CSS3(Cascading StyleSheet)

CSS stands for Cascading Style Sheets. CSS is a standard style sheet language used for describing the presentation (i.e. the layout and formatting) of the web pages.

Prior to CSS, nearly all of the presentational attributes of HTML documents were contained within the HTML markup (specifically inside the HTML tags); all the font colors, background styles, element alignments, borders and sizes had to be explicitly described within the HTML.

As a result, development of the large websites became a long and expensive process, since the style information were repeatedly added to every single page of the website.

To solve this problem CSS was introduced in 1996 by the World Wide Web Consortium (W3C), which also maintains its standard. CSS was designed to enable the separation of presentation and content.

Advantages of Using CSS

The biggest advantage of CSS is that it allows the separation of style and layout from the content of the document. Here are some more advantages, why one should start using CSS?

- **CSS Save Lots of Time** — CSS gives lots of flexibility to set the style properties of an element. You can write CSS once; and then the same code can be applied to the groups of HTML elements, and can also be reused in multiple HTML pages.
- **Easy Maintenance** — CSS provides an easy means to update the formatting of the documents, and to maintain the consistency across multiple documents. Because the content of the entire set of web pages can be easily controlled using one or more style sheets.
- **Pages Load Faster** — CSS enables multiple pages to share the formatting information, which reduces complexity and repetition in the structural contents of the documents. It significantly reduces the file transfer size, which results in a faster page loading.
- **Superior Styles to HTML** — CSS has much wider presentation capabilities than HTML and provide much better control over the layout of your web pages. So

you can give far better look to your web pages in comparison to the HTML presentational elements and attributes.

- **Multiple Device Compatibility** — CSS also allows web pages to be optimized for more than one type of device or media. Using CSS the same HTML document can be presented in different viewing styles for different rendering devices such as desktop, cell phones, etc.

Types of CSS

- **Inline styles** — Using the `style` attribute in the HTML start tag.
- **Embedded styles** — Using the `<style>` element in the head section of a document.
- **External style sheets** — Using the `<link>` element, pointing to an external CSS file.

Inline Styles

Inline styles are used to apply the unique style rules to an element by putting the CSS rules directly into the start tag. It can be attached to an element using the `style` attribute.

The `style` attribute includes a series of CSS property and value pairs. Each `"property: value"` pair is separated by a semicolon (`;`), just as you would write into an embedded or external style sheets. But it needs to be all in one line i.e. no line break after the semicolon, as shown here:

Example

```
<h1 style="color:red; font-size:30px;">This is a heading</h1>
<p style="color:green; font-size:22px;">This is a paragraph.</p>
```

Embedded Style Sheets

Embedded or internal style sheets only affect the document they are embedded in.

Embedded style sheets are defined in the `<head>` section of an HTML document using the `<style>` element. You can define any number of `<style>` elements in an HTML document but they must appear between the `<head>` and `</head>` tags. Let's take a look at an example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>My HTML Document</title>
    <style>
        body { background-color: YellowGreen; }
        p { color: #fff; }
    </style>
</head>
<body>
    <h1>This is a heading</h1>
    <p>This is a paragraph of text.</p>
</body>
</html>
```

External Style Sheets

An external style sheet is ideal when the style is applied to many pages of the website.

An external style sheet holds all the style rules in a separate document that you can link from any HTML file on your site. External style sheets are the most flexible because with an external style sheet, you can change the look of an entire website by changing just one file.

You can attach external style sheets in two ways — *linking* and *importing*.

Linking External Style Sheets

Before linking, we need to create a style sheet first. Let's open your favorite code editor and create a new file. Now type the following CSS code inside this file and save it as "style.css".

Example
Try this code »

```
body {
    background: lightyellow;
    font: 18px Arial, sans-serif;
}
h1 {
    color: orange;
}
```

An external style sheet can be linked to an HTML document using the `<link>` tag. The `<link>` tag goes inside the `<head>` section, as you can see in the following example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>My HTML Document</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <h1>This is a heading</h1>
    <p>This is a paragraph of text.</p>
</body>
</html>
```

Importing External Style Sheets

The `@import` rule is another way of loading an external style sheet. The `@import` statement instructs the browser to load an external style sheet and use its styles.

You can use it in two ways. The simplest is within the header of your document. Note that, other CSS rules may still be included in the `<style>` element. Here's an example:

```
<style>
    @import url("css/style.css");
    p {
        color: blue;
        font-size: 16px;
    }
</style>
```

Similarly, you can use the `@import` rule to import a style sheet within another style sheet.

```
@import url("css/layout.css");
@import url("css/color.css");
body {
    color: blue;
    font-size: 14px;
}
```

What is Selector?

A CSS selector is a pattern to match the elements on a web page. The style rules associated with that selector will be applied to the elements that match the selector pattern.

Selectors are one of the most important aspects of CSS as they allow you to target specific elements on your web page in various ways so that they can be styled.

Several types of selectors are available in CSS, let's take a closer look at them:

Universal Selector

The universal selector, denoted by an asterisk (`*`), matches every single element on the page.

The universal selector may be omitted if other conditions exist on the element. This selector is often used to remove the default margins and paddings from the elements for quick testing purpose.

Let's try out the following example to understand how it basically works:

Example
Try this code »

```
* {
    margin: 0;
    padding: 0;
}
```

The style rules inside the `*` selector will be applied to every element in a document.

Note: It is recommended *not to use* the universal selector (*) too often in a production environment, since this selector matches every element on a web page that puts too much of unnecessary pressure on the browsers. Use element type or class selector instead.

Element Type Selectors

An element type selector matches all instance of the element in the document with the corresponding element type name. Let's try out an example to see how it actually works:

Example
Try this code »

```
p {
    color: blue;
}
```

The style rules inside the `p` selector will be applied on every `<p>` element (or paragraph) in the document and color it blue, regardless of their position in the document tree.

Id Selectors

The id selector is used to define style rules for a *single* or *unique* element.

The id selector is defined with a hash sign (`#`) immediately followed by the id value.

Example
Try this code »

```
#error {
    color: red;
}
```

This style rule renders the text of an element in red, whose `id` attribute is set to `error`.

Class Selectors

The class selectors can be used to select any HTML element that has a `class` attribute. All the elements having that class will be formatted according to the defined rule.

The class selector is defined with a period sign (`.`) immediately followed by the class value.

Example
Try this code »

```
.blue {
    color: blue;
}
```

The above style rules renders the text in blue of every element in the document that has `class` attribute set to `blue`. You can make it a bit more particular. For example:

Example
Try this code »

```
p.blue {
    color: blue;
}
```

The style rule inside the selector `p.blue` renders the text in blue of only those `<p>` elements that has `class` attribute set to `blue`, and has no effect on other paragraphs.

Descendant Selectors

You can use these selectors when you need to select an element that is the descendant of another element, for example, if you want to target only those anchors that are contained within an unordered list, rather than targeting all anchor elements. Let's see how it works:

Example
Try this code »

```
ul.menu li a {
    text-decoration: none;
}
h1 em {
    color: green;
}
```

The style rules inside the selector `ul.menu li a` applied to only those `<a>` elements that contained inside an `<ul>` element having the class `.menu`, and has no effect on other links inside the document.

Similarly, the style rules inside the `h1 em` selector will be applied to only those `<em>` elements that contained inside the `<h1>` element and has not effect on other `<em>` elements.

Child Selectors

A child selector is used to select only those elements that are the direct children of some element.

A child selector is made up of two or more selectors separated by a greater than symbol (`>`). You can use this selector, for instance, to select the first level of list elements inside a nested list that has more than one level. Let's check out an example to understand how it works:

Example
Try this code »

```
ul > li {
    list-style: square;
}
ul > li ol {
    list-style: none;
}
```

The style rule inside the selector `ul > li` applied to only those `<li>` elements that are direct children of the `<ul>` elements, and has no effect on other list elements.

Adjacent Sibling Selectors

The adjacent sibling selectors can be used to select sibling elements (i.e. elements at the same level). This selector has the syntax like: E1 + E2, where E2 is the target of the selector.

The selector `h1 + p` in the following example will select the `<p>` elements only if both the `<h1>` and `<p>` elements share the same parent in the document tree and `<h1>` is immediately precedes the `<p>` element. That means only those paragraphs that come immediately after each `<h1>` heading will have the associated style rules. Let's see how this selector actually works:

Example
Try this code »

```
h1 + p {
    color: blue;
    font-size: 18px;
}
ul.task + p {
    color: #f0f;
    text-indent: 30px;
}
```

Grouping Selectors

Often several selectors in a style sheet share the same style rules declarations. You can group them into a comma-separated list to minimize the code in your style sheet. It also prevents you from repeating the same style rules over and over again. Let's take a look:

Example

Try this code »

```
h1 {
    font-size: 36px;
    font-weight: normal;
}
h2 {
    font-size: 28px;
    font-weight: normal;
}
```

As you can see in the above example, the same style rule `font-weight: normal;` is shared by the selectors `h1`, `h2` and `h3`, so it can be grouped in a comma-separated list, like this:

```
h1, h2, h3 {
    font-weight: normal;
}
h1 {
    font-size: 36px;
}
h2 {
    font-size: 28px;
}
h3 {
    font-size: 22px;
}
```

CSS for Background

```
#multibackground {
   background-image: url(/css/images/logo.png),
url(/css/images/border.png);
   background-position: left top, left top;
   background-repeat: no-repeat, repeat;
   padding: 75px;
}
```

Sr.No.	Value & Description
1	**background** Used to setting all the background image properties in one section
2	**background-clip** Used to declare the painting area of the background
3	**background-image** Used to specify the background image
4	**background-origin** Used to specify position of the background images
5	**background-size** Used to specify size of the background images

Styling Fonts with CSS

Choosing the right font and style is very crucial for the readability of text on a page.

CSS provide several properties for styling the font of the text, including changing their face, controlling their size and boldness, managing variant, and so on.

The font properties are: `font-family`, `font-style`, `font-weight`, `font-size`, and `font-variant`.

Let's discuss each of these font properties one by one in more detail.

Font Family

The `font-family` property is used to specify the font to be used to render the text.

This property can hold several comma-separated font names as a *fallback* system, so that if the first font is not available on the user's system, browser tries to use the second one, and so on.

Hence, list the font that you want first, then any fonts that might fill in for the first if it is not available. You should end the list with a generic font family which are five — `serif`, `sans-serif`, `monospace`, `cursive` and `fantasy`. A typical font family declaration might look like this:

```css
body {
    font-family: Arial, Helvetica, sans-serif;
}
```

Font Style

The `font-style` property is used to set the font face style for the text content of an element.

The font style can be `normal`, `italic` or `oblique`. The default value is `normal`.

Let's try out the following example to understand how it basically works:

Example
Try this code »

```
p.normal {
  font-style: normal;
}
p.italic {
  font-style: italic;
}
p.oblique {
  font-style: oblique;
}
```

Font Size

The `font-size` property is used to set the size of font for the text content of an element.

There are several ways to specify the font size values e.g. with keywords, percentage, pixels, ems, etc.

Setting Font Size with Pixels

Setting the font size in pixel values (e.g. 14px, 16px, etc.) is a good choice when you need the pixel accuracy. Pixel is an absolute unit of measurement which specifies a fixed length.

Let's try out the following example to understand how it basically works:

```
h1 {
    font-size: 24px;
}
p {
    font-size: 14px;
}
```

Defining the font sizes in pixel is not considered very accessible, because the user cannot change the font size from the browser settings. For instance, users with limited or low vision may wish to set the font size much larger than the size specified by you.

Therefore, you should avoid using the pixels values and use the values that are relative to the user's default font size instead if you want to create an inclusive design.

Font Weight

The `font-weight` property specifies the weight or boldness of the font.

This property can take one of the following values: `normal`, `bold`, `bolder`, `lighter`, `100`, `200`, `300`, `400`, `500`, `600`, `700`, `800`, `900` and `inherit`.

- The numeric values `100`-`900` specify the font weights, where each number represents a weight greater than its predecessor. `400` is same as `normal` & `700` is same as `bold`.
- The `bolder` and `lighter` values are relative to the inherited font weight, while the other values such as `normal` and `bold` are absolute font weights.

Let's try out an example to understand how this property basically works:

```
p {
    font-weight: bold;
}
```

Font Variant

The `font-variant` property allows the text to be displayed in a special small-caps variation.

Small-caps or *small capital letters* are slightly different to normal capital letters, in which lowercase letters appear as smaller versions of the corresponding uppercase letters.

Try out the following example to understand how this property actually works:

```
p {
    font-variant: small-caps;
}
```

CSS Lists

In this tutorial you will learn how to format HTML lists using CSS.

Types of HTML Lists

There are three different types of list in HTML:

- **Unordered lists** — A list of items, where every list items are marked with bullets.
- **Ordered lists** — A list of items, where each list items are marked with numbers.
- **Definition list** — A list of items, with a description of each item.

See the tutorial on HTML lists to learn more about the lists and how to create them.

Styling Lists with CSS

CSS provides the several properties for styling and formatting the most commonly used unordered and ordered lists. These CSS list properties typically allow you to:

- Control the shape or appearance of the marker.
- Specify an image for the marker rather than a bullet point or number.
- Set the distance between a marker and the text in the list.
- Specify whether the marker would appear inside or outside of the box containing the list items.

In the following section we will discuss the properties that can be used to style HTML lists.

```
ul {
    list-style-type: square;
}
ol {
    list-style-type: upper-roman;
}
```

Changing the Position of List Markers

By default, markers of each list items are positioned `outside` of their display boxes.

```
ol.in li {
    list-style-position: inside;
}
ol.out li {
    list-style-position: outside;
}
```

```
ul li { list-style-image: url("images/bullet.png"); }
```

```
Table
```

Adding Borders to Tables

The CSS `border` property is the best way to define the borders for the tables.

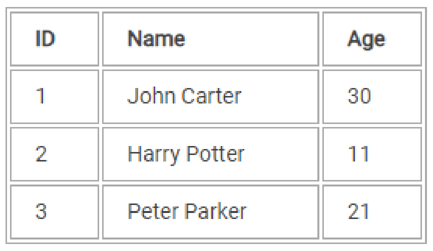
The following example will set a black border for the `<table>`, `<th>`, and `<td>` elements.

Example
Try this code »

```
table, th, td {
    border: 1px solid black;
}
```

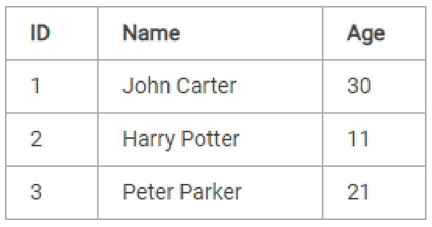
By default, browser draws a border around the table, as well as around all the cells, with some space in-between, which results in double border. To get rid of this double border problem you can simply collapse the adjoining table cell borders and create clean single line borders.

Let's take a look at the following illustration to understand how a border is applied to a table.



ID	Name	Age
1	John Carter	30
2	Harry Potter	11
3	Peter Parker	21

Separate Border (Default)



ID	Name	Age
1	John Carter	30
2	Harry Potter	11
3	Peter Parker	21

Collapse Border

Collapsing Table Borders

There are two distinct models for setting borders on table cells in CSS: *separate* and *collapse*.

In the separate border model, which is the default, each table cell has its own distinct borders, whereas in the collapsed border model, adjacent table cells share a common border. You can set the border model for an HTML table by using the CSS `border-collapse` property.

The following style rules will collapse the table cell borders and apply one pixel black border.

Example
Try this code »

```
table {
    border-collapse: collapse;
}
th, td {
    border: 1px solid black;
}
```

Adjusting Space inside Tables

By default, the browser creates the table cells just large enough to contain the data in the cells.

To add more space between the table cell contents and the cell borders, you can simply use the CSS `padding` property. Let's try out the following example and see how it works:

```
th, td {
    padding: 15px;
}
```

You can also adjust the spacing between the borders of the cells using the CSS `border-spacing` property, if the borders of your table are separated (which is default).

The following style rules apply the spacing of 10 pixels between all borders within a table:

```
table {
    border-spacing: 10px;
}
```

Setting Table Width and Height

By default, a table will render just wide and tall enough to contain all of its contents.

However, you can also set the width and height of the table as well as its cells explicitly using the `width` and `height` CSS property. The style rules in the following example will sets the width of the table to 100%, and the height of the table header cells to 40px.

Example
Try this code »

```
table {
    width: 100%;
}
th {
    height: 40px;
}
```

Controlling the Table Layout

A table expands and contracts to accommodate the data contained inside it. This is the default behavior. As data fills inside the table, it continues to expand as long as there is space. Sometimes, however, it is necessary to set a fixed width for the table in order to manage the layout.

You can do this with the help of CSS `table-layout` property. This property defines the algorithm to be used to layout the table cells, rows, and columns. This property takes one of two values:

- **auto** — Uses an automatic table layout algorithm. With this algorithm, the widths of the table and its cells are adjusted to fit the content. This is the default value.
- **fixed** — Uses the fixed table layout algorithm. With this algorithm, the horizontal layout of the table does not depend on the contents of the cells; it only depends on the table's width, the width of the columns, and borders or cell spacing. It is normally faster than auto.

```
table {
    width: 300px;
    table-layout: fixed;
}
th { text-align: left; }
```

```
th {
    height: 40px;
    vertical-align: bottom;
}
```