CSS Flexbox

CSS Flexbox (Flexible Box Layout) is a layout model that provides an efficient way to align and distribute space among items in a container, even when their size is unknown or dynamic. It helps design complex layouts with ease, especially when responsiveness is needed.

Key Concepts of Flexbox:

1. **Flex Container**: The parent element that holds flex items. You make an element a flex container by setting `display: flex` on it.

```
.container {
    display: flex;
}
```

2. **Flex Items**: The direct children of a flex container. These items will be laid out according to the flexbox model.

Main Properties:

On the **Flex Container**:

1. **`flex-direction`**: Defines the direction in which flex items are placed in the flex container.
   - `row` (default): Left to right.
   - `column`: Top to bottom.
   - `row-reverse`: Right to left.
   - `column-reverse`: Bottom to top.

```
.container {
    flex-direction: row;
}
```

2. **`justify-content`**: Aligns flex items along the main axis (horizontal in a `row`, vertical in a `column`).
   - `flex-start` (default), `flex-end`, `center`, `space-between`, `space-around`, `space-evenly`.

```
.container {
    justify-content: center;
```

```
}
```

3. **`align-items`**: Aligns flex items along the cross axis (vertical in a `row`, horizontal in a `column`).
    - `stretch` (default), `flex-start`, `flex-end`, `center`, `baseline`.

```
.container {
    align-items: center;
}
```

4. **`flex-wrap`**: Controls whether flex items should wrap onto multiple lines.
    - `nowrap` (default), `wrap`, `wrap-reverse`.

```
.container {
    flex-wrap: wrap;
}
```

5. **`align-content`**: Aligns rows of flex items when there's extra space in the cross axis.
    - `flex-start`, `flex-end`, `center`, `space-between`, `space-around`, `stretch`.

```
.container {
    align-content: space-around;
}
```

On the **Flex Items**:

1. **`flex-grow`**: Defines how much a flex item should grow relative to the other flex items. A value of `1` means it can grow, and `0` means it won't grow.

```
.item {
    flex-grow: 1;
}
```

2. **`flex-shrink`**: Defines how much a flex item should shrink relative to the other flex items when space is lacking. A value of `1` means it can shrink, and `0` means it won't shrink.

```
.item {
    flex-shrink: 1;
}
```

3. **`flex-basis`** : Specifies the initial size of the flex item before the remaining space is distributed. It can be a specific size (e.g., `200px`) or `auto`.

```
.item {
    flex-basis: 100px;
}
```

4. **`align-self`** : Allows a flex item to override the `align-items` value from the flex container.
    - `auto` (default), `flex-start`, `flex-end`, `center`, `baseline`, `stretch`.

```
.item {
    align-self: center;
}
```

Flexbox Example:

```
<div class="container">
    <div class="item">1</div>
    <div class="item">2</div>
    <div class="item">3</div>
</div>
```

```
.container {
    display: flex;
    justify-content: space-around;
    align-items: center;
    height: 200px;
    border: 1px solid black;
}

.item {
    background-color: lightblue;
    padding: 20px;
    border: 1px solid gray;
}
```

This example creates a flexible container with 3 items spaced evenly using `justify-content: space-around`, and each item is vertically centered with `align-items: center`.

Why Use Flexbox?

- **Responsive Layouts**: Flexbox makes it easy to create layouts that adjust and resize based on the screen size.
- **Centering**: It's a simple way to center items both horizontally and vertically.
- **Dynamic Resizing**: Flexbox allows items to grow, shrink, and distribute space according to available space.