CSS PART-3

INTRODUCTION TO WEB DEVELOPMENT

R.A: **Assad Ullah Khan**

assadullahkhan

AssadUllahKhan

FLEX IN CSS

- In CSS, the flexbox layout model provides a flexible way to distribute **space** and **align** items within a **container**. It allows you to create **responsive** and dynamic **layouts** for your web pages. The **flexbox** layout consists of flex containers and flex items.

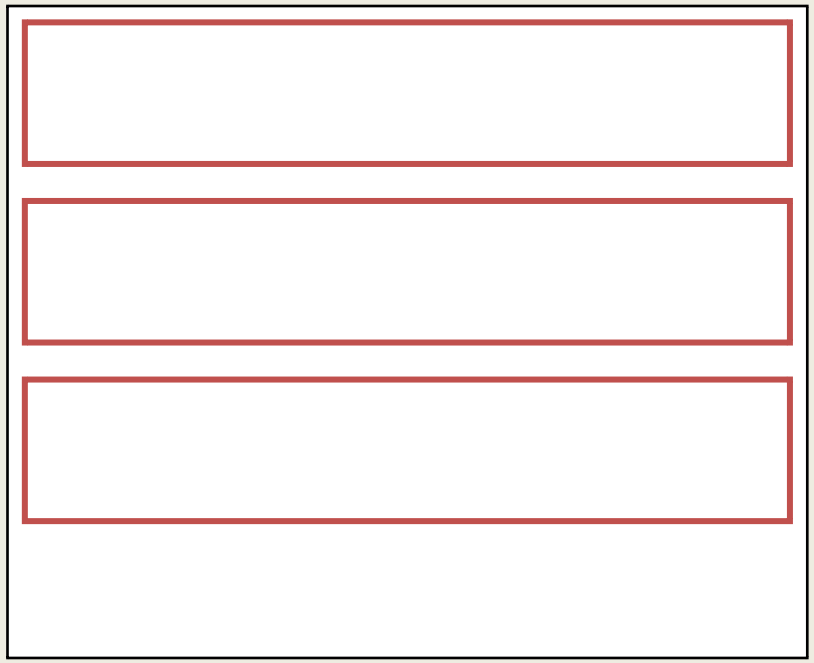
- To create a flex container, you need to apply the **display: flex** or **display: inline-flex** property to the parent element. This property enables the flexbox behavior on the container.

- Once you have a flex container, you can control the layout and alignment of its child elements, known as flex items, using various properties.

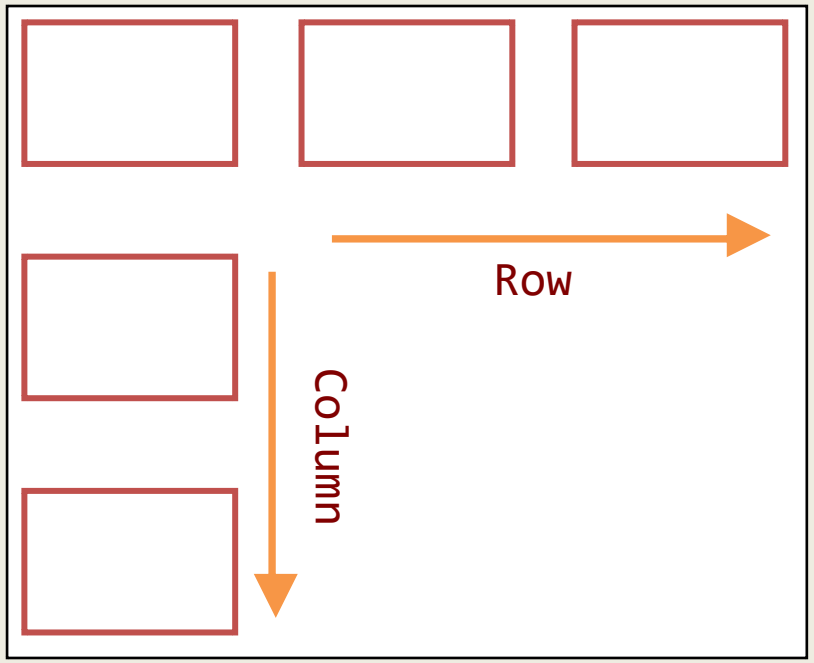
EXAMPLE

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

Without Flex



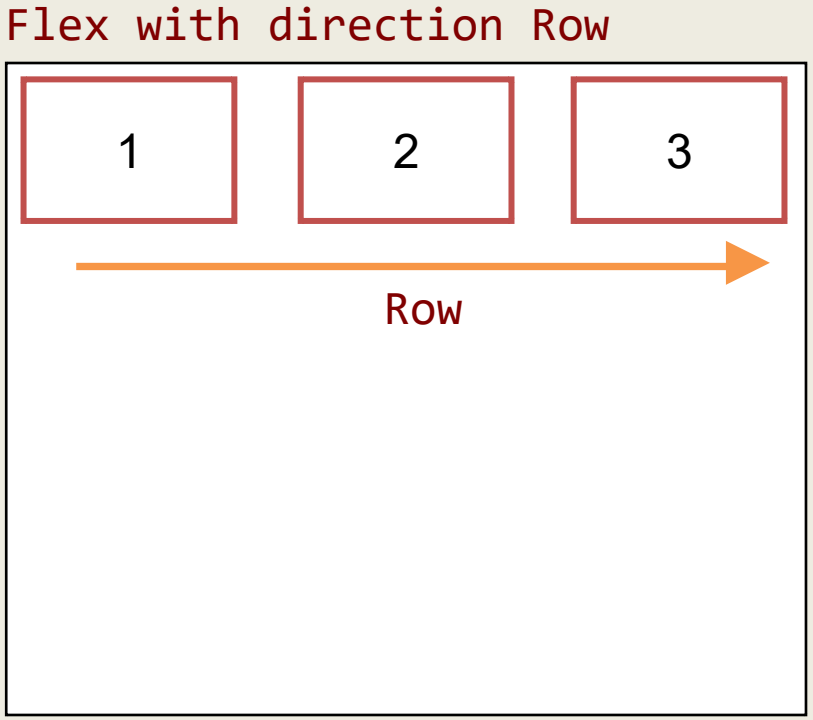
With Flex



FLEX DIRECTION

- The **flex-direction** property in CSS is used to determine the direction of the flex container's main axis, as well as the direction in which the flex items are laid out within the container. It accepts four possible values:
  1) Row
  2) Row-reverse
  3) Column
  4) Column-reverses

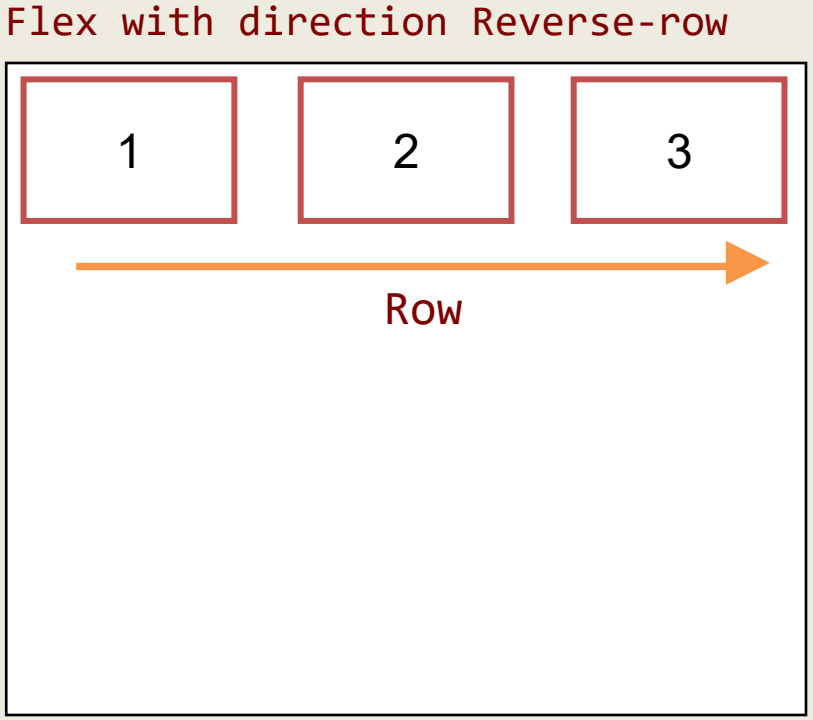
```
.container{
    display: flex;
    border: 2px solid blue;
    padding: 20px;
    flex-direction: row;
}
```



FLEX DIRECTION

- The **flex-direction** property in CSS is used to determine the direction of the flex container's main axis, as well as the direction in which the flex items are laid out within the container. It accepts four possible values:
  1) Row
  2) Row-reverse
  3) Column
  4) Column-reverses

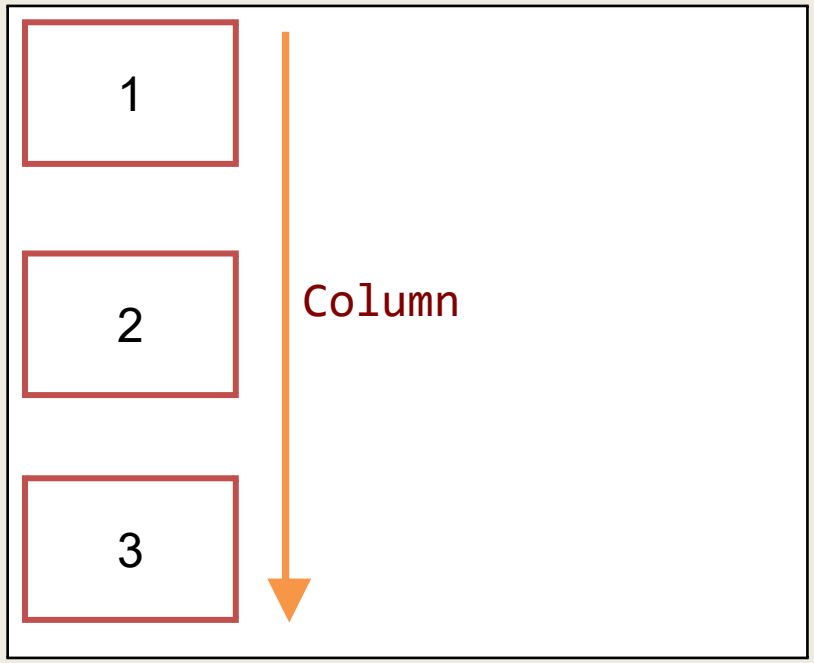
```
.container{
    display: flex;
    border: 2px solid blue;
    padding: 20px;
    flex-direction: row-reverse;
}
```



FLEX WITH COLUM DIRECTION

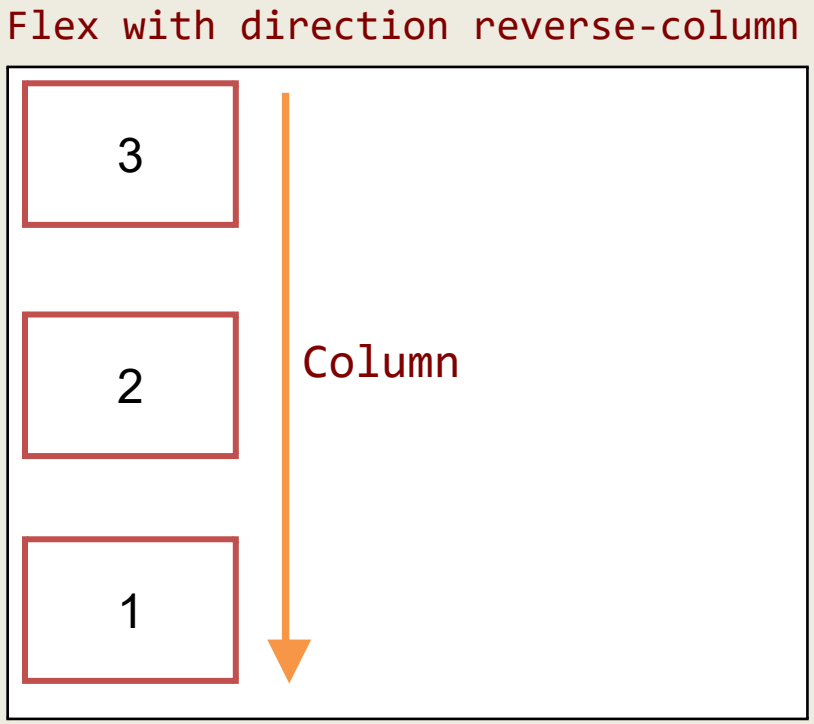
```
.container{
    display: flex;
    border: 2px solid blue;
    padding: 20px;
    flex-direction: column;
}
```

Flex with direction Column



FLEX WITH Reverse-column DIRECTION

```
.container{
    display: flex;
    border: 2px solid blue;
    padding: 20px;
    flex-direction: column-reverse;
}
```



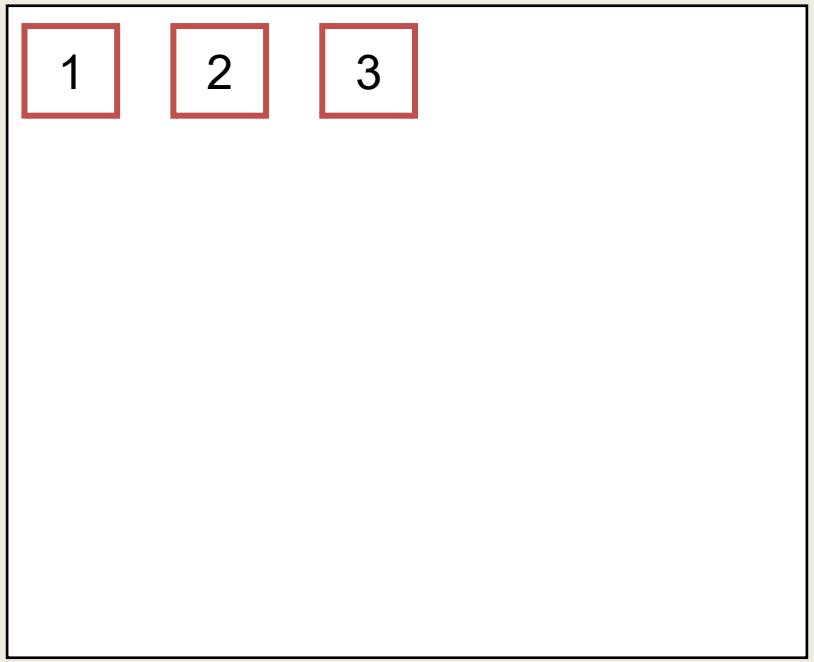
FLEX JUSTIFY CONTENT

- The **justify-content** property in CSS is used to control the alignment and positioning of flex items along the main axis of a flex container. It defines how extra space is distributed between and around the flex items.
    1) flex-start
    2) flex-end
    3) Center
    4) space-between
    5) space-around
    6) space-evenly

FLEX START

```
.container{
    display: flex;
    border: 2px solid blue;
    padding: 20px;
    flex-direction: row;
    justify-content: flex-start;
}
```

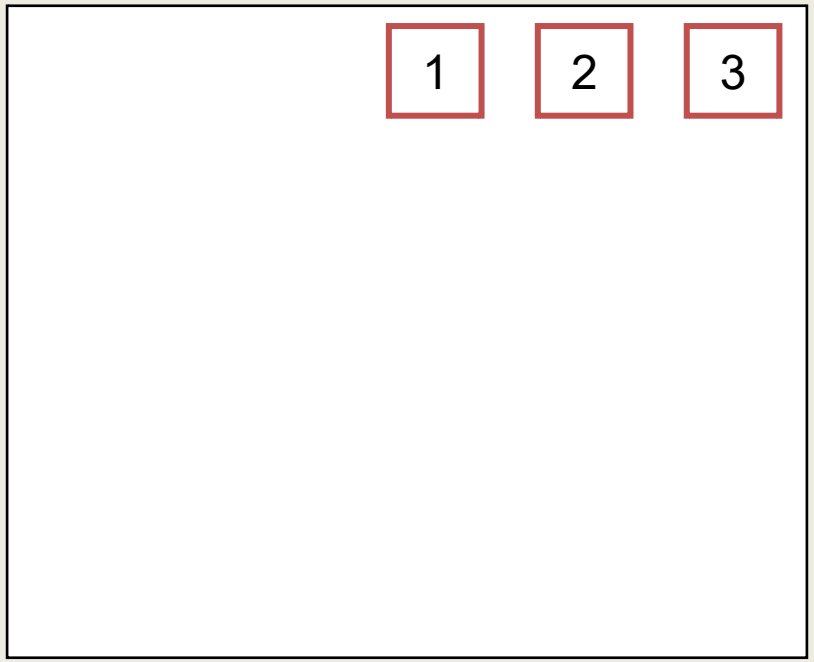
Flex with flex-start



FLEX END

```
.container{
    display: flex;
    border: 2px solid blue;
    padding: 20px;
    flex-direction: row;
    justify-content: flex-end;
}
```

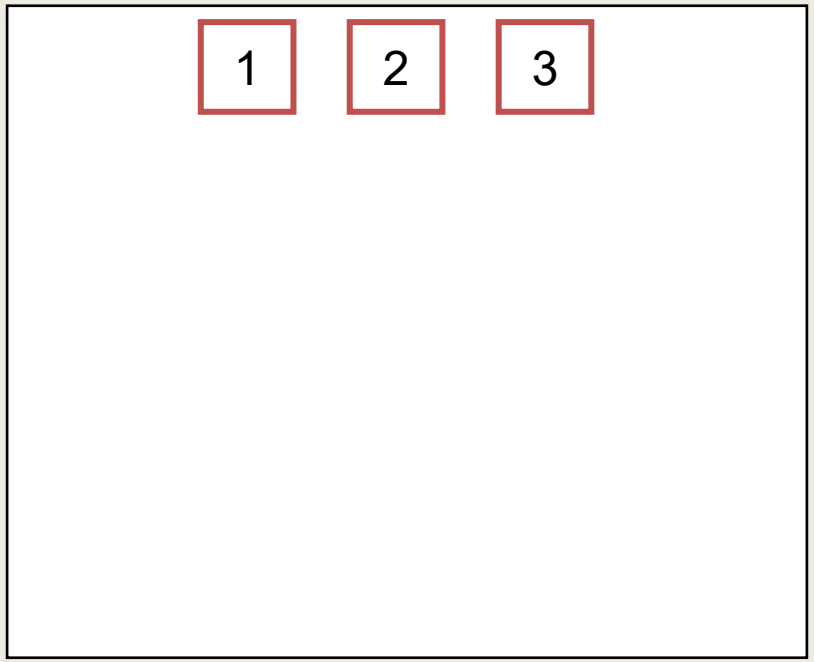
Flex with justify-content flex-end



FLEX CENTER

```
.container{
    display: flex;
    border: 2px solid blue;
    padding: 20px;
    flex-direction: row;
    justify-content: center;
}
```

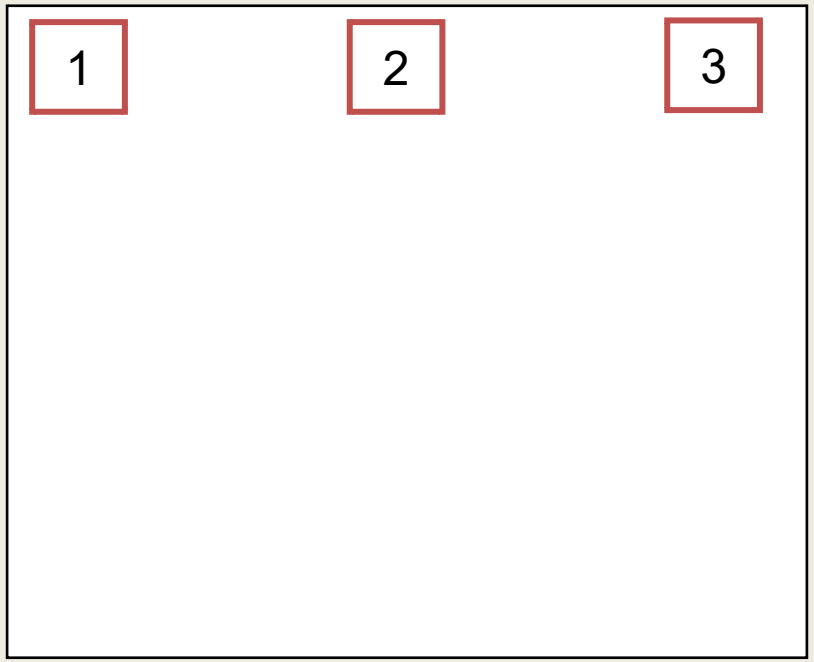
Flex with justify-content center



FLEX SPACE BETWEEN

```
.container{
    display: flex;
    border: 2px solid blue;
    padding: 20px;
    flex-direction: row;
    justify-content: space-between;
}
```

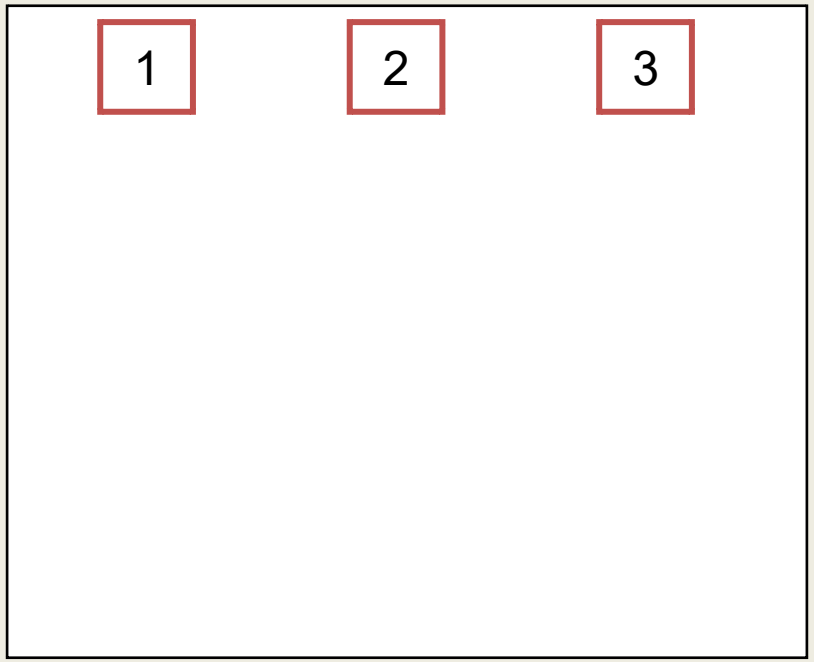
Flex with justify-content space between



FLEX SPACE AROUND

```
.container{
    display: flex;
    border: 2px solid blue;
    padding: 20px;
    flex-direction: row;
    justify-content: space-around;
}
```

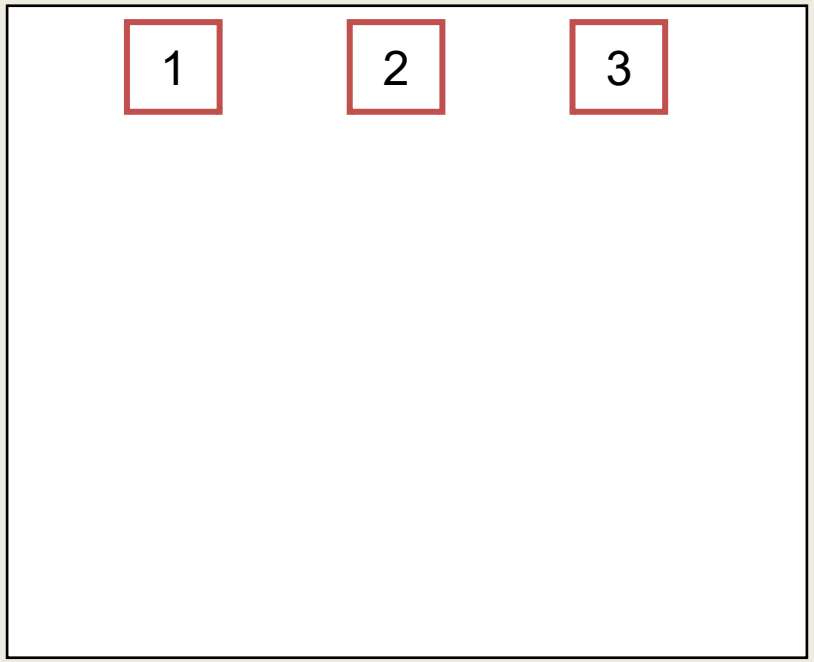
Flex with justify-content space around



FLEX SPACE EVENLY

```
.container{
    display: flex;
    border: 2px solid blue;
    padding: 20px;
    flex-direction: row;
    justify-content: space-evenly;
}
```

Flex with justify-content space evenly



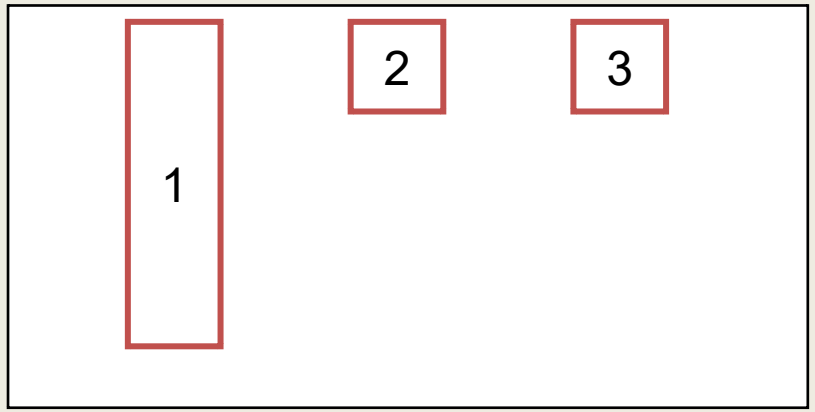
FLEX ALIGN ITEMS

- The **align-items** property in CSS is used to control the alignment and positioning of flex items along the cross axis of a flex container. It determines how flex items are vertically aligned within the container. The **align-items** property accepts several values:
  1. Stretch
  2. flex-start
  3. flex-end
  4. Center
  5. baseline

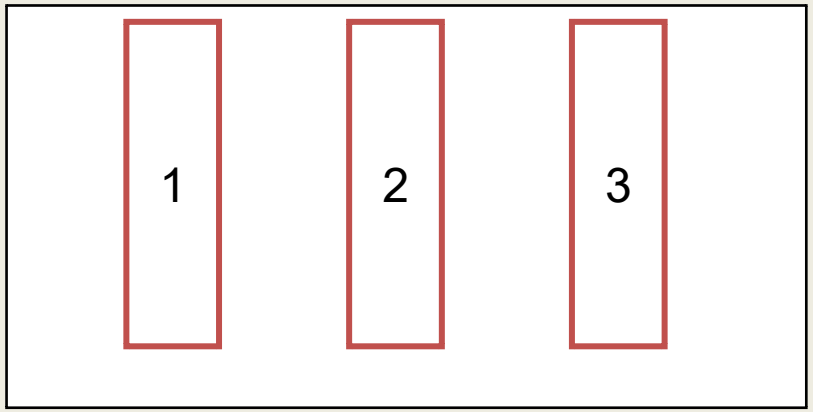
ALIGN ITEM STRTCH

```
.container{
    display: flex;
    border: 2px solid blue;
    padding: 20px;
    flex-direction: row;
    align-items: stretch;
}
```

Flex without align stretch



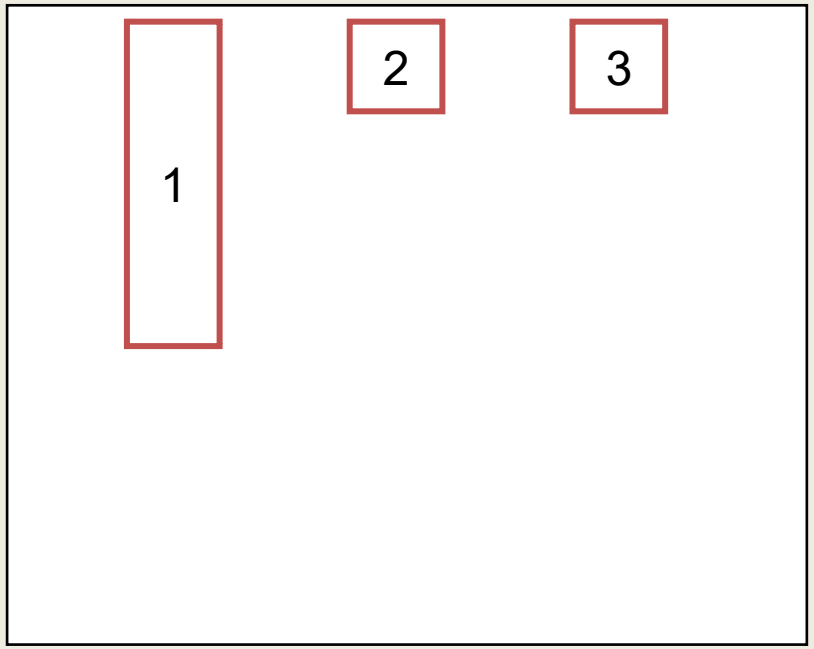
Flex with align stretch



ALIGN ITEM FLEX-START

```
.container{
    display: flex;
    border: 2px solid blue;
    padding: 20px;
    flex-direction: row;
    align-items: flex-start;
}
```

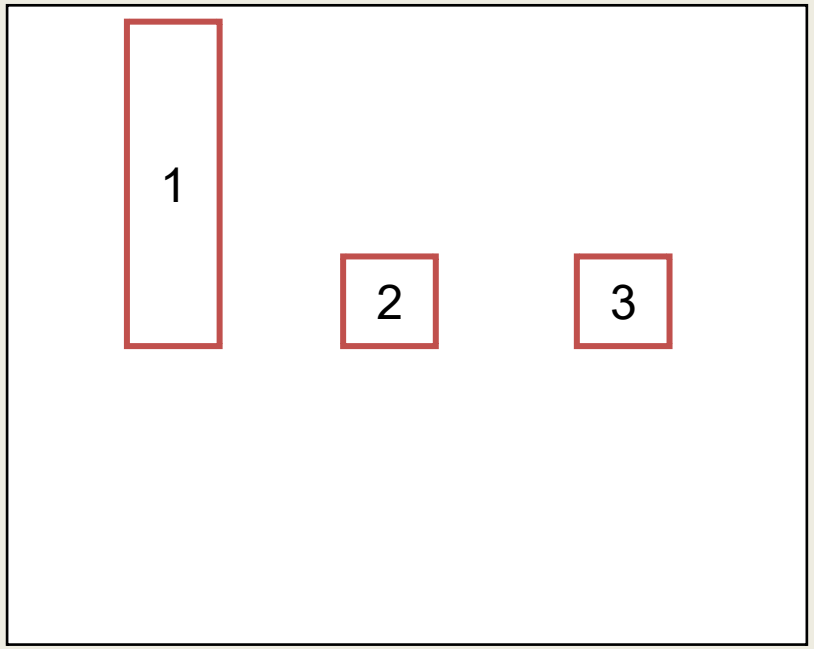
Flex with align flex-start



ALIGN ITEM FLEX-END

```
.container{
    display: flex;
    border: 2px solid blue;
    padding: 20px;
    flex-direction: row;
    align-items: flex-end;
}
```

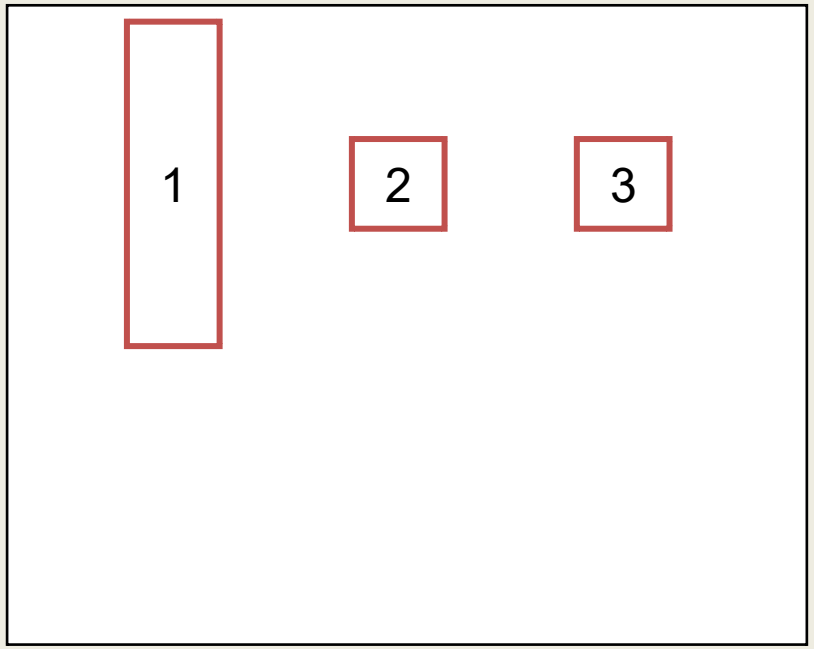
Flex wit align flex-end



ALIGN ITEM CENTER

```
.container{
    display: flex;
    border: 2px solid blue;
    padding: 20px;
    flex-direction: row;
    align-items: center;
}
```

Flex wit align center

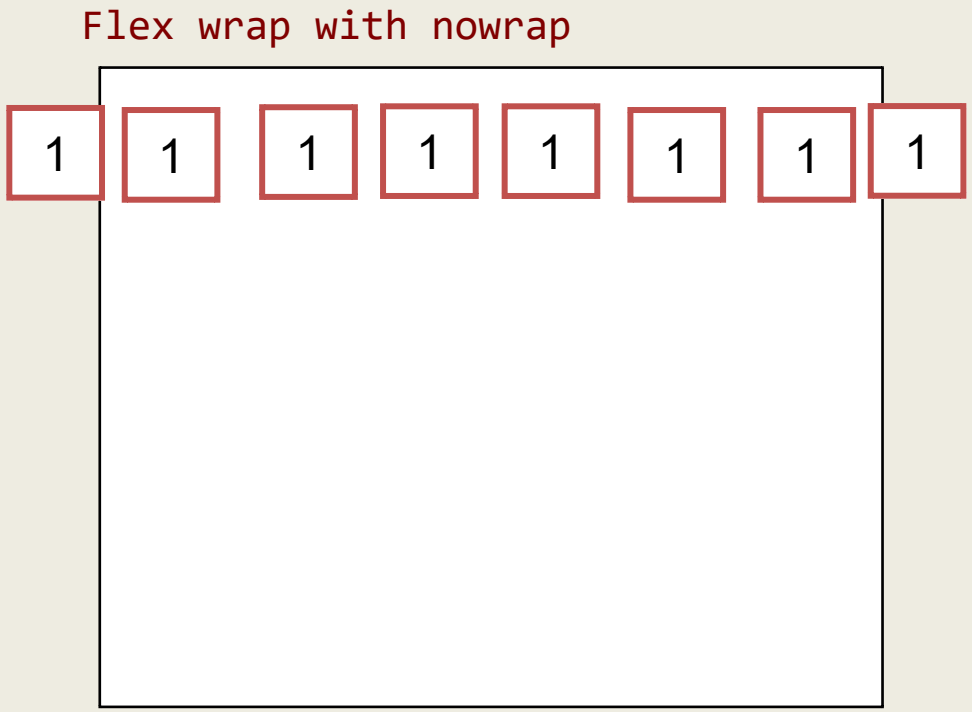


FLEX WRAP

- The **flex-wrap** property in CSS is used to control whether flex items should wrap to a new line or stay on the same line within a flex container when they exceed the container's width (or height, depending on the **flex-direction**). It accepts three possible values:
  1) Nowrap
  2) Wrap
  3) wrap-reverse

FLEX WRAP WITH NO WRAP

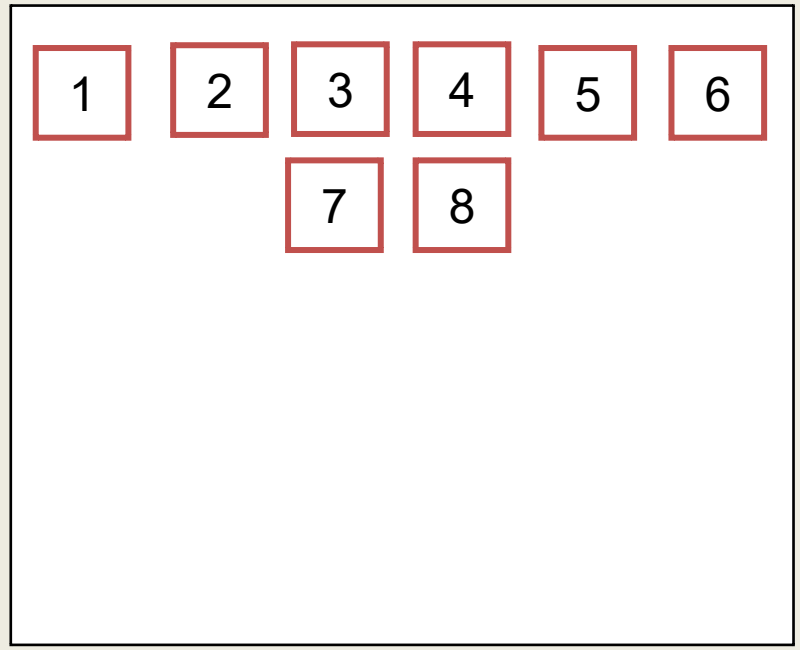
```
.container{
    display: flex;
    border: 2px solid blue;
    padding: 20px;
    flex-direction: row;
    flex-wrap: norwap;
}
```



FLEX WRAP WITH WRAP

```
.container{
    display: flex;
    border: 2px solid blue;
    padding: 20px;
    flex-direction: row;
    flex-wrap: wrap;
}
```

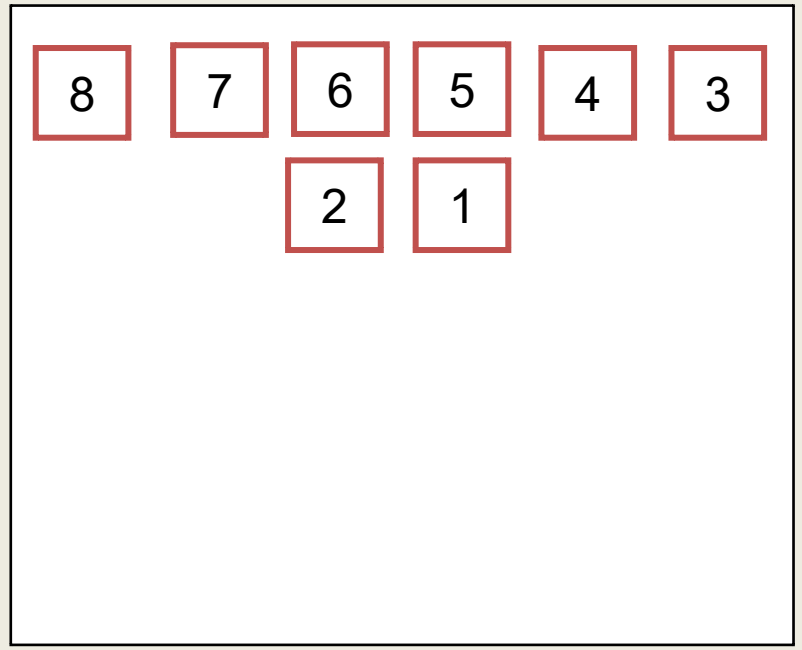
Flex wrap with wrap



FLEX WRAP WITH WRAP-REVERSE

```
.container{
    display: flex;
    border: 2px solid blue;
    padding: 20px;
    flex-direction: row;
    flex-wrap: wrap-reverse;
}
```

Flex wrap with wrap



FLEX GROW

- The **flex-grow** property in CSS is used to control the ability of a flex item to grow and fill the available space within a flex container.

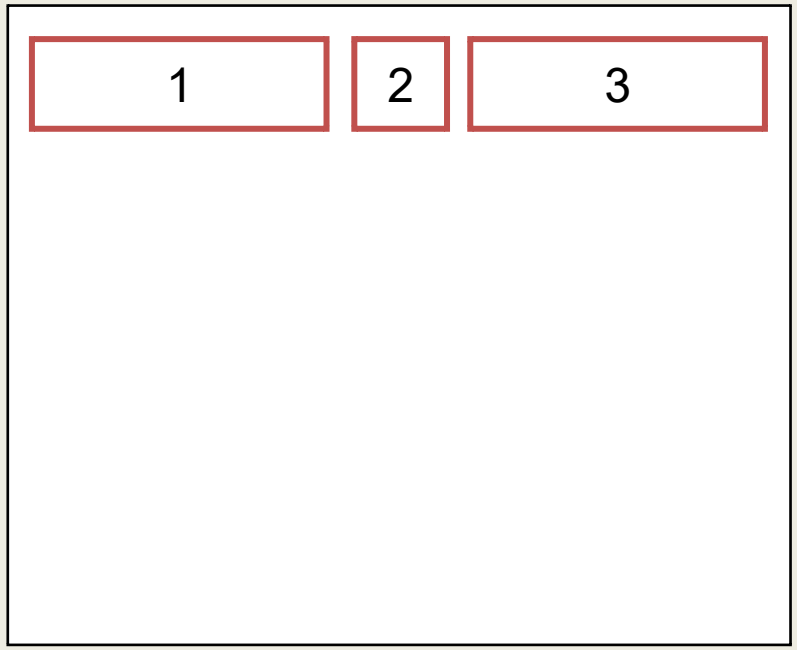
  - If all flex items have a **flex-grow** value of 0, they will not grow and will retain their original size.

  - If one or more flex items have a positive **flex-grow** value, the remaining space in the flex container is distributed among them proportionately. For example, if one item has a **flex-grow** of 2 and another has a **flex-grow** of 1, the first item will take twice as much space as the second item.

FLEX GROW

```
.div1{
    flex-grow: 1;
}
.div2{
    flex-grow: 0;
}
.div3{
    flex-grow: 1;
}
```

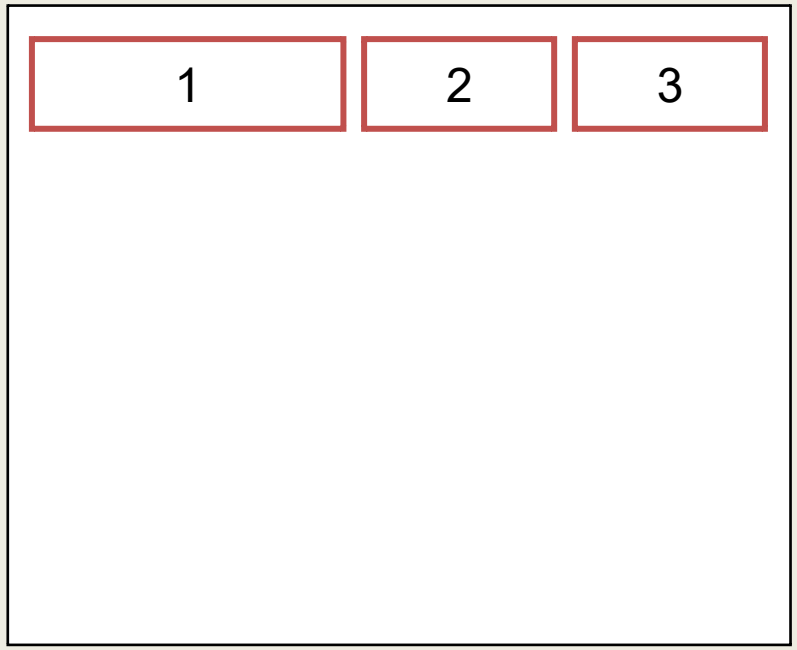
Flex Grow



FLEX GROW

```
.div1{
    flex-grow: 2;
}
.div2{
    flex-grow: 1;
}
.div3{
    flex-grow: 1;
}
```

Flex Grow



FLEX SHRINK

- The **flex-shrink** property in CSS is used to control the ability of a flex item to shrink when there is not enough space available in the flex container. It specifies the relative scale at which the flex item will shrink compared to other flex items in the container. The **flex-shrink** property accepts a unitless value that represents the shrink factor of the flex item.

    - If all flex items have a **flex-shrink** value of 0, they will not shrink and will retain their original size, potentially causing overflow of the container.
    - If one or more flex items have a positive **flex-shrink** value, the flex items will shrink proportionately based on their **flex-shrink** values to fit the available space.

FLEX BASIS

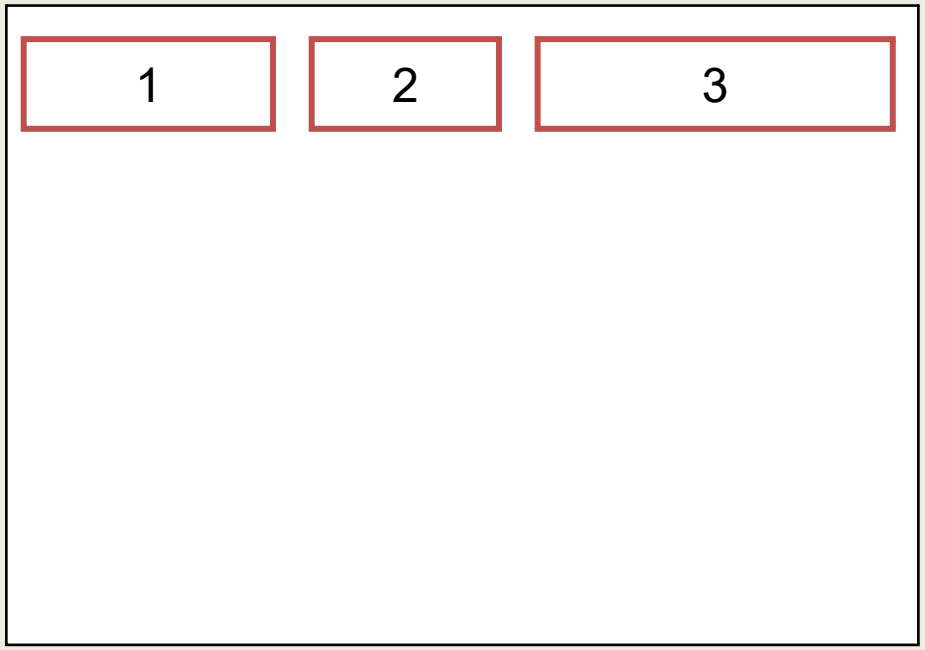
- The **flex-basis** property in CSS is used to specify the initial size of a flex item before any available space is distributed among the flex items. It determines the default size of a flex item along the main axis of the flex container.

1. Length value: You can specify a fixed length for the flex item using a value such as **px**, **em**, **rem**, etc. For example, **flex-basis: 200px** sets the initial width of the flex item to 200 pixels.
2. Percentage value: You can use a percentage value to specify the flex basis relative to the size of the flex container. For example, **flex-basis: 50%** sets the initial width of the flex item to 50% of the container's width.
3. **auto**: This value sets the initial size of the flex item based on its content and the flex container's available space. The flex item will expand or shrink as necessary.

FLEX BASIS

```
.div1{
    flex-basis: 400px;
}
.div2{
    flex-basis: 200px;
}
.div3{
    flex-basis: 700px;
}
```

Flex Basis



End

I hope this slide will help you to clear your idea