Flexbox

CSS Flexbox Layout Module

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

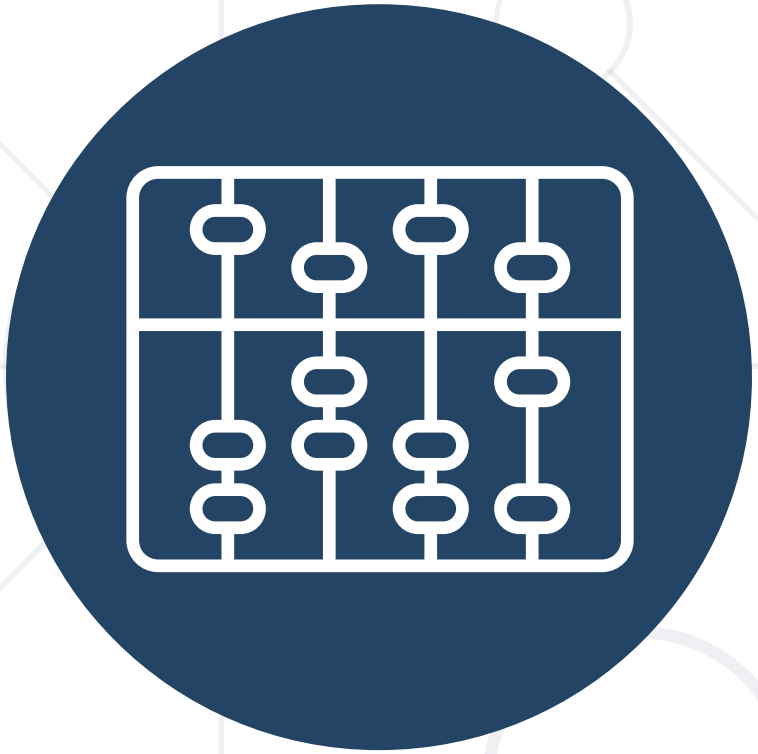
Have a Question?

Software University

sli.do

#html-css

Table of Contents



Flexbox

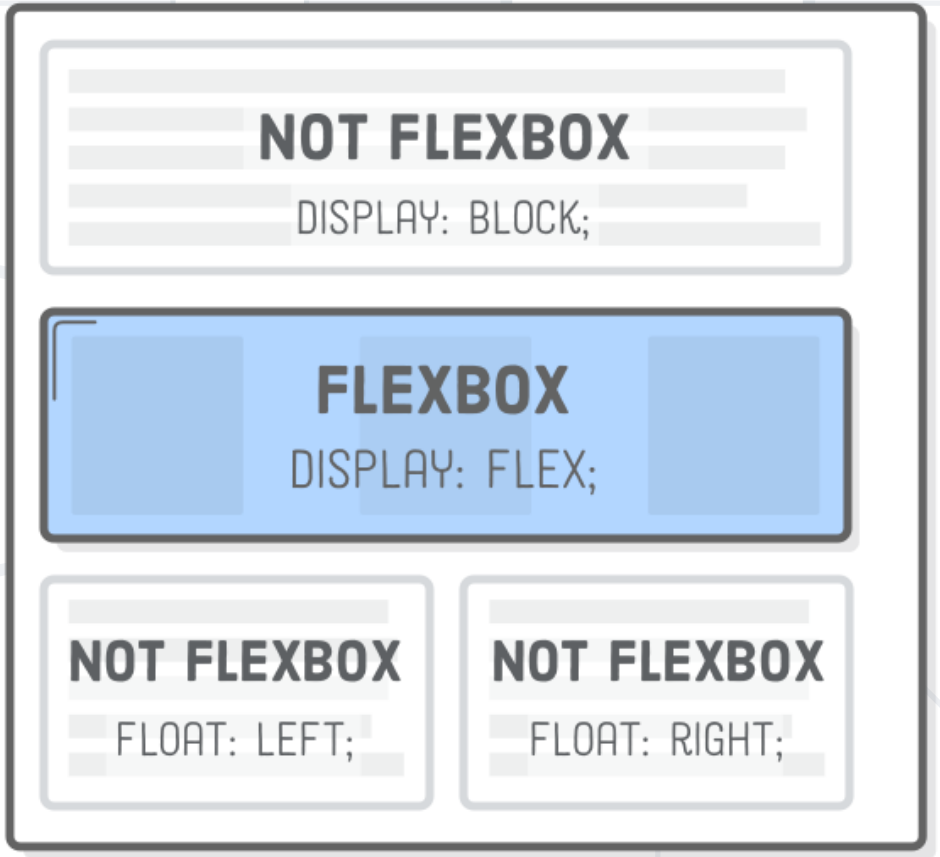
What is Flexbox?

- The Flexible Box Module, usually referred to as flexbox, was designed as a one-dimensional layout model, and as a method that could offer space distribution between items in an interface and powerful alignment capabilities
- https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox

Why Flexbox?

Software University

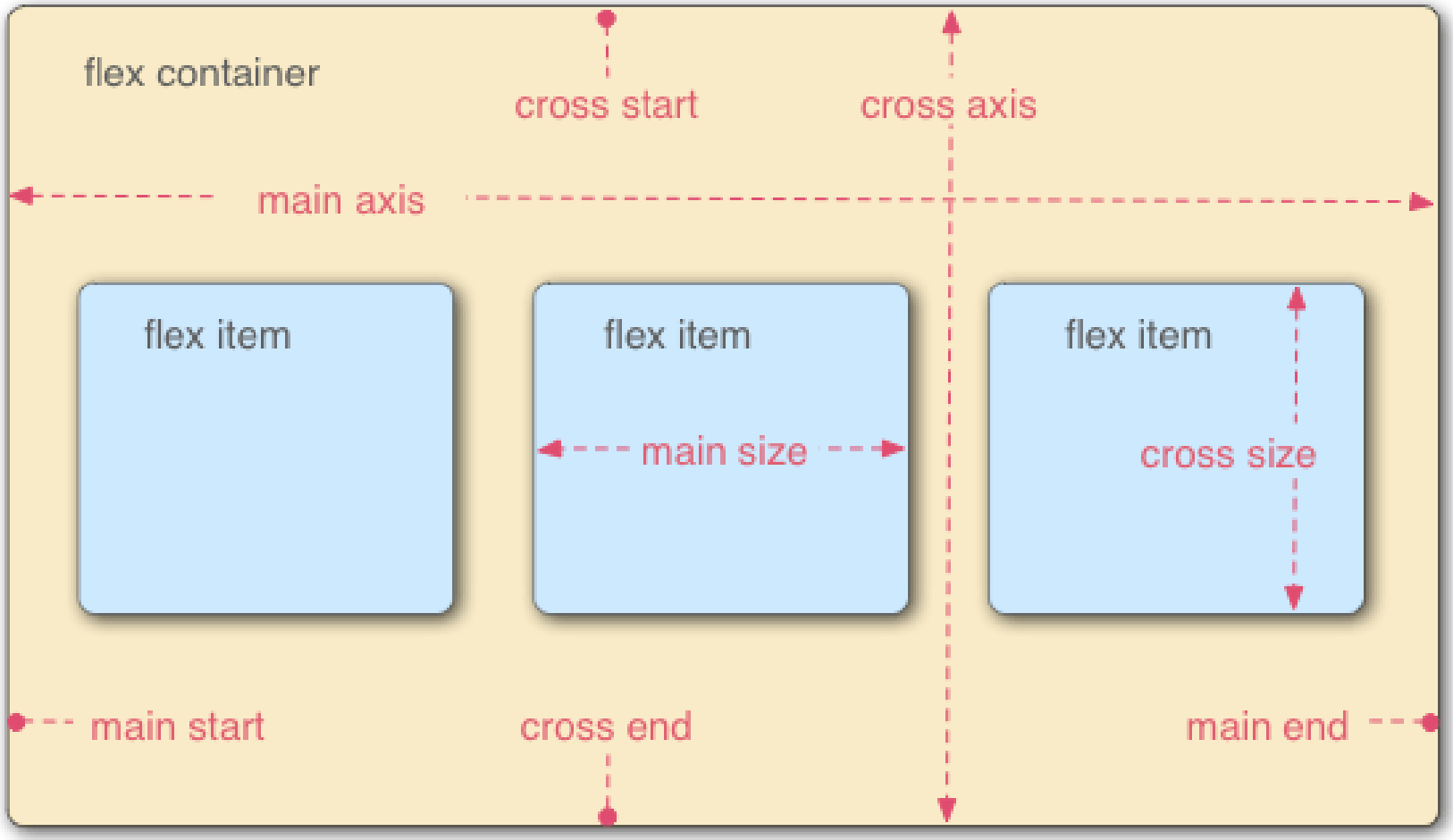
- For a long time, the only reliable cross browser-compatible tools available for creating CSS layouts were things like floats and positioning. These are fine and they work, but in some ways, they are also rather limiting and frustrating
- Flexbox can be look at as a logical system to arrange boxes in 2 dimensions. There are many places where an understanding of the ideas of Flexbox will be very useful



The Flex Model

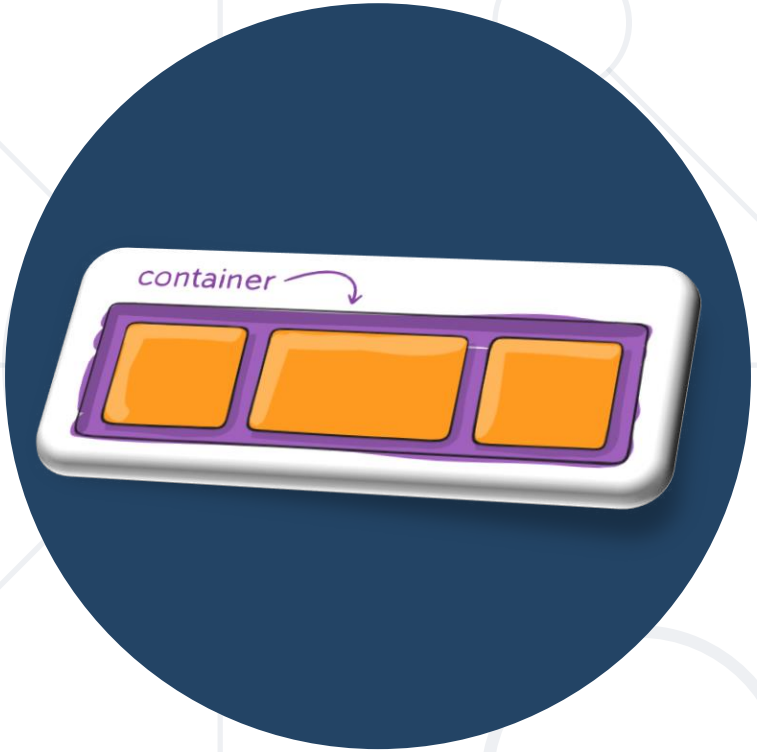
- For elements to be laid out as flexible boxes we set a special value of display on the parent element of the elements you want to affect. This causes the element to become a ***flex container*** and its children to become ***flex items***
- When elements are laid out as flex items, they are laid out along two axes:

The Flex Model



Flexbox properties

- Complexity and the value of reference
  - The entire specification with all possible properties that are related to flexbox can be overwhelming, especially in the beginning. Don't try to learn and remember everything. Try to understand the basic concepts – the properties will be learned through exercise and repetition
  - https://css-tricks.com/snippets/css/a-guide-to-flexbox/



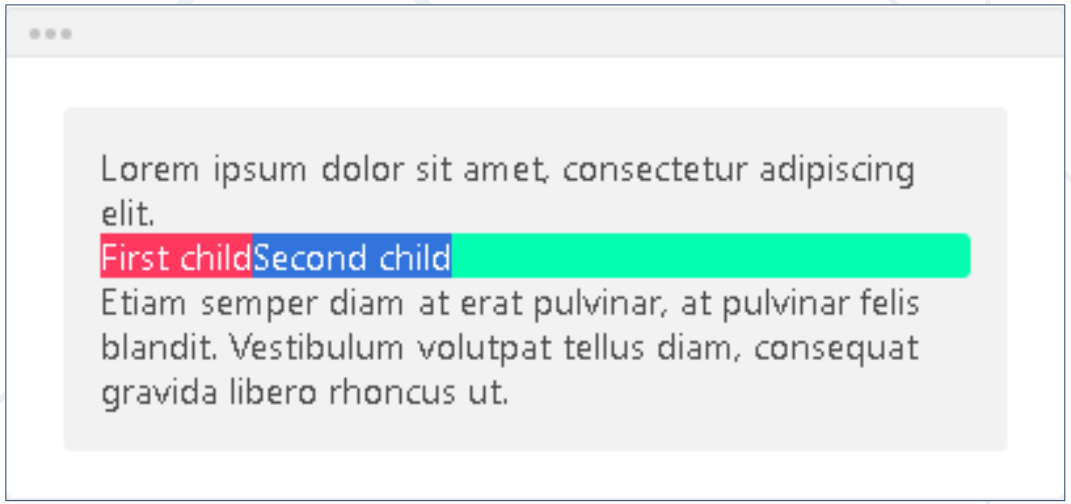
Properties for the Container

Display – Flex

- 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
ravida libero rhoncus ut.
    </p>
</body>
```

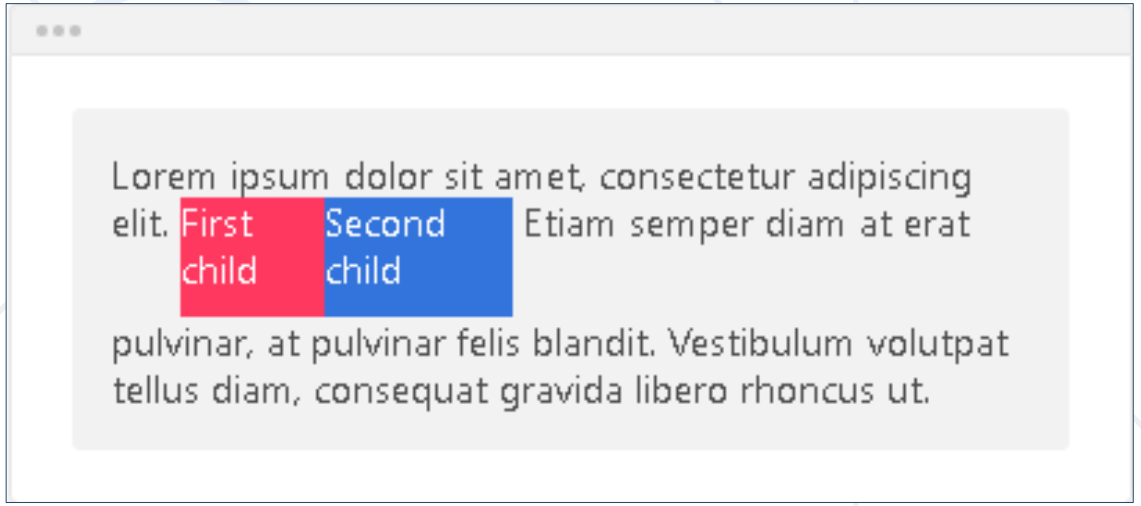
```
.container {
  display: flex;
}
```



Display – Inline-flex

- 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;
}
```



Flex Direction

- 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

Flex Direction

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

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

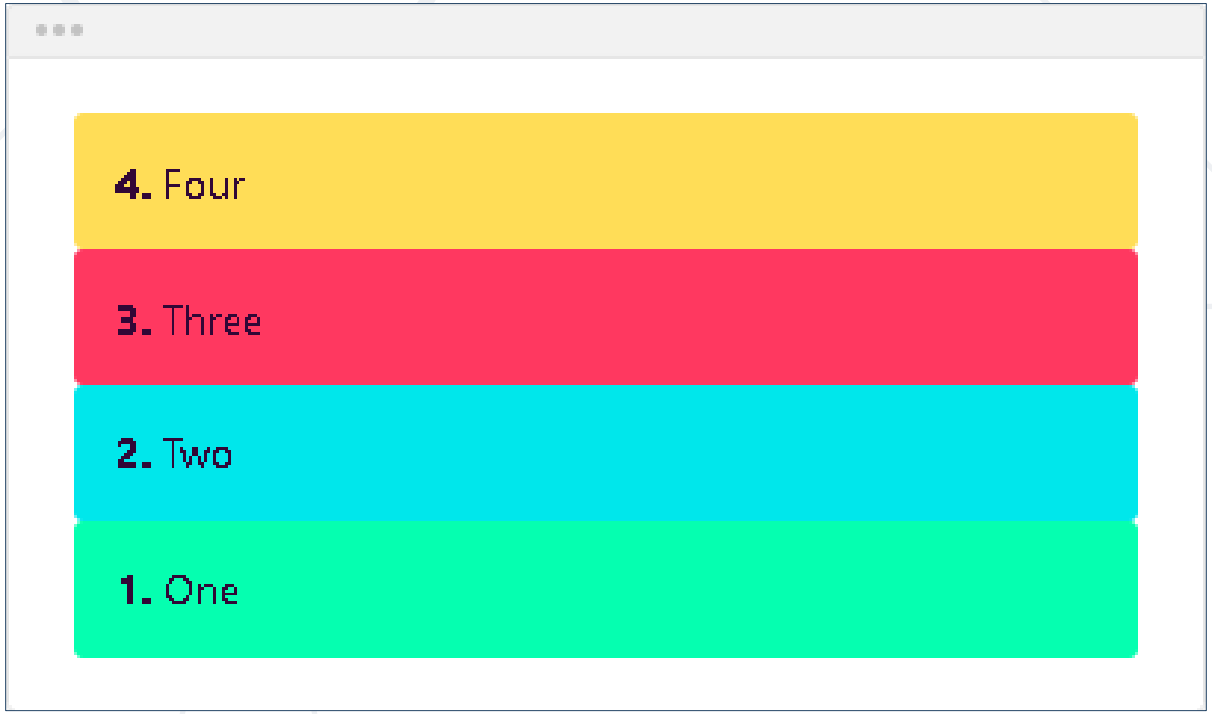
- The flexbox items are ordered the **same** way as the **text direction**, along the **cross axis**

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

Flex Direction

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

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



Flex Wrap

- 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

Flex Wrap

- 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

- `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;
}
```

Justify Content

- 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

Justify Content

- 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;
```

Justify Content

- 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

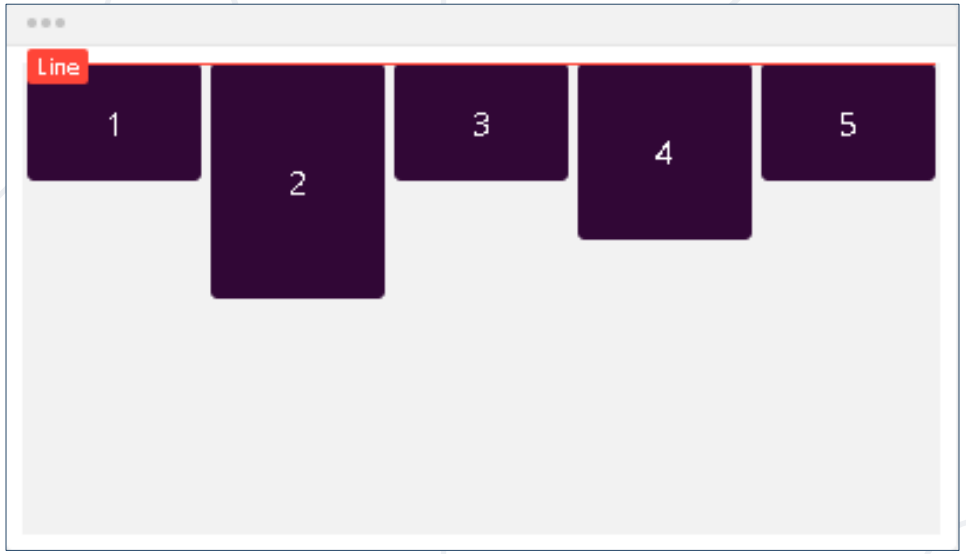
```
justify-content: space-around;
```

Align Items

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

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

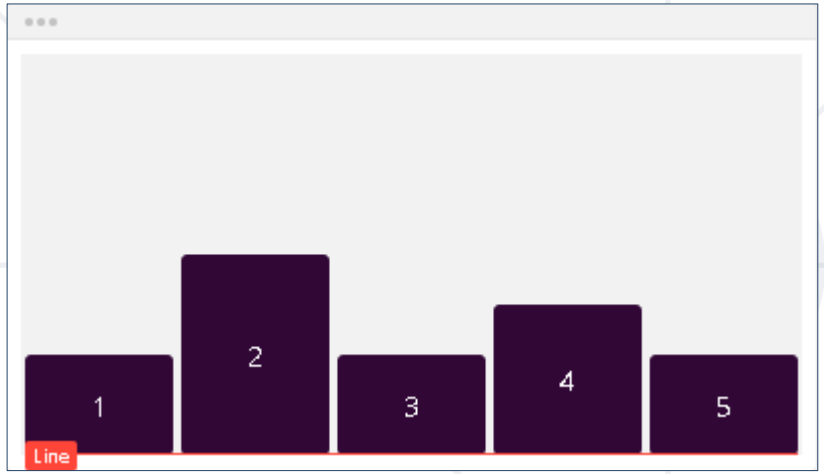
  - The flexbox items are aligned at the **start** of the **cross axis**



Align-items

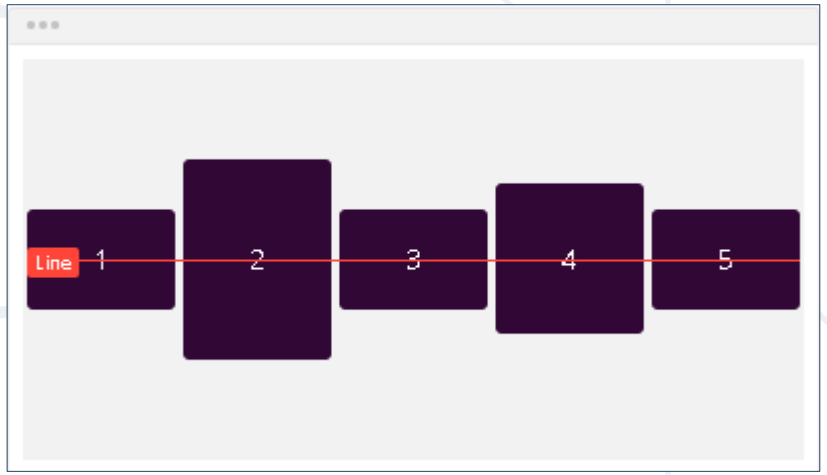
- The flexbox items are aligned at the **end** of the **cross axis**

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



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

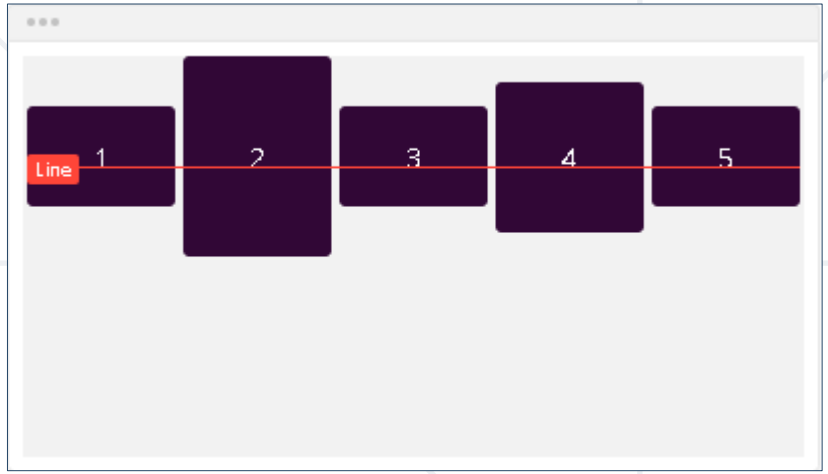
```
align-items: center;
```



Align-items

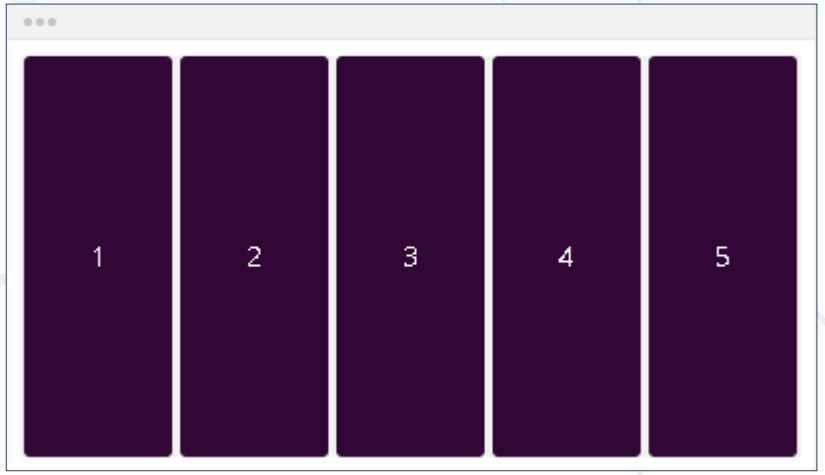
- The flexbox items are aligned at the **baseline** of the **cross axis**

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



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

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



Align Content

- 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

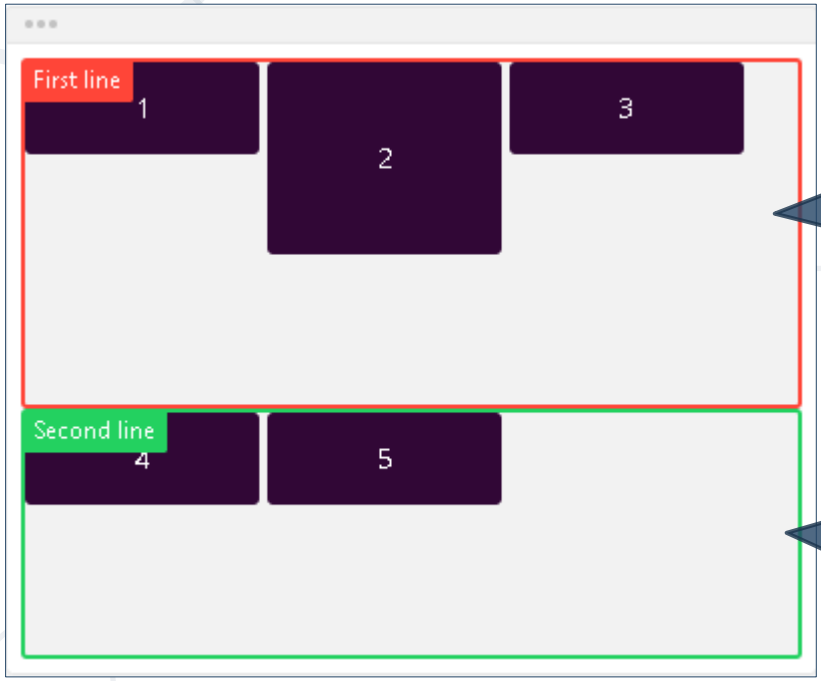
```
align-content: stretch;
```

  - 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



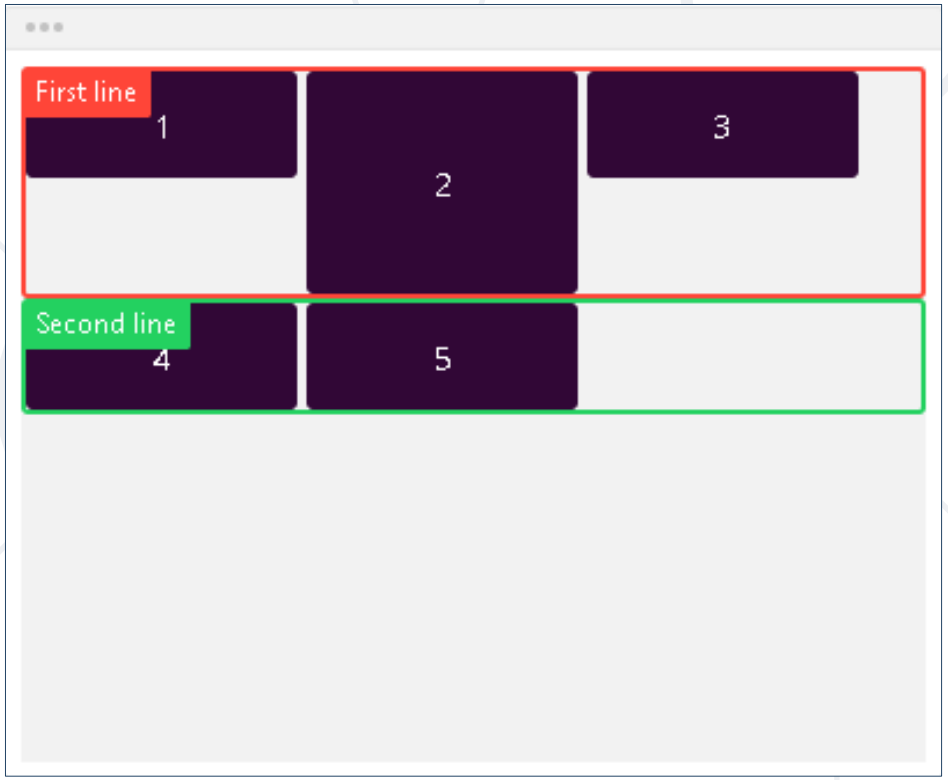
The first line is 175px high

The second line is 125px high

Align Content

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

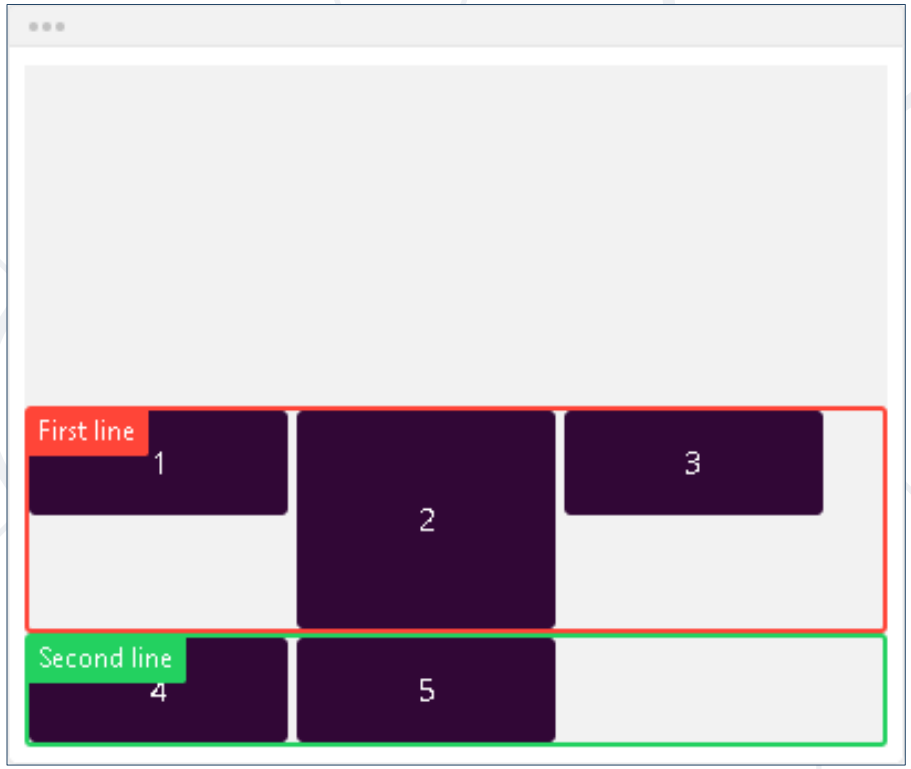
```
align-content: flex-start;
```



Align Content

- 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;
```



Align Content

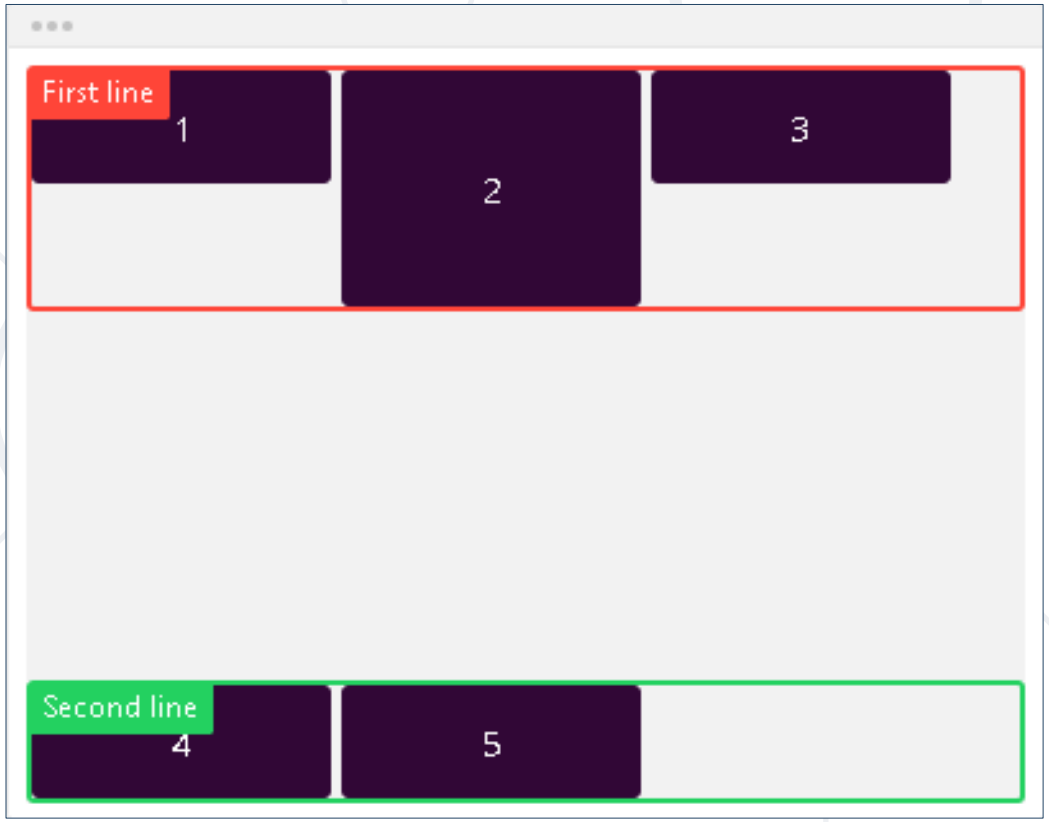
- Each line will only fill the space it **needs**
- They will all move towards the **center** of the flexbox container's cross axis

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

Align Content

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

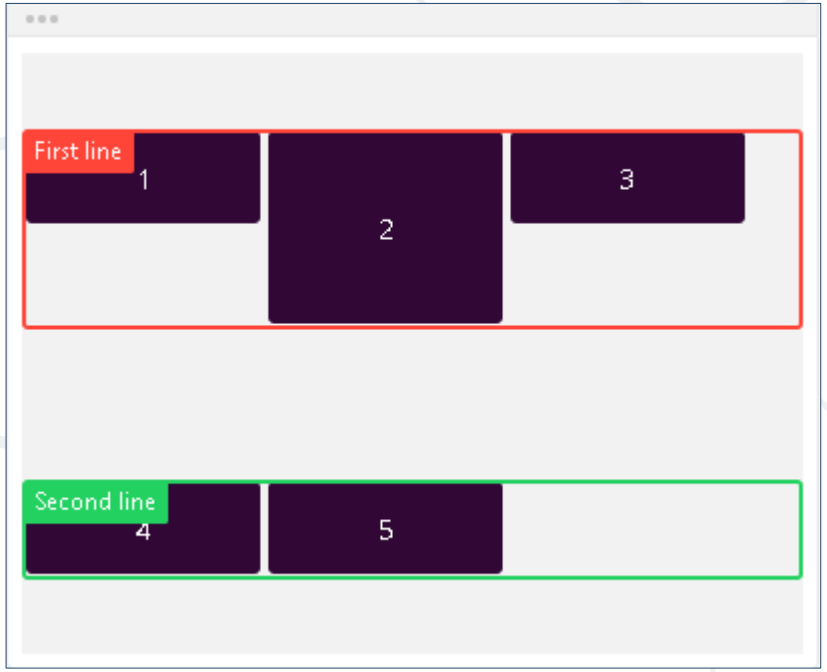
```
align-content: space-between;
```

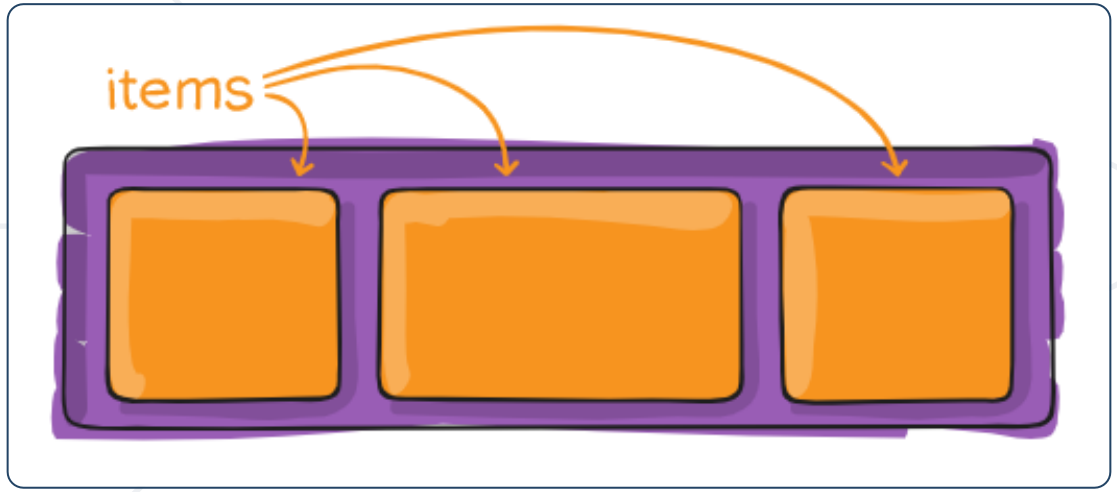


Align Content

- 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;
```





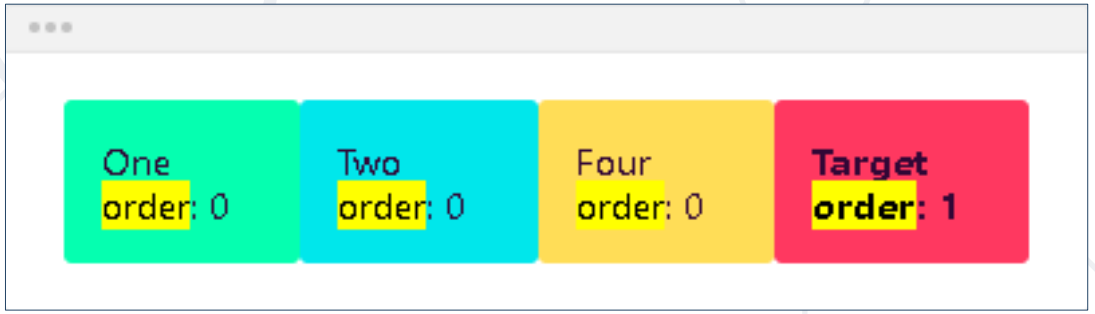
Properties for the Children

Order

- 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;
```

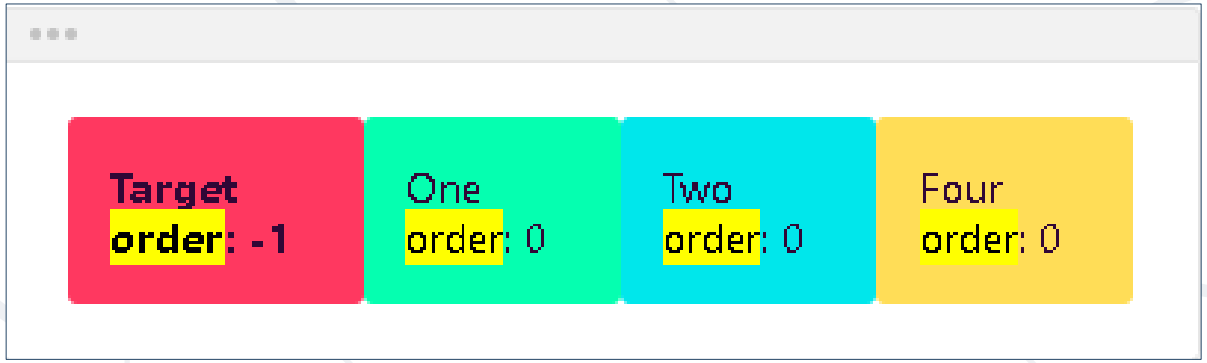

```
order: 1;
```



Order

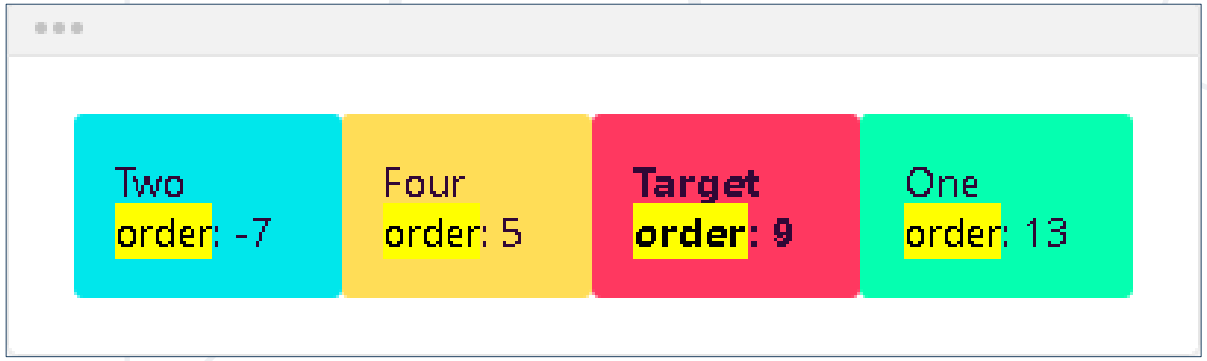
- You can use **negative** values

```
order: -1;
```



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

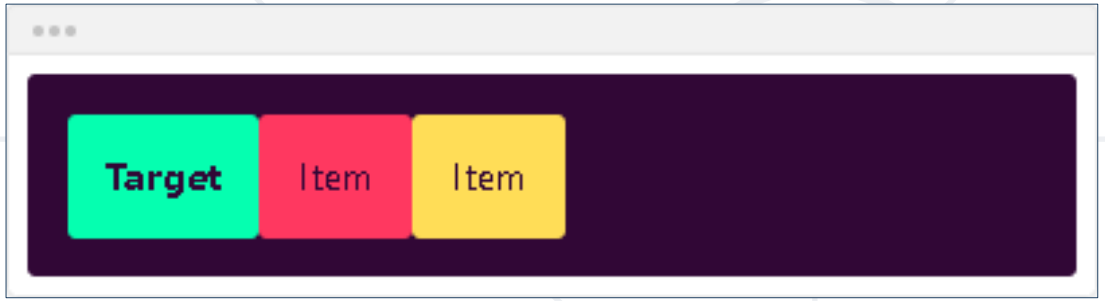
```
order: 9;
```



Flex Grow

- 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;
```



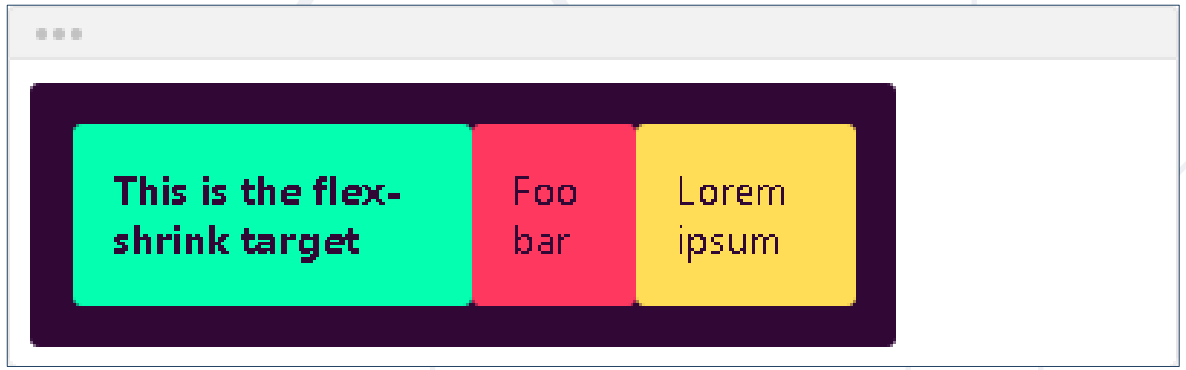

```
flex-grow: 1;
```

Flex Shrink

- 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



Flex Shrink

- 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**

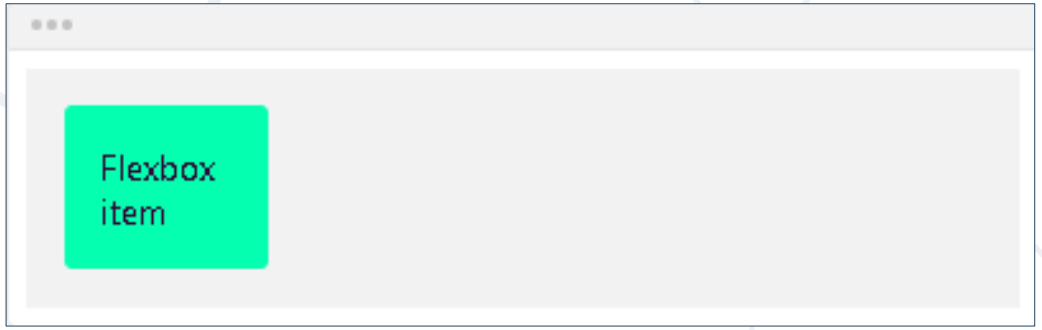
```
flex-shrink: 0;
```

Flex Basis

- 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

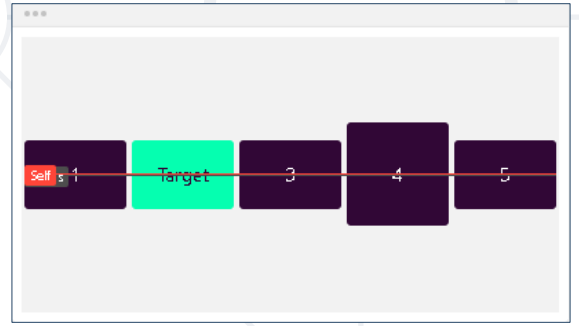
- Flex is the shorthand for:
  - `flex-grow`
  - `flex-shrink`
  - `flex-basis`
- The default value is `0 1 auto`

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

Align Self

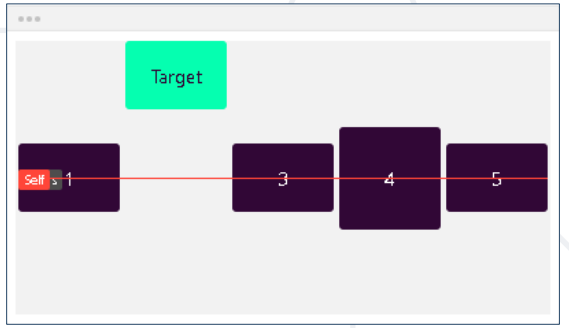
- Works like **align-items**, but applies only to a **single** flexbox item, instead of all of them
  - The target will use the value of **align-items**

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



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

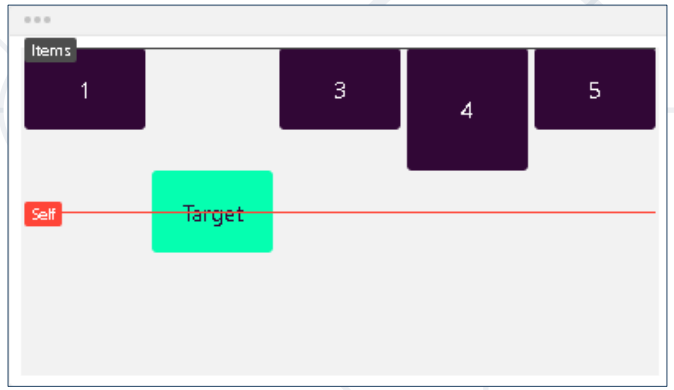
```
align-self: flex-start;
```



Align Self

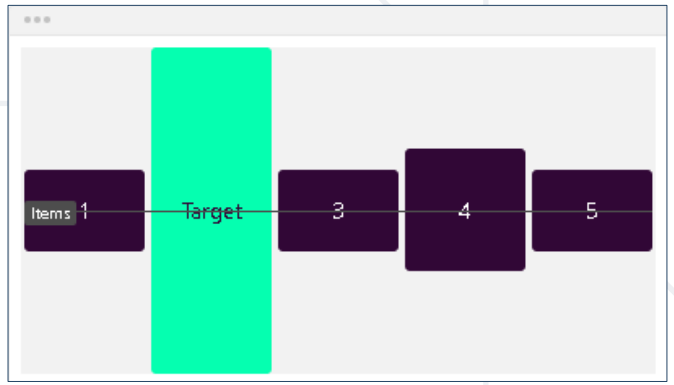
- The container has align-items: `flex-start`
- The target has `align-self: center`

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



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

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



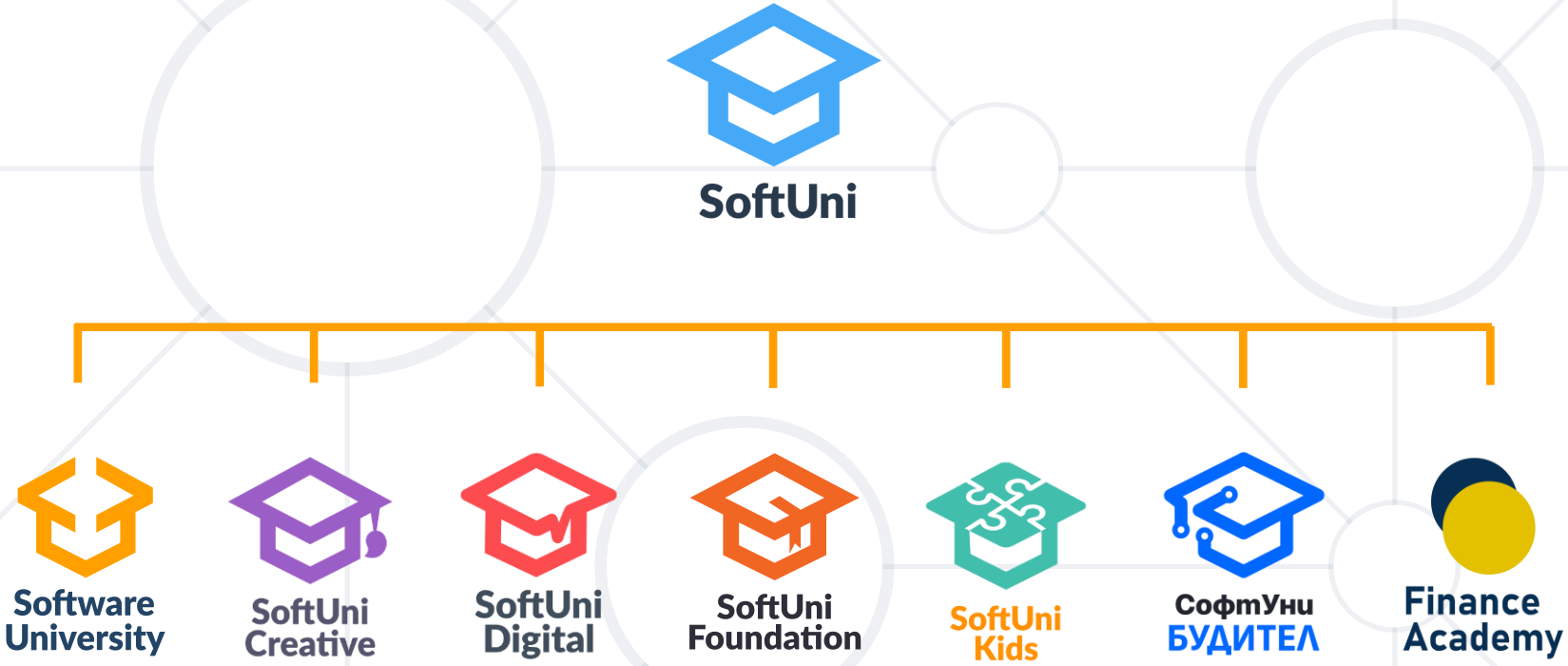
Resources

- https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox
- https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox
- https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- https://www.youtube.com/watch?v=hs3piaN4b5I
- https://www.flexboxgame.com/

Summary

- 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

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

Решения за твоето утре

DRAFT KINGS

THE CROWN IS YOURS

VIVACOM

AMBITIONED

INDEAVR

Serving the high achievers

PHAR VISION

Trainings @ Software University (SoftUni)

- 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

License

- 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