**CSS Layout Techniques**

**Class Outline**

**Objective:**

By the end of this class, students will understand various CSS layout techniques, including the use of display properties, positioning, and Flexbox. They will be able to create responsive and well-structured web layouts.

1. **Display Property**

**Understanding Display Property**

The display property specifies the display behavior of an element.

Common Display Values

**block**: Element takes up the full width available, starting on a new line.

div {

display: block;

}

**inline**: Element takes up only as much width as necessary and does not start on a new line.

span {

display: inline;

}

**inline-block**: Element is formatted as an inline element but can have a width and height.

div {

display: inline-block;

}

**none**: Element is not displayed and does not take up any space.

div {

display: none;

}

**2. CSS Positioning**

**Static Positioning**

Default positioning. Elements are positioned according to the normal flow of the document.

div {

position: static;

}

**Relative Positioning**

Positioned relative to its normal position.

div {

position: relative;

top: 10px;

left: 20px;

}

**Absolute Positioning**

Positioned relative to the nearest positioned ancestor or the initial containing block.

div {

position: absolute;

top: 50px;

left: 100px;

}

**Fixed Positioning**

Positioned relative to the viewport, remaining in the same position even when the page is scrolled.

div {

position: fixed;

bottom: 10px;

right: 10px;

}

**Sticky Positioning**

Switches between relative and fixed positioning depending on the scroll position.

div {

position: sticky;

top: 0;

}

1. **Practical Exercise 1: Positioning Elements**

**Objective**: Use different CSS positioning properties to arrange elements on a web page.

**Instructions**:

*Open your text editor and create a new HTML file named positioning.html.*

*Write the following HTML code:*

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Positioning Example</title>

<link rel="stylesheet" href="positioning.css">

</head>

<body>

<div class="static">Static Position</div>

<div class="relative">Relative Position</div>

<div class="absolute">Absolute Position</div>

<div class="fixed">Fixed Position</div>

<div class="sticky">Sticky Position</div>

</body>

</html>

*Create an external CSS file named positioning.css with the following content:*

.static {

position: static;

background-color: lightblue;

padding: 10px;

}

.relative {

position: relative;

top: 20px;

left: 20px;

background-color: lightgreen;

padding: 10px;

}

.absolute {

position: absolute;

top: 50px;

left: 50px;

background-color: lightcoral;

padding: 10px;

}

.fixed {

position: fixed;

bottom: 10px;

right: 10px;

background-color: lightgoldenrodyellow;

padding: 10px;

}

.sticky {

position: -webkit-sticky; /\* For Safari \*/

position: sticky;

top: 0;

background-color: lightpink;

padding: 10px;

}

**4. Introduction to Flexbox**

**What is Flexbox?**

Flexbox is a layout model that allows responsive elements within a container to be automatically arranged based on available space.

**Basic Flexbox Concepts**

Flex Container: The parent element where display: flex; is applied.

Flex Items: The child elements within the flex container.

**Setting Up Flexbox**

Example:

.container {

display: flex;

}

**Flex Direction**

Defines the direction in which the flex items are placed in the flex container.

row (default), row-reverse, column, column-reverse.

.container {

flex-direction: row;

}

**Justify Content**

Aligns flex items along the main axis.

flex-start, flex-end, center, space-between, space-around.

.container {

justify-content: center;

}

**Align Items**

Aligns flex items along the cross axis.

flex-start, flex-end, center, baseline, stretch.

.container {

align-items: center;

}

**Align Self**

Aligns individual flex items along the cross axis.

auto, flex-start, flex-end, center, baseline, stretch.

.item {

align-self: flex-end;

}

**5. Practical Exercise 2: Creating a Flexbox Layout**

**Objective**: Use Flexbox to create a responsive layout.

**Instructions**:

*Open your text editor and create a new HTML file named flexbox.html.*

*Write the following HTML code:*

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Flexbox Layout Example</title>

<link rel="stylesheet" href="flexbox.css">

</head>

<body>

<div class="container">

<div class="item">Item 1</div>

<div class="item">Item 2</div>

<div class="item">Item 3</div>

<div class="item">Item 4</div>

</div>

</body>

</html>

*Create an external CSS file named flexbox.css with the following content:*

.container {

display: flex;

flex-direction: row;

justify-content: space-around;

align-items: center;

height: 100vh;

background-color: #f0f0f0;

}

.item {

background-color: lightcoral;

padding: 20px;

margin: 10px;

text-align: center;

flex-grow: 1;

}