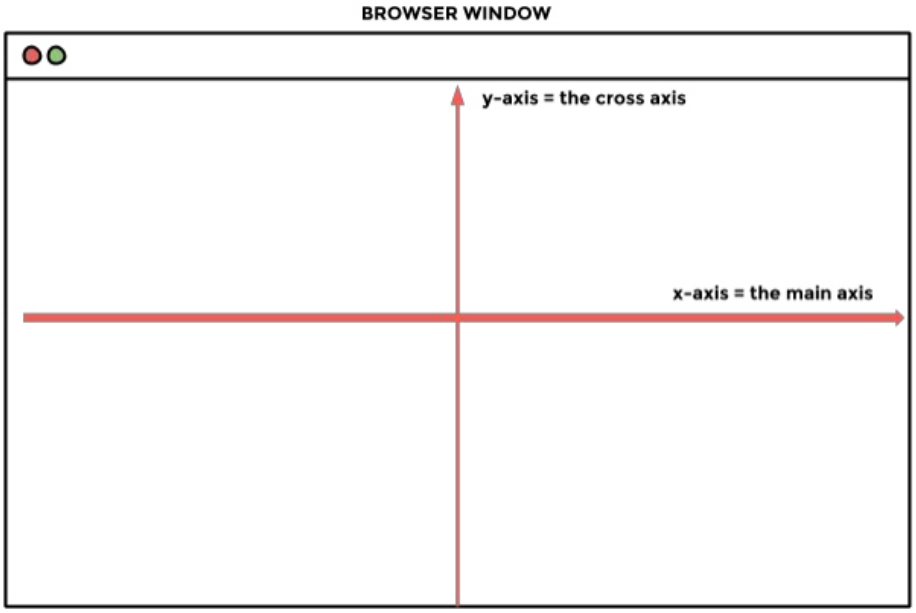
SKILLCRUSH 206 (HTTPS://LEARN.SKILLCRUSH.COM/CLASSES/CRUSH-WEB-DEVELOPMENT-WP-CW/) PREV (HTTPS://LEARN.SKILLCRUSH.COM/MODULE-6/FLEXBOX-INTRODUCTION/)

FLEXBOX

STEP 3 OF 7: THE BIG BAD GUIDE TO THE FLEXBOX MODEL

Flexbox is more than just one CSS property. It's a whole set of CSS properties that work together to allow for more flexible, fluid layouts. Flexbox is specifically intended to help make our websites work on all manner of devices. 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. It views the page as having two axes, like a graph:



Flexbox aligns elements horizontally and vertically along the main and cross axes - the former won't always be X, the latter not always Y..

The fact that flexbox has two axes makes a whole bunch of fancy things possible, such as:

1. **Flexbox is like text-align for layout elements.**
   By now, you're 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 lets 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. Want everything to be centered vertically? 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 just so you can hide them on desktop but show them on mobile? Like moving a nav from here to there? Never again!

SKILLCRUSH 206 (HTTPS://LEARN.SKILLCRUSH.COM/CLASSES/RESPONSIVE-WEB-DEVELOPMENT-WP-CW/) (HTTPS://LEARN.SKILLCRUSH.COM/MODULE-6/FLEXBOX-INTRODUCTION/)

We should emphasize that the direction of the main and cross axes depends on the flex-direction property. By default, the flex-direction property is set to 'row' (horizontal), which means that the main axis will be horizontal and the cross axis vertical. However, if you set the flex-direction to 'column', the main axis will be vertical and the cross-axis will be horizontal.

This is significant because many flexbox properties rely on one of the two axes. For example, 'justify-content' will determine the position of elements **along the main axis**. On the other hand, 'align-items' will determine the position of elements **along the cross axis**. This is why the values are 'flex-start' and 'flex-end' (and not 'flex-top' or 'flex-right') because the rule's effect on the position of the element will vary according to which axis we are working with.

COMMON FLEXBOX PROPERTIES

display

To turn on flexbox, write:

```
display: flex;
```

This turns the element in question into a flex container and its first-level child elements into flex-items. Easy, huh? Now you can start using flexbox!

MAIN AXIS ALIGNMENT

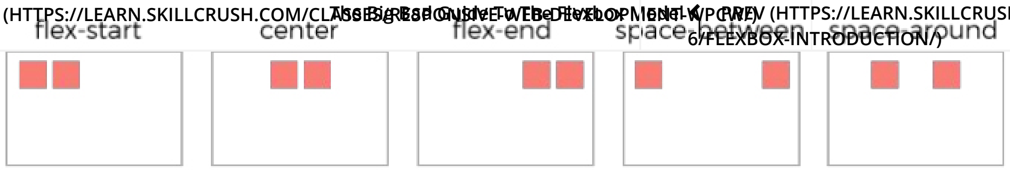
justify-content

The **justify-content** property tells your content how it should be aligned along the main axis. To get started with some centered content, write:

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

How do you want to align your elements along the main axis? All the possible justify-content property values include:

- **flex-start** - Left-aligned content is handled by the flex-start property value, which positions elements at the beginning of the container. If our main axis is horizontal, it will position it to the left. It if is vertical, it will position it at the top.
- **center -** Centering content is handled by using center, which positions elements in the center of the container.
- **flex-end** - Right-aligned content is handled by using flex-end, which positions elements at the end of the container. If our main axis is horizontal, it will position it to the right. If our main axis is vertical, it will position it at the bottom.
- **space-between -** aligns your elements across the container along the main axis with an even amount of space in between each one.
- **space-around -** ensures your elements have an even amount of space before, between and after each one.
- **initial -** uses the default value.
- **inherit** - makes an element take the positioning of its parent element.



How do you want to align your elements? Here are some options that are possible with flexbox!

CROSS AXIS ALIGNMENT

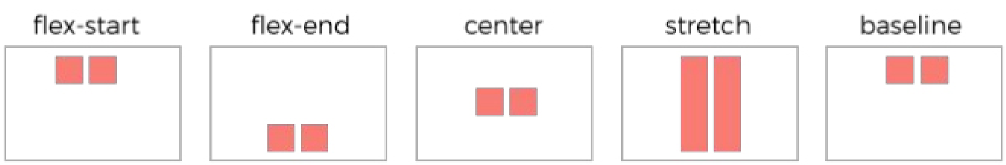
align-items

Want to align something along the cross axis? Use the **align-items** property, which acts exactly like justify-content, except that it aligns your elements along the cross axis. Pretty sweet!

Get started by adding the following code:

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

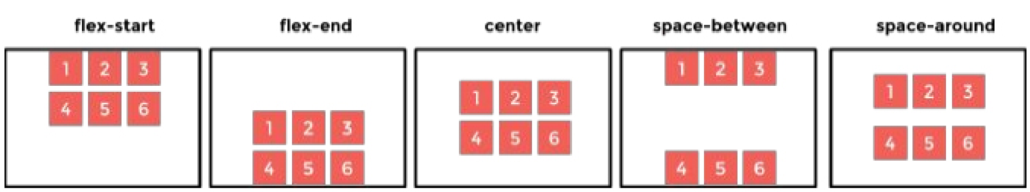
Presuming our flex-direction is 'row', the align-items property values **flex-start**, **flex-end**, **center**, **initial** and **inherit** function in the same way as they do for justify-content, but instead of operating horizontally, they're oriented vertically, according to the cross-axis. The values unique to align-items are stretch and baseline. **Baseline** positions items at the container's baseline, which is often the top of the container. Meanwhile, **stretch** elements are stretched vertically to fill the container.



Flexbox also provides many options for vertical alignment.

align-content

The **align-content** property has the same values as justify-content, but is used when there is wrapped content. For example - if our flex-direction is 'row' and we have two rows of content, 'align-content' will determine how these two rows will display in relation to each other inside their container.



Here are some examples of using align-content with wrapped content.

So, if you have some wrapped content that you'd like to align vertically to the center, you'd write:

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

If you do not have wrapped content (i.e., more than one row), align-content will have no effect.

flex-wrap

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

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

The possible values include:

- The **wrap** value tells the items to wrap.
- The **nowrap** value tells the items not to wrap.
- The **wrap-reverse** value tells the items to wrap in the reverse order.
- The **inherit** and **initial** values function in the same way as with other elements, either inheriting the parent's value or using the initial default value.

Some examples of flexbox-wrap in action!

Now that you've learned the basics, move on to the next step to learn more flexbox properties!

MARK AS COMPLETED

Have you completed this step? Make sure to mark this step as completed.