Unit - IV

4.1 CSS Formatting

Fonts

CSS fonts properties are used to set fonts for the text on a webpage. Some important font properties are

- font-family
- font-style
- font-weight
- font-size
- Google Fonts

font-family

This property is used to specify a list of fonts for an element.

Syntax

font-family: <family-name>, <generic-name>;

- **family-name**: name of a specific font given in quotes
- **generic-name**: used as a fallback mechanism to specify a font. Given as the last font in the list.

Generic Font Names

- serif: **stroke** at end
- sans-serif: **no stroke** at end
- cursive: appears like **handwritten** text
- monospace: each character same **width**
- fantasy: **decorative** text

font-style

This property is used to specify font style to be displayed such as italics, oblique or normal

Syntax

font-style: italics;

font-weight

This property is used to specify the weight of the font such as bold, bolder, lighter, normal or numeric value in the range 100-900

Syntax

font-weight: bold;

font-size

This property is used to set the size of the font in px, em or %. Font size can be set based on the size of the viewport using vw value.

Syntax

font-size: 5px;

Google Fonts

CSS allows the usage of fonts provided by google using link tags. This helps the developer to display fonts that are not present in the system.

Syntax

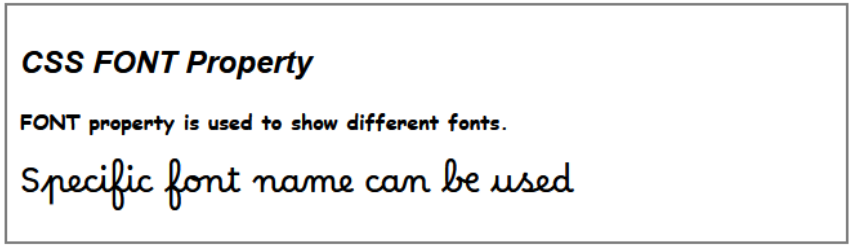
```
<link rel="stylesheet"
  href="https://fonts.googleapis.com/css?family = Borel">
```

Example

```
h1{
```

```
    font-family:serif;
}
h2{
    font-family: sans-serif;
    font-style: italic;
}
p{
    font-family: cursive;
    font-weight: bold;
}
section{
    font-family: Borel;
    font-size: 4vw;
}
<article>
    <h2>CSS FONT Property</h2>
    <p>
        FONT property is used to show different fonts.
    </p>
    <section>
        Specific font name can be used
    </section>
</article>
```

Output



Kerning

Kerning is the process of adjusting the space between individual letters in a word to improve readability and aesthetics. It is implemented by using the **letter-spacing** property.

Syntax

```
letter-spacing: spaceValue;
```

Leading

Leading in CSS, also known as **line height**, refers to the vertical space between lines of text. It is implemented using the **line-height** property.

Syntax

```
line-height: spaceValue;
```

Indenting

Indenting in CSS refers to the spacing between the left margin and the start of the first line of text. It is implemented using the **text-indent** property.

Syntax

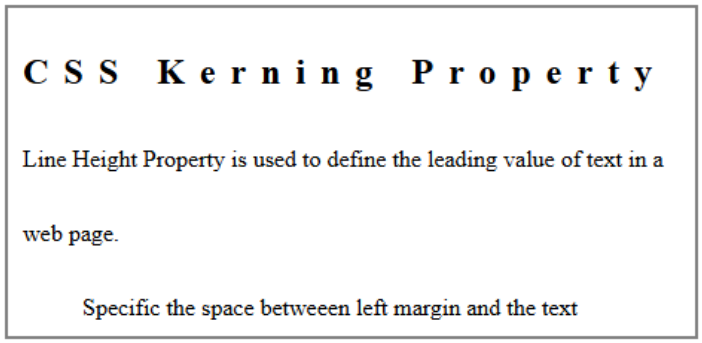
```
text-indent: spaceValue;
```

Example

```
h2{
    letter-spacing: 10px;
  }
p{
    line-height: 50px;
  }
section{
    text-indent: 40px;
  }
```

```
<article>
  <h2>CSS Kerning Property</h2>
    <p>
      Line Height Property is used to define the leading value of text in a web page.
    </p>
    <section>
      Specific the space betweeen left margin and the text
    </section>
</article>
```

Output



CSS Comments

Comments are used to provide documentation for the source code. It is not displayed in the browser. CSS comments are placed inside <style> tag.

Syntax

```
/*This is a comment*/
```

CSS Colors

In CSS colors play an important role in styling HTML documents. Colors can be specified using

- Color Names
- RGB, RGBA
- HEX
- HSL, HSLA

Color Names

In this colors are specified using predefined names. For example Blue, Coral, Consilk, YellowGreen, etc.

Example

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

RGB, RGBA

In this the color is given as a combination of red, green and blue values. Each color values if given in the numeric range 0-255

Syntax

```
rbg(red, green, blue);
```

Example

```
h1{
        color: rgb(255,0,0);
 }
```

RGBA adds alpha channel which specifies opacity for the color. The alpha value is given in the range 0.0 to 1.0. 0.0 specifies full transparency and 1.0 specifies no transparency.

Syntax

```
rgba(red, green, blue, alpha)
```

Example

```
h1{
```

```
        color: rgba(255,0,0,0.5);
    }
```

HEX value

In this the color is specified as **hexadecimal value** ranging from 00 to FF. It is a 6 digit value where first 2 specify red color value, next 2 specify green color value and last 2 specify blue color value.

Syntax

```
    #RRGGBB
```

Example

```
    h1{
        color:#00FF00;/*Green Color*/
    }
```

HSL, HSLA

In this the color is specified using hue, saturation and lightness value. Hue is a degree on color wheel, where 0 denotes red color, 120 denotes green color and 240 denotes blue color. Saturation is a percentage value, where 0% denotes gray and 100% denotes full color. Lightness is a percentage value, where 0% is black and 100% is white.

Syntax

```
    hsl(hue, saturation, lightness)
```

Example

```
    h1{
        color:hsl(0,100,50);
    }
```

HSLA adds alpha value to the color. The alpha value is given in the range 0.0 to 1.0. 0.0 specifies full transparency and 1.0 specifies no transparency.

Syntax

```
    hsla(hue, saturation, lightness, alpha)
```

Example

```
    h1{
        color:hsla(0,100,50,0.5);
    }
```

Background

The Background Property in CSS is used to define the background effects of an element. Some important background properties are

background-color: This is used to set the background color of an element.

Syntax

```
    background-color: colorvalue:
```

background-image: This is used to set the background image of an element.

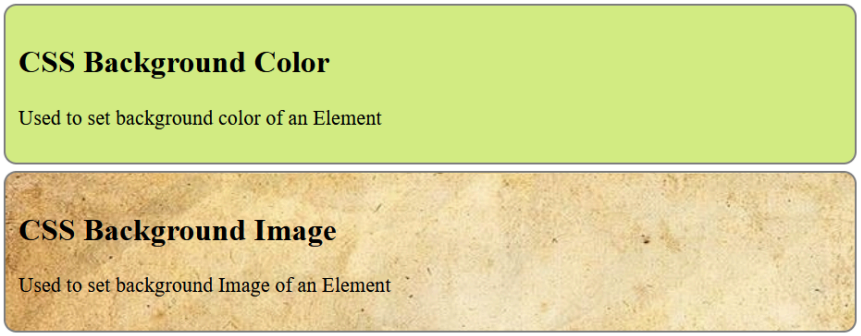
Syntax

```
    background-image: url(“Image URL”);
```

Example

```
article{
        width: auto;
        border: 2px solid gray;
        padding: 10px;
        border-radius: 10px;
        margin:5px;
        background-color: #d2ed83;
        }
div{
        width: auto;
        border: 2px solid gray;
        padding: 10px;
        border-radius: 10px;
        margin:5px;
        background-image: url("oldnote.jpg");
        background-color: #DFD2A6;
        }

<article>
   <h2>CSS Background Color</h2>
   <p>
        Used to set background color of an Element
   </p>
</article>
<div>
   <section>
        <h2>CSS Background Image</h2>
        <p>
         Used to set background Image of an Element
        </p>
   </section>
</div>
```

Output



background-position: This is used to set the position of the image within the element. Position is provided as top left right bottom (or) pixel position.

Syntax

```
background-position: right bottom;
```

background-repeat: This is used to define the direction in which an image is to be repeated. Values that are given for this property are repeat, repeat-x, repeat-y and no-repeat

Syntax

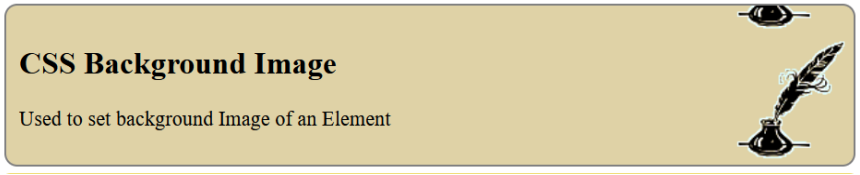
```
background-repeat: repeat-y;
```

Example

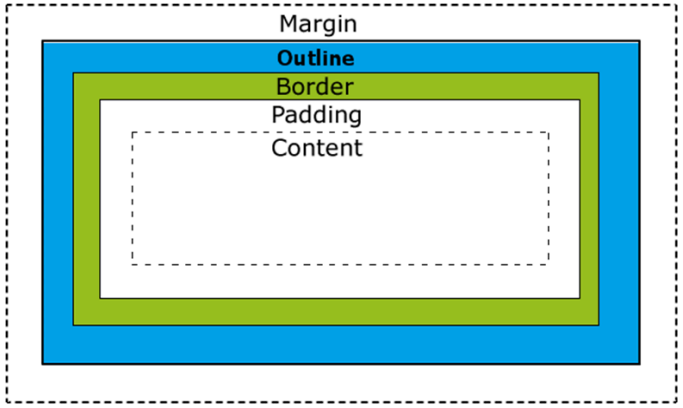
```
div{
    width: auto;
    border: 2px solid gray;
    padding: 10px;
    border-radius: 10px;
    margin:5px;
```

```
    background-image: url("pen1.png");
    background-position: right bottom;
    background-repeat: repeat-y;
    background-color: #DFD2A6;
 }
<div>
    <section>
        <h2>CSS Background Image</h2>
        <p>
         Used to set background Image of an Element
        </p>
    </section>
 </div>
```

Output



CSS Box Model



Borders

The CSS Border property is used to specify the border of the box representing an element. Some important border properties are

border-color: This is used to set the color of the border

Syntax

```
    border-color: red;
```

border-style: This is used to set the border line style. Examples of line style are dotted, dashed, solid, double, none, hidden, etc.

Syntax

```
    border-style: dotted;
```

border-width: This is used to set the width of the border line.

Syntax

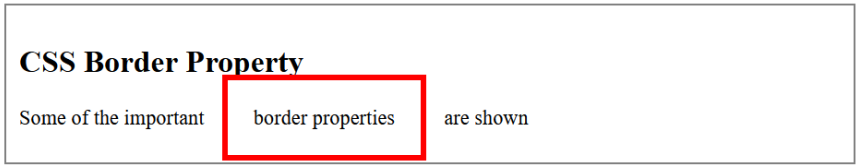
```
    border-width: 3px;
```

Example

```
 span{
    border-width: 5px;
    border-style: solid;
    border-color: RED;
    margin: 10px;
    padding: 20px;
 }

 <article>
    <h2>CSS Border Property</h2>
    <p>
```

Some of the important border properties are shown
</p>
</article>

Output



Outline

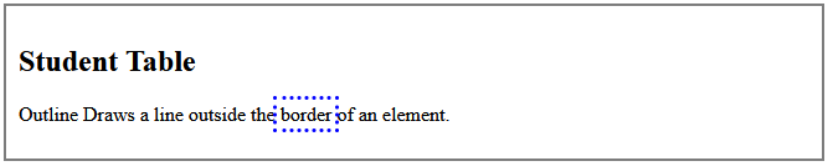
In CSS, the outline property is used to create a visible outline around an element. The outline property is separate from the element's border and does not affect the element's layout. Some of the outline properties are

- **outline-style**: Set the line style
- **outline-color**: Set the color of the line
- **outline-width**: Set width of the line
- **outline-offset**: space between outline and border

Example

```
span{
    outline-style: dotted;
    outline-color: blue;
    outline-width: 3px;
    outline-offset: 3px;
}
<article>
    <h2>Student Table</h2>
    <p>Outline Draws a line outside the <span>border</span> of
        an element
    . </p>
</article>
```

Output



Margin

The CSS margin property is used to set the margin space for an element. This sets the space between adjacent elements. Margin can be set separately using margin-top, margin-right, margin-bottom and margin-right or for all the 4 sides using margin.

Syntax

```
margin: 10px;
```

Example

Same as border Example

Padding

The CSS padding property is used to set space around an element within its border. Padding can be set separately using padding-top, padding-right, padding-bottom and padding-left or for all the 4 sides using padding.

Syntax

```
padding: 10px;
```

Example

Same as border Example

List

HTML list elements display property can be changed using CSS. This is done using **list-style-type** property. For unordered list styles such as circle, square and disc can be shown. For ordered list styles such as decimal, upper-alpha can be shown. To display an image as list style **list-style-image** property is used.

Example

```
ul{
        list-style-type: square;
        #list-style-image: url("dept.png");
}
ol{
        list-style-type: upper-alpha;
}
<section>
        <h4>Style Sheet Types</h4>
        <ul>
        <li>Embedded</li>
        <li>External</li>
        <li>In-Line</li>
        </ul>
        <ol>
        <li>HTML5</li>
        <li>CSS3</li>
        <li>JavaScript</li>
        </ol>
</section>
```

Output

Tables

HTML tables by default do not show border lines and other styles. CSS is used to display borders and other styles to the table. Some of the most frequently used CSS properties related to tables are

- **border**: This property is used to set borders for tables.
- **border-collapse**: This property is used to combine the double border created by border property into a single border.
- **width, height**: These two properties are used to set the width and height of the table.
- **text-align**: This property is used to set the text alignment within the table.

Example

```
table, th, td{
    border: 2px dotted;
}
table{
    border-collapse: collapse;
    width: 100%;
```

```
    height: 100px;
}
th{
    text-align: right;
}
<article>
    <h2>Student Table</h2>
    <table>
      <tr>
         <th>S.No.</th>
         <th>Name</th>
      </tr>
      <tr>
         <td>1</td>
         <td>Arvind</td>
      </tr>
      <tr>
         <td>2</td>
         <td>Bashir</td>
      </tr>
      <tr>
         <td>3</td>
         <td>Charan</td>
      </tr>
    </table>
</article>
```

Output

Student Table

S.No.	Name
1	Arvind
2	Bashir
3	Charan

Table Special Effects

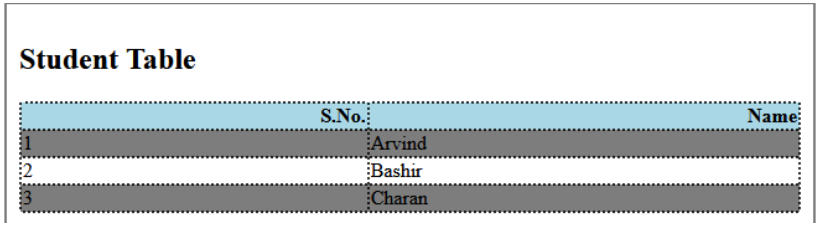
Using CSS some special effects can be added to the table like highlighting a row when the mouse is placed above it and creating stripped tables using the **nth-child()** selector.

Example

```
table, th, td{
    border: 2px dotted;
}
table{
    border-collapse: collapse;
    width: 100%;
}
tr:nth-child(even){
    background-color: grey;
}
tr:hover{
    text-align: right;
    background-color: lightblue;
}
<article>
    <h2>Student Table</h2>
    <table>
```

```
    <tr>
      <th>S.No.</th>
      <th>Name</th>
    </tr>
    <tr>
      <td>1</td>
      <td>Arvind</td>
    </tr>
    <tr>
      <td>2</td>
      <td>Bashir</td>
    </tr>
    <tr>
      <td>3</td>
      <td>Charan</td>
    </tr>
  </table>
</article>
```

Output



Forms

HTML forms are used to get input from the user. The form elements can be styled using CSS. For this element selector is used with minor changes. Modified selectors used for designing forms are

- **input[type=text]**: This selects all the text input elements.
- **input[type=button]**: This selects all the button element
- **select**: This selects all the dropdown element

The Following CSS properties can be applied to the form elements.

- Width
- Padding
- Border
- Margin
- Color
- Hover, focus
- Icon/Image

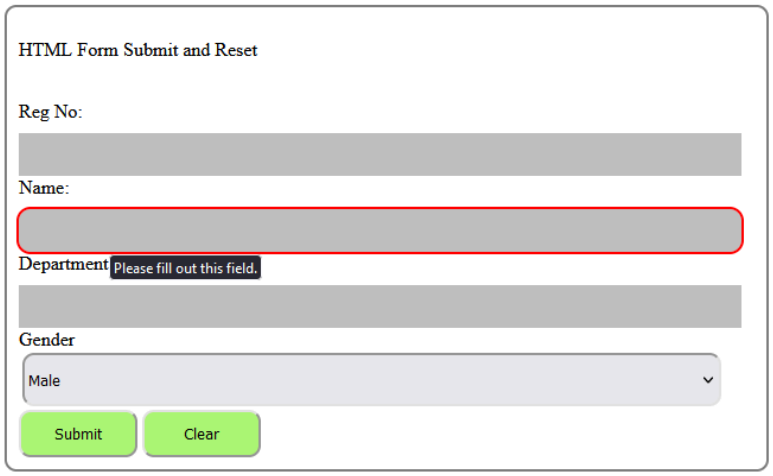
Example

```
input[type=text]{
    display: block;
    width: 95%;
    padding: 10px;
    margin-top: 10px;
    border: none;
    background-color: rgba(127,127,127,0.5);
}

input[type=text]:hover{
    outline: 2px solid red;
```

```
    border-radius: 10px;
}

input[type=text]:focus{
    background-color:white;
    color: black;
    outline:none;
    border: none;
    border-radius: 10px;
    border-bottom: 2px solid red;
}
input[type=text].dept:focus{
    background-image:url("dept.png");
    background-position: 10px 5px;
    background-repeat: no-repeat;
    padding-left: 50px;
}
select{
    display: block;
    width: 95%;
    height: 45px;
    padding: 3px;
    margin: 3px;
    border-radius: 10px;
}
select:hover{
    outline: 2px solid red;
    border-radius: 10px;
}
input[type=submit], input[type=reset]{
    width: 100px;
    height: 40px;
    border-radius: 10px;
    padding: 3px;
    background-color: #aef777;
}
input[type=submit]:hover, input[type=reset]:hover{
    outline: 2px solid red;
    border-radius: 10px;
    color: #aef777;
    background-color: grey;
}
<article>
    <p>HTML Form Submit and Reset</p><br>
    <form id="myform">
        <div>
        <label for="regno" >Reg No: </label>
        <input type="text" id="regno" required>
        </div>
        <div>
        <label for="txtname" >Name: </label>
        <input type="text" id="txtname" required>
        </div>
        <div>
        <label for="txtdpt" >Department: </label>
        <input class="dept" type="text" id="txtdpt">
        </div>
        <div>
        <label for="selgndr">Gender</label>
        <select id="selgndr">
         <option value="M">Male</option>
         <option value="F">Female</option>
```

```
        </select>
        </div>
        <div>
        <input type="submit" value="Submit">
        <input type="reset" value="Clear">
        </div>
    </form>
</article>
```

Output



Links

HTML links can be styled using CSS. A link has 4 states, normal unvisited link, visited link, mouse over link state and active link. These 4 states can be styled in CSS using the pseudo class of anchor tag, such as

a:link
a:visited
a: hover
a:active

Similarly to remove link style and display it as normal text **text-decoration** property with "**none**" value is used

Example

```
a:link{
        color: RED;
        font-family: monospace;
        font-size: 20px;
}
a:hover{
        text-shadow: 5px 5px 5px blue, 3px 3px 6px yellow;
}
a:visited{
        color: grey;
        font-family: monospace;
        font-size: 10px;
}
a:active{
        color: blue;
        font-family: monospace;
        font-size: 30px;
}
<article>
        <h2>Link Styles</h2>
        <section>
        <a  href="https:\\www.google.com"  taget="_blank">Sample Link</a>
        </section>
</article>
```

Output

4.2 Advanced CSS3

CSS Rounded Corners

In CSS, rounded corners can be created for elements such as boxes, buttons, and divs using the border-radius property. This property helps to control the curvature of an element's corners, giving the design a more visually appealing and modern look.

- **border-radius**: It can have 4 values for each corner or 2 values denoting diagonal values or 1 value that is used for all the four corners.

Syntax

border-radius: 10px; /*All four corners*/

Border Images

CSS border images are a powerful tool for creating decorative borders around HTML elements. They allow for the use of images instead of simple, solid-colored lines to enhance the visual appeal of the web content.

The CSS border-image shorthand notation is a concise way to define multiple properties related to border images. It simplifies the process of creating decorative borders with images. The shorthand notation includes the following properties:

1. border-image-source: This property specifies the path to the image that will be used as the border. It is usually a URL pointing to the image file.
2. border-image-slice: This property defines how the image is sliced into sections that will be used as border pieces. You can control the size and dimensions of these pieces to create the desired effect.
3. border-image-width: This property determines the width of the border image slices, specifying how they will be displayed around the element.
4. border-image-outset: This property specifies how far the border extends beyond the element's box, controlling the spacing between the element and the border image.
5. border-image-repeat: This property sets the tiling or repetition behavior of the border image, allowing you to choose from values like stretch, repeat, or round to achieve different effects.

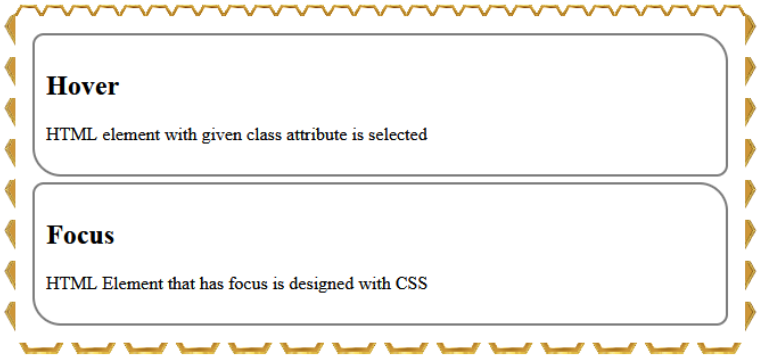
Syntax

border-image: url('border-image.png') 20 10px 5px round;

Example

```
.cont{
    width: auto;
    border: 10px solid transparent;
    padding: 10px;
    margin:5px;
    background-color: transparent;
    border-image: url("bord4.png") 40% 10% 20% 10% round;
}
article{
    width: auto;
    border: 2px solid gray;
    padding: 10px;
    border-radius: 10px 25px 10px 25px;
    margin:5px;
}
<div class="cont">
<article>
    <h2>Hover</h2>
    <p>HTML element with given class attribute is selected</p>
</article>
<article>
    <h2>Focus</h2>
    <p>HTML Element that has focus is designed with CSS</p>
</article>
</div>
```

Output



CSS Background Image

CSS allows you to set background images for HTML elements. Some of the important CSS background image properties are

- **background-image**: This property allows placement of multiple images as background of an element.
- **background-size**: This property is used to set the size of the background image. It takes the values pixel, contain and cover.
- **background-origin**: This property is used to position the image in the background using the box property. It takes the values padding-box, border-box, content-box.
- **background clip**: This property is used to control how the background of an element is clipped or painted in relation to the element's box. It takes the values padding-box, border-box, content-box.

Color Keywords

In CSS, color keywords are predefined words that represent specific colors. These keywords make it easy to set colors without specifying the exact numerical color values. Other than color names some of the important color keywords are

- transparent: This keyword denotes a completely transparent color.
- inherit: This keyword allows an element to inherit its color from its parent element.
- currentcolor: This keyword uses the color property of the element.

Gradient

Gradients in CSS are a way to create a smooth transition between multiple colors or color stops within a specified element. They are used for various design elements, such as backgrounds, borders, and text.

Types of Gradients are

- **linear-gradient**: Linear gradients create a color transition in a straight line.
- **radial-gradient**: Radial gradients create a color transition from a central point outward in a circular pattern.
- **conic-gradient**: Conical gradients create a color transition from a central point outward in a conical pattern.

Example

background: linear-gradient(to right, #FF0000 0%, #FFFF00 50%, #00FF00 100%);

background: conic-gradient(at 50% 50%, red 0%, yellow 25%, green 50%, blue 100%);

background: radial-gradient(circle, #FF0000, #0000FF);

Shadow Effects

In CSS, shadow effects can be applied to elements to create depth and dimension. These effects are commonly used to simulate the appearance of objects casting shadows on a surface. Some important shadow properties are

- **text-shadow**: It is specifically used for adding shadows to text within elements.
- **box-shadow**: It is used to add shadows to elements, typically boxes or containers.

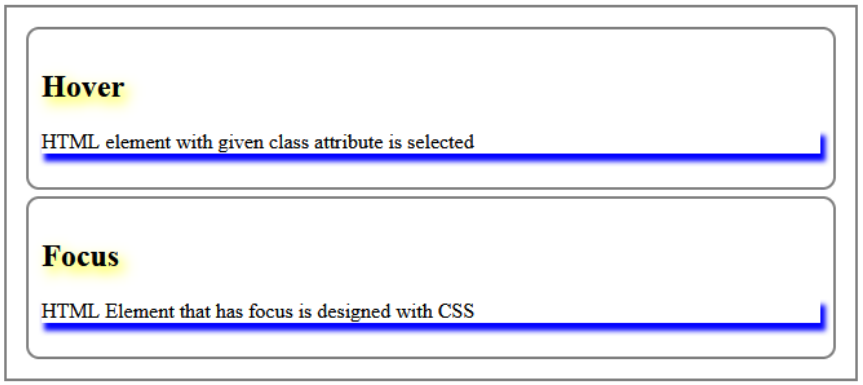
Syntax

```
text-shadow: x y blur color;
box-shadow: x y blur spread color <inset>;
```

Example

```
h1, h2{
   text-shadow: 4px 5px 10px yellow;
}
p{
   box-shadow: 3px 5px 3px 1px blue inset;
}
<article>
   <h2>Hover</h2>
   <p>HTML element with given class attribute is selected</p>
</article>
<article>
   <h2>Focus</h2>
   <p>HTML Element that has focus is designed with CSS</p>
</article>
```

Output



Web Fonts

Web fonts refer to custom fonts that can be used on web pages. They are not limited to the standard fonts that are available on users' devices. These fonts are downloaded from the server. Web fonts are included in a webpage using @font-face rule. It can take font types ttf, otf and woff.

Example

```
@font-face{
    font-family: myTamilFont;
    src: url(Font/Alankaram.otf);
}
h1{
    font-family: myTamilFont;
}
<header>
    <h1>jkpo;</h1>
</header>
```

Output

Text Effects

In CSS, various text effects can be applied to enhance the appearance of text content on the web pages. Text effects allow creation of eye-catching typography and improve the visual appeal of your website. Some of the text effect properties are

- **text-overflow**: This property controls how text behaves when it extends beyond the boundaries of its container. This takes the values clip and ellipsis.
- **word-wrap**: This property is used to control how long words or strings of text are handled when they extend beyond the boundaries of their containing element. This takes the value break-word.
- **writing-mode**: This property is used to define the orientation of text, which controls the flow and direction of text content within an element. This takes the values horizontal-tb, vertical-lr and vertical-rl.

Example

```
div{
    width: 50px;
    overflow: hidden;
    border: 2px solid black;
    //text-overflow: clip;
    word-wrap: break-word;
    //writing-mode: vertical-lr;
```

```
}
<article>
    <h2>Hover/Focus</h2>
    <div>HTML element with given class attributeisselected.</div>
</article>
```

Output

2D Transforms

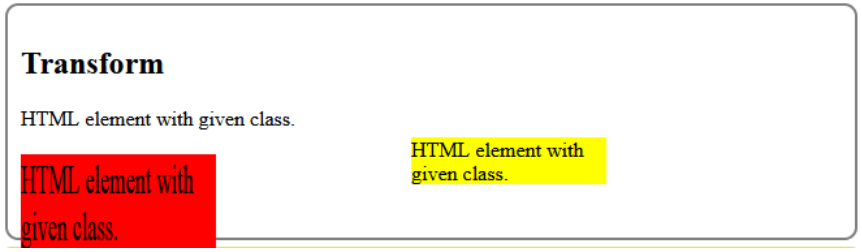
2D transforms in CSS allow the manipulation of the position, rotation, scaling, and skewing of HTML elements in a two-dimensional space. To implement 2D transformation **transform** property is used. Based on the values that are provided various 2D transformations are performed.

- **translate(x,y)**: This function moves an element along the horizontal and vertical axes.
- **rotate(deg)**: This function rotates an element by a specified angle around its center.
- **scale(x/y); [scaleX/scaleY]**: scaleX changes the size of an element on x-axis and scaleY changes the size of an element on y-axis.
- **skew(xdeg, ydeg); [skewX/skewY]**: skewX function skews an element by a specified angle in the horizontal direction and skewY function skews an element by a specified angle in the vertical direction.

Example

```
.div1{
    width: 150px;
    background-color: yellow;
    transform: translate(300px,5px);
    //transform: rotate(45deg);
}
.div3{
    width: 150px;
    background-color: Red;
    transform: scaleY(2);
}
<article>
    <h2>Transform</h2>
    <div>HTML element with given class.</div>
    <div class="div1">HTML element with given class.</div>
    <div class="div3">HTML element with given class.</div>
</article>
```

Output



3D Transforms

3D transforms in CSS enable the manipulation of the position, rotation, and scaling of HTML elements in a three-dimensional space. To implement 3D transformation **transform** property is used.

- **Scale**: The CSS scaleX(), scaleY(), and scaleZ() functions are used for scaling an element along the x, y, and z axes in 3D space.
- **Rotate**: The CSS rotateX(), rotateY(), and rotateZ() functions are used to apply rotation transformations to elements in 3D space.

Buttons

The appearance of HTML button created using <button> tag can be modified using the following CSS properties.

- border
- border-radius
- background-color
- color
- hover

Example

```
.danger{
    background-color: rgba(0,125,0,0.5);
    font-size: 24px;
    border-radius: 10px;
}
.danger:hover{
    background-color: rgba(123,0,0,0.5);
}
<div>
    <button class="danger">
    <span class="material-icons">edit</span>
    Edit
    </button>
</div>
```

Output

Counters

CSS counters are a way to automatically number elements in ordered lists or other elements. It has the following properties.

- **counter-reset**: This creates a counter for providing automatic numbering
- **counter-increment**: This increments the counter value by one.
- **content**: This is used to insert any text content.
- **counter()**: This is used to add the counter value.

Example

```
body{
    counter-reset: heading;
}
h2::before{
    counter-increment: heading;
    content: "H2 Tag " counter(heading) ": ";
}
<article>
```

```
    <h2>Hover</h2>
    <div>Hover CSS</div>
    <h2>Active</h2>
    <div>Active CSS</div>
    <h2>Focus</h2>
    <div>Focus CSS</div>
</article>
```

Output

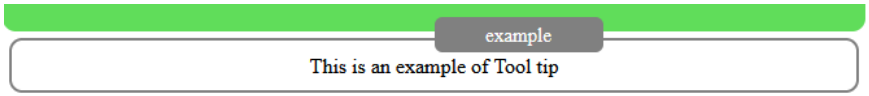
CSS ToolTip

Tooltips in CSS are informational pop-up text that appears when a user hovers over an element. These tooltips are commonly used to provide additional information about an element or to enhance user experience. tag is used to contain the tooltip text. Position property is used to place the tool tip text position.

Example

```
.tooltip{
    position: relative;
    display:inline-block;
}
.tooltip .tiptext{
        visibility: hidden;
        font-size: 15px;
        width: 120px;
        background-color: grey;
        color: #fff;
        padding: 5px;
        border-radius: 6px;

        position: absolute;
        z-index: 1;
        bottom: 105%;
        left: 50%;
}

.tooltip:hover .tiptext{
    visibility: visible;
}

<article>
    <div class="tooltip">
        This is an example of Tool tip
        <span class="tiptext">example</span>
    </div>
</article>
```

Output



CSS Transition

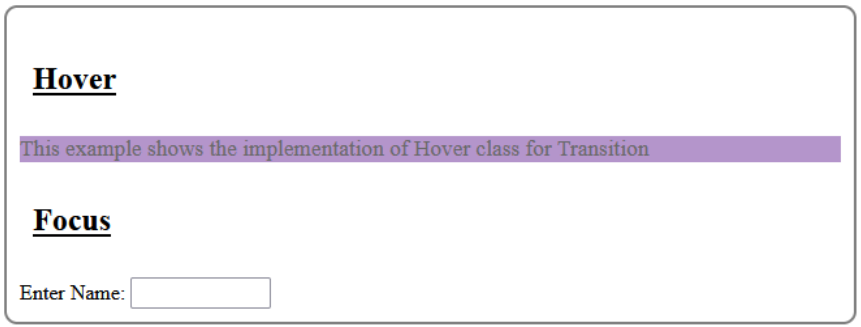
CSS transitions enable the smooth transformation of CSS properties from one state to another. These animations provide a visually appealing and user-friendly way to enhance the interactivity and user experience of web pages. Some of the transition properties are

- **transition-delay**: This defines the length of time to delay the start of a transition.
- **transition-duration**: This defines the length of time to perform a transition.
- **transition-property**: This defines the CSS properties for which the transition is to be applied. Multiple property names can be used separated by comma. The keyword "all" is used to implement transitions on all CSS properties.
- **transition-timing-function**: This defines the function that describes how a transition will proceed for the given duration. It takes the values ease, ease-in, ease-out and linear.
- **transition**: This is a shorthand notation for the transition property.

Example

```
p{
    transition-property: font-size, color, background-color;
    transition-duration: 3s;
    transition-timing-function: linear;
}
p:hover{
    color: white;
    font-size: larger;
    background-color: indigo;
}
input{
    transition: all 1s ease-in-out;
    width: 100px;
    height: 18px;
    font-size: normal;
}
input:focus {
    width: 250px;
    height: 50px;
    font-size: larger;
    background-color: lightgray;
}
<article>
    <h1>Hover</h1>
    <p>
        This example shows the implementation of Hover class for Transition
    </p>
    <h1>Focus</h1>
    <div>
        <label for="studname">Enter Name: </label>
        <input type="text" name="studname">
    </div>
</article>
```

Output



CSS Animation

CSS animations enable the creation of dynamic and visually engaging effects on web pages. Unlike transitions, which handle property changes from one state to another, animations allow you to define keyframes and control the intermediate steps in an animation sequence. It is a two step process

1. First step define individual keyframes and give it a name
2. Second step is referencing the keyframes by name using animation-name property

Defining Keyframes

This is used to specify the values for animating properties at various stages. It is defined using @keyframes or @-webkit-keyframes. The rule is given with (%) or keywords from which specifies the starting point (0%) and to which specifies the ending point (100%).

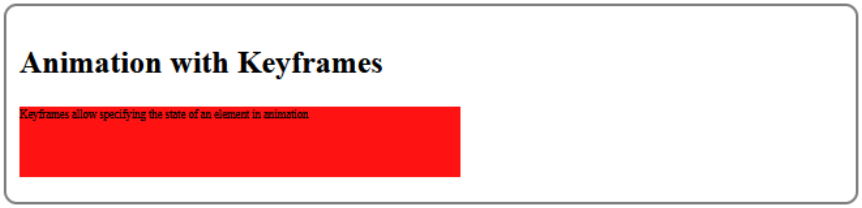
Important Animation Properties

- **animation-name**: The name of the animation defined using @keyframes.
- **animation-duration**: The duration of the animation in seconds
- **animation-delay**: The delay before the animation starts.
- **animation-iteration-count**: The number of times the animation repeats.
- **animation**: This is a shorthand notation.

Example

```
@-webkit-keyframes zoom{
    0% { width: 0px; height: 0px; font-size: 0px;}
    25% { background:RED;}
    100% { width: 100%; height: 100px; font-size: auto;}
}
@keyframes zoom{
    0% { width: 0px; height: 0px; font-size: 0px;}
    25% { background:RED;}
    100% { width: 100%; height: 100px; font-size:auto;}
}
p{
    -webkit-animation: 5s 2s zoom 2;

    animation: 5s 2s zoom 2;
}
<article>
    <h2>Animation with Keyframes</h2>
    <p>
        Keyframes allow specifying the state of an element in animation
    </p>
</article>
```

Output



Pagination

CSS pagination is a technique used to divide long web pages into multiple pages that can be navigated through sequentially. This is often used for articles, blog posts, or content that is too long to fit on a single page.

- **Divide Content:** Use HTML elements like `<div>` or `<section>` to divide your content into logical pages.
- **Hide Pages:** Initially, hide all pages except the first one using CSS.
- **Create Navigation:** Add navigation elements (buttons, links) to allow users to move between pages.
- **Show/Hide Pages:** Use JavaScript or CSS to show or hide pages based on user interaction with the navigation

Example

```
.pag-cont {
        max-width: 600px;
        margin: 0 auto;
        padding: 20px;
 }

 .content-page {
        display: none;
        padding: 20px;
        border: 1px solid #ccc;
        border-radius: 10px;
}

.content-page.active {
        display: block;
}

.pagination {
        text-align: center;
        margin-top: 10px;
}

.pagination a {
        display: inline-block;
        padding: 5px 10px;
        margin-right: 5px;
        text-decoration: none;
        color: #000;
        border: 1px solid #ccc;
        cursor: pointer;
}

.pagination a.active {
        background-color: #333333;
        color: #ffffff;
}

.pagination a:hover:not(.active) {
        background-color: #dddddd;
        border-radius: 5px;
}
```

```
<article>
<div class="pag-cont">
        <div class="content-page " id="page1">
        <h2>Page 1</h2>
        <p>This is the content of Page 1.</p>
        </div>
        <div class="content-page active" id="page2">
        <h2>Page 2</h2>
        <p>This is the content of Page 2.</p>
        </div>
        <div class="content-page " id="page3">
        <h2>Page 3</h2>
        <p>This is the content of Page 3.</p>
        </div>
        <div class="pagination">
        <a href="#" >1</a>
        <a href="#" class="active">2</a>
        <a href="#" >3</a>
        </div>
</div>
</article>
```

Output

Multiple Columns

CSS multiple columns enable the division of content within an element into two or more columns, similar to a newspaper or magazine layout. This feature is particularly valuable for enhancing the readability and visual presentation of text-heavy content. It has the following properties.

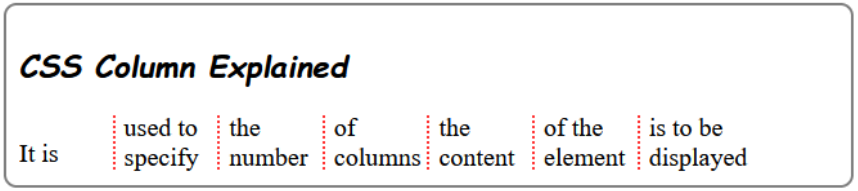
- **column-count**: This specifies the number of columns the content is to be distributed.
- **column-width**: This is used to set the width of the column in px
- **column-fill**: This is used to specify how to fill the columns. It takes the values balance and auto.
- **column-gap**: This is used to specify the gap between columns. It takes the values normal or pixel value.
- **column-rule**: This is used to specify width, style and color of the ruler line between columns.
- **column-span**: This is used to specify the columns an element should span across.

Example

```
section{
    column-width: 60px;
    column-fill: balanced;
    column-gap: normal;
    column-rule: 2px dotted red;
}
section > h2{
    font-style: oblique;
    font-family: cursive;
    column-span: all;
}
```

```
<section>
    <h2>CSS Column Explained</h2>
    <p>
    It is used to specify the number of columns the content of the element is to be displayed
    </p>
</section>
```

Output



User Interface

The CSS user interface properties allow modifying the interface property of an element. The resize property in CSS is used to specify if an element is resizable. It can take the values vertical, horizontal, both or none.

Syntax

```
resize: vertical;
```

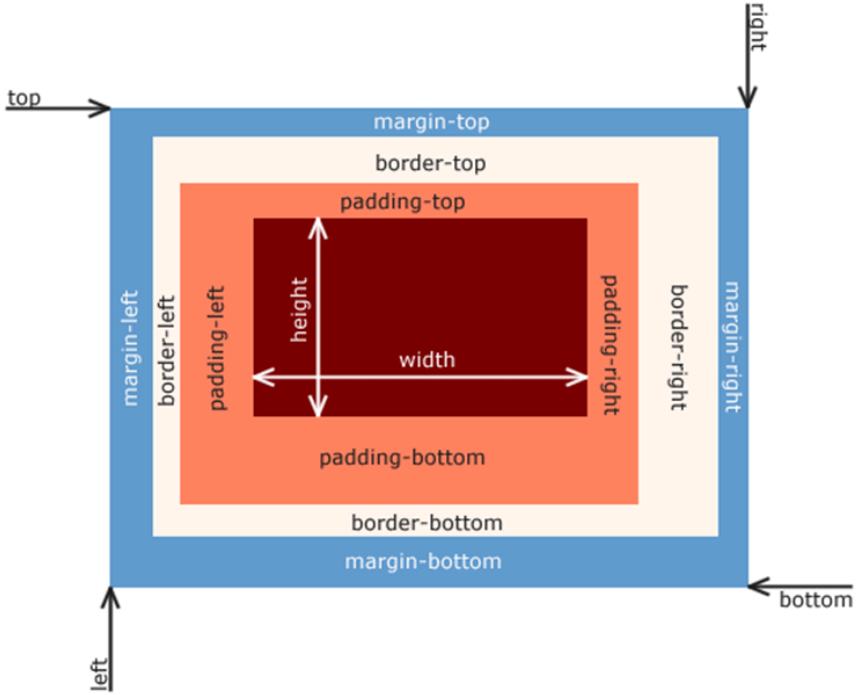
CSS filters

The filter property accepts one or more filter functions, each specifying a different visual effect on images. This is applied using the filter property. Some of the filters are

- blur(px);
- grayscale(%);
- brightness(%);
- opacity(%);
- sepia(%);

Box Model

The CSS box model is a fundamental concept in web design and layout. It describes how elements on a web page are rendered in rectangular boxes, with each box consisting of several layers, including content, padding, border, and margin.



Positioning

CSS (Cascading Style Sheets) positioning is a fundamental concept that allows web developers to control the layout and placement of HTML elements within a web page. Types of CSS positioning

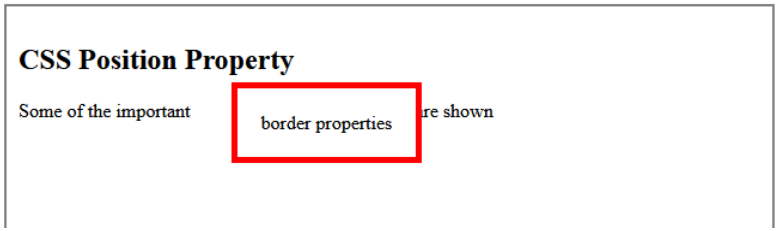
- **static**: Elements are displayed in their default position according to the document flow.
- **relative**: Elements are positioned relative to their default position in the document flow.

- **fixed**: Elements are positioned relative to the viewport (the browser window). They remain fixed in their position even when the user scrolls the page.
- **absolute**: Elements are positioned relative to the nearest ancestor with a non-static position (relative, absolute, or fixed).
- **sticky**: Elements are positioned based on user scroll.

Example

```
span{
    position: relative;
    top: 10px;
    left: 20px;
    border: 5px solid RED;
    margin: 10px;
    padding: 20px;
}
<article>
    <h2>CSS Position Property</h2>
    <p>
        Some of the important <span>border properties</span> are shown
    </p>
</article>
```

Output



Viewport

The viewport is the visible area of a web page that is currently being displayed on a device's screen. It's the rectangular region where your web page is rendered, and it's usually the same size as the device's screen. To control the viewport's behavior, the `viewport` meta tag is added in the HTML document's `<head>` section.

Syntax

<meta name="viewport" content="width=device-width, initial-scale=1.0">

Flex Box

CSS Flexbox, known as the Flexible Box Layout, is a layout model in CSS designed for creating complex and responsive layouts. It enables the distribution of space and alignment of content within a container, even when the size of elements is unknown or dynamic. It is used to specify

- Flex Container
- Flex Items

Flex Container

The element that contains a set of flex items. It is also referred to as the "flex container." The display property with flex is used to implement this. Additional container properties are

- flex-direction: [column, row/-reverse]
- flex-wrap: [wrap/nowrap]
- justify-content: [center, flex-start, flex-end, space-between, space-around]

Flex Items

Individual elements inside the flex container that are aligned and distributed based on the container's properties. It takes the following properties

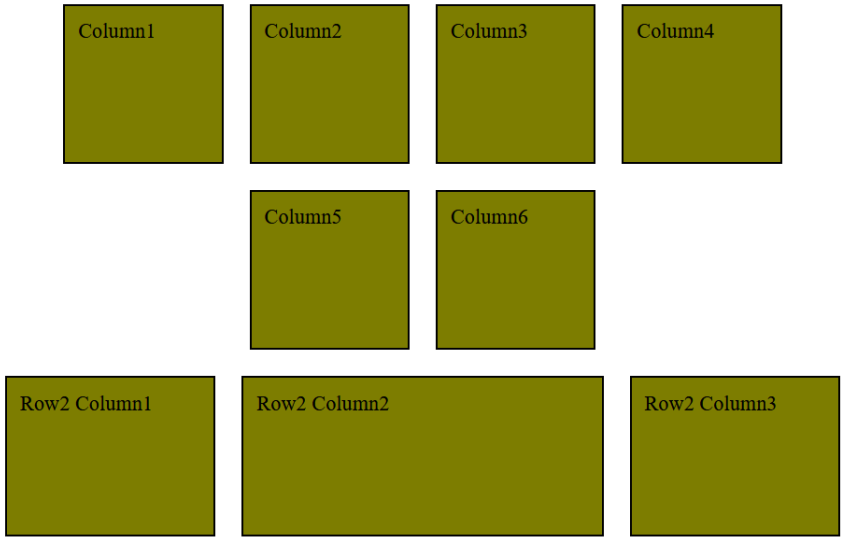
- **flex-grow**: grows an element relative to the remaining element.
- **flex-shrink**: shrinks an element relative to the remaining element.

Example

```
.container{
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: center;

}
.col{
    width: 100px;
    height: 100px;
    border: 2px solid black;
    background-color: olive;
    padding: 10px;
    margin: 10px;
}
.lg-1{
    flex-grow: 1;
}
.lg-4{
    flex-grow: 4;
}
```

```
<div class="container">
    <div class="col">Column1</div>
    <div class="col">Column2</div>
    <div class="col">Column3</div>
    <div class="col">Column4</div>
    <div class="col">Column5</div>
    <div class="col">Column6</div>
</div>
<div class="container">
    <div class="col lg-1">Row2 Column1</div>
    <div class="col lg-4">Row2 Column2</div>
    <div class="col lg-1">Row2 Column3</div>
</div>
```

Output



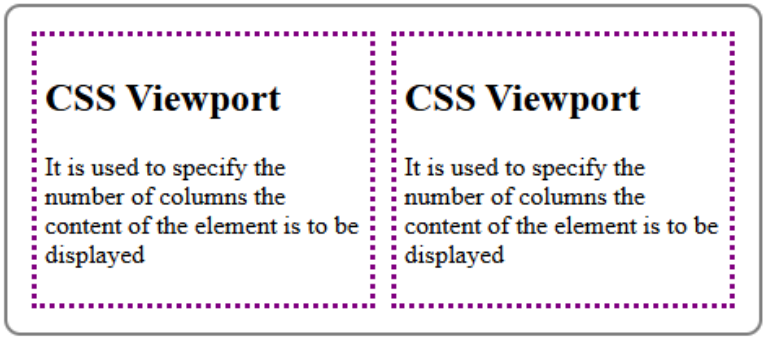
Fluid Width

In CSS, fluid width refers to the ability of an element to dynamically adjust its width based on the available space within its container. This makes it ideal for creating responsive designs that adapt to different screen sizes and devices. To implement this the **width** property of an element is set in **%** instead of **px** value.

Example

```
  div{
    width: 50%;
    padding: 5px;
    margin: 5px;
    border: 3px dotted purple;
  }
<article>
 <div>
  <h2>CSS Viewport</h2>
  <p>
   It is used to specify the number of columns the content of the element is to be displayed
  </p>
 </div>
 <div>
  <h2>CSS Viewport</h2>
  <p>
    It is used to specify the number of columns the content of the element is to be displayed
  </p>
 </div>
</article>
```

Output



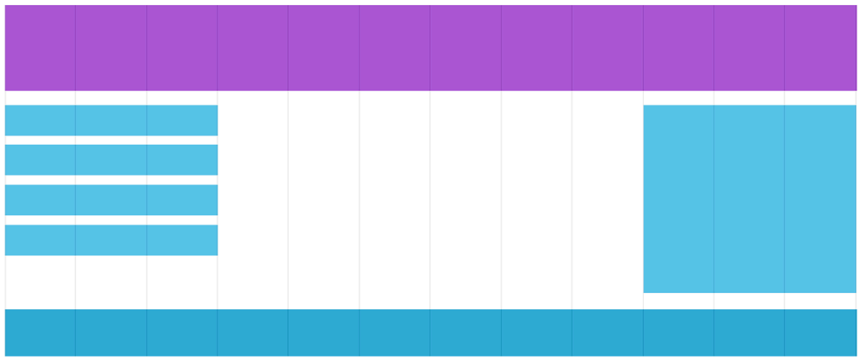
Grids

A web page is built based on grids for better designing using CSS. In this a web page is divided into 12 columns for the whole 100% width.

Fluid width property is used to divide and combine columns in a web page. 100% of the width is divided by 12 columns to get the width of one column as **8.33%**. A web page is designed using the above columns layout and its general form is given below.

Example

```
.col-1{
   width: 8.33%;/*Single column width*/
 }
.col-2{
  width: 16.66%;/*Double column width*/
}
```



Media Queries

Media queries are a powerful tool in CSS that allow you to create responsive web designs that adapt to different screen sizes, orientations, and device capabilities. They enable you to apply specific styles based on the characteristics of the device or the user's viewing environment.

Syntax

```
@media mediaType and (mediaFeature) {
        /* CSS rules to apply when the conditions are met */
}
```

- **mediaType**: This specify the medium in which the webpage is displayed. Example Screen, Print, Speech
- **mediaFeature**: This define the condition that is to be checked for applying the given css. Example width, height, max-width, min-width, orientation

Example

```
/* Media query for screens smaller than 600px */
@media screen and (max-width: 600px) {
  .menu {
     flex-direction: column; /* Change to a vertical layout */
    text-align: center; /* Center-align the items */
  }
 .menu li {
    display: block; /* Each item on a new line */
    margin-bottom: 10px; /* Add some spacing between items */
  }
a{
  width: 100%;
  }
}
```

Output