

## First, What is Flexbox?

# What is Flexbox?

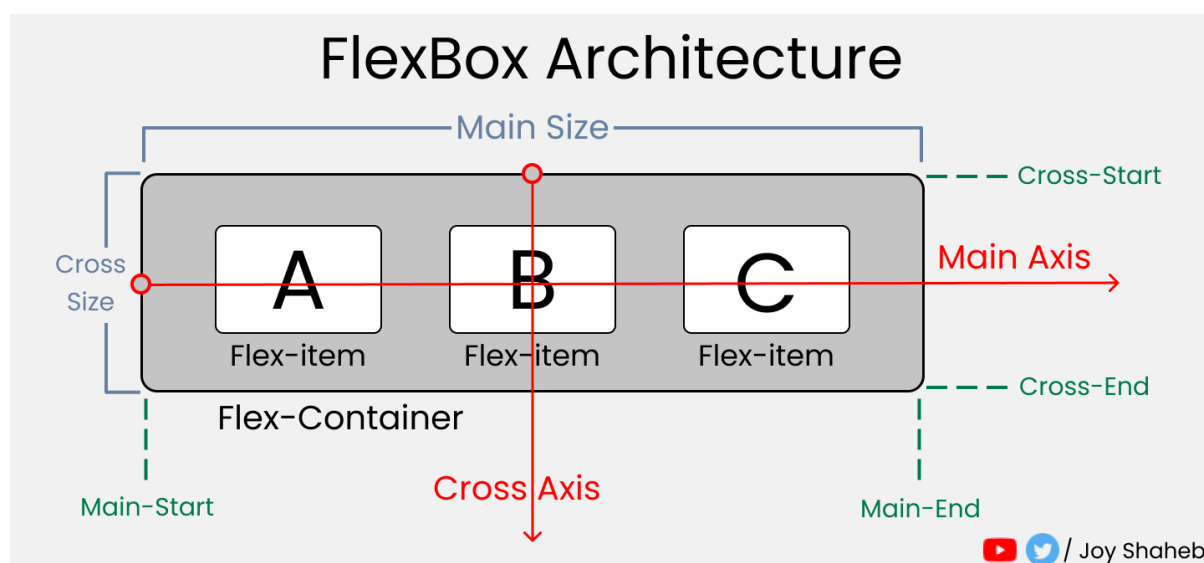


When you're building a house, you need a blueprint. In the same way, we need a blueprint when we're making websites. And Flexbox is the blueprint.

The Flexbox model allows us to **layout the content** of our website. Not only that, it helps us create the structures needed for creating **responsive websites** for multiple devices.

## Flexbox Architecture



So how does Flexbox architecture work? The flex-items [Contents] are distributed along the main axis and cross axis. And, depending on the flex-direction property, the layout position changes between rows and columns.



# Flexbox Chart

This chart contains every possible property and value you can use when you're working with Flexbox. You can reference it while doing your projects and experiment with different values.

Flex Box Chart	
Property	Value(s)
#1 display	flex
#2 flex-direction	row    column    column-reverse    row-reverse
#3 justify-content	flex-start    flex-end    center    space-between    space-around    space-evenly
#4 align-items	flex-start    flex-end    center    stretch    baseline
#5 align-content	flex-start    flex-end    center    stretch    space-between    space-around
#6 align-self	auto    flex-start    flex-end    center    baseline    stretch
#7 Order	/* Any positive Value */
#8 flex-grow	/* Any positive Value */
#9 flex-shrink	/* Any positive Value */
#10 flex-wrap	nowrap    wrap    wrap-reverse

  / Joy Shaheb

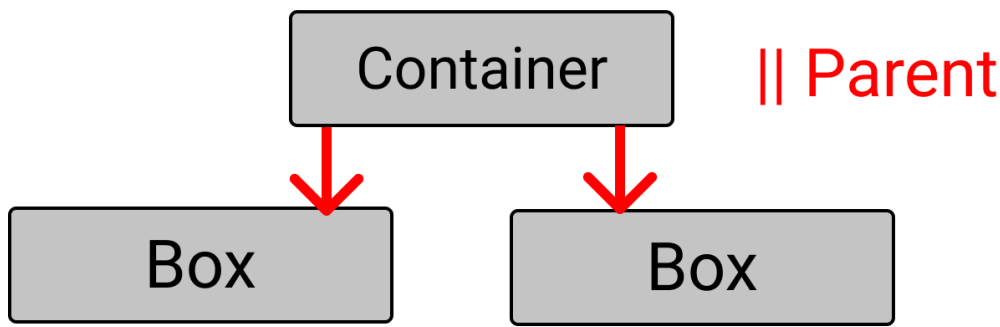
But Wait....

# Wait a Minute !



Before starting, you need to understand the relationship between parent and child classes.

# Flow



## Children

Flexbox works on the **parent class**, not on the child classes.

Here, the `.container` class is the **parent** and our `.box-*` classes are our **children**.

So, apply the `display: flex` inside the `.container` class. And place the letters at the center of the box like this:

```
.container{
  display : flex;
  height : 100vh;

  // To place some gap between boxes
  gap : 25px;
}

[class ^="box-"]{
  // Code from previous step are here

  // Placing text at center
  display : flex;
  justify-content : center;
  align-items : center;
}
```

And...we're all set! Let's start coding.

# flex-direction property

This property allows us to set the direction and orientation in which our flex-items should be distributed inside the flex-container.

flex-direction :

row



row-reverse



column

column-reverse

flex-direction



To recreate these results, let's write these lines in our CSS:

**Please note** that we'll write them inside the `.container` class.

```
.container{
```

```
//code from setup stage are here
```

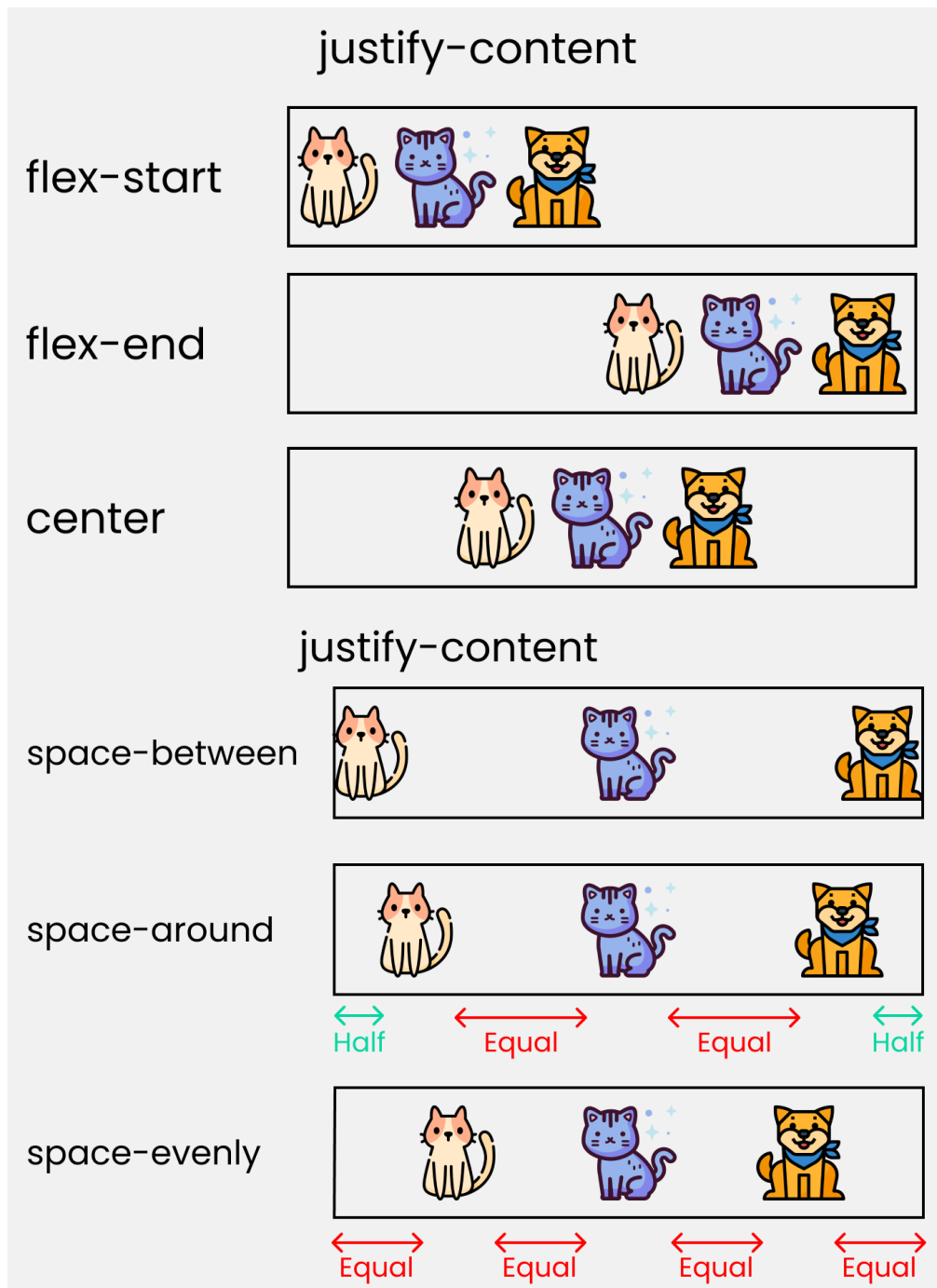
```
// Change the value 📌 here to see results
```

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

```
}
```

# justify-content property

This property arranges flex-items along the **MAIN AXIS** inside the flex-container.

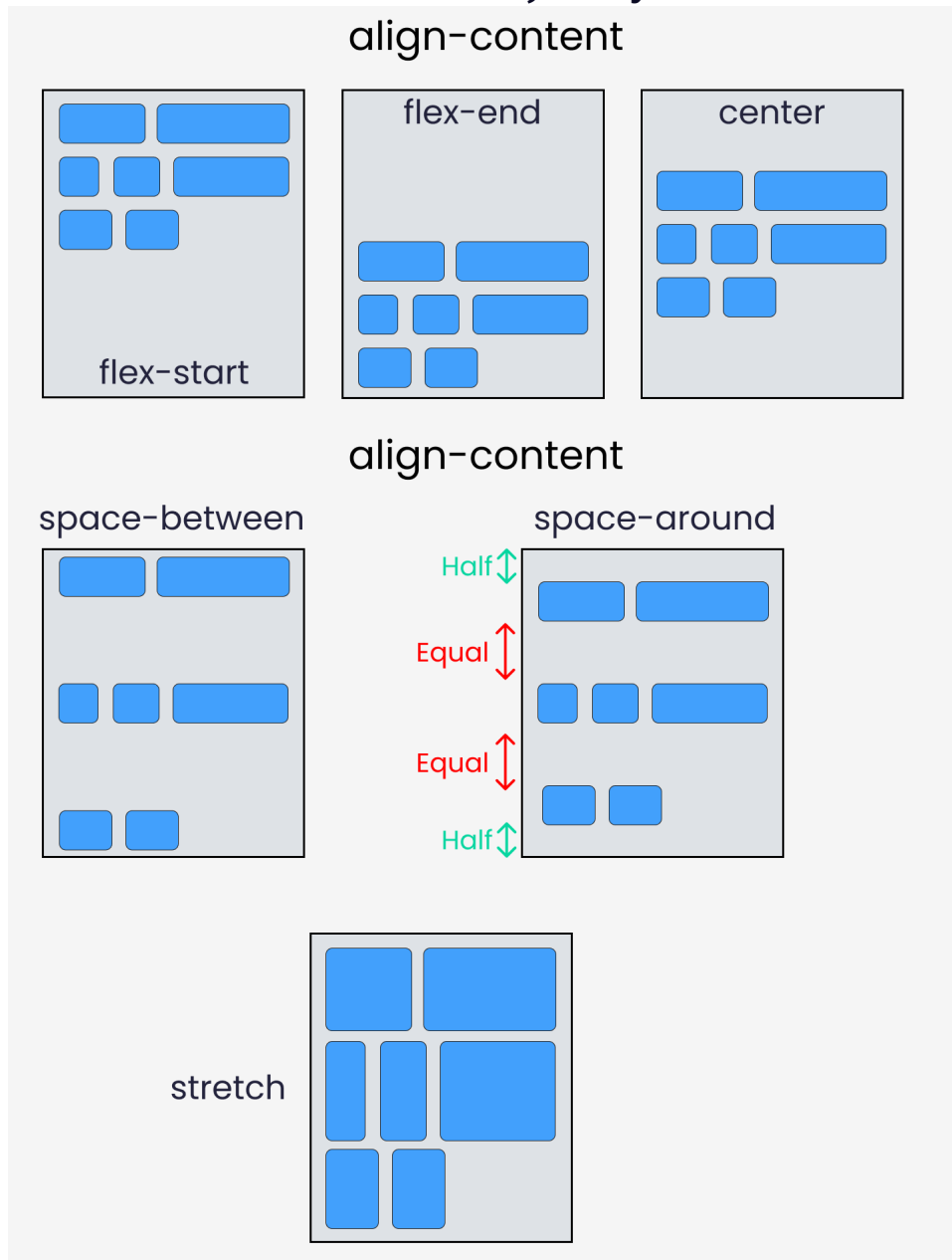


To recreate these results, write these lines in your CSS:

```
.container{  
  //code from setup stage are here  
  
  // Change the value 📌 here to see results  
  justify-content: flex-start;  
}
```

# align-content property

This property arranges flex-items along the **CROSS AXIS** inside the flex-container. This is similar to **justify-content**.

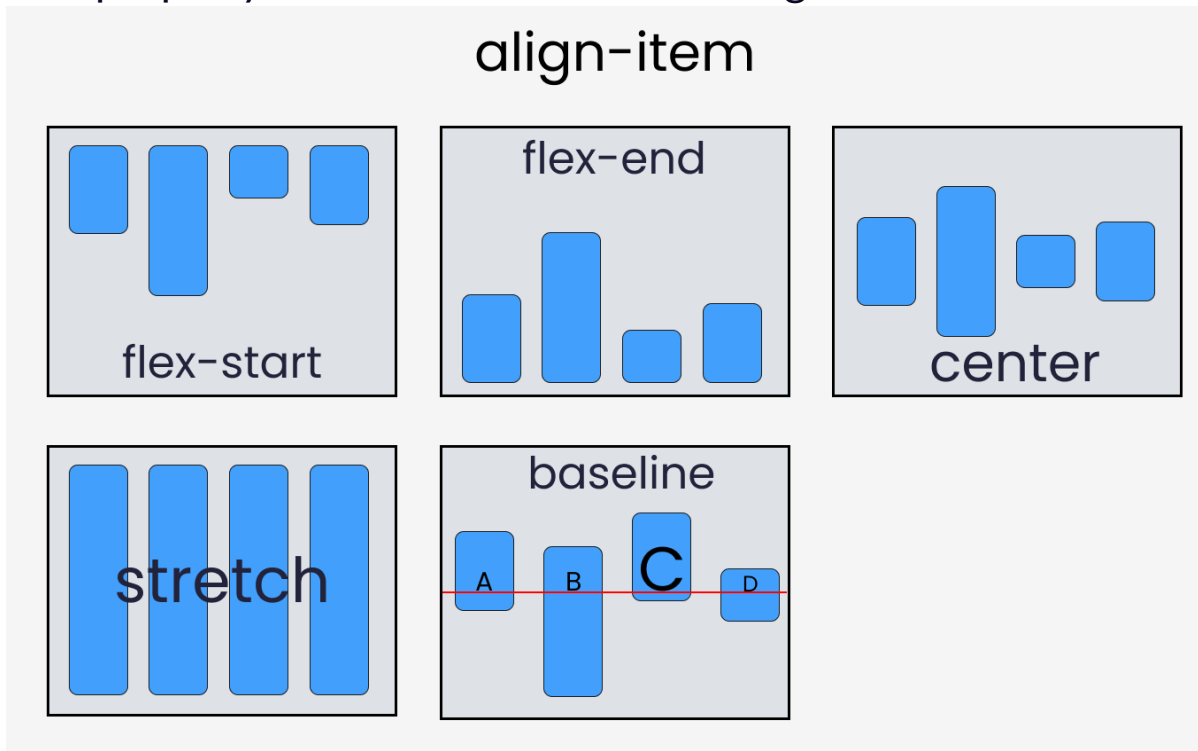


Please note that without the **flex-wrap** property, this property doesn't work. Here's a demo:

```
.container{  
  
  // Change the value 📌 here to see results  
  align-content: center;  
  
  // without this line, align-content won't work  
  flex-wrap: wrap;  
}
```

# align-items property

This property distributes Flex-items along the **Cross Axis**.



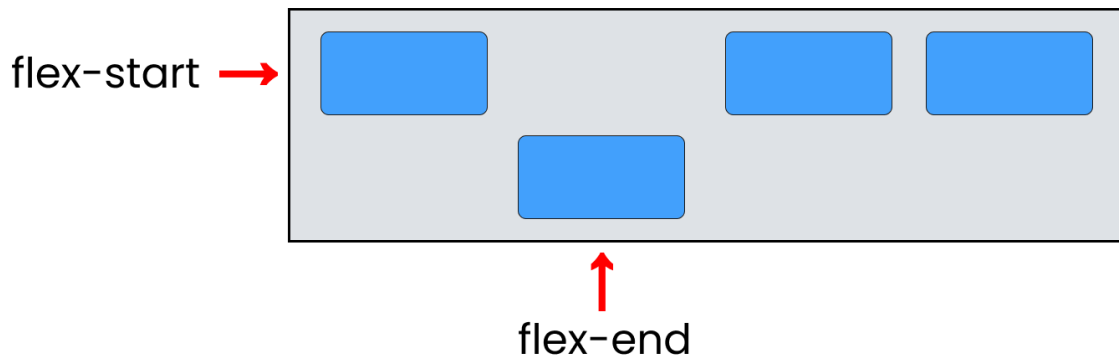
To recreate these results, let's write the following code in CSS:

```
.container{  
  //code from setup stage are here  
  
  // Change the value 📌 here to see results  
  align-items: flex-end;  
}
```

# align-self property

This property works on the child classes. It positions the selected item along the **Cross Axis**.

## align-self



In total we have 6 values:

- flex-start
- flex-end
- center
- baseline
- stretch
- auto

To recreate the results, select any `.box-*` and write the following code:

```
.box-2{  
  // Change the value 📌 here to see results  
  align-self : center;  
}
```



# flex - grow | shrink | wrap | basis properties

The properties we'll discuss now will work when we resize the window. Let's dive right in.

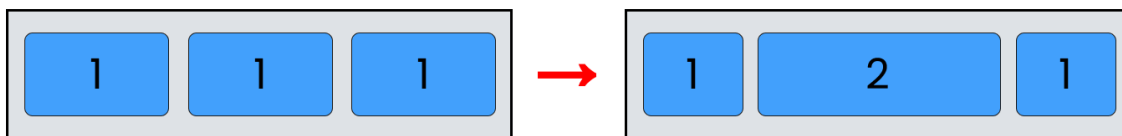
## flex-grow

This property grows the size of a flex-item based on the width of the flex-container.

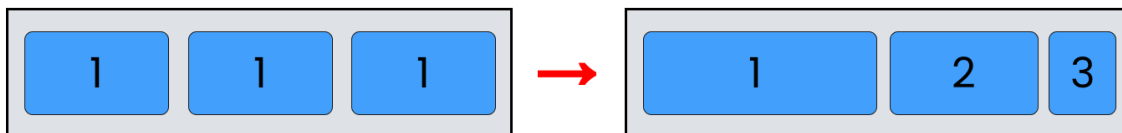
## flex-shrink

This property helps a flex item shrink based on the width of the flex-container. It's the opposite of flex-grow.

### flex-grow



### flex-shrink



To achieve these results, follow me.

**Please note** that flex-grow and flex-shrink work on child classes. So, we will target all our boxes like this:

```
.box-1{  
  flex-grow: 1;  
}  
.box-2{  
  flex-grow: 5;  
}  
.box-1{  
  flex-grow: 1;  
}
```

Resize the window and you'll see the results.

To duplicate the result of flex-shrink, write the following code:

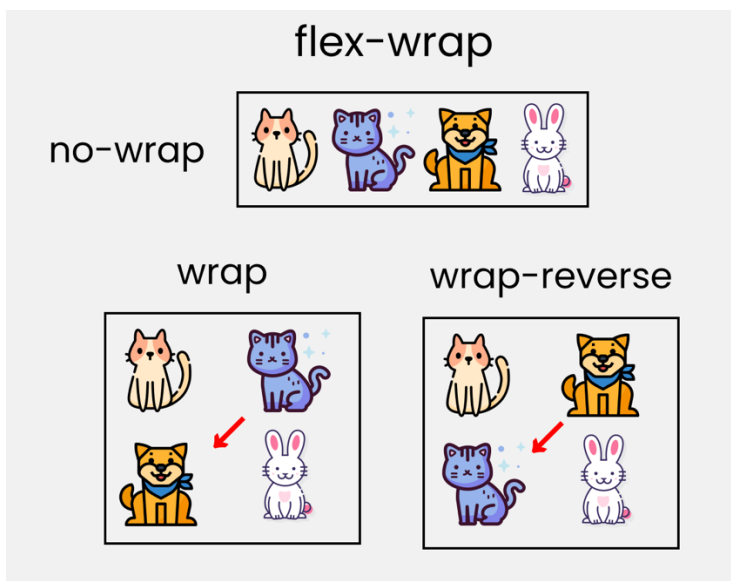
**Please note** that you need to delete the flex-wrap property first, otherwise it won't work.

```
.box-1{
  flex-shrink: 1;
}
.box-2{
  flex-shrink: 5;
}
.box-1{
  flex-shrink: 1;
}
```

Now, resize the window and you'll see the results.

## flex-wrap

This property helps you set the number of flex-items you want in a line or row.



This works on the `.container` parent class. So, write the following code:

```
.container{
  //other codes are here

  // Change value 📌 here to see results
  flex-wrap : wrap;
```

## flex-basis

This is similar to adding width to a flex-item, but only more flexible. flex-basis: 10em, for example, will set the initial size of a flex-item to 10em. Its final size will be based on the available space, flex-grow, and flex-shrink.

# Shorthand Flexbox Properties

## flex shorthand

This is the shorthand for the **flex-grow**, **flex-shrink** and **flex-basis** properties combined.

flex-grow  
flex-basis  
flex : 2 1 30em ;  
flex-shrink

You can try this by writing the following code:

**Please note** that it only works on the child classes:

```
.box-2{  
  flex : 2 1 30em;  
}
```

## flex-flow

This is the shorthand for the **flex-direction** and **flex-wrap** properties:

flex-wrap  
flex-flow : row wrap;  
flex-direction

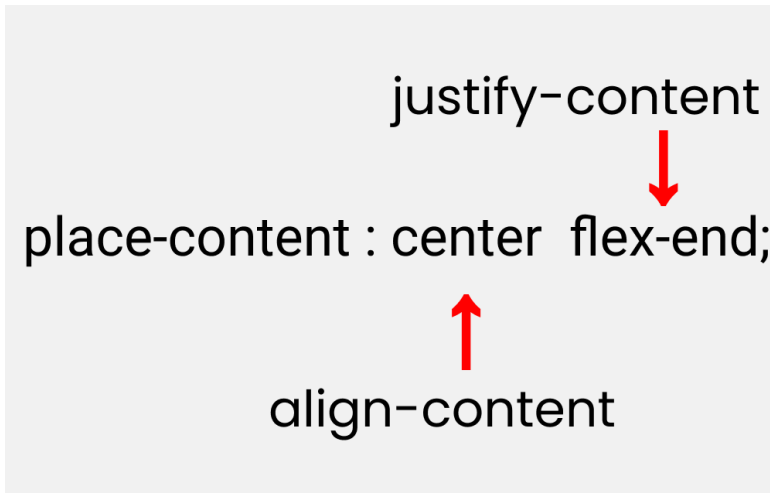
You can try this by writing the following code:

**Please note** that it only works on the parent class.

```
.container{  
  flex-flow : row wrap;  
}
```

## place-content

This is the shorthand for the justify-content and align-content properties:



Let's duplicate the results:

**Please note** that it works on the parent class.

```
.container{  
  place-content : center flex-end;  
}
```