Css Element

This document explains the following CSS Element:

1. **Flexbox**
2. **Grid**
3. **Overflow**
4. **Float**
5. **Inline-Block**

---

1. Flexbox

Flexbox is a **one-dimensional layout model** that allows you to arrange elements in a row or a column. It is ideal for creating flexible and responsive layouts.

Key Properties:

Container Properties:

- **`display: flex;`**: Defines a flex container.
- **`flex-direction`**: Sets the direction of the flex items.
    - `row` (default): Items are placed in a row.
    - `column`: Items are placed in a column.
- **`justify-content`**: Aligns items horizontally.
    - `flex-start`: Items align to the start.
    - `flex-end`: Items align to the end.
    - `center`: Items are centered.
    - `space-between`: Items are evenly distributed.
    - `space-around`: Items are evenly spaced with equal space around them.
- **`align-items`**: Aligns items vertically.

  - `stretch` (default): Items stretch to fill the container.
  - `flex-start`: Items align to the top.
  - `flex-end`: Items align to the bottom.
  - `center`: Items are centered vertically.
- **`flex-wrap`**: Controls whether items wrap to the next line.
  - `nowrap` (default): All items stay in one line.
  - `wrap`: Items wrap to the next line if needed.

Item Properties:

- **`flex-grow`**: Defines how much an item can grow relative to others.
- **`flex-shrink`**: Defines how much an item can shrink relative to others.
- **`flex-basis`**: Sets the initial size of an item before remaining space is distributed.

---

2. Grid

CSS Grid is a **two-dimensional layout system** that allows you to create complex layouts with rows and columns.

Key Properties:

Container Properties:

- **`display: grid;`**: Defines a grid container.
- **`grid-template-columns`**: Defines the number and size of columns.
  - Example: `grid-template-columns: 100px 200px auto;` (three columns).
- **`grid-template-rows`**: Defines the number and size of rows.
  - Example: `grid-template-rows: 50px 100px;` (two rows).
- **`gap`**: Sets the spacing between grid items.
  - Example: `gap: 10px;`.
- **`justify-items`**: Aligns items horizontally within their cells.

- `align-items`: Aligns items vertically within their cells.

Item Properties:

- `grid-column`: Specifies the column placement of an item.
    - Example: `grid-column: 1 / 3;` (spans from column 1 to column 3).
- `grid-row`: Specifies the row placement of an item.
    - Example: `grid-row: 2 / 4;` (spans from row 2 to row 4).
- `justify-self`: Aligns an item horizontally within its cell.
- `align-self`: Aligns an item vertically within its cell.

---

3. Overflow

The `overflow` property controls what happens when content overflows its container.

Key Values:

- `visible` (default): Content is not clipped and may overflow.
- `hidden`: Content is clipped and hidden.
- `scroll`: Adds scrollbars to view overflowed content.
- `auto`: Adds scrollbars only if necessary.

---

4. Float

The `float` property is used to position elements to the left or right, allowing text and other elements to wrap around it.

Key Values:

- `left`: Floats the element to the left.
- `right`: Floats the element to the right.
- `none` (default): The element does not float.

---

5. Inline-Block

The `display: inline-block;` property allows an element to behave like an inline element while retaining block-level properties.

Key Features:

- Elements can sit next to each other (like inline elements).
- Elements can have width, height, padding, and margin (like block elements).