<!-- ! FlexBox -->

Definition:
- Flexbox is a CSS layout model that allows you to design a flexible and responsive layout structure without using positioning.

Key Concepts:

- **Flex Container**: The parent element where `display: flex;` is applied.
- **Flex Items**: The direct children of a flex container.

Flex Container Properties:

1. **`display: flex;`**:
   - Defines a flex container, enabling flex context for all its direct children.

     - Example: `display: flex;`

2. **`flex-direction`**:

   - Specifies the direction of the flex items in the flex container.
   - Values:
     - `row`: Default, items arranged horizontally (left to right).
     - `row-reverse`: Items arranged horizontally (right to left).
     - `column`: Items arranged vertically (top to bottom).
     - `column-reverse`: Items arranged vertically (bottom to top).

         - Example: `flex-direction: row;`.

3. **`flex-wrap`**:

   - Controls whether flex items should wrap onto multiple lines.
   - Values:
     - `nowrap`: Default, all flex items will be on one line.
     - `wrap`: Flex items will wrap onto multiple lines, from top to bottom.
     - `wrap-reverse`: Flex items will wrap onto multiple lines, from bottom to top.

       - Example: `flex-wrap: wrap;`.

4. **`justify-content`**:

   - Aligns flex items along the main axis (horizontally if `flex-direction: row;`).
   - Values:
     - `flex-start`: Items align to the start of the container.
     - `flex-end`: Items align to the end of the container.
     - `center`: Items align at the center.
     - `space-between`: Items are evenly distributed, with the first item at the start and last at the end.
     - `space-around`: Items are evenly distributed with equal space around them.
     - `space-evenly`: Items are distributed with equal space between them.

   - Example: `justify-content: space-between;`.

5. **`align-items`**:
   - Aligns flex items along the cross axis (vertically if `flex-direction: row;`).
   - Values:
     - `stretch`: Default, items stretch to fill the container.
     - `flex-start`: Items align to the start of the cross axis.
     - `flex-end`: Items align to the end of the cross axis.
     - `center`: Items align at the center of the cross axis.

       - Example: `align-items: center;`.

6. **`align-content`**:

   - Aligns flex lines when there's extra space in the cross axis.
   - Values (similar to `justify-content` but applied to multiple lines):
     - `flex-start`, `flex-end`, `center`, `space-between`, `space-around`, `stretch`.

         - Example: `align-content: space-around;`.

Flex Item Properties:

1. **`order`**:

   - Controls the order of the flex items within the container.
   - Default is `0`, higher numbers appear later.
       - Example: `order: 2;`.

2. **`flex-grow`**:

   - Defines the ability of a flex item to grow if necessary.
   - Default is `0`, meaning items won't grow unless specified.
     - Example: `flex-grow: 1;` (item will take up remaining space).

3. **`flex-shrink`**:

   - Defines the ability of a flex item to shrink if necessary.
   - Default is `1`, meaning items will shrink to avoid overflow.
       - Example: `flex-shrink: 0;` (item won't shrink).

4. **`flex-basis`**:

   - Defines the default size of an element before the remaining space is distributed.
   - Can be set in length units (px, em, etc.) or `auto`.
       - Example: `flex-basis: 200px;`.

5. **`align-self`**:

   - Allows the default alignment (set by `align-items`) to be overridden for individual flex items.
   - Values: `auto`, `flex-start`, `flex-end`, `center`, `baseline`, `stretch`.
   - Example: `align-self: flex-end;`.