Styles and Bootstrap

Video Transcript

Video 1 – Introduction to Styles

The styling of a page may seem like an afterthought, something you do once you are done and your code is doing what you would like it to do. However, when it comes to delivering a good experience to a customer, to an end-user, the importance of user interface, a big part of which is styles has risen tremendously. And because of it, understanding styles is very important when it comes to a front end-user experience. We will give you here some of the basics to get you started. And you can invest much more on your own. If you're interested in user experience, this is a very rewarding career at the moment and it will be for quite a long time because creating effective, powerful interfaces that deliver value to customers to end users is something that will always be a challenge.

Video 2 – 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.

Video 3 – Separating HTML, CSS, and JavaScript Into Different Files

In our previous example, I created a simple string 'Hello World!' Then I added a header tag from HTML, which was 'h1'. I then styled that header using styles CSS and then I carried out some computation in a script block. Now the styles grow considerably on a site or on a page. The script rose tremendously as well and so it's convenient to break those off onto separate pages. I'm going to go ahead and do that next. I'm going to create a new file and the very first one that I will create will be called styles and the extension is CSS. I'm going to go ahead and save that. And then I'm going to create a new file called script.js.

And there's where I will hold my script and the .js is the extension for JavaScript. Now the first thing that I'm going to do is I'm going to take my styling and paste it onto the file. Note that I no longer need to use the style tags because this is a styles page. Then I'm going to do the same for the script. I'm going to take the script and paste it onto the script page. Once again, note I'm not using the tags. This is a JavaScript file, so there's no need to communicate that this is only going to be JavaScript. Then I'm going to go back and remove my tag and remove my script. And now I need to include those files within my hello.html file.

And first I'll bring in the styles and the tag that we used to do that starts with link. As you can see, there is a ' "stylesheet" ' in our file is ' "styles.css" '. This styles file is within the same directory, so I don't need to specify the path because they are in the same directory. Then I'm going to go ahead and add the script. '< script >', and I'm going to add a source. And in this case, the source is the name of our script file, which is ' "script.js" '. And you can call those styles the name of the file, whatever you like. The extension, in this case, is js for JavaScript files, and for styles is .css.

Now once we've done that and saved our file, our content should look the same. And if we reload the page, you can see that it does. So, let's go ahead and change it just to make sure we are live. And you can see there the change, change once more. Go ahead and save. I'll change this to be '1+1;'. And if we reload the file, you can see there the size of the header changed and you can see that the number at the Console changed as well. So as the size of your code grows, you'll find out that breaking out pieces of functionality, pieces of styling, it's quite convenient to clean up and get more clarity in your work.

Video 4 – Styling with Class

In previous examples, we have styled our HTML directly. That is, we have styled the tag itself. In this example, we're going to see how to style using HTML attributes. The attribute is going to be the class attribute. So let's go ahead and get started by creating a file, I'm then going to save, and I'm going to call it classes.html, and inside of it, I'm going to write some HTML. I'll start off by writing an unordered class. Inside of it, I will have a '< li >', and I will have three of these. And inside of that element, I'm going to add the 'class="" ' attribute. I'm going to add it to all three elements. And I'm going to enter some text, and I think I need a space there. Yep. I'm going to enter some text, I will enter 'One', 'Two', and 'Three'.

We have an unordered list, HTML syntax, and each of those list items has within it an attribute called class. We're now going to write the class style, and we will do that on a separate file called styles.css. And I'm going to start off by saving that file. I'll call it styles.css. And inside of it, we're going to write our style. We start off with a ' . ', and this is CSS notation. And I will call it the first one '.greenItems{}'. And inside of it, I'm going to add a couple of properties. Now before I do that, if you're curious about the amount of properties that you can use, you can see here a pretty long list of everything that you can use. In this case, I'm going to use font size, but there's many, many other, many other properties that you could use.

I'm also going to use some colors and I just wanted to show you that you can also find a list of predefined ones in MDN. Now inside of here, as mentioned, I'm going to go ahead and enter color. And I will, let me see that, I entered the wrong one. I'm going to go ahead and simply enter 'green;' the very basic green items, green color. Then I'm going to enter the 'font-size:'. And I'm going to make it twice the default. I will then use this as an example to follow for the next one, except I will call this one '.redItems'. I will change the color to 'red;', and I will make my 'font-size:' three times default. So, that's my styles.

And now, I'm going to use them on my list items. Here, I will enter ' "greenItems" '. Note that I did not use the dot. Here I will enter ' "redItems" ', and I will not enter anything for the third one, simply to show what happens when you do not have the style. I'm going to remove the class from there. And now I'm going to go ahead and drag and drop that file. I only need the HTML file. And as you can see there, we have no change. And the reason for that is that we have not yet brought in to this file, the HTML file, the styles. We do that by entering a link referenced to that file and then we enter the filename.

We do not need to specify a path because both of them are in the same directory. And so now after you save your file and you reload the page on the browser, you should see the following. We have our green items style, red item style, and we have our default styles, which is no styles at all. There are other ways to add styles to your document by using classes, but using classes is by far the most used one. This is what you will find if you review some of the major frameworks. And so it's one that you want to get some practice on, that you want to make sure that you're comfortable with the syntax and how to style your elements.

Video 5 – 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 6 – 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 7 – 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 8 – Resources

The web is full of powerful resources that you can bring in by including them in a tag within your HTML document. You can bring in libraries and create maps. You can use a tag for fonts, you can use a tag for styles, and you can bring in big building blocks that you can use for your application. In this section, we will take a look at some of those resources and point out those patterns so that you can go on to make use of the many more resources that exist.

Video 9 – Bootstrap Styles

Bootstrap is a free and open-source CSS framework made for responsive mobile-first front-end projects. Really a collection of CSS styles that had been bundled into one file that you can bring into your projects and it has topography forms, buttons, navigation, and really a great deal of UI components. This is a great choice if you don't have time to spend on your UI design or if you just want something very simple that will work and that is widely known to be a solid choice. So, what you see here is the website for Bootstrap, and we're going to navigate and take a look at how to use this style sheet. As you can see here, we have CSS only, where we only want to bring in the styles.

Or some of the JavaScript where we want to get some interactivity with some of the components that are offered. I'm going to go ahead here and start a file, create a new file, and I'm going to go ahead and save it, and I will call it boot.html. And inside of here, we're going to create some HTML boilerplate. And I'm going to remove this, this part of it because we're not setting, we're not working with the viewport yet. And so, this point here, I'm going to go ahead and bring in the styles and by performing this action, I'm going to bring in all of the styles in this framework. This is a project that is close to ten years in the making.

It has a tremendous amount of functionality. It is wildly popular, it has, it's always within the top ten of GitHub projects. Well over a 100,000 like stars. And so, there's a great deal of functionality, but by simply putting in that CSS, as you can see there, I have brought in all of that styling. I'm going to go ahead and wrap my line here so you can see all of it. You can see there that that's the link to that file. Now, once we've done that, we can go ahead and use the components that exist within Bootstrap. I'm going to go ahead and go to the navigation and select the documentation. And here, I'm going to look for the Navbar, and I'm going to scroll down to see an example. You can see here a pretty good Navbar example.

You're going to copy all of this, which you could use as a starting point when you are using this within your own work, and then modify it to your liking. I'm going to go ahead and put it into the '< body >' here and you can see there that it's the same text that I've pasted. I'm going to go ahead and save the file. And now, I will go ahead and load that file into my browser, and go ahead and drag and drop. And as you can see there, we have the same navigation bar. Now, if I wanted to get functionality into this bar, you know, make the pull-down work, I'd need to add the JavaScript libraries that we saw on the first page. And I'm going to go ahead and do that now. So, if I select all of that, and I go to the bottom of the page here, and I put them in.

Then you can see there that I've done the same that I did with styles, but now I'm doing it with the JavaScript part of it. And if I reload this, you can see here that now my Navbar is interactive and has all of the functionality. So, you can bring it in simply at the styles level, or you can bring in the JavaScript if you are making use of the interactive components. Now, let's say that next you wanted to add some widgets, some components, some blocks to layout content that is called cards within the speak of Bootstrap. And so, I'm going to enter here Cards, and perhaps go to Card style so we can get some nice colorful ones. Then I'm going to go ahead and copy. Let's copy all of them. And then I'm going to bring them into my document.

Then go ahead and save. And then, I'm going to go ahead and reload this page. And as you can see there, we brought in all of those cards. Each of them have a specific style here. And you can see the format that the framework here in the editor has. It's applying a class of card and then some additional styling. So, for example, in this case, the 'text' is 'white'. Down here at the bottom. The one right below it is 'light'. Let's see. In fact, it's not specifying it, which I suspect the default is 'dark'. Let's go ahead and reload that page. Yeah, and as you can see there, we can't see the text anymore. But it's a very modular off-the-shelf menu selection type of consumption of these styles. And you can see there that we have the card definition, then we have the ' "card-header" ', then we have the ' "card-body" '.

And so, all of the styling, which is significant once again, has been done for you. If we were to look at the file, and we can do so by navigating to the link. Let me make sure I'm in the right place. We can navigate to the link here. We would see all of that, and you can see here that it is, this is a minified file, but you can see the tremendous amount of styling that exist in there Now, the framework also provides some simple styles that are made, implemented into examples, as you can see here. And we're going to go ahead and navigate here to this one that shows a typical layout that might show some pricing, and we're going to look behind the scenes at the start, at the tags, and the UI components that are being used.

And I'll pull all of this here into our document. I'm going to go ahead and do away with the cards that we had pasted in. And I'm going to go ahead and put in that syntax that it's quite similar to what we did because we have some cards and we have some navigation bar. And now, if we go back to our document and we reload it, you'll see there that we have very much the same thing. We have those card components, we have another navigation bar, we have some content and the footer. There's an image that was broken because all I've done is taken the style, and that image was a local reference. So, you can see that you can move very fast and you can create something very quickly. That looks pretty good.

So, as you can see, this is a great resource if you don't want to spend a lot of time thinking about your styles in the front-end on your, or you're on a tight deadline and you don't have time to get to your styles. However, if you're trying to learn CSS and you're taking Bootstrap styles as a starting point, you are going to be fighting bootstrap all the way. And it's much better to start from scratch and build up from there. The other thing to consider as well is if you only need minimal styling, perhaps some of the other CSS functionality and features are good enough. Like CSS Grid and some of the other components that are native to the language itself. So, it is a great resource. Just think about what your use is for it.

Video 10 – Font Basics

Styling fonts is one of the most popular parts of CSS. Oftentimes if you only style one thing, you will style fonts. So, let's take a look at an example. We'll start off by creating a new file. I'm going to go ahead and save it, and I will call it fonts.html. Inside of it, I will paste some boilerplate. Just going to change the '< title >' here to 'Fonts'. And inside of the '< style >' tag, we're going to enter some styling for our fonts. Now, before we enter the styles, I'm going to go ahead and paste in some Latin so that we can tell the different font styles. Let's paste that in a '< p >'. And I'm going to go ahead and wrap so that we can see all of that. I'm going to do one more. And now, let's take a look at some of the properties for styling fonts.

We'll start off with the 'font-family'. Let's put that inside of a style called '.s1'. And we're going to give it a pretty common font. We'll call it ' 'Courier New' '. And we'll create a second style. Or we will set the 'font-family' to be something more common, let's say, 'verdana;'. So, that's where all we're going to set. Let's go ahead and go to our blank page, and let's load our file. And as you can see there, we have no changes because we have not yet applied our styles. So, now let's go ahead and enter our class. And for this one, we will enter the first style. Here, we will enter the second style.

So, let's go ahead and save, reload the page. And as you can see there, we have very different fonts. Now, that's a bit small. Let's go ahead and set the size to something more visible. We'll do '3em;'. Do the same thing here. And let's set it for now, at the same size. Let's go ahead and reload the page. And as you can see there, they're too big. Let's take it down a bit. Let's put '1.5em;' and '1.5em;'. Reload the page. And as you can see there, we have more, a more visible font. Now, there's a number of other things that we could do. We could vary the style, the variant, the weight.

And let's take a look at one more. We'll take a look at the variant. So, we will enter 'font-variant:'. And in this case, we will say, 'small-caps;'. And on the second style, let's set the 'font-variant: normal;'. Now, if we reload the page, as you can see there, we have very different styles. So, let me go ahead and go to '2em;', so we can see that better. And you can see there the difference in the styling. Now, there's a great deal more properties that you could add, and I will leave for those, I will leave those properties for you to discover.

Video 11 – Web Fonts

Fonts are important for branding, for readability, for typography, and for making pages beautiful. And web fonts make the web better. They make it more accessible, more legible. If you take a look at this page here from NASA, I scan the page, you'll see that we get this big block where the conference announcement is. And that is because that is an image that text within it is not selectable. It will not be indexed by the search engines. It will now be zoomable without degradation. and it has a lot of limitations. And so, having web fonts is a property that has been sought out for a long time. And that was realized with one of the most successful projects that Google has, which is web fonts. And let's go ahead and navigate to that site.

And it's simply fonts.google.com, and as you can see here, there are a number of fonts if I scroll down, and there are actually hundreds and hundreds of fonts, and these fonts are very successful, successful at the count of trillions of downloads. One specific successful font, which is Roboto, has been downloaded more than 12 trillion times. And so, we can use this as a resource in styling our pages. So, I'm going to go ahead here and navigate to a specific one that I selected that is called Tangerine. And you can see here the font. And I'm going to select here Bold 700. And as you can see there, I am given some guidance as to how to include this within my page and also the CSS rules that I can use within my styles.

So, let's go ahead and use it. I'm going to go ahead and create a new page. I'm going to go ahead and save it, and I'm going to call it gfonts.html. And to begin, we're going to write some boilerplate here. And I'm going to type some text and I will put it within a paragraph. And I will write < p >Making the web beautiful!< /p >. Now, I want to use that font from Google Fonts. But before I do that, let me go ahead and load the document into the browser. And as you can see there, we see "Making the web beautiful!" with the default style. Now we're going to go ahead and add our new style or the style we're going to bring in from Google Fonts. We will style the entire body. And that's not what I wanted. Let me go ahead and make sure I have the right one. And now we're going to bring in our font.

The font that we're going to use is Tangerine, as we mentioned. And we need to bring in this CSS for that style. So, this will import that font that does not exist and let me go ahead and wrap my text. That font does not exist in the browser by default. So, we need to bring it in, in a style sheet as you see there. And we have then set it as the 'font-family:' within the body. So, let's go ahead and see what that looks like within our document. Now, let's go ahead and reload the page with the tangerine style. And once I do, and you can see there that the style is quite different. And we have done all of that by bringing in those wants from the Google Fonts project.

All of these are vector, which means that they will scale to any sized device, smaller or bigger. And they are cached in many places, and as mentioned, once you bring it into your document, you can reuse it and cache it without having to download it again. Referencing resources as we have done here when we look into our external style sheet is a powerful mechanism for bringing in extra functionality into the browser be it data, programs, or in this case, styling. So, go ahead and give it a try and make sure you're comfortable with using these resources.

Video 12 – Applying Styles Programmatically

You can set your style statically using declarative statements, or you can do so dynamically using code within the page. In this case, we're going to do it dynamically. So, let's start off by giving the page a '< title >'. We will give it the name of 'Dynamic', then we will create some styles and we'll start off with '.on' style, a very simple style where all we're setting is the 'background:' of the div that we will write in a minute and I will give it the color of 'orange;'. Then I'm going to give some general 'div' styles. And in this case, I'm going to set the 'padding:' I want it to be nice and broad so we can see the div very well. I'm going to set the text to align in the 'center;'. Then I'm going to set the 'border-style:' to be 'solid;'.

And last, I'm going to set the 'font-size:' to be '2em;' nice and big font. Now inside of the body, we're going to write, a few UI elements. We'll write a couple of buttons and a div. And so, my first '< button >' will be the 'OFF< /button >', and I'll use that to turn off the style. I will have another one to turn it on. And then I will have a '< div >' that will be the target to apply that style to. So, let's give that the name of 'Target'. Let's go ahead and save the page, reload it. And as you can see there we have a nice big div. We have an OFF and an ON button. And now the next thing that we're going to do is that we're going to create some onclick events to call some functions that will change that style. As mentioned in the beginning, we want to do so programmatically.

So, let's go ahead and add that event. And we're going to call this function ' "off( )" ', which we'll call the off function. We will write that in a minute. Then the onclick event for on, we'll call the ' "on( )" 'function, which will also, we will also write in a minute. Then we need an id for our div so we can get a handle on it. So, we're going to give it the 'id' of ' "target" '. Now, we're going to go ahead and write some JavaScript in the < script > tag. And we're going to start off by writing the 'on( )' function inside of it. We're going to get a handle on that target, that div, that has id of target. And so, we're going to start off here and put that into a variable called 'element'. And we will access that through the 'document.getElementById'.

And the id of the element we want is '('target');'. Once we have the element, we will get a hold of the 'classlist', which is a property within it. And then we're going to '.add' the function that we wrote, the function of on. Now we will go ahead and add that '('on');' And so, at this point, I'm going to go ahead and save, reload the page. We can go ahead and add that style. Now we can't take it off because we have not written that function yet, but we can add the style. So now, let's go ahead and write the function to take it off. We're going to do the same thing. We will get a handle on that element. And then we will access, once again the 'classlist'.

However, in this case, we're going to remove the class. And so, the class we're going to '.remove' is '('on');'. Let's go ahead and save, reload the page. And now we can add the style, remove the style, add the style, remove the style. So, as you can see that we're doing this now programmatically, it is not set. We can change it while the page is live. And so, this same technique can be applied on a very rich UI where you have hundreds of elements and you want to make sweeping changes dynamically. This is a very powerful technique. Even though we have done a simple example here. Play around, add more styles, add more divs, and get comfortable with changing things programmatically.

Video 13 – Using Timers and Styles

In addition to changing styles dynamically, we can do so using a timer. As you can see here, I have the same 'Styles' in our previous example. I have a class that has a 'background' of color 'orange;'. I then have a general style for divs. In this case, 'padding' of '50;', the text is 'center;', the border is 'solid;', and the font size is to '2em;'. And then I have three buttons, in this case, one that will 'TOGGLE' the style. Then I have a 'START' and a 'STOP', and this will refer to the timer in this case. And then I have the 'Target< /div >', which is the one you see here in the page. Nice big div that is visible with a solid border. Now, let's go ahead and write our functions to toggle the styles, and turn off and turn on the timer that will do so automatically for us.

So, let's go ahead and move into our '< script >' block. And inside of it, we're going to write the 'function' of 'toggle( )'. And we're going to start off by getting a handle on the target. We do that by accessing 'document.getElementById'. And in this case, the name we need is '('target');' for our 'div', which is this one, here. And now, we are going to toggle the class. So, instead of turning, adding, and removing a class, in this case, we will toggle the class. So, that is if the class is on, it will be taken off. If it's off, it will be turned on. So, let's go ahead and 'toggle' and the class we will be toggling is the class of ' ('on');', which is the one here.

Okay. Now, we need to write the function for our timer. And we're going to start off by creating a variable that's outside the function so that it has scope and it's visible by the other functions. And we'll call it 'timer', and I will initialize at 'null;'. Then I'm going to write the 'function', which I will call 'start()'. Then in the body, I'm going to write a 'timer'. And this timer is going to return an identifier that we can later use to turn off the timer. So, here we will create a timer using 'setTimeout()'. And inside of it, we will write a function using the fat arrows notation. And this function is going to 'toggle', call the function that toggles the style. And then if we're still running, it will call itself.

It will call the start function so that it can continue to run. So, let's go ahead and enter that. And we will have it execute every second. Now, we need one more function to stop the timer. So, I'm going to go ahead and enter 'function stop()'. And in the body, we're simply going to use the command 'clearTimeout()', which is part of the language. And we will use the handle on the timer, which is 'timer', to be able to get read of that timer. So, let's go ahead and toggle that style. But before I do that, let me reload the page. And you can see there that I can toggle the style. I'm doing it by clicking on the TOGGLE button.

Now, we're going to do it programmatically by clicking on START. And you can see there that the style is changing. The class is toggling the style or the code, and the function I should say is toggling the style, and it's applying it and removing it on every iteration of that timer. Now, we can use the function 'stop' to stop the flashing. I'm going to go ahead and click it. And once I do, the timer will be destroyed and it will no longer flash. As you can see there, it has stopped. It stopped on, on, in this case, but the function is no longer applying and removing the style. We can go ahead and make this even more explicit by writing a 'console.log' here that says, '('hello from timer');'.

And we will see that that is written to the console. So, let's go ahead and reload the page. Open up developer tools. Let me move this a little bit down here so we can see. Then I'm going to go ahead and start the timer. And as you can see there, we get 'hello from timer', and the little number here continues to increase every single time the same message is written to the console. So, as you can see, not only can you apply styles programmatically, you can do so as well using a timer, and you could write conditions after which certain styles will take place. And you can see that this would be pretty useful for a certain UI states where you want to highlight something within the page or you want to have constant communication say, with the server and you have some repeating function that is firing on a certain interval.

Video 14 – Styling the Grid Programmatically

In this exercise, you're going to be working with a 3 by 4 grid. And each of those cells is going to have the following properties. You can see here the wrapper for the grid, which is 'main', and then you can see each of the elements underneath. Notice that the ids are numbered ' "s1" ' through ' "s12" '. Your work is going to be to create the ' "move()" ' function that has the onclick event on the button that you see here and the UI that once you collect the move button, this style is applied to each of the cells in order throughout the entire grid. The style that is being applied is the '.on' style that simply has a 'background:' of 'orange;'. You will write your code in a '< script >' block at the bottom of the document and review the lessons on timers and styles, as well as applying styles programmatically within the page.

Video 15 – Animated Style Application

In this exercise, you have a 10 by 10 grid, as you can see on the screen. And then you have the following properties for each of the cells. In the main body, you can see here that we have the wrapper of main, and then we have the long list of elements, 100 elements, as you can see on the page, your job is going to be to write the function to move a style throughout all of the cells. But as opposed to the previous exercise, you will be cleaning up the cells behind you. That is, you are moving one style throughout all of the cells and you're cleaning up the styles that you applied to the previous cells. So, let me go ahead and demonstrate here. As you can see there, the style is moving throughout the page, throughout the grid, and there's nothing that is orange behind it. Now the goal here is not for you to write a lot of code. In fact, you can write it in less than 20 lines. But to make use of timers, to make use of applying styles programmatically, and to figure out how to clean up as you move forward.

Video 16 – Bringing It All Together

We started out very simple by applying a style programmatically to a single cell. We then added to that, being able to do it with a timer. We then added the styles programmatically throughout a grid. And in this case, we're going to create the grid programmatically. As you can see here if I reload this page and I show you the code that we have here. There are no tags for the element of the grid. We simply have a 'main' tag and an 'id' of ' "target" ', but we have no elements. This solution, if I reload it, I can create the grid programmatically.

That is, I have a function that is being called by "createGrid( )" that creates all of those elements and adds them to this tag here. Now in addition to that, once you've created this tag programmatically, we want you to once again apply those styles dynamically throughout the page in order, taking into account that you have now created those elements programmatically. So, you have to take into account what your naming convention was to be able to style each of those elements and move throughout the entire grid.

So, this example captures all of the different parts that we saw previously, which were timers, which we're applying styles programmatically, which we're moving through a grid, which was cleaning up, and in this case, creates the grid itself. So, this is a good exercise on being able to manipulate a UI fully at many different levels, doing it programmatically. And in this case, focusing on the use of styles.

Video 17 – Eye Exercise

So, in this exercise, we're going to give you a web page with a single eye on it that follows the mass movement. So, you can see here that as I move the mass, the mass events are being picked up and we're moving the eyeball to match the mass. Now, what I want you to do is to convert this so that we have two eyes. Okay, let's see what that looks like. So, here's the two-eyed one. And now both are synchronized, they're both following the mass movement. So, I need you to add one extra eye so that it looks like a person following the mass. Okay, that's the exercise.