VIDEO TRANSCRIPT

CSS Grid in Divi

Hello and welcome to the last lesson inside the advanced CSS techniques module. And in this video, I would like to show you step-by-step how to implement a CSS grid inside Divi. So let's have a look at what we are going to create. This is a sample layout, a very simple use of grid, but as you can see, the type of layout would be difficult to achieve using the Divi Builder and the standard column structure. So, this is the perfect example where using grid is the best approach. So, let's go ahead and create that exact layout from scratch. Okay?

Basic Layout

So here I have a new page and now, I need to insert my modules. So when trying to create a layout with grid, we can use any modules we want. But there are some limitations and we have to remember that we are targeting columns as the parent container. So the column would be displayed as a grid and then inside each square inside each box in our grid, we can display it as a separate module. So for images it's best to use a divider module and insert your image as a background instead of using the image module which can be difficult to size because it needs to be responsive. And using a background image can use a background size cover, which fills the entire available space. So, since we are going to define the structure of the page with CSS, I do not need to choose different column structures here. I will use the full-width column and I will use three divider modules to insert three images, and three call-to-action modules to display text and a button inside each of the boxes

with text. So first - divider, and I will set a background image. And I will make sure not to show that single pixel line here at the top. So "Show Divider" no, and what we need to do is in the design settings we want to make sure that the module size will be filling the entire available space. So in the sizing, we want to set the minimum height to 100%. And that's for desktop, and on tablet our grid would go back to normal, so everything will be displayed full width. We can play with that in the next module. But for now, let's focus on the desktop version. So, for tablet - sorry for tablet here, I want to make sure that the image is visible. So I also have to set the minimum height to some pixel value and it could be smaller on a phone. And that would be my default divider and I will be using the grid template areas to define my grid. And that's why I want to name each module separately with the grid area property.

Using CSS Grid

So in the advanced custom CSS main element, I'm defining my grid area and I want to name it. The name can be anything you want. For me is just an image. So I will name it image 1. I will also do that in the label here. Okay, that's a good start. So I can create a call-to-action now. We'll leave everything at default, maybe use a bit less text here. I will just add an empty URL so we can see the button as well and I want to change the background color. Something that matches my photography here. And because our grid size will be defined by the size of the content inside the grid boxes, right? So, we want to make sure that our call to action module is big enough, so it shows enough of the photo. So, in the spacing section, I want to add some padding. Okay. So let's add a lot of top and bottom padding and also on the right and left maybe or maybe a bit less. So 80 pixels top and bottom and 40 on the side and then we can decrease that for the tablet and phones. Okay, that would be great.

VIDEO TRANSCRIPT

And again, in the advanced section custom CSS, I need to define the name of my grid area. So grid area, CTA one - call to action one. And now, I will also name it so I can see it better. And now inside the wireframe view, I can basically put the modules in any order I like, because then I can use the CSS to position it within the grid, but I want to maintain the order in which I want the modules to appear on mobile. Because on mobile, I will disable the grid. So on mobile, I want to have an image then call to action, then image, then call-to-action for each. Okay, so we want, we have the second and the third and now let's change the image. It's in the background. So that's the second one and we also have to change the grid area name. The same thing for the third image background. And grid area name image three and call to action. We want to change the background and also grid area inside the CSS. Call to action two. And the last one will also use a different background. And we need to change the grid area named CTA 3, okay? So everything is set up correctly except for some kind of identification for the column. So in the row settings in the column settings, that would be our parent container. We have to define either a CSS class or an ID, DSA grid. Okay, I will copy it because now I have my structure, ready. That's how it looks on, desktop, by default. But let's start adding our CSS. So, first of all, we want to specify the class, DSA grid, to be displayed grid. Then the column structure, so grid template columns will use three columns. One FR, one FR, one FR or repeat 3 comma, 1 FR. So that creates three columns and now for the grid template areas, We want to define each of the areas and we can use the names we assigned. So, in the first row, I would see the beginning of my image one. So image 1, and then image 2. Next call to action one. In the second row, we have the rest of the first image, so image one, then call to action two, and the beginning of the third image, image three, and in the last row, we have call to action three

which spans across two columns, And then we have image 3 the rest of it.

Margin Spacing

Okay, so I will save that because the preview is a bit, is going a bit crazy here, but trust me that it will be fine. But one other thing I forgot to fix is the bottom margin. So let me just publish that page. The minimum height is kind of buggy here. It sets the rows of the modules to be too long, but if I refresh the page, it gets back to normal. But the problem here is with the margin. So if I go here in the design spacing margin and let's remove the margin from each side and let's extend that to all modules. So we don't want any margin inside any module within this column. Okay, lovely. And that's it. Look, we, basically did that. We can also define the grid gap. So, in the custom CSS grid gap, 10 pixels, for example, or maybe only two that maybe works better with the rest of your design style, right?

Mobile Quick Fix

So you can use any values you like here, but we do want to make sure that this isn't displayed the same for mobile devices So I would just add a simple media query here. we will cover that in the next module, but just to fix this layout for now, I will target the maximum width of 980 pixels. So on any device that this size my grid will be displayed as the default. So I will just change the display from grid to block. And that will basically bring it back as how it would look if I haven't added any of this custom CSS here. So let's save that, let's exit the builder to see how that looks and maybe let's check if the mobile version looks fine as well. I think I've made some mistake inside the media query, let's check. I think my column is not, so let's see what I did wrong here for the media query.

VIDEO TRANSCRIPT

Oh, okay, so that's the mistake. Okay. Now if I save that, let's see -Yes. Now on the mobile it just goes to the default Divi layout.

Grid in Chrome Inspector

What's great inside the Chrome inspector here is that when an element is set to grid, you can see that great link here, that's the little button and that basically shows you grid areas name and how that grid is displayed in the layout tab. You can see what to show. So it kind of helps you to define your grid. If you're running into any issues, you can see that helpful information within the inspector here. So, that is basically it. You have to remember that you always have to target the parent container. So, for example, if you want to display some complex layout within a single box inside the grid, you can use rows as your children and then section would be your parent container. But it would be an advanced use case.

Just using simple modules and images as a background works really, really well and allows you to create beautiful layouts.