Why Use 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

Why to Learn CSS?

Cascading Style Sheets, fondly referred to as **CSS**, is a simple design language intended to simplify the process of making web pages presentable.

CSS is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning CSS:

- **Create Stunning Web site** - CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.
- **Become a web designer** - If you want to start a career as a professional web designer, HTML and CSS designing is a must skill.
- **Control web** - 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.
- **Learn other languages** - Once you understand the basic of HTML and CSS then other related technologies like JavaScript, php, or angular are become easier to understand.

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

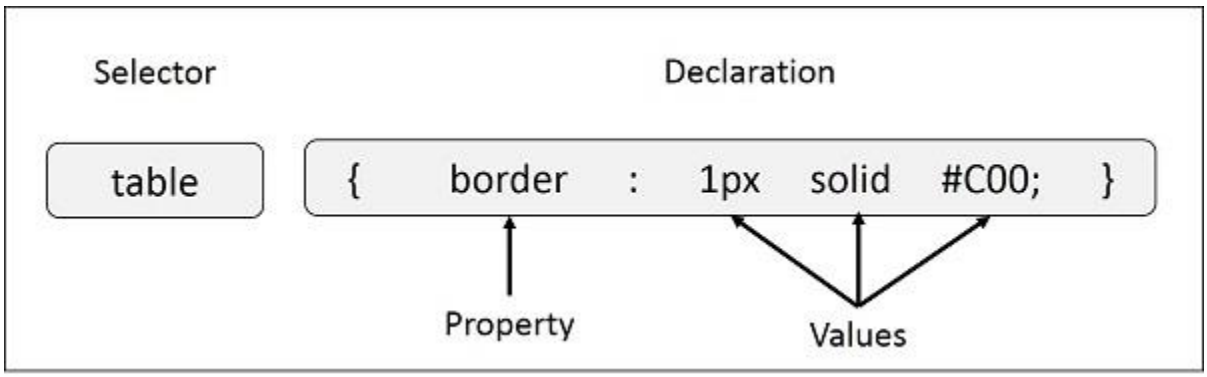
A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts –

- **Selector** – A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.
- **Property** – A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color, border etc.
- **Value** – Values are assigned to properties. For example, color property can have value either red or #F1F1F1 etc.

CSS Selectors

You can put CSS Style Rule Syntax as follows –

```
selector { property: value }
```



Example – You can define a table border as follows –

```
table{ border :1px solid #C00; }
```

The Type Selectors

This is the same selector we have seen above. Again, one more example to give a color to all level 1 headings –

```
h1 {
   color: #36CFFF;
}
```

The CSS element or Tag Selector

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

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

The Universal Selectors

Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type −

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

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

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 element only when it lies inside <ul> tag.

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

The Class Selectors

You can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

```
.black {
   color: #000000;
}
```

This rule renders the content in black for every element with class attribute set to *black* in our document. You can make it a bit more particular. For example −

```
h1.black {
   color: #000000;
}
```

This rule renders the content in black for only <h1> elements with class attribute set to *black*.

You can apply more than one class selectors to given element. Consider the following example −

```
<p class = "center bold">
   This para will be styled by the classes center and bold.
</p>
```

The ID Selectors

You can define style rules based on the *id* attribute of the elements. All the elements having that *id* will be formatted according to the defined rule.

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

This rule renders the content in black for every element with *id* attribute set to *black* in our document. You can make it a bit more particular. For example −

```
h1#black {
   color: #000000;
}
```

This rule renders the content in black for only <h1> elements with *id* attribute set to *black*.

The true power of *id* selectors is when they are used as the foundation for descendant selectors, For example −

```
#black h2 {
   color: #000000;
}
```

In this example all level 2 headings will be displayed in black color when those headings will lie with in tags having *id* attribute set to *black*.

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;
}

or


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

CSS Properties (Ordered by Categories)

The following section contains a complete list of standard properties belonging to the latest CSS3 specifications. All the properties are listed alphabetically.

Property	Description
align-content	Specifies the alignment of flexible container's items within the flex container.
align-items	Specifies the default alignment for items within the flex container.
align-self	Specifies the alignment for selected items within the flex container.
animation	Specifies the keyframe-based animations.
animation-delay	Specifies when the animation will start.
animation-direction	Specifies whether the animation should play in reverse on alternate cycles or not.
animation-duration	Specifies the number of seconds or milliseconds an animation should take to complete one cycle.

Property	Description
animation-fill-mode	Specifies how a CSS animation should apply styles to its target before and after it is executing.
animation-iteration-count	Specifies the number of times an animation cycle should be played before stopping.
animation-name	Specifies the name of @keyframes defined animations that should be applied to the selected element.
animation-play-state	Specifies whether the animation is running or paused.
animation-timing-function	Specifies how a CSS animation should progress over the duration of each cycle.
backface-visibility	Specifies whether or not the "back" side of a transformed element is visible when facing the user.
background	Defines a variety of background properties within one declaration.
background-attachment	Specify whether the background image is fixed in the viewport or scrolls.
background-clip	Specifies the painting area of the background.
background-color	Defines an element's background color.
background-image	Defines an element's background image.
background-origin	Specifies the positioning area of the background images.
background-position	Defines the origin of a background image.
background-repeat	Specify whether/how the background image is tiled.
background-size	Specifies the size of the background images.

Property	Description
`border`	Sets the width, style, and color for all four sides of an element's border.
`border-bottom`	Sets the width, style, and color of the bottom border of an element.
`border-bottom-color`	Sets the color of the bottom border of an element.
`border-bottom-left-radius`	Defines the shape of the bottom-left border corner of an element.
`border-bottom-right-radius`	Defines the shape of the bottom-right border corner of an element.
`border-bottom-style`	Sets the style of the bottom border of an element.
`border-bottom-width`	Sets the width of the bottom border of an element.
`border-collapse`	Specifies whether table cell borders are connected or separated.
`border-color`	Sets the color of the border on all the four sides of an element.
`border-image`	Specifies how an image is to be used in place of the border styles.
`border-image-outset`	Specifies the amount by which the border image area extends beyond the border box.
`border-image-repeat`	Specifies whether the image-border should be repeated, rounded or stretched.
`border-image-slice`	Specifies the inward offsets of the image-border.
`border-image-source`	Specifies the location of the image to be used as a border.
`border-image-width`	Specifies the width of the image-border.

Property	Description
`border-left`	Sets the width, style, and color of the left border of an element.
`border-left-color`	Sets the color of the left border of an element.
`border-left-style`	Sets the style of the left border of an element.
`border-left-width`	Sets the width of the left border of an element.
`border-radius`	Defines the shape of the border corners of an element.
`border-right`	Sets the width, style, and color of the right border of an element.
`border-right-color`	Sets the color of the right border of an element.
`border-right-style`	Sets the style of the right border of an element.
`border-right-width`	Sets the width of the right border of an element.
`border-spacing`	Sets the spacing between the borders of adjacent table cells.
`border-style`	Sets the style of the border on all the four sides of an element.
`border-top`	Sets the width, style, and color of the top border of an element.
`border-top-color`	Sets the color of the top border of an element.
`border-top-left-radius`	Defines the shape of the top-left border corner of an element.
`border-top-right-radius`	Defines the shape of the top-right border corner of an element.
`border-top-style`	Sets the style of the top border of an element.

Property	Description
`border-top-width`	Sets the width of the top border of an element.
`border-width`	Sets the width of the border on all the four sides of an element.
`bottom`	Specify the location of the bottom edge of the positioned element.
`box-shadow`	Applies one or more drop-shadows to the element's box.
`box-sizing`	Alter the default CSS box model.
`caption-side`	Specify the position of table's caption.
`clear`	Specifies the placement of an element in relation to floating elements.
`clip`	Defines the clipping region.
`color`	Specify the color of the text of an element.
`column-count`	Specifies the number of columns in a multi-column element.
`column-fill`	Specifies how columns will be filled.
`column-gap`	Specifies the gap between the columns in a multi-column element.
`column-rule`	Specifies a straight line, or "rule", to be drawn between each column in a multi-column element.
`column-rule-color`	Specifies the color of the rules drawn between columns in a multi-column layout.
`column-rule-style`	Specifies the style of the rule drawn between the columns in a multi-column layout.

Property	Description
column-rule-width	Specifies the width of the rule drawn between the columns in a multi-column layout.
column-span	Specifies how many columns an element spans across in a multi-column layout.
column-width	Specifies the optimal width of the columns in a multi-column element.
columns	A shorthand property for setting column-width and column-count properties.
content	Inserts generated content.
counter-increment	Increments one or more counter values.
counter-reset	Creates or resets one or more counters.
cursor	Specify the type of cursor.
direction	Define the text direction/writing direction.
display	Specifies how an element is displayed onscreen.
empty-cells	Show or hide borders and backgrounds of empty table cells.
flex	Specifies the components of a flexible length.
flex-basis	Specifies the initial main size of the flex item.
flex-direction	Specifies the direction of the flexible items.
flex-flow	A shorthand property for the flex-direction and the flex-wrap properties.

Property	Description
`flex-grow`	Specifies how the flex item will grow relative to the other items inside the flex container.
`flex-shrink`	Specifies how the flex item will shrink relative to the other items inside the flex container.
`flex-wrap`	Specifies whether the flexible items should wrap or not.
`float`	Specifies whether or not a box should float.
`font`	Defines a variety of font properties within one declaration.
`font-family`	Defines a list of fonts for element.
`font-size`	Defines the font size for the text.
`font-size-adjust`	Preserves the readability of text when font fallback occurs.
`font-stretch`	Selects a normal, condensed, or expanded face from a font.
`font-style`	Defines the font style for the text.
`font-variant`	Specify the font variant.
`font-weight`	Specify the font weight of the text.
`height`	Specify the height of an element.
`justify-content`	Specifies how flex items are aligned along the main axis of the flex container after any flexible lengths and auto margins have been resolved.
`left`	Specify the location of the left edge of the positioned element.
`letter-spacing`	Sets the extra spacing between letters.

Property	Description
`line-height`	Sets the height between lines of text.
`list-style`	Defines the display style for a list and list elements.
`list-style-image`	Specifies the image to be used as a list-item marker.
`list-style-position`	Specifies the position of the list-item marker.
`list-style-type`	Specifies the marker style for a list-item.
`margin`	Sets the margin on all four sides of the element.
`margin-bottom`	Sets the bottom margin of the element.
`margin-left`	Sets the left margin of the element.
`margin-right`	Sets the right margin of the element.
`margin-top`	Sets the top margin of the element.
`max-height`	Specify the maximum height of an element.
`max-width`	Specify the maximum width of an element.
`min-height`	Specify the minimum height of an element.
`min-width`	Specify the minimum width of an element.
`opacity`	Specifies the transparency of an element.
`order`	Specifies the order in which a flex items are displayed and laid out within a flex container.

Property	Description
`outline`	Sets the width, style, and color for all four sides of an element's outline.
`outline-color`	Sets the color of the outline.
`outline-offset`	Set the space between an outline and the border edge of an element.
`outline-style`	Sets a style for an outline.
`outline-width`	Sets the width of the outline.
`overflow`	Specifies the treatment of content that overflows the element's box.
`overflow-x`	Specifies the treatment of content that overflows the element's box horizontally.
`overflow-y`	Specifies the treatment of content that overflows the element's box vertically.
`padding`	Sets the padding on all four sides of the element.
`padding-bottom`	Sets the padding to the bottom side of an element.
`padding-left`	Sets the padding to the left side of an element.
`padding-right`	Sets the padding to the right side of an element.
`padding-top`	Sets the padding to the top side of an element.
`page-break-after`	Insert a page breaks after an element.
`page-break-before`	Insert a page breaks before an element.
`page-break-inside`	Insert a page breaks inside an element.

Property	Description
perspective	Defines the perspective from which all child elements of the object are viewed.
perspective-origin	Defines the origin (the vanishing point for the 3D space) for the perspective property.
position	Specifies how an element is positioned.
quotes	Specifies quotation marks for embedded quotations.
resize	Specifies whether or not an element is resizable by the user.
right	Specify the location of the right edge of the positioned element.
tab-size	Specifies the length of the tab character.
table-layout	Specifies a table layout algorithm.
text-align	Sets the horizontal alignment of inline content.
text-align-last	Specifies how the last line of a block or a line right before a forced line break is aligned when text-align is justify.
text-decoration	Specifies the decoration added to text.
text-decoration-color	Specifies the color of the text-decoration-line.
text-decoration-line	Specifies what kind of line decorations are added to the element.
text-decoration-style	Specifies the style of the lines specified by the text-decoration-line property
text-indent	Indent the first line of text.

Property	Description
text-justify	Specifies the justification method to use when the text-align property is set to justify.
text-overflow	Specifies how the text content will be displayed, when it overflows the block containers.
text-shadow	Applies one or more shadows to the text content of an element.
text-transform	Transforms the case of the text.
top	Specify the location of the top edge of the positioned element.
transform	Applies a 2D or 3D transformation to an element.
transform-origin	Defines the origin of transformation for an element.
transform-style	Specifies how nested elements are rendered in 3D space.
transition	Defines the transition between two states of an element.
transition-delay	Specifies when the transition effect will start.
transition-duration	Specifies the number of seconds or milliseconds a transition effect should take to complete.
transition-property	Specifies the names of the CSS properties to which a transition effect should be applied.
transition-timing-function	Specifies the speed curve of the transition effect.
vertical-align	Sets the vertical positioning of an element relative to the current text baseline.
visibility	Specifies whether or not an element is visible.

Property	Description
white-space	Specifies how white space inside the element is handled.
width	Specify the width of an element.
word-break	Specifies how to break lines within words.
word-spacing	Sets the spacing between words.
word-wrap	Specifies whether to break words when the content overflows the boundaries of its container.
z-index	Specifies a layering or stacking order for positioned elements.

Conversion of Table to CSS Layout

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
  border-collapse: collapse;
  border: 1px solid black;
}

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

```
table.a {
  table-layout: auto;
  width: 180px;
}

table.b {
  table-layout: fixed;
  width: 180px;
}

table.c {
  table-layout: auto;
  width: 100%;
}

table.d {
  table-layout: fixed;
  width: 100%;
}
</style>
</head>
<body>

<h1>The table-layout Property</h1>

<table class="a">
  <tr>
    <th>Company</th>
    <th>Contact</th>
    <th>Country</th>
  </tr>
```

```
  <tr>
    <td>ABC</td>
    <td>XYZ  ---- XYZ</td>
    <td>Germany</td>
  </tr>
  <tr>
    <td>PQR</td>
    <td>Person PQR</td>
    <td>UK</td>
  </tr>
  <tr>
    <td>KCES's IMR, Jalgaon</td>
    <td>Computer Depatrment</td>
    <td>India</td>
  </tr>
</table>
</body>
</html>
```

Types of CSS (Cascading Style Sheet)

There are three types of CSS which are given below:

- Inline CSS
- Internal or Embedded CSS
- External CSS

Inline CSS: Inline CSS contains the CSS property in the body section attached with element is known as inline CSS. This kind of style is specified within an HTML tag using the style attribute.

Example:

```
<!DOCTYPE html>

<html>

   <head>

      <title>Inline CSS</title>

   </head>



   <body>

      <p style = "color:#009900; font-size:50px;

            font-style:italic; text-align:center;">

                              Inline CSS Demo

                  </p>



   </body>

</html>
```

Internal or Embedded CSS: This can be used when a single HTML document must be styled uniquely. The CSS rule set should be within the HTML file in the head section i.e the CSS is embedded within the HTML file.

Example:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Internal CSS</title>
        <style>
            .main {
                text-align:center;
            }
            .GFG {
                color:#009900;
                font-size:50px;
                font-weight:bold;
            }
            .geeks {
                font-style:bold;
                font-size:20px;
            }
        </style>
    </head>
    <body>
        <div class = "main">
```

```
                <div class ="GFG">Internal CSS Demo</div>


                <div class ="geeks">
                        A Design Demo
                </div>
            </div>
        </body>
</html>
```

External CSS: External CSS contains separate CSS file which contains only style property with the help of tag attributes (For example class, id, heading, etc.) CSS property written in a separate file with .css extension and should be linked to the HTML document using **link** tag. This means that for each element, style can be set only once and that will be applied across web pages.

Example:

```
<!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>
```

mystyle.css code-

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

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

CSS Menu Design (Horizontal, Vertical)

- Vertical

```
<!DOCTYPE html>

<html>


<head>


        <!--Import Pure Css files-->

        <link rel="stylesheet" href=

"https://unpkg.com/purecss@1.0.0/build/pure-min.css">

        <!-- Let browser know website is

                optimized for mobile -->

                <meta name="viewport" content=

                "width=device-width, initial-scale=1.0" />


</head>

<body>
```

```
        <div class="pure-menu">
                <!--Main heading of menu-->
                <ul class="pure-menu-list">
                        <!--List items of menu-->
                        <li class="pure-menu-item">
                                <a href="#" class="pure-menu-link">
                                        Home
                                </a>
                        </li>
                        <li class="pure-menu-item">
                                <a href="#" class="pure-menu-link">
                                        About Us
                                </a>
                        </li>
                        <li class="pure-menu-item">
                                <a href="#" class="pure-menu-link">
                                        Contact
                                </a>
                        </li>
                        <li class="pure-menu-item">
                                <a href="#" class="pure-menu-link">
                                        Privacy Policy
                                </a>
                        </li>
                </ul>
        </div>

</body>

</html>
```

OR

```
<html>

        <head>
                <style>
                        ul {

        margin: 0;
        padding: 0;
        width: 500px;
        background-color: #f1f1f1;
        }
                                li a {
                         display : block;
        color: #000;
        padding: 8px 16px 10px 16px;
        text-decoration: underline dotted red;
        }


      li a:hover {
        background-color: #555;
        color: white;
        }


                </style>
        </head>

        <body>
```

```
            <div>
                <ul>
                    <li><a href="#home">Home</a></li>
                    <li><a href="#About">About</a></li>
                    <li><a href="#Customers">Customers</a></li>
                </ul>
            </div>
        </body>
</html>
```

- Horizontal

```
<!DOCTYPE html>
<html>

<head>

    <!--Import Pure Css files-->
    <link rel="stylesheet" href=
"https://unpkg.com/purecss@1.0.0/build/pure-min.css"
        integrity=
"sha384-nn4HPE8lTHyVtfCBi5yW9d20FjT8BJwUXyWZT9InLYax14RDjBj46LmSztkmNP9w"
        crossorigin="anonymous">

    <!-- Let browser know website is
        optimized for mobile -->
        <meta name="viewport" content=
        "width=device-width, initial-scale=1.0" />

</head>

<body>
```

```
<div class="pure-menu pure-menu-horizontal">

    <!--Main heading of menu-->

    <ul class="pure-menu-list">

        <!--List items of menu-->
        <li class="pure-menu-item">
            <a href="#" class="pure-menu-link">
                Home
            </a>
        </li>
        <li class="pure-menu-item">
            <a href="#" class="pure-menu-link">
                About Us
            </a>
        </li>
        <li class="pure-menu-item">
            <a href="#" class="pure-menu-link">
                Contact
            </a>
        </li>
        <li class="pure-menu-item">
            <a href="#" class="pure-menu-link">
                Privacy Policy
            </a>
        </li>

    </ul>
```

```
        </div>

</body>

</html>
```

OR

```
<!DOCTYPE html>
<html>
        <head>
                <style>
                        ul {
                        list-style-type: none;
                        margin: 0;
                        padding: 0;
                        overflow: hidden;
                        background-color: #333;
                        }

                        li {
                                float: left;
                        }

                        li a {
                                display: block;
                                color: white;
                                text-align: center;
                                padding: 14px 16px;
                                text-decoration: none;
                        }
```

```
                li a:hover {
                        background-color: #111;
                }
            </style>
        </head>
        <body>

                <ul>
                        <li><a class="active" href="#home">Home</a></li>
                        <li><a href="#news">News</a></li>
                        <li><a href="#contact">Contact</a></li>
                        <li><a href="#about">About</a></li>
                </ul>

        </body>
</html>
```

CSS 3 Selectors

- CSS3 Selectors includes all **css selectors** till CSS3. **CSS3 Selectors list** includes **Retional Selectors**, **Combinators**, **Grouping** and **Pseudo Selectors**.
- Earlier CSS versions use **Type**, **class**, **id** and **child selectors**. CSS 2 adds more **pseudo-elements**, **pseudo-classes**, and **combinators selectors**. And now With CSS3, we can target almost any element with **wide range of selectors** options.
- **CSS3 Selectors** are part of HTML5.

Rational Selectors or Combinators

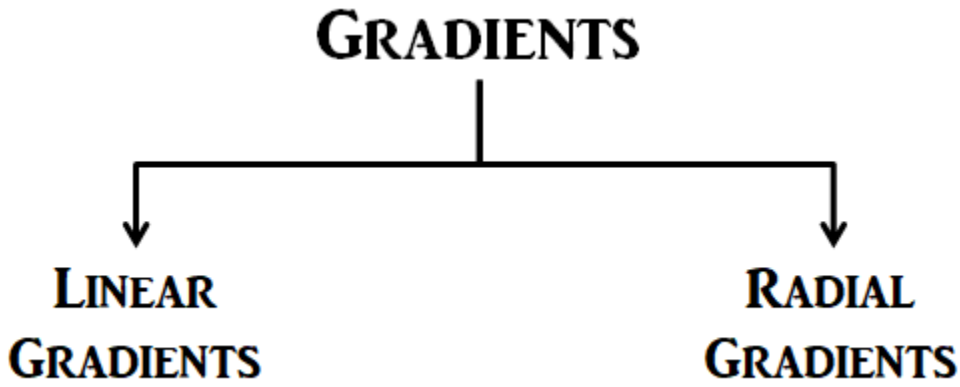
Rational Selectors or **Combinators** targets elements based on their relationship with another elements. These selectors were first introduced in CSS1 and then again in css2 and css3.

CSS Rational Selectors or Combinators			
Selector	**CSS Code**	**Use**	**CSS Level**
Descendant Selector	div p{ }	Target all p tags inside div. That includes all p elements that are **descendant** (*child, grandchild, great-grandchild, and so on)* of div.	1
Child Selector	div>p{ }	This selector matches only **direct child** of element	2
Adjacent Sibling	h4+p{ }	This selector matches p tag next to h4 and both sharing **same parent**. This means p should come directly after h4.	2
General Sibling	h3~p{ }	This selector matches all P elements exactly after h3 element.	

CSS Gradients

The **Gradient** in CSS is a special type of image that is made up of progressive & smooth transition between two or more colors. CSS is the way to add style to various web documents. By using the gradient in CSS, we can create variants styling of images which can help to make an attractive webpage.

The Gradients can be categorized into 2 types:



Linear Gradients: It includes the smooth color transitions to going up, down, left, right, and diagonally. The minimum two-color required to create a linear gradient. More than two color elements can be possible in linear gradients. The starting point and the direction are needed for the gradient effect.

Syntax:

```
background-image: linear-gradient(direction, color-stop1, color-stop2, ...);
```

The linear-gradient can be implemented in the following ways:

Top to Bottom: In this image, the transition started with white color and ended with green color. On exchanging the color sequence, the transition will start with green and will end with white.

Example:

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS Gradients</title>
    <style>
    #main {
        height: 200px;
        background-color: white;
        background-image: linear-gradient(white, #ADD8E6);
or
         background-image: linear-gradient(to top, white, #ADD8E6);
or
         background-image: linear-gradient(to right, white, #ADD8E6);
    }

    .txt_clr {
        text-align: center;
        font-size: 40px;
        font-weight: bold;
        padding-top: 80px;
    }

    .sub_txt {
        font-size: 17px;
        text-align: center;
    }
    </style>
</head>

<body>
    <div id="main">
        <div class="txt_clr">KCES's IMR, Jalgaon</div>
        <div class="sub_txt">

                    Computer Department
        </div>
    </div>
</body>
</html>
```

Using Angles

If you want more control over the direction of the gradient, you can define an angle, instead of the predefined directions (to bottom, to top, to right, to left, to bottom right, etc.). A value of 0deg is equivalent to "to top". A value of 90deg is equivalent to "to right". A value of 180deg is equivalent to "to bottom".

Syntax:

```
background-image: linear-gradient(angle, color-stop1, color-stop2);
```

Example:

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS Gradients</title>
    <style>
    #main {
        height: 200px;
        background-color: white;
        background-image: linear-gradient(180deg, #778899, #DCDCDC);
    }

    .txt_clr {
        text-align: center;
        font-size: 40px;
        font-weight: bold;
        padding-top: 80px;
    }

    .sub_txt {
        font-size: 17px;
        text-align: center;
    }
    </style>
</head>

<body>
    <div id="main">
        <div class="txt_clr">KCES's IMR, Jalgaon</div>
        <div class="sub_txt">

                    Computer Department
        </div>
    </div>
```

</body>
</html>

Using Multiple Color Stops

The following example shows a linear gradient (from top to bottom) with multiple color stops:

Example:

```
#grad {
  background-image: linear-gradient(red, yellow, green);
}
```

Using Transparency

CSS gradients also support transparency, which can be used to create fading effects.

To add transparency, we use the rgba() function to define the color stops. The last parameter in the rgba() function can be a value from 0 to 1, and it defines the transparency of the color: 0 indicates full transparency, 1 indicates full color (no transparency).

The following example shows a linear gradient that starts from the left. It starts fully transparent, transitioning to full color red:

Example:

```
#grad {
  background-image: linear-gradient(to right, rgba(255,0,0,0),
rgba(255,0,0,1));
}
```

CSS Radial Gradients: A radial gradient differs from a linear gradient. It starts at a single point and emanates outward. By default, the gradient will be elliptical shape, the size will be farthest-corner the first color starts at the center position of the element and then fades to the end color towards the edge of the element. Fade happens at an equal rate until specified.

Syntax:

```
background-image: radial-gradient(shape size at position, start-color, ..., last-color);
```

Radial Gradient - Evenly Spaced Color Stops (this is default)

Example:

The following example shows a radial gradient with evenly spaced color stops:

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS Gradients</title>
```

```
        <style>
        #main {
                height: 350px;
                width: 700px;
                background-color: white;
                background-image: radial-gradient(#DCDCDC,
                                                        #fff, #2F4F4F);
        }

        .gfg {
                text-align: center;
                font-size: 40px;
                font-weight: bold;
                padding-top: 80px;
        }

        .geeks {
                font-size: 20px;
                text-align: center;
        }
        </style>
</head>

<body>
        <div id="main">
                <div class="gfg">KCES's IMR, Jalgaon</div>
```

```
        <div class="geeks">
        Computer Department
        </div>
    </div>
</body>
</html>
```

Set Shape

The shape parameter defines the shape. It can take the value circle or ellipse. The default value is ellipse.

The following example shows a radial gradient with the shape of a circle:

Example:

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS Gradients</title>
    <style>
    #main {
        height: 350px;
        width: 700px;
        background-color: white;

        background-image: radial-gradient(circle, #778899, #2F4F4F, #fff);
    }
```

```
        . txt_main {
                text-align: center;
                font-size: 40px;
                font-weight: bold;
                padding-top: 80px;
                color: white;
        }

        . txt_Sub {
                font-size: 20px;
                text-align: center;
                color: white;
        }
        </style>
</head>

<body>
        <div id="main">
                <div class="txt_main ">KCES's IMR, Jalgaon</div>
                <div class="txt_Sub">
                Computer Department
                </div>
        </div>
</body>
</html>
```

OUTPUT:

Use of Different Size Keywords

The size parameter defines the size of the gradient. It can take four values:

- **closest-side**
- **farthest-side**
- **closest-corner**
- **farthest-corner**

Example:

A radial gradient with different size keywords:

```
#grad1 {
  background-image: radial-gradient(closest-side at 60% 55%, red, yellow,
black);
}

#grad2 {
  background-image: radial-gradient(farthest-side at 60% 55%, red, yellow,
black);
}
```

CSS Conic Gradients

A conic gradient is a gradient with color transitions rotated around a center point.

To create a conic gradient you must define at least two colors.

Syntax

```
background-image: conic-gradient([from angle]
[at position,] color [degree], color [degree], ...);
```

By default, *angle* is 0deg and *position* is center.

If no *degree* is specified, the colors will be spread equally around the center point.

Conic Gradient: Three Colors

The following example shows a conic gradient with three colors:

Example:

A conic gradient with three colors:

```
#grad {
  background-image: conic-gradient(red, yellow, green);
}
```

CSS Opacity / Transparency

The **opacity** in CSS is the property of an element that describes the transparency of the element. It is the opposite of transparency & represents the degree to which the content will be hidden behind an element.

We can apply the opacity with different styling properties to the elements. A few of them are discussed below:

Image Opacity: The opacity property is used in the image to describe the transparency of the image. The value of opacity lies between 0.0 to 1.0 where a low value represents high transparency and a high value represents low transparency. The percentage of opacity is calculated as Opacity% = Opacity * 100.

Example: This example describes the opacity property by applying it to the image.

```
<!DOCTYPE html>
<html>
<head>
   <title>Opacity property</title>
   <style>
   .forest {
      opacity: 0.5;
   }
   p {
      font-size: 25px;
      font-weight: bold;
      margin-bottom: 5px;
   }
   .opacity {
      text-align: center;
```

```
  }
      .img_size{
            height:300px;
            width:500px;
      }
  </style>
</head>
<body>
  <div class="opacity">
<p>Image with 100% opacity (original image)</p>
      <img class="img_size" src=
"https://cdn.pixabay.com/photo/2016/11/23/13/48/beach-
1852945_960_720.jpg"
          class="forest1">
    <br>
    <br>
<p>Image with 50% opacity</p>
      <img class="img_size" src=
"https://cdn.pixabay.com/photo/2016/11/23/13/48/beach-
1852945_960_720.jpg"
          class="forest">
  </div>
</body>
</html>
```

Image with 100% opacity (original image)

Image with 50% opacity

Transparency box and transparency using RGBA values:

In the transparency box, child property inherits the property from the parent property but in the case of transparency using RGBA, only opacity property is used or applied to add transparency to the background of an element.

CSS Transitions

The CSS transitions are effects that are added to change the element gradually from one style to another, without using flash or JavaScript.

You should specify two things to create CSS transition.

- The CSS property on which you want to add an effect.
- The time duration of the effect.

Note: If the duration part is not specified, the transition will have no effect, because the default value is 0.

The transition effect will start when the specified CSS property (width) changes value.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: background-color 5s;
  border-radius: 5px;
}

div:hover {
```

```
  background-color: green;
  cursor: pointer;
}
</style>
</head>
<body>

<div></div>

</body>
</html>
```

OUTPUT: Will be Display run-time (not as image)

The transition is the combination of four properties which are listed below:

- transition-property
- transition-duration
- transition-timing-function
- transition-delay

Note: The transition effect can be defined in two states (hover and active) using pseudo-classes like : hover or: active or classes dynamically set by using JavaScript.

Syntax:

```
transition: transition-property transition-duration
transition-timing-function transition-delay;
```

Note: If any of the values are not defined then the browser assumes the default values.

Property Values:

- **transition-property:** It specifies the CSS properties to which a transition effect should be applied.
- **transition-duration:** It specifies the length of time a transition animation should take to complete.
- **transition-timing-function:** It specifies the speed of transition.
- **transition-delay:** It specifies the transition delay or time when transition starts.

Example:

```
div {
    width: 100px;
    height: 270px;
    background: green;
    transition: width 5s ease-in .2s;
    display: inline-block;
  }
  div:hover {
    width: 300px;
  }
```

OR

```
div {
    width: 100px;
    height: 270px;
    background: green;
    display: inline-block;
  }
div:hover {
 transform: translateY(-10px);
}
```

Supported Browsers: The browser supported by *transition* property are listed below:

- Google Chrome 26 and above
- Edge 12 and above
- Internet Explorer 10 and above
- Firefox 16 and above
- Opera 12.1 and above
- Safari 9 and above

Difference between transitions and animations:

Transition	Animations
Transitions cannot loop (You can make them do that but they are not designed for that).	Animations have no problem in looping.
Transitions need a trigger to run like mouse hover.	The animation just starts. They don't need any kind of external trigger source.
Transitions are easy to work in JavaScript.	The animations are hard to work in JavaScript. The syntax for manipulating a key frame and assigning a new value to it, is very complex.
Transitions animate a object from one point to another.	Animation allows you to define Key frames which varies from one state to another with various properties and time frame.
Use transition for manipulating the value using JavaScript.	Flexibility is provided by having multiple key frames and easy loop.

CSS transform Property

The **transform property** in CSS is used to change the coordinate space of the visual formatting model. This is used to add effects like skew, rotate, translate, etc. on elements.

Syntax:

```
transform: none|transform-functions|initial|inherit;
```

Note: The transformations can be of 2-D or 3-D type.

Values:

- **none**: No transformation takes place.
- **matrix(x, x, x, x, x, x):** It specifies a matrix transformation of 2-D type. It takes 6 values.
- **matrix3d(x, x, x, x, x, x, x, x, x):** It specifies a matrix transformation of 3-D type. It takes 9 values.
- **translate(x, y):** It specifies a translation across the X and Y axes.
- **translate3d(x, y, z):** It specifies a translation across the X, Y, and Z axes.
- **translateX(x):** It specifies the translation across the X-axis only.
- **translateY(y):** It specifies the translation across the Y-axis only.
- **translateZ(z):** It specifies the translation across the Z-axis only.
- **rotate(angle):** It specifies the angle of rotation.
- **rotateX(angle):** It specifies the rotation along with the X-axis corresponding to the angle of rotation.
- **roatateY(angle):** It specifies the rotation along with the Y-axis corresponding to the angle of rotation.
- **roteteZ(angle):** It specifies the rotation along with the Z-axis corresponding to the angle of rotation.
- **scale(x, y):** It specifies the scale transformation along the X and Y axes.
- **scale3d(x, y, z):** It specifies the scale transformation along the X, Y, and Z axes.
- **scaleX(x):** It specifies the scale transformation along the X-axis.
- **scaleY(y):** It specifies the scale transformation along the Y-axis.
- **scaleZ(z):** It specifies the scale transformation along the Z-axis.
- **scale3d(x, y, z):** It specifies the scale transformation along the X, Y, and Z axes.
- **skew(angle, angle):** It specifies the skew transformation along the X and Y axes corresponding to the skew angles.
- **skewX(angle)**: It specifies the skew transformation along with the X-axis corresponding to the skew angle.
- **skewY(angle)**: It specifies the skew transformation along with the Y-axis corresponding to the skew angle.
- **skewZ(angle)**: It specifies the skew transformation along with the Z-axis corresponding to the skew angle.
- **perspective(x)**: It specifies the perspective of an element.

- **initial**: It initializes the element to its default value.
- **inherit**: It inherits the value from its parent element.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
div.a {
  width: 150px;
  height: 80px;
  background-color: skyblue;
  transform: rotate(20deg);
}

</style>
</head>
<body>

<h1>The transform Property</h1>

<b><h2>transform: rotate(20deg):</h2>
<div class="a">Hello World!</div></b>
<br>

</body>
</html>
```