

Introduction to CSS

Adding CSS to your websites will make them presentable and somewhat like the sites you see on the internet.

- **CSS is a shorthand for Cascading Style Sheets.**
- **CSS describes how the HTML elements are to be displayed.**
- **CSS is used to control the layout of multiple web pages at once.**
- **Includes the addition of visuals such as colour, fonts, layouts, etc.**

The syntax for CSS is:

```
selector {  
    property: values;  
}
```

- Each property ends with a semicolon.
- Each property includes a name for the css property and a value separated by a colon.

Below is an example, to see how CSS modifies the HTML code:

```
<h1>Welcome to Coding !</h1>  
  
<h2>Where coding is a way of life..</h2>
```

Now, adding CSS to the above HTML code:

```
h1{  
    font-family: monospace;  
}  
  
h2{  
    color: blue;  
}
```

The code snippet would now look like this:

CSS COMMENTS

Comments are code that is ignored by the browser. They help make code understandable, hence helping in changes in code that may be required at a later stage easily, by defining the various sections of HTML element styles.

A CSS comment initiates with `/*` and ends with `*/`. Comments can also span multiple lines.

ADDING CSS TO HTML PAGE

The browser formats the HTML document based upon the information in the stylesheet. The browser will access the stylesheets in the HTML document itself. There are 3 ways to add CSS styles to your document:

- Inline Styles
- External Styles
- Internal Styles

Cascading order decides which styles will be applied to elements when multiple styles are used.

Cascading order priority is given as:

Inline > (internal ≡ external) > browser default.

The browser treats both internal and external CSS equally, but the order in which they are defined, determines which property gets priority.

- If the link to internal CSS is defined *before the external CSS*, then properties of external CSS will get preference over internal CSS, i.e. **external CSS > internal CSS**.
- If the link to internal CSS is *defined after the external CSS*, then properties of internal CSS will get the preference, i.e **internal CSS > external CSS**.

Inline Styles

- The **style attribute** is used to apply an inline stylesheet directly to our HTML code.
- The inline stylesheet syntax will have the properties specified inside the style attribute.
- Multiple properties can be specified at a time.
- To **apply a unique style to a single element**, use inline styling.

You can use inline styles like this:

```
<p style="color:blue;font-size:40px">Inline CSS</p>
```

Internal Styles

An **internal** or **in-page** stylesheet:

- It contains the CSS style code for the web page.
- It provides the styles for that particular HTML document only.
- It cannot be reused.
- It can be specified with the help of the **<style>** tag inside the **<head>** tag.

You can use internal styles like this:

```
<style>

  h1 {
    color: blue;
  }

  h2 {
    color: red;
  }

  p {
    color: green;
  }

</style>
```

External styles

- You should use an external style sheet if you want to apply styling to a website using just one file.
- Although the **syntax is similar to internal stylesheets**, it is implemented

using a **separate CSS file**.

- The **'css' extension** is used to save it. Eg. 'styles.css'.

To use an external stylesheet, a reference is provided to file inside the **<link>** element:

```
<link rel="stylesheet" type="text/css" href="styles.css">
```

Attribute	Value
rel	Defines the linked document relationship
href	Specifies where the linked document is located
type	Defines the type of media of the linked document

NOTE: The link is placed within the head. CSS syntax code is contained in the 'styles.css' file only. The name of the file could be anything followed by .css extension, and the same should be specified while linking it to the html file using <link> tag.

SELECTORS

Selectors are used to point to the HTML element that needs to be styled. Selectors are used in both internal and external stylesheets.

Styles are applied using three different types of selectors:

- **Element selector**
- **Class Selector**
- **Id selector**

When multiple styles are applied to an element, specificity determines which style will be used.

The latest rule is applied, if the specificity is the same.

Specificity order:

inline > id selector > class selector > tag selector > browser default

NOTE: If the same property is defined inside the same type of selector, then the property which is defined at the last will be used by the browser.

Element Selector

The element selector will help us to select all elements with the same mentioned element name. This will select all the elements in the HTML document with the given name, but most of the time this is not our requirement. So, to apply styles to only some specific elements we need to have some restrictions. We will take a look at this later in this section only.

Syntax: `element { css declarations; }`

Eg., applying style to h2 tag like this:

```
<h1>Blue Color</h1>
```

and applying CSS like this:

```
h1{  
    color: blue;  
}
```

will show on the browser like this:



Class Selector

Multiple elements with a specific class attribute are selected using the class selector. To select elements with a specific class, type a period (.) followed by the class name.

Syntax: `.class-name {css declarations; }`

To use the class selector, the **class attribute** is used in the element's opening tag. The value of the class attribute contains the name of the class. There can be **multiple classes** added to the tag by giving space in between.

Eg., defining the classes in the HTML file like this:

```
<p class="red">I am red1</p>
```

```
<p class="blue right">I am blue1</p>
<p class="red">I am red2</p>
<p class="blue right">I am blue2</p>
```

The CSS code file

```
.red{
    color: red ;
}
.blue{
    color: blue;
}
.right{
    text-align: right;
}
```

will show on the browser like this:



Id selector

The id selector will help us to select only one element with that specific id. We need to write a hash(#) character and then id name to select an element with a specific id.

Syntax: `#class-name{css declarations;}`

To make use of the id selector, the **id attribute** is defined in the element's opening tag. The value of the id attribute will have the name of the id. The id is **unique** on an HTML page.

There can only be **one id** in the tag. If another element is having the same id, the styles would not be applied by the browser.

Eg., defining the ids in the HTML file like this:

```
<p id="one">This is id one!</p>
<p id="two">This is id two!</p>
<p id="three">This is id three!</p>
<p id="four">This is id four!</p>
```

and applying css like this:

```
#one {  
    color: blue;  
}  
  
#two {  
    background-color: teal;  
}  
  
#three {  
    color: green;  
}  
  
#four {  
    background-color: lightgrey;  
}
```

will show on the browser like this:

This is id one!

This is id two!

This is id three!

This is id four!

Grouping Selectors

We usually use the same CSS for multiple elements, and we can't have too many classes, as too many classes would become difficult to manage.

So, CSS helps us with a grouping feature where you can define the CSS rules to multiple elements with the use of a combination of either class, tag, or id.

We **need to use a comma separator for the different selectors** for grouping

Here are a few examples of how grouping can be used:

- `p, .class-name { CSS properties }` - apply styles to 'para' and element with class as 'class-name'
- `#id1, #id2, span { CSS properties }` - apply styles to 'span' and elements with ids as 'id1' and 'id2'

- `.class-name, #id1, div { CSS properties }` - apply styles to 'div' and elements with class 'class-name' and id as 'id1'.

Nesting Selectors

Whenever we require to target elements inside a particular section of our HTML page. Instead of using the classes there, we can use nesting that works like a hierarchy and is easier to understand.

To use nesting, you need to **add space between the selectors**. Hence the sequence formed represents a **hierarchy starting from the top**.

These are just a few examples:

- `.class-name span { CSS declarations }` - this will apply styles to only those 'span', which are present inside the element with class 'class-name'
- `#id1 .class-name span { CSS declarations }` - this will apply styles to only those 'span', which are present inside the element with class 'class-name' and 'class-name' is inside the element with id 'id1'

Chaining Selectors

There are times when we want to have the same class for multiple elements and we want to apply styles to them. In this scenario, we can use chaining selectors.

To use chaining we take the help of the combination of selectors without putting any space in between them.

Eg., we have a class 'header-style' applied to every heading. We can apply different styles to them like this:

```
<html>
  <head>
    <style>
      .header-style{
        background-color: aqua;
        display: inline-block;
      }
    </style>
  </head>
  <body>
```



```
<h1 class="header-style">Hi</h1>
<h1 class="header-style">Hello</h1>
</body>
</html>
```

Now, as the class **header-style** is given to both **<h1> tags**, we can apply styles to both of them together as done in the above example.