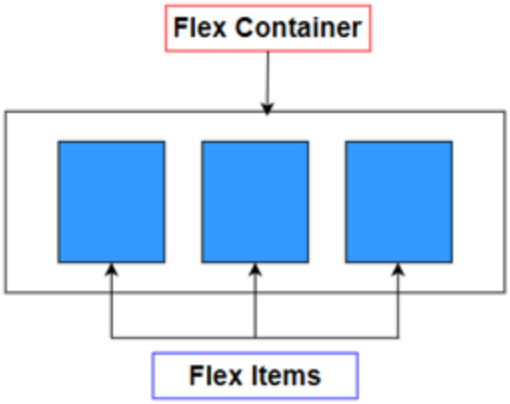
CSS Flex Box

1. **Introduction of CSS Flexbox:**
    a. CSS Flexbox is used to make flexible layout as well as responsive layout without using float and positioning.
    b. It is a 1 dimensional layout because it works primarily in one dimension (row or column).

2. **Features of Flexbox:**
    a. High Flexibility
    b. Arrangement & alignment of items
    c. Proper spacing
    d. Order & sequencing of items

3. **Before the Flexbox Model:**
    a. Before the flexbox we are using blocks in webpage to make sections, Inline elements for text and some tables with positioned elements to fix the position of elements.
    b. *Block* ⇒ Sections in webpage
    c. *Inline* ⇒ Text
    d. *Table* ⇒ Two-dimensional table data
    e. *Position* ⇒ To provide position to an element

4. **Flexbox Components:**
    a. Flexbox components is divided into 2 sections:
        i. *Flex Container* ⇒ The flex container becomes flexible by setting the display property to flex.
        ii. *Flex Items* ⇒ The items present inside the flex container.

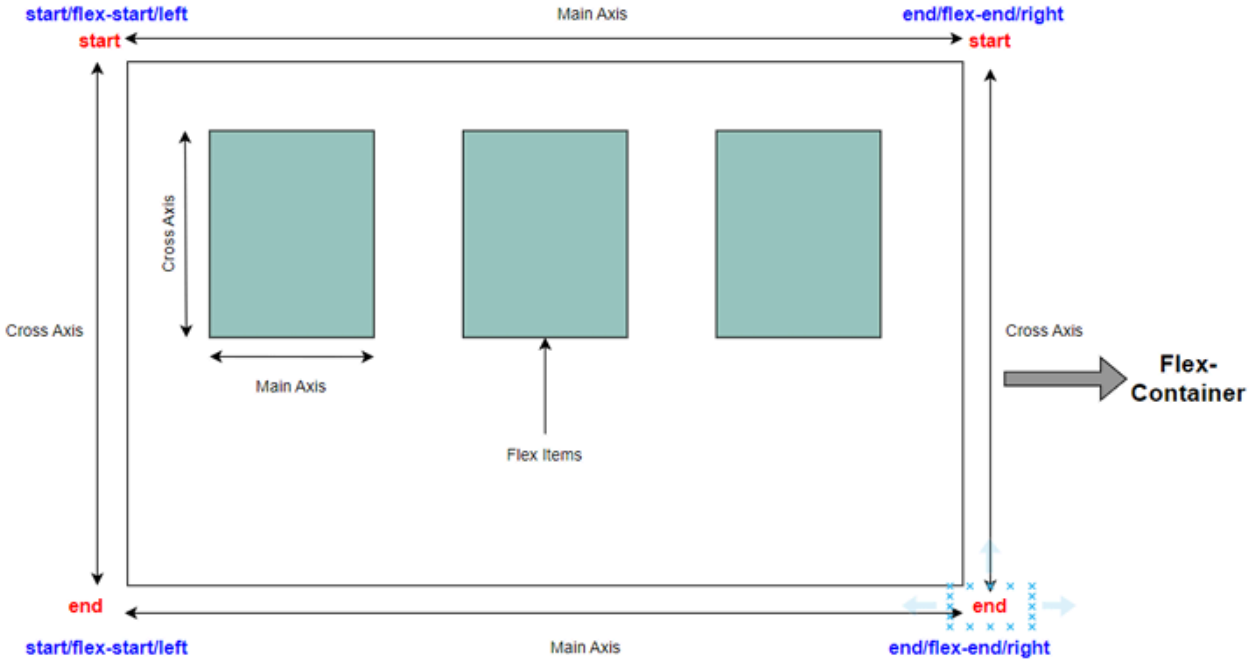


5. Flexbox Axes:

a. Flexbox works on two axes:

i. *Main Axis* ⇒ Runs from left to right by default.

ii. *Cross Axis* ⇒ Perpendicular to main axis, runs from top to bottom.



6. **Flexbox Properties:**
7. **Parent Properties:**
    a. **Parent Properties:**
        i. **display**: Defines a flex container. Always provide the *display : flex;* property to parent container.
        Syntax: display : flex;

        ii. **justify-content**: The *justify-content* property is used to align the flex items. It aligns the items along the main axis.
        Syntax:

```
justify-content: flex-start|flex-end|center|space-between|space-around|space-evenly;
```

        iii. **align-items**: The *align-items* property is used to align the flex items. Aligns multiple lines of items on the cross axis.
        Syntax:

```
.container {
  align-items: stretch | flex-start | flex-end | center | baseline | first baseline | last baseline | start | end | self-start;
}
```

        iv. **flex-direction**: Defines the main axis direction. By default *flex-direction : row;* for display flex property.
        Syntax:

```
.container {
  flex-direction: row | row-reverse | column | column-reverse;
}
```

v. **flex-wrap:** By default, flex items will all try to fit onto one line. The *flex-wrap* property specifies whether the flexible items should wrap or not.

Syntax:

```
.container {
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```

vi. **flex-flow:** The *flex-flow* property is a shorthand property for *flex-direction* and *flex-wrap*.

The default value for flex-flow is *row nowrap*.

Syntax:

```
flex-flow: flex-direction flex-wrap;
```

vii. **align-content:** This aligns a flex container's lines within when there is extra space in the cross-axis.

Syntax:

```
.container {
  align-content: flex-start | flex-end | center | spa
ce-between | space-around | space-evenly | stretch |
start | end | baseline | first baseline;
}
```

b. Child Properties:

i. order:

1. The *order* property specifies the order of a flexible item relative to the rest of the flexible items inside the same container.
2. By default, flex items are laid out in the source order.
3. However, the order property controls the order in which they appear in the flex container.
4. By default order value is **0** (zero).
5. **Syntax**:

```
.item {
  order: number;
 }
```

ii. flex-grow:

1. The *flex-grow* property specifies how much the item will grow relative to the rest of the flexible items inside the same container.
2. This defines the ability for a flex item to grow if necessary.
3. By default value for flex-grow is **0** (zero).
4. **Syntax**:

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

iii. **flex-shrink:**

1. The *flex-shrink* property specifies how the item will shrink relative to the rest of the flexible items inside the same container.
2. By default value for flex-grow is **1**.
3. *Negative numbers are invalid*.
4. **Syntax**:

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

iv. **flex-basis:**

1. The *flex-basis* property in CSS is used to specify the initial size of the flexible item.
2. The flex property is not used if the element is not a flexible item.
3. Default value is ⇒ auto .
4. **Syntax**:

```
.item {
  flex-basis: number | auto
}
```

v. **flex:**

1. The *flex* property is a shorthand property for *flex-grow* , *flex-shrink*, *flex-basis* combined.
2. The flex property sets the flexible length on flexible items.

3. The second and third parameters (flex-shrink and flex-basis) are optional.
4. The default is **0 1 auto.**
5. **Syntax**:

```
.item {
    flex: flex-grow, flex-shrink flex-bais;
}
```

vi. **align-self:**

1. The *align-self* property specifies the alignment in the block direction for the selected item inside a flexbox or grid container.
2. **Syntax**:

```
.item {
  align-self: auto | flex-start | flex-end | center | ba
seline | stretch;
}
```

CSS Flexbox

container

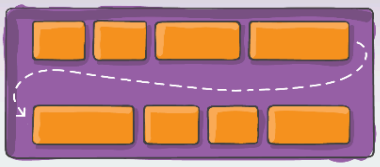
items

```
.container {
  display: flex; /* or inline-flex */
}
```

flex-direction

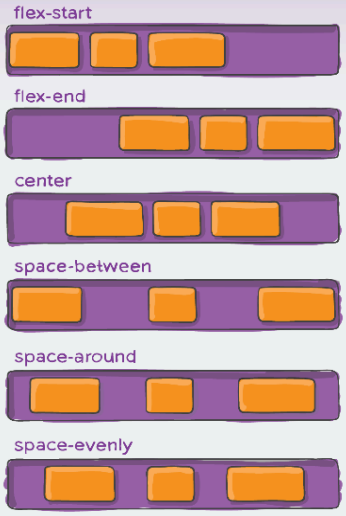
```
.container {
  flex-direction: row | row-reverse |
  column | column-reverse;
}
```

flex-wrap



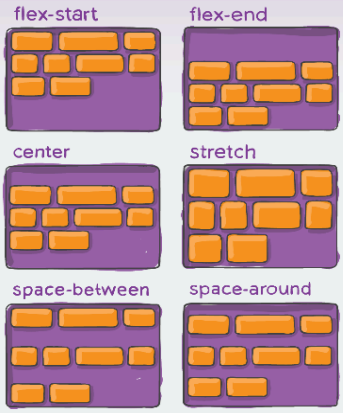
```
.container {
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```

justify-content



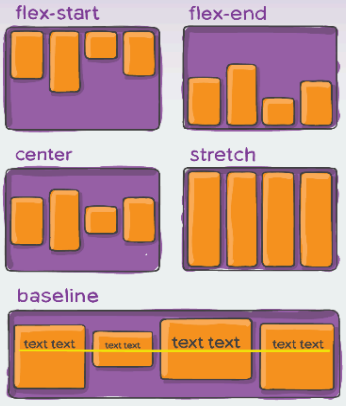

```
.container {
  justify-content: flex-start | flex-end |
  center | space-between | space-around |
  space-evenly;
}
```

align-content



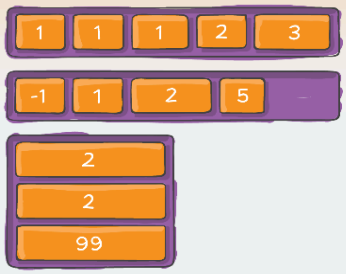

```
.container {
  align-content: flex-start | flex-end |
  center | space-between | space-around |
  space-evenly | stretch;
}
```

align-items




```
.container {
  align-items: stretch | flex-start |
  flex-end | center | baseline;
}
```

order

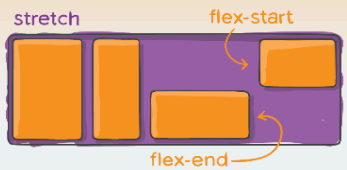



```
.item {
  order: 5; /* default is 0 */
}
```

flex-shrink, flex-grow, flex-basis


```
.item {
  flex-shrink: 1; /* default is 1 */
  flex-grow: 2; /* default is 0 */
  flex-basis: 50px; /* default auto */
}
```

align-self




```
.item {
  align-self: auto | flex-start |
  flex-end | center | baseline | stretch;
}
```