CSS Flexbox

Hello and welcome to the very important lesson, where I'd like to explain the basic syntax of the flexible box layout module, otherwise known as flexbox. With CSS flexbox we can align website elements and create complex layouts in a very simple way. And understanding it is very important and makes our work much much easier. So, let's get started.

Flexbox

I have a div with an ID container. And I have six divs inside, each has a class box and then number at class because I wanted to use different colors for the backgrounds and they are numbered 1, 2, 3 and so on. Okay, so the important part about understanding flexbox is that it needs to be applied to the parent container. Whenever we want to align something using that method, we have to target the parent container. So I am going to target my container. I am going to add a border around it so we can see better - where exactly is that container. Okay, so this is my container and as you can see with the block elements, even if I specify the width of 100 pixels, the element is taking the whole space and then the next element is below and so on. And that is how everything is aligned with Divi as well. When you add a new module to a column, it goes underneath the previous one, and so on. With flexbox we can change that default alignment that default behavior of block elements in a few, very cool ways.

Display Type

So let's take a look. As I said, we are going to target the parent container and I will start with showing you all the different properties you can add to the parent container to affect all these children elements. First, the most important thing is we need to change the display type. So, display: flex. And as you can see, it aligns everything horizontally to all the children elements and by default this has a flex direction. So flex-direction row is the default but we can also use row or reverse which kind of reverse every element and it starts from the last one. We can also change this to a column if we want. So that would be the standard layout or column-reverse. Which will change the order and flip everything upside down. But let's actually go back to default, so to row, or we can remove it, and let's see what happens if I decrease this window size. It actually stretches let me removing that padding here. Okay so as you can see it will be as small as the content and what if I want to maintain the width here?

Flex Wrap

There is additional flex property I can use and that is called flex-wrap. And by default, it's set to no wrap, but if I set it to wrap, it will sort of adjust to the available space it has, and then once there isn't enough space for the next element, it will wrap on to the second line, okay? So let's remove it for now because I want to show you a few other things. Let's maybe make it a bit bigger.

Justify Content

There is another property that will affect the children element and it's called justify content. Any value we put here will align these items horizontally. So if I type - center, that would align all the content horizontally in the center, I can use flex-start, which is the default or flex-end to align it to the right side. Now, if I add more height to that container, Let's do half of the window. So 50 viewport height. Now, I can use the align items to align the items vertically. Okay, so I can use flexbox to align all the content vertically and horizontally in the center, very, very useful.

Spacing and Multiple Value Usage

Now, there are more values. I can use space-around which will basically put the same amount of space to each item which kind of makes them aligned well kind of incorrectly. There is less space here and more in between each so spaced evenly does add exactly the same amount of space to the justify content property, let's center for a moment. I can also align and stretch the items vertically. If there is no height set. This will stretch to the same size as the container. Okay. So, with align items and justify-content, we have a lot of control on where to place the children elements. Now, if I change this stretch back to center, it will be only the same size as the content inside, if I don't have the height set, and that is the default behavior with no height. Even if I remove that align items, the stretch is the default value. So, if I don't want to stretch it and I'm using flex, I can either specify the height or use center to only use the height of the actual content. Now, let's see what happens when we change the direction? And what I mean by that, let's remove this and add the height back here, so, if I change the flex-direction, to column, and use justify-content: center. You will see that now, this property aligns content vertically where with the flex direction row, that aligns it horizontally. So basically when we changing the flex direction to column the justify-content and align items properties switch places. You see now the align items, aligns things horizontally, so flex-end would align it to the right. Flex-start to the left and flex-start for the justify-content, would align it to the top. Okay, so with the different direction, these two properties do the opposite thing kind of so things like flex wrap, still apply. So if I add flex-wrap, default is no wrap and change the size of my container. The items will wrap to a new line. With the no wrap it will kind of shrink to the size the same as it works in the opposite way where the direction is set to row and now the last parent flexbox property that we can use but let's clean this up a little bit. Let's get back to row. And let's change the size. Let's make it a bit wider and wrap. Okay, and now we can use the maybe not so much that's fit to in space. So now there is the aligned content property and similar to others. We can align everything in the center that kind of applies to all elements together. Even on the separate lines you can use to center, space-around, space-between. That stretches the items leaving no space at the ends. If we do a column and different sizing, okay, that stretches it other way around. Okay, so the direction is important here. So aligned content, basically aligns all the columns and rows. Flex-start - to put it to the left side, but with the row, it would put it to the top, right? And now I would like to show you what properties can be applied to a child element. Okay? So let's remove everything we've done and just leave the flex property like this.

Ordering Elements

The first thing you can use is order because with flexbox, we can control each of these items separately. So let's say I will add an order property to each of these boxes, order one, order two, order three, four, five, and six, okay, and now, let's say I want to switch this item with this. The second with the fifth, all I need to do is change the order value and you can imagine this being your modules inside the column. So, with different order value, I can very easily change the order of elements, I do not need to change my HTML structure. I do not need to move things around in the builder. I can do that with the order property. As long as the parent container uses display flex. Okay, I want the last one to be here. Let's switch them, switch this places. Simple as that. Okay, so that's the one thing.

Flex Grow

Now, let's look at the flex grow property. You see how we defined the width of these items and they do not fill the entire space. So I can have my last box, fill it with the flex grow one value, the default is 0. I could add that flex-grow: 1 to my fifth element and they together will fill the available space in between them, or I can add it to all the boxes. Okay. That would make sure that they always fill the entire available space. And then decide that I want this to be bigger than others by flex-grow: 2. Okay? I want the first one to be bigger than others. There's not enough space here, but if we kind of make it smaller. Now we can see it better. We can say the first one should be three times as big, okay. But anyway, it's useful for making sure that your content fills the entire available space. But with the flex-wrap: wrap, it doesn't work because with flex-grow our boxes kind of don't know when they should wrap, okay?

Flex Basis

And that is why we can also use the flex-basis property. It kind of works like a width so by setting a flex-basis to let's say 50 pixels. We will make sure, okay, 50 is a bit too small, that's right - 150. Okay, so with that, with the set flex-basis and the flex-grow, the item that is wrapped onto the second line, will always fill the entire available space. Let's remove that from the first one, so they are all equal and now you can see they align nicely. So with the flex-basis we can do percentages here. We can kind of set the width of the item and also use the flex grow to make sure they fill the entire available space. So the last one, maybe let's remove that width here, and this. Another useful, maybe, value would be flex-shrink. So on my second box and flex shrink by default is 1. So if I set it to 0, this way, my second, let's actually change the flex-base - oh disable the wrap. And as you can see now with the set flex-basis, my second box will not shrink to be smaller than this flex-basis size. Okay. So that's the flex-shrink and we can also define these three values: flex-grow, flex-shrink and flex-basis as a shorthand, Okay? So flex-wrap, this way, we are defining the minimum width, kind of, for our elements with the flex-shrink. With the flex-grow we'll make sure that if they wrap, they also fill the available space. So the short hand is just flex and three values. The first one is grow. The second is shrink, and the last one is flex-basis. Okay, so we'll do this. Exactly would be also the same as this. Now, another value we can use - and let me remove that. Let's actually leave the flex-basis. Okay, let's make it a bit smaller. So with the, let's change the container size to something bigger. We have another property called align-self, okay? And now I can align the second box using align-self:, flex-end. This will kind of move it and align it in a different way that the rest of the elements are aligned. So if all my items would be centered I could move just one to be at the top for example, flex-start. Similarly, if we change the flex-direction, to column. The flex-start will move it to the left instead of to the top, right? So that would work the same way.

So these are all the properties I wanted to show you when it comes to flexbox. And if you'd like a fun way to practice it, I highly recommend you check out the flexbox froggy.com, the kind of game that takes you through all the different flexbox properties with these fun assignments inside this pond. So you should definitely check it out if you haven't seen it before.