CSS stands for ***Cascading Style Sheets***.

CSS is a ***declarative*** ***styling*** language.

The word **cascading** means that a style applied to a parent element will also apply to all children elements within the parent.

CSS was first proposed by **Hakon Wium Lie** on October 10, 1994.

CSS **saves** a **lot of work**. It can control the **layout** of multiple web pages all at once.

CSS is used to format the layout of a webpage.

***CSS can be added to HTML documents in 3 ways:***

* **Inline** - by using the style attribute inside HTML elements
  + An inline CSS is used to apply a unique style to a **single HTML element.**

<p style="color: red; border: 2px solid grey;">A red paragraph.</p>

* **Internal/ Embedded** - by using a <style> element in the <head> section
  + An internal CSS is used to define a style for a **single HTML page.**

<style>  
body {background-color: powderblue;}  
h1   {color: blue;}  
p    {color: red;}  
</style>

* **External** - by using a <link> element to link to an external CSS file
  + An external style sheet is used to define the style for **many HTML pages.**
  + <head>  
      <link rel="stylesheet" href="styles.css">  
    </head>
  + "styles.css":

body {  
  background-color: powderblue;  
}  
h1 {  
  color: blue;  
}  
p {  
  color: red;  
}

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section) [depends on what come last to read]
3. Browser default

# Syntax

Declaration Box



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

# CSS Selectors

* **Simple selectors**- ***select elements based on element name, id, class***

|  |  |  |
| --- | --- | --- |
| Selector | Example | Example description |
| [#*id*](about:blank) | #firstname | Selects the element with id="firstname" |
| [.*class*](about:blank) | .intro | Selects all elements with class="intro" |
| [*element.class*](about:blank) | p.intro | Selects only <p> elements with class="intro" |
| [\*](about:blank) universal selector | \* | Selects all elements |
| [*element*](about:blank) | p | Selects all <p> elements |
| [*element,element,..*](about:blank) | div, p | Selects all <div> elements and all <p> elements |

* [**Combinator selectors**](about:blank)- ***select elements based on a specific relationship between them***
* ***descendant selector (space)***

Eg div p {} Every p inside div irrespective of div -> section -> p

* ***child selector (>)***

Eg div > p {} Only child p inside div and div -> section -> p is not allowed

* ***adjacent sibling selector (+)***

Eg div + p {} first p element that are placed immediately after div elements, not previous.

* ***general sibling selector (~)***

Eg div ~ p {} All p elements that are placed immediately after div elements

* [**Pseudo-class selectors**](about:blank)**-** ***select elements based on a certain state***

A pseudo-class is used to define a special state of an element.

* Style an element when a user mouses over it
* Style visited and unvisited links differently
* Style an element when it gets focus

selector:pseudo-class {  
  property: value;  
}

* [**Pseudo-elements selectors**](about:blank)- ***select and style a part of an element***

A CSS pseudo-element is used to style specified parts of an element.

* + Style the first letter, or line, of an element
  + Insert content before, or after, the content of an element

selector::pseudo-element {  
  property: value;  
}

* [**Attribute selectors**](about:blank)- ***select elements based on an attribute or attribute value***

HTML elements that have specific attributes or attribute values.

The [attribute] selector is used to select elements with a specified attribute.

a[target] {  
  background-color: yellow;  
}

The [attribute="value"] selector is used to select elements with a specified attribute and value.

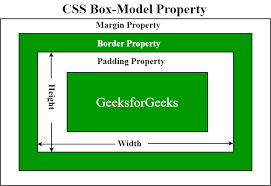
a[target="\_blank"] {  
  background-color: yellow;  
}

# Comments

Single Line = Multi Line = /\*…\*/

# Border Box

All HTML elements can be considered as a boxes and the CSS box model refers to how**HTML elements are modeled in browser engines.**



Content

# Position

**Static** = Default value. Elements render in order, as they appear in the document flow

Static positioned elements are not affected by the top, bottom, left, and right properties.

**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.

A fixed element does not leave a gap in the page where it would normally have been located.

**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.

Note: Absolute positioned elements are removed from the normal flow, and can overlap elements.

**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

Position = relative

Position = absolute

# !important

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!

# Z-index z-index - Codrops

Higher z-Index

Lower Z-index

# Math Functions

|  |  |
| --- | --- |
| Function | Description |
| [calc()](about:blank) | Allows you to perform calculations to determine CSS property values |
| [max()](about:blank) | Uses the largest value, from a comma-separated list of values, as the property value |
| [min()](about:blank) | Uses the smallest value, from a comma-separated list of values, as the property value |

# Specificity

If there are two or more CSS rules that point to the same element, the selector with the highest specificity value will "win", and its style declaration will be applied to that HTML element.

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

|  |  |
| --- | --- |
| CSS Selector | Description |
| Inherited styles | Lowest specificity of all selectors - since an inherited style targets the element's parent, and not the HTML element itself. |
| \* | Lowest specificity of all directly targeted selectors |
| element | Higher specificity than universal selector and inherited styles. |
| attribute | Higher specificity than element selector |
| class | Higher specificity than attribute, element and universal selectors. |
| ID | Higher specificity than class selector. |
| Combined selectors | Gets the specificity of the selectors combined. |
| CSS properties set directly on element, inside style attribute. | Stronger specificity than ID selector. |

# Counter

CSS counters are "variables" maintained by CSS whose values can be incremented by CSS rules (to track how many times they are used). Counters let you adjust the appearance of content based on its placement in the document.

* counter-reset - Creates or resets a counter
* counter-increment - Increments a counter value
* content - Inserts generated content
* counter() or counters() function - Adds the value of a counter to an element

# Units

## **Absolute Lengths**

The ***absolute length units are fixed***, and a length expressed in any of these will appear as exactly that size.

Absolute length units are not recommended for use on screen, because screen sizes vary so much. However, they can be used if the output medium is known, such as for print layout.

## **Relative Lengths**

Relative length units specify a length relative to another length property. ***Relative length units scale better between different rendering mediums***.

Flexbox

The Flexible Box Layout Module makes it easier to design ***flexible responsive layout structure*** without using float or positioning.

Flexbox mostly helps align content & move blocks.

Flexbox works better in ***one dimension only (either rows OR columns).***

Grid

The CSS Grid Layout Module offers a ***grid-based layout system***, ***with rows and columns,*** making it easier to design web pages without having to use floats and positioning.

CSS Grids helps you create the outer layout of the webpage. You can build complex as well responsive design with this. This is why it is called ‘***layout first’***.

CSS grids are for ***2D layouts. It works with both rows and columns.***