VIDEO TRANSCRIPT

Responsive Units in CSS

Hello, and welcome to the lesson about CSS Units. I know, it can be quite confusing with all the different available units to decide which one to use. Should we use ems, rems, percentages viewport width, or just stick with pixels. And I'm afraid that this is one of those 'that depends' situations. But in this video, I would like to explain how exactly the most popular units work and share my thoughts on which units to use when working with Divi.

Unit Overview

First, let me give you a little overview of the available options, pixels, one of the absolute length units in CSS. For absolute length units, we could also use centimeters, millimeters, points, inches, but if you're not writing CSS for printed media, and you want to use a fixed size, you would use pixels. That is the easiest unit to use, we all know what a pixel is. But in addition to absolute units, we also have a number of different relative units whose size is not fixed. And it depends on the size of another length property. First, the easiest to understand - percentages. Percentages are the most useful when setting the width of elements and the value is always relative to the parent container. Next, we have ems, relative to the font size of the element. Rems, relative to the font size of the document root which basically means the HTML tag, and let's have a closer look at these two units. Both of these units are relative to the font size and the first thing we need to understand is which font size exactly they are relative to. So this is my HTML structure. I have two

divs with a class "box". First one has an additional class of "box-em", and the second one has a class "box-rem". Inside each div, we have h2 and 3 paragraphs. And I have some basic styling here and now if we look at the font sizing so if I set the font size on the box, the parent container, and declare font-size: 20 pixels that will apply to everything inside, right?

Ems

But if I want to target the first paragraph in the em box, that would be box-em p: first-of-type, right? And now let's say font size: 1 em And that doesn't change anything because 1 em is a sort of multiplier of the parent font-size value, so it's also 20 pixels. If that would be two ems, that would be 40 pixels, okay? So if I target the second paragraph, using the nth-of-type, 2, let's say font-size: 1.5 em, that would mean this will use 30 pixels, it's 1.5 times 20. The third paragraph could use 0.5 that would be 10 pixels, okay? So hopefully you see the logic here, the font size set in ems will look at the closest parent font-size and calculate the value-based on that.

Rems

With rems, it's a bit different if I copy that and use it on my rem box and change the unit to rem - you will see that these are smaller. And the reason for that is the rem unit doesn't calculate its value based on the parent size, but document root text size. So what is the document root? It's basically the first element that the document sees in our case that's the HTML tag. So if I set the font size on my HTML tag, to 20 pixels. This would be exactly the same em looks at its parent and rem looks at the HTML tag. We have to be aware that each browser has its default font size of the document root. In Chrome the default is 16 pixels. So if I set the font size on the HTML document to 16 pixels which is basically the

default you can that it will work the same here if it's set as the parent container for the em. So I could just as well remove that and that would also be 16 pixels. So if your root element, your document root doesn't have a font size set. The rem unit will be calculated based on the browser default and browser default can be edited to user preferences. You can go into your Chrome settings and change the default font size to larger font for example. And in this case - setting your font-sizes in the rems will be relative to user-defined font size. So, for rems, that would be always one size that they are relative to whether it's the default 16 pixels or some other value. It's always the same size that these rems are looking at, whereas ems are relative to the parent and it causes these compounding issues.

Font size example

A little bit different HTML structure, I have a div with a paragraph text and then nested lists inside. And now in the style sheet, if I set the paragraph text inside my box to be 1.2 em font-size. Or let's make it 1.5 so that we can see it better. And let's say I want to use that same font-size inside my list. So I could specify box ul and what it does, each list looks at its parent and that causes that compounding issue and the text gets bigger. So if we have a lot of nested divs and we're using ems, it can get confusing because It always will look at the parents. So that would be 24 and that would be 24 x 1.5 so that would make it 36 and so on. So now if that would be in a rems, we wouldn't have that problem because rem doesn't look at its parent. It looks at the document root right. Okay, but that is just for the font-size. we can use ems and the rems for different properties, like padding, margin, width, and all that.

Difference between ems and rems in example

VIDEO TRANSCRIPT

So let's have a look at a different example. So here I have three links which are styled as little buttons, each has a "btn" class and this has an additional class with small and this is big one. Okay, so in my stylesheet, I defined some styling for that, each of the buttons and I chose different font sizes, and I set them in pixels, but it's just to demonstrate. It doesn't matter what unit would be used here, it's the calculated font size we want to look at. So if we set padding using em units, so padding, 1em and two ems on the side or maybe let's make it a bit smaller. With ems, this size will be relative to the font-size of the element, okay? So that space will be bigger when the text gets bigger that space gets bigger and it's smaller when the text is smaller. If I use rems here that spacing will always be the same in pixel value. So using ems for things like padding here, when it's relative to the font-size of the element, works much better because it scales together with the font-size while using rems would be the same as using the same pixel value for each of the font-size. So you can see that this big button doesn't look as good as if it would have more spacing that's relative to its font-size. So hopefully that explains the difference between ems and rems. And next relative units are viewport width and viewport height. They are relative to one percent of the width of the viewport. And viewport is basically the browser window. Now, when should we use each of these units?

When declaring font sizes

When declaring font sizes, the recommended approach would be to go with rems, but I don't think that it is as useful when we work with Divi, the purpose of using rems is so it adapts to user preferences. If someone has set their default font size to 20 pixels using rems will display everything bigger for that user. The visual hierarchy would be kept, but the problem with this approach in Divi is that the Divi uses pixel based

media queries. If we are using rems for the text size, we should also use rems or ems for media queries, to make sure our layout adapts to a new breakpoint in correct moments. With fixed, absolute pixel based media queries, using rems may cause the layout to break if the user has a bigger default font set. Because the text may not fit within the pixel-based column structure. Now, if your layout is simple and you are not worried about the text size breaking the layout, going with rems is okay, but if your page layout and column structure is more complex and the font size does matter. I would stick with pixels.

When declaring padding and margin

When declaring padding or margin, you can go with the em unit if that size has to be relative to the font size. If you use percentages, it will be relative to the parent container width. Viewport width, or height can also be useful. But be careful with using viewport height, because you need to be mindful of the different aspect ratio that the phone uses when the height is higher than the width. And I also think it's okay to use pixel values. But be careful with using large pixel values for margin and make sure that absolute size you said works well across all breakpoints. So you have to make sure that if you set a margin of 100 pixels on something that when the screen gets smaller, it still fits its place.

When declaring width

Now, when declaring width or maximum width or minimum width, I would usually stick to percentages when defining width of the content. You might want to use the viewport width, to make sure an element is full width regardless of the size of its parent container because 100% is only as wide as the parent container, while 100 vw is exactly as wide as

your screen. If you want to use some absolute fixed width, try defining the maximum width property instead of the width, and use the maximum width in pixels and keep the width, in relative units.

When declaring height

When declaring height, in general, avoid setting a fixed height on anything, even using relative units, because it can become problematic very quickly. So, if you really need a fixed height and you are sure you need to set height on an element, use the min height - minimum height instead. This way, if the content gets longer for any reason you won't see that overflowing issues.

Absolute vs Relative Units

I hope that overview is helpful. Please remember that these are only my suggestions and you don't need to use the exact units I mentioned here. And since we are working with Divi, there are some styles that we have to work with and not against. That's why going to a fully relative unit, may not be possible or it would be very difficult. So whatever unit you decide to use, just simply make sure that your layout looks good, across all devices. I think it's just as simple as that.