

# What Is CSS?

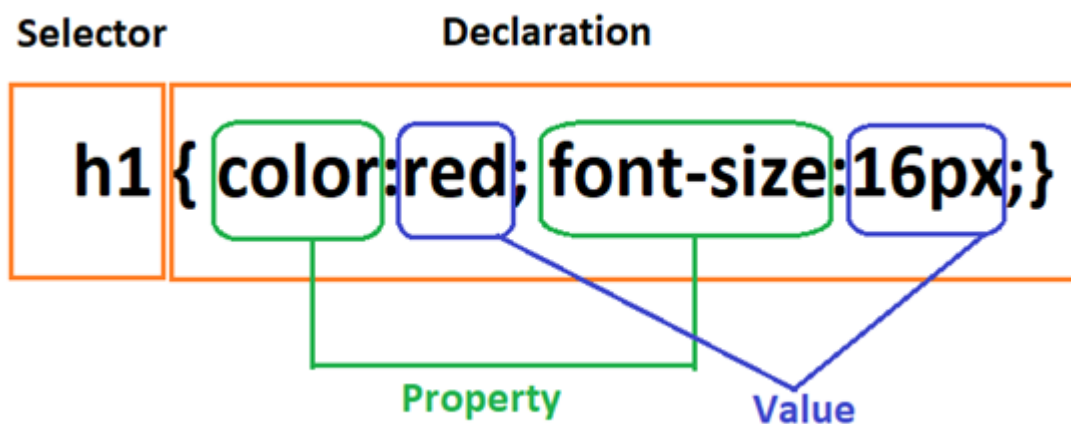
CSS stands for **Cascading Style Sheets**.

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.

With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colors are to be used, different displays for different devices and screen sizes, and much more!

## Basic 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.

## Using CSS

CSS can be added to HTML documents in 3 ways:

- Inline - by using the style attribute inside HTML elements
- Internal - by using a <style> element in the <head> section
- External - by using a <link> element to link to an external CSS file

The most common way to add CSS, is to keep the styles in external CSS files.

## Inline CSS

An inline CSS is used to apply a unique style to a single HTML element.

An inline CSS uses the style attribute of an HTML element.

The following example sets the text color of the <h1> element to blue, and the text color of the <p> element to red:

```
<h1 style="color:blue;">A Blue Heading</h1>
```

```
<p style="color:red;">A red paragraph.</p>
```

## Internal CSS

An internal CSS is used to define a style for a single HTML page.

An internal CSS is defined in the <head> section of an HTML page, within a <style> element.

The following example sets the text color of ALL the <h1> elements (on that page) to blue, and the text color of ALL the <p> elements to red. In addition, the page will be displayed with a "powderblue" background color:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

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

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

## External CSS

An external style sheet is used to define the style for many HTML pages.

To use an external style sheet, add a link to it in the <head> section of each HTML page:

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="style.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```

```
</body>
```

```
</html>
```

The external style sheet can be written in any text editor. The file must not contain any HTML code, and must be saved with a .css extension.

Here is what the "styles.css" file looks like:

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

# Selectors

## Basic Selectors

- **Element Selector**
- The element selector selects HTML elements based on the element name.

```
h1 {  
font-size 16px;  
color: red;  
}
```

Here, all <h1> elements on the page will be 16px in font and red in color.

- **Id selector**

- The id selector uses the id attribute of an HTML element to select a specific element.
- The id of an element is unique within a page, so the id selector is used to select one unique element!
- To select an element with a specific id, write a hash (#) character, followed by the id of the element.

- 

- ```
#p-head {  
  font-size 16px;  
  color: red;  
}
```

- 

- **Note: An id name cannot start with a number!**

- 

- **Class selector**

- The class selector selects HTML elements with a specific class attribute.
- To select elements with a specific class, write a period (.) character, followed by the class name.

- 

- ```
.text-red {  
  font-size 16px;  
  color: red;  
}
```

- 

- **Universal selector**

- The universal selector (\*) selects all HTML elements on the page.

- 

- ```
* {  
  font-size: 16px;  
  color: red;  
}
```

- 
- **Grouping selector**
- The grouping selector selects all the HTML elements with the same style definitions.
- 
- `h1, p, .text-red {`  
`font-size 16px;`  
`color: red;`  
`}`

## Advanced Selector

### Pseudo class selector

- 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

### Syntax:

```
selector:pseudo-class {
property: value;
}
```

Examples:

```
/* unvisited link */
a:link {
    color: #FF0000;
}

/* visited link */
a:visited {
    color: #00FF00;
}

/* mouse over link */
a:hover {
    color: #FF00FF;
}

/* selected link */
a:active {
    color: #0000FF;
}
```

```
tr:hover {
    background-color: #f5f5f5;
}
/* table-striped */
tr:nth-child(even) {
    background-color: #f2f2f2;
}
```

```

li:first-child {
  /* css styles here */
}
li:last-child {
  /* css styles here */
}
li:nth-child(4) {
  /* css styles here */
}
li:nth-child(odd) {
  /* css styles here */
}
li:nth-child(even) {
  /* css styles here */
}
input:focus {
  /* css styles here */
}

```

## Pseudo element selector

- 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:

```

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

```

Examples,



```
p::first-letter {
  text-transform: uppercase;
}
p::first-line {
  font-size: 16px;
}
h1::before {
  content: url("image.png");
}
h1::after {
  content: "This is after content";
}
::selection {
  color: ■red;
  background-color: ■white;
}
::marker {
  color: ■blue;
}
```

**Note:** Marker is used in lists to define the style of the lists.

### **::marker**

The ::marker CSS pseudo-element selects the marker box of a list item, which typically contains a bullet or number. It works on any element or pseudo-element set to display: list-item, such as the <li> and <summary> elements.

- **Attribute selector**

It is possible to style HTML elements that have specific attributes or attribute values.

The following example selects all <a> elements with a target attribute.

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

```
}
```

The following example selects all <a> elements with a target="\_blank" attribute.

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

The attribute selectors can be useful for styling forms without class or ID.

```
input[type="text"] {  
width: 150px;  
display: block;  
margin-bottom: 10px;  
background-color: yellow;  
}
```

```
input[type="button"] {  
width: 120px;  
margin-left: 35px;  
display: block;  
}
```

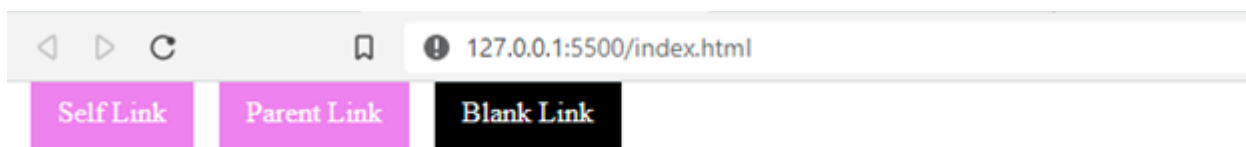
Examples,

```
<body>
  <div class="main-div">
    <a href="" target="_self">Self Link</a>
    <a href="" target="_parent">Parent Link</a>
    <a href="" target="_blank">Blank Link</a>
    <p></p>
  </div>
</body>
```

```
a[target] {
  background-color: #violet;
  color: #white;
  padding: 15px;
  text-decoration: none;
  margin: 5px;
}
```

```
a[target="_blank"] {
  background-color: #black;
  color: #white;
  padding: 15px;
  text-decoration: none;
  margin: 5px;
}
```

Output:



Example 2:

```
<body>
  <div class="main-div">
    <form>
      <input type="text" placeholder="Enter first name" /><br /><br />
      <input type="button" value="Submit" /><br />
    </form>
  </div>
</body>
```

```
input[type="text"] {
  color: ■ white;
  padding: 5px;
  border-radius: 5px;
  border: 2px solid ■ green;
  font-size: 16px;
}

input[type="button"] {
  background-color: ■ purple;
  color: ■ white;
  padding: 15px;
  margin: 5px;
  font-size: 20px;
  border-radius: 15px;
  border: none;
}
```

Output:



## Combinator Selector

- 1. Descendant Selector

The descendant selector matches all elements that are descendants of a specified element.

The following example selects all `<p>` elements inside `<div>` elements.

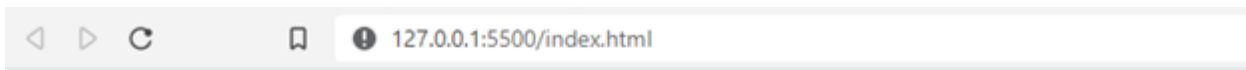
```
div p {  
background-color: yellow;  
}
```

Example,

```
<body>  
  <div class="main-div">  
    <h1>This is heading 1</h1>  
    <h2>This is heading 2</h2>  
    <div>  
      <h1>This is another heading 1</h1>  
    </div>  
  </div>  
</body>
```

```
.main-div h1 {  
background-color: orange;  
color: white;  
}
```

Output:



**This is heading 1**

**This is heading 2**

**This is another heading 1**

## 2. Child Selector (>)

The child selector selects all elements that are the children of a specified element.

The following example selects all <p> elements that are children of a <div> element

- ```
div > p {  
  background-color: yellow;  
}
```

```
.main-div > h1 {  
  background-color: orange;  
  color: white;  
}
```

Output:

**This is heading 1**

**This is heading 2**

**This is another heading 1**

### 3. Adjacent Sibling Selector (+)

The adjacent sibling selector is used to select an element that is directly after another specific element.

Sibling elements must have the same parent element, and "adjacent" means "immediately following".

The following example selects the first `<p>` element that are placed immediately after `<div>` elements

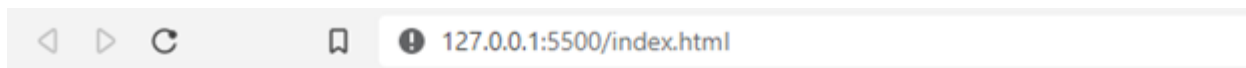
```
div + p {  
background-color: yellow;  
}
```

Example,

```
<body>
  <div class="main-div">
    <h1>This is heading 1</h1>
    <h2>This is heading 2</h2>
    <div>
      <h1>This is another heading 1</h1>
    </div>
    <h2>This is another heading 2</h3>
  </div>
</body>
```

```
h1 + h2 {
  background-color: orange;
  color: white;
}
```

Output:



**This is heading 1**

**This is heading 2**

**This is another heading 1**

**This is another heading 2**

#### 4. General Sibling Selector (~)

The general sibling selector selects all elements that are siblings of a specified element.



The following example selects all <p> elements that are siblings of <div> elements.

```
div ~ p {  
background-color: yellow;  
}
```

Example,

```
h1 ~ h2 {  
background-color: orange;  
color: white;  
}
```

Output:



## Cascading Order

1. **!important** will override everything.
2. Inline CSS can override everything except !important.
3. External or Internal will work according to specificity.




**Note:** Id (#) selector has highest value and universal (\*) selector has lowest value.

# General Rule Of Specificity

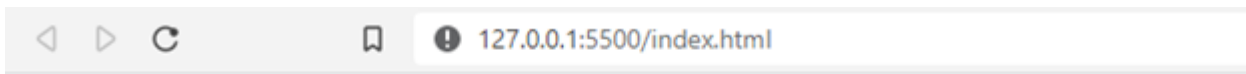
1. !important
2. Inline CSS
3. Id
4. Class, Attribute and Pseudo-class
5. Element selector and Pseudo-elements
6. Universal selector

Example,

```
<body>
  <div>
    <p class="red">Content goes here...</p>
  </div>
</body>
```

```
<style>
  p {
    color: red;
  }
  .red {
    color: green;
  }
  div .red {
    color: blue;
  }
</style>
```

Output:



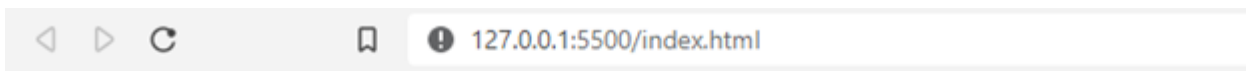
Content goes here...

**Note:** if the same rule is specified two or more times in external or internal stylesheet, the last specified rule will get more priority.

Example,

```
<style>
  div .red {
    color: blue;
  }
  div .red {
    color: red;
  }
</style>
```

Output:




Content goes here...

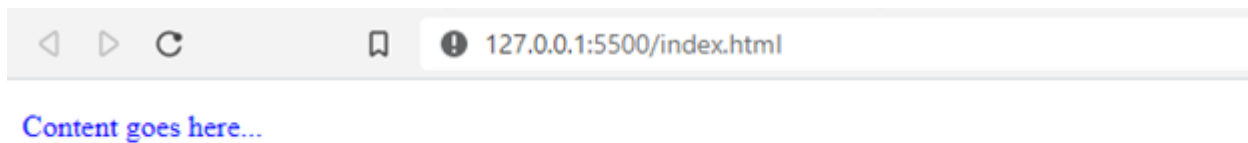
## !Important Use Case

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!

```
<style>
  div .red {
    color: blue !important;
  }
  div .red {
    color: red;
  }
</style>
```

Output:



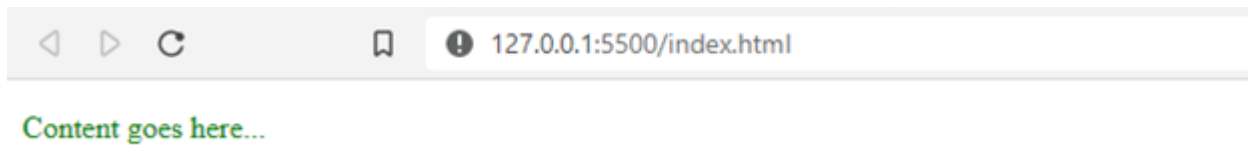
## Id Use Case

```
<body>
  <div>
    <p class="red" id="p-red">Content goes here...</p>
  </div>
</body>
```

```
<style>
  #p-red {
    color: green;
  }

  div .red {
    color: red;
  }
</style>
```

Output:



## Comments

Comments are used to explain the code, and may help when you edit the source code at a later date.

Comments are ignored by browsers.

A CSS comment starts with `/*` and ends with `*/`

```
/* style for body tag */
body {
  background-color: red;
  margin: auto;
  padding: auto;
}
/* body tag style
ends here */
```

## CSS Units

CSS has several different units for expressing a length.

Many CSS properties take "length" values, such as width, margin, padding, font-size, etc.

Length is a number followed by a length unit, such as 10px, 2em, etc.

There are two types of length units: absolute and relative.

## **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.

Centimeters (cm)

Millimeters (mm)

Inch (in) [1in = 96px = 2.54cm]

Pixels (px) [1px = 1/96th of 1in]

Points (pt) [1pt = 1/72 of 1in]

Picas (pc) [1pc = 12 pt]

## **Relative Lengths**

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

- em : Relative to the size of its direct parent.
- rem: Relative to the size of root element.
- vh: Relative to 1% of the height of the viewport
- vw: Relative to 1% of the width of the viewport
- %: Relative to the parent element

```

/* root element */
html {
  font-size: 20px;
}

#div1 {
  font-size: 2rem; /* 2*20 =40px */
}

#parent {
  font-size: 16px;
}

#child {
  width: 50vw;
  height: 50vh;
  font-size: 1em; /* 1*16=16px */
}

```

## Colors


Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

This link <https://htmlcolorcodes.com/color-chart/> can be useful in selecting appropriate color.

- Color name: Color can be applied using color names like red, blue, green and so on.  
For more color names visit this link <https://htmlcolorcodes.com/color-names/>.
- 
- RGB: An RGB color value represents RED, GREEN, and BLUE light sources. This is a color value between 0 and 255. For example,
- rgb(0,0,0) – Black
- rgb(255,255,255) – White
-







- **RGBA:** This is almost same as RGB and has an extra parameter named as ALPHA whose value can be between 0 and 1. The alpha value is used for transparency/opacity. (0.0 means full transparent and 1.0 means no transparent at all). For example,
  - `rgba(255, 70, 50, 0.5)`
  - `rgba(255, 70, 50, 0.0)`
  -
- **HEX:** This is combination of rr (red) gg(green) bb(blue) colors and has hexadecimal values which is between 0-9 and A-F. For example,
  - `#ff0000`, `#000000`, `#ffffff`
  -
- **HSL (Hue, Saturation, Lightness)**
  - Hue is the degree on the color wheel from 0 to 360
  - Saturation is the percentage value. 0% means shade of gray and 100% means full clear.
  - Lightness is also a percentage. 0% is black and 100% is white.
  - For example, `hsl(0, 100%, 20%)`
  -
- **HSLA (Hue, Saturation, Lightness, Alpha)**
  - `hsla(0, 100%, 20%, 0.5)`

Example,

 **HEX** `#FF0000` **RGB** `rgb(255, 0, 0)` **HSL** `hsl(0, 100%, 50%)`



```
<body>
  <p class="color_name">This is use of color name</p>
  <p class="color_rgb">This is use of color rgb</p>
  <p class="color_rgba">This is use of color rgba</p>
  <p class="color_hex">This is use of color hex</p>
  <p class="color_hsl">This is use of color hsl</p>
  <p class="color_hsla">This is use of color hsla</p>
</body>
```

```
.color_name {
  color:  red;
}
.color_rgb {
  color:  rgb(255, 0, 0);
}
.color_rgba {
  color:  rgba(255, 0, 0, 0.5);
}
.color_hex {
  color:  #ff0000;
}
.color_hsl {
  color:  hsl(0, 100%, 50%);
}
.color_hsla {
  color:  hsla(0, 100%, 50%, 0.5);
}
```

Output:

This is use of color name

This is use of color rgb

This is use of color rgba

This is use of color hex

This is use of color hsl

This is use of color hsla