CSS Text Formatting Properties:

<table>
<tr><th>Property</th><th colspan="2">Description</th></tr>
<tr><td>color</td><td colspan="2">Set the color of the text (use hex-code)</td></tr>
<tr><td>text-align</td><td colspan="2">Set the horizontal alignment of a text. values: left, right, justified or centered</td></tr>
<tr><td>text-decoration</td><td colspan="2">Short hand property for text-decoration-line, text-decoration-color and text-decoration-style</td></tr>
<tr><td>text-transform</td><td colspan="2">Specify uppercase and lowercase letters in a text. values: uppercase, lowercase, capitalize</td></tr>
<tr><td>text-indent</td><td colspan="2">Specify the indentation of the first line of a text (in px)</td></tr>
<tr><td>letter-spacing</td><td colspan="2">Specify gapping between two consecutive characters (in px)</td></tr>
<tr><td>word-spacing</td><td colspan="2">Specify gapping between two consecutive words (in px)</td></tr>
<tr><td>line-height</td><td colspan="2">Specify gapping between consecutive two lines (in %, default 100%)</td></tr>
<tr><td>direction</td><td colspan="2">Specify direction of text. Values: rtl,ltr</td></tr>
<tr><td rowspan="5">white-space</td><td colspan="2">Specifies how white-space inside an element is handled</td></tr>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>wrap</td><td>Sequences of whitespace will collapse into a single whitespace. Text will wrap when necessary. This is default</td></tr>
<tr><td>nowrap</td><td>Sequences of whitespace will collapse into a single whitespace. Text will never wrap to the next line. The text continues on the same line</td></tr>
<tr><td>pre</td><td>Whitespace is preserved by the browser. Text will only wrap on line breaks. Acts like the <pre> tag in HTML</td></tr>
<tr><td></td><td>pre-wrap</td><td>Whitespace is preserved by the browser. Text will wrap when necessary, and on line breaks</td></tr>
</table>

CSS text-shadow:

- The text-shadow property adds shadow to text.
- This property accepts a comma-separated list of shadows to be applied to the text.

In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px):

Example:

```
h1 {
  text-shadow: 2px 2px;
}
```

Adding a color to the shadow:

```
h1 {
  text-shadow: 2px 2px red;
}
```

Adding a blur effect to the shadow:

```
h1 {
  text-shadow: 2px 2px 5px red;
}
```

Example: white text with red shadow:

```
h1 {
  color: white;
  text-shadow: 2px 2px 4px red;
}
```

To add more than one shadow to the text, you can add a comma-separated list of shadows.

Example:

```
h1 {
  text-shadow: 0 0 3px red, 0 0 5px blue;
}
```

CSS Font Properties:

Property	**Description**	
font-family	Set the font-family property. If the name of a font family is more than one word, it must be in quotation marks, like: "Times New Roman".	
font-style	Specify font-style	
	Value	**Description**
	normal	The text is shown normally
	italic	The text is shown in italics
font-size	Sets the size of the text. Default 16px = 1em	
font-weight	Sets how thick or thin characters in text should be displayed	
	Value	**Description**
	normal	Defines normal characters. This is default
	bold	Defines thick characters
font-variant	Sets the font variation	
	Value	**Description**
	normal	The browser displays a normal font. This is default
	small-caps	The browser displays a small-caps font. all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text

CSS list Properties:

Property	**Description**	
list-style-type	Specify type of list item marker some values are as follows	
	Value	**Description**
	disc	This is default value
	square	Use square symbol for list
	circle	Use circle symbol for list
	decimal	Use decimal numbers for list

	lower-alpha	Use lower case symbol for list
	upper-alpha	Use upper case symbol for list
list-style-image	Specify image as list item marker	

Example:

```
<!DOCTYPE html>

<html>

<head>
    <style>
        .a {
            list-style-type: circle;
        }

        .b {
            list-style-type: square;
        }

        .c {
            list-style-type: upper-roman;
        }

        .d {
            list-style-type: lower-alpha;
        }
    </style>
</head>

<body>

    <h2>The list-style-type Property</h2>

    <p>Example of unordered lists:</p>
    <ul class="a">
        <li>Coffee</li>
        <li>Tea</li>
        <li>Coca Cola</li>
    </ul>

    <ul class="b">
        <li>Coffee</li>
        <li>Tea</li>
        <li>Coca Cola</li>
    </ul>
```

```
    <p>Example of ordered lists:</p>
    <ol class="c">
        <li>Coffee</li>
        <li>Tea</li>
        <li>Coca Cola</li>
    </ol>

    <ol class="d">
        <li>Coffee</li>
        <li>Tea</li>
        <li>Coca Cola</li>
    </ol>

</body>

</html>
```

Adding Google fonts and Icons in our webpage:

- Search the google font in google.
- Select the icon tab
- Copy the link for the static/variable icon and paste it inside the <head>
- Insert the style related code inside the webpage.

to increase the size of the font:

set the font-size of the corresponding class inside the <style> tag.

Applying the font:

- select the font tab
- choose the desired font
- click on Get font button
- click on Get Embed code
- follow the instructions

Font Awesome: https://fontawesome.com/

- Create an account and get a kit, add the kit to the <head> tag, for more info refer to the docs.

Dafont: https://www.dafont.com/

Step1: download the the specific font, and unzip.

Step2: copy the ttf/otf file inside the working directory (create a separate folder called **fonts**)

Step3: Inside the <style> tag create a font-face by giving a name and the source of the font.

Example:

```
@font-face {
    font-family: myfont1;
    src: url(./fonts/Komigo3D-Regular.ttf);
}
```

Step4: use the font for any element

```
h1{
    font-family: myfont1;
}
```

Lab assignment: create the following layout using CSS:

Title

Panda Card

John Doe

Computer Science 1
Computer Science 2

Year: 3

☆☆☆☆☆

CSS box-sizing property:

The box-sizing CSS property sets how the total width and height of an element is calculated. The CSS box-sizing property allows us to include the padding and border in an element's total width and height.

Without the CSS box-sizing Property:

By default, the width and height of an element is calculated like this:

width + padding + border = actual width of an element
height + padding + border = actual height of an element

This means: When you set the width/height of an element, the element often appears bigger than you have set (because the element's border and padding are added to the element's specified width/height).

Example:

```
<!DOCTYPE html>
<html lang="en">
```

```html
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

    <style>
        #box1 {

            width: 300px;
            height: 100px;
            border: 1px solid blue;

        }

        #box2 {

            width: 300px;
            height: 100px;
            border: 10px solid red;
            padding: 50px;

        }

    </style>

</head>

<body>

    <div id="box1">
        Lorem ipsum dolor sit amet.
    </div>

    <br><br><br>

    <div id="box2">
        Lorem ipsum dolor sit amet.
    </div>
```

```
</body>

</html>
```

The above problem we can solve using box-sizing property.

The **box-sizing** property allows us to include the padding and border in an element's total width and height.

Values are:

- **content-box:** It is the default value, The content-box value for box-sizing calculates an element's width and height, excluding padding and borders, which are added to the specified dimensions.
- **border-box:** The border-box value for box-sizing calculates an element's width and height, including padding and borders within the specified dimensions, making layout design more straightforward and predictable.

Example:

```
  #box2 {
width: 300px;
height: 100px;
border: 10px solid red;
padding: 50px;
box-sizing: border-box;
}
```

CSS Gradients:

CSS gradients let you display smooth transitions between two or more specified colors.

CSS defines three types of gradients:

- Linear Gradients (goes down/up/left/right/diagonally)
- Radial Gradients (defined by their center)
- Conic Gradients (rotated around a center point)

1. **Linear Gradients:**

    - To create a linear gradient you must define at least two color stops. Color stops are the colors you want to render smooth transitions among. You can also set a starting point and a direction (or an angle) along with the gradient effect.

    Syntax:

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

    Example:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

    <style>
        #box1 {

            width: 500px;
            height: 300px;
            border: 10px solid red;
            box-sizing: border-box;
            background-image: linear-gradient(red, green);
        }
    </style>
</head>
```

```
<body>

    <div id="box1">
        Lorem ipsum dolor sit amet.
    </div>


</body>

</html>
```

Direction:

- **Top to bottom:** this is the default behaviour.
- **Left to right:** `background-image: linear-gradient(to right, red , yellow);`
- **Diagonal:** You can make a gradient diagonally by specifying both the horizontal and vertical starting positions.
  `background-image: linear-gradient(to bottom right, red, yellow);`

Angles:

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

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

Example: `background-image: linear-gradient(180deg, red, yellow);`

Example: Rainbow background

```
background-image: linear-gradient(to right,
red,orange,yellow,green,blue,indigo,violet);
```

Using Transparency:

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

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

Example:

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

Repeating a linear-gradient:

The repeating-linear-gradient() function is used to repeat linear gradients:

Example:

```
background-image: repeating-linear-gradient(red, yellow 10%, green 20%);
```

2. **CSS Radial Gradients:**

- A radial gradient is defined by its center.
- To create a radial gradient you must also define at least two color stops.

Syntax:

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

By default, **shape** is ellipse, **size** is farthest-corner, and **position** is center.

Example:

```
background-image: radial-gradient(red, yellow, green);
```

Differently Spaced Color Stops: radial gradient with differently spaced color stops:

```
background-image: radial-gradient(red 5%, yellow 15%, green 60%);
```

Setting the Shape:

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

Example:

```
background-image: radial-gradient(circle, red, yellow, green);
```

Using of Different Size:

Determines the size of the gradient's ending shape. If omitted it defaults to farthest-corner.

It can take 4 values:

1. **closest-side:** The gradient's ending shape meets the side of the box closest to its center (for circles) or meets both the vertical and horizontal sides closest to the center (for ellipses).
2. **closest-corner:** The gradient's ending shape is sized so that it exactly meets the closest corner of the box from its center.
3. **farthest-side:** Similar to closest-side, except the ending shape is sized to meet the side of the box farthest from its center (or vertical and horizontal sides).
4. **farthest-corner:** The default value, the gradient's ending shape is sized so that it exactly meets the farthest corner of the box from its center.

Example:

```
background-image: radial-gradient(closest-side at 60% 50%, red, yellow, blue);
```

Repeating a radial-gradient:

```
background-image: repeating-radial-gradient(red, yellow 10%, green
15%);
```

3. Conic Gradients:

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

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

Syntax:

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

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

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

Example:

```
background-image: conic-gradient(red, yellow, green);
```

A conic gradient with five colors:

```
background-image: conic-gradient(red, yellow, green, blue, black);
```

A conic gradient with three colors and a degree for each color:

```
background-image: conic-gradient(red 45deg, yellow 90deg, green 210deg);
```

With starting deg to stop deg:

```
background-image: conic-gradient(red 0deg, red 90deg, yellow 90deg, yellow
180deg, green 180deg, green 270deg, blue 270deg);
            /* border-radius: 50%; */
```

Repeating a Conic Gradient:

```
background-image: repeating-conic-gradient(red 10%, yellow 20%);
border-radius: 50%;
```

With starting deg to stop deg repeating conic gradient:

```
 background-image: repeating-conic-gradient(red 0deg 10deg, yellow 10deg
20deg, blue 20deg 30deg);
 border-radius: 50%;
```

CSS Positioning:

The **position** CSS property specifies how an element is positioned in a document. To position an element '**top, bottom, right, left**' properties are used. Make sure to use the **position** property before using the **top, bottom, right & left**. Let us take a look at the positioning method:

1. position: static;

- HTML elements are positioned static by default.
- Static positioned elements are not affected by the top, bottom, left, and right properties.
- An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

Example:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>

  <style>
    .d1 {
      background-color: green;
```

```
        border: 2px solid red;
        width: 800px;
        height: 50px;
        position: static;
        /* not affected by top, right, left, bottom properties */
        /* top: 20px;
        leftt: 40px;  */

      }
    </style>


</head>

<body>

    <h1>Welcome to Chitkara</h1>

          <p>
          An element with position: static; is not positioned in any special way; it is always positioned according to the
          normal flow of the page:
          </p>

    <div class="d1">
      This div element has position: static;
    </div>

</body>

</html>
```

2. position: relative;

The element is positioned according to the normal flow of the document, and then offset *relative to itself* based on the values of **top, right, bottom, and left**. The offset does not affect the position of any other elements; thus, the space given for the element in the page layout is the same as if position were static.

It will move from its relative position(actual position).

Example1:

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        .d1 {
            background-color: green;
            border: 2px solid red;
            width: 300px;
            height: 100px;
            position: relative;
            /* top: 100px;
            left: 100px; */
        }
    </style>
</head>
<body>
```

```
    <h1>Welcome to Chitkara</h1>
    <div class="d1">
        This div element has position: relative;
    </div>
</body>
</html>
```

Example2: effect with relative to other elements

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        div{
            width: 300px;
            height: 100px;
        }
        .d1 {
```

```
            background-color: green;
            border: 2px solid red;
            position: relative;
            /* top: 100px;
            left: 100px; */
        }
        .d2{
            margin-top: 20px;
            background-color: gold;
            border: 2px solid;
        }
    </style>
</head>
<body>
    <h1>Welcome to Chitkara</h1>
    <div class="d1">
        This div element has position: relative;
    </div>
    <div class="d2">
        This div element has position: relative;
    </div>
```

```
</body>

</html>
```

Note: Here we can notice that space created after the moving of first div is not affecting the other element position(2nd div).

3. <u>position: absolute;</u>

The element is removed from the normal document flow, and no space is created for the element in the page layout. The element is positioned relative to its closest **positioned** ancestor (if any) or to the initial containing block. Its final position is determined by the values of top, right, bottom, and left.

By default it will move the element with respect to its viewport, but if the parent element of this element is positioned element, then it will work **relative** to its parent **positioned** element.

Example1: **box with absolute value**

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Document</title>

    <style>

        .box {

            width: 100px;
```

```
            height: 100px;

            border: 2px solid red;

            background-color: aqua;

            /* this box will come out from the normal document flow and the next
element will accquire its space */

            position: absolute;

            top: 40px;

            left: 40px;

             /* top: 0px;

            right: 0px; */



        }

    </style>

</head>

<body>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Accusantium qui
dolor unde? Odio natus explicabo placea nam exercitationem iusto corrupti
cupiditate consequuntur optio quasi iure dolor, sed cumque modi
necessitatibus?</p>



    <div class="box"></div>
```

```html
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Accusantium qui
dolor unde? Odio natus explicabo placeat nam exercitationem iusto corrupti
cupiditate consequuntur optio quasi iure dolor, sed cumque modi
necessitatibus?</p>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Accusantium qui
dolor unde? Odio natus explicabo placeat nam exercitationem iusto corrupti
cupiditate consequuntur optio quasi iure dolor, sed cumque modi
necessitatibus?</p>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Accusantium qui
dolor unde? Odio natus explicabo placeat nam exercitationem iusto corrupti
cupiditate consequuntur optio quasi iure dolor, sed cumque modi
necessitatibus?</p>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Accusantium qui
dolor unde? Odio natus explicabo placeat nam exercitationem iusto corrupti
cupiditate consequuntur optio quasi iure dolor, sed cumque modi
necessitatibus?</p>

</body>

</html>
```

Example2: absolute with respect to its parent(if parent element is the **positioned** element)

Positioned element: any element whose position is other than static, is called the **positioned** element.

In this case, the parent element acts as the reference point for positioning the child element. So if you move the parent element, the child element will move relative to it.

```html
<!DOCTYPE html>

<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        .container {
            width: 400px;
            height: 400px;
            background-color: beige;
            border: 2px solid black;
            /* position: relative; */
        }
        .box {
            width: 100px;
            height: 100px;
            border: 2px solid red;
            background-color: aqua;
            /* position: absolute;
            top: 40px;
            left: 40px; */
        }
```

```html
    </style>

</head>

<body>

<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Accusantium qui dolor unde? Odio natus explicabo placeat nam exercitationem iusto corrupti cupiditate consequuntur optio quasi iure dolor, sed cumque modi necessitatibus?</p>

    <div class="container">

        <div class="box"></div>

    </div>

<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Accusantium qui dolor unde? Odio natus explicabo placeat nam exercitationem iusto corrupti cupiditate consequuntur optio quasi iure dolor, sed cumque modi necessitatibus?</p>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Accusantium qui dolor unde? Odio natus explicabo placeat nam exercitationem iusto corrupti cupiditate consequuntur optio quasi iure dolor, sed cumque modi necessitatibus?</p>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Accusantium qui dolor unde? Odio natus explicabo placeat nam exercitationem iusto corrupti cupiditate consequuntur optio quasi iure dolor, sed cumque modi necessitatibus?</p>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Accusantium qui dolor unde? Odio natus explicabo placeat nam exercitationem iusto corrupti cupiditate consequuntur optio quasi iure dolor, sed cumque modi necessitatibus?</p>

</body>

</html>
```

Example: Using positioning putting a div in the center of the webpage.

```
<!DOCTYPE html>

<html lang="en">




<head>

    <meta charset="UTF-8">

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

    <title>Document</title>

    <style>

        body {

            margin: 0;

            padding: 0;

            /* Allows absolute positioning of children */

            position: relative;

            /* Ensure the body takes the full viewport height */

            height: 100vh;



        }

        div {
```

```
            width: 200px;
            height: 200px;
            border: 5px solid red;
            position: absolute;
            top: 0;
            left: 0;
            right: 0;
            bottom: 0;
            margin: auto;
            /* Automatically centers the content */
        }
    </style>
</head>
<body>
    <div></div>
</body>
</html>
```

Example: Creating 3 divs of 100*100px and putting them in the diagonal form, e.g 1st div in top right, 2nd div in the center and 3rd div in the bottom left.

```
<!DOCTYPE html>
```

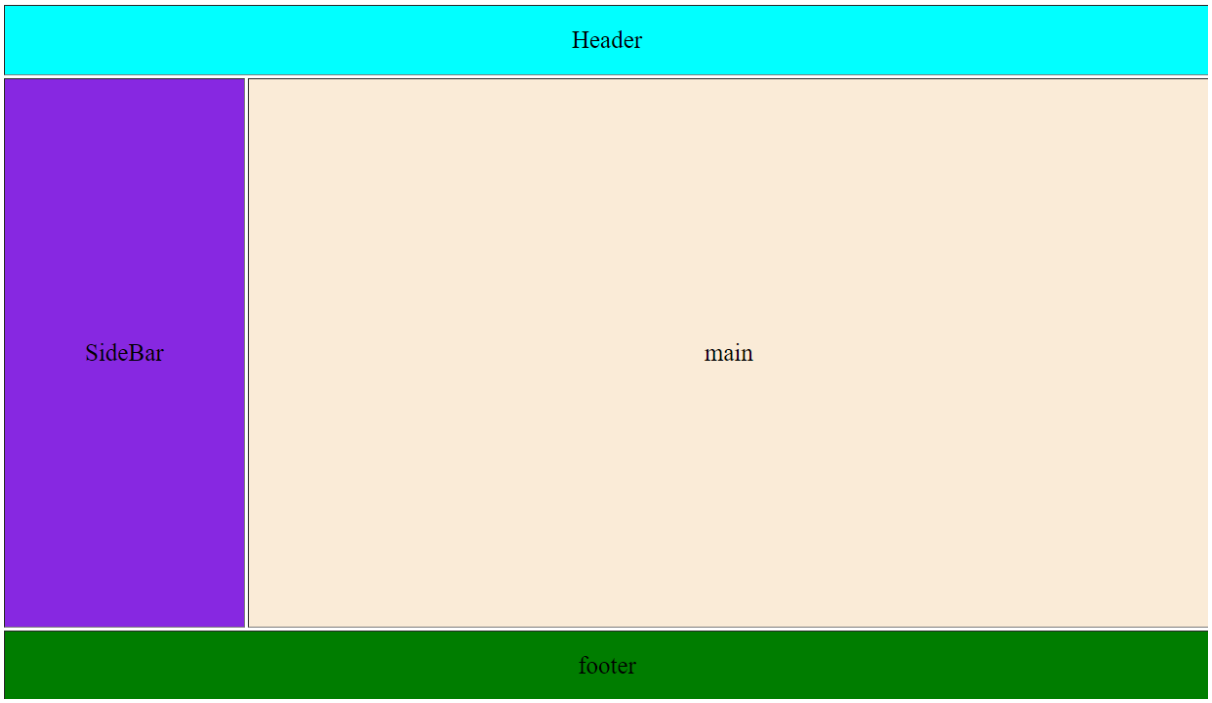
```
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Diagonal Divs Example</title>
    <style>
        body {
            margin: 0;
            padding: 0;
            background-color: gray;
            /* Allows absolute positioning of children */
            position: relative;
            /* Ensure the body takes the full viewport height */
            height: 100vh;
        }
        .box {
            width: 100px;
            height: 100px;
            background-color: gold;
            border: 2px solid red;
```

```
    color: white;
    text-align: center;
    position: absolute;
}
/* 1st div: Top-right */
.box1 {
    top: 0;
    right: 0;
}
/* 2nd div: Center */
.box2 {
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    margin: auto;
}
/* 3rd div: Bottom-left */
.box3 {
    bottom: 0;
    left: 0;
```

```
        }
    </style>
</head>
<body>
    <div class="box box1">1st Div</div>
    <div class="box box2">2nd Div</div>
    <div class="box box3">3rd Div</div>
</body>
</html>
```

Lab assignment1: create the following layout using CSS positioning



Solution:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple Layout</title>
    <style>
        body {
            margin: 0;
            padding: 0;
            height: 100vh; /* Ensure the body takes full viewport height */
            position: relative; /* Allows absolute positioning of children */
            text-align: center;
        }
        .header {
            position: absolute;
            top: 0;
            left: 0;
            width: 100%;
            height: 10vh;
            background-color: wheat;
```

```
}

.sidebar {

    position: absolute;

    top: 10vh; /* Below the header */

    left: 0;

    width: 25%;

    height: 80vh;

    background-color: cyan;

}

.main {

    position: absolute;

    top: 10vh; /* Below the header */

    left: 25%; /* Next to the sidebar */

    width: 75%;

    height: 80vh;

    background-color: lightgreen;

}

.footer {

    position: absolute;

    bottom: 0;

    left: 0;
```

```
            width: 100%;
            height: 10vh;
            background-color: gold;
        }
    </style>
</head>
<body>
    <div class="header">Header</div>
    <div class="sidebar">Sidebar</div>
    <div class="main">Main Content</div>
    <div class="footer">Footer</div>
</body>
</html>
```

Task: convert the non-semantic tag with semantic tags in the above example.

Lab Assignment2: creating the above layout using <table> tag:

Solution: Using simple table

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

    <style>
        body {
            margin: 0px;
            padding: 0px;
        }

        table {
            width: 100%;
            height: 100vh;

        }

        #header {
            height: 10vh;
            background-color: aqua;

        }

        #footer {
            height: 10vh;
            background-color: green;

        }

        #side {
            width: 20%;
            background-color: blueviolet;

        }
```

```
        td {
            text-align: center;
        }

        #main {
            width: 80%;
            background-color: antiquewhite;

        }
    </style>

</head>

<body>

    <table border="2px">

        <tr id="header">
            <td colspan="2">Header</td>
        </tr>

        <tr id="center">
            <td id="side">SideBar</td>
            <td id="main">main</td>

        </tr>
        <tr id="footer">
            <td colspan="2">footer</td>

        </tr>

    </table>

</body>

</html>
```

4. position: 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. The top, right, bottom, and left properties are used to position the element.

Example: We can see the effect by adding some more content to the <p> tag and by scrolling the window.

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Document</title>

    <style>

        .box {

            width: 100px;

            height: 100px;

            border: 2px solid red;

            background-color: rgb(11, 234, 234,0.1);

            /* position: fixed;

            top: 50px;

            left: 25px; */
```

```
        }
    </style>
</head>
<body>
<p>Lorem ipsum, dolor sit amet consectetur adipisicing elit. Cupiditate eum, illum autem quisquam quo quis? Amet quasi expedita numquam nulla. Labore corrupti ratione amet. Asperiores vero dignissimos minus commodi placeat.</p>
<div class="box"></div>
</body>
</html>
```

5. position: sticky;

- An element with position: sticky; is positioned based on the user's scroll position.
- A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

Example: Add some content(1orem1000) to the last <p> tag and scroll the page to see the effect:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
```

```
    <style>
        .box {
            width: 100px;
            height: 100px;
            border: 2px solid red;
            background-color: aqua;
            position: sticky;
            top: 20px;
            /* after adding the Lorem1000 content to the last <p>,then while
scrolling the page after reaching the 20px from the top the <div> will stick */
        }
    </style>
</head>

<body>

    <p>
       Lorem ipsum dolor sit amet consectetur adipisicing elit. Ab nesciunt
      laboriosam, molestiae eius, nostrum reprehenderit architecto saepe maiores
      consectetur sint vero porro ratione odit rem repellat nam placeat iusto
      voluptatem.

      Lorem, ipsum dolor sit amet consectetur adipisicing elit. Temporibus
      debitis harum rem tempora voluptate itaque beatae nobis magni minima,
      commodi aspernatur fuga molestiae quae quo, eaque similique quaerat
      laudantium ab.

      Lorem ipsum dolor sit amet, consectetur adipisicing elit. Cumque, et velit
      accusantium ipsam deserunt rerum voluptates illum iure autem unde
      temporibus repellendus? Non impedit nobis natus vero, neque accusantium
      dicta.

    </p>

    <div class="box"></div>

    <p>
```

```
        Lorem ipsum, dolor sit amet consectetur adipisicing elit. Cupiditate eum,
        illum autem quisquam quo quis? Ametquasi expedita numquam nulla. Labore
        corrupti ratione amet. Asperiores vero dignissimos minus commodi
        placeat.

        Lorem1000

        </p>

</body>

</html>
```