skillcrush

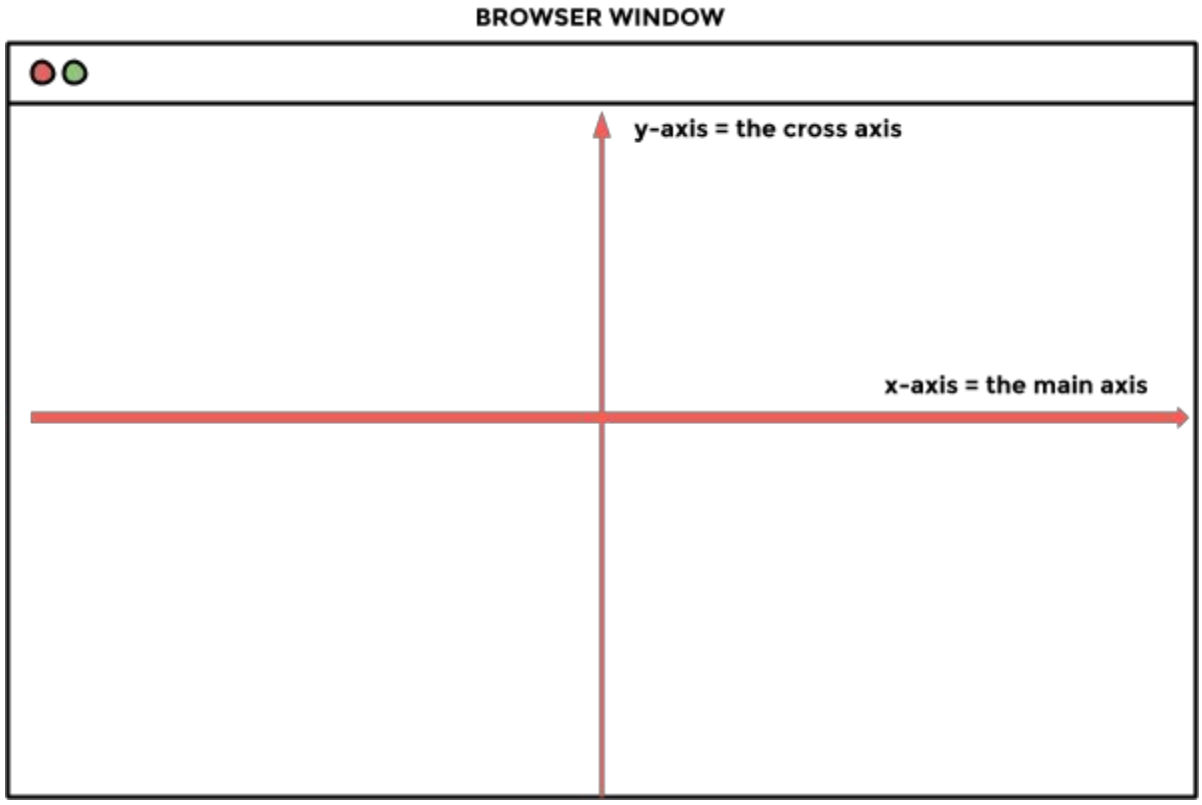
THE BIG BAD GUIDE TO THE FLEXBOX MODEL

INTRO

Flexbox is a whole set of CSS properties that work together to allow for much more flexible and fluid layouts and are specifically intended to help us developers make our websites work on all manner of devices. To be 100% clear, flexbox isn't just ONE property, but a whole collection of properties that work together, which you will learn about in MUCH greater detail in the rest of this guide.

A key feature of flexbox is that, unlike display:block; which is *vertically oriented*, and display:inline; which is *horizontally oriented*, flexbox is both horizontally AND vertically oriented.

This means that flexbox is designed around the idea of having two axes, like a graph:



The fact that flexbox has two axises makes a WHOLE bunch of fancy things possible, but in particular it allows for three things:

1. Flexbox, it's like text align for layout elements

By now, you are no doubt familiar with using the text-align property for your content, right?

You can center align, left align, right align, and justify text. Plus you can modify how your text handles line breaks and super long words.

Well NOW you can do the same with your HTML elements!

2. Flexbox, it let's you write styles for how things behave vertically...

Since flexbox works on the main and cross axes, that means you can ALSO set how things behave vertically. Meaning...you want everything to be centered vertically? Yep, flexbox can do that.

How about top aligned? Bottom aligned? Stretched to fill the full vertical height? No problem!

3. Flexbox, makes it possible to change the order of HTML elements!

Are you sick of repeating certain elements of your site just so you can hide them on desktop but show them on mobile?

You know, like when a nav needs to move from over here to over there? *Never again!*

ALL THE FLEXBOX PROPERTIES

---

display

In order to turn on flexbox all you need to do is write:

```
display: flex;
```

What this does is turn the element in question into a flex container, and all it's child elements into flex-items. You mean it's *that* easy?! Yep, this is all you need to do to use the following properties to start wielding flexbox's power.

justify-content

This is the property that will tell your content how it should be aligned *horizontally*, or along the main axis. To get started write:

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

All the possible justify-content property values include:

```
justify-content:
flex-start|flex-end|center|space-between|space-around|initial|inherit;
```

Now the question is, how do you want to align your elements?

Left aligning content is handled by using **flex-start** which will position elements at the beginning of the container.

Right aligning content is handled by using **flex-end** which will position elements at the end of the container.

Centering content is handled by using **center** which will position elements at the center of the container.

How about making sure your content is evenly spaced out with even space in between each element or even space around each element? *No problem.*

Space-between handles aligning your elements across your container with an *even amount of space in between*.

Meanwhile, **space-around** makes sure your elements have an *even amount of space* before, between AND after your elements.

What about those two remaining values - **initial** and **inherit**?

Those act in the way their names suggest. **Initial** uses the default value and **inherit** makes the element take on the positioning of it's parent element.

align-items

This property acts pretty much exactly like justify-content, except that it aligns your elements *vertically*! Long gone are days of having to find just the right combination of a few properties to get your content to vertically align - this property takes care of it for you all in one line. Pretty sweet.

Get started by adding the following:

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

All the possible **align-items** property values include:

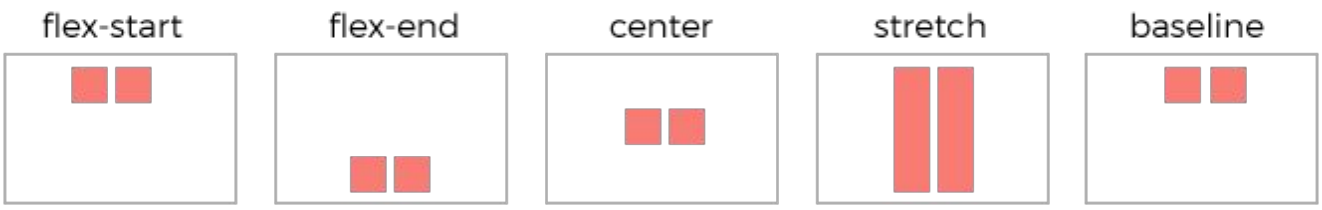
```
align-items: stretch|center|flex-start|flex-end|baseline|initial|inherit;
```

In the align-items property, the values **center, flex-start, flex-end, initial and inherit** function in the same way as they do for justify content, but instead of operating horizontally they are oriented vertically.

The values unique to align-items are **stretch** and **baseline.**

Baseline positions items at the containers baseline which is often the top of the container.

Meanwhile, **stretch** elements are stretched vertically to fill the container.



align-content

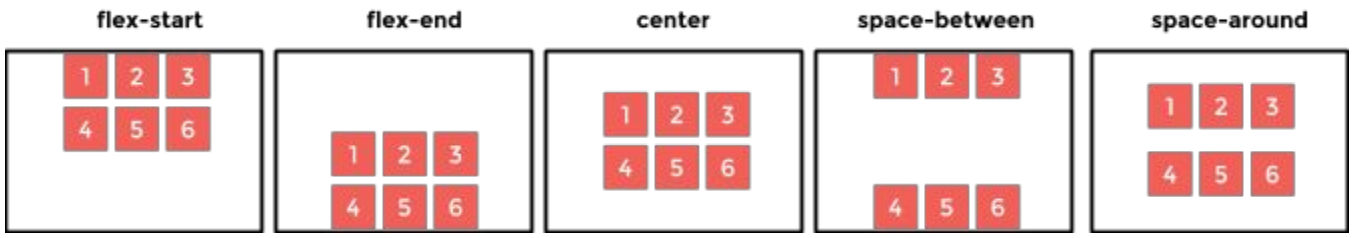
The property **align-content** and it's values work in the same way as align-items and it's values.

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

All the possible align-content property values include:

```
align-content:
stretch|center|flex-start|flex-end|space-between|space-around|initial|
inherit;
```

The align-content property can be a source of confusion in the flexbox model. So, to combat confusion, one of the ways to get familiar with this property and when it's best to use. *Align-items* is best to use when wanting to align a row of content vertically, while *align-content* is often used to align wrapped content vertically. Let's see some examples with wrapped content below:



flex-wrap

The **flex-wrap** property dictates whether the items should wrap or not.

```
flex-wrap: nowrap;
```

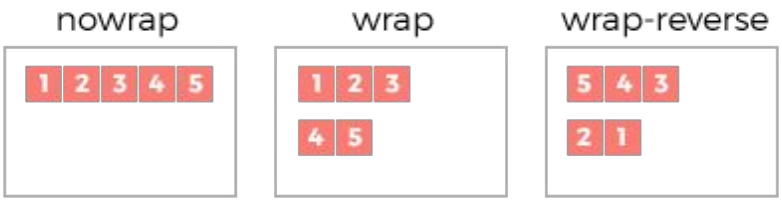
All the possible flex-wrap property values include:

```
flex-wrap: nowrap|wrap|wrap-reverse|initial|inherit;
```

The **nowrap** value tells the items not to wrap, while the **wrap** value tells them to wrap.

The **wrap-reverse** value tells the items to wrap in the reverse order.

Meanwhile, **inherit** and **initial** function in the same way as with other elements, either *inheriting* the parent's value or using the *initial* default value.



flex-direction

The **flex-direction** property dictates the direction of the items.

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

All the possible flex-direction property values include:

```
flex-direction: row|row-reverse|column|column-reverse|initial|inherit;
```

The value **row** makes the items display horizontally, in a row, while **row-reverse** displays them in a row but in reverse order. The value **column** makes the items display vertically in a column, while the value **column-reverse** makes them display in a column but in the reverse order.

Meanwhile, **inherit** and **initial** function in the same way as with other elements, either *inheriting* the parent's value or using the *initial* default value.

---

flex-flow

The **flex-flow** property is a shorthand property that combines the values for the **flex-direction** and the **flex-wrap** properties into one line of CSS, like so:

```
flex-flow: flex-direction flex-wrap;
```

An example of the flex-flow property in use:

```
flex-flow: row wrap;
```

flex-grow & flex-shrink

All the possible flex-grow and flex-shrink property values:

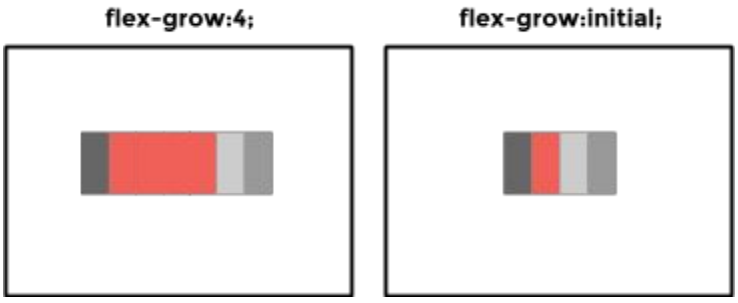
```
flex-grow: number|initial|inherit;
flex-shrink: number|initial|inherit;
```

The **flex-grow** property dictates how the item will grow in relation to the rest of the items inside the container. The **flex-shrink** property dictates how the item will shrink in relation to the rest of the items inside the container.

The **number** dictates how much the item will grow or shrink in relation to the other items in the container, the default is 0 for *flex-grow* and 1 for *flex-shrink*.

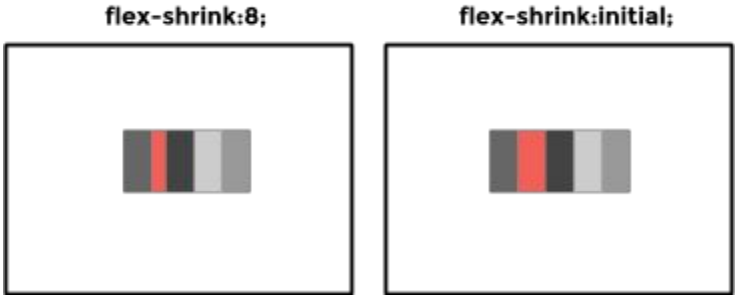
Flex-grow in action:

```
flex-grow:4; vs flex-grow:initial;
```



Flex-shrink in action:

```
flex-shrink:8; vs flex-shrink:initial;
```



flex-basis

The **flex-basis** property dictates the initial length of a flexible item. Which is the fancy way flexbox says it's setting the width of an item.

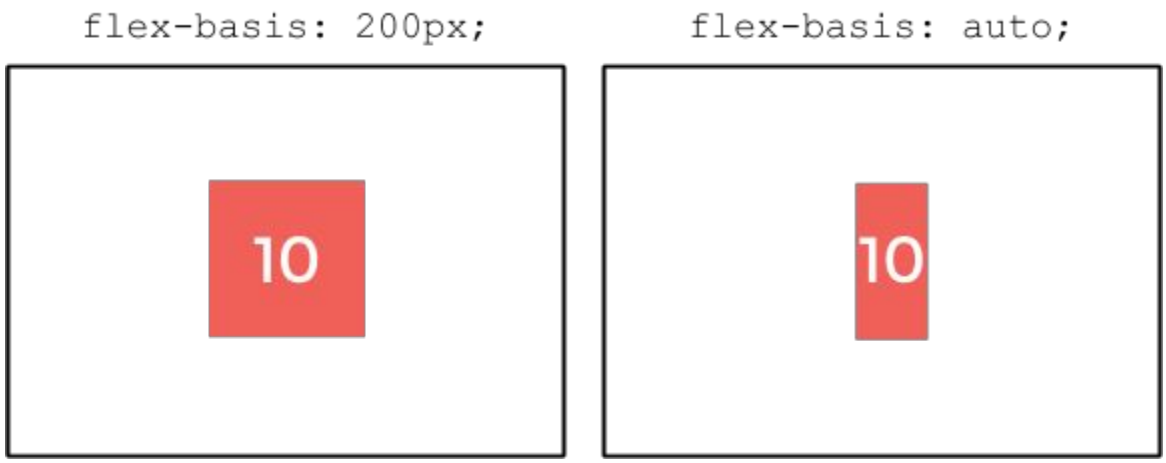
```
flex-basis: 200px;
```

All the possible flex-basis property values:

```
flex-basis: number|auto|initial|inherit;
```

The **number** in this case is a unit of length, or a percentage, that gives the items initial length.

If there is no length to the item, when set to **auto** the length will be determined by the items content.



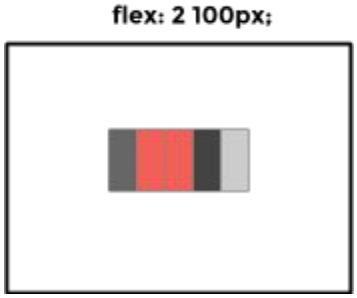
flex

The **flex** property dictates the length of the item, relative to the rest of the items inside the container. It is a shorthand value for the *flex-grow, flex-shrink*, and the *flex-basis* properties.

An example of flex in use:

```
flex: 2 100px;
/*is the same as*/
flex-grow: 2;
flex-basis: 100px;
```

So when the CSS above is applied to one div, it would look like this:



All the possible flex property values:

```
flex: flex-grow flex-shrink flex-basis|auto|inherit|initial;
```

FLEXBOX IN ACTION

So by now you might be starting to get the feeling that flexbox is pretty cool and useful, but you might be thinking - “All of these *sound* like a great shortcuts, but how do I actually use them?”.

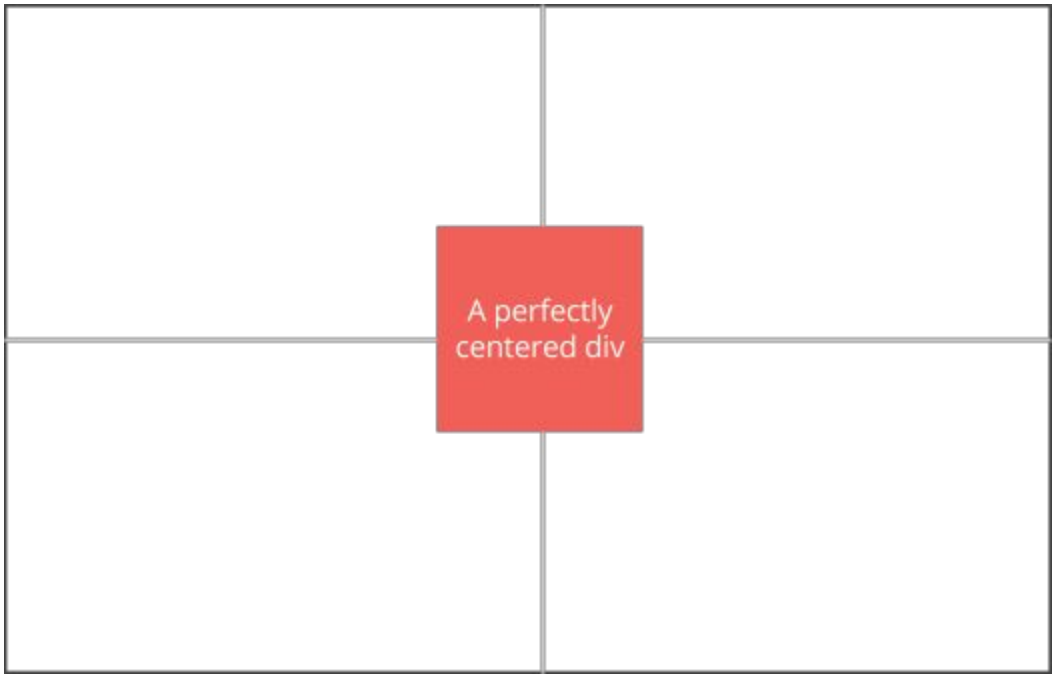
Let’s take a look some simple but powerful examples of how you can put them to use to make your life easier.

You can horizontally & vertically align a div:

The css:

```
.body {
  display:flex; /*This makes all body elements flex items*/
}
.div {
  justify-content:center;
  align-items:center;
}
```

The result:

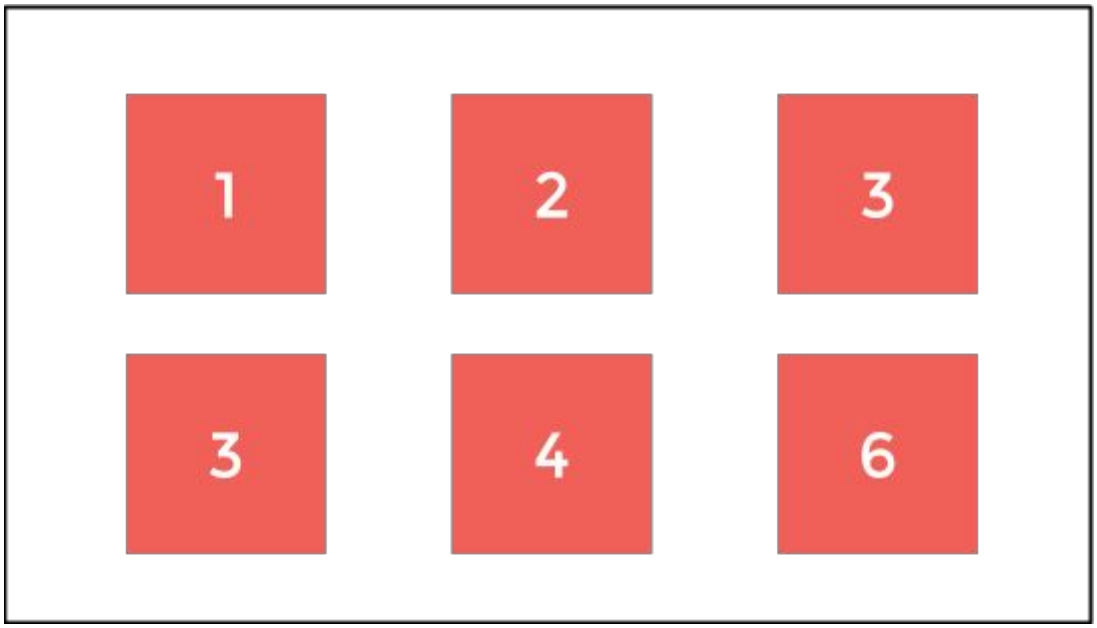


You can wrap a list of 6 evenly sized items:

The css:

```
ul.flex-container {
  display:flex; /*This makes the element flexible*/
  flex-flow: row wrap:
  justify-content: space-around;
}
li.flex-item {
  width:200px;
  height:200px;
  background-color:#f16059;
}
```

The result:



BROWSER SUPPORT

This is a brief chart breaking down which browser versions **FULLY** support the flexbox model at the moment. There are some browsers and mobile operating systems partially supporting it as well, and for a full breakdown, you can find it here.

IE	FIREFOX	CHROME	SAFARI	OPERA	iOS	ANDROID
11+	31+	31+	7+	30+	7.1+	4.4+

Now while this chart gives you the breakdown of all the browse *fully* supporting flexbox, there are ways to work around browsers that are only partially supporting flexbox properties by using prefixes! Here is a breakdown of prefixes per term that you just learned for those times you're *really* excited to use a certain property:

display

```
display: -webkit-box;   /* OLD: Safari,  iOS, Android browser, older
WebKit browsers.  */
display: -moz-box;      /* OLD: Firefox (buggy) */
display: -ms-flexbox;   /* MID: IE 10 */
display: -webkit-flex;  /* NEW, Chrome 21-28, Safari 6.1+ */
display: flex;          /* NEW: IE11, Chrome 29+, Opera 12.1+, Firefox
22+ */
```

justify-content

```
-webkit-box-pack: center; -moz-box-pack: center;
-ms-flex-pack: center;
-webkit-justify-content: center;
justify-content: center;
```

align-items

```
-webkit-box-align: center;
-moz-box-align: center; /* OLD… */
-ms-flex-align: center; /* You know the drill now… */
-webkit-align-items: center;
align-items: center;
```

align-content

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

flex-wrap

```
-webkit-box-lines: multiple | single
-moz-box-lines: multiple | single
-ms-flex-wrap: none | wrap | wrap-reverse
-webkit-flex-wrap: nowrap | wrap | wrap-reverse
flex-wrap: nowrap | wrap | wrap-reverse
```

flex-direction

```
-webkit-box-direction: normal;
-moz-box-direction: normal;
-webkit-flex-direction: row;
-ms-flex-direction: row;
flex-direction: row;
```

flex-flow

```
-ms-flex-flow: <-ms-flex-direction> || <-ms-flex-wrap>
-webkit-flex-flow: <flex-direction> || <flex-wrap>
flex-flow: <flex-direction> || <flex-wrap>
```

flex-grow & flex-shrink

```
-webkit-flex-grow: 1;
flex-grow: 1;

-webkit-flex-shrink: 0;
flex-shrink: 0;
```

flex-basis

```
-webkit-flex-basis: auto;
flex-basis: auto;
```

flex

```
-webkit-flex: 1 100px;
flex: 1 100px;
```