

Flexbox

CSS Flexbox Layout Module



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>

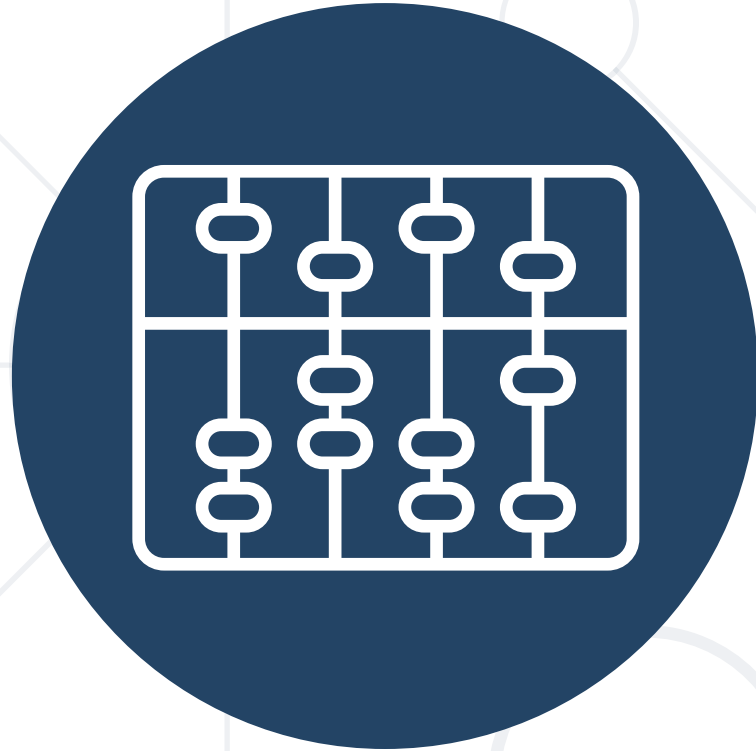
Table of Contents

1. Flexbox
2. Flexbox Properties for the **Parent**
3. Flexbox Properties for the **Children**



sli.do

#html-css



CSS Flexbox Layout Module

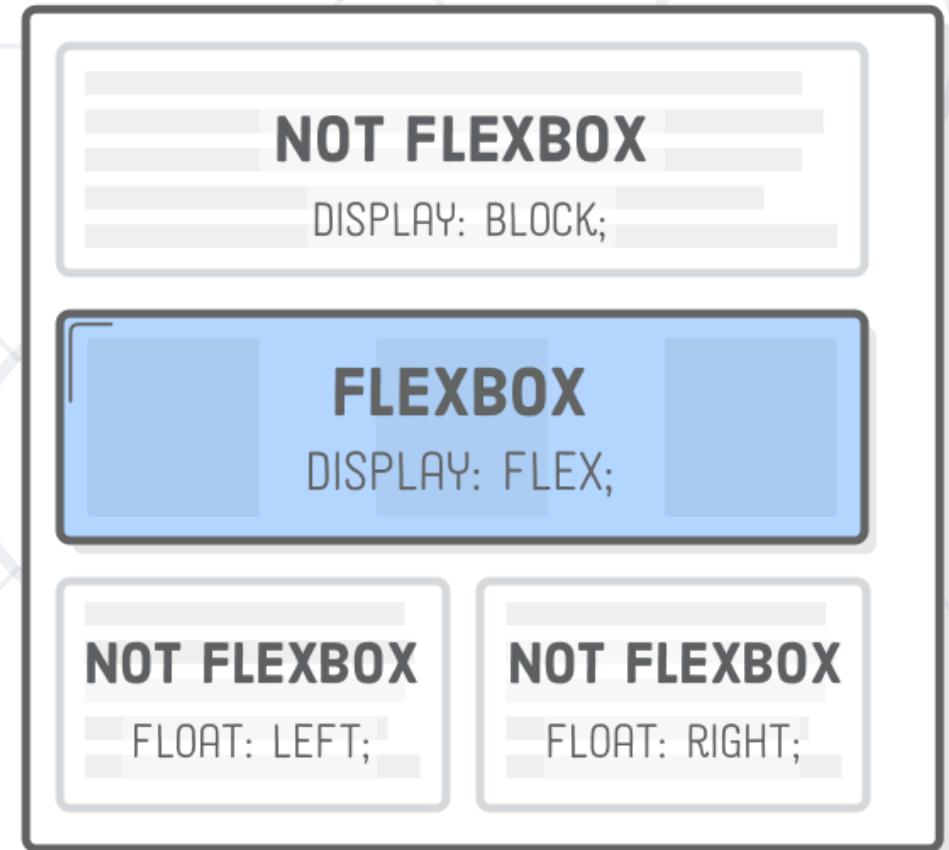
What is Flexbox?

- Offers **space distribution** between items in an interface and powerful **alignment** capabilities
- Flexbox is a method for laying out items in **rows** or **columns**
- Items flex to **fill** additional space and **shrink** to fit into smaller spaces

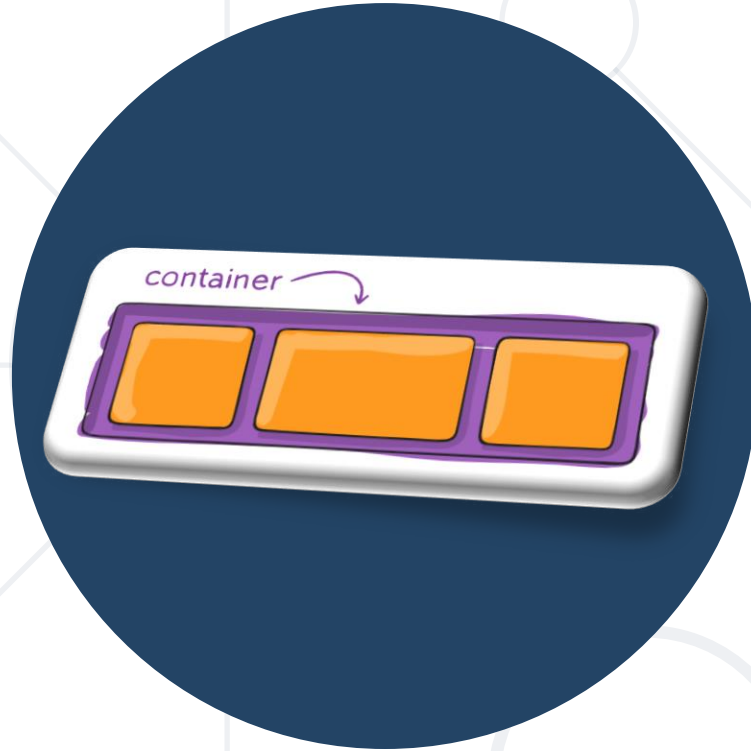


Why Flexbox?

- For a long time, the only reliable cross browser-compatible tools available for creating CSS layouts were **floats** and **positioning**
- These are fine and they work, but in some ways, they are also rather limiting and frustrating



- The following simple layout requirements are either **difficult** or **impossible** to achieve with such tools:
 - **Vertically centering** a block of content inside its parent
 - Making all the children of a container take up an **equal** amount of the available **width/height**
 - Making all columns in a multiple column layout adopt the **same height** even if they contain a different amount of content

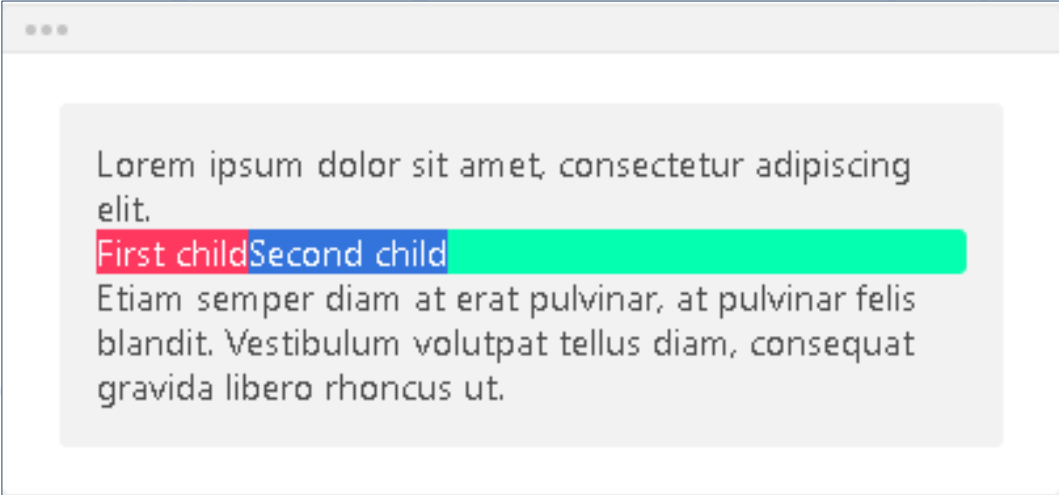


**Expands Items to Fill Available Free Space or
Shrinks Them to Prevent Overflow**

- The element is turned into a **flexbox** container
- Its child elements will be turned into **flexbox items**

```
<body>
  <p>Lorem ipsum dolor sit amet, con
sectetur adipiscing elit.</p>
  <div class="container">
    <p>First child</p>
    <p>Second child</p>
  </div>
  <p>
    Etiam semper diam at erat pulv
inar, at pulvinar felis blandit. Vesti
bulum volutpat tellus diam,consequat g
ravidam libero rhoncus ut.
  </p>
</body>
```

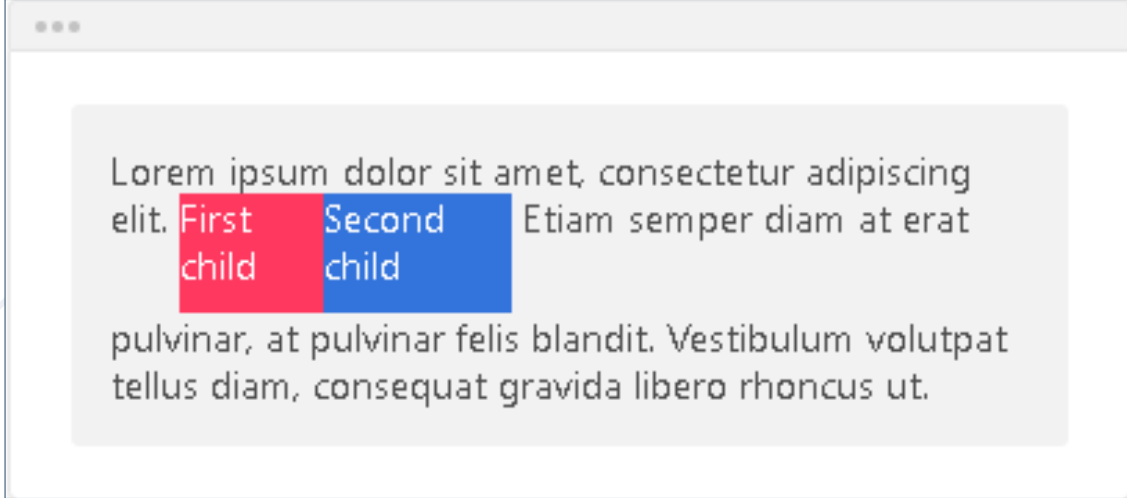
```
.container {
  display: flex;
}
```



...
Lorem ipsum dolor sit amet, consectetur adipiscing
elit.
First childSecond child
Etiam semper diam at erat pulvinar, at pulvinar felis
blandit. Vestibulum volutpat tellus diam, consequat
gravidam libero rhoncus ut.

- The element shares properties of both an **inline** and a **flexbox** element:
 - **inline** because the element behaves like simple text, and inserts itself in a block of text
 - **flexbox** because its child element will be turned into flexbox items

```
.container {  
  display: inline-flex;  
  height: 3em;  
  width: 120px;  
}
```

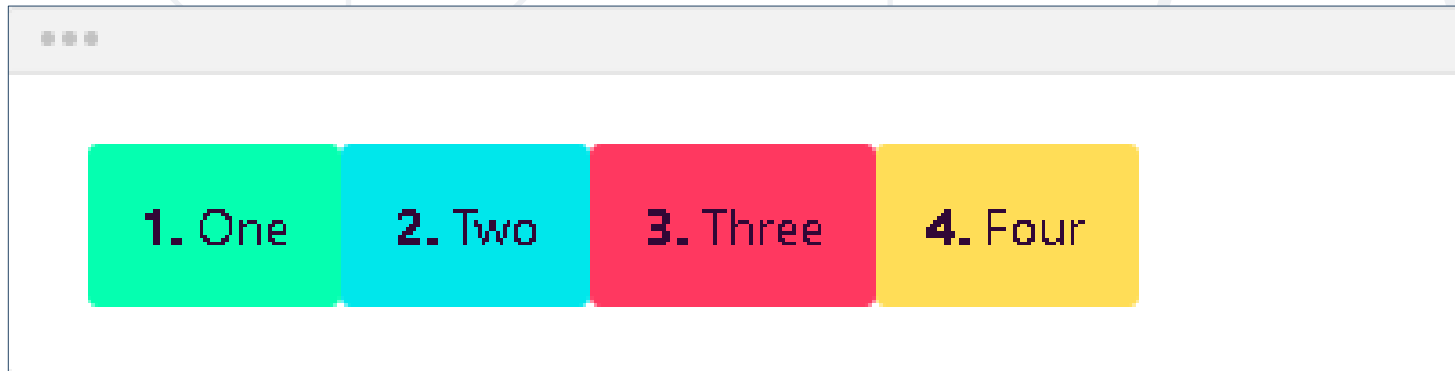


Lorem ipsum dolor sit amet, consectetur adipiscing elit. **First child** **Second child** Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

- Defines how flexbox items are ordered within a flexbox container

```
flex-direction: row;
```

- The flexbox items are ordered the **same way** as the text direction, along the main axis

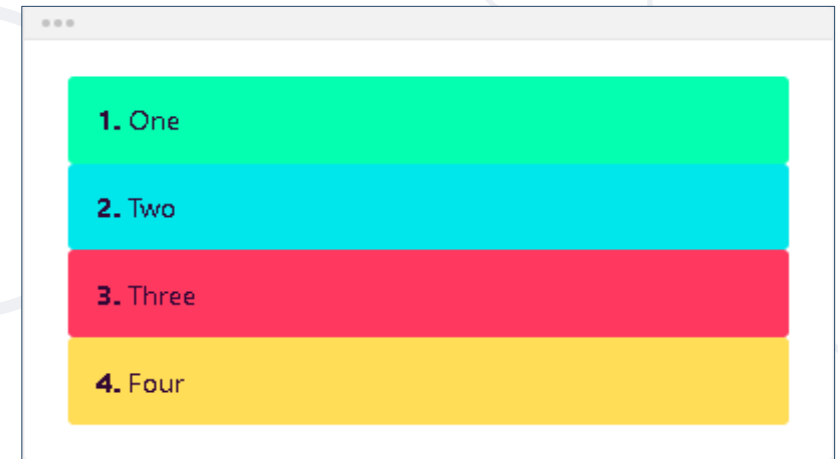


- The flexbox items are ordered the **opposite** way as the **text direction**, along the main axis

```
flex-direction: row-reverse;
```

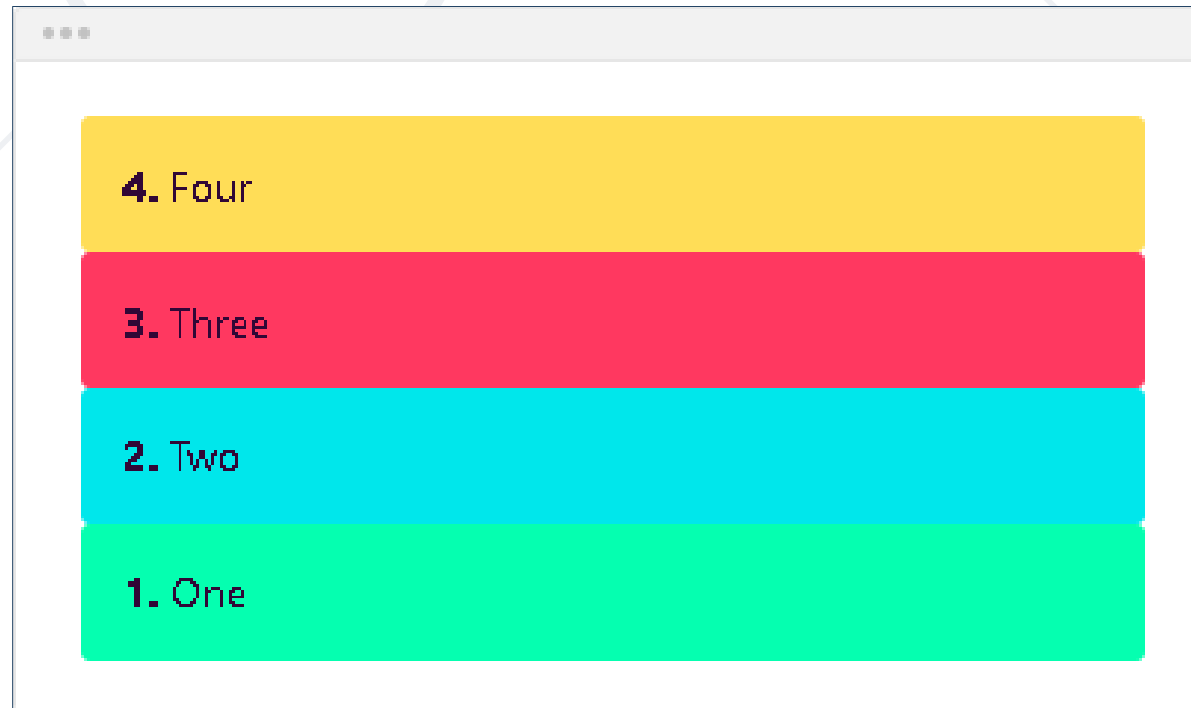
- The flexbox items are ordered the **same** way as the **text direction**, along the **cross axis**

```
flex-direction: column;
```



- The flexbox items are ordered the **opposite** way as the **text direction**, along the **cross axis**

```
flex-direction: column-reverse;
```



- Defines if flexbox items appear on a **single line** or on **multiple lines** within a flexbox container

```
flex-wrap: nowrap;
```

- The flexbox items will remain on a **single line**, no matter what, and will eventually overflow if needed



- The flexbox items will be distributed among **multiple lines** if needed

```
flex-wrap: wrap;
```



- The flexbox items will be distributed among **multiple lines** if needed

- Any additional line will appear **before** the previous one

```
flex-wrap: wrap-reverse;
```



- **flex-flow** is a shorthand for the **flex-direction** and **flex-wrap** properties
- The default value is **row nowrap**

```
flex-flow: <flex-direction> || <flex-wrap>  
  
.container {  
  flex-flow: row wrap;  
}
```


- Defines how **flexbox/grid** items are aligned according to the **main** axis, within a flexbox container

```
justify-content: flex-start;
```

- The flexbox items are pushed towards the **start** of the container's main axis



- The flexbox items are pushed towards the **end** of the container's main axis

```
justify-content: flex-end;
```



- The flexbox items are **centered** along the container's main axis

```
justify-content: center;
```



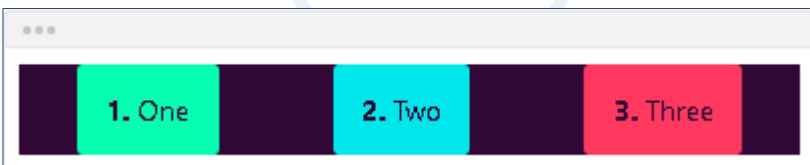
- The remaining space is distributed **between** the flexbox items

```
justify-content: space-between;
```



- The remaining space is distributed **around** the flexbox items: this adds space **before** the first item and **after** the last one

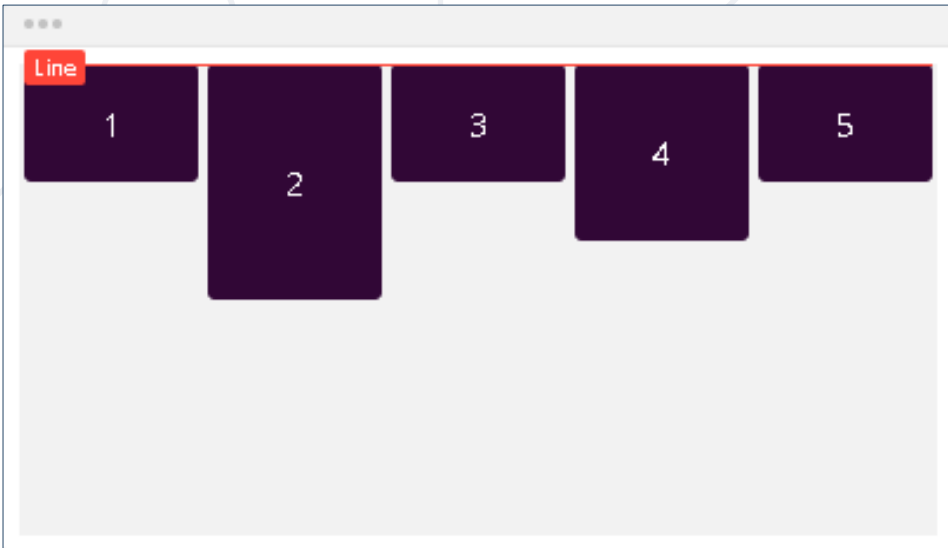
```
justify-content: space-around;
```



- Defines how flexbox items are aligned according to the **cross** axis, within a line of a flexbox container

```
align-items: flex-start;
```

- The flexbox items are aligned at the **start** of the **cross axis**

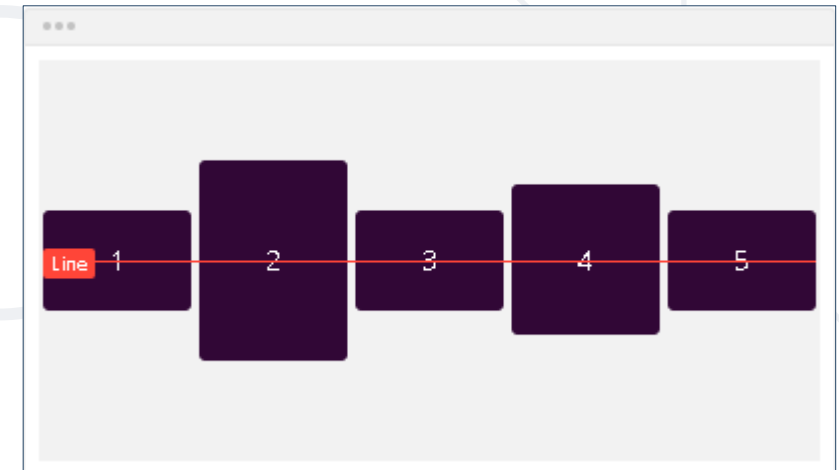
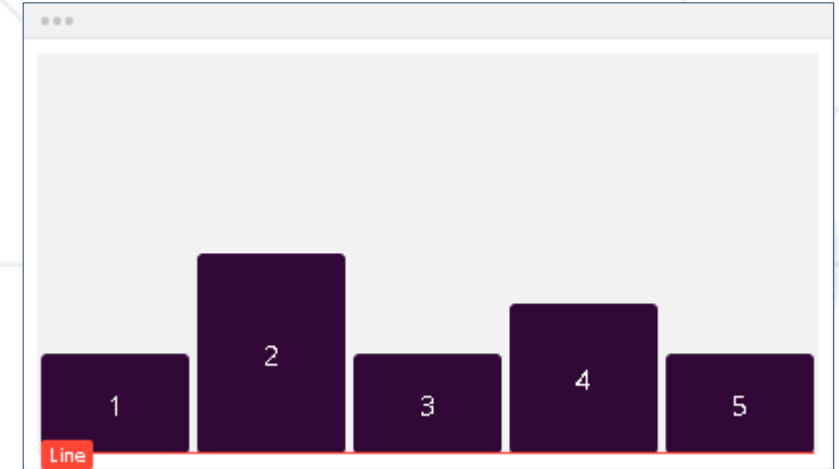


- The flexbox items are aligned at the **end** of the **cross axis**

```
align-items: flex-end;
```

- The flexbox items are aligned at the **center** of the **cross axis**

```
align-items: center;
```

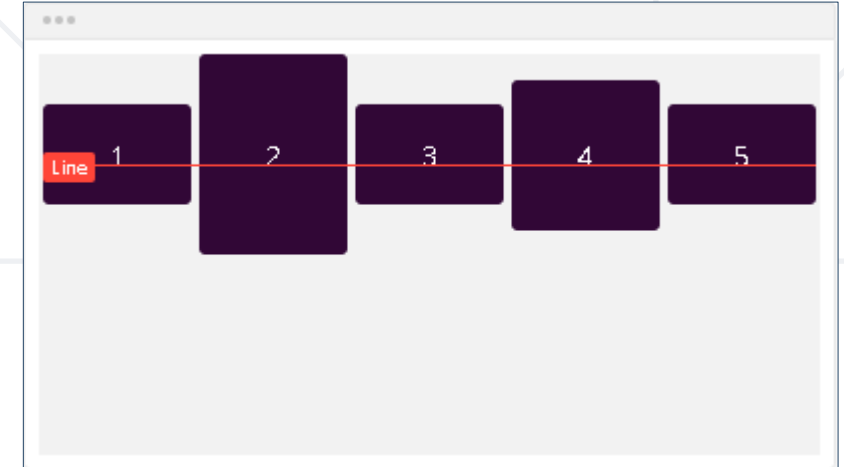


- The flexbox items are aligned at the **baseline** of the **cross axis**

```
align-items: baseline;
```

- The flexbox items will stretch across the whole **cross axis**

```
align-items: stretch;
```



- Defines how each line is **aligned** within a flexbox container
- It only applies if **flex-wrap: wrap** is present, and if there are multiple lines of flexbox items

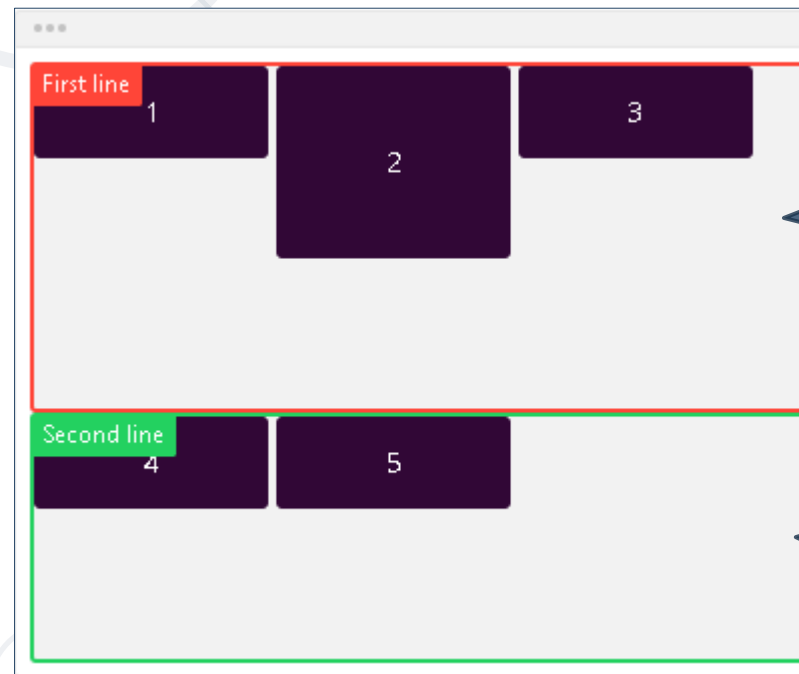
```
align-content: stretch;
```

- Each line will stretch to **fill** the remaining space

Align Content: Stretch Example

- The first line is **100px** high
- The second line is **50px** high
- The remaining space is **150px** and it is distributed equally amongst the two lines

The container is 300px high
All boxes are 50px high
The second box is 100px high

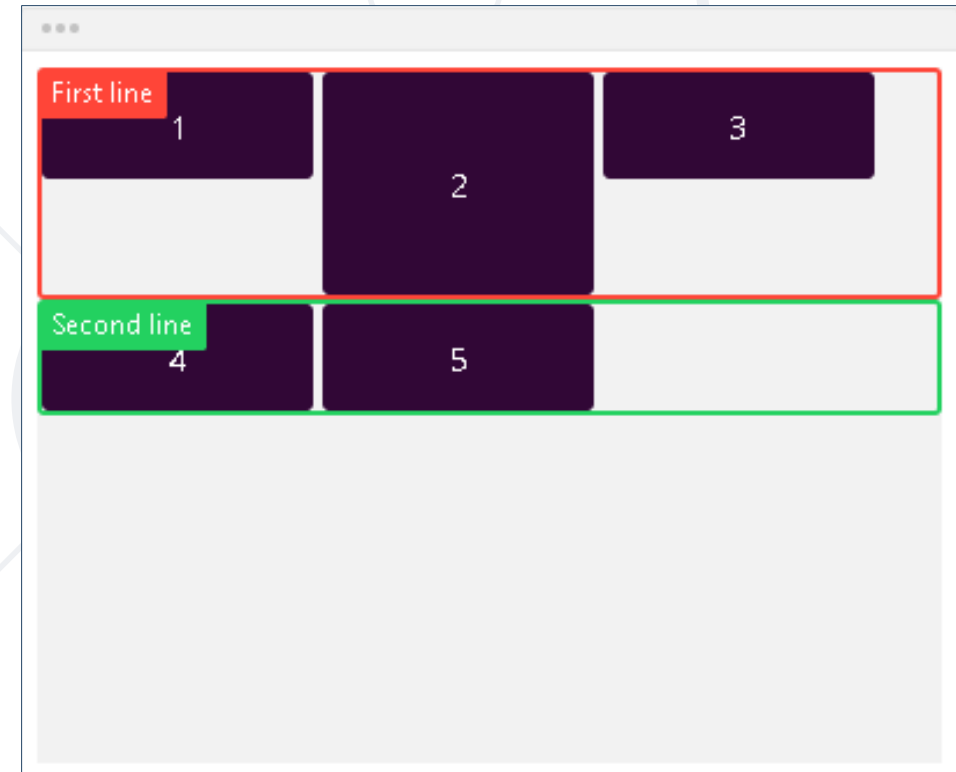


The first line is
175px high

The second line is
125px high

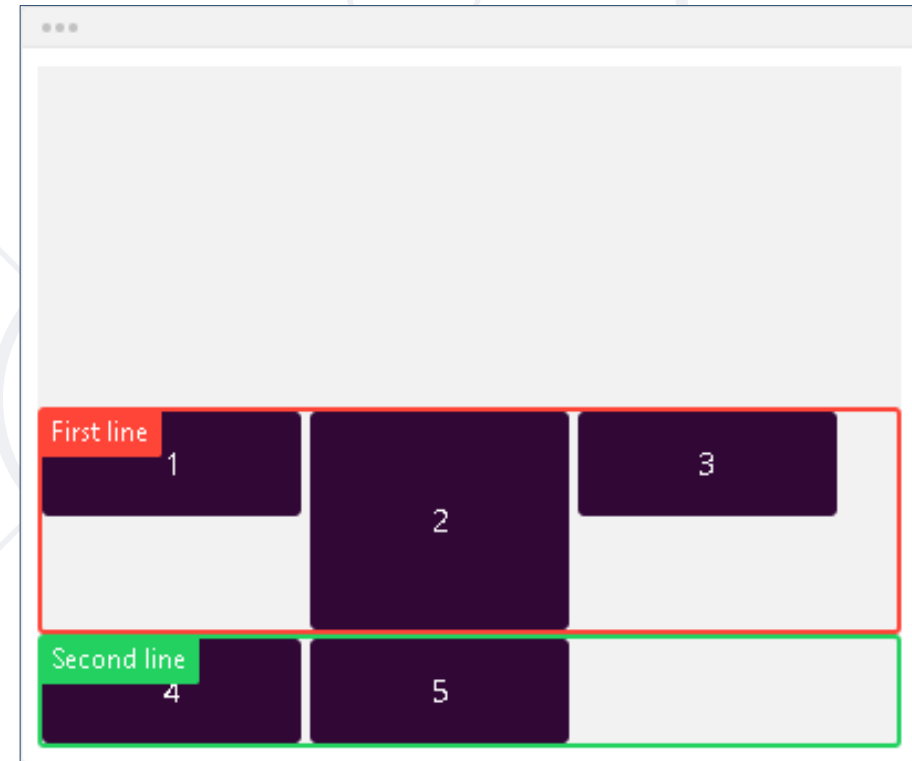
- Each line will only fill the space it **needs**
- They will all move towards the **start** of the flexbox container's cross axis

```
align-content: flex-start;
```



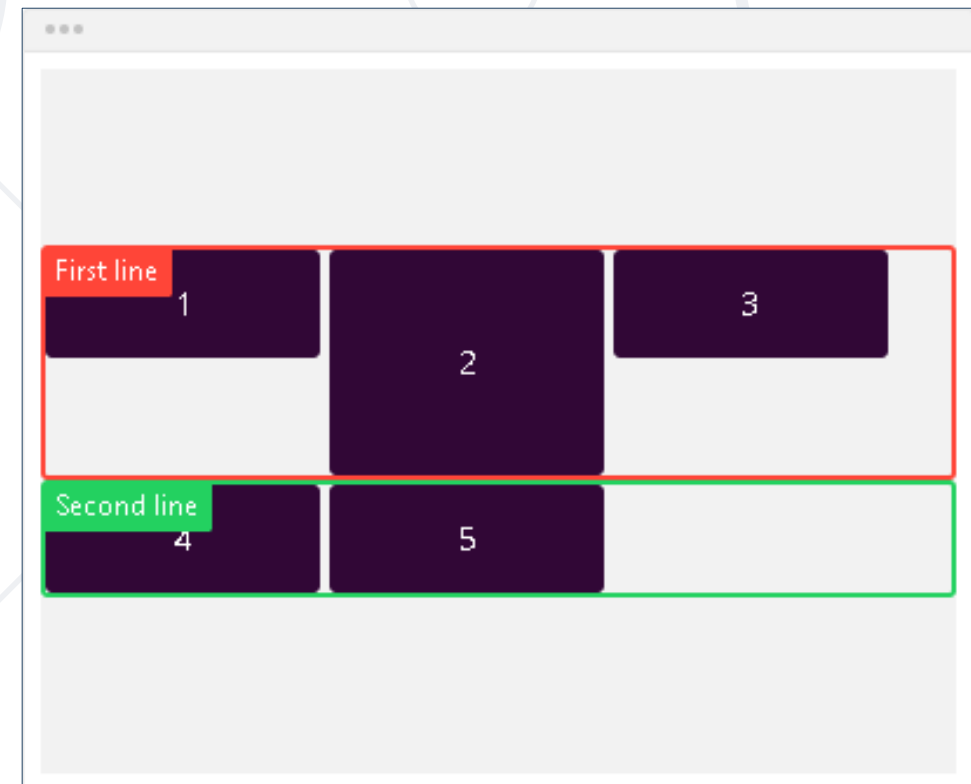
- Each line will only fill the space it **needs**
- They will all move towards the **end** of the flexbox container's cross axis

```
align-content: flex-end;
```



- Each line will only fill the space it **needs**
- They will all move towards the **center** of the flexbox container's cross axis

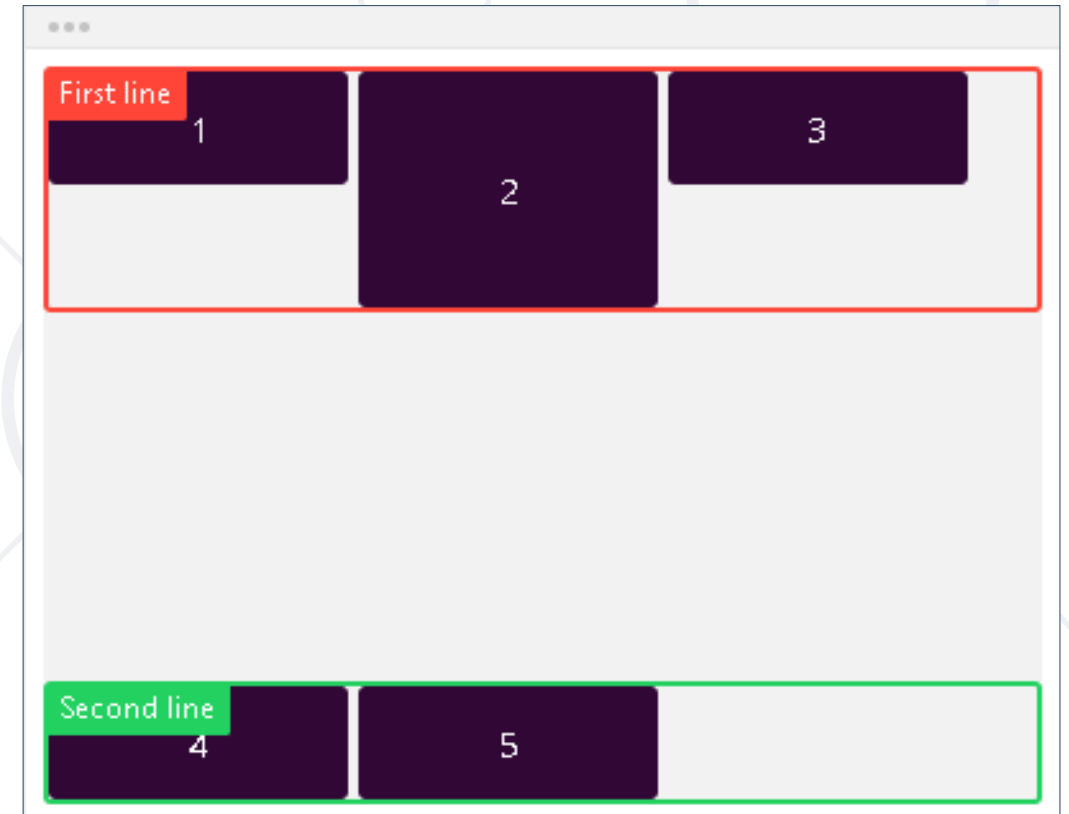
```
align-content: center;
```



Align Content

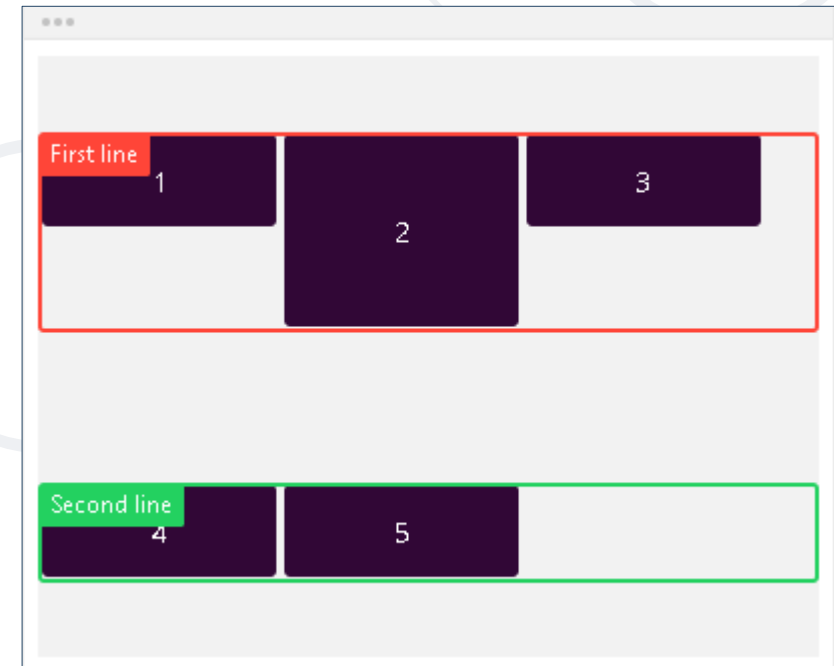
- Each line will only fill the space it **needs**
- The **remaining** space will appear **between** the lines

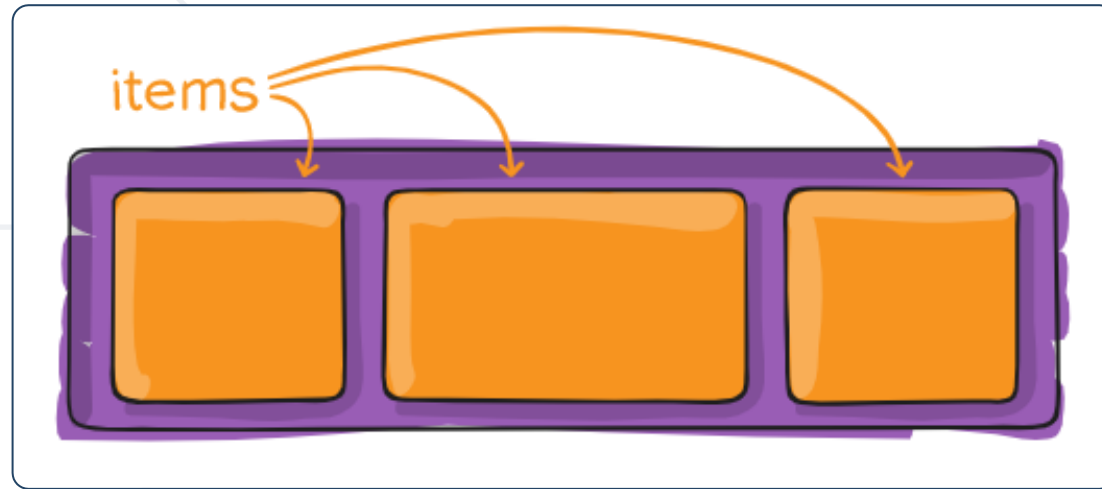
```
align-content: space-between;
```



- Each line will only fill the space it **needs**
- The **remaining** space will be distributed equally **around** the lines: before the first line, between the two, and after the last one

```
align-content: space-around;
```

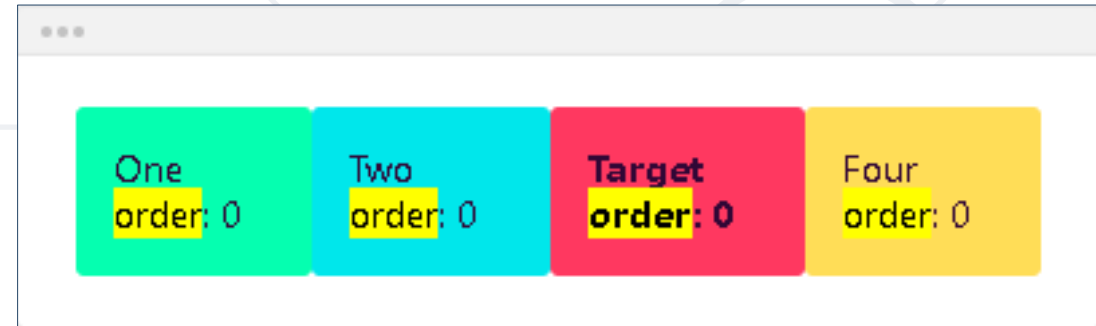




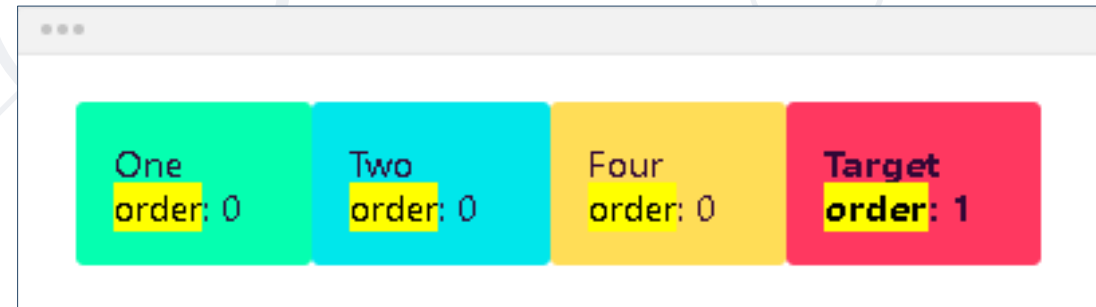
Expands items to Fill Available Free Space or Shrinks Them to Prevent Overflow

- Order - defines the order of a flexbox item
 - The order of the flexbox items is the one defined in the **HTML code**
- The order is **relative** to the flexbox item's **siblings**
- The final order is defined when all individual flexbox item order values are considered

order: 0;

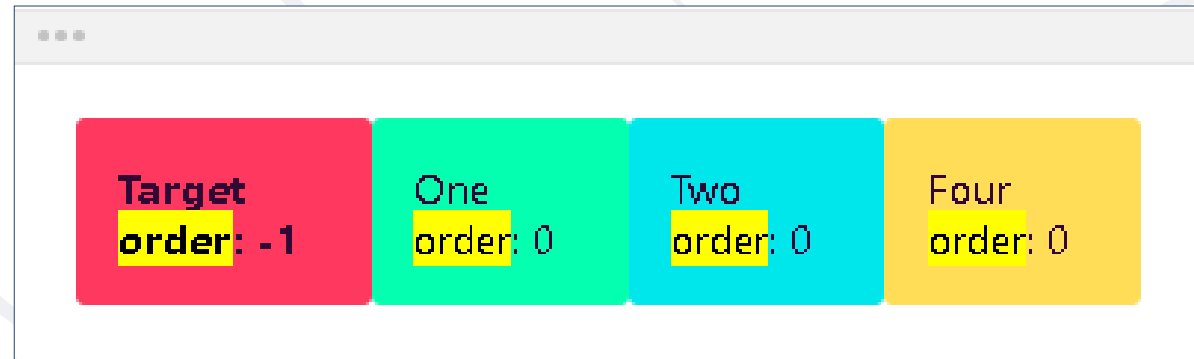


order: 1;



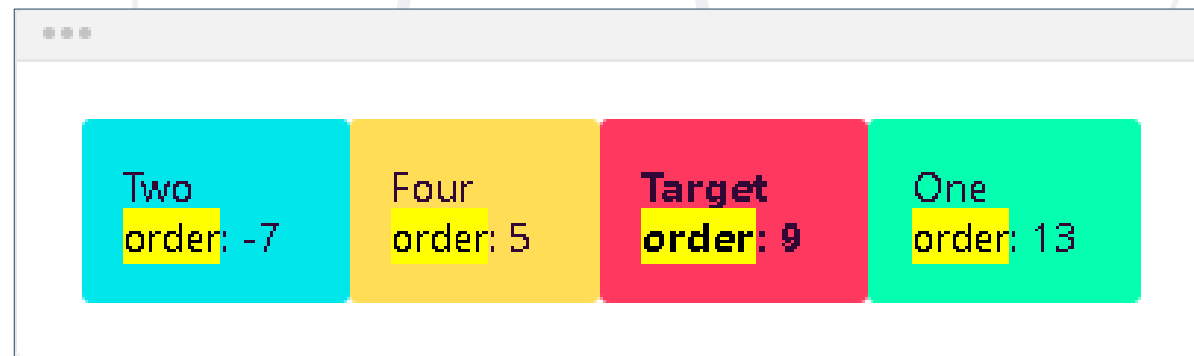
- You can use **negative** values

```
order: -1;
```



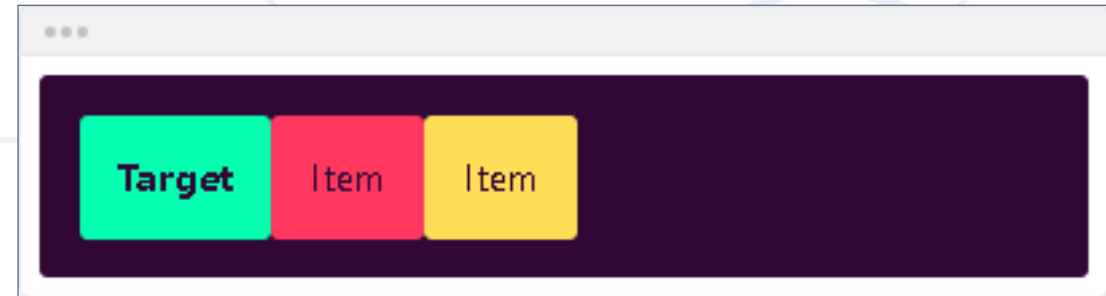
- You can set a **different** value for each flexbox item

```
order: 9;
```

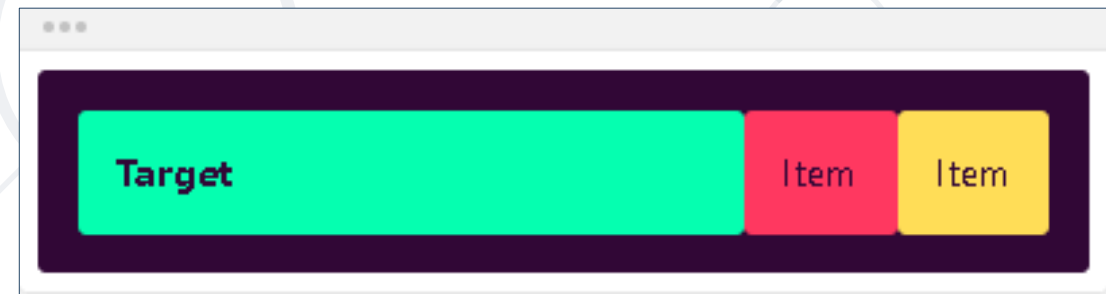


- Defines how much a flexbox item should **grow** if there's space available
 - The element will **NOT** grow if there's space available
 - It will only use the space it needs
- The element will **grow** by a factor of 1
- It will fill up the remaining space if no other flexbox item has a **flex-grow** value

```
flex-grow: 0;
```



```
flex-grow: 1;
```



- Defines how much a flexbox item should **shrink** if there's **NOT enough** space available

```
flex-shrink: 1;
```

- If there's **NOT enough** space available in the container's main axis, the element will **shrink** by a factor of **1**, and will wrap its content



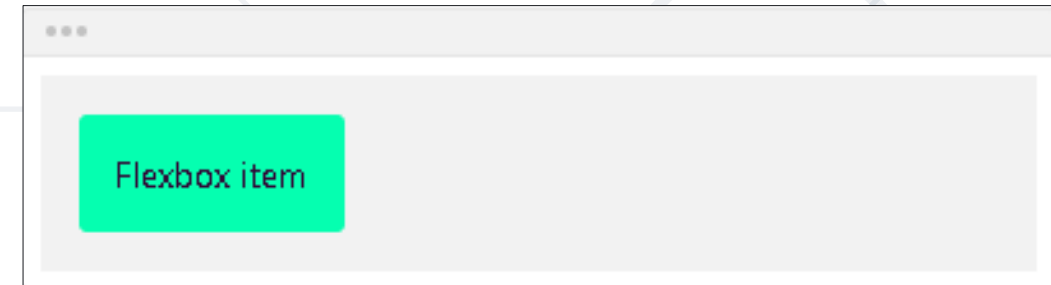
- The element will **NOT** shrink it will retain the width it needs, and **NOT** wrap its content
- Its siblings will shrink to give space to the target element.
 - Because the target element will **NOT** wrap its content, there is a chance for the flexbox container's content to **overflow**

```
flex-shrink: 0;
```



- Defines the **initial size** of a flexbox item
 - The element will be automatically sized based on its **content**, or on any **height** or **width value** if they are defined
- You can define **pixel** or **(r)em** values
- The element will wrap its content to avoid any overflow

```
flex-basis: auto;
```



```
flex-basis: 80px;
```



- Flex is the shorthand for:
 - **flex-grow**
 - **flex-shrink**
 - **flex-basis**
- The default value is **0 1 auto**

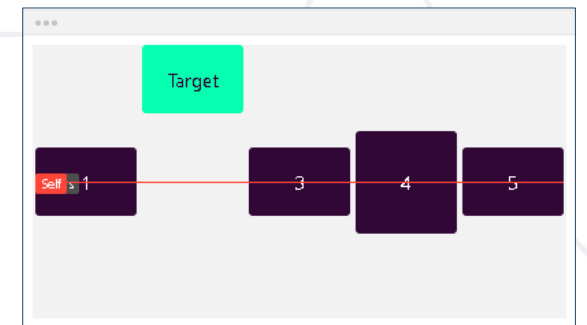
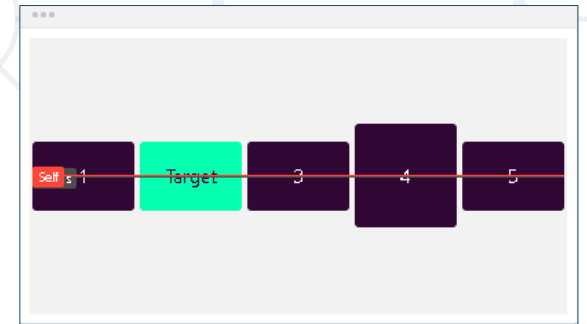
```
.item {  
    flex: <flex-grow> <flex-shrink> <flex-basis>  
}
```

- Works like **align-items**, but applies only to a **single** flexbox item, instead of all of them
 - The target will use the value of **align-items**

```
align-self: auto;
```

- The container has **align-items: center**
- The target has **align-self: flex-start**

```
align-self: flex-start;
```

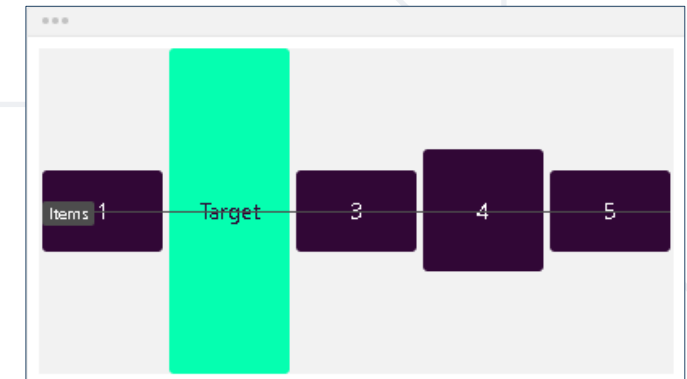
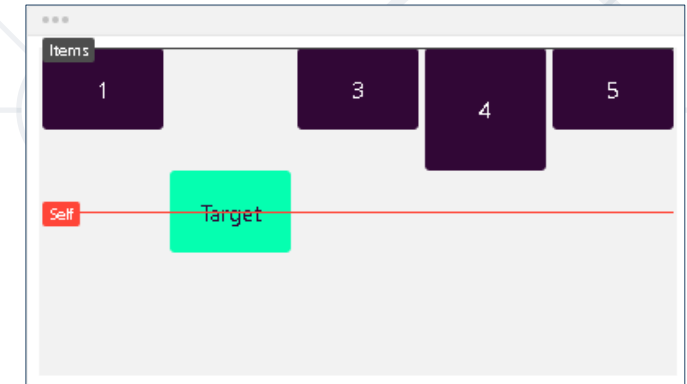


- The container has align-items: **flex-start**
- The target has **align-self: center**

```
align-self: center;
```

- The container has align-items: **center**
- The target has **align-self: stretch**

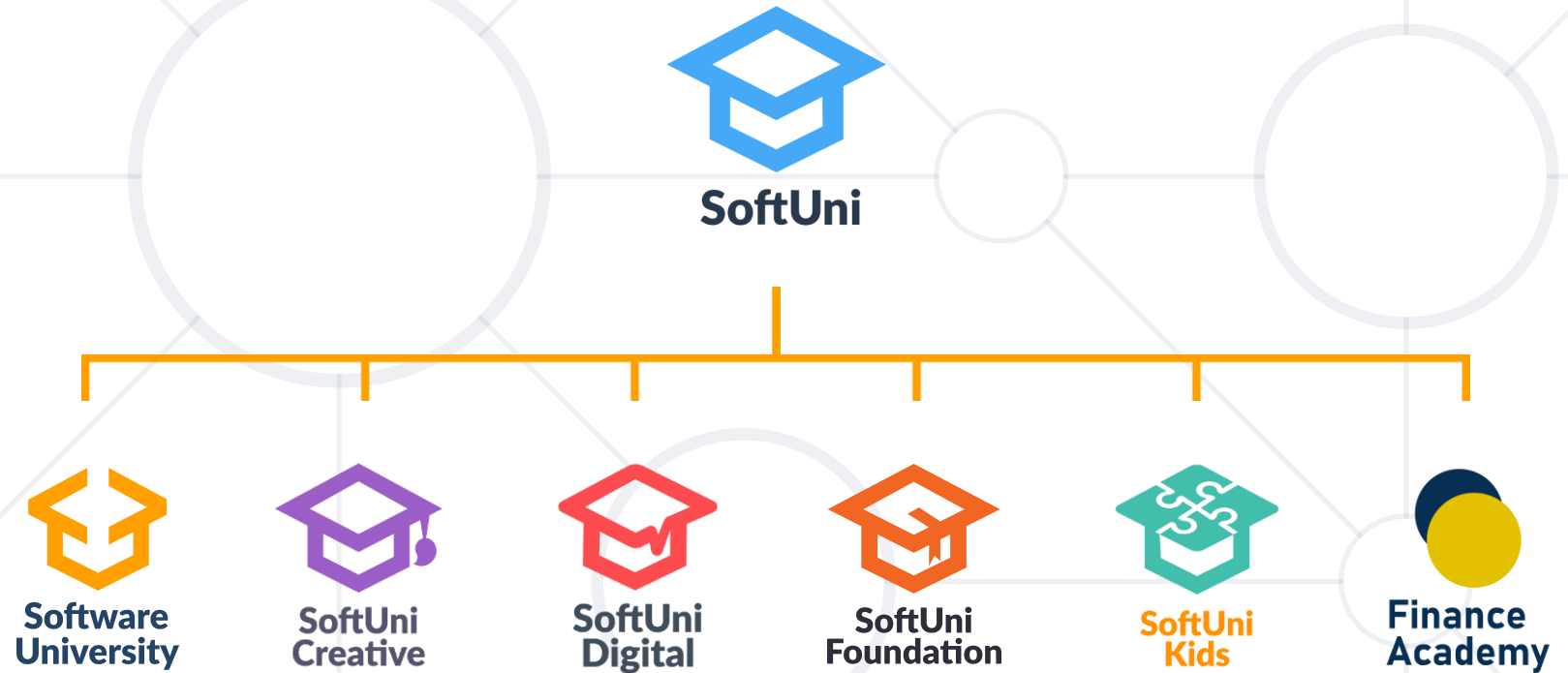
```
align-self: stretch;
```



- What is **Flexbox**?
 - Why **Flexbox**?
- Properties for the Parent: **display**, **direction**, **wrap**, **justify**, **align**
- Properties for the children: **order**, **shrink**, **align**



Questions?



SoftUni Diamond Partners

**SUPER
HOSTING
.BG**



**Coca-Cola HBC
Bulgaria**



POKERSTARS
POKER | CASINO | SPORTS
a Flutter International brand

INDEAVR
Serving the high achievers



AMBITIONED


**DRAFT
KINGS**



**SOFTWARE
GROUP**

createX



Postbank
Решения за твоето утре

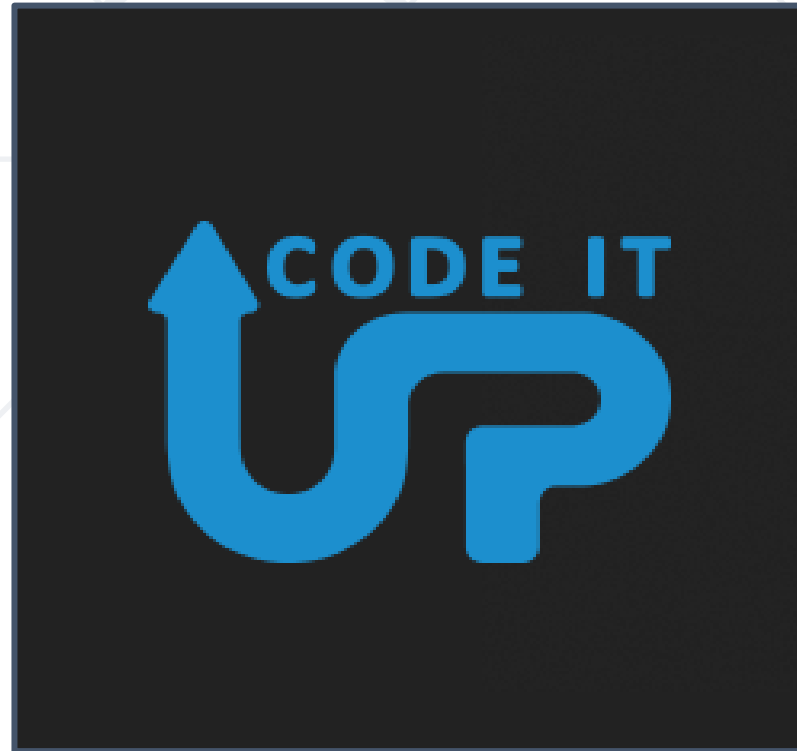


BOSCH

DXC
TECHNOLOGY



SmartIT



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>



- Software University – High-Quality Education, Profession and Job for Software Developers

- softuni.bg, about.softuni.bg

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity

- Software University Forums

- forum.softuni.bg

