Day-3

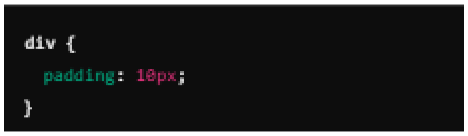
Task-1 : CSS Box Model and Layout

The **CSS Box Model** and layout concepts are fundamental to controlling the spacing, positioning, and alignment of HTML elements on a webpage. These tools are essential for creating visually appealing and organized designs.

CSS Box Model

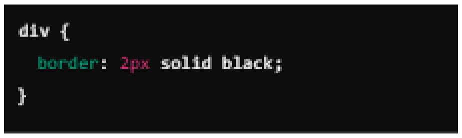
The CSS Box Model describes the structure of a box that wraps around HTML elements. It consists of the following components:

1. **Content**:
   - The main content of the element (e.g., text or images).
2. **Padding**:
   - Space between the content and the element's border.
   - Transparent and expands the clickable area of an element.



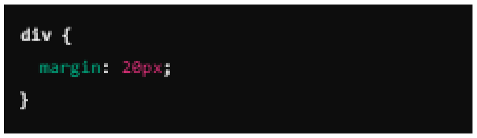
```
div {
  padding: 10px;
}
```

3. **Border**:
   - A boundary that wraps around the padding and content.
   - Can be styled with width, color, and type (e.g., solid, dashed).



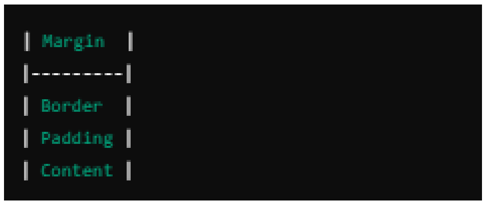
```
div {
  border: 2px solid black;
}
```

4. **Margin**:
   - Space outside the element's border.
   - Used to create distance between elements.



```
div {
  margin: 20px;
}
```

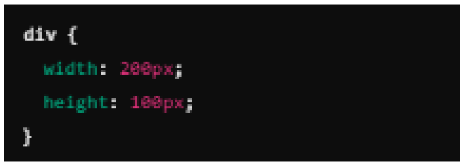
Visualization:



```
| Margin  |
|---------|
| Border  |
| Padding |
| Content |
```

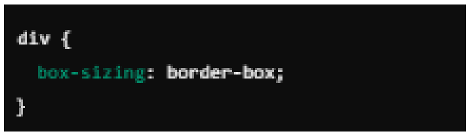
Box Model Properties

- **Width and Height**: Define the size of the content area.



```
div {
  width: 200px;
  height: 100px;
}
```

- **Box-Sizing**:
  - Determines how the total size of an element is calculated.
  - Options:
    - `content-box`: Width and height include only the content.
    - `border-box`: Includes padding and border in width and height.



```
div {
  box-sizing: border-box;
}
```

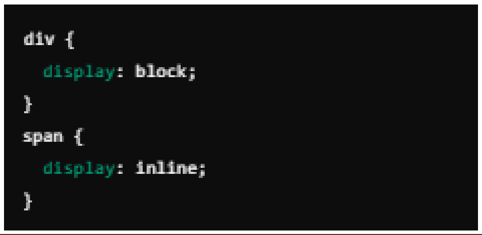
CSS Layout

Layouts are used to arrange elements on a page. CSS provides several techniques for layout design:

1. Normal Flow

Elements are displayed in the order they appear in the HTML:

- Block elements (e.g., `div`, `p`) stack vertically.
- Inline elements (e.g., `span`, `a`) flow horizontally.

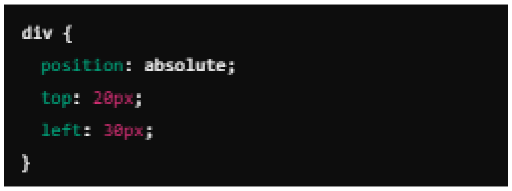


```
div {
  display: block;
}
span {
  display: inline;
}
```

2. Positioning

Controls the placement of elements on the page:

- **Static** (default): Elements flow naturally.
- **Relative**: Positions relative to its normal position.
- **Absolute**: Positioned relative to its nearest positioned ancestor.
- **Fixed**: Positioned relative to the viewport.
- **Sticky**: Toggles between relative and fixed based on scroll.



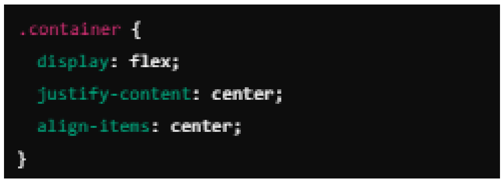
```
div {
  position: absolute;
  top: 20px;
  left: 30px;
}
```

3. Flexbox

A one-dimensional layout system for aligning and distributing space.

Key Properties:

- `display: flex`: Activates flexbox for the container.
- `justify-content`: Aligns items horizontally.
- `align-items`: Aligns items vertically.



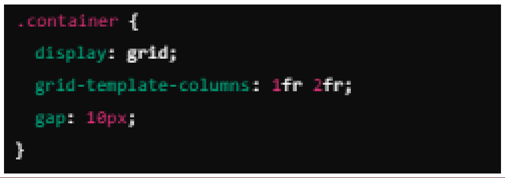
```
.container {
  display: flex;
  justify-content: center;
  align-items: center;
}
```

4. Grid

A two-dimensional layout system for complex designs.

Key Properties:

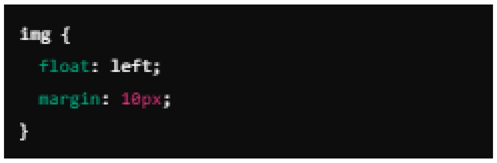
- `display: grid`: Activates grid layout.
- `grid-template-columns`: Defines columns.
- `grid-template-rows`: Defines rows.



```
.container {
  display: grid;
  grid-template-columns: 1fr 2fr;
  gap: 10px;
}
```

5. Float

Aligns elements to the left or right, allowing content to wrap around them.



```
img {
  float: left;
  margin: 10px;
}
```