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

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.

Cascading Style Sheets (CSS) is used to format the ***layout*** of a webpage.

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

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** - 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;  
}

**Inline CSS has the highest priority, then comes Internal/Embedded followed by External CSS which has the least priority**.

# Syntax



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 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 d and div -> section -> p is not allowed

*adjacent sibling selector (+)*

Eg div + p {} The following example selects the first p element that are placed immediately after div elements

*general sibling selector (~)*

Eg div ~ p {} The general sibling selector (~) selects all elements that are next siblings of a specified element.

* [**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.

For example, it can be used to:

* 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.

For example, it can be used to:

Style the first letter, or line, of an element

Insert content before, or after, the content of an element

## **Syntax**

The syntax of pseudo-elements:

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;  
}

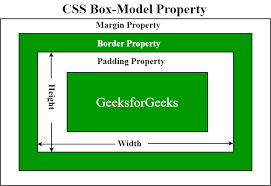
The [attribute="value"] selector is used to select elements with a specified attribute and value.

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

# Comments

Single Line = Multi Line = /\*…\*/

# Border Box



Content

# Position

* Static = By default
* relative
* fixed = Stays in the same place even if the page is scrolled.
* absolute
* sticky = A sticky element toggles between relative and fixed

Position = absolute

Position = relative

# Z-index z-index - Codrops

Lower Z-index

Higher z-Index

# !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!

# 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 scales better between different rendering mediums.