Module 2: Website Layouts

Video Transcripts

Video 2.1: Control Layout Using CSS Grid

Creating CSS layouts has been a challenge for a long time. The functionality and the primitives of CSS have been a poor fit for what we wanted to do within the page. And so, generation after generation of CSS extensions has addressed this need by offering new functionality. The latest CSS grid is a true grid, as you can see here in this illustration. And you can target areas of it. You can express the syntax more simply, and it has cleaned up a lot and has provided a model that more readily fits the needs of layout. So, let's go ahead and write some code to illustrate the use of grids within CSS and HTML. I'm going to go ahead and start off here by creating a new file. I'm going to save it. I'm going to call it grids.html. Inside of it, I'm going to paste some boilerplate, simply an HTML page with a header and a title. And let's go ahead and enter here 'Grid' for the '< title>'. Save. And I'm going to go ahead and drag and drop that file onto the browser, as you can see there. And we have nothing other than the title of the page, which is what we just set. Next, I'm going to bring in some tags that I wrote beforehand. And as you can see here, I simply have a '< main >' tag. And then 12 '< div >'s numbered or written there as 1 through 12. So, I'm going to go ahead and save this, reload the page. And as you can see there, we have the numbers 1 through 12, vertical, vertically on the page.

So, let's get started with our styles. I'm going to enter a '< style >' tag. And inside of it, I'm going to style my 'div'. I'm going to do that before I start the work on the grid because I want to set some styling so that each of the elements in the grid are visible. So, I'm going to go ahead and set the 'background:' color here. I'm going to make it 'cadetblue;'. Then going to set the 'padding:' on each of the elements to be '30px;'. That is, so we can have some nice big elements. And then I'm going to 'text-align:' my content to be in the middle or 'center;'. Now, let's go ahead and load this. And as you can see there, everything is loaded onto the page just as it was before, except that now we have padding, so we have bigger spacing between each of the numbers.

Now, it would be nice to have different colors as we move through each of those elements, and we can do that by entering one more style and entering 'nth-of-type'. In here we will enter '(odd)'. And then I will change the color here. I will make it ' burlywould; '. That is a brownish type color. And you can see there that now we have alternating colors, and it's easier to call out the separate '< div >'s. Next, let's add a 'class' to main, and we're going to call it ' "content" '. And inside of styles, we will create the style for '.content', which will be the grid that we're going to define. So, we will enter 'display:' and the property will be 'grid;'. And next, we're going to create our columns.

We can do that using a property of the grid, that is 'grid-template-columns:'. And here, we get to define how many columns we want. I will enter three columns, and I will use the fractional notation and I will make them all the same size. So, I'm going to simply enter one '1fr 1fr 1fr;', which is the fractional notation, three times. And I'm going to go ahead and reload the page. And as you can see there now we have columns, and the content is wrapping through the page. We'd specify three columns, and so, the layout takes the first element, second element, the third element. And then the next three are laid in the document in the next row.

So, as you can see there with minimal syntax, we have been able to lay out a grid that looks pretty good, and that really only took us a couple of lines of code that were specific to the grid. Now, it would be nice to have some spacing between each of these '< div >'s. So, I'm going to go ahead and enter a property that is 'gap:', and I will give it '5px;' of a gap between the elements. And if I reload the page, you can see there that now we have some very nice spacing between them. Now, the last thing that I want to show you is how to control the rows. And for that we will use 'grid-template-rows:', and we will do something similar to what we did on top. We will use fractional notation, in this case, I will define four rows because that's what we have here.

We have, if we're doing three columns, we can go four rows. And I'm going to enter the same as before, equal size for all of my rows. So, if I go ahead and reload, you will see no change. Now, if I change the size of my first row, you'll see there that now the first row is twice as big as the next one after it. And if I do the first and the last, you can see there that we have different sizes for these. Now, we can do the exact same thing for the columns as well. And so, if we wanted the center column to be much bigger, you can see there that we have done that. Once again, pretty transparent notation, minimal syntax to be able to achieve a great deal of control in the layout.

Video 2.2: Understanding Grid Lines

Understanding grid lines will allow us to have finer grain control of our layout. So, let's write some code and see how we do that. As you can see here, I have some starter code, and let's review what that is. I have a '< main >' tag, and inside of it, I have six '< div >'s. I also have a 'class' called ' "content" ' that we will write shortly. And when it comes to the '< div >'s, the first '< div >' that you see here, the style sets a 'background:' color of blue or 'BlueCadet'. And then the following sets the 'BurlyWood;' color for all of those odd number columns. And you can see the output there on the right-hand side on the browser. Now, let's go ahead and write some styles for our grid.

And we're going to start off by writing that '.content ' class. And the very first thing that we're going to do is set our 'display:' to 'grid;'. And then, we're going to go ahead and enter our 'grid-template', 'grid-template-columns:'. And we're going to set the repeat notation. This sets a value that is repeated across all of the columns and then it takes a parameter for the column number. So, let's go ahead and enter 'repeat();'. And I'm going to have six columns and one fractional unit for all of them. So, I'm going to go ahead and save, reload the document. And as you can see there, we have columns going across the screen, 1 through 6.

Now, let's go ahead and write our rows. This is going to be 'grid-template-rows:'. We're going to use the same notation. And in this case, we're going to have '(4, 150px);' each. So, let's go ahead and reload. And as you can see there, the first row has gotten thicker, and there are three more rows that we cannot see at this moment because there's no additional content. We only have six '< div >'s. Now, let's go ahead and enter a 'gap:' so that we have some nicer spacing between the elements, and this will be '5px;'. Let's go ahead and reload. And we have now our gap for all of our divs. And before we continue, let's review grid lines. As you can see in the diagram, grid lines start at 1 not at 0. So if you have 6 columns, you're going to have 7 grid lines.

The same thing with rows. They start, the row lines start at 1. And so, if you have 6 rows, you're going to have 7-row lines. And so if you had a '< div >' that stretch throughout 2 columns, you know, say this column, and this column. You would then go from 1, the line number of column line number 1, all the way to 3. So, with that in mind, let's go back to our code, and let's add classes to each of the '< div >'s that we will style individually. And we will number them after the number value that they have. Next, let's write the style for 'class="one" '. Going to enter here that '.one '. Inside of it, we're going to specify where the column starts and where the column ends.

So, we're going to go ahead and enter the property 'grid-column-start:'. And we will start at line '1;', and then we will go all the way to 'grid-column-end:'. And we will end at grid column line '3;'. So, the '< div >' that has 'class="one" ' is going to start at column line 1 and end at column line 3. Now, we can write that notation or that property in a more compact form by entering 'grid-column:', specifying the first column line, then a ' / ', then the ending line, which in this case is '3;'. So, I'm going to go ahead and remove those two and reload the page and as you can see there, nothing has changed.

So, for the first '< div >', we started at line 1, went all the way to line 3. For the second '< div >', we're going to go from line 3 all the way to line 7.So, let's enter that. This is '.two{ grid-column:'. And we're going to go from line, as mentioned, we're going to go from line '3' all the way to line '/ 7;'. So, let's go ahead and reload that. And as you can see there, it filled out the rest of that row as expected. On the third '< div >'. We're going to enter the following. We're going to go from line '1 / 4;'. And now, we're also going to specify the row. For the row, we are going to start at line '2', and we're going to go all the way to '/ 4;'. So, on our diagram, we're going to start at 2 and go all the way to 4. So, if we save and reload the page, you can see there now that the third '< div >' is taking up all the way from red row, line 2, all the way to 4. For '.four{', we're going to go from 'grid-column:' line '4 / 7;'. Then 'grid-row:' from '2 / 4;'. Then we're going to go ahead and enter '.five{ '. On '.five{ ', we're going to go from 'grid-column:' line '3 / 7;'. Now, it looks like I left out an 'n' there. So, let's go ahead and save and reload the document and you can see there that we have filled out the horizontal space in the following two rows.

And for the last one, the sixth '< div >', we're going to do something different, although it's all the way at the bottom and it's right now on '< style >'. We want to move it back onto that open slot. So, we're going to do the following. We're going to enter the sixth style as 'grid-column: 1 / 3;', and our 'grid-row:' at '4;'. So, let's go ahead and save and reload. And as you can see that has moved into that slot. The column is the same as 1, 1 to 3. However, the row is at 4. We go 1, 2, 3, 4. So, as you can see with an understanding of grid lines, you can have finer grain control of your layout.

Video 2.3: Holy Grail

It used to be so hard to create a layout like the one you see before you where you had a header, a footer, some menu, some main content, and some ad space that it was called the HolyGrail. And if you're interested, you can navigate on Wikipedia. And there's a page with the history and all of the challenges that have needed to be overcome. However, with grid, we can do so in a much easier way. So, let's go ahead and create one of these Holy grail layouts. We'll start off by creating a new file and into it, I'm going to put in some boilerplate. And then I'm going to go ahead and save this file. I'll call it holygrail.html, and I'm also going to title it the same. And now inside of the body, I'm going to create a '< div >'.

And I'm going to use the 'class="grid" ', which we will write shortly. And inside of "grid", we're going to write our header, our footer. All of the components of the Holy grail. We'll use the HTML tags that are closely labeled in appropriate, in this case, we will use '< header >', then '< footer>'. These are plain vanilla HTML. We'll have an '< aside >'. Then we will have another '< aside >'. And in the middle, we will have our '< article >'. Now inside of these tags, we will simply write 'Header', then we will write 'Left'. Then we will write 'Article', 'Right', and 'Footer'. Now we will create our '< style >' Inside of it we will create the '.grid'.

And we're going to set our 'display:' to 'grid;'. Then we're going to set our template for the columns, which will be 'grid-template-columns:'. We're going to make the left-hand side column '200 px'. Then the main body is going to be '1 fr' then the right-hand side is going to be '200 px;'. Next, we're going to set our rows and we will set 'grid-template-rows:' and we will 'auto' size, then '1 fr' then 'auto;' again. Now we're going to set our 'gap:' to be 10 pixels. Well, let's make it a little bit smaller. Let's make it '5px;'. And we're going to set our 'height:' to fill up most of the page. We will set it at 90% of the vertical height. Now, to complete our layout, our Holy grail, we're going to set the 'header' and the 'footer' to have the following 'grid-column:' we're going to run from line '1'.

So, line column '1', all the way to '/ 4;' which means that the header and the footer column is going to span all three of the columns that we have in our template. So, surprisingly, this is all we need. Now, I'm going to go ahead and move to the browser, and I'll drag and drop that file. And as you can see there, we have in fact the Header, Left, Article, Right, and the Footer. Now that's a little bit hard to see that we really have to layout, so I'm going to go ahead and add a little bit more style so we can see it better. We will add a style to 'header, aside, article, footer' And the styles that we will add will be 'border-style:'.

So, we can see the edges. And we're going to set it to 'solid;' We're then going to set the 'background:' to be 'silver;'. And the last thing we're going to set is 'font-size:' to be a little bit bigger. And we're going to set it to '2em;'. We're going to go ahead and save. Now, if I reload the page, you can see there that we have in fact that layout. So, you can see here that we have the Header, the Footer, the three columns in the middle with a main content Article at the center. So, that's not bad at all. That was pretty quick. And you can read some of the history if you want to see some of the difficulties that have existed throughout time.

Video 2.4: Three Languages: HTML, CSS, and JavaScript

One of the challenges of working in the browser, the universal client, is working with three different languages. As you can see here, we have HTML, which is used for layout. We have CSS, which is used for styles. And then we have JavaScript, which is used for the logic. But let's take a look at what this means by writing a simple example. As you can see here, I have an editor and a browser side-by-side, and we're going to start off by creating a new file. Then we're going to save it to a directory called sample, and I'm going to call it hello.html. And inside of it, we're going to write 'Hello World!' Now we're going to go ahead and drag and drop that file onto the browser.

And you can see here that I am inside of sample, and that is the directory that I'm holding this in. And when I drag that in you can see there that we see Hello World! Now, we're using no syntax whatsoever. We're not using HTML, we're not using CSS for styling. We're not using any logic. So, the browser simply renders the text because there's no additional information, no additional instructions. So, in this case, what I'm going to do or what I'm going to do next is I'm going to add a tag, and this is part of HTML. So, what I'm going to do is I'm going to put this in the 'Hello World!' inside of the '< h1 >' tags. And I'm going to save the file, then I'm going to reload the page.

And as you can see there, that has changed. This is a header tag and so the browser recognized it. This was communicated in HTML. And so, it changed the rendering of this text string to match that instruction. Now let's say we wanted to style the header one, '< h1 >' with a style that was not that the default, the one we just used. You would do that using a style in CSS. And so we indicate that by creating a '< style >' tag. And inside of that, we write the tag that we're going to style, in this case, the 'h1' tag. And inside of it, we're going to style the 'font-size:'. In this case, I'm going to make it five times the normal size.

So, I wanted it to be nice and big. And then the 'color:' that I am going to use is 'green;', nice visible color. So now, I'm going to go ahead and save the file, reload the page. And as you can see there, the style has changed, and any other header tags that I have in the file, the same style would be applied to it. So, if I said here, 'Good night!' and let's get rid of this one. I'm going to go ahead and save. And as you can see there, we have the same styling. Now let's take a look at how you would add some logic, some computation. There's a number of ways you could do this. But in this case, I wanted to illustrate using a script.

And so, I'm going to go ahead and enter a '< script >' tag, just like before we entered a tag to express styles. And initially, we did one for layout. In this case, I'm indicating that this is going to be a script. And I'm going to create a variable and I'm going to assign to it the result of doing '3+3;'. I'm then going to write to the 'console'. And you'll see why I'm doing that in a moment. And I'm going to write the '(result)'. So, as you can see here, we have three things on the page. We have a style, We have a header one, 'Hello World!' that is appearing on the page and being styled.

And now we're going to write something separate that is just a few instructions that we want to carry out. And they're going, the result of them are going to be written to the console. So, let's go ahead and open up our developer tools as you can see here. And when I reload the page, we see the 6 that we expected to see, just making it bigger so it's visible. So, here we have the three pieces that I mentioned. We have the HTML, we have the CSS, the styles, and we have JavaScript, the computation that runs behind the scenes. And that could be updating styles, could be updating the layout of the document, or it could be communicating with the server. It could be carrying out many other things.