## **Alignment**

So far everything we’ve touched with flexbox has used the rule flex: 1 on all flex items, which makes the items grow or shrink equally to fill all of the available space. Very often, however, this is not the desired effect. Flex is also very useful for arranging items that have a specific size.

### **Learning Outcomes**

* You’ll learn how to align items inside a flex container both vertically and horizontally.

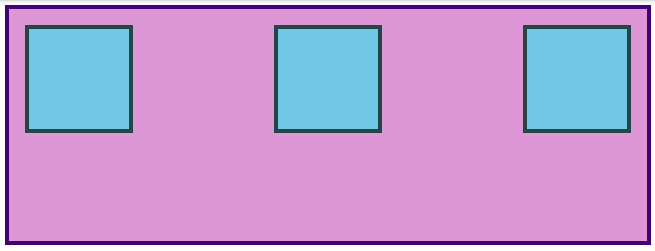
Let’s look at another example.

<https://codepen.io/TheOdinProjectExamples/pen/MWoyBzR>

You should be able to predict what happens if you put flex: 1 on the .item by now. Give it a shot before we move on!

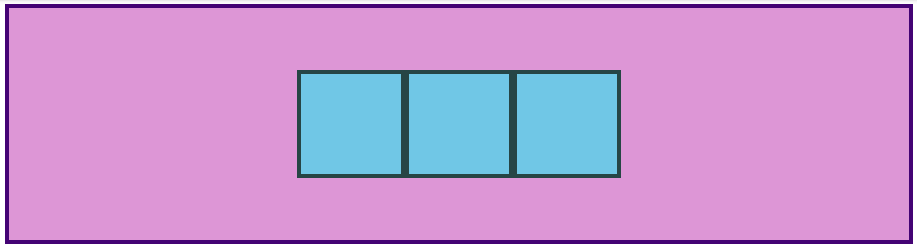
Adding flex: 1 to .item makes each of the items grow to fill the available space, but what if we wanted them to stay the same width, but distribute themselves differently inside the container? We can do this!

Remove flex: 1 from .item and add justify-content: space-between to .container. Doing so should give you something like this:



justify-content aligns items across the main axis. There are a few values that you can use here. You’ll learn the rest of them in the reading assignments, but for now try changing it to center, which should center the boxes along the main axis.

To change the placement of items along the cross axis use align-items. Try getting the boxes to the center of the container by adding align-items: center to .container. The desired result looks like this:



Because justify-content and align-items are based on the main and cross axis of your container, their behavior changes when you change the flex-direction of a flex-container. The most common behavior is the default, i.e. justify-content aligns items horizontally (because the main axis defaults to horizontal), and align-items aligns them vertically. One of the biggest sticking points that beginners have with flexbox is confusion when this behavior changes.

#### **GAP**

One more very useful feature of flex is the gap property. Setting gap on a flex container simply adds a specified space between flex items, very similar to adding a margin to the items themselves. gap is a *very* new property so it doesn’t show up in very many resources yet, but it works reliably in all modern browsers, so it is safe to use and is very handy! Adding gap: 8px to the centered example above produces the result below.

<https://codepen.io/TheOdinProjectExamples/pen/qBjZyea>

There’s more for you to learn in the reading below, but at this point surely you can see how immensely useful flexbox is. Using just the properties we’ve already covered you could already put together some impressive layouts!

Take your time going through the reading. There will be some review of the items we’ve already covered here, but it goes into more depth and touches on a few things that haven’t been mentioned yet. Don’t stress too much about trying to memorize every little detail yet; just code along with the examples and do your best to internalize everything that is *possible* with flexbox. You’ll have to reach for these resources again once you get to the practice exercises, but that’s perfectly acceptable. The more you use this stuff the better it will stick in your mind… and you will be using it *constantly*. Have fun!

### **Assignment**

1. The [Basic concepts of flexbox](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox) article on MDN is a good starting point. There are helpful examples and interactive sections.
2. [Aligning Items in a Flex Container](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Aligning_Items_in_a_Flex_Container) goes into more depth on the topic of axes and align-items vs justify-content.
3. [Typical use cases of Flexbox](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Typical_Use_Cases_of_Flexbox) is one more MDN article that covers some more practical tips. Don’t skip the interactive sections! Playing around with this stuff is how you learn it!
4. The [CSS Tricks “Guide to Flexbox”](https://css-tricks.com/snippets/css/a-guide-to-flexbox/) is a classic. There isn’t any new information for you here, but the images and examples are super helpful. This one is a great cheat sheet that you’ll probably return to often. (Keep it handy for the practice exercises!)

### **Practice**

When doing the following exercises, please use all the documentation and resources you need to accomplish them. You are *not* intended to have any of this stuff memorized at this point. Check the docs, use google, do what you need to do (besides checking the solutions) to get them done.

Go back to our [CSS exercises repository](https://github.com/TheOdinProject/css-exercises) (you’ve done these previously, but don’t forget that the instructions are in the README). Do the exercises in the ‘flex’ directory in the following order:

1. flex-center
2. flex-header
3. flex-header-2
4. flex-information
5. flex-modal
6. flex-layout
7. flex-layout-2

It may take you a while to get through all of them, and the difficulty ramps up as you progress. Stick with it! If you can get through all of them, then you will be in *really* good shape moving forward.

### **Additional Resources**

This section contains helpful links to other content. It isn’t required, so consider it supplemental.

* [What the Flexbox!?](https://flexbox.io/) is a great free video course from Wes Bos. If videos are your thing, this is a great resource for you. It’s a good refresher if you’re having a hard time getting this stuff to stick.
* [Flexbox Froggy](https://flexboxfroggy.com/) is a funny little game for practicing moving things around with flexbox.
* [Flexbox Zombies](https://mastery.games/flexboxzombies/) is another gamified take on flexbox. Free, but requires an account.
* This [Flexbox Tutorial](https://www.freecodecamp.org/news/css-flexbox-tutorial-with-cheatsheet/) from freecodecamp is another decent resource.

### **Knowledge Check**

This section contains questions for you to check your understanding of this lesson. If you’re having trouble answering the questions below on your own, review the material above to find the answer.

* What is the difference between justify-content and align-items?
* How do you use flexbox to completely center a div inside a flex container?
* What’s the difference between justify-content: space-between and justify-content: space-around?