p> tag of its parent |
| [:only-of-type](https://www.w3schools.com/cssref/sel_only-of-type.asp) | p:only-of-type | every <p> tag that is the only <p> tag of its parent |
| [:only-child](https://www.w3schools.com/cssref/sel_only-child.asp) | p:only-child | every <p> tag that is the only child of its parent |
| [:optional](https://www.w3schools.com/cssref/sel_optional.asp) | input:optional | input tags with no "required" attribute |
| [:out-of-range](https://www.w3schools.com/cssref/sel_out-of-range.asp) | input:out-of-range | input tags with a value outside a specified range |
| [::placeholder](https://www.w3schools.com/cssref/sel_placeholder.asp) | input::placeholder | input tags with the "placeholder" attribute specified |
| [:read-only](https://www.w3schools.com/cssref/sel_read-only.asp) | input:read-only | input tags with the "readonly" attribute specified |
| [:read-write](https://www.w3schools.com/cssref/sel_read-write.asp) | input:read-write | input tags with the "readonly" attribute NOT specified |
| [:required](https://www.w3schools.com/cssref/sel_required.asp) | input:required | input tags with the "required" attribute specified |
| [:root](https://www.w3schools.com/cssref/sel_root.asp) | :root | the document's root tag |
| [::selection](https://www.w3schools.com/cssref/sel_selection.asp) | ::selection | the portion of an tag that is selected by a user |
| [:target](https://www.w3schools.com/cssref/sel_target.asp) | #news:target | the current active #news tag (clicked on a URL containing that anchor name) |
| [:valid](https://www.w3schools.com/cssref/sel_valid.asp) | input:valid | all input tags with a valid value |

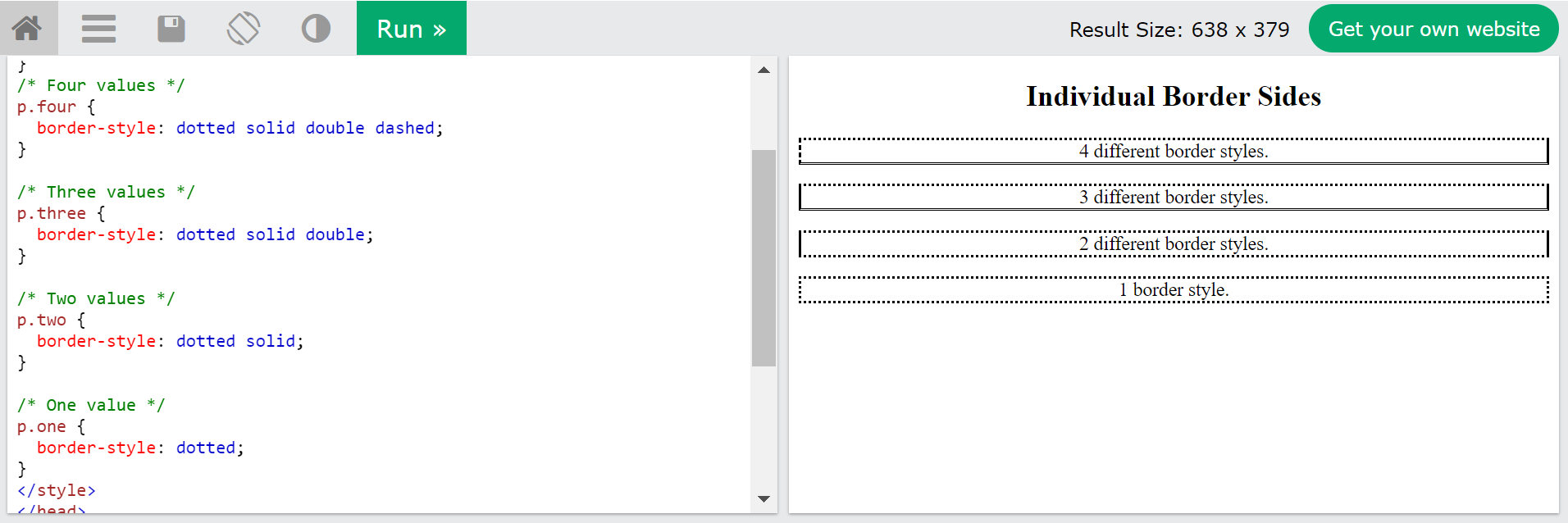
Example on ::after selector

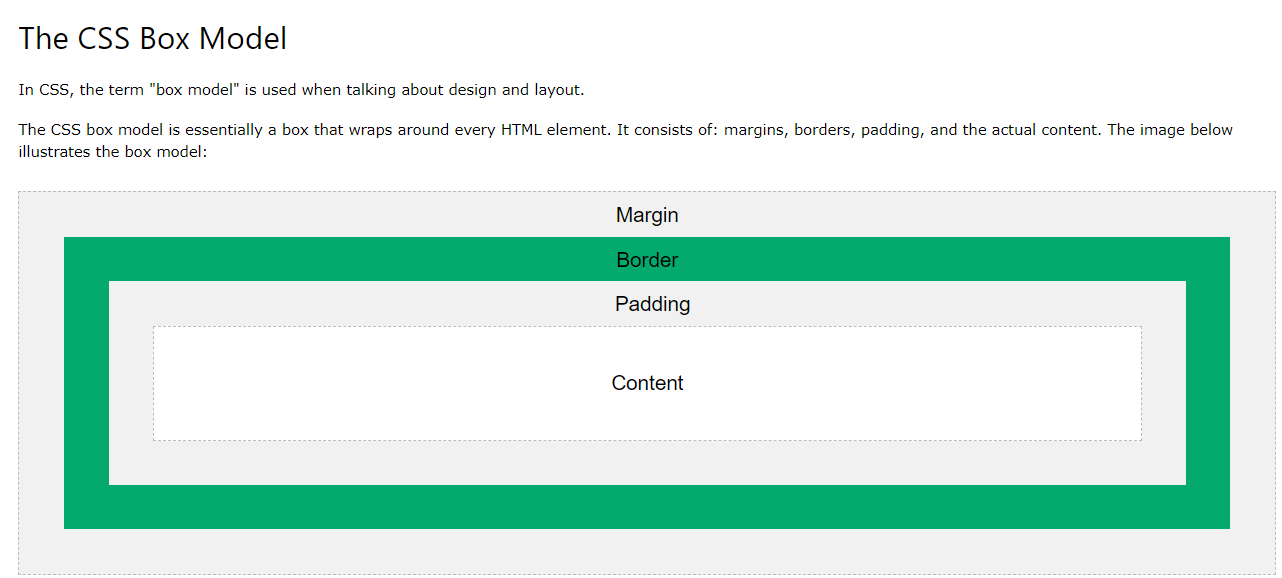


Defining fixed background:



Border sides:





When the text-align property is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers):

In CSS there are five generic font families:

1. **Serif** fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.
2. **Sans-serif** fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.
3. **Monospace** fonts - here all the letters have the same fixed width. They create a mechanical look.
4. **Cursive** fonts imitate human handwriting.
5. **Fantasy** fonts are decorative/playful fonts.



Font Pairing Rules

Here are some basic rules to create great font pairings:

1. Compliment 🡪 It is always safe to find font pairings that complement one another.

A great font combination should harmonize, without being too similar or too different.

2. Use Font Superfamilies 🡪 A font superfamily is a set of fonts designed to work well together. So, using different fonts within the same superfamily is safe.

For example, the Lucida superfamily contains the following fonts: Lucida Sans, Lucida Serif, Lucida Typewriter Sans, Lucida Typewriter Serif and Lucida Math.

3. Contrast is King 🡪 Two fonts that are too similar will often conflict. However, contrasts, done the right way, brings out the best in each font.

Example: Combining serif with sans serif is a well-known combination.

A strong superfamily includes both serif and sans serif variations of the same font (e.g. Lucida and Lucida Sans).

4. Choose Only One Boss

One font should be the boss. This establishes a hierarchy for the fonts on your page. This can be achieved by varying the size, weight and color

Units in CSS:

Angles

deg --> degrees

grad --> grads

rad --> radians

turn --> turns

Time

ms --> mili-seconds

s --> seconds

Frequency

Hz --> hertz

kHz --> kilo-hertz

Absolute Measurement

% 🡪 percentage

cm 🡪 centimeter

in 🡪 inch

mm 🡪 milimeter

pc 🡪 pica (1p = 12 points)

pt 🡪 int (1pt = 1/72 inch)

px 🡪 pixel (1px = 1/96 inch)

Relative Measurement

ch 🡪 width of the “0” glyph found in the font for the font size used to render

em 🡪 1em = current font size of current element

ex 🡪 x-height of the element’s font

gd 🡪 the grid defined by ‘layout-grid’

rem 🡪 the font size of the root element

vh 🡪 the viewport’s height

vw 🡪 the viewport's width

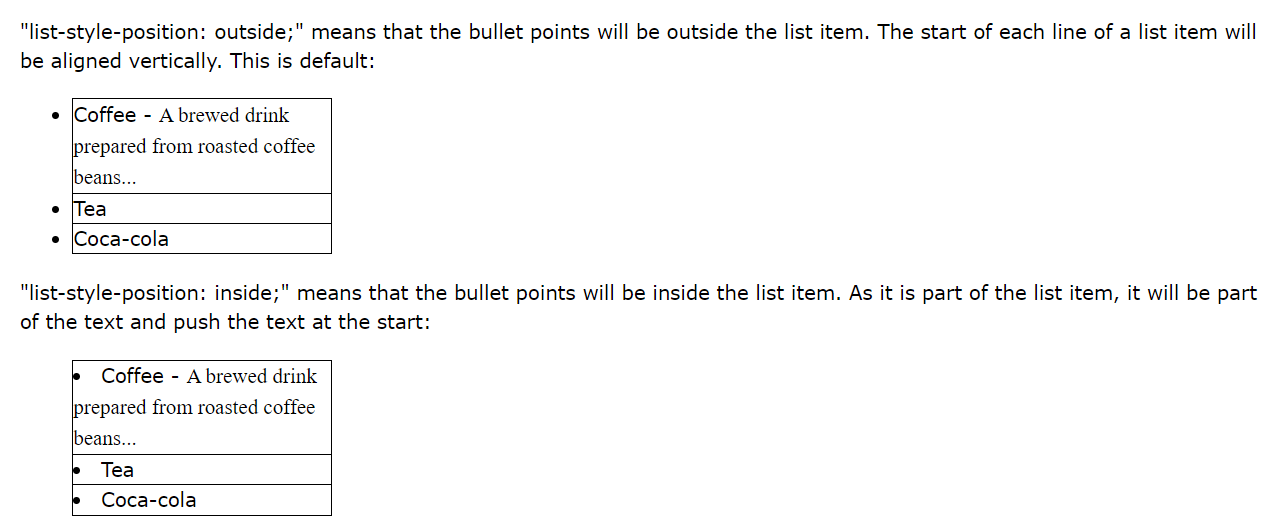
vm 🡪 viewport’s height or width, whichever is smaller of the two

Css icons

<https://fontawesome.com/>

add this in head of your pages

<script src="https://kit.fontawesome.com/b760a80f7e.js" crossorigin="anonymous"></script>



## CSS Table Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| [border](https://www.w3schools.com/cssref/pr_border.asp) | Sets all the border properties in one declaration |
| [border-collapse](https://www.w3schools.com/cssref/pr_border-collapse.asp) | Specifies whether or not table borders should be collapsed |
| [border-spacing](https://www.w3schools.com/cssref/pr_border-spacing.asp) | Specifies the distance between the borders of adjacent cells |
| [caption-side](https://www.w3schools.com/cssref/pr_tab_caption-side.asp) | Specifies the placement of a table caption |
| [empty-cells](https://www.w3schools.com/cssref/pr_tab_empty-cells.asp) | Specifies whether or not to display borders and background on empty cells in a table |
| [table-layout](https://www.w3schools.com/cssref/pr_tab_table-layout.asp) | Sets the layout algorithm to be used for a table |

## **Block-level Elements**

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The <div> element is a block-level element.

Examples of block-level elements:

* <div>
* <h1> - <h6>
* <p>
* <form>
* <header>
* <footer>
* <section>

## **Inline Elements**

An inline element does not start on a new line and only takes up as much width as necessary.

This is an inline <span> element inside a paragraph.

Examples of inline elements:

* <span>
* <a>
* <img>

## **Display: none;**

display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved.

Hiding an element can be done by setting the display property to none. The element will be hidden, and the page will be displayed as if the element is not there:

visibility:hidden; also hides an element.

However, the element will still take up the same space as before. The element will be hidden, but still affect the layout:

## **he position Property**

The position property specifies the type of positioning method used for an element.

There are five different position values:

* Static
  + An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:
* Relative
  + Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.
* Fixed
  + An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.
* Absolute
  + An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
  + However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.
* Sticky
  + An element with position: sticky; is positioned based on the user's scroll position.
  + A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

**Note:**Internet Explorer does not support sticky positioning. Safari requires a -webkit- prefix (see example below). You must also specify at least one of top, right, bottom or left for sticky positioning to work.

The CSS overflow property controls what happens to content that is too big to fit into an area

The overflow property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

The overflow property has the following values:

* visible - Default. The overflow is not clipped. The content renders outside the element's box
* hidden - The overflow is clipped, and the rest of the content will be invisible
* scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content
* auto - Similar to scroll, but it adds scrollbars only when necessary

When we use the float property, and we want the next element below (not on right or left), we will have to use the clear property.

The clear property specifies what should happen with the element that is next to a floating element.

The clear property can have one of the following values:

* none - The element is not pushed below left or right floated elements. This is default
* left - The element is pushed below left floated elements
* right - The element is pushed below right floated elements
* both - The element is pushed below both left and right floated elements
* inherit - The element inherits the clear value from its parent
* Compared to display: inline, the major difference is that display: inline-block allows to set a width and height on the element.
* Also, with display: inline-block, the top and bottom margins/paddings are respected, but with display: inline they are not.
* Compared to display: block, the major difference is that display: inline-block does not add a line-break after the element, so the element can sit next to other elements.

There are four different combinators in CSS:

* descendant selector (space)
* child selector (>)
* adjacent sibling selector (+)
* general sibling selector (~)

Pseudo-Class

:active 🡪 an activated element

:focus 🡪 an element while the element has focus

:hover 🡪 an element when you mouse over it

:link 🡪 an unvisited link

:disabled 🡪 an element while the element is disabled

:enabled 🡪 an element while the element is enabled

:checked 🡪 an element that is checked

:selection 🡪 an element that is currently selected or highlighted by the user

:lang 🡪 allows the author to specify a language to use in a specified element

:nth-child(n) 🡪 an element that is the n-th sibling

:nth-last-child(n) 🡪 an element that is the n-th sibling from last sibling

:first-child 🡪 an element that is the first sibling

:last-child 🡪 an element that is the last sibling

:only-child 🡪 an element that is the only child

:nth-of-type(n) 🡪 an element that is the n-th sibling of its type

:nth-last-of-type(n) 🡪 an element that is the n-th sibling from the last sibling

:last-of-type 🡪 an element that is the last sibling of its type

:first-of-type 🡪 an element that is the first sibling of its type

:only-of-type 🡪 an element that is the only child of its type

:empty 🡪 an element that has no children

:root 🡪 root element within the document

:not(x) 🡪 an element not represented by the argument ‘x’

:target 🡪 a target element as specified by a target in a UR

Pseudo-Element

::first-letter 🡪 Adds special style to the first letter of a text

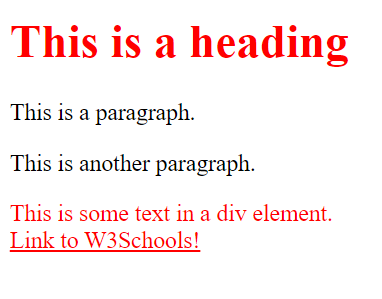
::first-line 🡪 Adds special style to the first line of a text

::before 🡪 Inserts some content before the content of an element

::after 🡪 Inserts some content after the content of an element

The ::marker pseudo-element selects the markers of list items.

**Note:** a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective! a:active MUST come after a:hover in the CSS definition in order to be effective! Pseudo-class names are not case-sensitive.

:not(p) { color: #ff0000;}

## **All CSS Pseudo Classes**

|  |  |  |
| --- | --- | --- |
| **Selector** | **Example** | **Example description** |
| [:active](https://www.w3schools.com/cssref/sel_active.asp) | a:active | Selects the active link |
| [:checked](https://www.w3schools.com/cssref/sel_checked.asp) | input:checked | Selects every checked <input> element |
| [:disabled](https://www.w3schools.com/cssref/sel_disabled.asp) | input:disabled | Selects every disabled <input> element |
| [:empty](https://www.w3schools.com/cssref/sel_empty.asp) | p:empty | Selects every <p> element that has no children |
| [:enabled](https://www.w3schools.com/cssref/sel_enabled.asp) | input:enabled | Selects every enabled <input> element |
| [:first-child](https://www.w3schools.com/cssref/sel_firstchild.asp) | p:first-child | Selects every <p> elements that is the first child of its parent |
| [:first-of-type](https://www.w3schools.com/cssref/sel_first-of-type.asp) | p:first-of-type | Selects every <p> element that is the first <p> element of its parent |
| [:focus](https://www.w3schools.com/cssref/sel_focus.asp) | input:focus | Selects the <input> element that has focus |
| [:hover](https://www.w3schools.com/cssref/sel_hover.asp) | a:hover | Selects links on mouse over |
| [:in-range](https://www.w3schools.com/cssref/sel_in-range.asp) | input:in-range | Selects <input> elements with a value within a specified range |
| [:invalid](https://www.w3schools.com/cssref/sel_invalid.asp) | input:invalid | Selects all <input> elements with an invalid value |
| [:lang(*language*)](https://www.w3schools.com/cssref/sel_lang.asp) | p:lang(it) | Selects every <p> element with a lang attribute value starting with "it" |
| [:last-child](https://www.w3schools.com/cssref/sel_last-child.asp) | p:last-child | Selects every <p> elements that is the last child of its parent |
| [:last-of-type](https://www.w3schools.com/cssref/sel_last-of-type.asp) | p:last-of-type | Selects every <p> element that is the last <p> element of its parent |
| [:link](https://www.w3schools.com/cssref/sel_link.asp) | a:link | Selects all unvisited links |
| [:not(selector)](https://www.w3schools.com/cssref/sel_not.asp) | :not(p) | Selects every element that is not a <p> element |
| [:nth-child(n)](https://www.w3schools.com/cssref/sel_nth-child.asp) | p:nth-child(2) | Selects every <p> element that is the second child of its parent |
| [:nth-last-child(n)](https://www.w3schools.com/cssref/sel_nth-last-child.asp) | p:nth-last-child(2) | Selects every <p> element that is the second child of its parent, counting from the last child |
| [:nth-last-of-type(n)](https://www.w3schools.com/cssref/sel_nth-last-of-type.asp) | p:nth-last-of-type(2) | Selects every <p> element that is the second <p> element of its parent, counting from the last child |
| [:nth-of-type(n)](https://www.w3schools.com/cssref/sel_nth-of-type.asp) | p:nth-of-type(2) | Selects every <p> element that is the second <p> element of its parent |
| [:only-of-type](https://www.w3schools.com/cssref/sel_only-of-type.asp) | p:only-of-type | Selects every <p> element that is the only <p> element of its parent |
| [:only-child](https://www.w3schools.com/cssref/sel_only-child.asp) | p:only-child | Selects every <p> element that is the only child of its parent |
| [:optional](https://www.w3schools.com/cssref/sel_optional.asp) | input:optional | Selects <input> elements with no "required" attribute |
| [:out-of-range](https://www.w3schools.com/cssref/sel_out-of-range.asp) | input:out-of-range | Selects <input> elements with a value outside a specified range |
| [:read-only](https://www.w3schools.com/cssref/sel_read-only.asp) | input:read-only | Selects <input> elements with a "readonly" attribute specified |
| [:read-write](https://www.w3schools.com/cssref/sel_read-write.asp) | input:read-write | Selects <input> elements with no "readonly" attribute |
| [:required](https://www.w3schools.com/cssref/sel_required.asp) | input:required | Selects <input> elements with a "required" attribute specified |
| [:root](https://www.w3schools.com/cssref/sel_root.asp) | root | Selects the document's root element |
| [:target](https://www.w3schools.com/cssref/sel_target.asp) | #news:target | Selects the current active #news element (clicked on a URL containing that anchor name) |
| [:valid](https://www.w3schools.com/cssref/sel_valid.asp) | input:valid | Selects all <input> elements with a valid value |
| [:visited](https://www.w3schools.com/cssref/sel_visited.asp) | a:visited | Selects all visited links |

The following example makes the selected text red on a yellow background:

::selection { color: red; background: yellow;}

## **What is Specificity?**

If there are two or more conflicting CSS rules that point to the same element, the browser follows some rules to determine which one is most specific and therefore wins out.

Think of specificity as a score/rank that determines which style declarations are ultimately applied to an element.

The universal selector (\*) has low specificity, while ID selectors are highly specific!

## **Specificity Hierarchy**

Every selector has its place in the specificity hierarchy. There are four categories which define the specificity level of a selector:

**Inline styles** - An inline style is attached directly to the element to be styled. Example: <h1 style="color: #ffffff;">.

**IDs** - An ID is a unique identifier for the page elements, such as #navbar.

**Classes, attributes and pseudo-classes** - This category includes .classes, [attributes] and pseudo-classes such as :hover, :focus etc.

**Elements and pseudo-elements** - This category includes element names and pseudo-elements, such as h1, div, :before and :after.

### **Example**

A: h1  
B: #content h1  
C: <div id="content"><h1 style="color: #ffffff">Heading</h1></div>

The specificity of A is 1 (one element)  
The specificity of B is 101 (one ID reference and one element)  
The specificity of C is 1000 (inline styling)

Since 1 < 101 < 1000, the third rule (C) has a greater level of specificity, and therefore will be applied.

**Equal specificity: the latest rule counts**- If the same rule is written twice into the external style sheet, then the lower rule in the style sheet is closer to the element to be styled, and therefore will be applied:

**ID selectors have a higher specificity than attribute selectors** - Look at the following three code lines:

div#a {background-color: green;}  
#a {background-color: yellow;}  
div[id=a] {background-color: blue;}

the first rule is more specific than the other two, and will be applied.

**Contextual selectors are more specific than a single element selector -**The embedded style sheet is closer to the element to be styled. So in the following situation

**A class selector beats any number of element selectors**- a class selector such as .intro beats h1, p, div, etc:

The !important rule in CSS is used to add more importance to a property/value than normal.

In fact, if you use the !important rule, it will override ALL previous styling rules for that specific property on that element!

The only way to override an !important rule is to include another !important rule on a declaration with the same (or higher) specificity in the source code - and here the problem starts! This makes the CSS code confusing and the debugging will be hard, especially if you have a large style sheet!

One way to use !important is if you have to override a style that cannot be overridden in any other way. This could be if you are working on a Content Management System (CMS) and cannot edit the CSS code. Then you can set some custom styles to override some of the CMS styles.

## **How var() Works**

First of all: CSS variables can have a global or local scope.

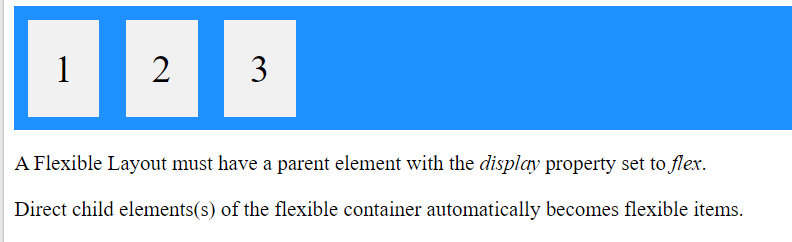
Global variables can be accessed/used through the entire document, while local variables can be used only inside the selector where it is declared.

To create a variable with global scope, declare it inside the :root selector. The [:root](https://www.w3schools.com/cssref/sel_root.asp) selector matches the document's root element.

To create a variable with local scope, declare it inside the selector that is going to use it.

The following example is equal to the example above, but here we use the var() function.

First, we declare two global variables (--blue and --white). Then, we use the var() function to insert the value of the variables later in the style sheet:



.flex-container > div {

background-color: #f1f1f1;

margin: 10px;

padding: 20px;

font-size: 30px;

}