CSS FLEX

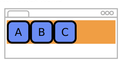
by linkedin.com/in/josenoevg

What is FLEX?
Flexible Model (aka flexbox or flex for short) is a CSS **layout** model for positioning elements called **items** inside a parent called **container** (actually the names "container" and "item" do not matter, they are just flex jargon to understand/explain the properties).

PROPERTIES FOR CONTAINERS

```
display: <flex | inline-flex>
```

When setting the display property to **flex** or **inline-flex**, the element becomes a flex container, and all direct children become **flex items**

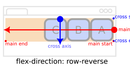


inline-flex is useful if we want other elements to share the horizontal space, e.g. place multiple flex containers next to each other.

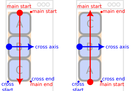
```
flex-direction: <row(default) | row-reverse | column | column-reverse>
```

flex-direction sets the **main axis AND its direction**. **Items will follow the main axis from its start to its end.**

flex-direction: row



flex-direction: row-reverse



flex-direction: column

flex-direction: column-reverse

Note how the items distribute and order along the main axis.
It is important to know the main axis because other properties take effect in this specific axis.

```
justify-content: <flex-start(default) | flex-end | center | space-around | space-between | space-evenly>
```

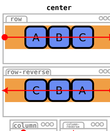
justify-content specifies how items spread **along the main axis**.

In the following images the **height** was set for the **flex-direction: column** examples, otherwise you will not see the effect because the height (by default) is the same as the sum of the items height.

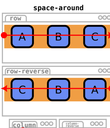
flex-start

flex-end

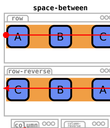
center



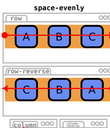
space-around



space-between



space-evenly

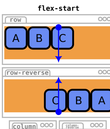



```
align-items: <flex-start(default) | flex-end | center | baseline | stretch>
```

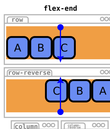
align-items aligns items on the **cross axis.**

In the following images the **height** and **width** were set (except for stretch examples), otherwise you will not see the effect.

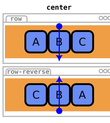
flex-start



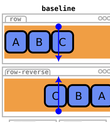
flex-end



center

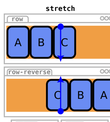


baseline



baseline aligns items using the text baseline

stretch



row and row-reverse examples does NOT have a height set in items

column and column-reverse examples does NOT have a width set in items

Note for stretch: if the item does **not** have a height set or if the item **has** a min-height set then the item will stretch if possible

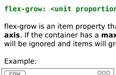
PROPERTIES FOR ITEMS

By this part you already know how the flex-direction affect some properties, next examples will only use the default option (row) for simplicity.

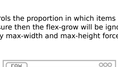
```
flex-grow: <unit proportion>
```

flex-grow is an item property that controls the proportion in which items will grow when the container grows in the **main axis**. If the container has a **max-** measure then the flex-grow will be ignored (note: height, min-height, width, and min-width will be ignored and items will grow, only max-width and max-height force items to have a max measure ignoring flex-grow).

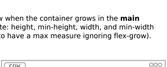
Example:



In this example we have a container that expans 100% horizontally (that is the default when the display is flex), and the items does **not** have a width specified, therefore the width is the same as the content width. NOTE: the main axis is horizontal from left to right



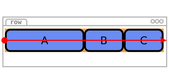
If we set the flex-grow property in all items (not in the container, this an item property remenber) to the value of 1, then all items will stretch in the same proportion to cover the 100% of the container's width.



We can confirm this by resizing the window, we will see how the items grow. REMEMBER: flex-grow affects the main axis, and the main axis is defined by flex-direction in the container.

In the example above all items have this property: flex-grow **1**.
1 in all items means they will grow in the same proportion (1/3).

Another example:
.itemA {flex-grow: 4;}
.itemB {flex-grow: 1;}
.itemC {flex-grow: 1;}
This means itemA will grow in a proportion of 4/6 and other items will grow with a 1/6 proportion.



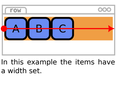
NOTE:
The default value for flex-grow is 0

Column containers will grow their items vertically (because that is the main axis)

```
flex-shrink: <unit proportion>
```

flex-shrink specifies how the item will shrink relative to the rest of the flexible items inside the same container. The default value is 1. An item will shrink unless the min-width property is set (with width the item will shrink).

Example:



In this example the items have a width set.

If we resize the window the items will shrink because the default shrink is 1.

If we modify the proportions we can make an item(s) shrink faster than others, in the example above:
.itemA {flex-shrink:3}
.itemB {flex-shrink:2}
.itemC {flex-shrink:1}
Remember, this happens when we resize the container and its width is shorter that the sum of the items width.

```
flex-basis: <unit proportion>
```

flex-basis specifies specifies the initial length of an item. Of course if you set a basis that exceeds the available width of the container then the item will shrink proportionally (unless you set shrink to 0, then it will overflow the container).

```
flex: <flex-grow value>   <flex-shrink value>   <flex-basis value>
```

flex is a shorthand to write these three values in one single line.

Example:
flex: 1 3 50px;
In this example flex-grow is 1, flex-shrink is 3 and flex-basis is 50px;

PROPERTIES FOR CONTAINERS

```
flex-wrap: <wrap| wrap-reverse | nowrap(default)>
```

flex-wrap is used to indicate if the items will shrink or move to the next line.

Example using nowrap

Example using wrap

Example using wrap-reverse


```
align-content: <flex-start(default) | flex-end | center | space-between | space-around | stretch>
```

If items will wrap then you can use align-content to align the items in the cross axis.

```
flex-flow: <flex-wrap value> <flex-direction value>
```

This is just a shorthand.

Example:
flex-flow: column wrap;