00:00 I'm gonna call it a race. Now, go ahead and do a shortcut exclamation tab, and give it a meaningful title.

00:14 And I'm also going to go ahead and link up, our JavaScript. And I'm gonna give it the same name, a race.js, and put a different, let's go ahead and make that a race.js.

00:35 Yes. And We're gonna start with an, an array called steps, and we're gonna use square brackets to give it three string values, the string of the word one, and, two, and three.

00:57 Alright, let's go through some of the array methods that we learned about. let's try the for each array method. You're going, to mention the array's name, which is steps.

01:06 And then add that method with dot notation for each. And it's camel case with just one initial cap on the second word.

01:15 And then you're either going to call a function, or you can put an anonymous function in here. Okay. Okay. But we're always going to have a parameter coming in.

01:24 And even if you call the function, you don't have to send in an argument to become that parameter. I could have put, um, anything here.

01:33 I could have put x or, um. steps, or whatever, whatever we, I wanted to get. Item is common. And now if we console log each one of those items, and we'll take a look and see what that looks like.

01:51 So, No. For each is the method that's going to go through each element of our array and do something to it.

01:57 And in this case, we are just having it go on to the console. And there we go. One, two, uhh, and three came on to the console, line four, and that's where the console log is.

02:09 So we know that that dot for each, the first time it goes through item will be the string one. It will get called again for however many elements there.

02:20 So we in the array, so the second time this will be two, and then the third time this will be three.

02:26 So three different times this anonymous function ran because we have three different values in our array. So that is the four each, just doing something.

02:36 With each element of our array. Alright, so let's try the same array, but let's go ahead and show how we could have just called a function.

02:49 Just so you can, I see the difference there. We haven't really talked. It's a lot about functions yet. We're gonna come to that later, but we can also do something like this.

02:58 Steps, again, the same for each method. But this time I could call a function called show steps. And then have a named function instead of an anonymous one this time.

03:12 Same name, show step. Maybe I should put an S on each one. And then again, we could put item. And then in the curly braces.

03:24 We'll do the same thing. We'll just go ahead and console log that item. And I commented out the other one quickly with a commander control backslash.

03:34 So it's not running anymore but this one should be and it should do the exact same thing. So instead of just having an anonymous function , as the parameter, we are just calling a function as the parameter and then it will come down and find that and run that for each one again.

03:48 One, two, and three. We'll go through that function. Now, you don't always have to call it item. Like I said, you could call it X and it will do the same thing.

03:57 You could make it more meaningful and just call it each time a step is coming in, so don't confuse that with steps with an S, but step.

04:04 As long as whatever you refer to the parameter here and you are also referring to it inside the curly braces, you should be fine.

04:12 With whatever word you decide to use. Alright, so that's for each. Let's try dot map. Now dot map function or array method is going to also go through each element of the array.

04:27 But it's going to change it in some way. And it's going to be stored in a new array. So an extra step here of creating at a new array.

04:38 let's go in and add an ordered list. That's just going to be empty. And I'm going to give it an ID of my list with camel case.

04:48 And now we can use that to get some outputs instead of just the console this time. let me make some- comments here.

04:56 This was for each, and this one's going to be .map. So let's set up a variable called my list. Yes.

05:10 And it's going to be the element that we're targeting, and that's going to be the empty UL that we just put in there.

05:20 Oops. That should be a hashtag. Okay, so my list is now storing that element dot empty UL. set up a new variable And we're going to call it steps HTML.

05:34 And I just use camel case there and we're going to refer to the same array that we had up here in line two.

05:42 Steps, but instead of for each we're going to do map. And it will call a function again called this template.

05:59 And again, we could put step in here or X or whatever. X isn't very meaningful, but you could put it wherever you want it there.

06:06 And we're going to- Now this is how it's a little different. It's going to call every single one again, one, two, and three.

06:13 It's going to go into this function as item. But then it's going to return a value each time that goes into this new array called steps HTML.

06:23 So we're using this original array which won't get changed, but we're using the values in this array to do something to them.

06:31 We're going to return it somehow. We're going to process it somehow and it's going to get returned back into here.

06:36 So steps that HTML will have the same number. But they'll be changed in some way. And how we're going to change that is we're going to create them.

06:43 We're going to turn them into list items or the li element that we can put into that ul. And because right now they're just a string, we need to surround that with the actual li tab.

06:55 So that we can put it into HTML, HTML dynamically. So it's not going to be static like I could come in here and just create them here, right?

07:04 I could say one, two, three. But we want to do it dynamically through JavaScript. So this is going to be a little different.

07:15 So we want to return, and I'm going to use a back tick here. If you've never used back ticks, right below the escape key.

07:24 It's not a quote there. It's, It's a back tick. And this is called a template literal. And it's going to be a little more than a string, because it's going to allow us to put this kind of variable of item inside of it.

07:39 And how we do that is we're going to, have a dollar sign, and curly braces, and then we'll repeat that parameter, name of item, and then we'll finish out the ending l-i tag.

07:53 So maybe in our JavaScript, this array changes, like, these values might change according to maybe something the user is doing, and we don't know what strings are going to be each time, but we do know we want to put them on the screen.

08:08 So we are going to just go ahead and say, you know, look at that step's array, and I'm, We want to put into a new array here called steps.html, and all we're doing is adding those li tags on either side.

08:21 But because we don't know what the values are, we can plug that in kind of like a variable in between those opening and closing li tags.

08:29 So the first. Time item will be one, second time two, second, third time three string. So let's go ahead, go ahead and add my, semi-colons.

08:44 but I haven't put it onto the screen yet. we have targeted that empty UL right here. So we're going to go ahead and.

08:43 I'm going to say take that empty UL that I targeted and change its inner HTML to equal the steps dot HTML, which is that new array with the three values with the li tags around them at this point.

08:59 And we can. And see it will come on to the screen as list items. And if you go into the elements here, you'll see in that UL that we actually have those allies coming in, which is kind of cool.

09:11 It did. It does put a comma in between each one. So there is a. A handy little, uh, method you can put here, and it's called that join.

09:20 And you can join it with whatever you want. If we leave it empty parens, or I'm sorry, empty quotes, then it will just put nothing in between.

09:28 You know, we could put dashes. You could put whatever you want. But since we wanted to get rid of those commas, we'll just say, let's not put any, let's join each one of those together with nothing.

09:38 And it will look a little better. Okay, so there, that was our first little bit of dynamic content. This is how our HTML started with.

09:48 And we dynamically brought in three array values and onto our page by making them HTML elements using the dot map.

09:58 This original array is unchanged. Steps is just fine, but we now have a new array called step study. HTML that we were able to put onto our screen here.

10:08 We didn't just console log at that time. So pretty cool. Just remember, those are back ticks and And you do need this dollar sign.

10:14 You may see an older, um, code. Regular quote marks, but then you've got to do a little extra here. You won't have the dollar sign curly brace, but you would have a concatenation, um, symbol, which is the plus, and you'd have to do something like that.

10:34 That would do a similar thing, but it is best to probably just go ahead and use those back ticks, and you don't have to worry about the concatenation and getting all the right quote marks.

10:50 But you do need to put the item coming in, or the argument, or the parameter coming in, inside of the dollar sign curly braces.

11:01 So, um, nope, I need a back tick there. Okay. So, just remember you may see it this way in, in some older code.

11:09 That really doesn't matter, but this is probably a little more efficient, especially when we get to multi-lined, um, dynamic feedback that we're going to be putting in with larger template literals.

11:19 So, every time you use this back tick, and the dollar sign- uh, curly braces, it will be called a template literal.

11:27 Alright, so we've got for each, and we have map. Let's try one more map. This time we're going to be looking at a GPA.

11:38 So say you're a student, and you have some grades, and they're going to be an array of all the different grades that you have.

11:46 So in one class you had an A, in another class you had a B, and in your really hard class you got C, and those are your- grades for the semester.

11:56 And you want to get a GPA points from that. So we're going to go ahead, and we're going to set up a point variable.

12:04 We don't have to give it a value, just so that we have that. So when we start using it, we don't have to put lead again.

12:10 And we're going to use- a switch statement. And I actually have this one in here, because it's a little bit of typing.

12:16 So I'm going to copy that and paste it from our ponder. And, oh, I did call it points with an S.

12:23 So make sure you have an S on that variable called point. And go ahead and try out a dot map on this one.

12:33 So this is kind of like, you know, if they choose A, then it's going to get four points. If they have a B, it's going to get three points for their GPA.

12:40 If they have a C, it's going to get two more points. One for D, zero for F's. So we could have done it big if else, if it helps statement, but switch worked pretty good here.

12:51 And we'll go ahead and use the map this time. So again, map is making a new variable. So we're going to set up a new variable.

12:57 This time it's going to have, the GPA points in it. Whoops. So we'll call it GPA points. And we'll take our original grades array.

13:07 We'll map it. So each one of those strings, A, B and C, are going to come through. And they're going to go through this variable.

13:15 Or, I'm sorry, this function called convert. So A is going to come through as grade the first time. It will hit K, say, and it will return point.

13:26 It will assign points to four. B will go in again as grade and see how it's not- An item this time it's grade.

13:33 And B will then have points be assigned three and so forth. Each time it's going to return those points back into GPA points.

13:42 So again, we're going to have a four returned here, a three. And then a two. So GPA points, we'll go ahead and console log it, should have those values in there.

14:24 Okay. So there we go. Our new array, GPA points, is getting, It had returned the points that it got each time to GPA points.

14:06 So the first time A came through and it returned to 4, B came through as grade, it returned to 3, C came through as grade, and it returned to 2, 4, 3, and 2.

14:16 Or what's showing up when we do console ones. Now notice if they entered an odd number, even if it's just lower case, they're going to get an alert that says not a valid grade.

14:29 So again, is case sensitive and it wouldn't have found any case that was true. So it would have gone to the default.

14:37 And notice the break is going to make so it doesn't check all the other cases once it finds the, the right case.

14:44 It's going to just get out of the switch. Once it finds one, but it does run three times because we had three strings going in and that's why we have three values and our new array that's called GPA points.

14:56 let's do the dot reduce method now. down. We're going to continue with our GPA example. we're going to go ahead and instead of just getting a GPA point for each grade, we want to kind of merge them all together and get your GPA for the whole semester for all three of those grades.

15:14 So let's go ahead. Add and set up a new variable here called total points. And it's going to be taking our new GPA points array that has the fourth I'm cuz my Okay.kered your turn.

15:28 And three and two in it. And we're going to use the dot reduce array method. And we're going to call a function called get total.

15:40 And remember the. Dot reduce function can take two parameters. One is the accumulator, which we'll call total. And again, you could use whatever bird you want.

15:51 And then the second one is each item coming in. So this time it's going to be four, three and two coming in.

15:58 As item. And then total is going to be accumulating those together. Okay, so we're going to return. It the accumulator is going to be adding each of those up.

16:14 So what's happening here is it's going to be four is going to come through and then total will be four.

16:19 Three is going to come two and then total will be seven. Two is going to come through and then it will be nine.

16:24 So it's just adding each of those numbers on. So if I was to console log. That Okay, there we go.

16:33 Nine. Find 63. Okay. But that's still not our GPA. So what do we, What need to do to make it our GPA?

16:41 We need to divide that by three, or if we didn't know how many grades we're coming through our function, we could just say by the length of the array.

16:50 So again, let's go ahead and put GPA average. A new variable that's going to take those total points, and divide it by GPA points dot length.

17:09 Dot length is a property that's just going to get how many values are in that. So we know there's three, four, three, and two.

17:16 So it's going to have three there. So we could have just said three, right? So that would have worked. But sometimes we don't know how many are coming through.

17:23 We don't know the length of our array. So this is a nice way to, To just get, grab that length there.

17:31 And then we can go ahead and console log this one. The GPA average this time. And there's where we should finally get our GPA.

17:43 Because we have a three. So you could play around with it and you could say, no, I got an A, B, and an A.

17:49 And my GPA will come up a little bit. And I got straight A's. I'd have a 4.0, right? So that is the idea.

17:59 Umm. With reduce, you're going to get one value coming back from a number of values. So total points is getting a number of values, the 4, 3, and the 2, reduced down to 1.

18:10 In this case, it was totaling them together. Then we divided it by 3. So, so that we could get the average.

18:19 go ahead and try the dot filter array method. Okay, this one will do a lot like map, but it's only going to return it to a new array if it passes a certain condition.

18:31 So we're going to go ahead and grab this array that has a bunch of fruit names in it, strings, and the, array is called words.

18:43 And it's got, let me turn this off so we have a little more room. It's got five different fruit. And we're going to go ahead and use the filter with this one.

18:52 So again, it's going to make a new array, so let's give a new array an- ehm, short words. We want to just have the short words come out of that array and be assigned to this new array called short words.

19:03 So we're going to take the original array with a dot filter array method. And this time, we'll do an anonymous function instead of.

19:11 Calling a function. Sending in, instead of item, we'll go ahead and put word this time, or you could have put fruit or whatever.

19:19 And we'll say, only return this if the word length, there's that length property again, is less. Yes. Then, six. So word length is going to return however many characters there are.

19:34 So, watermelon would be one, two, three, four, five, six, seven, eight, nine, ten. It's not gonna be less than six.

19:42 So, w-so, watermelon. And will not be assigned to short words. Okay. Peach well, because it only has five and so forth.

19:52 So, let's go ahead and run that. And we'll go ahead and console log the new array called short words. And it should only have words in it that are less than six characters.

20:05 Peach, apple, and grape were the only ones that came through. Now, tomato had exactly six characters. But we didn't say less than or equal to, or, So it would have showed up.

20:17 We just said less than six. So, only those that are truly less than six are going to end up in the new array called short words.

20:26 So that is the dot filter. Filtering out results depending on a certain condition. Alright, one last array method and we didn't really see those, haven't really seen this one before, but it's called index of, again, camel case.

20:43 And we're going to go ahead and Called my array and we're going to assign it this time, just numbers, no strings, so we don't need quotes this time.

21:13 what index of will do is it's going to return the index value of one of our re-alments. So we want to get that into our lucky index.

21:19 So instead of returning twenty-one, we're gonna actually want to, We have zero, one, two, and then fifty-fourths, three. So two would be the index of our lucky number.

21:30 So we want to say our lucky index. And then here's where the array method comes in my array, so we reference the original array.

21:38 Put the array method index of following it. lucky number, which is going to put in twenty-one. So we're putting in twenty-one in there.

21:49 And what's going to do, what it's going to do is it's going to say, hey, in the my array, which one of these is twenty-one, and it's going to return the index of zero, one, two.

21:59 So now if we console log, lucky index. We could, umm, lucky. We should see two show up here in our console.

22:17 Okay, so if you ever need your reference in index of an array item, then you would use the index of, array methods.

22:24 So, a few different array methods there. Let's go ahead and go a step further toward how our ponder and our prove assignment is going to be.

22:34 And we're gonna be looking again at some more template literals. As well as an array of objects this time. So, this is a little bit new.

22:45 We're gonna go ahead and we're gonna add a div. To our HTML. And we'll give it an ID of student container.

22:58 And again, we're just gonna leave it empty because we want to dynamically put umm, some content in- into there. And we're gonna come back here.

23:09 And we're going to have a different kind of array this time. So this time, actually, let's go ahead, let me make a thing We'll call it dynamic content.

23:21 And let's go ahead and target that container really fast. That student ID container that we just put in each TML, so we've got it ready to go.

23:29 To use later to put some information into. And it was an ID, so student container. And we have a new array.

23:45 And we're going to go ahead and grab it from here. So we'll copy that code. And this is a little different.

23:51 So we could have had this in one big line. In fact, we could have even had it kind of looking like this.

23:58 Sometimes you'll see array. Um, object arrays like this. Especially if they have a lot of different property value pairs. Anyway, it doesn't matter.

24:10 Whatever is good and readable for you. Um, so what we're looking at here is still an array, but instead of a, a number or a string, we actually have an entire three objects.

24:22 And these represent students. So this is an object. This is an object. And this is an object still separated by commas, just like our other values in their arrays.

24:30 But this time we're looking at name values. We have that go together. So this is one student with the last and first name, another student, and another student.

24:39 We have our name value pairs, last is the name, addresses the value, first is the name, Aaron is the value.

24:47 And they all match. Now that you have a little last and first name value pair, but of course, different values for different students.

24:53 We will see more about objects in a later lesson, especially when we start talking about making our own methods for our own objects.

25:02 But this week we're just going to be looking at the these name value pairs. So now how can we use one of our array methods that we've learned to actually go ahead and put some information from this object array onto our page?

25:17 Okay, so we want to just kind of go through each element of our array or e- each object of our array and grab some information from it so that we can put it on the screen.

25:27 So let's go ahead and try this out. So we're going to refer to our array called students and we're going to use a 4-e each.

25:35 We aren't filtering. We're not making a new array. So we're just going to be going through each one. I'm going to go ahead and use an anonymous function here.

25:43 I'll use the old item as the parameter. And we're going to go ahead and create a new div. So what we want to have happen is, inside of this div, we want to have three more divs with the first and last name of each student inside of it.

26:01 So we're going to go ahead and say, hey, let's say me. Let's set up a, uh, variable called name. and we're going to create an element.

26:10 So remember from the DOM lesson, you can create an element and we'll just call it a div again. Could have been paragraph, could have been any other one.

26:18 We'll just have a div inside of a div. And then, we're going to take that div and give it a class.

26:25 And that class is going to be called format. So what we're doing here is I want to show that you can create an element for each item that we're going to bring in as well as give each of those elements a class.

26:37 Now we don't have any CSS for the class yet. So let's just throw it in here. I'm not going to take the time to make a whole new file, CSS file.

26:45 It's not the greatest to put, um, the embedded CSS, but we'll go ahead and make that format class. And we'll quickly just go ahead and put something like text.

26:56 I'm going center. Just so you can see that you can also make a class on the fly in JavaScript as well.

27:03 And assign a new created element, a class. Alright, so that would be a little bit of a review because we've seen that with the DOM already.

27:12 Now let's go ahead, and while we're still going through each one of these, let's create some HTML that's going to go on our page.

27:22 And again, I'm going to use the back text. This time I'm going to kind of put each one on their own line just for readability.

27:28 You don't, you don't have to. But I'm going to create a span just like we did the allies before. Now we're going to do a span.

27:36 And because we're inside back text, we can use that temporal, um, kind of variable information with the, dollar sign curly braces.

27:47 We did call our parameter item. Now we are going to have to have a dot first. Reason we need this is because when we say item, it's going to bring this whole thing in.

27:57 And it's not sure which part of this item we want. So we do need to say I want the first name.

28:03 I want the property or the name value pair first in there. So that's why we needed to dot first there.

28:10 Because this span is just going to contain part of the object, the first name. And again, we can do a very similar theme for the last name.

28:22 Another span. And I'm using spans so they can be side by side. Whoops, item last. This time we want their last name to come in after the first.

28:37 put that new name div that we just made and change its inner HTML to be.

28:43 That template literal. So assign whatever this created into that new div that we just created here. And then we're going to get it on the screen by taking our container.

28:59 Which was our empty div that was already in HTML, the student container, that we had actually referenced right here. And we're going to say go ahead and append child name.

29:14 Okay. And that's the name that we just put that HTML in to. So now if I save that. Okay, I have a typo here.

29:23 I do want to end that with a, uhm, a semicolon, but I do need a cap. I'll enter HTML here.

29:30 There we go. Now we get our three names to come up here. So pretty cool. And we could actually even add other elements in here if we wanted.

29:39 If we wanted to put, like, a horizontal rule between each one of them. And we could- also come in here and put some CSS to go with that.

29:51 HR. We could give it, like, a smaller width. 200 pixels. Something like that. So there's some cool things that you- You can do, um, dynamically with creating this.

30:07 And we didn't even have an HR on our page to begin with, but we went ahead and just added one along with each of the two spans so that each name had a little line up below it.

30:17 So, some pretty cool, different things. And if you look at the element tab here, you can see that inside of that div, we have three more divs with the class format because we added that class right here.

30:32 And it would hit. And did that CSS because of that, and you need each side of, you know, inside of each of those new divs, we have the two spans showing up and the HR.

30:42 So, using a template, literal, with the back tics, to create that h- HTML, we created a brand new div going in.

30:52 We used, instead of just a regular array, an object array as well. So, getting a little closer to what you're going to be seeing in the next ponder and in your homework.