# Cascading Style Sheets

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

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

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!

The word **cascading** means that a style applied to a parent element will also apply to all children elements within the parent. So, if you set the color of the body text to "blue", all headings, paragraphs, and other text elements within the body will also get the same color (unless you specify something else)!

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

**CSS** stands for **"Cascading Style Sheets"** which is a technology to control how elements are presented in the Web page.

The term ***Cascading*** refers to the procedure that determines which style will apply to a certain section, if we have more than one style rule.

The term ***Style*** refers to how you want a certain part of our page to look. We can set colors, fonts, alignment, borders, backgrounds, spacing, margins, and much more.

The term ***Sheets*** represents that the "sheets" are like templates, or a set of rules, for determining how the webpage will look.

So, CSS (all together) is a ***styling language*** that defines a set of rules to tell browsers how the webpage should look.

A style sheet is a document that contains style information about one or more documents written in markup languages. It enables us to control rendering of styles such as fonts, color, type face, size and other aspects of document style.

## What is “Style”?

“Style” is a command that you set to tell the browser how a certain section of your webpage should look. You can apply style on many HTML “elements like <p> <h1> <table> etc.

**Advantages:**

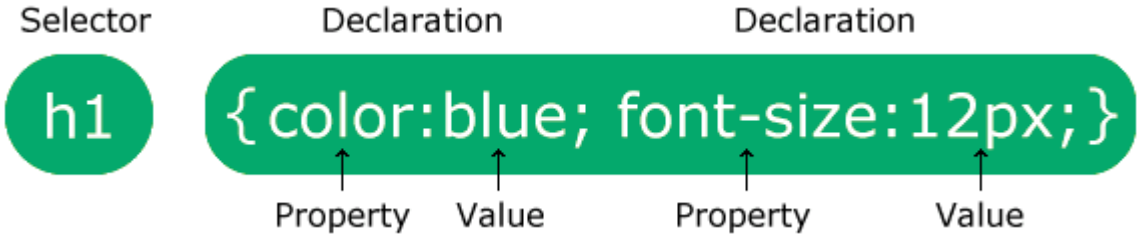
Most of the browsers cache external style sheets. So, once a style sheet is cached, there is no delay in document presentation.

The size of a document using external style sheet is comparatively smaller and hence, download time is also smaller. This speeds up overall response time.

# What is CSS?

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

# CSS Syntax

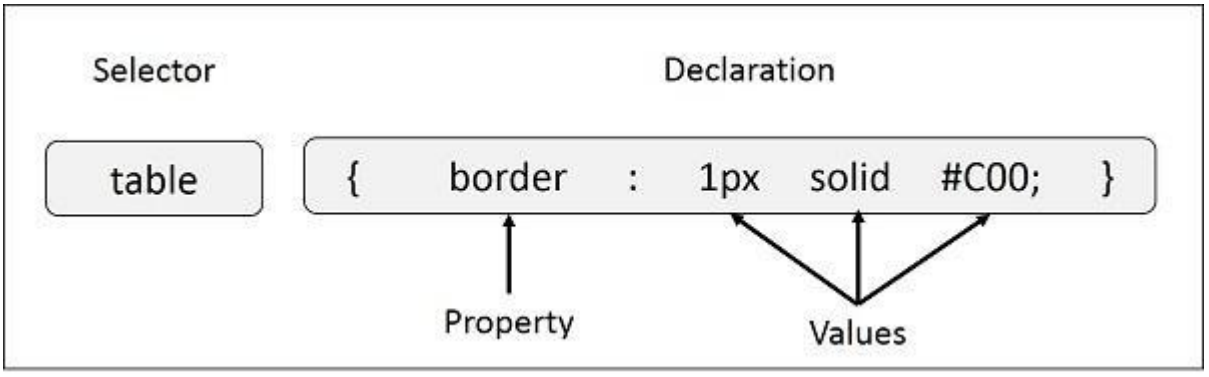



The selector points to the HTML element you want to style.

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.



**Example** − You can define a table border as follows −

table{ border :1px solid #C00; }

Here table is a selector and border is a property and given value *1px solid #C00* is the value of that property.

Or

H1{color:red;}

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  color: red;
  text-align: center;
}
</style>
</head>
<body>

<p>Hello World!</p>
<p>These paragraphs are styled with CSS.</p>

</body>
</html>
```

**Example Explained**

- p is a selector in CSS (it points to the HTML element you want to style: <p>).
- color is a property, and red is the property value
- text-align is a property, and center is the property value

---

# How To Add CSS

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.

**Three Ways to Insert CSS**

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

# 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:

With an external style sheet, you can change the look of an entire website by changing just one file!

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

Example

External styles are defined within the <link> element, inside the <head> section of an HTML page:

DEMO.HTML(FILENAME)

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

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

</body>
</html>
```

An external style sheet can be written in any text editor, and must be saved with a .css extension.

The external .css file should not contain any HTML tags.

Here is how the "mystyle.css" file looks:

"mystyle.css"

```
body {
  background-color: lightblue;
}

h1 {
  color: navy;
  margin-left: 20px;
}

p {
  color: red;
}
```

**Note:** Do not add a space between the property value and the unit (such as margin-left: 20 px;). The correct way is: margin-left: 20px;

## 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:

Example

```
<!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>
```

### Example: 2

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: lightblue;
}

h1 {
  color: white;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}
</style>
</head>
<body>

<h1>My First CSS Example</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

## 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:

```
<!DOCTYPE html>

<html>

<body>

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

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

</body>

</html>
```

---

## CSS Selectors

CSS selectors are used to "find" (or select) the HTML elements you want to style.

You can define selectors in various simple ways based on your comfort. Let me put these selectors one by one.

## The CSS element Selector

The element selector selects HTML elements based on the element name.

Example

Here, all <p> elements on the page will be center-aligned, with a red text color:

```
<!DOCTYPE html>

<html>

<head>

<style>

p {

  text-align: center;

  color: red;
```

```
}
</style>
</head>
<body>

<p>Every paragraph will be affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>

</body>
</html>
```

## The Universal Selectors

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

Example

The CSS rule below will affect every HTML element on the page:

```
* {
  color: #000000;
}
```

This rule renders the content of every element in our document in black.

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
  text-align: center;
  color: blue;
}
</style>
```

```
</head>
<body>

<h1>Hello world!</h1>

<p>Every element on the page will be affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>

</body>
</html>
```

## The Descendant Selectors

Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, style rule will apply to <b> element only when it lies inside <ul> tag.

```
ul b {
  color: #000000;
}
```

## The Class Selectors

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.

**Example**

In this example all HTML elements with class="center" will be red and center-aligned:

```
<!DOCTYPE html>

<html>

<head>

<style>

.center {

  text-align: center;
```

```
  color: red;
}
</style>
</head>
<body>

<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>

</body>
</html>
```

You can also specify that only specific HTML elements should be affected by a class.

**Example**

In this example only <p> elements with class="center" will be red and center-aligned:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.center {
  text-align: center;
  color: red;
}
</style>
</head>
<body>
```

**<h1 class="center">This heading will not be affected</h1>**

** This paragraph will be red and center-aligned. **

**</body>**

**</html>**

# The ID Selectors

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.

**Example**

The CSS rule below will be applied to the HTML element with id="para1":

**<!DOCTYPE html>**

**<html>**

**<head>**

**<style>**

**#para1 {**

**text-align: center;**

**color: red;**

**}**

**</style>**

**</head>**

**<body>**

** Hello World! **

** This paragraph is not affected by the style. **

**</body>**

**</html>**

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

## The Child Selectors

You have seen the descendant selectors. There is one more type of selector, which is very similar to descendants but have different functionality. Consider the following example −

```
body  p {
  color: #000000;
}
```

This rule will render all the paragraphs in black if they are direct child of <body> element. Other paragraphs put inside other elements like  or <td> would not have any effect of this rule.

## The Attribute Selectors

You can also apply styles to HTML elements with particular attributes. The style rule below will match all the input elements having a type attribute with a value of *text* −

```
input[type = "text"] {
  color: #000000;
}
```

The advantage to this method is that the <input type = "submit" /> element is unaffected, and the color applied only to the desired text fields.

## Grouping Selectors

The grouping selector selects all the HTML elements with the same style definitions.

Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

```
h1 {
  text-align: center;
  color: red;
}

h2 {
  text-align: center;
```

```
  color: red;
}

p {
  text-align: center;
  color: red;
}
```

It will be better to group the selectors, to minimize the code.

To group selectors, separate each selector with a comma.

Example

In this example we have grouped the selectors from the code above:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1, h2, p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<h1>Hello World!</h1>
<h2>Smaller heading!</h2>
<p>This is a paragraph.</p>

</body>
</html>
```

---

## CSS Rules Overriding

We have discussed four ways to include style sheet rules in a an HTML document. Here is the rule to override any Style Sheet Rule.

- Any inline style sheet takes highest priority. So, it will override any rule defined in <style>...</style> tags or rules defined in any external style sheet file.
- Any rule defined in <style>...</style> tags will override rules defined in any external style sheet file.
- Any rule defined in external style sheet file takes lowest priority, and rules defined in this file will be applied only when above two rules are not applicable.

## CSS Comments

Many times, you may need to put additional comments in your style sheet blocks. So, it is very easy to comment any part in style sheet. You can simple put your comments inside /*.....this is a comment in style sheet.....*/.

You can use /* ....*/ to comment multi-line blocks in similar way you do in C and C++ programming languages.

Example

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p {
        color: red;
        /* This is a single-line comment */
        text-align: center;
      }
      /* This is a multi-line comment */
    </style>
  </head>

  <body>
    <p>Hello World!</p>
  </body>
</html>
```

## Advantages of CSS

- **CSS saves time** − You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
- **Pages load faster** − If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.

- **Easy maintenance** − To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- **Superior styles to HTML** − CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- **Multiple Device Compatibility** − Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
- **Global web standards** − Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

## CSS Solved a Big Problem

HTML was NEVER intended to contain tags for formatting a web page!

HTML was created to describe the content of a web page, like: <h1>This is a heading</h1>

This is a paragraph.

When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

To solve this problem, the World Wide Web Consortium (W3C) created CSS.

CSS removed the style formatting from the HTML page!

# Cascading Order

What style will be used when there is more than one style specified for an HTML element?

All the styles in a page will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

So, an inline style has the highest priority, and will override external and internal styles and browser defaults.

<!DOCTYPE html>

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
<style>
body {background-color: linen;}
</style>
</head>
<body style="background-color: lavender">

<h1>Multiple Styles Will Cascade into One</h1>
<p>Here, the background color of the page is set with inline CSS, and also with an internal CSS, and also with an external CSS.</p>
<p>Try experimenting by removing styles to see how the cascading stylesheets work (try removing the inline CSS first, then the internal, then the external).</p>

</body>
</html>
```

---

## CSS Border Color

You can set the color of borders:

```
<!DOCTYPE html>
<html>
<body>

<h1 style="border: 2px solid Tomato;">Hello World</h1>

<h1 style="border: 2px solid DodgerBlue;">Hello World</h1>

<h1 style="border: 2px solid Violet;">Hello World</h1>
```

</body>

</html>

Or

```
<h1 style="background-color:rgb(255, 99, 71);">...</h1>
<h1 style="background-color:#ff6347;">...</h1>
<h1 style="background-color:hsl(9, 100%, 64%);">...</h1>
```

## BACKGROUND IMAGE

```
<!DOCTYPE html>

<html>

<head>

<style>

p {

  background-image: url("paper.gif");

}

</style>

</head>

<body>


<h1>Hello World!</h1>


<p>This paragraph has an image as the background!</p>


</body>

</html>
```

## CSS Border Style

The border-style property specifies what kind of border to display.

The following values are allowed:

- dotted - Defines a dotted border

- dashed - Defines a dashed border
- solid - Defines a solid border
- double - Defines a double border
- groove - Defines a 3D grooved border. The effect depends on the border-color value
- ridge - Defines a 3D ridged border. The effect depends on the border-color value
- inset - Defines a 3D inset border. The effect depends on the border-color value
- outset - Defines a 3D outset border. The effect depends on the border-color value
- none - Defines no border
- hidden - Defines a hidden border

The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

Example

Demonstration of the different border styles:

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
```

# CSS LISTS

### The list-style-type Property

```
<html>
  <head>
  </head>

  <body>
    <ul style = "list-style-type:circle;">
      <li>Maths</li>
      <li>Social Science</li>
      <li>Physics</li>
    </ul>

    <ul style = "list-style-type:square;">
      <li>Maths</li>
      <li>Social Science</li>
      <li>Physics</li>
    </ul>
```

```
    <ol style = "list-style-type:decimal;">
      <li>Maths</li>
      <li>Social Science</li>
      <li>Physics</li>
    </ol>

    <ol style = "list-style-type:lower-alpha;">
      <li>Maths</li>
      <li>Social Science</li>
      <li>Physics</li>
    </ol>

    <ol style = "list-style-type:lower-roman;">
      <li>Maths</li>
      <li>Social Science</li>
      <li>Physics</li>
    </ol>
  </body>
</html>
```

# CSS MARGINS

The *margin* property defines the space around an HTML element. It is possible to use negative values to overlap content.

The values of the margin property are not inherited by the child elements. Remember that the adjacent vertical margins (top and bottom margins) will collapse into each other so that the distance between the blocks is not the sum of the margins, but only the greater of the two margins or the same size as one margin if both are equal.

We have the following properties to set an element margin.

- The **margin** specifies a shorthand property for setting the margin properties in one declaration.
- The **margin-bottom** specifies the bottom margin of an element.
- The **margin-top** specifies the top margin of an element.
- The **margin-left** specifies the left margin of an element.
- The **margin-right** specifies the right margin of an element.

Now, we will see how to use these properties with examples.