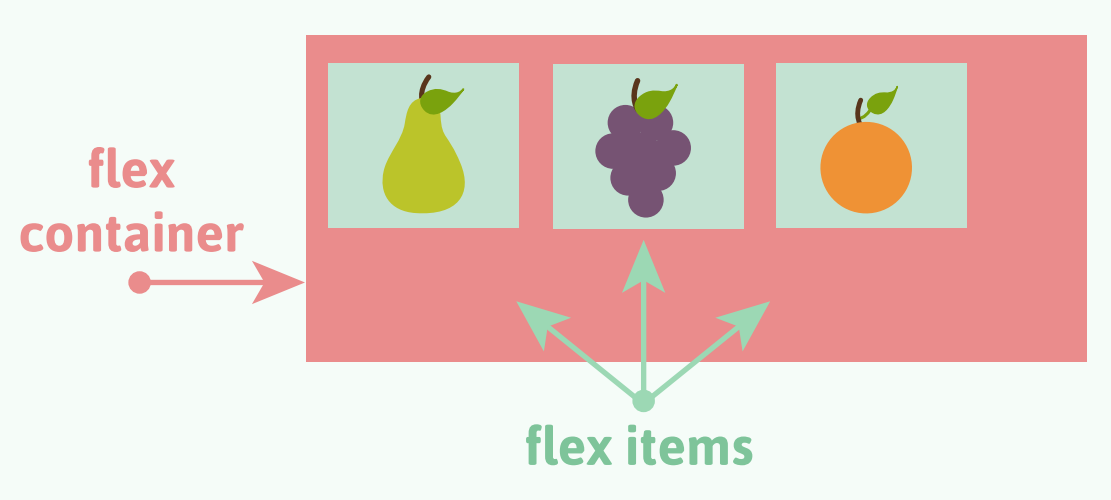
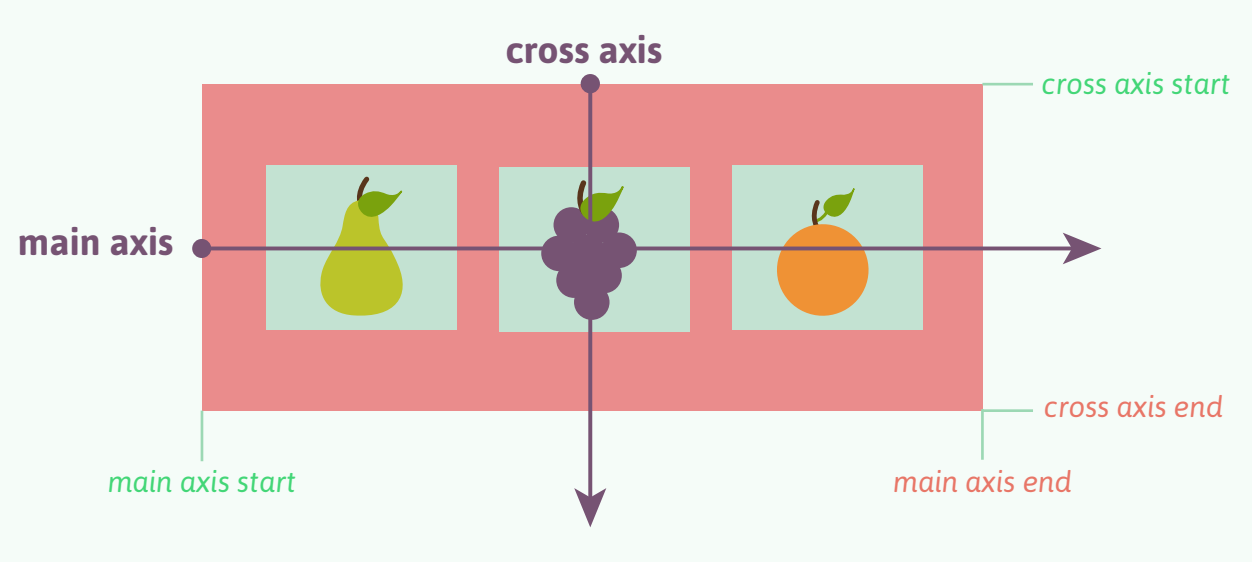
Flexbox Cheatsheet Cheatsheet





To activate powers

`display: flex;` on flex container

Do you want rows or columns?

rows

`flex-direction: row;` on container

OR

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

columns

`flex-direction: column;` on container

OR

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

Do you want these items located at the beginning of the main axis?

YES, beginning

OK, default value has you covered.

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

NO, other

OK, use one of these:

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

OR

`justify-content: center;` on container

OR

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

OR

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

Do you want the items on one line or to move to another when adjusting viewport?

one line

OK, default value has you covered.

`flex-wrap: nowrap;` on container

move to another

OK, use one of these:

`flex-wrap: wrap;` on container

OR

`flex-wrap: wrap-reverse;` on container

How do you want these items laid out on the cross axis?

stretched

OK, default value has you covered.

`align-items: stretch;` on container

not stretched

OK, use one of these:

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

OR

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

OR

`align-items: center;` on container

OR

`align-items: baseline;` on container

If you have multiple lines of content, how do you want this aligned?

stretched

OK, default value has you covered.

`align-content: stretch;` on container

not stretched

OK, use one of these:

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

OR

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

OR

`align-content: center;` on container

OR

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

OR

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

Do you want to change the order of the items?

NO

Awesome, don't do anything. That was easy.

YES

OK, use this:

`order: <whole number>;` on item

Do you need some items to grow if necessary?

NO

Awesome, don't do anything. That was easy.

YES

OK, use this:

`flex-grow: <whole number>;` on item

Do any of these items need to be aligned differently than the others?

NO

OK, you're done. That was super easy.

YES

OK, use these:

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

OR

`align-self: flex-end;` on item

OR

`align-self: center;` on item

OR

`align-self: baseline;` on item

OR

`align-self: stretch;` on item

Made with ♥ by Joni Bologna

Flexbox in CSS

The CSS properties that allow you to use the CSS3 Flexbox capabilities

align-content

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

`align-content: stretch;` default

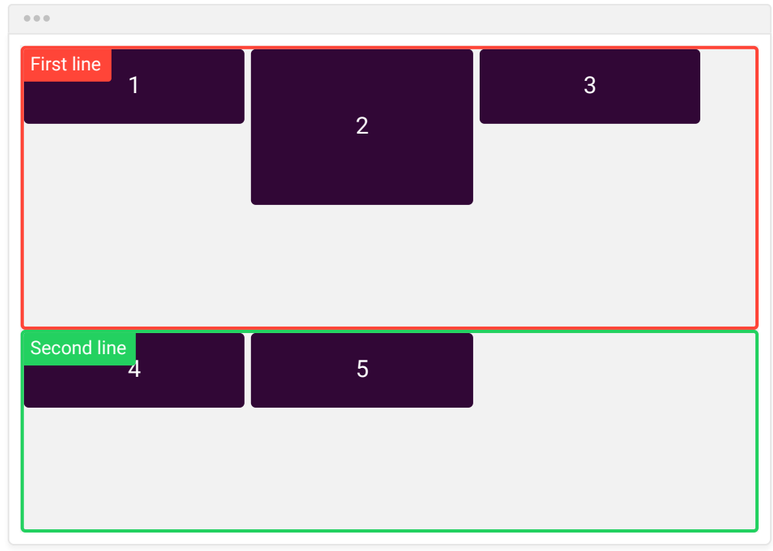
Each line will stretch to *fill* the remaining space.

In this case, the container is `300px` high. All boxes are `50px` high, apart from the second one who is `100px` high.

- The first line is **100px** high
- The second line is **50px** high
- The remaing space is **150px**

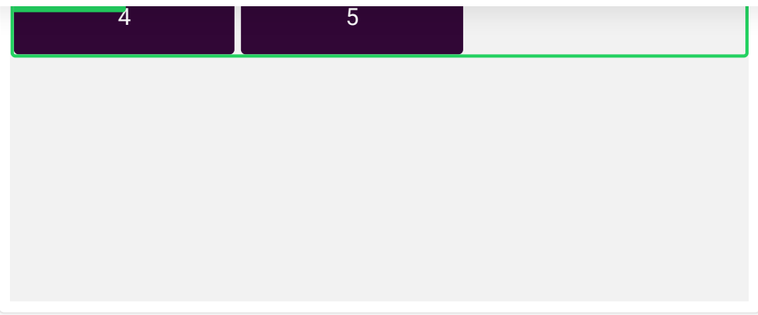
This remaining space is distributed equally amongst the two lines:

- The first line is now **175px** high
- The second line is now **125px** high



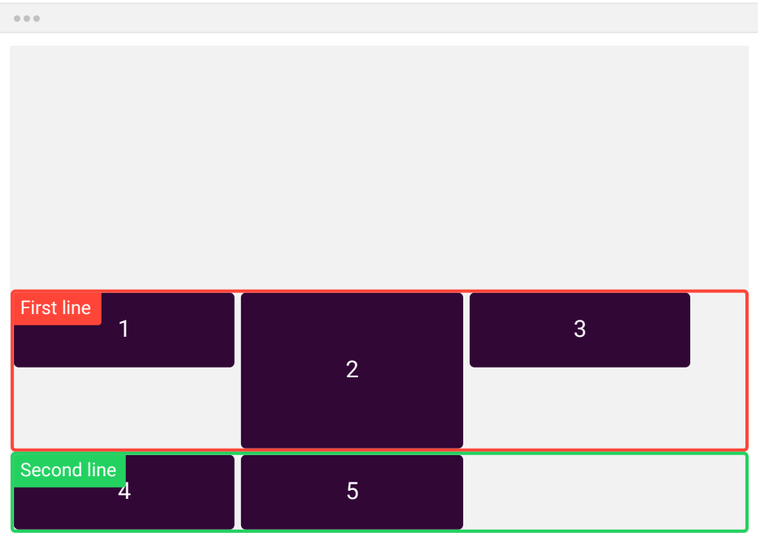
`align-content: flex-start;`

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



`align-content: flex-end;`

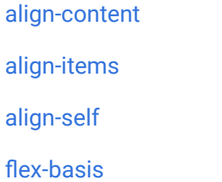
Each line will only fill the space it *needs*. They will all move towards the **end** of the flexbox/grid container's cross axis.

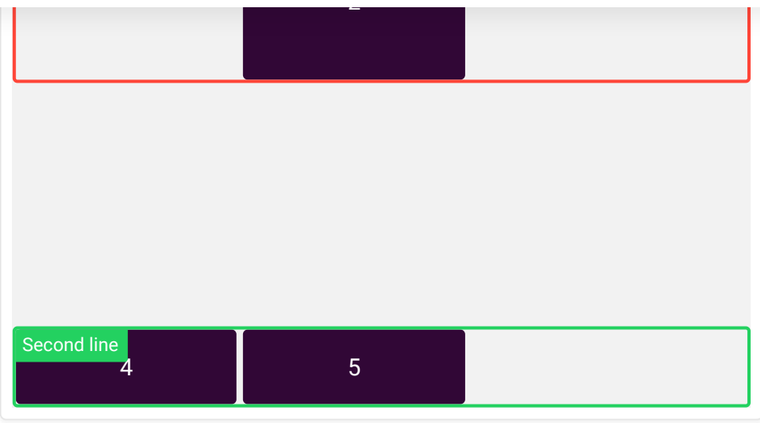


`align-content: center;`

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

`align-content: space-between;`






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

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-items

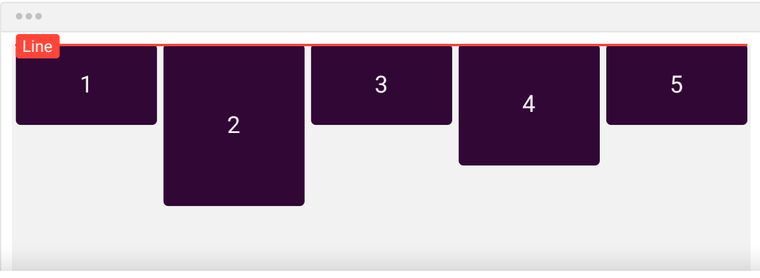
In collection: **Flexbox** Permalink Share Can I use MDN

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**.

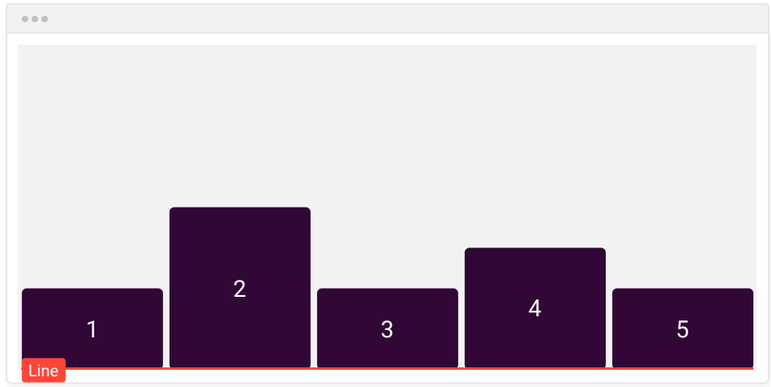
By default, the cross axis is vertical. This means the flexbox items will be aligned *vertically* at the *top*.



`align-items: flex-end;`

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

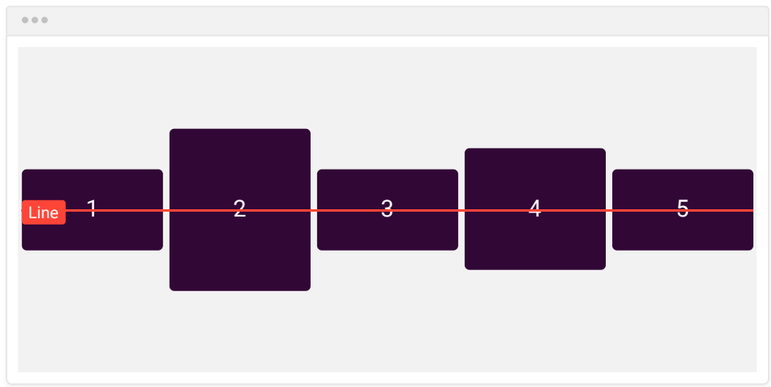
By default, the cross axis is vertical. This means the flexbox items will be aligned *vertically* at the *bottom*.



`align-items: center;`

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

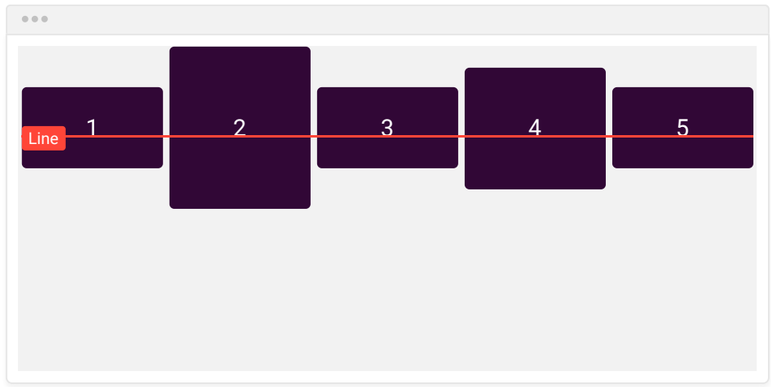
By default, the cross axis is vertical. This means the flexbox items will be **centered** *vertically*.



`align-items: baseline;`

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

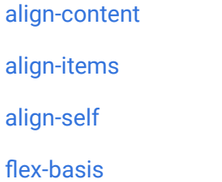
By default, the cross axis is vertical. This means the flexbox items will align themselves in order to have the *baseline* of their *text* align along a horizontal line.

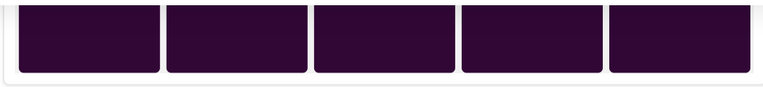


`align-items: stretch;`

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

By default, the cross axis is vertical. This means the flexbox items will fill up the whole vertical space.



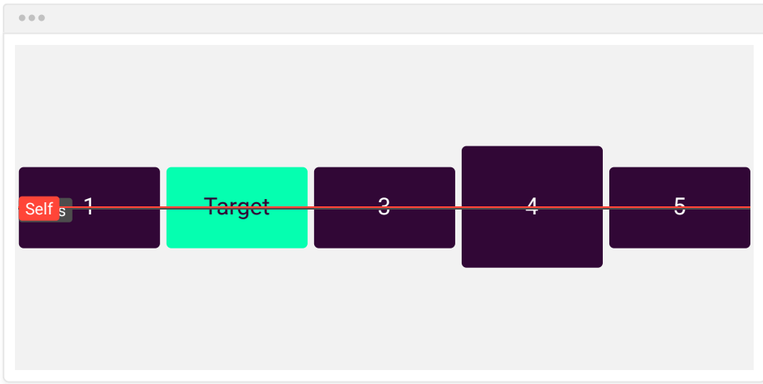


align-self

Works like `align-items`, but applies only to a **single** flexbox item, instead of *all* of them.

`align-self: auto;` default

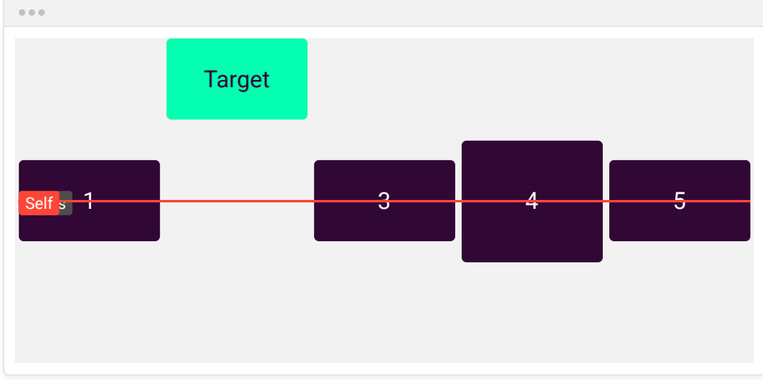
The target will use the value of `align-items`.



`align-self: flex-start;`

If the container has `align-items: center` and the **target** has `align-self: flex-start`, only the target will be at the start of the cross axis.

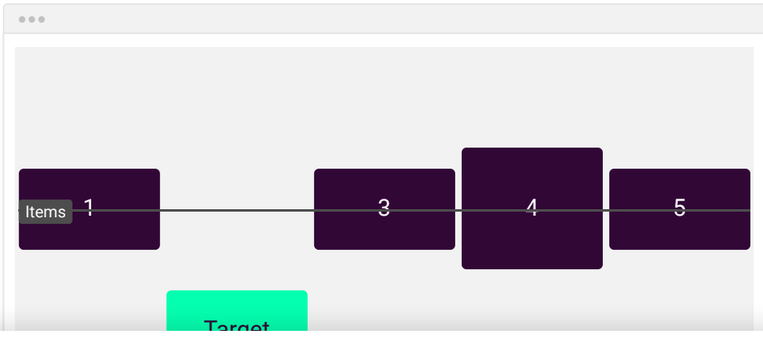
By default, this means it will be aligned **vertically** at the **top**.



`align-self: flex-end;`

If the container has `align-items: center` and the **target** has `align-self: flex-end`, only the target will be at the end of the cross axis.

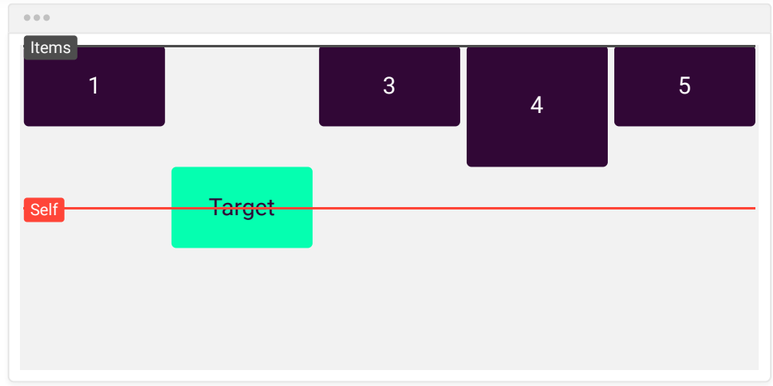
By default, this means it will be aligned **vertically** at the **bottom**.



`align-self: center;`

If the container has `align-items: flex-start` and the **target** has `align-self: center`, only the target will be at the center of the cross axis.

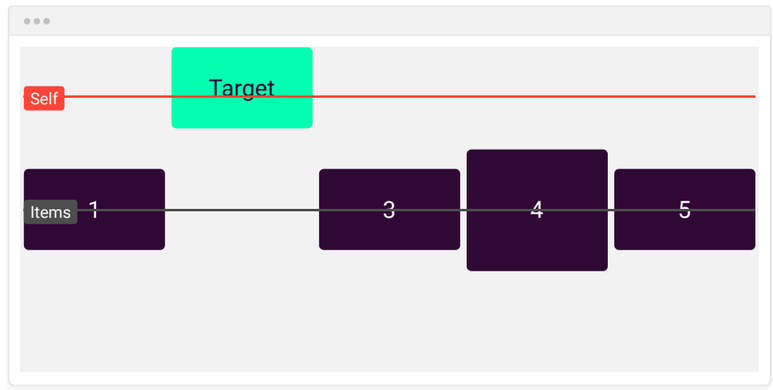
By default, this means it will be **vertically centered**.



`align-self: baseline;`

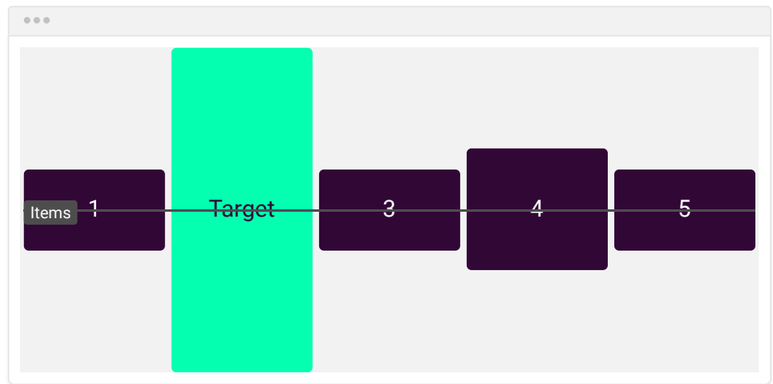
If the container has `align-items: center` and the **target** has `align-self: baseline`, only the target will be at the baseline of the cross axis.

By default, this means it will be aligned along the basline of the text.



`align-self: stretch;`

If the container has `align-items: center` and the **target** has `align-self: stretch`, only the target will stretch along the whole cross axis.



flex-basis

In collection: **Flexbox** Permalink Share Can I use MDN

Defines the initial size of a flexbox item.

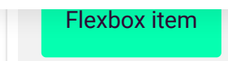
cssreference.io

All properties Animations
Backgrounds Box model
Flexbox CSS Grid
Positioning Transitions
Typography

Search for a property

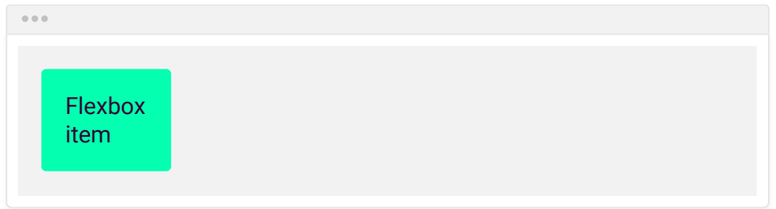
align-content
align-items
align-self
flex-basis
flex-direction
flex-flow
flex-grow
flex-shrink
flex-wrap
justify-content
order

value if they are defined.




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

You can define **pixel** or **(r)em** values. The element will wrap its content to avoid any overflow.



flex-direction

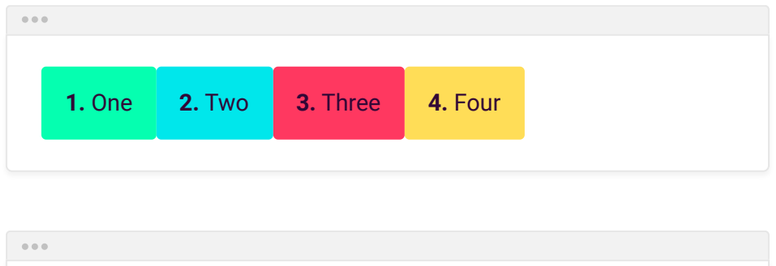
In collection: **Flexbox** Permalink Share Can I use MDN

Defines how flexbox items are ordered within a flexbox container.

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

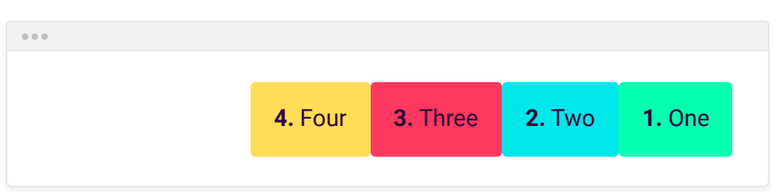
default

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




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

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




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

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

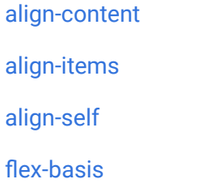

```
flex-direction: column-reverse;
```

cssreference.io

All properties Animations
Backgrounds Box model
Flexbox CSS Grid
Positioning Transitions
Typography

Search for a property

align-content
align-items
align-self
flex-basis
flex-direction
flex-flow
flex-grow
flex-shrink
flex-wrap
justify-content
order



flex-flow

In collection: **Flexbox** Permalink Share Can I use MDN

Shorthand property for `flex-direction` and `flex-wrap`.

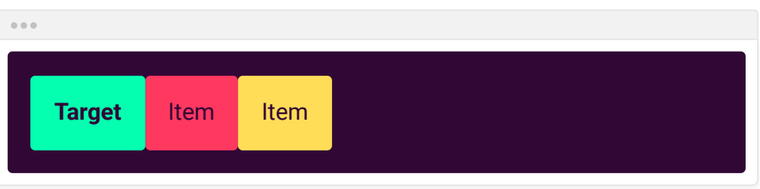
flex-grow

In collection: **Flexbox** Permalink Share Can I use MDN

Defines how much a flexbox item should **grow** if there's space available.

`flex-grow: 0;` default

The element will **not** grow if there's space available. It will only use the space it needs.



`flex-grow: 1;`

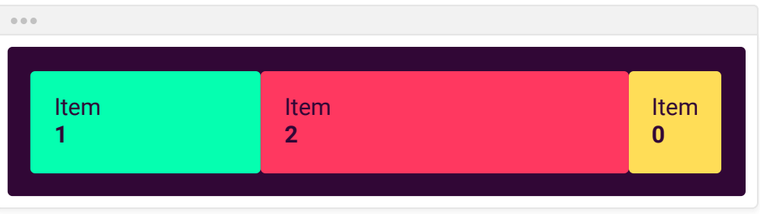
The element will **grow** by a factor of **1**. It will fill up the remaining space if no other flexbox item has a `flex-grow` value.

`flex-grow: 2;`

Because the flex-grow value is **relative**, its behaviour depends on the value of the flexbox item **siblings**.

In this example, the remaining space is divided in **3**:

- **1** third goes to the **green** item
- **2** thirds go to the **pink** item



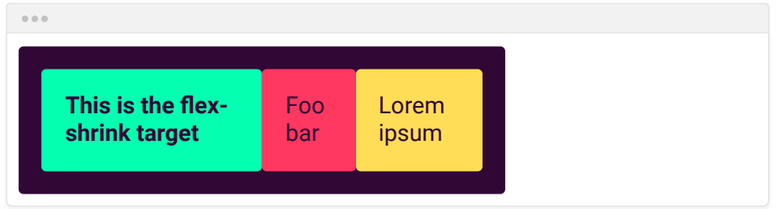
flex-shrink

In collection: **Flexbox** Permalink Share Can I use MDN

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

`flex-shrink: 1;` default

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: 0;`

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.

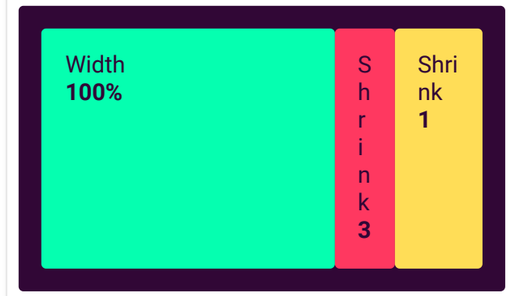
Because the target element will not wrap its content, there is a chance for the flexbox container's content to **overflow**.

`flex-shrink: 2;`

Because the flex-shrink value is **relative**, its behaviour depends on the value of the flexbox item **siblings**.

In this example, the green item wants to fill 100% of the width. The space it needs is **taken from** its two siblings, and is divided in **4**:

- **3** quarters are taken from the **red** item
- **1** quarter is taken from the **yellow** item



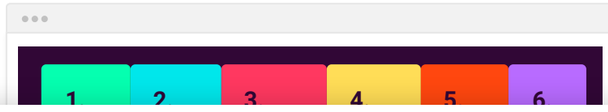
flex-wrap

In collection: **Flexbox** Permalink Share Can I use MDN

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

`flex-wrap: nowrap;` default

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



`flex-wrap: wrap;`

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

`flex-wrap: wrap-reverse;`

The flexbox items will be distributed among **multiple lines** if needed. Any additional line will appear **before** the previous one.

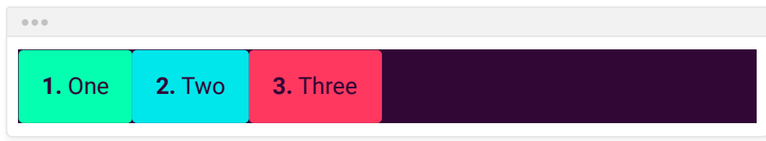
justify-content

In collection: **Flexbox** Permalink Share Can I use MDN

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

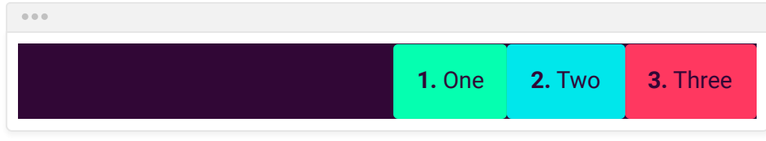
`justify-content: flex-start;` default

The flexbox/grid items are pushed towards the **start** of the container's main axis.



`justify-content: flex-end;`

The flexbox/grid items are pushed towards the **end** of the container's main axis.

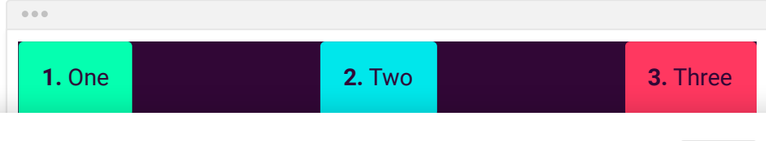


`justify-content: center;`

The flexbox/grid items are **centered** along the container's main axis.

`justify-content: space-between;`

The remaining space is distributed **between** the flexbox/grid items.



`justify-content: space-around;`

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

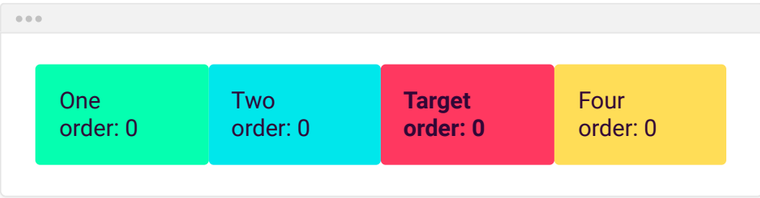
order

In collection: **Flexbox** Permalink Share Can I use MDN

Defines the order of a flexbox item.

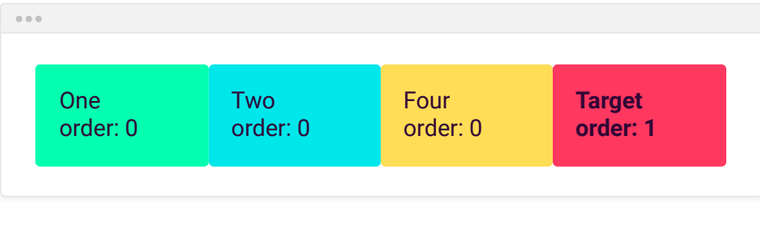
`order: 0;` default

The order of the flexbox items is the one defined in the **HTML code**.



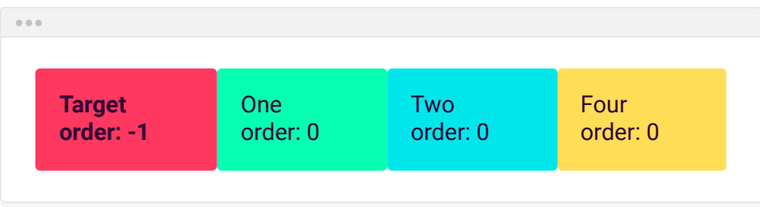
`order: 1;`

The order is **relative** to the flexbox item's *siblings*. The final order is defined when all individual flexbox item order values are taken into account.



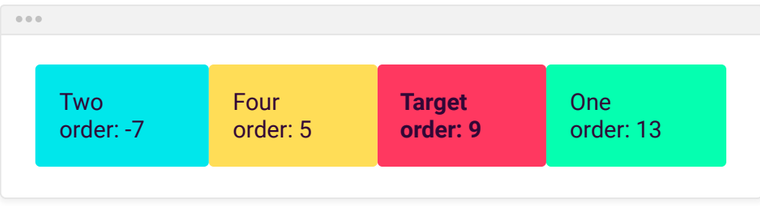
`order: -1;`

You can use **negative** values.



`order: 9;`

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



Layout with Flexbox

CSS Flexbox

The CSS `display: flex` property sets an HTML element as a block level flex container which takes the full width of its parent container. Any child elements that reside within the flex container are called flex items.
Flex items change their size and location in response to the size and position of their parent container.

```
div {
  display: flex;
}
```

`justify-content` Property

The CSS `justify-content` flexbox property defines how the browser distributes space between and around content items along the main-axis of their container. This is when the content items do not use all available space on the major-axis (horizontally).
`justify-content` can have the values of:

- `flex-start`
- `flex-end`
- `center`
- `space-between`
- `space-around`

```
/* Items based at the center of the parent container: */
div {
  display: flex;
  justify-content: center;
}

/* Items based at the upper-left side of the parent container:
*/
div {
  display: flex;
  justify-content: flex-start;
}
```

`flex` Property

The `flex` CSS property specifies how a flex item will grow or shrink so as to fit within the space available in its `flex` container. This is a shorthand property that declares the following properties in order on a single line:

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

```
/* Three properties declared on three lines: */
.first-flex-item {
  flex-grow: 2;
  flex-shrink: 1;
  flex-basis: 150px;
}

/* Same three properties declared on one line: */
.first-flex-item {
  flex: 2 1 150px;
}
```

`flex-direction` Property

The `flex-direction` CSS property specifies how flex items are placed in the flex container - either vertically or horizontally. This property also determines whether those flex items appear in order or in reverse order.

```
div {
  display: flex;
  flex-direction: row-reverse;
}
```

`align-content` Property

The `align-content` property modifies the behavior of the flex-wrap property. It determines how to space rows from top to bottom (ie. along the cross axis). Multiple rows of items are needed for this property to take effect.

`flex-grow` Property

The CSS `flex-grow` property allows flex items to grow as the parent container increases in size horizontally. This property accepts numerical values and specifies how an element should grow relative to its sibling elements based on this value.
The default value for this property is `0`.

```
.panelA {
  width: 100px;
  flex-grow: 1;
}

/* This panelB element will stretch twice wider than the
panelA element */
.panelB {
  width: 100px;
  flex-grow: 2;
}
```

`flex-shrink` Property

The CSS `flex-shrink` property determines how an element should shrink as the parent container decreases in size horizontally. This property accepts a numerical value which specifies the ratios for the shrinkage of a flex item compared to its other sibling elements within its parent container.
The default value for this property is `1` .

```
.container {
  display: flex;
}

.item-a {
  flex-shrink: 1;
  /* The value 1 indicates that the item should shrink. */
}

.item-b {
  flex-shrink: 2;
  /* The value 2 indicates that the item should shrink twice
than the element item-a. */
}
```

Css flex-basis property

The `flex-basis` CSS property sets the initial base size for a flex item before any other space is distributed according to other flex properties.

```
// Default Syntax
flex-basis: auto;
```

The CSS `flex-flow` property

The CSS property `flex-flow` provides a shorthand for the properties `flex-direction` and `flex-wrap` . The value of the `flex-direction` property specifies the direction of the flex items and the value of the `flex-wrap` property allows flex items to move to the next line instead of shrinking to fit inside the flex container. The `flex-flow` property should be declared on the flex container.

```
// In this example code block, "column" is the value of the
property "flex-direction" and "wrap" is the value of the
property "flex-wrap".

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

CSS `display: inline-flex` property

The CSS `display: inline-flex` property sets an HTML element as an inline flex container which takes only the required space for the content. Any child elements that reside within the flex container are caleld flex items. Flex items change their size and location in response to the size and position of their parent container.

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

Flexbox Properties align-items

When working with CSS flexbox `align-items` is used to align flex items vertically within a parent container.

Css flex-wrap property

The `flex-wrap` property specifies whether flex items should wrap or not. This applies to flex items only. Once you tell your container to `flex-wrap` , wrapping become a priority over shrinking. Flex items will only begin to wrap if their combined `flex-basis` value is `greater` than the current size of their flex container.

```
.container {
  display: flex;
  flex-wrap: wrap;
  width: 200px;
}
```