VIDEO TRANSCRIPT

Custom Columns Structure and Order

Hello and welcome to the lesson about Custom Column Structure and Order for mobile. As we learned, the breakpoint Divi uses to switch from desktop to tablet view is at 980 pixels and that is problematic because by default if you use a two-column layout or even a three-column layout, it will change to single column on tablets. Many times we would fit more than a single column on tablet view. And we need custom CSS to achieve that. In this video, I will show you how to do it, how to define different numbers of columns for rows on tablets and phones, and how to change the order of these columns as well.

Default Example

So, I have a sample layout here, with the representation of the column structure in Divi. So, if I switch to tablet view, the two-column row uses a single-column layout, three-column row as well. The four-column row starts with two columns for tablets. And if we use a six-column row for desktop, that changes to three columns on tablets, and custom layout structure, like this one, when we have some small element on the side and then content in the main column, if we go to tablet, both of these columns are now equally sized - they're both full width. And for phones, any column structure will change to a single-column layout. It may be especially problematic when using three-column layouts for something like images. So let me add a section here. So that section uses three images here, the smaller images that have less visual hierarchy importance, right. And then we have our main image which we want the

main focus on. And when I switch to tablet view, all these images are equally sized. Making them visually, equally important when it comes to visual hierarchy and that is not what we want, right? If you want to keep this image as the main focus, even on smaller screen sizes, we need to be able to edit the size of that column structure for mobile, okay?

Defining Media Queries

So how do we do it? We need to define our media queries, okay? And then use flexbox to change the column structure. So, here's the custom CSS box inside the page settings, Advanced tab. And now I can define my media query. So @media and in the round brackets maximum width 980 pixels, that will target both phones and tablets everything that's below 980 pixels or equals 980 pixels. And I've added a CSS class to each of the rows. So I know that this one with two columns has a class of row 1, display: flex. Simple as that. Okay? Now it uses two columns and the same would apply to rows with three-column layout so I can target row 1 and also, row 2. That's the CSS class, I've added to the row settings here. So display: flex is all we need to make it use the same column structure, but we do need to add some padding here or margin. So any column inside a row 1 or row 2 should have a right padding added. So as a parent that would be row 1 and then we are targeting a column which uses the standard Divi class, et_pb_column. And again, for row 2 - et_pb_column. And we want to add a right margin. That's right with 5% and usually, the problem would be on the right side here that we need to remove the margin from the last column but Divi actually already does that in the custom CSS so we don't need to worry about that, and it's simple as that. So you can change that value to anything that matches your content. And as you can see that, three image row also use the row 2 CSS class and that fixes that alignment for tablets and it will also use

VIDEO TRANSCRIPT

the same size for phones. So using three columns on the phone for text may not be the best option, but it can work well with three images. So depending on the content you might want to keep or change back to single column layout for phones. So, if we only want to target tablets and not phones, we have to also define the minimum width that this CSS applies to. Right now the minimum width is 0. So min-width: 768 pixels, so that would be the standard Divi breakpoint and we have to include the end operator and now it only applies to a tablet. For on the phone we have the standard single-column structure and on the tablet, we can see three or two columns, so the original column structure. Now, let's look at the layout that has four columns, so row 3, if I use the same display: flex - on row 3, that doesn't help because it is already using flex. So, let's save that, and let's preview the page in the browser. This row already uses flexbox, its display: flex and also flex-wrap: wrap. And that means that if the column has a specific width set, they will wrap to a second line. So If we check the column element it uses the specified width of 47.25 pixel and right margin. That percentage would depend on the gutter size that's set on the column, but we can try overwriting it. So, what if I would specify the flex-basis to 25%? Okay, that works. So, I could make sure that my columns are using the 100 width and then change the sizing with flex-basis property. So let's try that - here. So for the row three column, width 100%. And I can already see that, that won't work because it isn't specific enough. If I add important - that will work, but we can also try changing the selector to be more specific. Let's leave the important for now. And let's try adding flex-basis and that can be anything I want, if I want to change the sizing of these columns to three-column layout, for example, I could use 33%. It also uses margin. So, we have to make sure that the margin doesn't exceed 100%. So, let's say we will use 30% for width, and then margin-right could be 5%, 5%

VIDEO TRANSCRIPT

5% here and here, zero on the last one that equals 100 percent. So margin-right 5%, let's try with important. Now, we need to remove the margin from the third column and we can use the nth-child selector. 3n that would select every third element. Margin-right, 0, important. Okay, so basically, you can play around with that approach and apply it to any number of columns. So for example, here in the last one, we have, that's the row 5. Let's try displaying it, flex row five, that makes these columns equal and as you remember on a desktop that uses the small column on the side and the main column is bigger. If we want to keep that on a tablet, I can specify the row 5 et_pb column and then maybe use the first-child or nth-child one, let's try with that nth-child one selector that selects the first one. Let's make it a bit bigger so you can see better. So, first column and let's define flex-basis maybe 30%. Okay, and then the second column nth-child two, let's try with 65, okay? So there's five percent, but they are still together, so this row five can also use the justify-content property: space between. Okay. And that moves it to the right side, we have the five percent space in between. So, by applying the flexbox properties inside the media query, we can control the sizing of columns. Sometimes we need to work with different margins, so just be mindful that some column structure will use some different margin values, and sizing values for the columns, for smaller screen sizes, but with the default, three-column layout, or two columns, it's very easy to change that column structure. And then we can also change the order of columns, just by simply applying the order property. So, here, in this, four-column layout, let's say, I want the third one to be displayed first, so to target it - I can add a custom CSS class to that column, but I can also use the nth-child selector. So inside row 3, et_pb_column nth-child 3, so only the third one. Order: -1 because the default is zero, okay? So using one will actually put it to the end. If we select different orders

for different columns, we can use one, two, three, four. But if you only want to put one first, it has to have a negative one order. Okay, now it's the first one. Then you have to make sure that the margin also works here because We targeted this to have zero margin before, so maybe let's use the last one as the first one, okay? So that brings the fourth column - here at the beginning. I think it also doesn't have any margin on the side, so we would need to add it again. Like I said, playing with margin can be problematic on this different column structure, but it's a matter of defining your values for each column, the way you want it to display. And it's not like we can only move one column to the beginning. We can basically define the order of columns for each. So let's say, here on this column structure, that's a row five, I can target each of the columns with the nth-child selector. And that could be order: 2, and now first is second, and so on. So, let's make this or the third one could be the first okay, I hope you can - oh sorry, row 4. Okay, like I said, the default is zero. So if we specify each one, we have six: one, two, three, four, five, and six. Okay, and now we can play with the order. So the last one - let's make it first. The fifth one, maybe third this one second three, four, five and so we don't do not repeat ourselves, let's make this one the sixth one. Okay, so here there's also a margin issue that we would need to fix. If we remove the margin from each of the columns, you will see that the order fixes the margin for us and then we can define the correct margin for the correct column. Okay? So the number one, column number one is now fifth. One, two, three, four five - there's our column one.

Okay, so you can see how easy it is to change the order of columns. And then you can write another media query, which only targets phones and also, use different column structure, different order anything you like, as long as you understand, how flexbox work and you are mindful of

different margin and sizing that can be applied to columns by Divi which you can easily inspect here in the browser inspector. It should be easier for you to Define your own custom column structure and order for mobiles. I hope it's helpful.