CSS: Flexbox

Lec-5

What is a Flexbox ?

- Flexbox is a tool use to **simplify positioning** of an element.
- It is a short form for- **"flexible box"** layout model.
- It is used to **arrange an element (vertically and horizontally)** within a container in a **flexible** and **responsive** way **without using float** and **positioning**.
- There are **two** important components to a flexbox layout:

1. **flex container-** It is an element on a page which contains **flex items.**
2. **flex items-** All direct **child elements** of a flex container.

- Any element can be flex container; **child** of flex container will **change size and location** in **response to the size and position of their parent container**.

Flex Container Properties

- Following are the properties of the flex container:

(i) display: flex/inline-flex

(ii) flex-direction

(iii) flex-wrap

(iv) justify-content

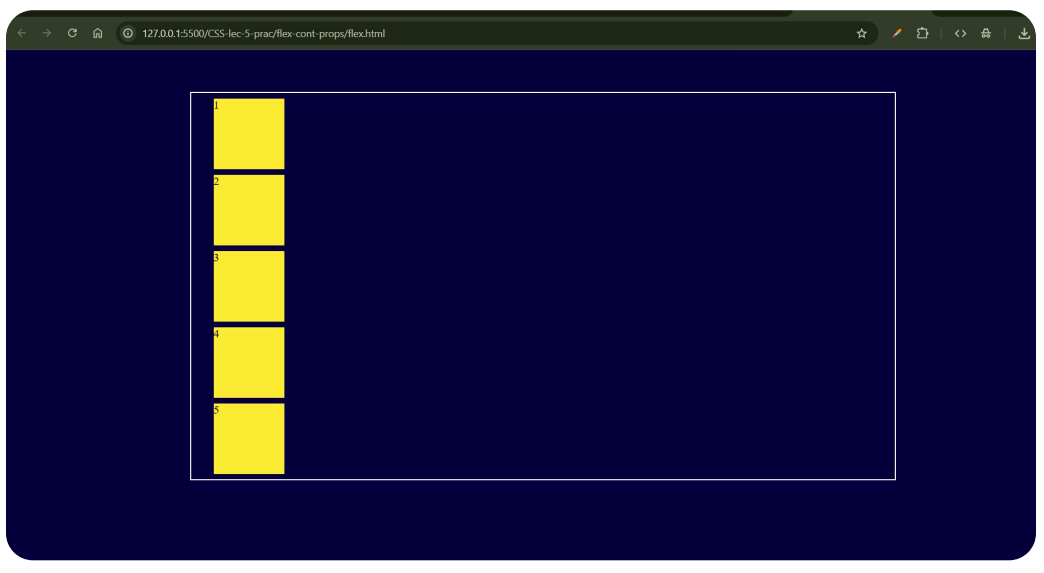
(v) align-items

(vi) align-content

1. display: flex

- This property will **transform any container** to a **flex container.**
- The **children's** of any such container would **flex items.**
- If we apply **display: flex** property on any block level element **it will still be a block level element.** However, **the behavior of it's child elements will change** such that **child elements will not begin on a new line**.

Eg:



before

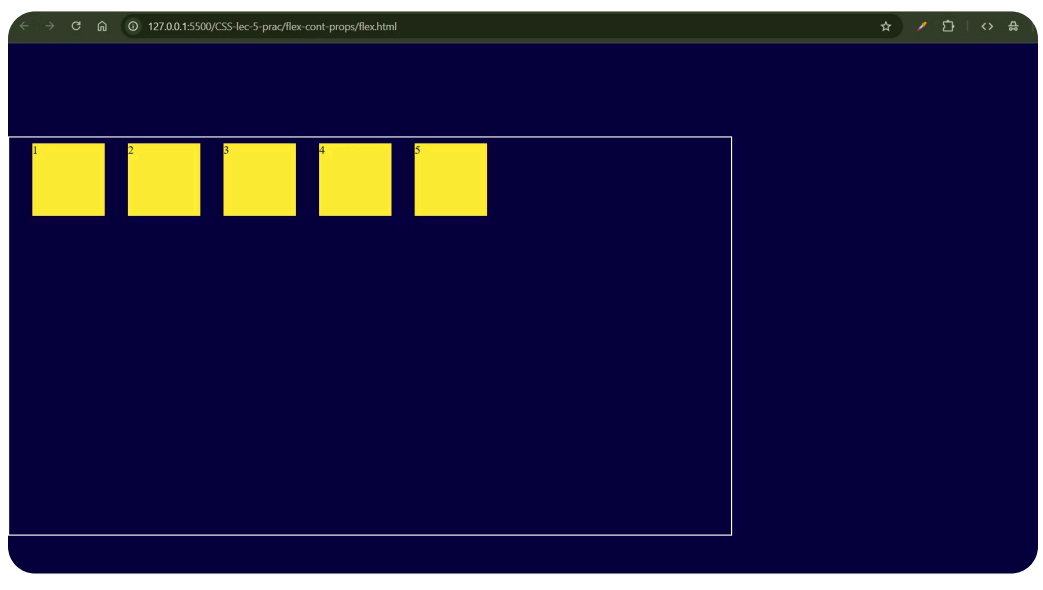
```
.container{
        display: flex;
      }
```

after

2. display: inline-flex:

- This will work in a same way as of **display: flex.**
- The only difference will be when we apply **display: inline-flex** property on any **parent container**, which is a **block level element; such element would be treated as a inline element.**

display: flex



display: inline-flex

Flex Container Properties

3. flex-direction:

- We must know that the **flex containers have two axes**.
- **Main Axis (x- axis) -** The Horizontal axis.
- **Cross Axis (y-axis) -** The Vertical axis.
- Using **flex-direction** property we can **decide the axis flex container should consider** while **arranging it's flex items.**
- The **flex-direction** property takes values:

1. **row**
2. **row-reverse**
3. **column**
4. **column-reverse**

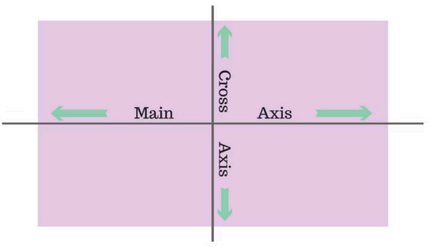


image credit: https://freecodecamp.org/

(i) row:

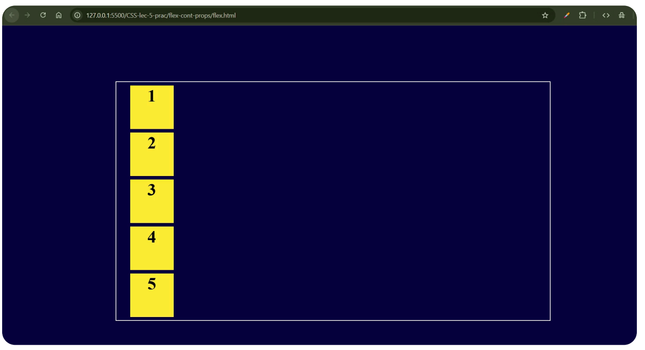
- This is the default value.
- It will **position flex items** from **left to right across the parent element starting from the top left corner**.

(ii) row-reverse:

- It will **position flex items** from **right to left across the parent element starting from the top right corner**.

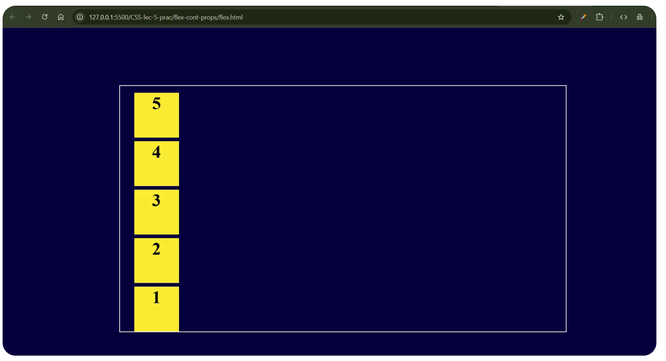
(iii) column:

- It will **position flex items** from **top to bottom across the parent element starting from the top left corner**.



(iv) column-reverse:

- It will **position flex items** from **top to bottom across the parent element starting from the bottom left corner**.



Flex Container Properties

4. flex-wrap:

- If there **isn't an enough space for flex items in one single flex line**, then by using **flex-wrap** property we can **decide whether the flex items should wrap or not.**
- **flex-wrap** property accepts **3 values:**

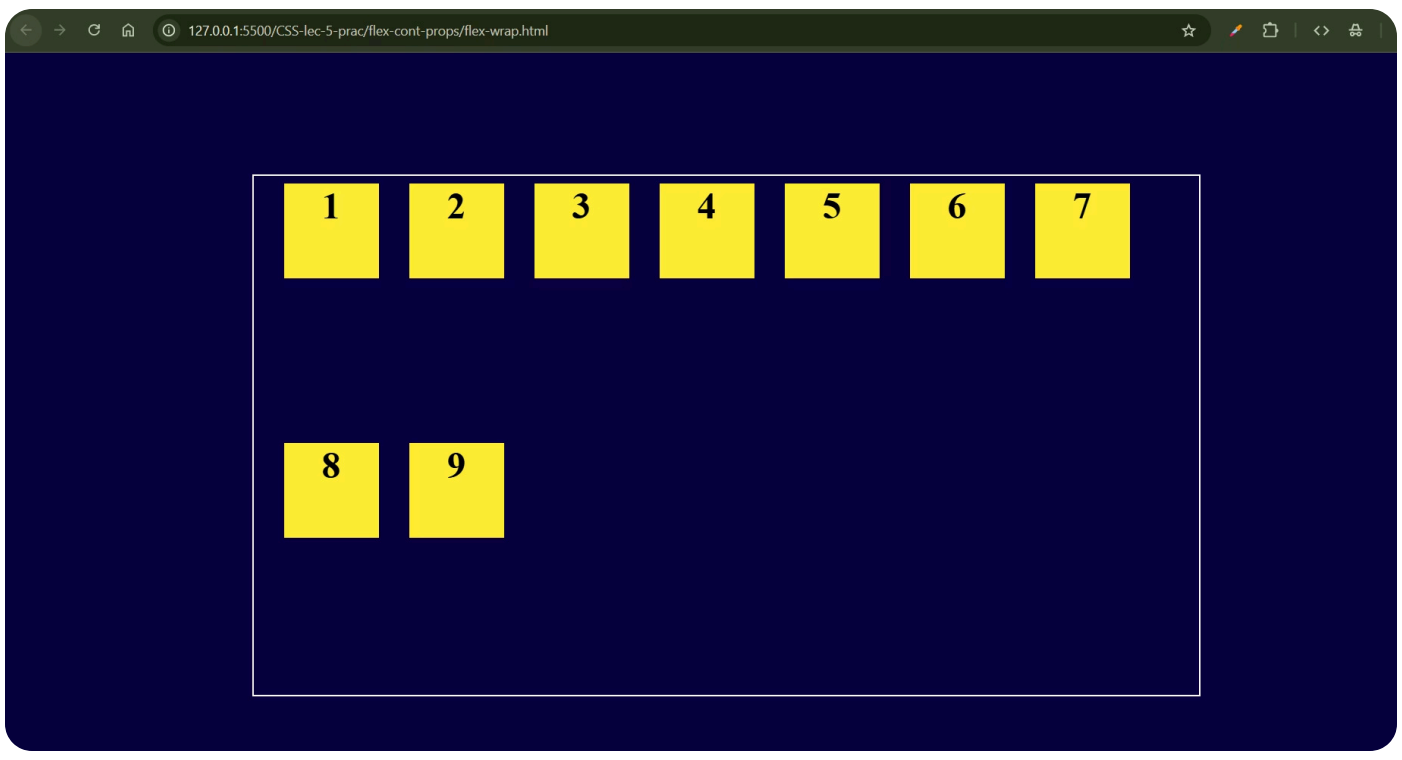
1. **nowrap**
2. **wrap**
3. **wrap-reverse**

(i) nowrap:

- This is the default value.
- It **prevents flex-items from getting wrapped**.

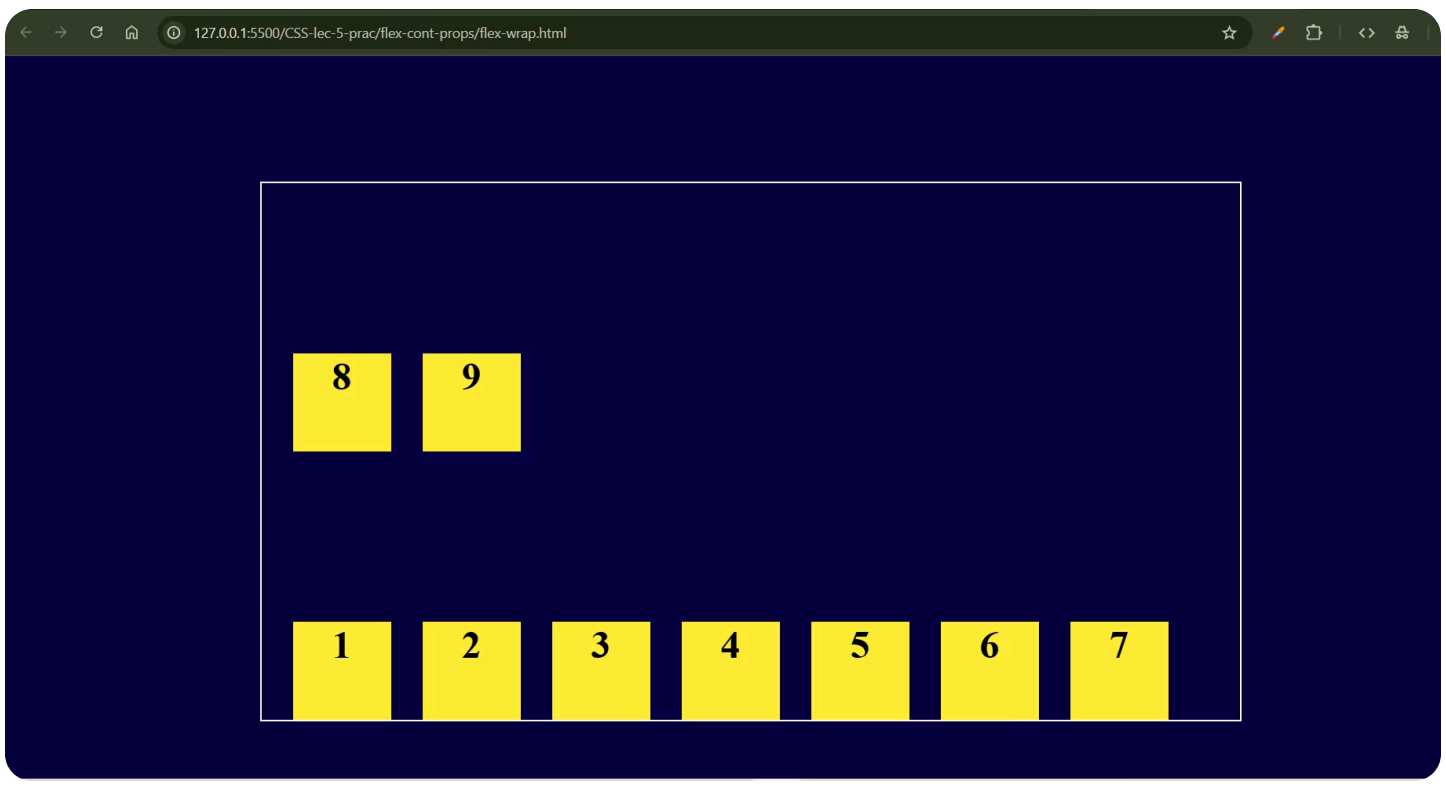
(ii) wrap:

- If any **flex item is not fitting in a single flex line**, it will **move that item into next line**.



(ii) wrap-reverse:

- If any **flex item is not fitting in a single flex line**, it will **move that item into next line**, but the **order of flex items would be reverse**.



Note:

- There is a property called **flex-flow**, which is a **shorthand property** to declare values for **flex-direction** and **flex-wrap.**
- Eg: **flex-flow: row wrap**

  flex-flow: column nowrap

  flex-flow: row-reverse wrap-reverse

Flex Container Properties

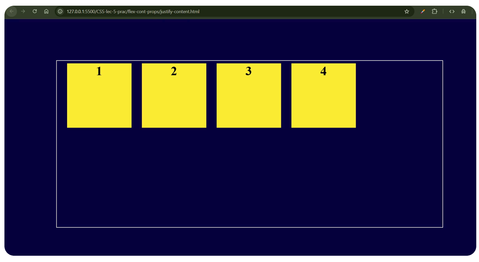
5. justify-content:

- In order to position the **flex items** from **left to right / horizontally / along x-axis / along main axis**, we use the property called as **justify-content.**
- It accepts **6 values:**

1. **flex-start**
2. **flex-end**
3. **center**
4. **space-around**
5. **space-between**
6. **space-evenly**

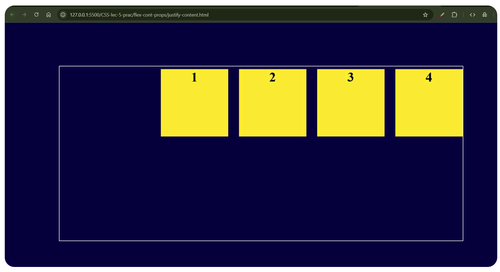
(i) flex-start:

- All flex items **will be positioned starting from the left of the parent container**, with **no extra space between or before them**.



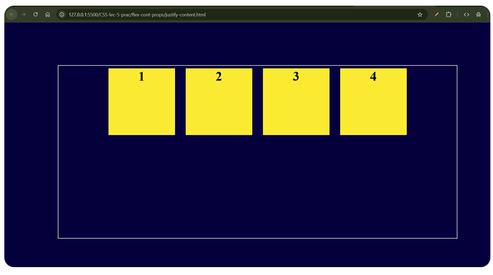
(ii) flex-end:

- All flex items **will be positioned starting from the right of the parent container**, with **no extra space between or before them**.



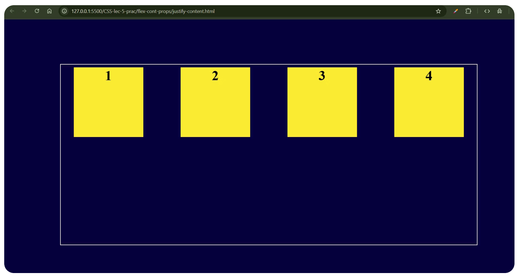
(iii) center:

- All flex items **will be positioned in the horizontal center of the parent container**, with **no extra space between or before them**.



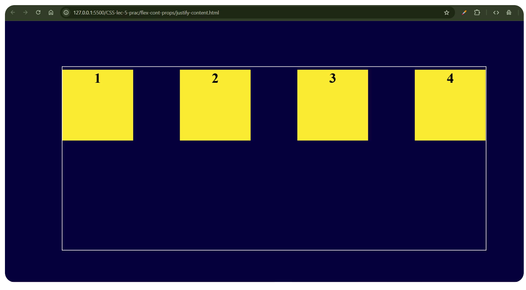
(iv) space-around:

- All flex items **will be positioned with equal space before and after each item,** resulting in **double space between elements.**



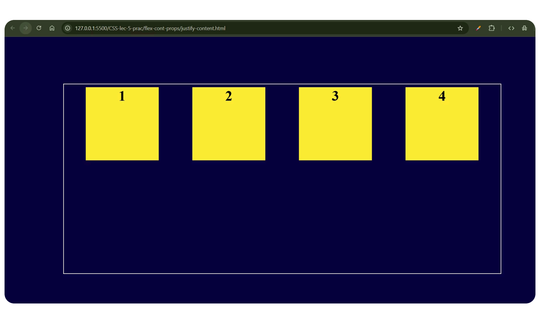
(v) space-between:

- All flex items **will be positioned with equal space between them,** but **no extra space before the first or after the last element.**



(v) space-evenly:

- All flex items **will be positioned with equal space between them,** and also **equal space before the first element and after the last element.**



Flex Container Properties

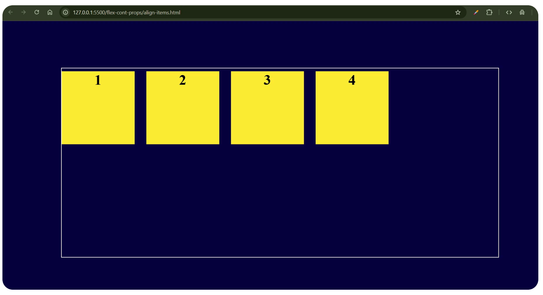
6. align-items:

- In order to position the **flex items vertically / along y-axis / along cross axis**, we use the property called as **align-items.**
- It accepts **5 values:**

1. **flex-start**
2. **flex-end**
3. **center**
4. **baseline**
5. **stretch**

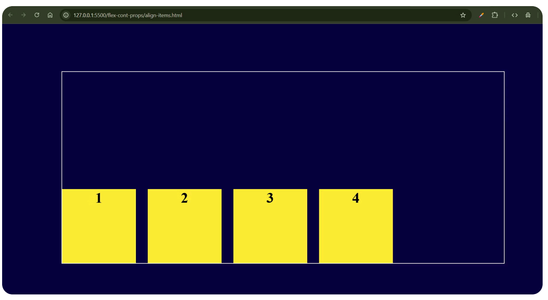
(i) flex-start:

- All elements will be **positioned at the top of the parent container**.



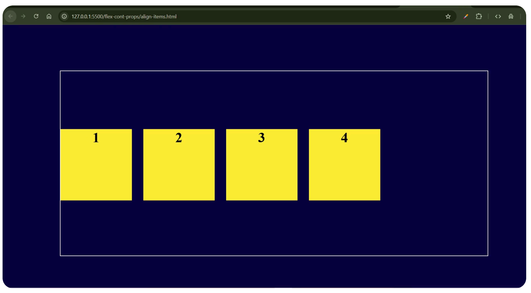
(ii) flex-end:

- All elements will be **positioned at the bottom of the parent container**.



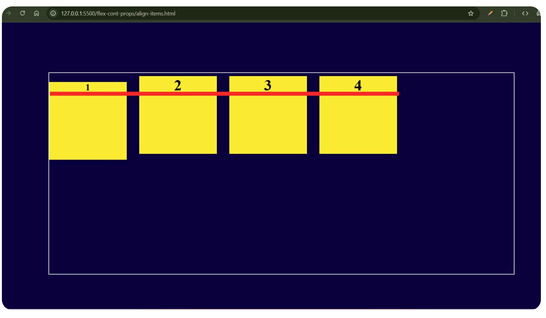
(iii) center:

- The center of all elements will be **positioned halfway between top and bottom of parent container.**



(iv) baseline:

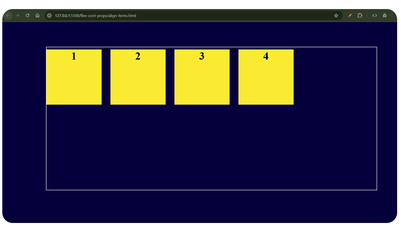
- The **bottom of the content of all items will be aligned with each other**.



(v) stretch:

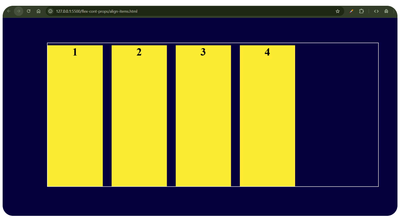
- This is a **default value.**
- If possible, the flex items will be **stretched from top to bottom of the container**. **(only when height of elements are not specified or has a minimum height).**

```
.items{
      width: 200px;
      height: 200px;
      background-color: yellow;
    }
```



before

```
.items{
      width: 200px;
      background-color: yellow;
    }
.container{
        align-items: stretch;
      }
```



after

Flex Container Properties

7. align-content:

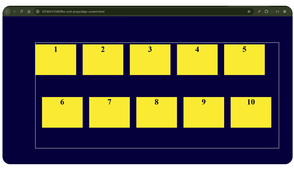
- This property is **similar to align-items.**
- As **align-items** property is to **aligning the elements within a single row**, but **if a flex container has multiple rows of content** we use **align-content** property to **space rows from top to bottom (across y axis / cross axis).**
- It accepts **7 values:**

1. **center**
2. **flex-start**
3. **flex-end**
4. **space-around**
5. **space-between**
6. **space-evenly**
7. **stretch**

(i) center:

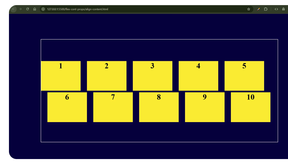
- The **flex lines** will be **packed towards the center of the container**.

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



before

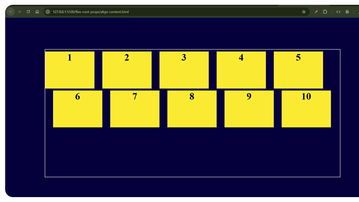
```
.container{
        display: flex;
        flex-wrap: wrap;
        align-content: center;
      }
```



after

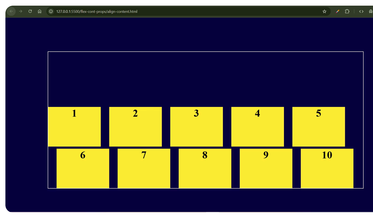
(ii) flex-start:

- The **flex lines** will be **packed towards the start of the container**.



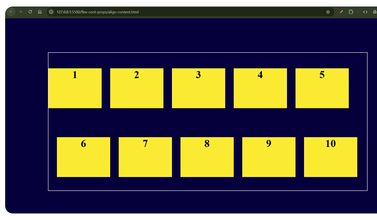
(ii) flex-end:

- The **flex lines** will be **packed towards the end of the container**.



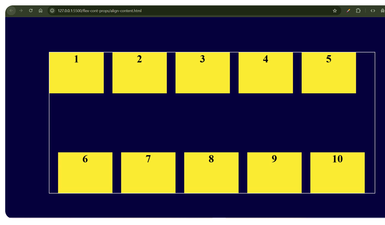
(iii) space-around:

- The **flex lines** will be **packed evenly from the top to the bottom of the container**. with **same amount of space at the top and bottom and between each element**.



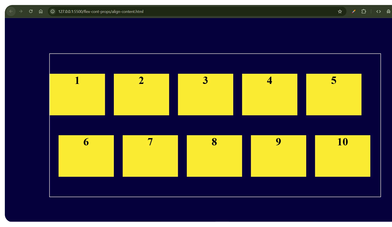
(iv) space-between:

- The **flex lines** will be **packed evenly from the top to the bottom of the container**. with **no space above the first element and below the last element.**



(v) space-evenly:

- The **flex lines** will be **packed evenly from the top to the bottom of the container**. with **equal space on top, bottom and between the elements.**



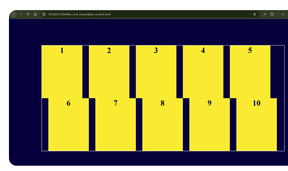
(vi) stretch:

- This is the **default property.**
- If **minimum height or no height is specified** the **rows of elements** will **stretch to fill the parent container from top to bottom.**

```
.items{
      width: 200px;
      height: 150px;
      background-color: yellow;
    }
```

before

```
.items{
      width: 200px;
      background-color: yellow;
    }
.container{
      align-content: stretch;
      }
```



after

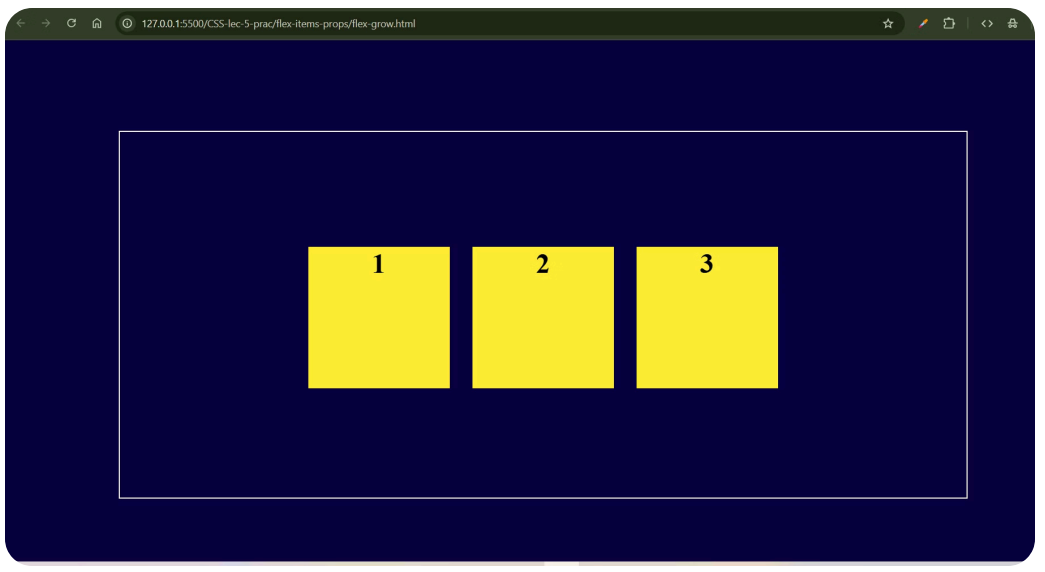
Flex Item Properties

- Following are the properties can be applied on the flex items.

1. **flex-grow**
2. **flex-shrink**
3. **flex-basis**
4. **align-self**
5. **order**

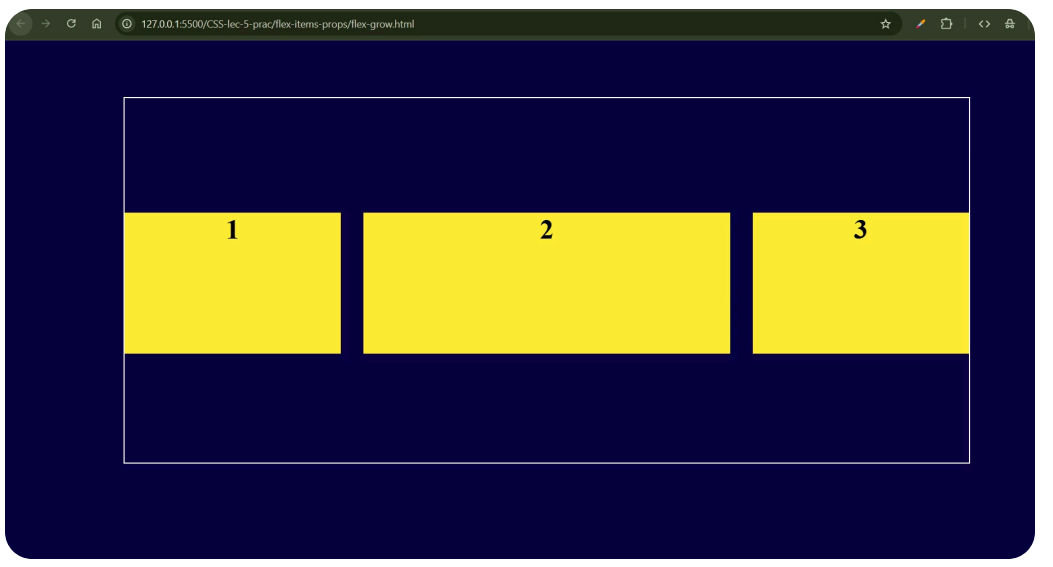
(i) flex-grow:

- This property allows us **to specify if the flex items should grow to fill a container** and **which flex items should grow proportionally more or less than others**.
- The **value this property takes is a number** and **default value is 0**.



before

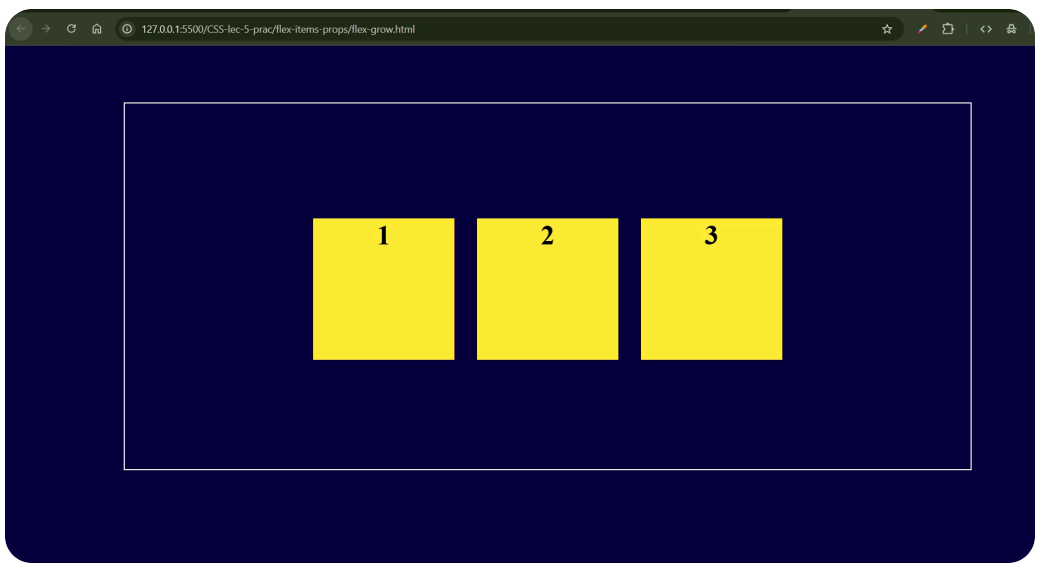
```
#first {
  flex-grow: 1;
}
#second {
  flex-grow: 3;
}
#third {
  flex-grow: 1;
}
```



after

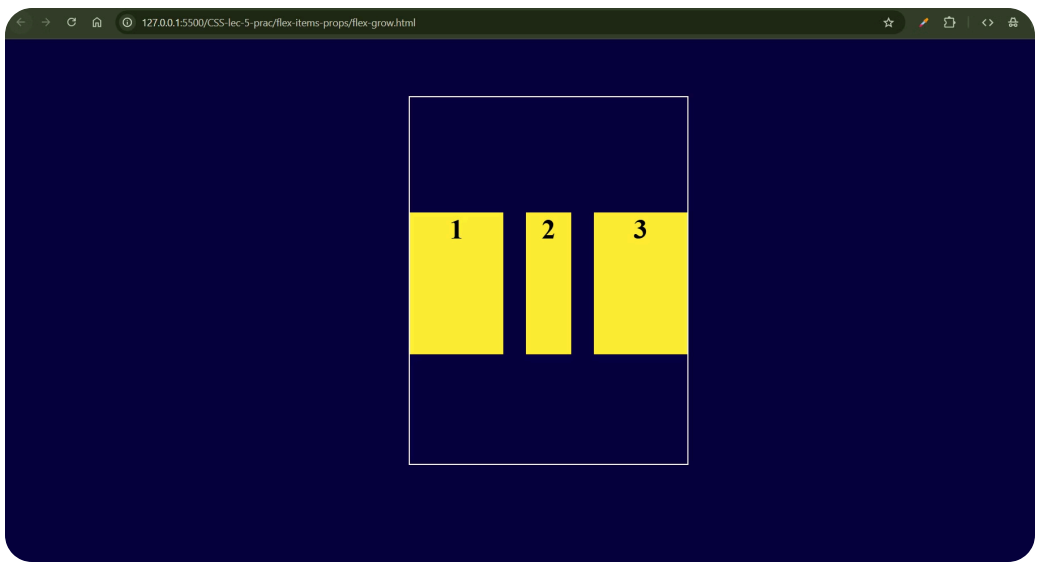
(ii) flex-shrink:

- This property allows us **to specify if the flex items should shrink in a container** and **which flex items should shrink proportionally more or less than others**.
- The **value this property takes is a number** and **default value is 1**.



before

```
#first {
  flex-shrink: 2;
 }
 #second {
  flex-shrink: 4;
 }
 #third {
  flex-shrink: 2;
 }
```

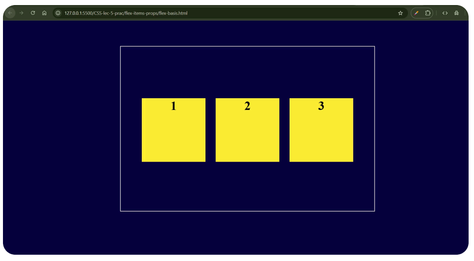


after

Flex Item Properties

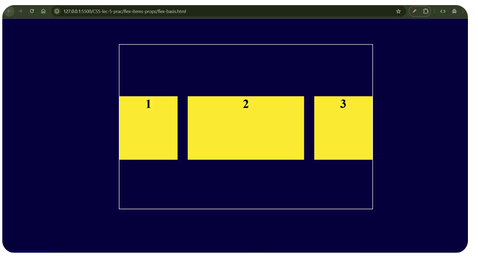
(iii) flex-basis:

- This property allows us to **specify the width of an item before it stretches or shrinks**.



before

```
#first {
    flex-shrink: 2;
  }
  #second {
    flex-shrink: 4;
    flex-basis: 400px;
  }
  #third {
    flex-shrink: 2;
  }
```

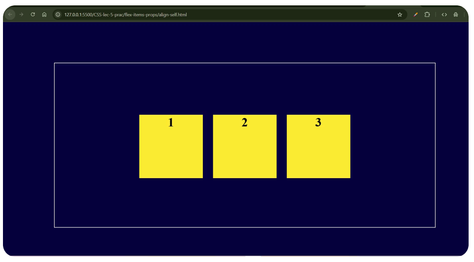


after

> **Note:**
>
> - There is a property called **flex**, which is a **shorthand property** to declare values for **flex-grow, flex-shrink** and **flex-basis.**
> - Eg: **flex: 1 2 250px;**

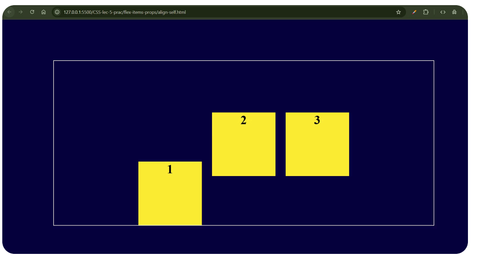
(iv) align-self:

- This property **specifies the alignment for the selected item inside the flexible container**.
- This property **overrides the default alignment set by the container's align item's property**.



before

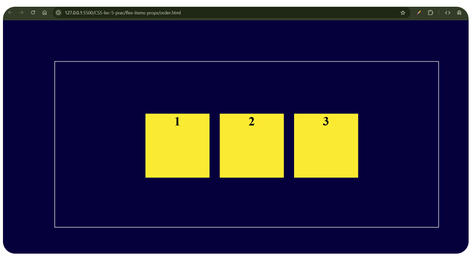
```
.container{
        display: flex;
        justify-content: center;
        align-items: center;
      }

#first{
     align-self: flex-end;
    }
```



after

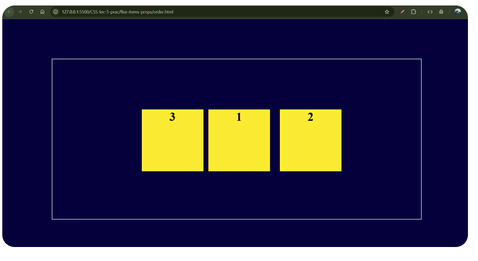
(v) order:

- This property specifies the **display order of the flex items inside the flex container**.
- The **first flex item** in the source code **does not** have **to appear as the first item in the layout**.
- The **order value** must be a **number**, and the **default value is 0**.



before

```
#first {
    order 2;
  }
  #second {
    order:3;
  }
  #third {
    order: 1;
  }
```



after