CSS(Cascading Style Sheets)

- **CSS:** Cascading Style Sheets (CSS) is a stylesheet language used to describe the look & presentation of a document written in HTML or XML. It defines how elements are displayed on a web page.
- CSS makes websites visually appealing and user-friendly.
- **Cascading Style Sheets use in HTML.**
- There are three ways to style a web page.
  1. Inline styling/ style attribute.
  2. Internal style sheet/style tag.
  3. External style sheet (CSS).
- **Inline styling.**
- An inline style may be used to apply a unique style for a single element.
- **We have universal style.**
- Meaning of universal style attribute.
- It can be used with all the HTML tags.
- **Syntax:** <tagname style=”property:value;property:value;”>content</tagname>
- **Code HTML inline styling.**

```
<!DOCTYPE html>
<html>
<head>
   <title>Document</title>
</head>
<body>
   <h1 align="center">The Lion and The Mouse</h1>
   <p style="color:red; background-color: blue">The Lion and The
Mouse.</p>
</body>
</html>
```

- **Internal style sheet.**
- An internal style sheet may be used if one single HTML page has a unique style.
- Style tag <style>.
- Style tag add to the head tag because head tag is a parents of style tag.
- **Code HTML internal style sheet.**

```
<!DOCTYPE html>
<html>
<head>
   <title>Document</title>
   <style>
   h1 {
      color:green;
      background-color: pink;
      font-size: 24px;                  }
   p{
      color:red;
      background-color:brown;
      font-size: 14px;                }
   </style>
</head>
<body>
   <h1>The Lion and The Mouse</h1>
   <p>This is a paragraph. </p>
   <p>This is another paragraph. </p>
</body>
</html>
```

- **External style sheet (CSS).**
- An external style sheet is a separate CSS file that can be accessed by creating a link within the head section of the webpage.
- First file name index.html.
- Second file name index.css.
- **Code HTML external style sheet (CSS).**

```
<!DOCTYPE html>
<html>
<head>
    <title>Document</title>
    <link rel="stylesheet"href="style.css">
</head>
<body>
    <h1>this is a heading.</h1>
    <p>this is a paragraph.</p>
</body>
</html>
```

- **Code Style.css.**

```
h1{
    color: red;
    background-color:yellow;
    font-size: 24px;                }
p{
    color: red;
    background-color:yellow;
    font-size: 24px;                }
body {
    background-color:aqua;          }
```

Selectors.

- **Selectors.**
- A CSS selector is a pattern that selects and styles HTML document elements. It's the first part of a CSS rule and tells the browser which HTML elements to select.
- **Syntax.**

```
Selector        {
          Property-1:value-1;
          Property-1:value-1;          }
```

- **Element/ type selectors.**
- They are used to style all the elements of a particular type at one.
- **Element selectors code HTML.**

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>selector</title>
   <link rel="stylesheet"href="style.css">
</head>
<body>
   <h1>This is a selector element 1.</h1>
   <p>They are used to style all the elements of a particular type at
one..</p>
</body>
</html>
```

- **Element selector code style.css.**

```
body {
   background-color: blueviolet;          }
h1 {
   text-align: center;
   background-color: yellow;
   font-size: 30px;          }
p {
   background-color: red;
   font-size: 24px;          }
```

- **Force refresh:** ctrl + shift + R

- **Id selectors.**
- They are used to style only a single element of a particular type.
- **Naming conventions for unique ID.**

1. It can have alphanumeric character or hyphen or underscore.
2. It cannot have any special characters.
3. The name cannot start with a number.

- **Id selector code HTML.**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>selector</title>
    <link rel="stylesheet"href="style.css">
</head>
<body>
    <h1>This is a selector.</h1>
    <p>selector element: They are used to style all the elements of a
particular type at one..</p>
    <p id="abc">id selector: They are used to style only a single element
of a particular type.</p>
</body>
</html>
```

- **Id selector code style.css.**

```
body {
    background-color: blueviolet;           }
h1 {
    text-align: center;
    background-color: yellow;
    font-size: 30px;                   }
p {
    background-color: red;
    font-size: 24px;           }
#abc {
    background-color:brown;
    font-size: 24px;                   }
```

- **Class selectors.**
- They are used when we want to style more than one element.
- **Class selector code HTML.**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>selector</title>
    <link rel="stylesheet"href="style.css">
</head>
<body>
    <h1 class="xyz">This is a selector.</h1>
    <p>selector element: They are used to style all the elements of a
particular type at one..</p>
    <p class="xyz">id selector: They are used to style only a single
element of a particular type.</p>
    <p class="xyz">class selector: They are used when we want to style
more than one element.</p>
</body>
</html>
```

- **Class selector style.css.**

```
body {
    background-color: burlywood;        }
p{
    font-size: 24px;           }
.xyz {
    background-color:red;
    font-size: 24px;           }
```

- **Combination selectors.**
- A combination is something that explains the relationship between the selectors.
- **Code style.css combination selectors.**

```
body
{
    background-color: burlywood;
}
p{
    font-size: 24px;             }
p.xyz {
    background-color:red;
    font-size: 24px;
}
```

- **Child selectors.**
- We use the concept of parent-child hierarchy to select any HTML element.
- **Code HTML child selector.**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>selector</title>
    <link rel="stylesheet"href="style.css">
</head>
<body>
    <div>
        <ul>
            <li>
                <a href="http://www.google.com"target="_blank">Google</a>
            </li>
            <li>
                <a
href="http://www.facebook.com"target="_blank">facebook</a>
            </li>
        </ul>
    </div>
    <div>
        <p>
            <a
href="http://www.intershala.com"target="_blank">intershala</a>
        </p>
    </div>
</body>                              /</html
```

- **Code style.css.**

```
body {
    background-color: burlywood;        }
p > a {
    color:green;        }
```

- What would be the CSS code, if we only want to style the first two <a> elements using child selectors?

- **Code style.css.**

```
body {
   background-color: burlywood;        }
p > a {
   color:green;                  }
li > a {
   color:blue;         }
```

- **Descendant selectors.**
- We use the concept of descendants to select the HTML elements.
- **HTML code same 5 selectors. Code style.css.**

```
body {
   background-color: burlywood;        }
p > a {
   color:green;       }
div  a {
   color:blue;         }
```

- **Grouping selectors.**
- They help us to apply the same style to multiple selectors of elements.
- **Code HTML grouping selector.**

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>selector</title>
   <link rel="stylesheet"href="style.css">
</head>
<body>
   <h1>This is a selector.</h1>
   <h2>selector element: They are used to style all the elements of a
particular type at one.</h2>
   <p>id selector: They are used to style only a single element of a
particular type.</p>
   <p>class selector: They are used when we want to style more than one
element.</p>
</body>
</html>
```

- **Code style.css.**

```
body {
   background-color: burlywood;        }
p, h2 {
   color: blue;        }
```

- **Attribute selectors.**
- Attribute selectors are used select and style HTML elements with the specified attributes and values.
- **Code HTML attribute selector.**

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>selector</title>
   <link rel="stylesheet"href="style.css">
</head>
<body>
  <form>
   First name: <br/>
   <input type="text"name="firstname"value="Mickey"><br/>
   Last name: <br/>
   <input type="text"name="lastname"value="Mouse"><br/>
   Password: <br/>
   <input type="password"name="password"value="Disney"><br/>
   <input type="submit"value="submit">
  </form>
</body>
</html>
```

- **Code style.css.**

```
body {
   background-color: burlywood;
}
input[type="text"]          {
   color: red;
}
```

- **Pseudo-Class Selector.**
- <a href="https://www.youtube.com/watch?v=oMXdbUN6h-Q&t=9s"target="_blank">home</a>
- **CSS code Style.css.**

```
a:hover {
    background-color: red;
    color: black;       }
```

- **CSS Comments.**
- /* This is a comment */

RGB Color.

- RGB Color in CSS.

RGB value.	Color.
rgb (0,0,0)	Black
rgb(255,0,0)	Red
rgb(0,255,0)	Green
rgb (0,0,255)	Blue
rgb (255,255,0)	Yellow
rgb (255,0,255)	Purple
rgb (0,255,255)	Cyan
rgb (255,255,255)	White

- **Syntax.**

```
P       {
Color: rgb (255,0,0);       }
```

- 255*255*255 = 16777216 (more than 16 million colors)
- **Hexadecimal specify value.**

Hexadecimal value.	Color.
#000000	Black
#FF0000	Red
#00FF00	Green
#0000FF	Blue
#FFFF00	Yellow
#FF00FF	Purple
#00FFFF	Cyan
#FFFFFF	White

- **GRAPHIC DESGINER – CSS.**
  1. Hue: A pure color.
  2. Tint: A pure color with just white added.
  3. Shade: a pure color with just black added.
  4. Tone: A pure color with just grey added
- **CSS properties.**
  1. Color properties.
  2. Background-color properties.
  3. Font-family properties.
  4. Font-size properties.

- **Color properties.**
- The default value of color property is black.
- **Code HTML color properties.**

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>CSS properties</title>
   <link rel="stylesheet"href="style.css">
</head>
<body>
  <h1>Color properties</h1>
  <div>
    <p>
      The default value of color property is black.
    </p>
    <p class="abc">
      The default value of color property is black.
    </p>
  </div>
</body>
</html>
```

- **Code style.css.**

```
body {
   background-color: burlywood;        }
.abc{
   color: red;
}
```

Back-ground Properties in CSS.

- **Back-ground.**
- The back-ground shorthand CSS property sets all background style properties at once, such as color, image, origin and size or repeat method.
- The default value of this property is transparent (no color).
- **Back-ground properties in CSS.**

1. Back-ground-color.
   Body{ background-color: blueviolet;}
2. Back-ground-image.
   Body { background-image: url("C:/Users/Admin/Pictures/images.jpg");}
3. Back-ground-repeat.
   Body {background-repeat: no-repeat;}
4. Back-ground-position.
   Body {background-position: center;}
5. Back-ground-size.
   Body { background-size: 50%;}
6. Back-ground-attachment.
   Body {background-attachment: fixed;}

Unit in CSS.

- **Unit in CSS.**
- A Unit determines the size of a property you are setting for an element or its content.
- **Absolute Lengths.**

1. **Pixels (px): pixels are relative to the viewing device.**

(width: 600px;   height: 200px;).

2. **Inches (in).**

- **CSS code.**

```
#box{
  width: 100vw;
  height: 100vh;
  border: 2px solid black;
  font-size: 1in;
  font-size: 0.5 in;            }
```

3. **Points (pt).**

- **CSS code.**

```
#box{
  width: 600px;
  height: 200px;
  border: 2px solid black;
  font-size: 15pt;                  }
```

- **Relative Lengths.**

1. **em:** it is used to font-size.(font-size: 2em;).
2. **Root em (rem):** it is work on root.(#box font-size: 2rem;).

- **CSS code.**
- html{

```
  font-size: 5px;               }
```

3. **Percentage (%):** (width: 100%;)
4. **Viewport width:** Viewport width is a relative unit of measurement used in CSS to define sizes and layouts on web pages.
5. **Viewport height:** 100vh would represent 100% of the viewport's height, or the full height of the screen.

- **CSS code.**
```
#box{
  width: 100vw;
  height: 100vh;
  border: 2px solid black;                    }
```

Font in CSS.

- **Font.**
- Font is used to specify a font type to a text.
- **Font Properties in CSS.**
  1. Font-Family.
  2. Font-Size.
  3. Font-Weight.
  4. Font-Style.
  5. Font-Variant.
- **Font properties code.**
- h1 {
```
  font-family: Georgia, "Times New Roman", Times, serif;
  font-family: "Urbanist", sans-serif;
  font-family: "thapaFont", "Urbanist", sans-serif;
  font-size: 54px;
  font-weight: 900;
  font-style: italic;
  font-variant: small-caps;          }
```
- p {
```
  font-family: "Urbanist", sans-serif;
  font-size: 20px;
  font-weight: 300;          }
```

Texts in CSS.

- **Texts in CSS.**
- **TEXT PROPERTIES - CSS**
  1. Text Alignment.
  2. Text Decoration.
  3. Text Transform.
  4. Text Spacing.
  5. Text Shadow.
- **Text Properties code.**

-
  ```
  p,h1{
     text-align: center;
     text-decoration: underline;
     text-transform: uppercase;
     letter-spacing: 1px;
     word-spacing: 2px;
     text-shadow: 4px 4px red;     }
  ```

- **how to make first letter big in this paragraph?**

-
  ```
  p::first-letter{
     text-transform: uppercase;
     font-size: 50px;
     font-weight: 600px;
  }
  ```

BORDER IN CSS.

- **BORDER IN CSS.**
- **Properties of Borders.**

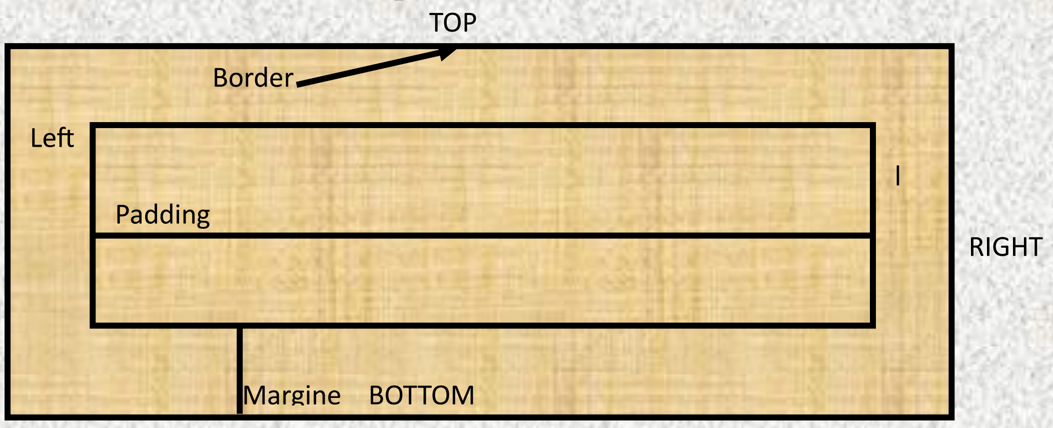
1. Border-style.
2. Border-width
3. Border-color.
4. Border-radius.
5. Solid _______________.
6. Dotted.…………………
7. Dashed_ _ _ _ _ _ _ _ _ _ _.

- h1{

```
text-align: center;
border-color: black;
border-style: solid;
border-radius: 20px;
border:7px double green;
border-bottom-left-radius:50px;
border-top-right-radius: 20px;        }
```

- div{

```
text-align: center;
border:5px solid black;
border-radius: 20px;
border-top: 5px solid;
border-bottom: 5px solid;      }
```



PADDING IN CSS.

- **Padding CSS.**
- Padding is the space around the content of an element inside its border.
- Default value is 0px.
- **Every element side different padding.**
- Top padding = 10px.
- Right padding = 40px.
- Bottom padding = 20px.
- Left padding = 5px
- **Sub properties of padding.**
  1. Padding-top.
  2. Padding-right.
  3. Padding-bottom.
  4. Padding-left
- **Padding CSS code.**

-
  ```
  h1{
     padding-top: 10px;
     padding-bottom: 10px;
  }
  ```

-
  ```
  div{
     padding-top: 10px;
     padding-right: 20px;
     padding-bottom: 10px;
     padding-left: 20px;
  }
  ```

Margin in CSS.

- **Margin in CSS.**
- It is the space around the element outside its border and is added to maintain space between adjacent elements.
- **Sub properties of margin.**
  1. Margin-top.
  2. Margin-right.
  3. Margin-bottom.
  4. Margin-left.
- **Margin code.**

-
  ```
  h1{
     margin-top: 200px;
     margin-right: 200px;
     margin-bottom: 200px;
     margin-left: 200px;
  }
  ```

-
  ```
  div{
     margin-top: 200px;
     margin-right:200px;
     margin-bottom: 200px;
     margin-left: 200px;
  }
  ```

Gradients in CSS.

- **Gradients in CSS.**
- Gradients in CSS are gradients of colors that are considered background images.
- There are two main types of CSS gradients:
- linear and radial.
- **Gradients in CSS.**
- background-image: linear-gradient(red, yellow);
- **Gradient with Direction – CSS**
- background-image: linear-gradient
  (to right, red, yellow);
- **Gradient with Angles – CSS**
- background-image: linear-gradient
  (90deg,red, yellow);
- **Radial Gradient – CSS**
- background-image: radial-gradient(red, yellow);
- **Gradient with Shape – CSS**
- background-image: linear-gradient
  (circle ,red, yellow);
- **Gradient code.**
-
```
body{
    width: 100%;
    height: 100vh;
    background-image: linear-gradient(black,white);
}
```

Box Shadow in CSS.

- **Box Shadow in CSS.**
- The CSS box-shadow is a property adds a shadow around a box and value an element on a webpage.
- **Box shadow.**
- box-shadow: -2px 1px 2px 4px #61677a;
- **Values.**
- Offset-X = -2px;
- Offset-Y = 1px;
- Blur-radius = 2px;
- Density/Spread = 4px;
- Color = #61677a;

Drop Shadow 16 in CSS.

- **Drop Shadow 16 in CSS.**
- **The drop shadow is effectively and drop shadow removes the image shadow.**
- **Drop Shadow Properties.**
- drop-shadow: -2px 1px 2px #61677a;
- **Values.**
- Offset-X = -2px;
- Offset-Y = 1px;
- Blur-radius = 2px;
- Color = #61677a;
-
```
.box{
    filter: drop-shadow(0 0 0 #61677a);
}
```
-
```
img{
    filter: drop-shadow(0 0 0 #61677a);
    -webkit-filter: drop-shadow(10 10 200px #61677a );
}
```

Filters in CSS.

- **Filters in CSS.**
- The filter applies graphical effects like blur or color shift to an element.
- **Properties of Filters in CSS.**
  1. Grayscale().
  2. Blur().
  3. Brightness().
  4. Opacity().
  5. Contrast().
  6. Invert().
- HTML code.

```
<div class="box">
        <h1>Shaan</h1>
        <img src="C:\Users\Admin\Desktop\New folder\download.webp"/>
      </div>
```

- **Filter code.**
- img{

```
        width: 50%;
        border-radius: 20px;
        -webkit-border-radius: 20px;
        -moz-border-radius: 20px;
        -ms-border-radius: 20px;
        -o-border-radius: 20px;
        box-shadow: rgba(0, 0, 0, 0.16) 0px 10px 36px 0px,
          rgba(0, 0, 0, 0.06) 0px 0px 0px 1px;
        /*-webkit-filter: opacity(1);
        -webkit-filter: grayscale(); = blank in white.
        */
        filter: blur(3px) grayscale(0.5) contrast(1.2);
        -webkit-filter: blur(3px) grayscale(0.5) contrast(1.2);
}
```

List in CSS.

- **List.**
- **Properties of list.**
  1. List-style-type.
  2. List-style-image.
  3. List-style-position.
  4. List-style-property.
- **HTML code.**
-
```
<section>
      <div class="lists">
        <ul>
          <li>LIKE</li>
          <li>
            SHARE
            <ul>
              <li>SHARE</li>
              <li>SUBSCRIBE</li>
            </ul>
          </li>
          <li>SUBSCRIBE</li>
        </ul>
      </div>
  </section>
```
- **List code.**
-
```
body {
    width: 100%;
    height: 100vh;
    background-image: linear-gradient(to bottom, #5b697a 0%, #5b697a
50%),
      linear-gradient(to bottom, #fafbfd 0%, #fafbfd 50%);
    background-size: 100% 50%, 100% 50%;
    background-repeat: no-repeat;
    display: grid;
    place-items: center;                          }
```

-
  ```
  .lists {
    width: 500px;
    padding: 40px;
    background-color: #fff;
    border-radius: 10px;
    box-shadow: rgba(0, 0, 0, 0.24) 0px 3px 8px;
    -webkit-border-radius: 10px;
    -moz-border-radius: 10px;
    -ms-border-radius: 10px;
    -o-border-radius: 10px;
  }
  ```

-
  ```
  ul {
    list-style-type: decimal;
    list-style-position: inside;
    /* list-style-image: url("../images/mario-run.gif"); */
  }
  ```

-
  ```
  li {
    font-size: 32px;
    font-weight: 400;
    text-transform: capitalize;
    margin-bottom: 16px;
    border: 2px solid #ed5656;
  }
  ```

Anchor in CSS.

- **Anchor in CSS.**
- **Anchor properties.**
  1. Link('a:link')
  2. Visited ('a:visited')
  3. Hover('a:hover')
  4. Active('a:active')
- **Anchor HTML code.**
-
```
<div class="box">
<h1>this is a link</h1>
<a href="https://www.youtube.com/watch?v=5ccq_nLHneE">Shaan</a>
    </div>
    <div>
      <h1>this is a link</h1>
      <a href="https://www.youtube.com/watch?v=5ccq_nLHneE"
target="_blank">Shaan</a>
    </div>
```
- **Anchor CSS code.**
-
```
a:visited{
  color: red;
}
```
-
```
a:hover{
  color:black;
}
```
-
```
a:active{
  color:yellow;
}
```

Combinators in CSS.

- **Combinators in CSS.**
- CSS combinators are used to define relationships between different elements in your HTML document. They allow you to target elements based on their position and hierarchy within the document structure.
- **Combinators properties in CSS.**
  1. Descendant Selector (Space): Selects an element that is a descendant of another element.
  2. Child Selector (>): Selects an element that is a direct child of another element.
  3. Adjacent Sibling Selector (+): Selects an element that is immediately preceded by another element.
  4. General Sibling Selector (~): Selects all elements that are siblings of another element and share the same parent.
  5. Universal Selector (*): Selects all elements.
- **HTML code.**

```
<section>
     <div class="content">
       <ul>
          <p>Welcome to our World Best CSS Full Course Tutorial for
         Beginners in Hindi.</p>
         <li>inside</li>
         <li>
           inside
           <ol>
             <li>nested</li>
           </ol>
           <ul>
             <li>deep nested</li>
           </ul>
             <li>SUBSCRIBE</li>
           </ol>
         </li>
         <li>inside</li>
         <p>The Shaan websites</p>
         <li>last child</li>   </ul>          </div>        </section>
```

- **CSS code.**
- **Descendant Selector.**

```
.content ul p{
  font-weight: bold;
  color: red;
}
.content ul li{
  color: blue;
}
```

- **child selector.**

```
.content > ul > li{
  color:brown;
}
```

- **Adjacent Sibling Selector.**

```
.content ul p + li{
  color: pink;
}
```

- **General Sibling Selector.**

```
.content ul p ~ li{
  color: purple;
}
```

- **Universal selector.**

```
*{
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
  font-family:'Courier New', Courier, monospace;
}
```

Display in CSS.

- **Display in CSS.**
- it is used to specify how an element will align horizontally with respect to other adjacent elements.
- **Display properties value.**
  1. inline.
  2. block.
  3. Inline-block.
  4. None.
- Default value is different for different elements.
- For paragraph elements, the default display property value is block.
- For anchor elements, the default display property value is inline.
- Display: inline.
- An element starts beside the previous element, provided the previous element allows it to sit next to it and there is enough space.
- Width and height CSS properties have no effect.
- **HTML code.**

```
<body>
   <h1>Shanlhai.</h1>
   <ul>
      <li>List Number 1.</li>
      <li>List Number 2.</li>
      <li>List Number 3.</li>
      <li>List Number 4.</li>
      <li>List Number 5.</li>
      <li>List Number 6.</li>
   </ul>
        <span>one</span>
        <span>two</span>
        <span>three</span>
        <span>four</span>
        <a href="">first</a>
        <a href="">second</a>
  </body>
```

- **Display Properties works.**
- **Display None:** With a none value for this property, the display for the element is turned off.
- **Display inline:** It can also be used to place a div inline, or in a horizontal manner. It is used in single line.
- **Display block:** An element that has the display property set to block starts on a new line and takes up the available screen width.
- **Display inline-block:** inline and block work together.
- **CSS Code.**

```
ul{
   border: 4px solid black;
}
li{
   border: 1px solid black;
   display: inline;
}
span{
   border: 2px solid black;
   font-size: 20px;
   /* display: block; */
   display: inline-block;
   width: 150px;
   padding: 10px;
}
a{
   border: 2px solid black;
   font-size: 20px;
   display: inline-block;
}
```

positions in CSS.

- **Position in CSS.**
- It helps us to move HTML elements from their existing positions.
- It has 5 values.
  1. Relative.
  2. Absolute.
  3. Static.
  4. Fixed.
  5. Sticky.   {position: sticky; top: 50%}
- **Position Relative.**
- It helps move an HTML element with respect to its original position.
- **Position static.**
- Default value is different for different elements.
- Elements cannot be moved from their position.
- **Position fixed properties.**
- It moves the element with respect to the browser window.
- The position of the element remains fixed even if we scroll the web page.
- The next element present after the positioned element takes the place of the positioned element leaving no gap.
- {Position : fixed; top = 0px;}
- **Position absolute properties.**
- It moves the element with respect to the nearest positioned ancestor element.
- **Ancestor meaning:** ancestor means any element that is present up in the HTML hierarchy.
- **Positioned value meaning:** it means any element which has the position value other than the static.
- **Nearest position element meaning:** it means the nearest ancestor that has a non-static position value.
- **Position properties.**

1. Top.
2. Right.
3. Bottom.
4. Left.

- **HTML code.**

-
  ```
  <h1 id="main-heading">CSS Positions</h1>
      <section class="parent-div">
        <div class="child child-1">child 1</div>
        <div class="child child-2">child 2</div>
        <div class="child child-3">child 3</div>
        <div class="child child-4">child 4</div>
        <div class="child child-5">child 5</div>
      </section>
  ```
- **Position work.**
- **Relative:** relative to its normal position. It is use to top left right properties.
- **Absolute:** absolute work on according screen.
- **Fixed:** An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled.
- **Sticky**: Scroll down this page to see how sticky positioning works.
- **Static**: Static is the default position for HTML elements.
- **CSS code.**

-
  ```
  #main-heading {
    color: #1e1403;
    text-align: left;
    margin-bottom: 30px;    }
  ```

-
  ```
  .parent-div{
    width: 1000px;
    height: 400px;
    padding: 10px;
    border-color: black;
    border-style: solid;
    background: linear-gradient(to right, #514a9d, #24c6dc);
    box-shadow: rgba(0, 0, 0, 0.1) 0px 4px 12px;
    border-radius: 5px;                   }
  ```

-
  ```
  .child {
    width: 150px;
    height: 150px;
    background: linear-gradient(to right, #93edc7, #1cd8d2);
    box-shadow: rgba(100, 100, 111, 0.2) 0px 7px 29px 0px;
    /* to make the text center   */
    display: grid;
    place-items: center;
  ```

  font-size: 32px;
  color: #003180;   }

- .child:nth-child(2),
  .child:nth-child(4)   {
    background: linear-gradient(to right, #3cd3ad, #4cb8c4);
    box-shadow: rgba(100, 100, 111, 0.2) 0px 7px 29px 0px;   }
- .child {
    position: absolute;
  }
  /* top, left, bottom, right */
- .child-2 {
    top: 20%;
    right: 14%;
  }
- .child-3 {
    left: 1.5%;
    bottom: 3%;
  }
- .child-4 {
    right: 170px;
    bottom: 20px;
  }
- .child-5 {
    left: 445px;
    top: 250px;
     /* left: 50%;
    top: 50%;
    transform: translate(-50%, -50%); */
  }
- **Z-index.**
- It is used to position an element along the z axis.
- It is used mainly when there are overlapping elements and we want to explicitly put one element over the other.
- Higher the z-index value of an element, the more-close it is to us.
- It works only on positioned elements.
- **CSS code.**
- .child-4{
    z-index: 1;
  }

Overflow in CSS.

- **Overflow in CSS.**
- The overflow property in CSS is used to control how content overflows its containing element when it doesn't fit within the available space. It's a useful property for handling content that exceeds its container's dimensions.
- **The properties of overflow.**
  1. Overflow: Visible (Default).
  2. Overflow: Hidden.
  3. Overflow: Scroll.
  4. Overflow: Auto.
- **HTML code.**
-
```
<h1 id="main-heading">CSS Overflow</h1>
    <hr />
    <br />
    <section>
      <div class="overflow-div">
        <p>
          Discover a world of possibilities with Thapa Technical
comprehensive CSS course. Dive into over 50 real-world interview
questions to boost your job prospects. Unlock the secrets of CSS with
30+ invaluable tips and tricks. Engage in dynamic learning with our
animated PowerPoint presentations. Showcase your creativity with 10+
captivating animation projects. Take on the ultimate challenge with a
mega, fully multipage responsive website project. Elevate your web
presence with SEO insights and website performance testing. Plus, get
expert guidance on choosing between free and paid hosting options. Our
course is more than just CSS; it's your gateway to mastering web
development.
        </p>
      </div>
```
- **CSS code.**

```
.overflow-div{
  width: 500px;
  height: 200px;
  box-shadow: burlywood;
  border: 4px solid darkgrey;
  padding: 10px;
  overflow: hidden;
  /* overflow-y: scroll;
  overflow: scroll; */        }
```

PSEUDO ELEMENTS IN CSS.

- **Pseudo Elements.**
- A CSS pseudo-element is used to style specified parts of an element.
- **Properties of pseudo elements.**
  1. ::before
  2. ::after.
  3. ::first-letter.
  4. ::first-line.
  5. ::selection.
  6. ::placeholder.
- **::before Pseudo-Element:** The ::before pseudo-element is used to insert content before the content of an element. It is often employed to add decorative elements, icons, or text before an element without modifying the actual HTML content.
- **CSS code.**
- p::before{
  content: "Hello!";
  color: black; }
- **::after Pseudo-Element:** The ::after pseudo-element functions similarly to ::before, but it inserts content after the content of an element. It's often used for adding decorative elements, icons, or textual information after an element.
- **CSS code.**
- p::after{
  content:"hi";
  color:black; }
- **::first-letter-pseudo-Element:** The first-letter pseudo-element represents the first letter of an elements.
- **CSS code.**
- p::first-letter{
  color:black;
  font-size: 30px;
  font-weight: bold;
  }

- **::first-line** pseudo-Element: The first-line pseudo-element is for applying styles to the first line of an element.
- **CSS code.**

-
  ```
  p::first-line{
    color: black;
    font-size: 30px;
    font-weight: bold;
  }
  ```
- **::selection pseudo-Element:** it is used to apply styles to the part of a document that has been highlighted by the uses.
- **CSS code.**

-
  ```
  p::selection{
  background-color: blue;
  color: #fff;
  }
  ```
- **::placeholder pseudo-Element:** The placeholder pseudo-element is used to design the placeholder text by changing the text color and it allows to modify the style of the text.
- **HTML code.**

-
  ```
  <form>
       Name
       <input type="text"name="name"placeholder="enter the name"/>
  ```
- **CSS code.**

-
  ```
  ::placeholder{
    color:red;
    background-color: gray;                    }
  ```

Pseudo class in CSS.

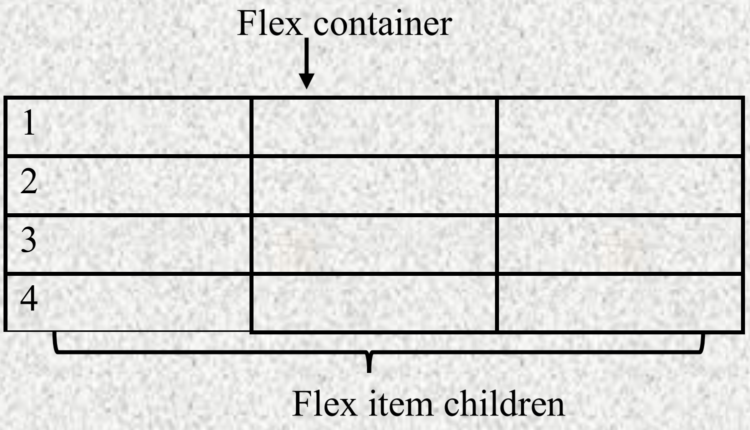
- **Pseudo class in CSS.**
- A CSS pseudo-element is a keyword that can be added to a selector to style a specific part of the selected elements.
- **Pseudo class properties.**
  1. :hover.
  2. :first child.
  3. :last child.
  4. :nth-child.
  5. :first-of-type.
  6. :last-of-type.
- **CSS code.**
- ul a:link{
  color:red; }
- ul a:hover{
  color:red; }
- ul a:visited {
  color:green; }
- p:first-child{
  color: red; }
- p:last-child{
  color: red; }
- p:nth-child(3){
  color:red; }
- li:nth-child(2n){
  color:red; }
- p:nth-last-child(2){
  color:red; }
- p:first-of-type{
  color:red; }
- p:last-of-type{
  color:green; }

Column Layout in CSS.

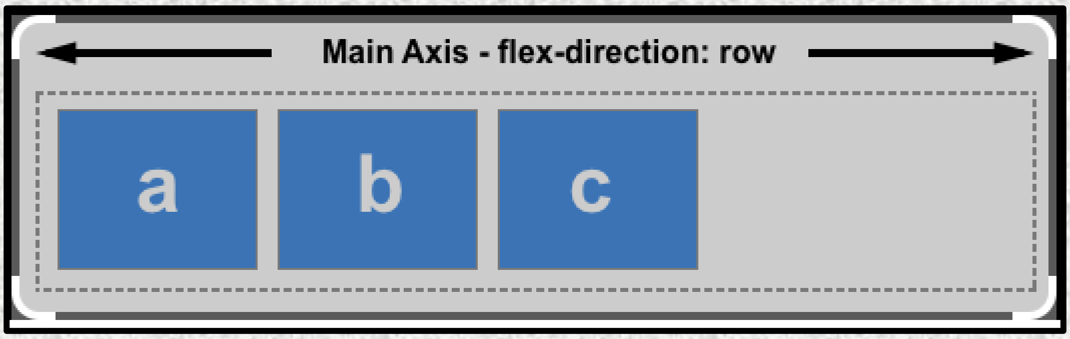
- **Column Layout in CSS.**
- The CSS multi-column layout allows easy definition of multiple columns of text - just like in newspapers.
- column-count
- column-gap
- column-rule-style
- column-rule-width
- column-rule-color
- column-rule
- column-span
- column-width
- **column layout.**
  1. Column-Count.
  2. Column-gap.
  3. Column-rule.
  4. Column-fill.
- **HTML code.**
-
```
<section class="hero-section">
      <br />
      <h1>Best CSS Course Ever Created</h1>
      <br />
      <p>
Discover a world of possibilities with our comprehensive Thapa Technical CSS course.Elevate your web presence with SEO insights
and website performance testing. Plus, get expert guidance on choosing between free and paid hosting options. Our course is more than just CSS; it's your gateway to mastering web development.
      </p>
  </section>
```
- **CSS code.**
-
```
P{
  column-count: 3;
  column-rule: 4px solid red;
  column-rule-color: red;
  column-rule-style: dotted;
  column-rule-width: 5px;
  column-gap: 50px;
  text-align: justify;                    }
```

Flexbox in CSS.

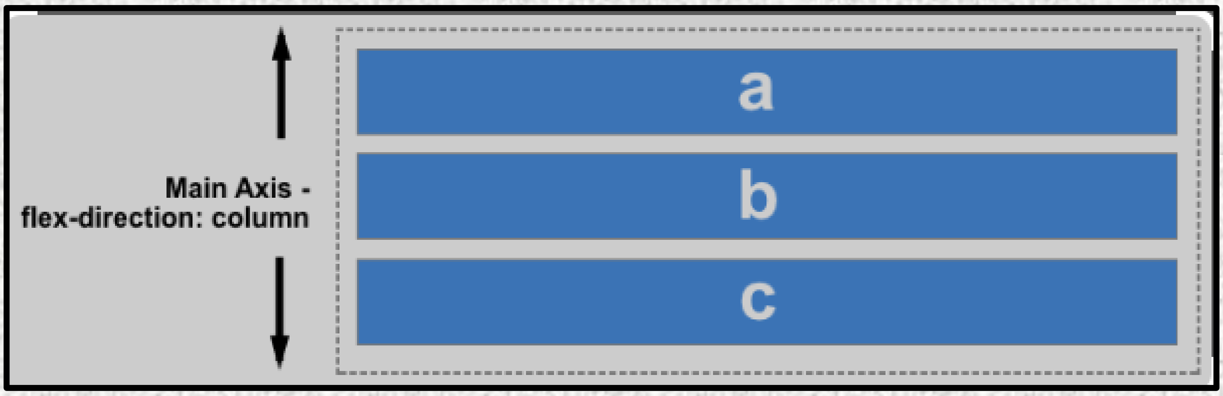
- **Flexbox.**
- Flex box is a one-dimensional layout method for arranging items in rows or columns. It is used to horizontally or vertically.
-
-



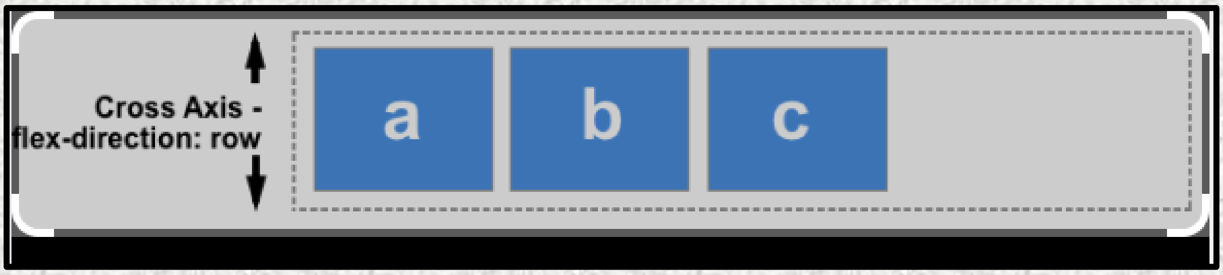
- **Display properties values.**
  1.Block property.
  2.Inline property.
  3.Inline-block property.
  4.None property.
  5.Flex property.
- **They have a two axis.**
1. Main-axis.
2. Cross-axis.
- **Main-axis.**
- In which direction do we flex item align main-axis.



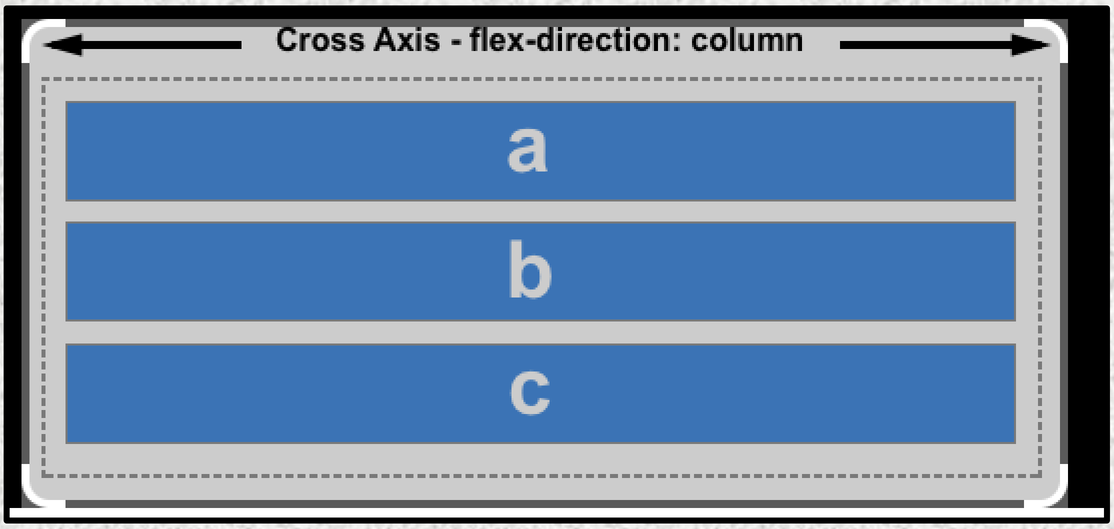
- Flex-direction: row left to right. it is a horizontal parpendicular.
- Flex-box terminal point.
  1. Start lines.
  2. End lines.
  3. Start line and end line ko hum reverse be kr sekta hai.



- Flex-direction: column from top to bottom. it is a vertically.
- **Cross-axis.**



- Flex-direction: row left to right. It is a vertically parpendicular.



- Flex-direction: row top to bottom. It is a horizontal parpendicular.
- **Question: How can we define the direction of flex items inside the flex container?**
- Answer: flex direction property.
- **Question: to which of the following is the flex-direction property applied to?**
- Answer: parent element.

- **Flex box horizontally CSS code.**
- **Horizontally properties.**
  1. **Start.**
  2. **Center.**
  3. **Flex-end.**
  4. **Space-between.**
  5. **Spece-round.**
- **CSS code.**
-
```
.flex-container{
  width: 500px;
  height: 200px;
  border: 2px solid;
  display: flex;
  flex-direction:  row;
  justify-content: start;
}
```
- **Flex box vertically CSS code.**
- **Vertically properties.**
  1. **Start.**
  2. **Center.**
  3. **End.**
- **CSS code.**
-
```
.flex-container{
  width: 500px;
  height: 200px;
  border: 2px solid;
  display: flex;
  flex-direction:  row;
  justify-content: center;  /center
  align-items:center ; /end
}
```
- **Flex Item Properties (children).**
- **Order :-** Determines the order in which a flex item appears relative to other flex items within the container. Lower values come first.
- Order: -2;

- **flex-grow :-** Specifies how much a flex item should grow to fill available space along the main axis. Default value is 0, meaning it won't grow.
- flex-wrap: 1;
- **flex-shrink :-** Specifies how much a flex item should shrink when there isn't enough space along the main axis.- Default value is 1, meaning it will shrink.
- flex-shrink: 5;
- **flex-basis :-** Defines the initial size of a flex item along the main axis. Default value is auto, which means the item's size is determined by its content.
- flex-basis: 100px;
- **flex (Shorthand for flex-grow, flex-shrink, and flex-basis):-** Combines the three flex item properties in one declaration.
- **Flex box properties.**
- flex-direction.
- flex-wrap.
- flex-flow.
- justify-content.
- align-items.
- align-content.
- **HTML code.**
-
```
<section class="flex-container">
    <div class="item-1">item 1</div>
    <div class="item-2">item 2</div>
    <div class="item-3">item 3</div>
  </section>
```

- **Flex box CSS code.**
- .flex-container {

```
    width: 60%;
    height: auto;
    padding: 50px 0;
    margin: 0 auto;
    box-shadow: rgba(0, 0, 0, 0.16) 0px 1px 4px, rgb(51, 51, 51) 0px 0px
0px 3px;
    background: linear-gradient(to right, #514a9d, #24c6dc);
    color: #fff;
    /* flexbox container properties */
    display: flex;
    /* flex-direction: row-reverse; */
    /* flex-direction: column-reverse; */
    flex-direction: row;
    /* justify-content: center; */
    /* justify-content: space-between; */
    justify-content: center;
    align-items: start;
    gap: 20px;
    flex-wrap: wrap;
    align-content: center;                                    }
```

GRID in CSS.

- **GRID in CSS.**
- Grid, is a two-dimensional layout system in CSS. The Grid layout module offers a grid-based layout system, with row and columns, making it easier to design web pages without having to use floats and positioning.
- **The grid property is a shorthand property for:**
- **grid-template-rows:** it is used to create a row.
- **CSS code.**
  (grid-template-rows:repeat(4,100px) ;/ gap: 5px;
- **grid-template-columns:** it is used to create a column.
  (grid-template-columns: repeat(4,100px);
- grid-template-areas.
- **grid-auto-rows**: grid-auto-rows specifying the size of the grid rows that were created without having an explicit size.
- **CSS code.**
  (grid-auto-rows: 250px;)
- grid-auto-flow.
- **1fr.**
- 1fr is for 1 part of the available space.
- **CSS code.**
  grid-template-columns: 1fr 2fr 1fr;
- **Align-items:** (center/ self-start/self-end/ stretch.)
- **Justify horizontally content and justify items both are same properties.**
- Justify-content: (self-start/ self-end/center/ stretch);
- justify-items: (self-start/ self-end/center/ stretch);
- **Position change grid-items property.**
- grid-row-start.
- grid-column-start.
- grid-row-end.
- grid-column-end.
- **CSS code.**
- background-color: green;
  grid-row-start: 1;
  grid-column-end: 2;
  grid-column-start:2 ;
  grid-column-end: 3;

- **column spend kar na ka liya.**
- background-color: gray;
  grid-row-start: 1;
  grid-column-end: 2;
  grid-column-start:2 ;
  grid-column-end: 4;
- **row spen kar na ka liya.**
- background-color: blue;
  grid-column: 1/ -1;
- **explicit and implicit.**
- **Explicit:** the grid-template-rows and grid-template-columns. This manually defined grid is called the explicit grid.
- **Implicit:** the grid-template-rows and grid-template-columns does not defined manually is called the implicit grid.
- **CSS code stortform.**
- grid-area: 2/2/3/3;
- grid-template-columns: 1fr min-content repeat(3, 1fr);
- grid-template-columns: 1fr max-content repeat(3, 1fr);
- grid-template-columns: 1fr minmax(250px, 1fr) repeat(1, 1fr);
- **fixed size in min contains and max contains in box.**
- **Min contains:** grid-template-columns: min-content repeat(2,1fr);
- **Max contains:** grid-template-columns: max-content repeat(2,1fr);
- **Minmax-contains:** grid-template-columns: minmax(200px,250px) repeat(2,1fr);
- **Responsive web layout.**
- grid-template-columns: repeat(auto-fill, minmax(200px, 250px));
- grid-template-columns: repeat(auto-fit, minmax(200px, ifr));
- **HTML code.**

```
<div class="grid-container">
    <div class="item-1">item-1 Shanelhai World Best CSS Course for every one. it is a advance course for all students.</div>
    <div class="item-2">item-2</div>
    <div class="item-3">item-3</div>
    <div class="item-4">item-4</div>
    <div class="item-5">item-5</div>
    <div class="item-6">item-6</div>
  </div>
```

- **CSS code.**
- body{

```
body{
  background-color: bisque;
  margin: 0px;
  padding: 100px;
}
.grid-container{
  width: 90%;
  height: 50px auto;
  border: 5px solid;
  display: grid;
  grid-template-rows: repeat(2,200px);
  grid-template-columns: repeat(auto-fit, minmax(200px, ifr));
  gap:20px;
  grid-auto-rows: 200px;
    /* Align-items: center;
    justify-content:stretch;
    justify-items: center;   */
}
.item-1{
  background-color: red;
}
.item-2{
  background-color: blue;
}
.item-3{
  background-color: green;
}
.item-4{
  background-color: yellow;
}
.item-5{
  background-color: gray;
}
.item-6{
  background-color: brown;
}
```

Transitions in CSS.

- **Transitions.**
- transition is a property that specifies how a CSS property should transition from one state to another over a specified period of time.
- **Transitions properties.**
  1. Transition-property.
  2. Transition-duration.
  3. Transition-delay.
  4. Transition-timing-function.
  5. Shorthand property transition.
- **Transitions-timing-function properties.**
  1. Ease: slow start, then fast, then end slowly.
  2. Linear: same speed from start to end.
  3. Ease-in: slow start.
  4. Ease-out: slow end.
  5. Ease-in-out: slow start and end.
  6. Step-start.
  7. Step-end.
  8. Step(4,end).
  9. Cubic-bezier: lets you define your own values.
- **Shorthand property transition.**
- transition: all 2s linear 1s;
- **HTML code.**
-
```
<div class="container">
     <div id="box"></div>
</div>
```
- **CSS code.**
-
```
#box{
 width:150px;
  height:150px;
  background-color: pink;
  transition-property:all;
  transition-duration: 2s;
  transition-timing-function: linear;
  transition-delay: 2s;                         }
#box:hover{
  width: 300px;
  height: 300px;
  background: red;                              }
```

- @media only screen and (max-width:960px){
  #box{
    width: 300px;
    height: 300px;   }   }

Transform in CSS.

- **Transform.**
- **Transform properties.**

1. Transform: translate().
2. Transform: scale().
3. Transform: rotate().
4. Transform: skew().

- **Transform-2D values.**

1. Rotate(angle) : Defines a 2D rotation, the angle is specified in the parameter like positive 30deg and negative – 30deg.
2. Translate(x, y): Defines a 2D translation.
3. Translate X(x): Defines a translation, using only the value for the X-axi.
4. Translate Y(y): Defines a translation, using only the value for the Y - axi.
5. Scale (x, y): Defines a 2D scale transformation.
6. Scale X(x): Defines a scale transformation by giving a value for the X-axis.
7. Scale Y(y): Defines a scale transformation by giving a value for the y-axis.
8. Skew (x-agnle, y-angle): Defines a 2D skew transformation along the X- and the Y-axis.
9. Skew X(angle): Defines a 2D skew transformation along the X-axis.
10. Skew Y(angle): Defines a 2D skew transformation along the Y-axis.
    Matrix (n, n, n, n ,n, n): Defines a 2D transformation, using a matrix of six values.
11. None: Defines that there should be no transformation.

- **CSS code.**

-
  ```
  #box{
    width:150px;
    height:150px;
    background-color: pink;
    transform: rotate(60deg);
    transform: skew(30deg);
    transform: skewX(30deg);
    transform: skewY(30deg);
    transform: skew(30deg, 20deg);
    transform-origin: left top; creating a transform origin points.
    transform: translate(30px 100px);
    transform: translateX(100px);
    transform: translateY(100px);
    transform: scale(2);
    transform: scaleX(2);
    transform: scaleY(2);
    transform-origin: left top; creating a transform origin points.
    transform: matrix(1,-0,3,0,1,0,0);
    transform: none;             }
  matrix(scaleX, skewY, sekewX, scaleY, translateY, translateX());
  ```
- how to used animations.
  ```
  #box{
    width:150px;
    height:150px;
    background-color: pink;
    transform: scale(1);
    transform: translate(0, 0);
    transform: rotate(0deg);
    transition: tranlate 1s;
    transform-origin: left top;
  }
  #box:hover{
    transform: scale(2);
    transform: translate(50px, 50px);
    transform: rotate(360deg);
  }
  ```

Translate 3D in CSS.

- **Translate 3D in CSS.**
- matrix3d (n, n ,n ,n ,n, n, n, n, n, n, n, n, n, n, n, n) Defines a 3D transformation, using a 4x4 matrix of 16 values.
- translate3d(x,y,z)   Defines a 3D translation.
- translateX(x): Defines a translation, using only the value for the X-axis.
- translateY(y): Defines a translation, using only the value for the Y-axis.
- translateZ(z):Defines a 3D translation, using only the value for the Z-axis.
- scale3d(x,y,z): Defines a 3D scale transformation.
- scaleX(x): Defines a scale transformation by giving a value for the X-axis.
- scaleY(y): Defines a scale transformation by giving a value for the Y-axis.
- scaleZ(z): Defines a 3D scale transformation by giving a value for the Z-axis.
- rotate3d(x,y,z,angle): Defines a 3D rotation.
- rotateX(angle): Defines a 3D rotation along the X-axis.
- rotateY(angle): Defines a 3D rotation along the Y-axis.   .
- rotateZ(angle): Defines a 3D rotation along the Z-axis.
- perspective(n): Defines a perspective view for a 3D transformed element.
- initial Sets this property to its default value. Read about initial.
- Inherit: Inherits this property from its parent element. Read about inherit.
- **HTML code.**
-
```
<div class="container">
     <div id="box">Shaan</div>
</div>
```
- **CSS code.**
-
```
.container{
  width: 300px;
  height: 250px;
  background-color: white;
  font-size: 18px;
  text-align: center;
  margin: 100px 0 0 100px;
}
```

```
#box{
  width:300px;
  height:250px;
  background-color: pink;
  transform: perspective(800px) rotateX(20deg);
  transform-origin: left center;
 transform: perspective(800px) rotateY(30deg);
 transform: perspective(800px) rotateY(-50deg);
  transform: perspective(800px) rotateZ(60deg);
  transform: perspective(800px) rotateY(60deg) translateZ(0px);
  transform: perspective(800px) rotateY(60deg) translateX(30px);
  transform:perspective(800px)rotateY(60deg)
translate3D(50px,50px,0px);
  transform: perspective(800px) rotateX(60deg) scaleZ(1);
  transform: perspective(800px)  scale3D(2,2,1) rotateX(60deg);
}
```

Animation in CSS.

- **Animation.**
- An animation lets an element gradually change from one style to another.
- **Animation properties.**
  1. Animation-name.
  2. Animation-duration.
  3. Animation-timing-function.
- **Animation value.**
- Ease: slow start, then fast, then end slowly.
- Linear: same speed from start to end.
- Ease-in: slow start.
- Ease-out: slow end.
- Ease-in-out: slow start and end.
- Step-start: Equivalent to steps(1, start).
- Step-end: Equivalent to steps(1, end).
- Step(#,start/end).
- Cubic-bezier: lets you define your own values(0 to 1).
  4. Animation-delay.
  5. Animation-iteration-count.
  6. Animation-direction.
     (value: Normal, Reverse, Alternate, Alternate-Reverse).
  7. Animation-fill-mode.
     (value: None, Forwards, Backwards, Both.)
  8. Animation-play-state. (value: paused running).
  9. Animation-shorthand for all properties.
- **Syntax of animation.**
- @keyframes animation-example{
  From 0%{CSS properties}
  To 100% {CSS properties}
  }
- **HTML code.**
  <h1>Shaan Animation in CSS.</h1>
  <div class="card">Shaan Animation.</div>

- **CSS code.**

-
  ```
  h1{
    margin-bottom: 50px;
    font-size: 50px;
    text-align: center;
  }
  .card{
    width: 300px;
    height: 300px;
    position: absolute;
    top: 50%;
    left: 50%;
    line-height: 300px;
    margin: -150px;
    background: white;
    color: #03446A;
    text-align: center;
    font-size: 40px;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.1);
    animation-name: card;
    animation-duration: 2s;
    animation-timing-function: ease;
    animation-delay: 2s;
    animation-iteration-count: infinite;
    animation-direction: alternate;
    /* animation: card 2s infinite alternate; */
  }
  @keyframes card{
      from {border-radius: 0%;}
      25%{background-color: green;}
      50%{background-color: yellow;}
      to  {border-radius: 50%; background-color: red; color: black;}
    }
  ```

CSS Variables in CSS.

- **CSS Variables.**
- CSS variables, also known as CSS custom properties, allow you to define reusable values in your CSS code. They provide more flexibility and maintainability to your styles.
- **Declaration and Syntax.**
- Declare a CSS variable with the -- prefix followed by a name. For example: --main-color.
- Assign a value to the variable using the var() function, like this: var(--main-color).
- Variables can store various types of values, such as colors, numbers, text, and more.
- --main-color: #3498db;
- **Variable Definition.**
- You can define variables in various places, including at the root level, within a selector, or even inside a specific CSS rule.
- :root { --main-color: #3498db; }
- **Variable Usage.**
- Use the var keyword

```
.button {
  ! background-color: var(--main-color);
}
```

- **CSS code.**

```
:root{
  --main-color: brown;
  --head-font: arial;
  --p-font: black;                          }
.heading{
  width: 99%;
  height: 800px;
  border: 5px solid black;
  font-size: 30px;
  background-color: var(--head-font);
  color: white;                             }
```

Specificity in CSS.

- **Specificity in CSS.**
- CSS specificity is a concept that determines which CSS rules are applied to an HTML element when multiple conflicting rules exist. It helps browsers decide which style declarations should take precedence when more than one rule targets the same element. Specificity is a fundamental part of understanding CSS, and it's crucial for resolving styling conflicts
- Every CSS selector has its place in the specificity hierarchy.
- **Selectors in CSS.**
  1. *(universal selector).
  2. Element / pseudo-element.
  3. Class or pseudo-class.
  4. Id.
  5. Inline style.
  6. !important.
- **Universal selector:** The universal selector has 0 specificity (No priority)
- **Element:** Includes only element selectors, such as h1, img, p::before. For each ID in a matching selector, add 0-0-0-0-1 to the weight value.
- **Class:** Includes class selectors, such as .myClass, [type="radio"], :hover. For each class, attribute selector, or pseudo-class in a matching selector, add 0-0-0-1-0 to the weight value.
- **Id selector:** Includes only ID selectors, such as #btn. For each ID in a matching selector, add 0-0-1-0-0 to the weight value.
- **Inline style:** For each inline style ex
  <div class="container" style="background: #000;">,
  add 0-1-0-0-0 to the weight value.
- **Important:** For each important keyword ex h1 {color: red! important} add 1-0-0-0-0 to the weight value.

New CSS Features.

- **New features.**
- **New features.**
  1. :is, :has, :not, :where.
  2. Media query range syntax.
  3. Container queries.
  4. Accent-color.
  5. Aspect ratio.
  6. Scroll snap.
  7. Individual transform properties.
  8. CSS nesting.
  9. Grap property for flexbox.
  10. CSS logical properties (inline and blocks).
  11. CSS writing mode.
  12. :focus-visible.
- **CSS code.**
- **1.:is, :has, :not, :where.**

```
:is(.parent-1, .parent-2){
  border: 5px solid black;
  color: red;                          }
.parent-1:has([type="checkbox"]:checked)p{
  color:brown;                         }
input[type="checkbox"]{
  accent-color: pink;
  width: 50px;
  aspect-ratio: 1;
}
not
nav ul li:first-child{
padding:0px;
}
```

10.CSS logical properties (inline and blocks).

```
body{
  background-color:palegoldenrod;
  border: 5px solid black;
  margin-top: 100px;
  color:red;
  border-inline: 5px solid black;
  border-block: 5px solid red;
}
```

8.CSS nesting.

```
.heading{
  border: 5px solid black;
   & h1{
    text-align: left;
    font-size: 40px;
    color: red;
  }
  p{
    font-size: 20px;
    letter-spacing: 1px;
    margin: 2rem 0 4rem 0;
  }
}
```

11.CSS writing mode.

```
.heading{
  border: 5px solid black;
   & h1{
    text-align: left;
    font-size: 40px;
    color: red;
    writing-mode: vertical-lr;
    position: absolute;
    top:5%;
    left: 5%;
  }
  p{
    font-size: 20px;
    letter-spacing: 1px;
    margin: 2rem 0 4rem 0;
}
}
```

Container queries in CSS.

- **Container queries.**
- Container queries enable you to apply styles to an element based on the size of the element’s container.
- **Container types.**
- Container types CSS properties is used to define the type of containment used in a container query.
- Container-types: normal;
- Container-types: size;
- Container-types: inline-size;
- **HTML code.**

```
<div class="container-1">
   <div>
     <img src="C:\Users\Admin\Desktop\New folder\1.jpg" alt="Italian
Trulli">
     <h1>CSS container queries</h1>
   </div>
  </div>
  <br/>
  <div class="media">
   <div>
     <img src="C:\Users\Admin\Desktop\New folder\1.jpg" alt="Girl in a
jacket">
     <h1>CSS container queries</h1>
   </div>
</div>
```

- **CSS code.**
- body{

```
  background-color:palegoldenrod;                }
.container-1,
.media{
  max-width: 800px;
  height: auto;
  border: 5px solid black;
  margin-right:200px;
  & div{
    display: grid;
    grid-template-columns: 1fr 0.5fr;
    align-items: center;      }                                        }
.container-1{
  container-type:inline-size;
  container-name:shanelhai;
}
.img{
  width: 100%;
  height: auto;
}
h1{
  text-align: center;
  color: red;
  font-weight: bold;
  text-transform: capitalize;
  font-size: 32px;
  text-shadow: -3px 10px 10px rebeccapurple;
  animation: floating 2s linear infinite alternate;
}
@container-1 shanelhai(width<600px){
  :is(.media,.container-1)div{
  grid-template-columns: 1fr;
  }
  h1{
    color: orange;
    text-transform: uppercase;
    padding: 20px 0;                    }                }
@media(width<600px){
  :is(.media,.container-1)div{
  grid-template-columns: 1fr;
  }
  h1{
    color: orange;
    text-transform: uppercase;
    padding: 20px 0;                    }                              }
```

Bonus in CSS.

- **Bonus in CSS.**
- **Bonus.**
  1. Floats.
  2. CSS Clip path.
  3. CSS AOS plugin.
  4. CSS shape divider.
  5. Scroll bar styling.
  6. Website References.
- **Float.**
- **HTML code.**

```
<div class="contanir">
        <div class="logo-brrand">
          <a href="#">Shanelhai</a>
        </div>
        <nav class="navbar">
          <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">Home</a></li>
            <li><a href="#">Home</a></li>
            <li><a href="#">Home</a></li>
            <li><a href="#">Home</a></li>
          </ul>
        </nav>
      </div>
```

- **CSS code.**
- .header{

```
  width: 100%;
  height: 10vh;
  background-color: purple;
  color: white;                }
a{
  color: white;
  font-size: 40px;             }
.contanir{
  width: 90vw;
  margin: 0 auto;              }
```

```
.logo-brrand {
   width: 30%;
   line-height: 10ch;
   background-color: #662549;
   font-size: 24px;
   float: left;
   text-align: center;                 }
.navbar{
   width: 70%;
   line-height: 10ch;
   background-color: #662549;
   font-size: 24px;
   float:right;
   text-align: center;        }
ul li{
  list-style: none;
  display: inline-block;
  padding:0 25px;
  border: inset solid black;          }
```

- **HTML clip path code.**

```
<h1>clip path</h1>
   <img src="C:\Users\Admin\Desktop\New folder\1.jpg" alt="Girl in a
jacket">
```

- **CSS Clip path.**
- body{

```
  width: 100%;
  height: 100vh;
  background-color: hsl(0, 0%, 91%);
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  gap: 50px;                          }
img{
  width: 30%;
  border-radius: 10px;
  clip-path: polygon(50% 0%, 0% 100%, 100% 100%);    }
```

- **CSS SHAPE DIVIDER.**
- **CSS code.**

```
<div class="hero">
   <div class="content">
     <h1>Shanelehai</h1>
   </div>
   <div class="custom-shape-divider-bottom-1705038209">
     <svg   data-name="Layer   1"   xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 1200 120" preserveAspectRatio="none">
         <path  d="M321.39,56.44c58-10.79,114.16-30.13,172-41.86,82.39-
16.72,168.19-17.73,250.45-
.39C823.78,31,906.67,72,985.66,92.83c70.05,18.48,146.53,26.09,214.34,
3V0H0V27.35A600.21,600.21,0,0,0,321.39,56.44Z"            class="shape-
fill"></path>
     </svg>
   </div>
  </div>
```

- **CSS code.**
- .hero{

```
  background-image:linear-
gradient(90,2deg,rgba(75,68,229,1)2%rgba(97,212,202,1)98.3%);
  position: relative;
  overflow: hidden;
  display: flex;
  align-items: center;
  justify-content: center;
  border: none;
  height: 90vh;                                   }
.here .content h1{
  font-size: 45px;
  font-weight: 700px;
  font-family:'montserrat' , sans-serif;          }
.here .content h1{
  font-family:'montserrat' , sans-serif ;
```

```
}.custom-shape-divider-bottom-1705038209 {
  position: absolute;
  bottom: 0;
  left: 0;
  width: 100%;
  overflow: hidden;
  line-height: 0;
  transform: rotate(180deg);                }
.custom-shape-divider-bottom-1705038209 svg {
  position: relative;
  display: block;
  width: calc(144% + 1.3px);
  height: 115px;                         }
.custom-shape-divider-bottom-1705038209 .shape-fill {
  fill: #FFFFFF;                }
```

- **Scroll bar styling.**
- **CSS code.**

```
::-webkit-scrollbar {
  width: 40px;
}
::-webkit-scrollbar-track {
  background:green;
}

::-webkit-scrollbar-thumb {
  background:red;
}
::-webkit-scrollbar-thumb:hover {
  background: blue;
}
```

- **HOW TO CENTER A DIV INSIDE DIV IN CSS.**
- **CSS code.**

```
.contener{
    width: 100%;
    height: 640px;
    border: 4px solid black;
    background-color: red;
    /* position: relative; First trcik to center div*/
    /* display: flex;
    justify-content: center;
    align-items: center; Second trcik to center div*/
    /* display: grid;
    place-items: center; Third trcik to center div*/
}
.box-1{
    width: 60%;
    height: 400px;
    border: 4px solid black;
    background-color: green;
    /* position: absolute;
    left: 50%;
    top: 50%;
    transform: translate(-50%, -50%); First trcik to center div*/
}
```

- **how to center text in div.**
- **CSS code.**
- display: flex;

```
display: flex;
justify-content: center;
align-items: center;
text-transform: capitalize;
font-size: 50px;
```

- **image Carousal in CSS.**
- **HTML code and CSS code.**

-
  ```
  <div class="container-img">Shaan</div>
  body{
    height: 100vh;
    display: grid;
    place-items: center;
  }
  .container-img{
    width: 60vw;
    height: 50rem;
    border-radius: 1rem;
    box-shadow: rgba(100,100,111,0.2)0px 7px 29px 0px;
    background-image:                    url("https://images.unsplash.com/photo-
  1500622944204-
  b135684e99fd?w=500&auto=format&fit=crop&q=60&ixlib=rb-
  4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8M3x8bmF0dXJhbHxlbn
  wwfHwwfHx8MA%3D%3D");
    background-size:cover ;
    animation: gallery 20s linear infinite;
  }
  @keyframes gallery {
    20%{
      background-image:url("https://images.unsplash.com/photo-");
    }
    40%{
      background-image:url("https://plus.unsplash.com/premium_photo-") ;
    }
   60%{
    background-image:url("https://plus.unsplash.com/premium_photo-");
  }
    80%{
      background-image:url("https://images.unsplash.com/photo-");
    }
  }
  ```

Vendor Prefixes in CSS.

- **Vendor Prefixes.**
- A vendor prefixes is a special prefix added to a CSS property. Each rendering engine has it is own prefix which will only apply the property to that particular browser.
- **Vendor Prefixes Properties.**
- The -webkit- prefixed property will work in WebKit-based browsers such as Chrome and Safari.
- The -moz- prefixed property will work in Firefox.
- The -ms- prefixed property will work in Internet Explorer.
- The -o- prefixed property will work in Opera.
- **CSS code.**

```
nav {
  background-color: red;
  background-image: url(gradient-slice.png);
  background-image:-webkit-linear-gradient(top right, #A60000,
#FFFFFF);
  background-image: -moz-linear-gradient(top right, #A60000, #FFFFFF);
  background-image: -ms-linear-gradient(top right, #A60000, #FFFFFF);
  background-image: -o-linear-gradient(top right, #A60000, #FFFFFF);
  background-image: linear-gradient(top right, #A60000, #FFFFFF);
}
```

BEM (Block, Element, Modifier) Methodology in CSS.

- **BEM (Block, Element, Modifier) Methodology.**
- That is rule: .item-1—child{}
- **SMACSS (Scalable and Modular Architecture for CSS)**
- SMACSS (Scalable and Modular Architecture for CSS) is a style guide and methodology that helps make CSS more consistent and maintainable. It's a collection of guidelines that teach how to write scalable and readable CSS.

SEO IN CSS.

- **Seo in CSS.**
- SEO (Search Engine Optimization) is the process of making a website more visible in search results, also termed improving search rankings.
- **Seo features.**
- Include a descriptive and keyword-rich <title> tag within the
- <head> section of your HTML document.
- Meta Description Tag.
- Proper Heading Tags.
- Image Optimization.