**The Flex Container**

A flexbox layout is applied to a container, by setting display value. It defines a flex container; inline or block depending on the given value. It enables a flex context for all its direct children.

.container {

display: flex; /\* or inline-flex \*/

}

### The Flex Container Properties

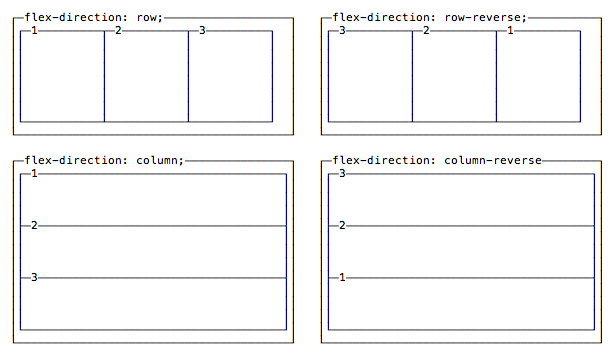
Flexbox properties apply to the container, which sets the general rules for its items. They are

* flex-direction
* justify-content
* align-items
* flex-wrap
* flex-flow

### Align rows or columns

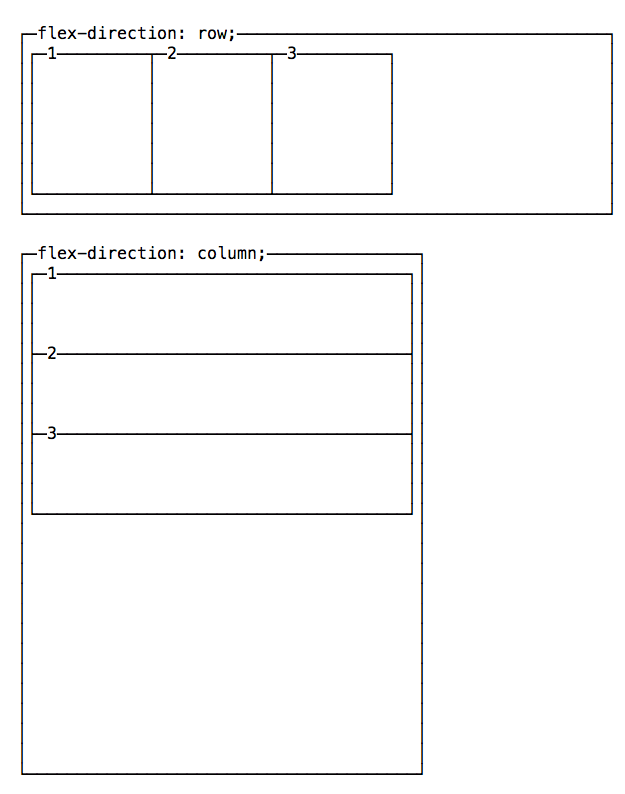
The first property we see, **flex-direction**, determines if the container should align its items as rows, or as columns:

* flex-direction: row places items as a **row**, in the text direction (left-to-right for western countries)
* flex-direction: row-reverse places items just like row but in the opposite direction
* flex-direction: column places items in a **column**, ordering top to bottom
* flex-direction: column-reverse places items in a column, just like column but in the opposite direction



### Vertical and horizontal alignment

By default items start from the left if flex-direction is row, and from the top if flex-direction is column.

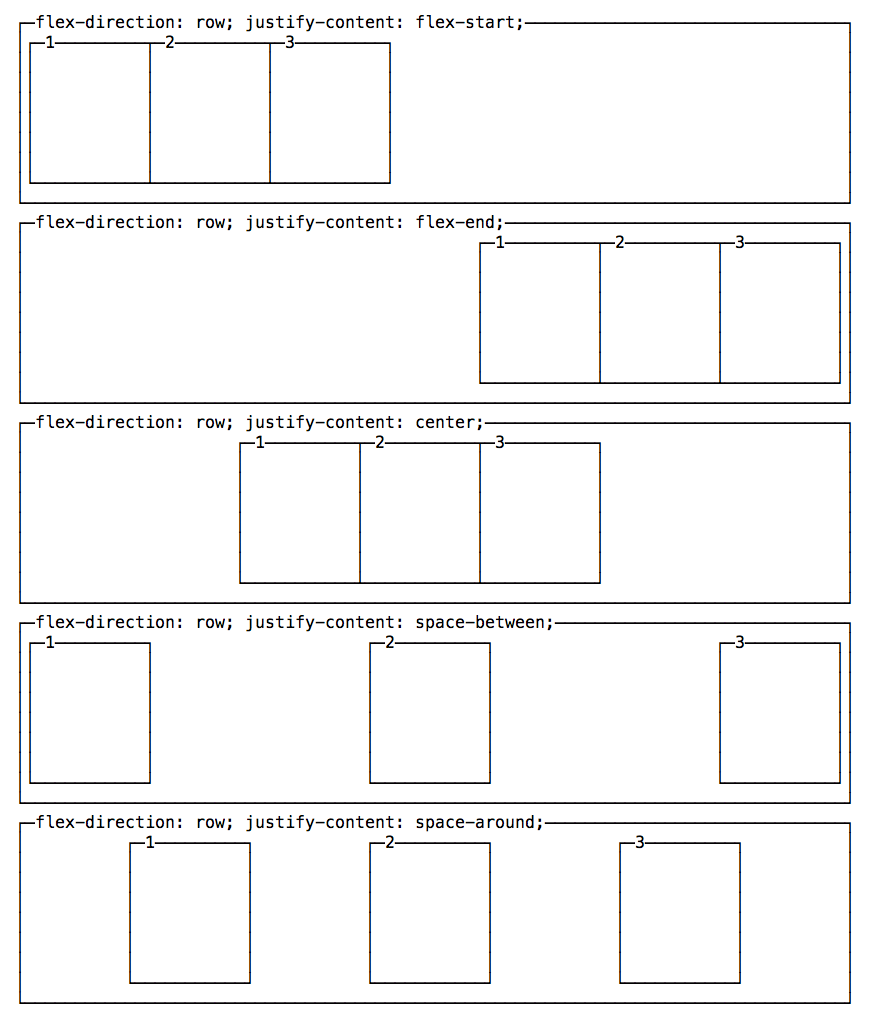


You can change this behavior using justify-content to change the horizontal alignment, and align-items to change the vertical alignment.

#### Change the horizontal alignment

**justify-content** has 5 possible values:

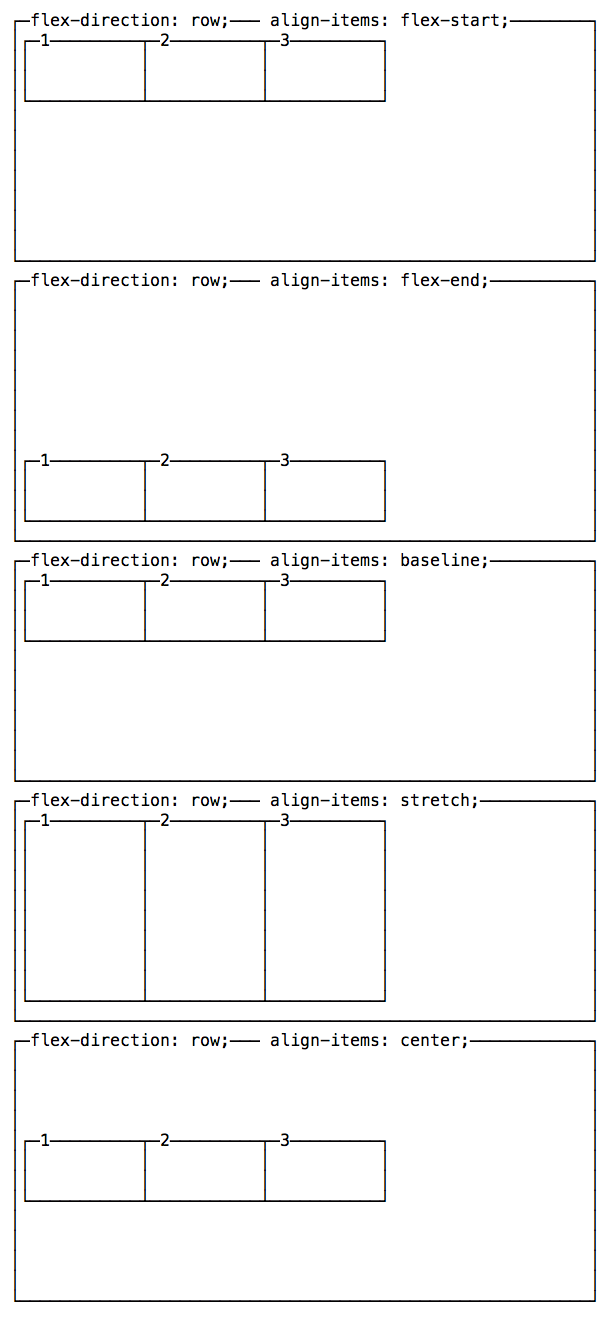
* flex-start: align to the left side of the container.
* flex-end: align to the right side of the container.
* center: align at the center of the container.
* space-between: display with equal spacing between them.
* space-around: display with equal spacing around them



#### Change the vertical alignment

**align-items** has 5 possible values:

* flex-start: align to the top of the container.
* flex-end: align to the bottom of the container.
* center: align at the vertical center of the container.
* baseline: display at the baseline of the container.
* stretch: items are stretched to fit the container.



**The Flex Item Properties**

Single items can have a certain amount of independence and flexibility, and you can alter their appearance using those properties:

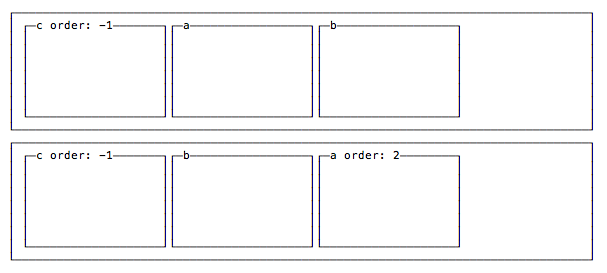
* order
* align-self
* flex-grow
* flex-shrink
* flex-basis
* flex

Let’s see them in detail.

**Moving items before / after another one using order**

Items are ordered based on a order they are assigned. By default every item has order 0 and the appearance in the HTML determines the final order.

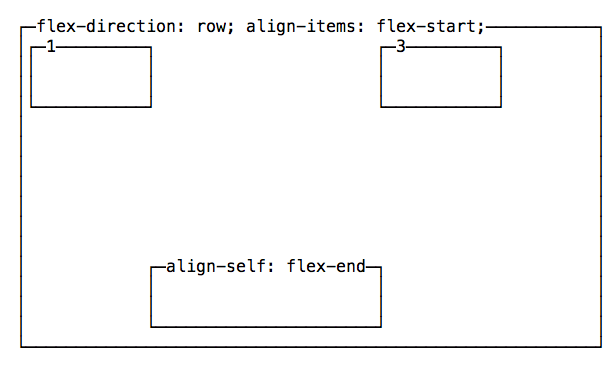
You can override this property using order on each separate item. This is a property you set on the item, not the container. You can make an item appear before all the others by setting a negative value.



**Vertical alignment using align-self**

An item can choose to **override** the container align-items setting, using **align-self**, which has the same 5 possible values of align-items:

* flex-start: align to the top of the container.
* flex-end: align to the bottom of the container.
* center: align at the vertical center of the container.
* baseline: display at the baseline of the container.
* stretch: items are stretched to fit the container.



### Grow or Shrink

##### flex-grow

The defaut for any item is 0.

If all items are defined as 1 and one is defined as 2, the bigger element will take the space of two “1” items.

##### flex-shrink

The defaut for any item is 1.

If all items are defined as 1 and one is defined as 3, the bigger element will shrink 3x the other ones. When less space is available, it will take 3x less space.

##### flex-basis

If set to auto, it sizes an item according to its width or height, and adds extra space based on the flex-grow property.

If set to 0, it does not add any extra space for the item when calculating the layout.

If you specify a pixel number value, it will use that as the length value (width or height depends if it’s a row or a column item)

##### flex

This property combines the above 3 properties:

* flex-grow
* flex-shrink
* flex-basis

and provides a shorthand syntax: flex: 0 1 auto

### Media Queries

A major component of responsive design is creating the right experience for the right device. With a gazillion different devices on the market, this can be a tall task. This is where media query come to help.

Media query is a CSS technique introduced in CSS3. Lets learn it through the video of how it works and can be used.

<https://youtu.be/y5ruVTGmj4Q>