# **FLEXBOX** CHEATSHEET


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

To "activate" Flexbox, you simply add **`display: flex;`** to an element.

All of this element's direct children then become *flex items* and can be controlled by the various Flexbox properties. These properties won't work without this first step.

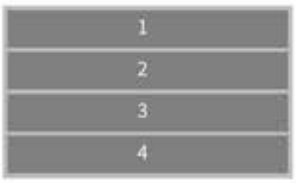
The majority of these Flexbox properties are applied to the container, unless otherwise stated.

***Note:*** *this is all you need to do to achieve equal height columns due to the default values of the Flexbox properties.*

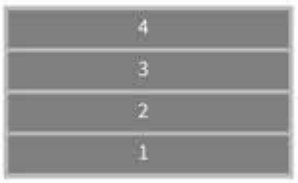
## flex-direction


```
/* DEFAULT VALUE */
flex-direction: row;
```


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




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




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

## flex-wrap


```
.container {
  /* DEFAULT VALUE */
  flex-wrap: nowrap;
}
```

```
.box-child {
  width: 25%;
}
```

By default, flex items will all be forced onto one single row.

You can see that even though the boxes on the left have been given a width of 25%, this is ignored and they remain on one row.

To override this behaviour, we can use **`flex-wrap: wrap;`**


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

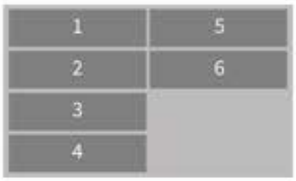
You can also make the elements align to the bottom and wrap upwards using the **`wrap-reverse`** value.


```
flex-wrap: wrap-reverse;
```

## flex-flow

The **`flex-flow`** property is a shorthand property for specifying both the **`flex-direction`** and **`flex-wrap`** properties.

For example...




```
flex-flow: column wrap;
```

## justify-content

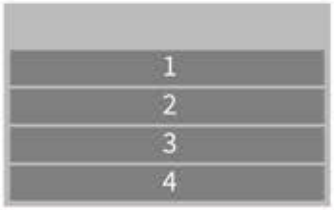
***Note:*** *this property behaves differently when used with flex-direction: column; More info below*


```
/* DEFAULT VALUE */
justify-content: flex-start;
```

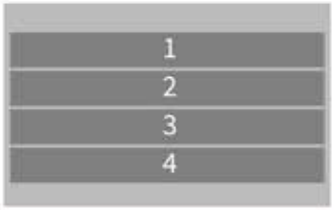

```
justify-content: flex-end;
```


```
justify-content: center;
```


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




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




```
justify-content: center;
```

By default, this property aligns elements horizontally.

However, you can see on the left that when using **`flex-direction: column;`** this property aligns elements vertically.


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

## align-items

***Note:*** *this property behaves differently when used with flex-direction: column; More info below*


```
/* DEFAULT VALUE */
align-items: stretch;
```


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


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

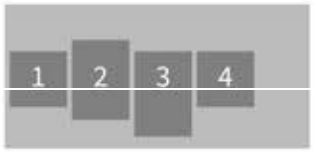

```
align-items: center;
```


```
align-items: center;
/* using flex-direction: column; */
```

The default value of **`stretch`** is what gives flex items equal height automatically (or equal width when using **`flex-direction: column;`**).

This property basically aligns flex items vertically, unless you are using **`flex-direction: column;`**

***Note:*** *EASY VERTICAL CENTERING!!!*




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

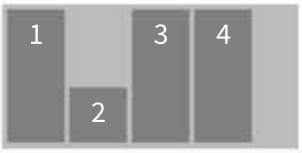
## align-self

***Note:*** *this property is applied to a flex item, NOT the container*

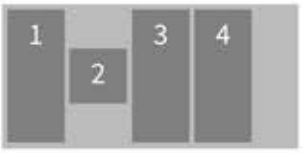
This property is the same as **`align-items`** and uses the same values, but is applied to a flex item rather than the container.

This is to allow you to align a particular element differently to the rest if needed. For example...


```
.box2 {
  align-self: flex-start;
}
```




```
.box2 {
  align-self: flex-end;
}
```




```
.box2 {
  align-self: center;
}
```

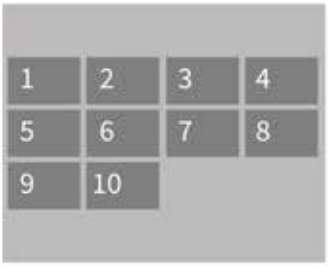
## align-content

This property is used on flex containers that contain multiple rows; i.e. if you have used **`flex-wrap: wrap;`** to allow elements to wrap onto multiple rows.

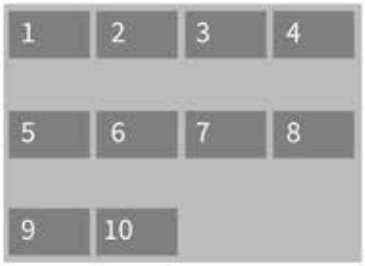

```
/* DEFAULT VALUE */
align-content: stretch;
```


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

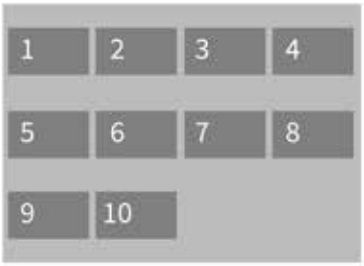

```
align-content: flex-end;
```




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




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




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

## `order`

***Note:*** *this property is applied to a flex item, NOT the container*

This property allows you to change the order of your flex items, regardless of their order in the source code.

It is placed on the actual flex item rather than the container.

It accepts any integer values (whole numbers), including minus values.

The concept is similar to **`z-index`**.


```
.box-child {
  /* DEFAULT VALUE */
  order: 0;
}
```


```
.box3 {
  order: -1;
}
```


```
.box3 {
  order: 1;
}
```

## `flex`

***Note:*** *this property is applied to a flex item, NOT the container*

This property allows you to apply flexible widths to your flex items and ensures they fill the entire container.

It can get very complicated as it is actually a shorthand for 3 different properties; **`flex-grow`** **`flex-shrink`** and **`flex-basis`**.

This section will explain the most common ways in which you might use the **`flex`** property.


```
.box-child {
  flex: 1;
}
```

All elements will fill the entire width of the container and take up exactly the same amount of space.


```
.box-child {
  flex: 1;
}

.box2 {
  flex: 2;
}
```

The space in this container is basically split into 5 segments. Boxes 1, 3 and 4 get 1/5 of the available space and box 2 gets 2/5 of the available space.

**`flex: 1;`** ensures each element takes up exactly the same amount of space


```
.box-child {
  flex: 1;
}
```

**`flex: auto;`** ensures each element has the exact same amount of left/right padding (great for navs!)


```
.box-child {
  flex: auto;
}
```

## `margin: auto;`

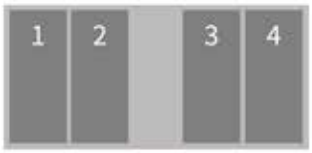
***Note:*** *this property is applied to a flex item, NOT the container*

The **`margin: auto;`** rule takes on a whole new power when used on flex items.

It allows you to control where left over space in your container is distributed.


```
.box1 {
  margin-left: auto;
}
```


```
.box2 {
  margin-left: auto;
}
```




```
.box3 {
  margin-left: auto;
}
```

Just **`margin: auto;`** is all that is needed to center a flex item both horizontally *and* vertically!


```
.box1 {
  margin: auto;
}
```

Aligning items to the left/right with the remaining space in the middle is now a simple task with Flexbox.


```
.nav li:nth-child(3) {
  margin-left: auto;
}
```

## Browser Support

• Full un-prefixed Flexbox support is now available in ALL current major browser versions.

• Some recent WebKit based browsers require the **`-webkit-`** prefix.

• IE9 and below does NOT support Flexbox at all

• IE10 has full support, but only for an old version with slightly different property/value names (which you can see below) and requires the **`-ms-`** prefix

• IE11 and above has full unprefixed support

### `display: flex;`

```
display: -webkit-flex;
display: -ms-flexbox;
display: flex;
```

### `flex-direction`

```
-webkit-flex-direction
-ms-flex-direction
flex-direction
```

### `flex-wrap`

```
-webkit-flex-wrap
-ms-flex-wrap
flex-wrap
```

### `flex-flow`

```
-webkit-flex-flow
-ms-flex-flow
flex-flow
```

### `order`

```
-webkit-order
-ms-flex-order
order
```

### `flex`

```
-webkit-flex
-ms-flex
flex
```

### `justify-content`

```
-webkit-justify-content
-ms-flex-pack
justify-content
```

| Possible values (standard) | Possible values (IE10) |
|---|---|
| `flex-start` | `start` |
| `flex-end` | `end` |
| `center` | `center` |
| `space-between` | `justify` |
| `space-around` | `distribute` |

### `align-content`

```
-webkit-align-content
-ms-flex-line-pack
align-content
```

| Possible values (standard) | Possible values (IE10) |
|---|---|
| `stretch` | `stretch` |
| `flex-start` | `start` |
| `flex-end` | `end` |
| `center` | `center` |
| `space-between` | `justify` |
| `space-around` | `distribute` |

### `align-items`

```
-webkit-align-items
-ms-flex-align
align-items
```

| Possible values (standard) | Possible values (IE10) |
|---|---|
| `stretch` | `stretch` |
| `flex-start` | `start` |
| `flex-end` | `end` |
| `center` | `center` |
| `baseline` | `baseline` |

### `align-self`

```
-webkit-align-self
-ms-flex-item-align
align-self
```

| Possible values (standard) | Possible values (IE10) |
|---|---|
| `stretch` | `stretch` |
| `flex-start` | `start` |
| `flex-end` | `end` |
| `center` | `center` |
| `baseline` | `baseline` |

## Dealing with IE9

IE9 (and below if you still need to support the older IE versions) is the only real reason we can't go mad and use Flexbox for our entire layout framework.

Until this browser has faded into insignificance, it's best to limit Flexbox usage to certain areas where you can 'progressively enhance' using Flexbox.

For example, if you have a row of items that are all different heights, you can use Flexbox to easily make them all the same height. The fact that they won't in IE9 generally isn't a major issue.

Also, Flexbox will completely ignore/override any floats. So you can freely use float properties as a fallback for IE9 on any flex containers or flex items.

CSS Table Layout could also be a viable fallback for certain Flexbox features like alignment.

Modernizr could be useful in certain cases where you need to specifically target browsers that do/don't support Flexbox.