# WEB DESIGN

**CSS flexbox layout**

**Amal elnuri**

# CSS flexbox layout

The Flexbox Layout (Flexible Box) module aims at providing a more efficient way to lay out, align and distribute space among items in a container, even when their size is unknown and/or dynamic (thus the word “flex”).

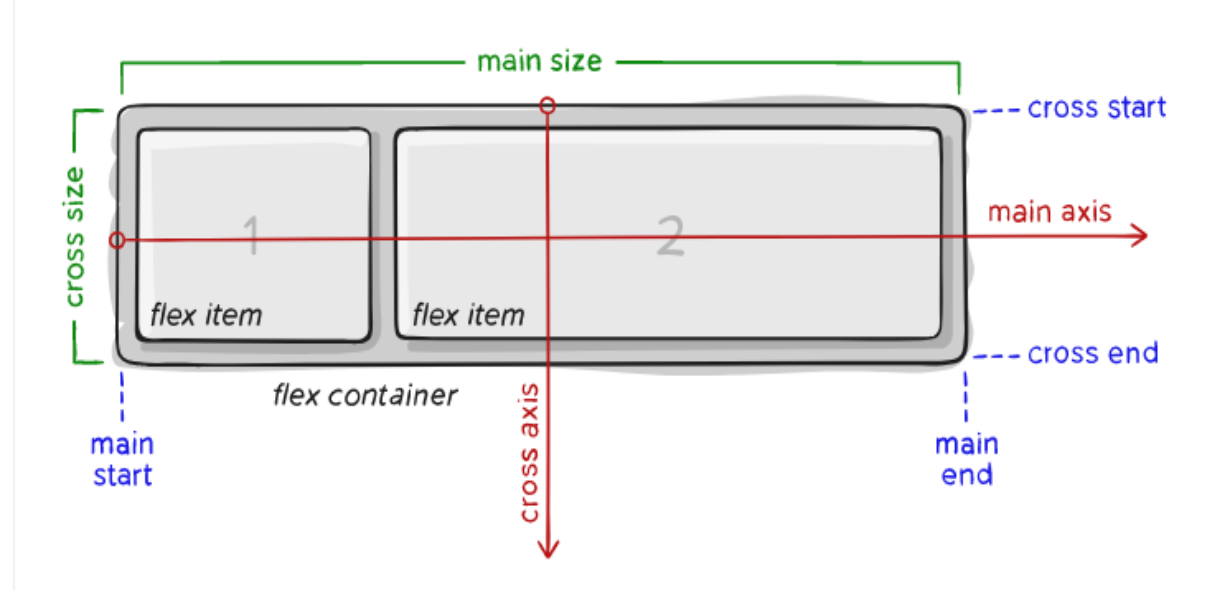
The main idea behind the flex layout is to give the container the ability to alter its items’ width/height (and order) to best fill the available space (mostly to accommodate to all kind of display devices and screen sizes). A flex container expands items to fill available free space or shrinks them to prevent overflow.

**Note**: Flexbox layout is most appropriate to the components of an application, and small-scale layouts, while the Grid layout is intended for larger scale layouts.

# Basics and terminology

Since flexbox is a whole module and not a single property, it involves a lot of things including its whole set of properties.

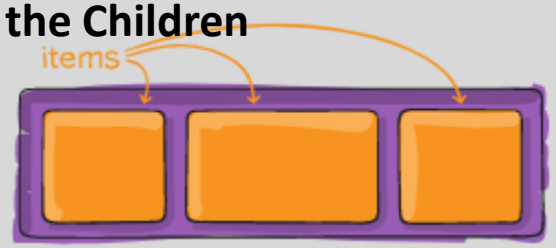
Some of them are meant to be set on the container (parent element, known as “flex container”) whereas the others are meant to be set on the children (said “flex items”).



# Flexbox properties

**Properties for the Parent**
**(flex container)**


**Properties for the Children**
**(flex items)**




## parent element

1. display: flex;
2. flex-direction: row;
3. flex-wrap: wrap;
4. flex-flow: row wrap;
5. justify-content: center;
6. align-content: center;

## child element

1. flex-grow: 0;
2. flex-shrink: 1;
3. flex-basis: auto;
4. flex: 0 1 auto;
5. order: 0;

# Display

This defines a flex container;
inline or block depending on the given value.
It enables a flex context for all its direct children.

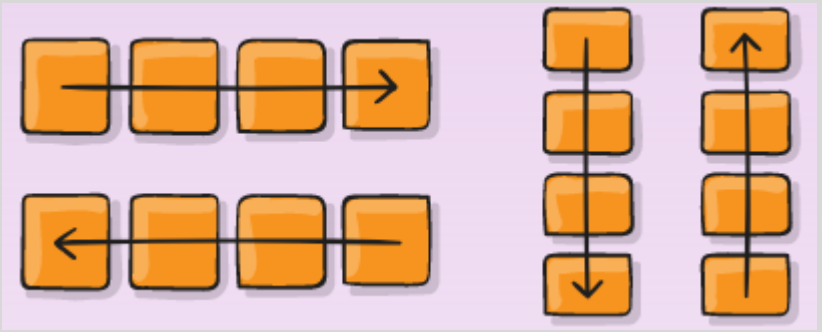
```css
.container {
  display: flex; /* or inline-flex */
}
```

```html
<!DOCTYPE html>
<html lang="ar">
  <head>
    <style>
      .container {display: flex;}
    </style>
  </head>

  <body>
    <div class="container">
      <div class="child1">الرئيسية</div>
      <div class="child2">الرابط الثاني</div>
      <div class="child3">الرابط الثالث</div>
    </div>
  </body>
</html>
```

# flex-direction

This establishes the main-axis, thus defining the direction flex items are placed in the flex container. Flexbox is (aside from optional wrapping) a single-direction layout concept. Think of flex items as primarily laying out either in horizontal rows or vertical columns.

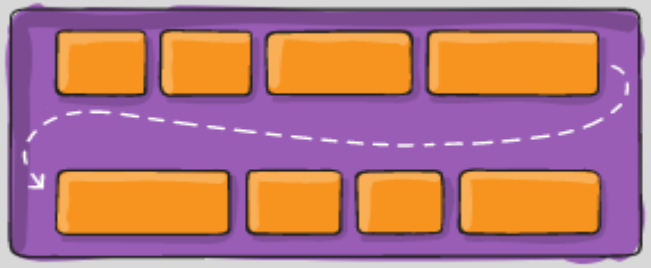


```css
.container {
  flex-direction: row | row-reverse
}
```

- row (default): left to right in ltr; right to left in rtl
- row-reverse: right to left in ltr; left to right in rtl
- column: same as row but top to bottom
- column-reverse: same as row-reverse but bottom to top

# flex-wrap

By default, flex items will all try to fit onto one line. You can change that and allow the items to wrap as needed with this property.



```
.container {
  flex-wrap: nowrap | wrap | wrap-re
}
```

- nowrap (default): all flex items will be on one line
- wrap: flex items will wrap onto multiple lines, from top to bottom.
- wrap-reverse: flex items will wrap onto multiple lines from bottom to top.

# flex-flow

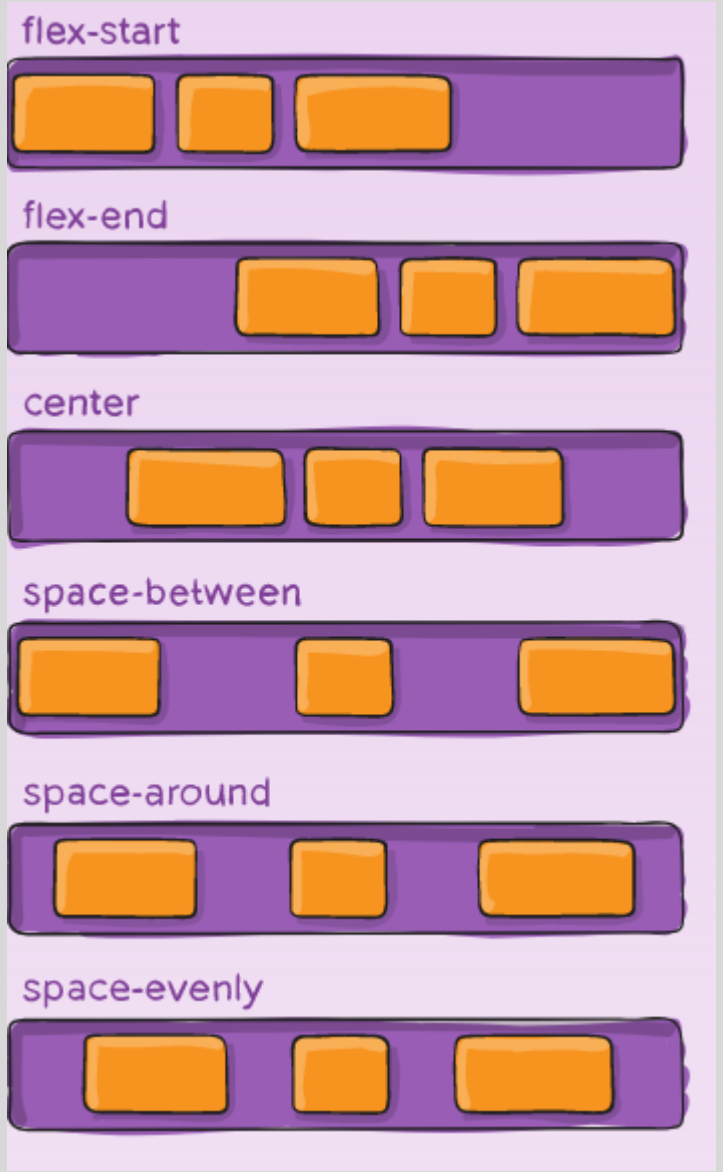
This is a shorthand for the flex-direction and flex-wrap properties, which together define the flex container's main and cross axes. The default value is row nowrap.

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

# justify-content

This defines the alignment along the main axis. It helps distribute extra free space leftover when either all the flex items on a line are inflexible, or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line.

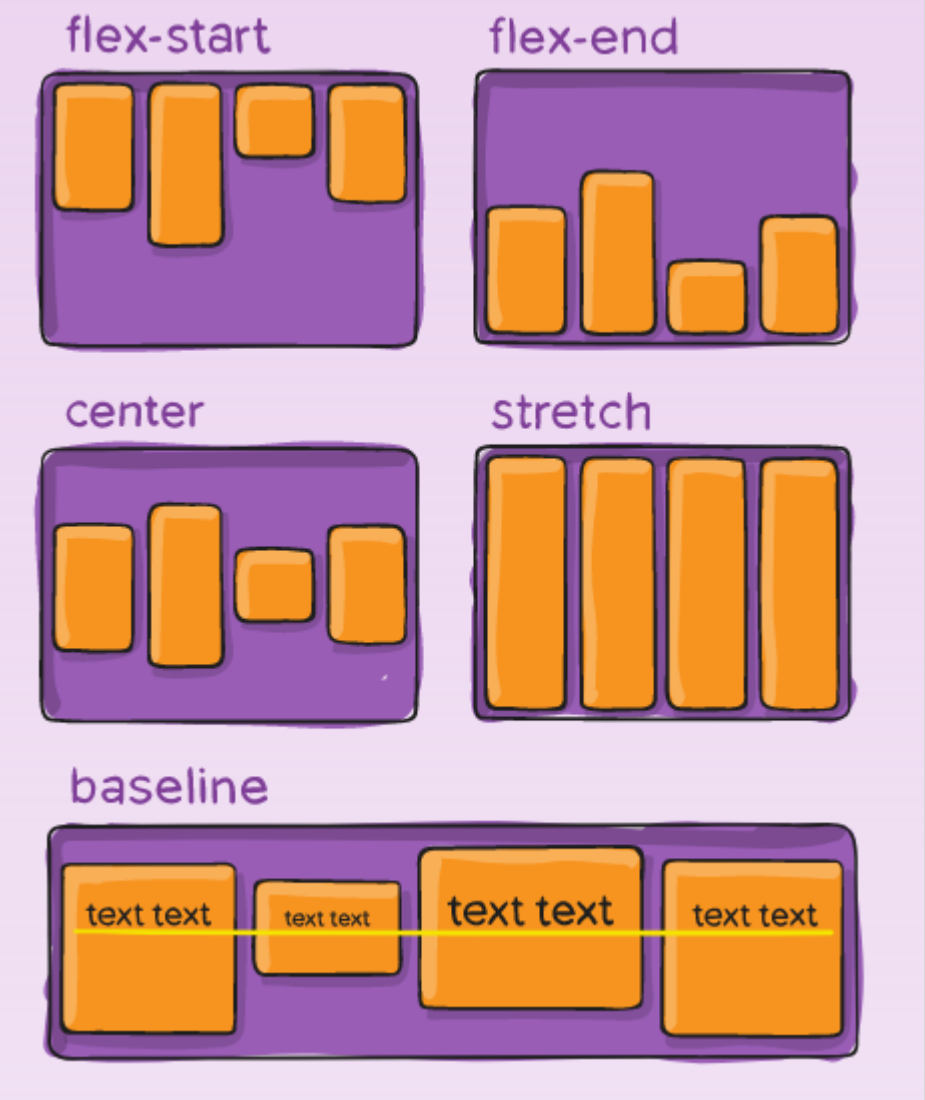
```css
.container {
  justify-content: flex-start | flex
}
```



# align-items

This defines the default behavior for how flex items are laid out along the cross axis on the current line. Think of it as the justify-content version for the cross-axis (perpendicular to the main-axis).

```css
.container {
  align-items: stretch | flex-start
}
```

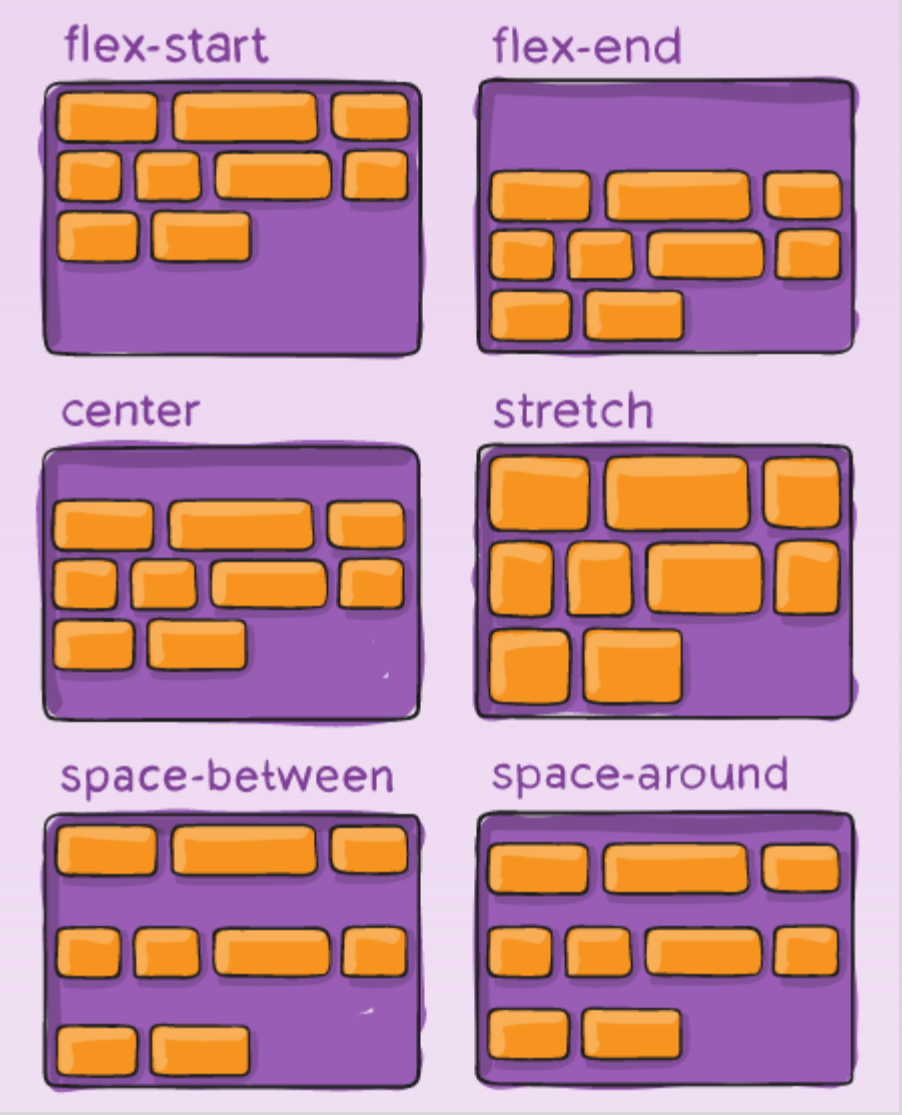


# align-content

This aligns a flex container's lines within when there is extra space in the cross-axis, similar to how justify-content aligns individual items within the main-axis.

```
.container {
  align-content: flex-start | flex-e
}
```

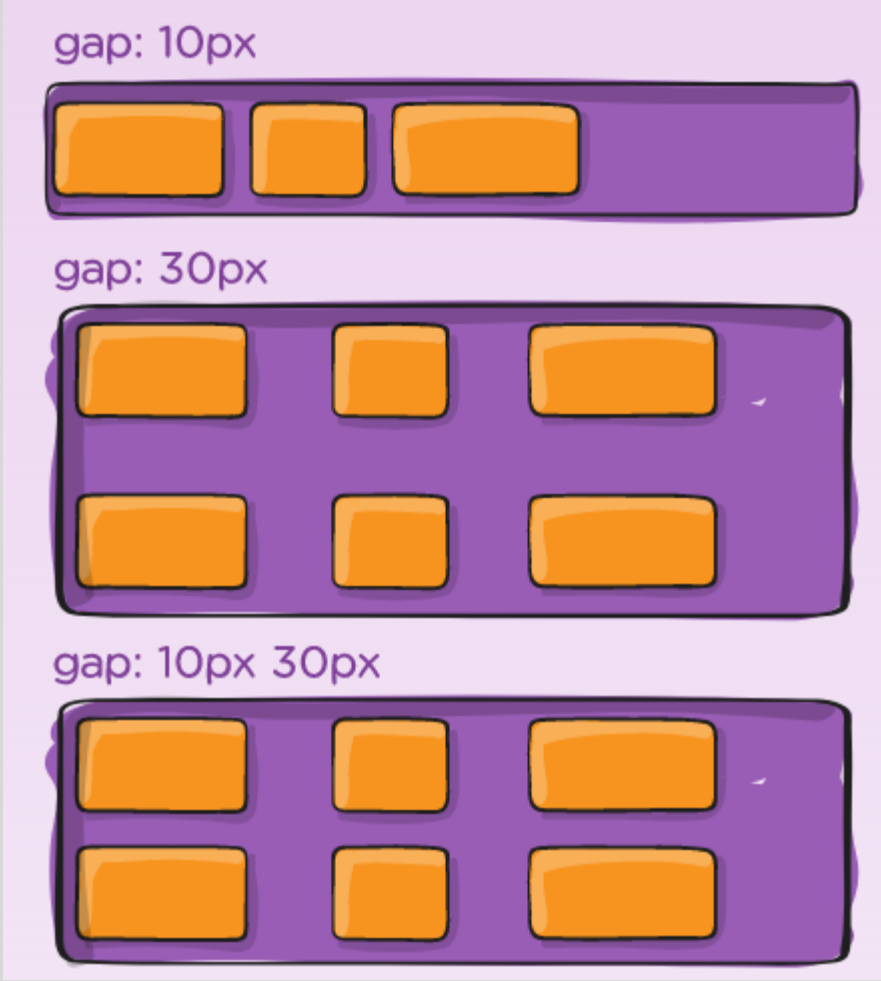
More explain



# gap, row-gap, column-gap

The gap property explicitly controls the space between flex items. It applies that spacing only between items not on the outer edges.

```css
.container {
  display: flex;
  ...
  gap: 10px;
  gap: 10px 20px; /* row-gap column
  row-gap: 10px;
  column-gap: 20px;
}
```

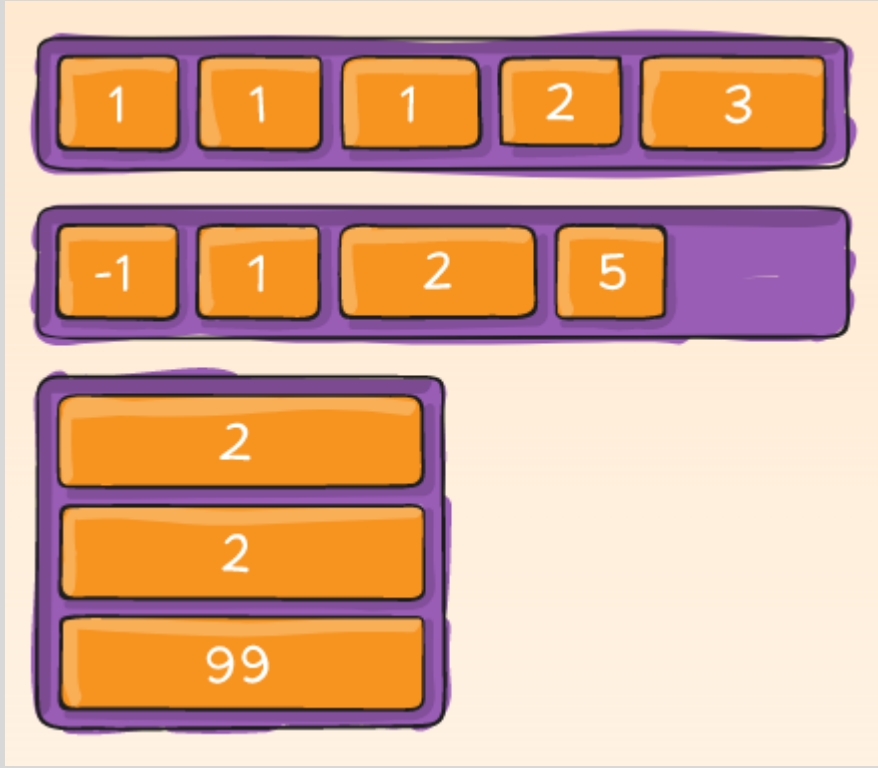


# Properties for (flex items)

- **order**

By default, flex items are laid out in the source order. However, the order property controls the order in which they appear in the flex container.

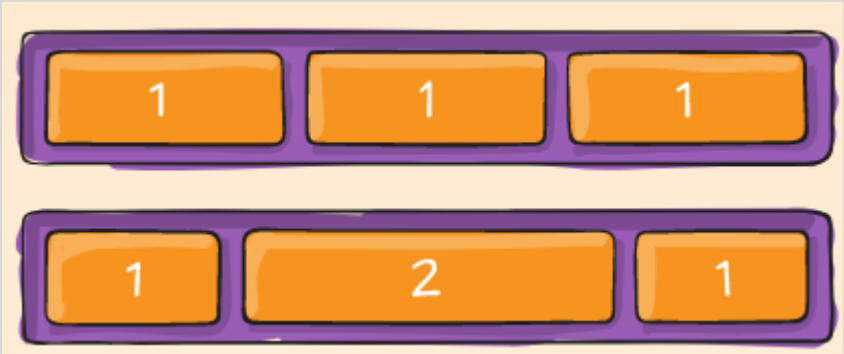
```css
.item {
  order: 5; /* default is 0 */
}
```




More explain

- **flex-grow**

This defines the ability for a flex item to grow if necessary. It accepts a unitless value that serves as a proportion. It dictates what amount of the available space inside the flex container the item should take up.




If all items have flex-grow set to 1, the remaining space in the container will be distributed equally to all children. If one of the children has a value of 2, that child would take up twice as much of the space either one of the others (or it will try, at least).

```css
.item {
  flex-grow: 4; /* default 0 */
}
```

- **flex-shrink**

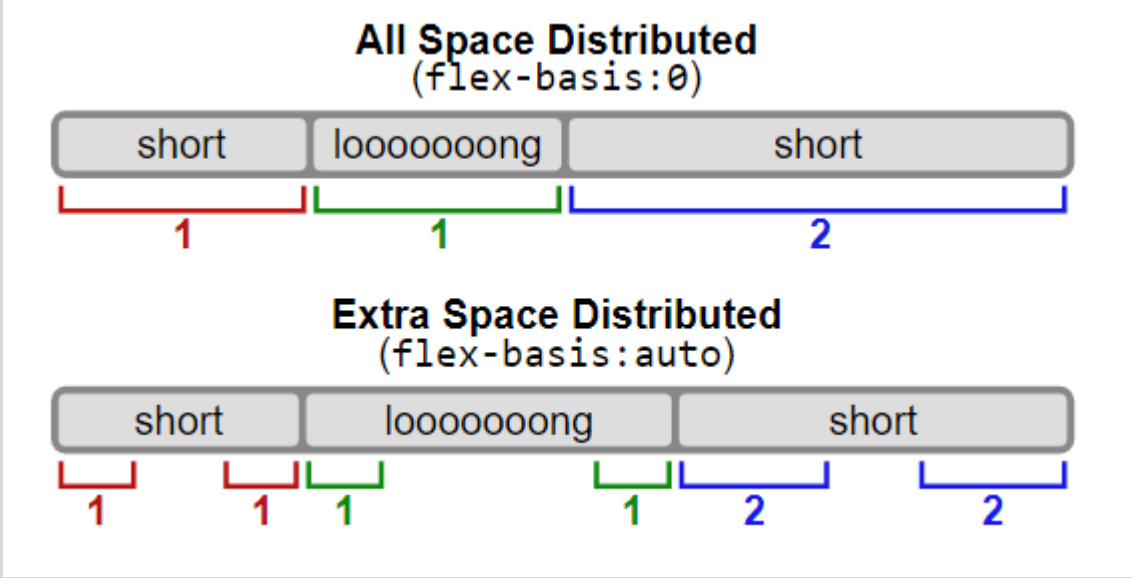
This defines the ability for a flex item to shrink if necessary.

```css
.item {
  flex-shrink: 3; /* default 1 */
}
```

- **flex-basis**

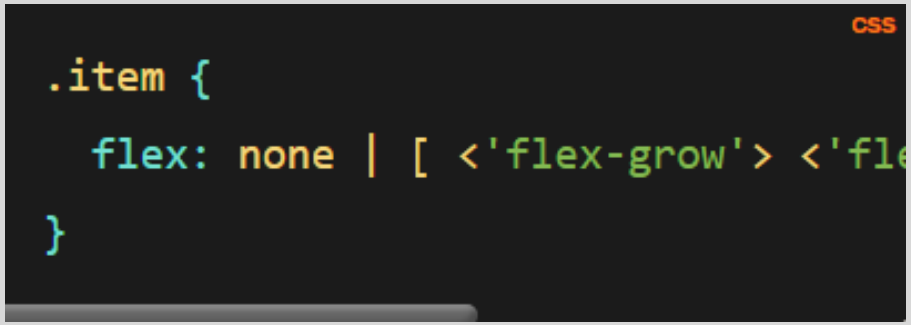
```css
.item {
  flex-basis:  | auto; /* default au
}
```

If set to 0, the extra space around content isn't factored in. If set to auto, the extra space is distributed based on its flex-grow value.



- **flex**

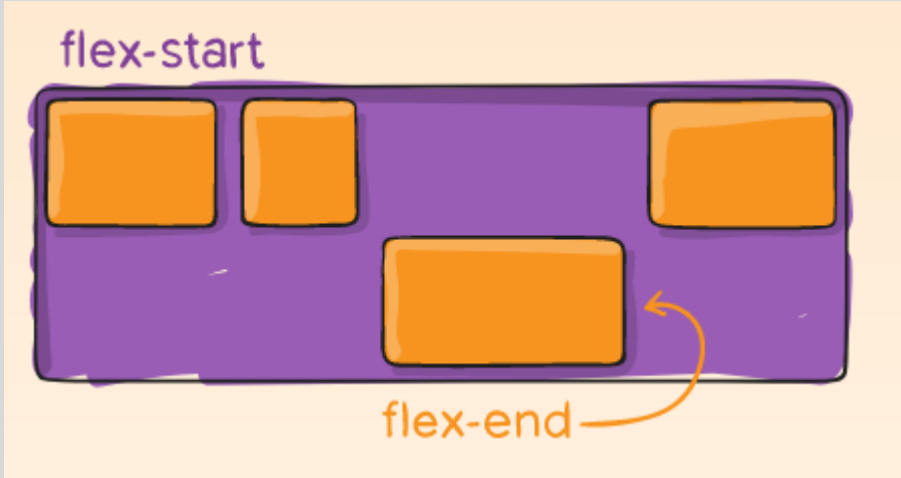
This is the shorthand for flex-grow, flex-shrink and flex-basis combined. The second and third parameters (flex-shrink and flex-basis) are optional.
The default is 0 1 auto, but if you set it with a single number value, like flex: 5;,
that changes the flex-basis to 0%, so it's like setting flex-grow: 5; flex-shrink: 1; flex-basis: 0%;



- **align-self**

This allows the default alignment (or the one specified by align-items) to be overridden for individual flex items.

```css
.item {
  align-self: auto | flex-start | fl
}
```



example for **CSS flexbox layout**