CSS Grid Basics

Hello and welcome to the lesson about the CSS Grid Layout System. This is a very complex and powerful CSS system, which allows you to align website elements and create any type of grids with any number of rows and columns. I have to say that you probably have the need to use grids in only a few situations. And you can do most of the alignment using flexbox and because we are working with Divi the grid system is probably not as useful as it would be if we hand-coded a new HTML website from scratch based on the CSS grid. But there are situations where using a grid is the perfect solution and you'll see that in a moment.

Basic Syntax

But first, let's get familiar with the basic syntax. Similarly, to the flexbox example, I have a container div with a five divs inside. And as with the flexbox to use the CSS grid system, we need to target the parent container. So what happens if I say display: grid? Nothing. Well, the change that of the display property to grid, doesn't do anything, but let's look at another property: grid-template-columns. And to specify our template columns and to specify our template. Let's choose some value. Let's say, 200 pixels. And then again, 200 pixels and that gives me 2 columns 200 pixels wide. What if I want the third column? I can do 200 pixels again. What if I want the middle column to be wider? I can change the value. Okay, so it's that easy. And what if I want five columns instead of you know repeating myself, I can use the repeat and then specify the number of times I want to repeat something. So let's say 5 and then 150 pixels. And that gives me 5 columns of 150 pixels width.

FR Unit

There is as one special unit because we can use EMs or percentages or any unit we like but there is one special unit for grid the "FR" unit. It's a fraction of the available space. So if we want to fill the whole space, we can just simply specify five times one FR, okay? So it is super easy to create any number of columns we like and we can do similar thing for our rows. So let's say I only have two columns. Let's look at the grid-template-rows property and it works very similar. So let's say, I want to have rows which are 10 em tall, right? We can specify this value exactly the same way as for the columns. But it's probably not as useful, but there is even a better property, but to be able to use it, we have to define our grid areas and we can name them. So grid-area, each of my divs with a class box, I will name it accordingly. So my grid area name would be box one. And so on: box two, box three, box four and box five. And it does some crazy things right now. But as soon as we define grid-template-areas, you'll see where the whole magic happens. So for that property, we have to use double quotes, Okay? And let's say I want to have three columns and now I can use these names and just say box one, box two, box three. And what if I want a second row? Let's do box four, and then dot and dot, okay? As for the third row, I can go box five, box five, box five. Okay. So as you can see using dot create this empty space and I can stretch my elements anywhere I like. So here, for example, I instead of the second space, let's say box three. And now, maybe I want my second box to be here next to the fourth one. And I want to stretch my first box right here or maybe I want the fifth one to be only here in the middle, and the fourth one to go here. So, hopefully you can see visually how easy it is to define different areas inside your containers, just by using a different grid area names. But if this approach doesn't speak to you, we have a different way to define our alignment of each of the children elements. So let's remove that and let's remove the grid area property. Now it comes back to default but what if I want to specify that I want the second box to stretch from this column over to this column. Okay? So what I have to set is the grid column property - grid-column and I have to specify two values and in this case, it would be. 2/4. And how that works is it counts the lines of the grid were here we have first line. That's the second line, third and fourth, okay? That's why we go from the second to four. If I want the third one to stretch over from here to here. Okay, so here for the grid column would be one from the first line and then one, two, three. So 1/3 and now the fifth one, let's stretch it all the way. So grid-column 1 4. One, two, three four and instead of counting lines, there is a different method we can use span.

Span

So span defines, how many columns to span across. So instead we start at the second and then we can span by two columns, that would work the same way, okay? And for the third box, the pink one, we start at the first one and then we can span across three columns, maybe that kind of is better or two columns, okay? Maybe this is easier for you to kind of remember and it works the same way, okay? And similarly to grid column, we also have a grid row. So if I want the first box to span across these two rows. The first line is here for one, two, three. So I want this to go from one to third line. Perfect. So I hope you see how powerful this is. So, the third box would use a grid-row of 2 / 2, 3, 4. So 2 to 4 or two, and then span across two rows, that's the same. So hopefully that's helpful. I kind of like the naming approach. So if we can go back to it.

Gap

Okay, so we can also specify gap in between these columns and we can either specify grid-column-gap. Let's say 10 pixels. Okay. And we can do the same for the grid-row-gap, okay? Or just simply grid gap. Very nice and it just, you know, works perfectly.

Uses of Grid

So, it's great to create that kind of layout, when you kind of want to display a photo and then if we change the grid-template-row. And let's say we want this to be at least 300 pixels tall, okay? Or maybe 200 pixels that gives us this nice grid, I could place an image here and here maybe some call to action and a different image and a button and some text, you know, mix all these modules together and it's super easy to achieve with the grid if we wanted to position modules in this way in Divi, it would be very difficult. Even when we try, you know, positioning absolutely or moving it to the left or right with the relative position and the top right left bottom values or even with flexbox, that would also be difficult. So grid is especially useful in these types of layouts the complex grid. If you just want to place 4 items next to each other, go with the flexbox, but if you want to create something like that, then grid is perfect for that.

Align Property

And another thing we can do is to kind of align items within the grid so you can see the grid we've created here and I can justify items similarly to flexbox - center. Okay, this moves everything to the center, horizontally, if I want to move it to the center vertically, as well, align-items: center. Okay, it's still within the grid, it's just hard to see. But if I open that page in Chrome, for example, by selecting my container, I can click the grid here and it shows me my grid. So that was my area. And now my element is in the center and this is in the center of this grid and and all that. And similarly to flexbox I can align each item separately. So, let's say in the first one, I want to align - align-self. Let's try stretch, that stretches it vertically. I have aligned self and also justify itself.

Stretch Property

Stretch. Okay, so I can keep all this in the center and have the stretch or other way around. So, let's say I want everything to be stretched and that is the default actually, so I can remove that. And I want this box one to be aligned at the left. So that would be start and at the top also start, we don't have to do grid start like with flex start just the start or end. Okay, so we can align elements within the grid that we created, for the container. It doesn't have to be stretched all the way.

I hope that you can clearly see the power of this grid layout and I hope that explanation was clear.