Day-5: CSS Flex

Introduction:

- CSS flexbox is a one-dimensional layout pattern that makes it easy to design flexible and effective layouts.
- Divide space between items and control their alignment in a given container flex layout.
- It also provides a lot of flexibility. With flexbox, we can organize items from left to right, top to bottom, and at the same time control the spacing and order of the items in the container.
- In flexbox, there are mainly two entities: `a parent container` (the flex container) and the immediate `children elements` (flex items).

Parent Container Properties

These are some of the properties that can be applied to a flex parent container:

- `display`
- `flex-direction`

- `flex-wrap`
- `flex-flow`
- `justified-content`
- `align-items`
- `align-content`

Without Flexbox

```
<div class="box"> <div>1</div> <div>2</div> <div>3</div> </div>
```

Output:

Live Code: Codepen

Flex display Property:

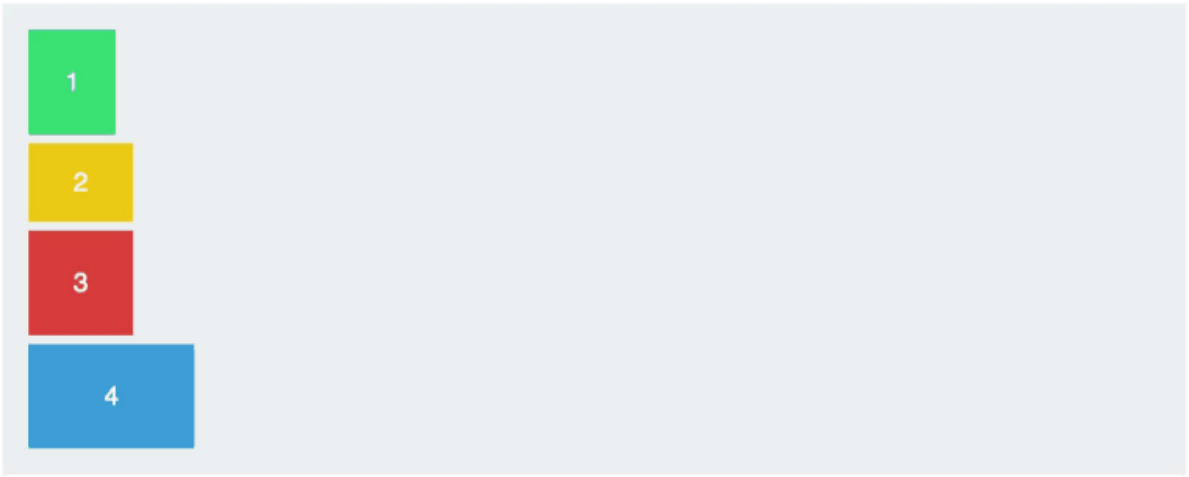
- First, we have the `display` property. This is what defines the flex container and it's mandatory when working with Flexbox.

```
<style> .box { display: flex; } </style> <div class="box"> <div>1</div> <div>2</div> <div>3</div> </div>
```

Output

Live Code : Codepen

display: block;



flex-direction

- Next is `flex-direction`, which defines the direction in which the flex items are placed in the container.
- `flex-direction` can accept one of four values:
  - `row`
  - `row-reverse`
  - `column`
  - `column-reverse`

flex-direction: `row`

- The first value is a row that is the `default` value of `flex-direction`. It doesn't make any changes by default. It's placed on the main axis from left to the right.

```
<style> .box { display: flex; flex-direction: row; } </style> <div class="box
"> <div>1</div> <div>2</div> <div>3</div> </div>
```

Output:

Live Code : Codepen

flex-direction: `row-reverse`

- This sets the main access direction from right to left, which results in the flex items being placed from right to left. In the example below, you can see that the items are now placed in the reverse order. Item 1 is placed to the right:

```
<style> .box { display: flex; flex-direction: row-reverse; } </style> <div cl
ass="box"> <div>1</div> <div>2</div> <div>3</div> </div>
```

Output:

Live Code : Codepen

flex-direction: `column`

- When you set `flex-direction` to `column`, the main axis goes from top to bottom, so the items are now stacked on top of each other:

```
<style> .box { display: flex; flex-direction: column; } </style> <div class="box"> <div>1</div> <div>2</div> <div>3</div> </div>
```

Output:

Live Code : Codepen

flex-direction: `column-reverse`

- We also have `column-reverse`, which stacks the items in the reverse order. Look at the example below. You can see Item 1 is at the bottom and Item 3 is at the top:

```
<style> .box { display: flex; flex-direction: column-reverse; } </style> <div class="box"> <div>1</div> <div>2</div> <div>3</div> </div>
```

Output:

Live Code : Codepen

flex-wrap

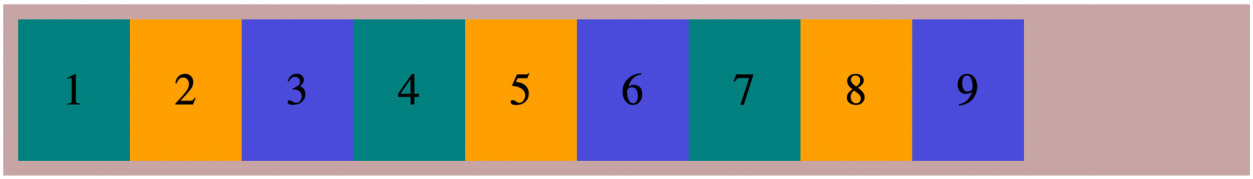
- `flex-wrap` is used to control the wrapping of items within a container. If we reduce the browser width, we lose some items for the browser width. The behavior changes with the `flex-wrap` property. It can accept three possible values:
- `nowrap` (default value)
- `wrap`
- `wrap-reverse`

flex-wrap: `nowrap`

- This is the `flex-wrap` property default value. If you set the property value to `nowrap`, there are no changes.

```
<style> .box { display: flex; flex-wrap: nowrap; } </style> <div class="box">
<div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div
> <div>7</div> <div>8</div> <div>9</div> </div>
```

Output:



Live Code : Codepen

Flex with defaults

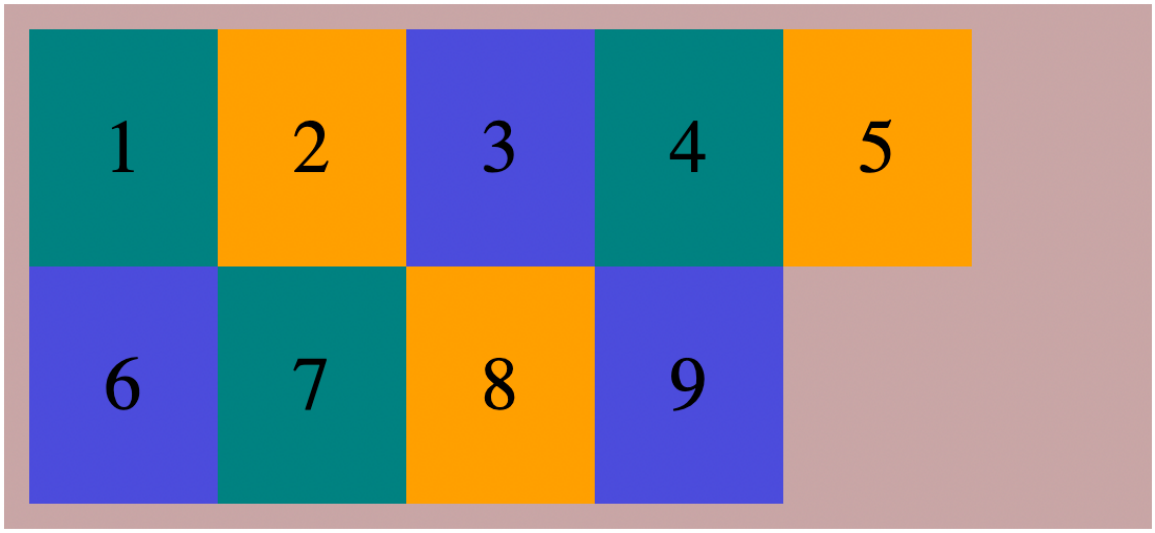
A b c d e f g h i j k l Middle n o p q r s t u v w x y z

flex-wrap: `wrap`

- When you set the `flex-wrap` property to `wrap`, you reduce the browser width that the items have wrapped in the container:

```
<style> .box { display: flex; flex-wrap: wrap; } </style> <div class="box"> <
div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div>
<div>7</div> <div>8</div> <div>9</div> </div>
```

Output:



Live Code : Codepen

Flex with defaults + flex-wrap: wrap;

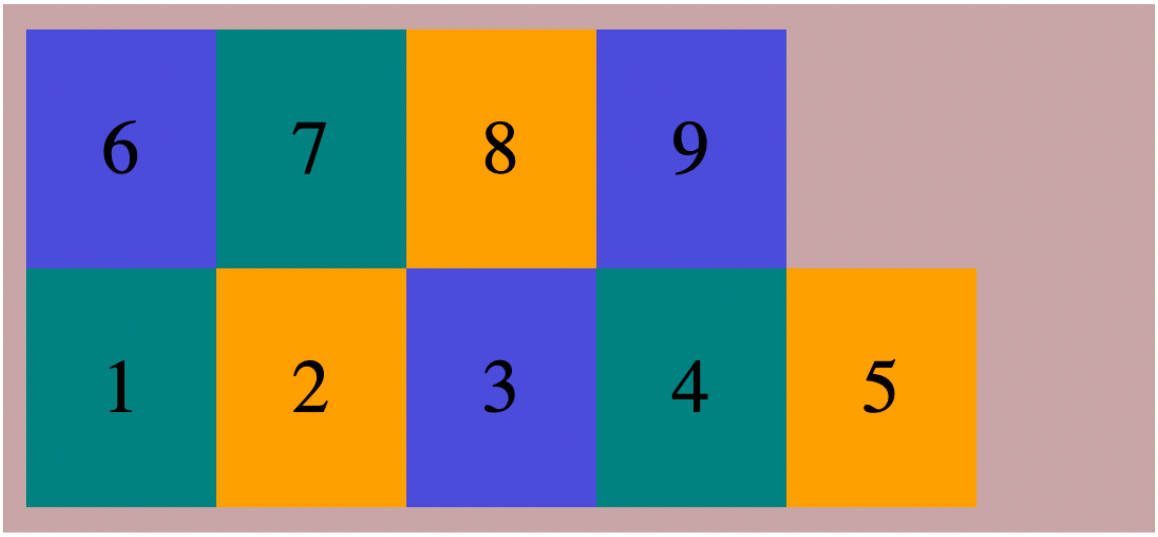
A b c d e f g h i j k l Middle n o p q r s t u v w x y z

flex-wrap: `wrap-reverse`

- Instead of items falling into the row below, they climb into the row above. Wrapping always occurs from the last item. `wrap-reverse` pushes the last item above instead of below:

```
<style> .box { display: flex; flex-wrap: wrap-reverse; } </style> <div class=
"box"> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div
>6</div> <div>7</div> <div>8</div> <div>9</div> </div>
```

Output:



Live Code : Codepen

justified-content

- `justified-content` defines the alignment of the items along the main axis. There are six possible values for the `justified-content` property:
  - `flex-start`
  - `flex-end`
  - `center`
  - `space-between`
  - `space-around`
  - `space-evenly`

justified-content: `flex-start`

- Setting the value to `flex-start` places the flex items at the beginning of the main axis, which is also known as the main start. `flex-start` is the default value in this property.

```
<style> .box { display: flex; justify-content: flex-start; } </style> <div cl
ass="box"> <div>1</div> <div>2</div> <div>3</div> </div>
```

Output:

Live Code : Codepen

justified-content: `flex-end`

- This aligns the items to be placed at the end of the main axis:

```
<style> .box { display: flex; justify-content: flex-end; } </style> <div clas
s="box"> <div>1</div> <div>2</div> <div>3</div> </div>
```

Output:

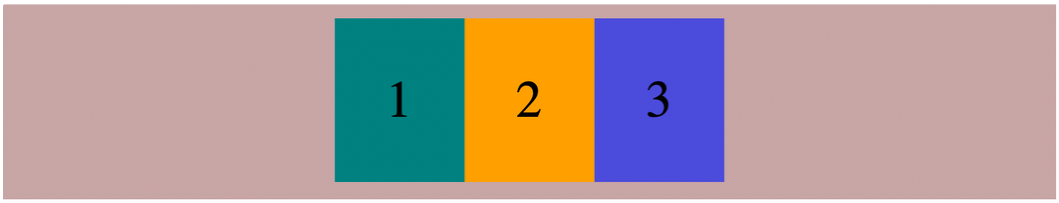
Live Code : Codepen

justified-content: `center`

- The `center` value aligns all content at the center of the main axis:

```
<style> .box { display: flex; justify-content: center; } </style> <div class=
"box"> <div>1</div> <div>2</div> <div>3</div> </div>
```

Output:



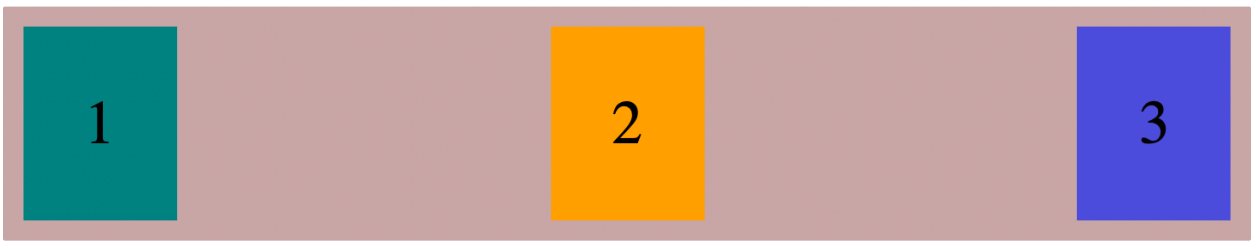
Live Code : Codepen

justified-content: `space-between`

- This value helps to evenly split extra space and add it in between the flex items:

```
<style> .box { display: flex; justify-content: space-between; } </style> <div class="box"> <div>1</div> <div>2</div> <div>3</div> </div>
```

Output:



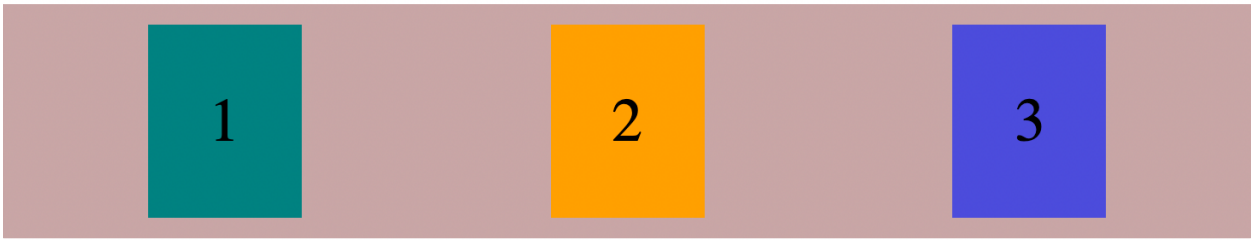
Live Code : Codepen

justified-content: `space-around`

- This value splits the extra space at the beginning and the end. The space in question is equal to half of the space between the flex items:

```
<style> .box { display: flex; justify-content: space-around; } </style> <div class="box"> <div>1</div> <div>2</div> <div>3</div> </div>
```

Output:



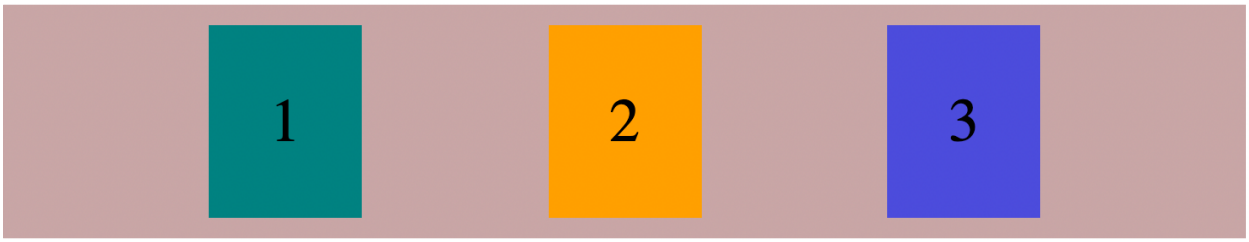
Live Code : Codepen

justified-content: `space-evenly`

- This value distributes the extra space in the container:

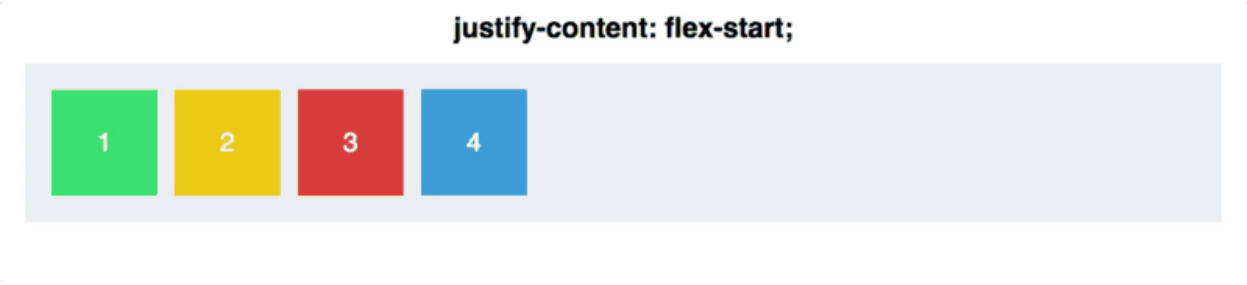
```
<style> .box { display: flex; justify-content: space-evenly; } </style> <div class="box"> <div>1</div> <div>2</div> <div>3</div> </div>
```

Output:



Live Code : Codepen

Justify-content Overview



align-items

The `align-items` property defines how the flex items are laid out along the cross axis.

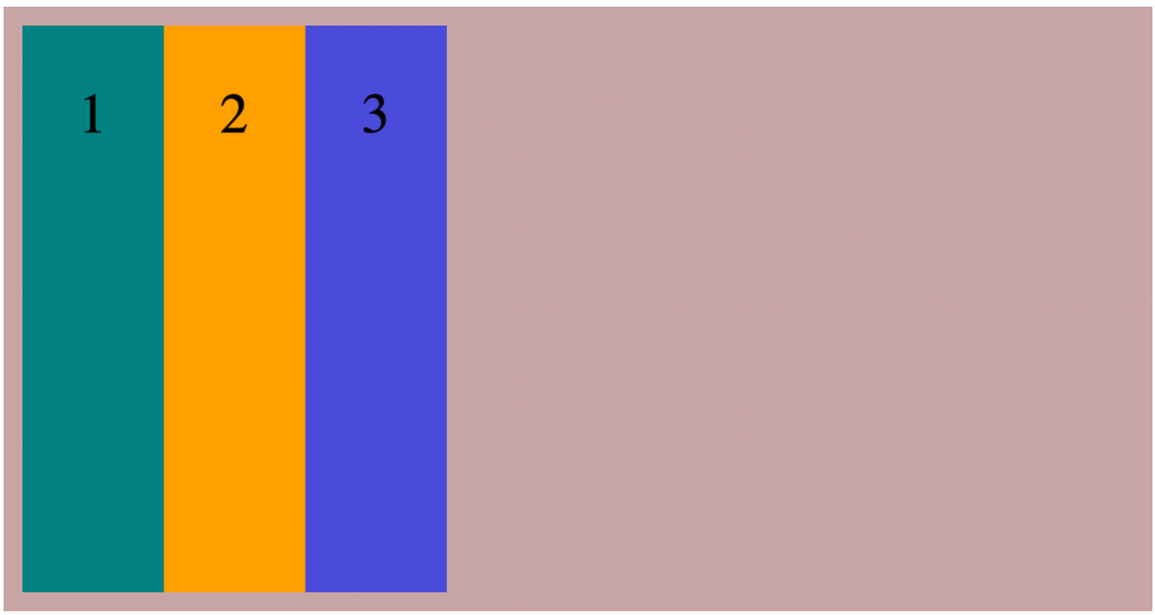
- `align-items: stretch`
- `align-items: flex-start`
- `align-items: flex-end`
- `align-items: center`
- `align-items: baseline`

align-items: `stretch`

- The items stretch the entire length of the cross axis and `stretch` is the `default` value:

```
<style> .box { display: flex; justify-content: space-evenly; } </style> <div class="box"> <div>1</div> <div>2</div> <div>3</div> </div>
```

Output:



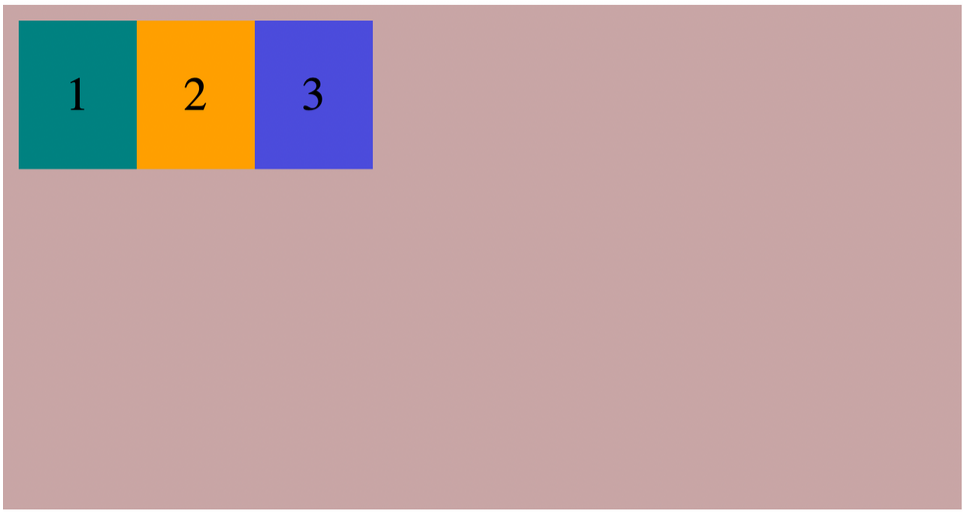
Live Code : Codepen

align-items: `flex-start`

- This is the starting point of the cross axis. The cross axis flows from top to bottom. The item doesn't stretch and is aligned with the cross start of the line:

```
<style> .box { height: 300px; display: flex; align-items: flex-start; } </sty
le> <div class="box"> <div>1</div> <div>2</div> <div>3</div> </div>
```

Output:



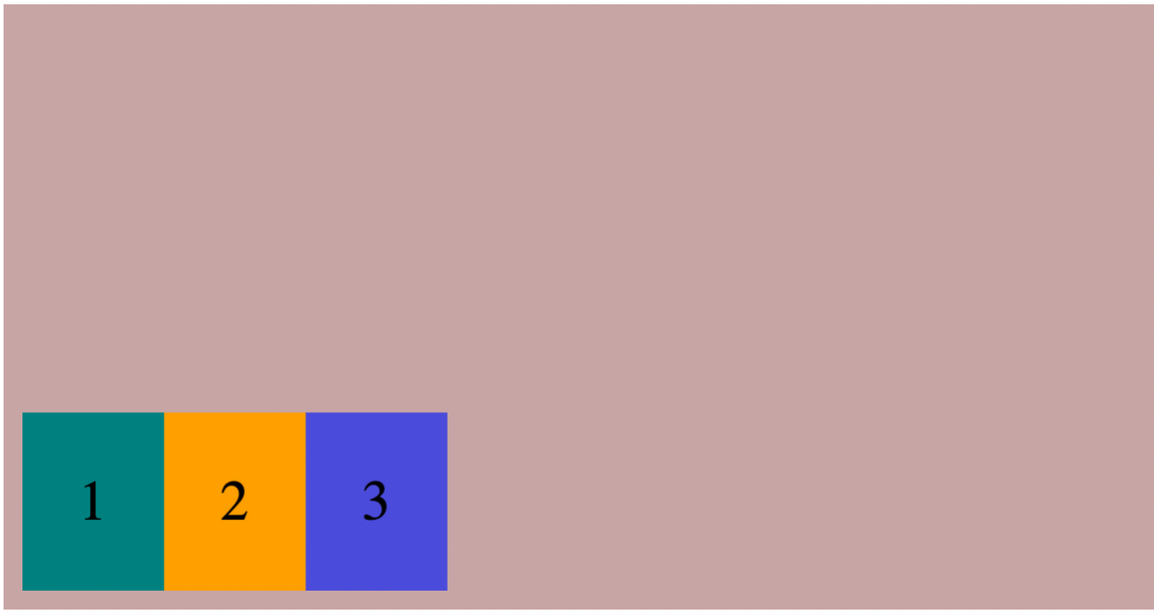
Live Code : Codepen

align-items: `flex-end`

- This value pushes the items to the end of the cross axis:

```
<style> .box { height: 300px; display: flex; align-items: flex-end; } </style
> <div class="box"> <div>1</div> <div>2</div> <div>3</div> </div>
```

Output:



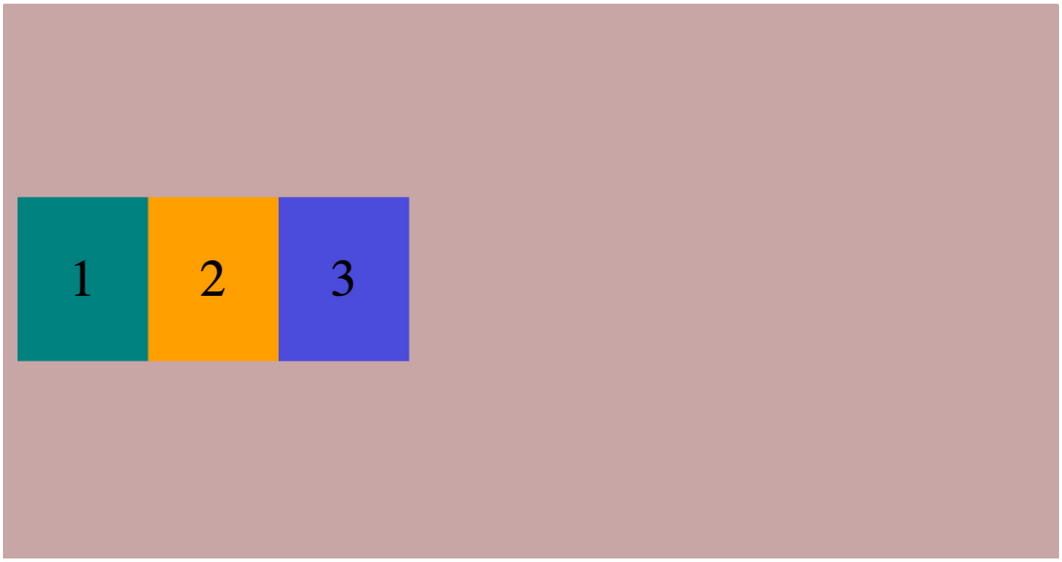
Live Code : Codepen

align-items: `center`

- This centers the content along the cross axis:

```
<style> .box { height: 300px; display: flex; align-items: center; } </style>
<div class="box"> <div class="box-item">1</div> <div class="box-item">2</div>
<div class="box-item">3</div> <div class="box-item">4</div> </div>
```

Output:



Live Code : Codepen

Align-items Overview

align-content

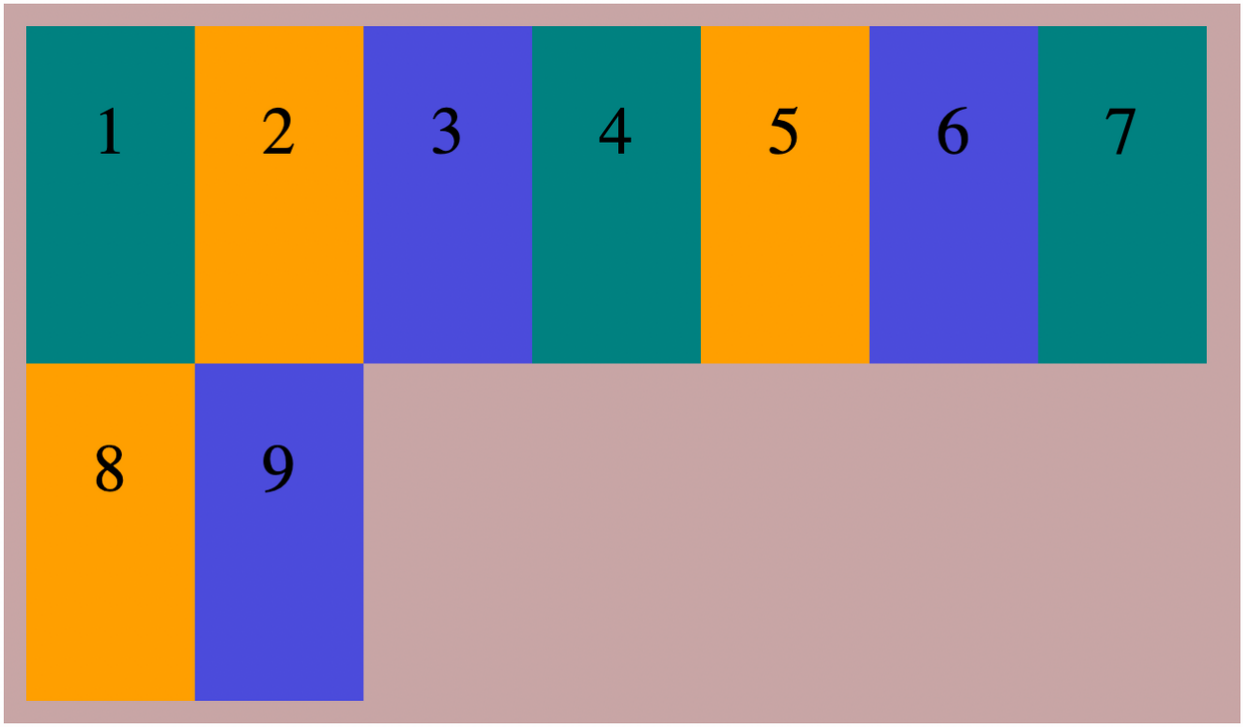
- `align-content` is similar to `justify-content` with the difference being this will align along the cross axis instead of the main axis. Also, `align-content` works only when there are multiple rows of flex items in the container. The container must have a height and also wrapping.
- `align-content: stretch`
- `align-content: flex-start`
- `align-content: flex-end`
- `align-content: center`
- `align-content: space-between`
- `align-content: space-around`

align-content: `stretch`

- This is the `align-content` default value. It has no effect on alignment.

```
<style> .box { height: 300px; display: flex; flex-wrap: wrap; align-content: stretch; } </style> <div class="box"> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> <div>7</div> <div>8</div> <div>9</div > </div>
```

Output:



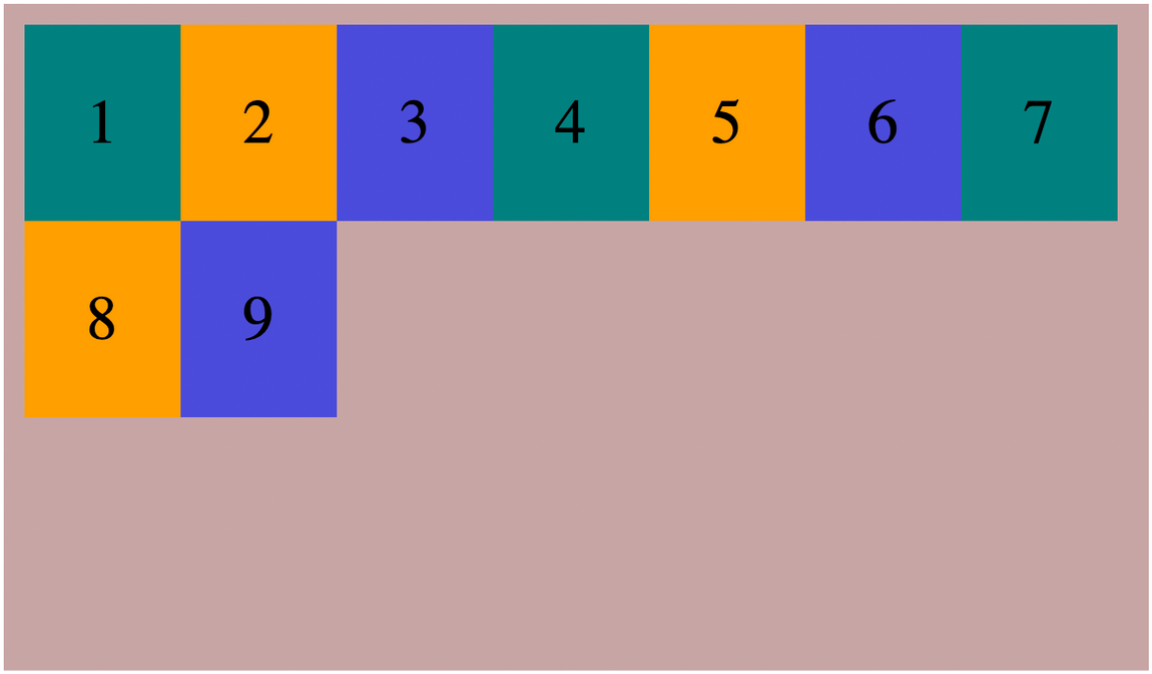
Live Code : Codepen

align-content: `flex-start`

- This value pulls the lines to the beginning of the cross axis:

```
<style> .box { height: 300px; display: flex; flex-wrap: wrap; align-content: flex-start; } </style> <div class="box"> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> <div>7</div> <div>8</div> <div>9 </div> </div>
```

Output:



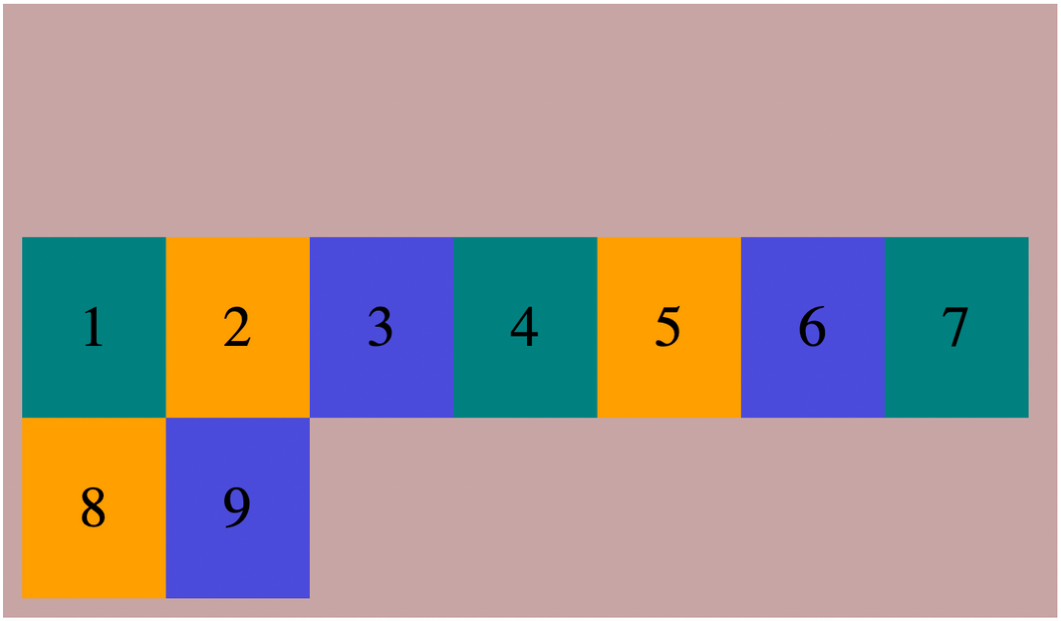
Live Code : Codepen

align-content: `flex-end`

- This value pushes the lines to the end of the cross axis:

```
<style> .box { height: 300px; display: flex; flex-wrap: wrap; align-content: flex-end; } </style> <div class="box"> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> <div>7</div> <div>8</div> <div>9</div> </div>
```

Output:



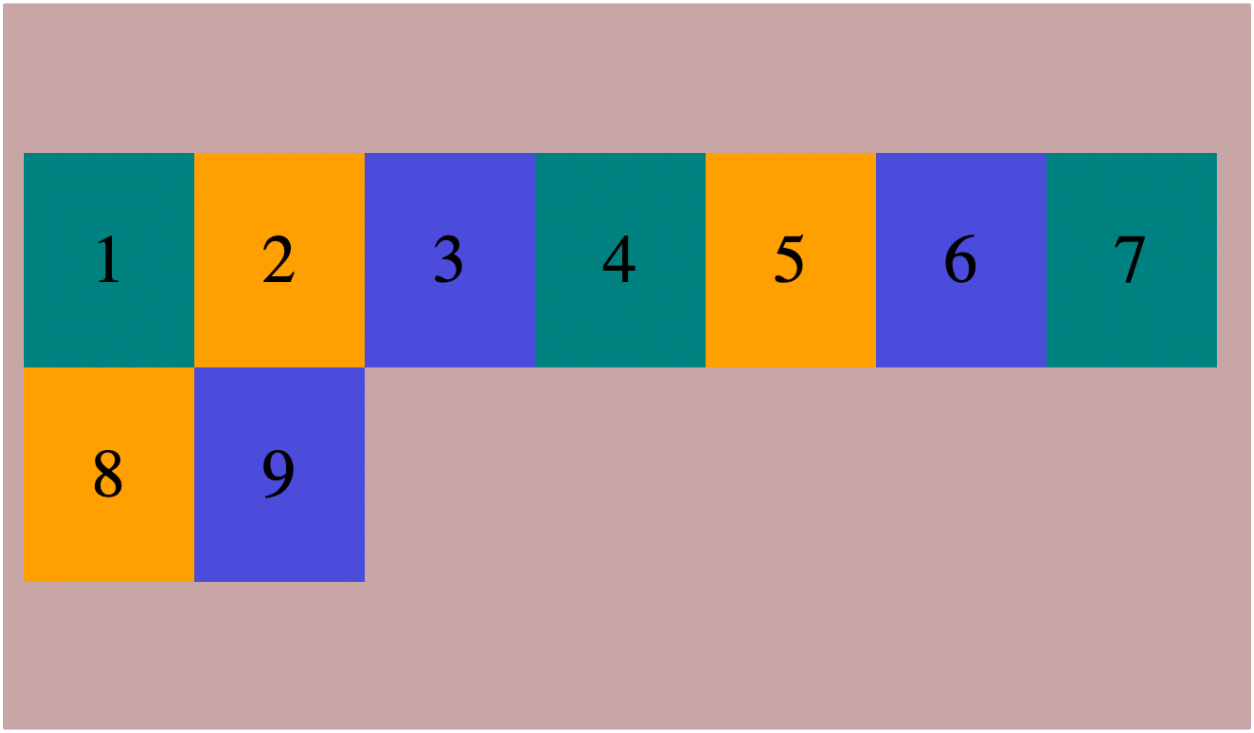
Live Code : Codepen

align-content: `center`

- This value pushes the lines to the center of the cross axis:

```
<style> .box { height: 300px; display: flex; flex-wrap: wrap; align-content: center; } </style> <div class="box"> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> <div>7</div> <div>8</div> <div>9</div > </div>
```

Output:



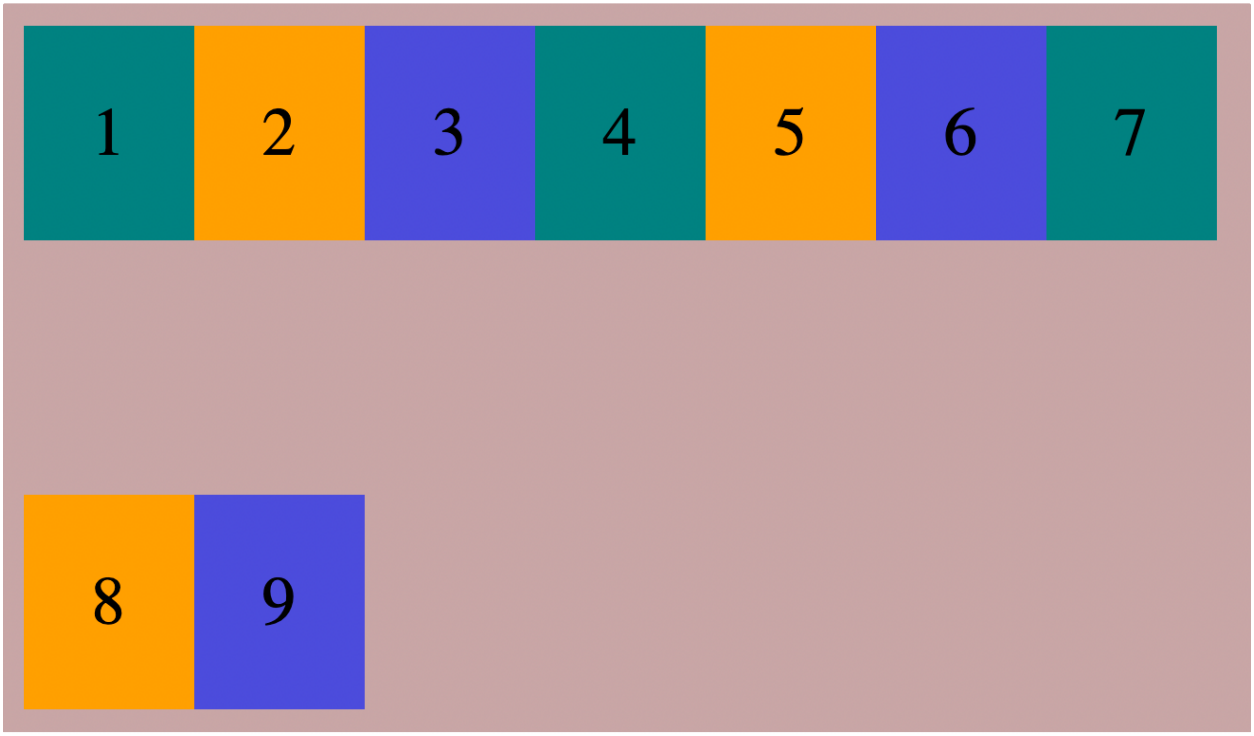
Live Code : Codepen

align-content: `space-between`

- This value takes all the extra space and puts it in between the lines:

```
<style> <style> .box { height: 300px; display: flex; flex-wrap: wrap; align-c
ontent: space-between; } </style> <div class="box"> <div>1</div> <div>2</div>
<div>3</div> <div>4</div> <div>5</div> <div>6</div> <div>7</div> <div>8</div
> <div>9</div> </div>
```

Output:



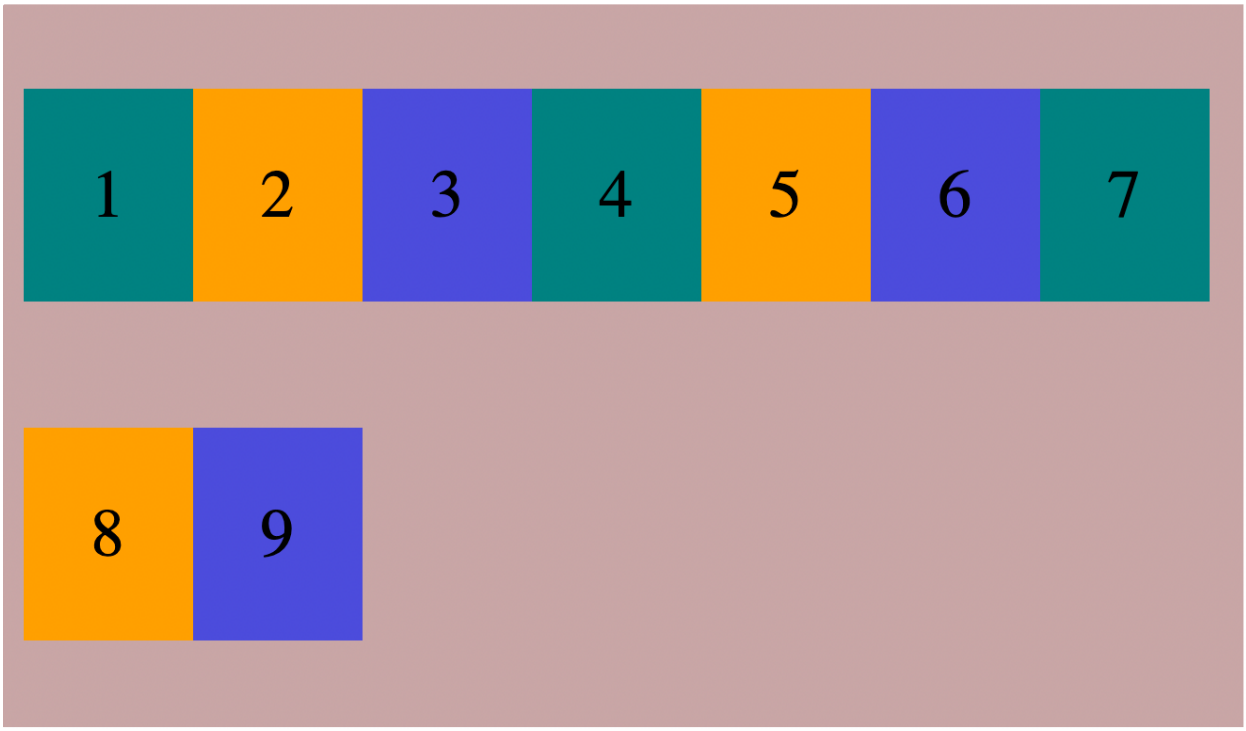
Live Code : Codepen

align-content: `space-around`

- This distributes the space around the lines:

```
<style> .box { height: 300px; display: flex; flex-wrap: wrap; align-content: space-around; } </style> <div class="box"> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> <div>7</div> <div>8</div> <div>9</div> </div>
```

Output:



Live Code: Codepen

Child Properties

- `order`
- `flex-grow`
- `flex-shrink`
- `flex-basis`

`order`

- The **order** property of CSS can be used for ordering flex items. It specifies the order of a flex item with respect to the other flex items. The element has to be a flexible item for the order property to work. The elements are displayed in ascending order of their order values. If two elements have the same order value then they are displayed on the basis of their occurrence in the source code.

```
<style> .box { display: flex; } .item-1 { order: 1; } .item-2 { order: 8; }
</style> <div class="box"> <div class="item-1">1</div> <div class="item-2">2
</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> <div>7</div> <div
>8</div> <div>9</div> </div>
```

Output:

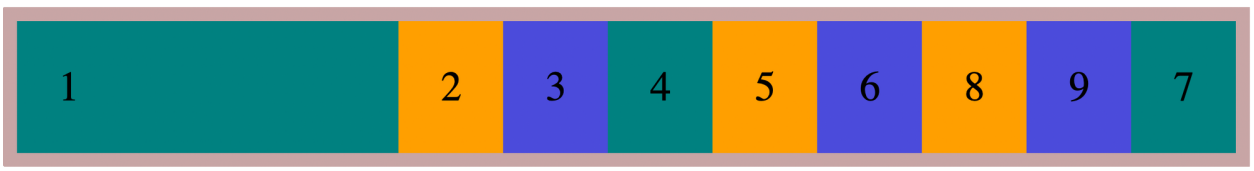
Live Code : Codepen

`flex-grow`

- `flex-grow` allows a flex item to grow if necessary. This property specifies what amount of remaining space inside the flex container the item should take up. All flex items have a `flex-grow` of zero.

```
<style> .box { display: flex; } .item-1 { flex-grow: 1; } </style> <div class
="box"> <div class="item-1">1</div> <div>2</div> <div>3</div> <div>4</div> <
div>5</div> <div>6</div> <div>7</div> <div>8</div> <div>9</div> </div>
```

Output:



Live Code : Codepen

`flex-shrink`

- `flex-shrink` defines the capacity for a flex item to shrink if necessary. The default value of `flex-shrink` is one.

```
<style> .box { display: flex; } .item-1 { flex-shrink: 4; } </style> <div cla
ss="box"> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <
div>6</div> <div>7</div> <div>8</div> <div>9</div> </div>
```

Output:

`flex-basis`

- `flex-basis` specifies the initial main size of a flex item before the extra space in the