Course Overview

Course Overview

[Autogenerated] Hello. My name is Paul Chiney. Let me tell you about this great hands on JavaScript course. I believe that the best learning comes when you and I worked together to build projects that you might see in the real world. In this plebiscite course, we will be learning about JavaScript functions, loops, switch statements and gathering information from the document Object model or D O. M. Using an authentic hands on approach. Next, we'll apply everything. We have learned to create a web out that allows customers to calculate how many solar panels they need to power their home. Now, I realize you may not need a solar calculator for your next client, but the information you learn here will transfer to other JavaScript projects you work on as you do. So you will remember what you learned so you can apply it in the future. That's the power of hands on learning. So join me as we work together to learn the fundamentals of Java script

Introduction

Introduction

[Autogenerated] Hello, This is Paul Cheney, but then you already knew that. So let's get started. You cannot be a Web developer unless you know Javascript. So the purpose of this course is to give you hands on practice with some of these JavaScript concepts. During the last module of this course, you will combine the concept you have learned to make a fully functional solar calculator to motivate you to stay with me. Let's take a peek at the final project. This is where the final project will look like on a phone. Here's the same project on a tablet and finally on a desktop computer screen. But customer will input their monthly kilowatt usage up here and then use a map to estimate the number of hours of Sunday get per day. Then they select a solar panel from the drop down list and click the calculate button at this point that JavaScript takes over and does its magic. It is my belief that if you can see how JavaScript concepts fit into a real world, application will be easier to learn and remember. So let's take a quick look at the entire course in module to we will explore JavaScript functions while passing values in and out. In Module three, we'll explore loops and several ways to track your JavaScript variables. In Module four, we'll explore a switch statement along with an HTML select element in Module five will practice extracting information from the document Object model or D. O. M. Finally, in Module six will combine everything you have learned and build a responsive, mobile friendly Web application to calculate the number of solar panels you need to run your home. Throughout this course, you will not only learn what is happening, but you'll be making it happen on your computer. To be successful in this course, you will need some kind of text editor like sublime tax, dreamweaver brackets, visual studio code or even text ring with no pad plus. Plus, you should also know how to connect an external JavaScript file to an HTML document and how to access the JavaScript console in a Web browser. It will also be helpful to have several Web browsers and mobile devices on hand to test your code. So that's it for the introduction. Let's now explore functions

Exploring Functions

Overview of Functions

[Autogenerated] Let's take a good hard look at functions in this module. We will first create a function and then look at several ways to call a function. Next will pass values into the function and then return values from the function. Finally, we will call a function from another function as we build a project that calculates the material needed for a round room. A function is simply a block of code that contains one or more statements that complete a task. The statements inside a function are not triggered until the function is called. To call this function. I used test with a left and right parentheses. E. The result would be a pop up on my browser screen. You can also pass values to a function in this case. When we call the test function, we will include the name of Jane. Jane gets put into a basket and carried to the function where it is transferred to a new basket called Use her name. When we perform the alert, we pull the name Jane out of the user named Basket and show it on the screen. You can also return values from a function back to a variable. In this case, we're going to create a function that calculates the mother's age as 20 years more than our sub. Down here we create a variable called mother Age and assign it to the name of the function. We put this son's age between the parentheses. When we call the test function, we put it 12 into the basket. The 12 is then put into a variable called son age. We use a local variable axe and at 20 then return the value of X, which is put into the mother's age variable. Once the program advances to the next line of code, an alert is called and we get a pop up showing the mother's age as 32. Alright, it's time to get out of theory and start coating.

Setup

[Autogenerated] Now we're ready to jump into the actual coating. I've got sublime text I'll be using for this particular project, and I have a copy of this start Js doc Zip file, which I will expand and open. It's got a CSS that is completely empty as a JavaScript file that is completely empty and then an index file. It's almost completely empty. So let's take that folder and let's rename it we'll call it practice. One will drag it to our sublime text sidebar, and now we can open it and see there's the index file jeans this, but we're ready to go.

Basic Function

[Autogenerated] then we have our project. I'll set up open in a browser and openers text editor. Let's go ahead and put an alert inside. The alert will put a very simple message. When we refresh the page, that alert automatically pops up because it's in line and the JavaScript is executed from top to bottom. If we know wrap this inside of a function and we'll just call the function test, put the closing function bracket down here, have that in to make it look nice. Save it now when I refresh it, the alert is not triggered because it's hidden inside of a function and there's nothing calling the function. So let's look at two different ways to trigger the content of this function. First of all, we can just go down later in her JavaScript, but test save it now when we run it Lane six calls Lane, too. Like to then traders line three, and that gives us this pop up. It's comment that out. Another way to do it is from the HTML, and we're going to create a very simple button test the function right now. It does nothing except sit there, look pretty so inside the opening tag, we will put on click equals in the name of the function test. Save it now when we refresh it test it now calls that function, and there's our message. So there's two different ways to call a function one from the civil button and one from a line of code inside of our JavaScript itself. So online. Three we've used in alert to get a message out of a function and to the user so that I can see it. Another way to do that is actually to do a console. Don't log, and here we can put a similar message. Save it. They were currently triggering this from a button Inter html. If we hit, refresh and then test the function, nothing appears to have happened in order to see the results of a consul log we're going to inspect. I'm using chrome and click on the console, and now we can see our message popping up in the console. Using council belt log is a great way to track your variables so that you can see what's happening in your program as you're moving from line to line

Function with One Parameter

[Autogenerated] Now let's move on to some more advanced function technique. So going back to my start Js file, which you can download from Pearl site, let's rename this one practice too. And we will open practice too. Inside of our text editor. We're still going to be using sublime text. Drag it over. Open it up, insider Js We're ready to go to work. Let's build a new function. We'll call it floor. What does it take to calculate how much carpet you would need for a round room? Well, first of all, you have to have the radius, you take the radius and you multiply it by itself and then you multiply it by pi, which is a math function built in the Java script, which is very convenient for us. So we've got a variable and we call it Radius. It's Quinn, the equal four, and you will create a new variable called floor area. And what that's going to equal is \_\_\_\_. Don't pie. Why is that 3.14 number that you learned about in junior high? We're then going to multiply it by the radius squared. Now there is a math function for doing radius squared, but we're just gonna keep it really simple and do radius times radius. It's just much easier now that we have that. Let's do a council, don't log and we'll take the floor area and will see what it iss. It's good. And call it Let's go back to our index. Change this to practice, too. Double click practice. Two. There it is it refresh. Once again, we gotta pull up the console and switch, and there it is. 50.26 to 4 is the area of a radius of four, which means it's four from the center to the side. Well, who on earth can walk into a perfectly round room? Know exactly where the center is in measure from the center to the side. That's not easy. It's much easier to get. A diameter with a diameter is the area from side to side, So let's do variable diameter equals eight, and the radius then would equal the diameter divided by to try that out. Still get the same number. Let's try changing this to 10 and sure enough, our number updates to 78. Well, this is useful if all of our rooms in the whole wide world had a diameter of 10 but they don't. So now we need to pass a value in to this function so that it can process and give us the correct floor area for any sized room. So this is the neat thing about it. We're just come down here into floor and we're gonna feed it. A number in this case will put the eight that eight is going to be put in a little basket. It will be sent up here and we will call it diameter. So it moves from the basket, gets put in diameter. We no longer need this because Diameter is already assigned a value from this eight. And now let's save it it refresh. And we're back to 50 for a floor of eight. So we just change this to 10 back to 78 so we can put any number in here, and it automatically returns the area of that floor

Function with Two Parameters

[Autogenerated] now that we have our floor function performing well, it's at another function to calculate the area for the walls, which is similar but slightly more complex. Function walls. In order to calculate the area of a round room wall, we need to have this circumference, which is the distance are round, and we need to have the height of the walls. So let's take these two step by step. Variable diameter equals 10 height equals 88 ft walls. Now we can calculate the radius, which we've already done up here. So the radius is that diameter divided by two. Great. Now the Formula Four circumference is actually two times pi times the radius. So let's do this. Variable circumference equals two times math that pi times the radius. Check our work and save it. No, we have to trigger it down here so we'll do walls. Semicolon, Save it. Refresh. We now have a value for the area of the floor and we have a circumference. Remember the sir coverage is the distance around a circle. In order to get the area, we'll do another variable called wall area, and it equals that circumference times the height now with show. The wall area has their second variable get refresh. We now have a wall area of 90. In order to make this function usable, we need to pass these two values in because not all rooms in the world have a diameter of 10 and a haIf of eight. So let's modify this just a little bit. Let's come up here and let's move diameter into there. Karma, and we'll put height as these second variable. So now we're gonna be passing two baskets, a diameter basket and a height basket separated by a common so down here diameter basket. Let's do it eight diameter and let's do 10 ft walls this time and let's change our floor so it also has a diameter of eight. Therefore, this is the same room. Save it. Hit. Refresh. We have 50 square feet for the floor and 251 square feet for the walls, so this represents carpet that could represent paint

Functions Calling Functions and Returning Variables

[Autogenerated] so we're making progress. But we want to introduce another step, and that is a function that could be called that will actually call both of these functions at the same time. So we don't have to have a double call down here. Let's make a new function and we'll call it materials. I need it. You know, this is something I like to do when I start getting many lines of code. End of function. So I know what they act. That particular curly is I'll just clear my code a little bit and we're ready to go. Let's come in here because we're doing a single room. The diameter is the diameter, the same diameter used for the floor function as the wall function. So it's true a new very well called D for diameter, and it's going to equal eight. Let's do a new variable, and we'll call it H for height, and it's going to equal Tim. Now let's do another variable carpet needed equals Carpet is going to call the floor function floor. What are we gonna send it? The floor has to have a diameter. The diameter is in the variable d. So between these. We simply put a D, which stands for the diameter, which has a value of eight. We need a new variable, paint needed equals and this is walls. The walls has to have two parameters. The first one is that diameter, which is D common. The second parameter needs its height, which we can see online. 18 is H. Now let's do Consul that lug and we'll show the carpet needed and we will show the paint needed as well. The last thing we need to do is call this function. We already know how to call it from the script. We just put this in here and it works. So we called our function. We can see from Line six that we have 50 point someone square feet of floor area. We can see from Line 13 that we have 251 square feet of wall area. We can see from line 23 24 that the variable carpet needed and paint needed don't have a value. So here's what happened. We called floor from 1 20 It went up to floor. It did its little magic and it was done. Same thing with paint. What we need to dio online. Six. Instead of printing the floor area to the consul, we need to return it back to the variable so carpet needed will have a value. So let's switch this to return and switch this to return and they're not inside of parentheses, so we'll calculate it. Will return. Floor Area will calculate the wall will return the wall area. Save it. Refresh. And sure enough, there's or 50 are 102 These two numbers are now coming from, like 23 24 not from line six in length. 13. An important distinction. Yeah, one more thing. I want to dio. Let's take this line and get rid of it. It's come back to our index. Let's create a button. You don't know how to do this. Calculate materials on Click and we'll pace that function in there. Save it. Refresh calculate materials. And there it is. In my luck

HTML Inputs and their Values

[Autogenerated] well. This particular application functions beautifully for calculating the diameter of a room that's eight and the height of the walls that are 10. It's not going to be very useful for customers who have different sized rooms. So let's jump back to our index. Let's add a couple of inputs type equals number because we want numbers in these now. In order to access them from Java script, they have to have an I D. So this is going to be the distance across the room. Next thing we need is the height of the walls. We'll make another input type equals number, worry about the name instead. Given I d equals no, because we don't have any CSS going yet. I'll just put a brake tagging here and in here. This should be done with CSS, but we're not focused on that right now. It refresh. Now we have the ability to enter a number and enter a height, but it's still going to default to the values hard coated in our JavaScript. So now we have to do is figure out. How do we get the values from this input and from this input into D and h. So that's our next task. Well, here's what it looks like. Document that gets element by I. D. And that's the I D that we typed in earlier. So this one is the distance across, so that's going to give us the idea of the input. But we don't want that. We want the value that the customer typed into that. So we're gonna do dot value because the value is a child on the input. Let's try the same thing for the haIf by simply switching this to height so across and height across in haIf. Let's test it and see if it's working. When I refresh the page and I could calculate, I get a 00 which is a good sign, but not the best. So let's actually put a value in here. So the Walter Wallace eight I through the walls is 10 and there's my two numbers, so now I can dio a distance across of 20 ft. Well, height of 8 ft and there's my two numbers

Cleanup the Output

[Autogenerated] So as we use our fancy new area calculator, we've got some values down here that are very useful. Imagine walking into a store and asking for 314.15926 square feet of carpet. They'd probably laugh at you, so we need to round these. We can't round them down, though, because then you'd be 0.1. This would be this much short, so we're going to round them up. So let's come down here to the variable carpet needed, and we'll do math dot c e i l. And we'll wrap that function. Call inside of parentheses through the same thing for the walls. Save it. Remember, we had 3 14 and 1 30 Get refresh. So now it's rounded each of those numbers up. That's a nice round number, but it still doesn't tell us much. Let's come down to the consul logline 25 26. Let's put in some useful information we're going to can can eight a string with the variable, which you hopefully done before. This shouldn't be too new, so the plus allows us to look variables to string. So here we have a string carpet needed it is, and then who put space? S q f t. Same thing down here. Single quote. Single quote plus the variable plus two more single quotes. Space as q F T and this is paint needed. Yeah, that we've tweaked or tax a little bit. Let's refresh one more time. We'll do a 10 and a six carpet needed 79 square feet. Paint needed 68 square feet while calculating the paint and carpet needed for a round room is marginally interesting. What is important is what we've accomplished. We've built a function that takes one variable end and returns of variable. We built another function that takes two variables in and returns the value. We've made another function, which calls these two functions. We've also extracted information from the HTML and put it into a JavaScript variable. We've used the math functions to manipulate our variables a little bit, and we've used the Consul Doubt log to spit out the data so that it's available to the user. You know, the next march will look at another way to get information out to the HTML document itself, so that's it for functions. Now let's take a look at loops

Exploring Loops

Overview of Loops

[Autogenerated] Hello, This is Paul Cheney. Let's take a good hard look at loops. We will start with the structure of a while loop and then jumped to a four loop. Next, we'll use a loop to extract data from elements inside the HTML. Finally, we will build a reusable function with parameters toe loop through multiple HTML elements.

While Loops

[Autogenerated] Let's take a look at a loop that counts to 100. In this case, we're going to use a while loop. Here's the basic structure of of Wild Do Notice that it has four parts where just start in this case, the very well. I miss it toe one when to end because we're using the less than or equal to, it will finish at 100. How quickly to go. This tells the loop to count by once. Once the value of I is 101 the loop will end. The last part is what to do while it's running. In this case, we're printing the numbers to the JavaScript console. Let's take a look at this in practice with some alterations. So here's the while loop. We just looked up. We're setting the variable I at the first. Here's our condition to end, and here's our increment with what's gonna happen. If we refresh this page, we can see that it counts all the way to 100 then stops. Great. If you remember back in, say, 4th, 5th grade, you learned about skip counting. Well, the purpose of skip counting is to help you learn math where we can make this while loop skip count for us instead of adding one. Let's try adding five will save it, refresh the page and now you can see that increments by five. But that's probably not what your teacher had you do. And fourth grade, you're supposed to count 5, 10 15 20 So we can modify this by setting the variable I 20 And that way, when you had five, you get five. There we go. 5, 10 15 20 Now it currently starts at zero. If you didn't want it to start at zero, we could actually switch the order of these two instructions. We would then first add five and then print the value of I to the log. Let's refresh it. And now we get 5 10 15 Problem is, now it actually goes over 100 before the loop quits. So another way to manage that is to put this back and to start it at five. And now we get 5 10 all the way to 100 which is probably what your teacher had in mind when she asked you to skip count. Now there's another thing you can do. You've heard of the old adage. Two steps forward, one step back. What? We can do the same thing here with a well, Lou. We'll start this back at one. We'll start by taking two steps forward, so I equals eye plus two. Then we'll print out the console log and then we'll take away one. This is the one step backward animal. Print I one more time. So now we should have two steps forward, one step back and then we repeat the while loop. You've got three back 2 to 4 back to 35 back before, so it's working for us. In this case, it goes all the way to 102 back to one a one before it comes back to the wall. Loop realizes that I is indeed above 100 and then the wallet quits and it moves on. In this case, there's no more JavaScript code to execute, so it ends

For Loops

[Autogenerated] Now let's take a look at another loop that counts to 100. Here's the basic structure of a for loop notice that it's more compact. However, it still has the same for part. Where to start This is run before the loop begins, when to end. How fast to go This is run after the code block has been executed. In this case, we're using a shorthand version of I equals I Plus one. The last part is what to do while the loop is running. Let's take a look at this one in practice. Here's the code block for the four lip that we just saw in Power Point, and this is what it looks like when it runs. It simply counts 1 200 just like the previous one. Let's do some modifications on this. Let's try reversing this so account backward. So instead of starting, I at one will start it at 100. Instead of ending it 100 we will end it at one, so it will run well. I is greater than one, and then it will stop running. So if I said 100 and we wanted to go backwards, then we need to do. I equals I minus one or we can do I minus minus. Save it. Let's refresh. We now get 100 counting all the way down to two. If we wanted it to go all the way to one would have to put it greater than or equals. And then we could get it to go all the way to one once again to count by fives. You do. I equals I minus five. That would count in reverse, starting at 100 going all the way down to five.

Accessing DOM Data

[Autogenerated] So I've modified my index that hte email and I've added a select called state, and I'm just giving it for values to start with. As long as the select has an I d, we can access the Children of it, we can access the value and we can access the text of any one of these. Let's go ahead and jump back toward JavaScript and let's start right running some tests. First of all, we're gonna create a variable, and we're gonna call it Element I d. And we're gonna set it equal to the document. Don't get element by i d. And the idea we're looking for is the I. D of state. And of course, we need it's semicolon Now to make sure on the right track let's do Consul, don't log And let's display what we have found in Element I d save it, run it And we can see over here on the right that we have a select with an idea of state. If we open it, we can see all of the Children. That item so far so good. Another thing. Weaken dio Consulate out log is we can ask for the length of this element I d. Now the element I d is an array and a race have the length or the number of Children in this case, therefore Children Wyoming, Utah, Idaho, Montana. So if we ask for that length, we should get the number four showing up in the console. And sure enough, there's a four because their four Children, if we were to go back here and where to add another state in this case will add Colorado, save it and refresh it. The length is now jumped to five, and we can see Number five interlace. We now have one of the piece of information that will need for Loop. Let's take a look at a couple of other things console dot log elements now because it's an array, which simply means it's a group of items. We use square brackets and we'll ask for the text of one of the items. So let's ask for the text of item number one. Save it, refresh and we can see it displays Utah. If we look over here, Utah's actually item number two, not item number one. That's because a raise are zero based. To further demonstrate that Let's ask for item five, which should be the last one which should be Colorado and will refresh it. And we get an air because there is no five in the array. What we've actually got here 01234 So, four, I would give us the value, the last item and zero. I would give us the value of the first item. So here's something we can do inside of our loop. In addition to asking for the text, we can also ask for the value. So in this case, Wyoming, the state abbreviation is W why Number two number to save. Refresh. We now have Idaho, and I d. Now we're ready to build a four loop for open qwerty close. Curly, what are the pieces we need inside the parentheses? Well, we need a starting point, so we're gonna use the variable pie and set it equal to zero. Because, remember, arrays are zero indexed semicolon. Now, where do we end? Well, I is less than remember appear. We know the length of the array, which is the end point. Now, the reason we use less than and not less than or equal to, is because the length of this is actually five. But if we zero index it it 01234 and that's where it stops. So I is less than the length would stop it for semicolon. And now how fast we move up. Well, we want every single Adam in the lists or do I? Plus plus. Now let's take these two and we will display both the text and the value of each element. I simply changing this to an eye, which is a changing value inside the for loop that saved that, that let's test our work. And there we have Wyoming as the text and the value as thes short postal code for each one of the items in our list.

Placing Loops in a Function

[Autogenerated] Now we're going to advance to the next step by adding a second select to our page. So now we have a state dropped down and we have a homestyle dropped down. We currently have a single four loop. That access is all of the items within the state. One. If we came up here and change this element by I d to home and ran our code, you can see that we get the same results. Now we're accessing homestyle instead of state. Well, let's turn this into a function that we can use over and over. So it come down here and we'll create a new function and we'll call it shows Duff. Well, what pieces up here need to go into the function where we need to know what the element ideas So put that first, once we know the element, I d. Then we can go ahead and we can run our loop so we'll put that next and loop and function. Gonna keep this stuff straight so that we don't all lose our minds. And we may even want to end end stuff to keep it nice and clean. That's grand. Strip out this first batch of code we no longer need will save it, run it and nothing happened. So I need to call the function in order to see any results. Show stuff, save it, refresh. And there's our single family. Now, instead of hard coding this element I d in here, let's actually pass it as a parameter. So it's creative variable up here called Element, and we'll use that element here. We'll just use short yell for it. Now we're going to pass in a string. Let's start with home fresh and there's home no change pass in state. Save, refresh. And there's our state. Well, now we can do this copy, paste home, Save it. And now we can use that function, which is a flexible function, because it's being passed a parameter to access either one of our selects here from our homepage. Let's now take this one step further by putting both of these inside their own function so that we can then call it with a single button here in a second. So make a new function and we will call it evil you eight page. And inside this function we will run the function to call this state and the home which runs this stuff over here. And once we clean it up, it looks like this. You know, if we run our page, we should have nothing happening because nothing is calling the evaluate page function. Therefore, nothing is calling the show stuff function, so everything is dead in the water. Let's come back to our index. Let's make a new button. We'll call this button rent script and when we click this button, we wanted to run this Evaluate page script so we'll put that right there. Save it fresh. There's our rent script, but click. And here's the evaluation of both the state and the home style. Now that we have our JavaScript running under the control of a button, we actually have the ability to make selections here and then run the script so we should be able to come back to our script and ask which one is selected? Well, it's going to be very, very similar to what we've already done, except that the word value now becomes the word selected. Let's refresh the page. Let's choose Utah and multi family and run the script. We can see that Utah now has a value of true and multi family now has a value of true. So there's yet another thing that we can check on when we loop through all the values off these elements.

Finding the Selected Item

[Autogenerated] so far, all of the output from our function is being displayed in the JavaScript console. Let's no change that's were actually putting it out here on the page may be at the bottom. So it's going back. Torrey Index and we will add a break tag here just to make things look a little bit nicer. Then we'll add a division tack this division tags not going to have anything in it. But we are going to give it an I D of output so that we can put stuff into it. Now let's come back here toward JavaScript and let's run a test document dot gets element by i. D. And the element we're looking for has the name of output, and we're going to set its inner HTML equal. Teoh hi and semicolon. Let's save out, Run it And there's the word high insider empty division. Now that we know how to access it, we can change our function. If element my i d equals true, then we're gonna do some stuff that's couldn't put this instead of saying hi will say, found me, give it to this one. Save it. That's good refresh and runner script so it did say Found me So this loop is being triggered, but found me doesn't say anything about what was found. So instead of that less than helpful message, let's put something more useful in here. When the value of any item is selected, then we can access the value and the text. So let's inside this loop se variable and let's just use X equals the element I D. And we will get its text. Once the text has been put into the variable of X, we'll change. This found me to the value of X, Save it, run it, run the script and we have single family. If we hit multi family that now says multi family. Well, what about this one up here? Because it's actually being checked to what's happening. It's actually putting Wyoming, but then it is so quickly erasing it and replacing it with multi family that we don't actually see the difference. So let's make some modifications now to our script. Our function show stuff ends right here, so let's Instead of displaying stuff here, let's actually return the value of X so we can come down here inside of our second function and we can create a variable and we'll call it feedback. And everyone has said it equal to show stuff for the state. And then we're going to take feedback that's couldn't remove. This appeared because it's not really serving us well. Instead of putting X inside, the division will put feedback. I will say that and Renate and now we get Wyoming showing up. Switch it. We get you test showing up. So the first one is working the second one. This is still not quite there, but we're making progress. So let's do this. Let's do variable feedback state, and then we'll do another one, which is just like a Except it will be feedback for the home and we will run the function home. So when we put stuff in this division, we can put Feedback State plus feedback home. Make this a little bit whiter. Save it. Let's try that and see what we've got. A runner script. Leah, Wyoming And then we get single family. We could do Idaho Mobile Home, Idaho Mobile help. That's okay. Let's put a break tag in here. And then, of course, we need another plus. Multi family Idaho we now have Idaho and multi family. So now we're able to track with Java script what the user is selecting. Put it into a very well store that variable and then at will either displayed on the screen or display it in the consul log Before we leave. Let's clean up our code a little bit. We, first of all, don't need all of these consul lock statements now that everything is working so we can remove that these extra spaces here can go away. We could add a comment here that says End of the if so we know what that is, this line. So everything now is tightened up, and of course, it runs beautifully.

Finding the Checked Item

[Autogenerated] so we know how to use a loop to access the selected item. What about a input type of check box? Here? I've added a lighting heading and I have three check boxes, one for incandescent, one fairly DIY and one for CFL on our Web browser. It looks like this These air checkable, multi checkable or single so you can select one some or all we want to do is when we run the script. In addition to knowing which one of these was selected, we want to know which one of the lighting types was selected because we're no longer using a select with multiple options, but instead multiple inputs who have the same name in this case bulb. We're gonna have to create a different function. Let's come back to our JavaScript and let's start from the basics. One more time was created variable, and we'll call it element I d again and we're going to set it equal to document dot gets elements, noticed the s by name, semicolon. And then let's do a console dot luck and we'll drop the element I d inside there and see if we get anything out of it. Appear we used an element by I D because it was a single idea with multiple Children here. We don't have that. We have three different inputs, but they all have the same name. So the name Bold is going to go into here, say that Let's see what we get here in our output when there are three inputs showing up. So let's add a dot length, just like we did in the above function. And sure enough, we get three because there's three incandescent led and CFL, so that's going to be the same. So how do we tell now which ones are checked? Let's do a console dot log and once again will do element. I d remember, because it's an array. We use square brackets and we're gonna ask for the checked status Appear. It was the selected status down here. It's the check status. Well, it needs to know 0 to 2. So let's start with zero to see if that one is checked. Refresh it and it is a false If I click it and refresh it, it's still a false because it's reloading the page, and in the process of reloading the page, it is un checking my selection. So at this point, we're kind of stymied until we create this as a function. So let's wrap this whole thing inside of a new function, and we'll call it Show More open, Curly and come all the way down here for close Curly and then a comment that says, End of function that takes all of this code and makes it so. It will not run until we activate it some way. So down here inside this function, let's call it There's our show. More function. Save it, get refresh Nothing. It's great and click incandescent Run it. And sure enough, we get a true If we uncheck it, we get a false So this seems to be working. So now how do we put this inside of a loop? Because that's the whole point is to say how many of them are checked not just to check one random with four open Kerney and and, uh, loop. So one of the things that go inside the parentheses well, we need to start I at zero, which is the beginning of the loop semi colon. I is less than and here we've got element I d. That length already specified for us. It was dropped that in with a semi colon. And then, of course, you want to add one to it, so check each one in the loop. Let's take this, cut it, put it inside of the loop and change the zero to an eye. So it's incriminated. Now we'll check each item and evaluated to tour falls. Let's refresh the page. We'll do a true false true boom. True, false. True. So that was tracking it in the console. We simply need to get it out of here and over here to this. Now this is going to get a little bit messy, but hang in there with me and we'll see if we can figure it out. We have the ability to check the checked state as tour faults as we loop through each of the items. In addition to the Czech statement to riff faults, we can also check the value of each one just like we did before. So we run the script. We have the name or the value of each and whether it's checked or not. Now, really, all we want down here is the ones that have a checked equals. True. So we need to add inside of this loop and if statement yes, open close and if so, what are we checking for? We're checking to see if the value of this is equal to true, because remember, we have false is and true's appearing over here. Well, if it is true, we're creating new variable called my check and said it equal to the value which we already know how to get right there. Now this my check needs to be set up initially above the four loop. So we'll come up here and we'll say, Variable, my check is going to equal and will make sure it's empty. So there's nothing in it. And then, if something is checked, will put the value inside there. When the if is through, then we're going to console. Don't log my check and we'll see what's in their refresh. We're in the script, and we have an awful lot of stuff over here, so let's clean up a little bit. Let's get rid of these, so just comment them out. Comment this out now are Log should only contain the results of Line 23 We're going to refresh the page and click incandescent and run the script. We don't have incandescent appearing three times. It's clear it. Let's check incandescent and led. Run it. We have incandescent and led, appearing to times. So this is not exactly what we're after because we put the consul log inside the for loop. It's spitting out this value three different times, and it's always speeding out, whether it's checked or not. So we actually had to move it down. So it's outside the four loop, so it only presents its results once. So let's try that. Get refresh. We cleared it. Greek incandescent notice it prints at once. If we click two of them, I'm now getting just the l e d. Let's check that for the script. Sure enough, it's just the led. So what's happening in Line 21 is the value of my check is being replaced and not added to. So in order to get the list off all the items that are checked, we need to set my check equal to itself. My check, plus the new element. So now let's run it with incandescent and led, and now notice that they're together if we had all three of them. I'm now getting Incandescent led and CFL not very readable. So let's add a plus and then a break tag. This will make it a little bit more ledge a bill, because now the results will be on separate lines. Well, quick, just light. There's led with the break. Click on three and we're getting them like that. Instead of logging this to the console, let's stick it out over here. So we need to come down here and tweet this a little bit, create a new variable called Feed back for the light and said it equal to the results of Show more in order for that to work, then we need to return my check. So when it comes back here, it gets put into feedback light. And, of course, then we have to add that over here, plus feedback light. Save it back to our browser, get refresh. It runs script. There's nothing there because nothing's checked. We'll check out three of them, and sure enough, there's incandescent led and CFL noticed that incandescent and single family on the same line. That's because we need to add another break tag between these two, so we'll drop in b r Tex. Save it. Let's test it one more time and see how we're doing. Refresh, Select Run script. We've got Wyoming single family incandescent led. Let's make some changes. Eido Apartment, CFL boo, Idaho Apartment CFL What we're using Both Led and CFL. This script is now replacing the contents of this empty division and refreshing it every time I click the Run script button. Based on my selections here in the HTML in our current H c mo page, we only have one of these checked boxes in the page, but we can make our JavaScript more usable in case in the future. We decided to add an additional check box type. So let's come back here to our JavaScript down here, where we have I call on Line 35. Let's put the value of both. So now the value of bull was going to be sent to our function, will bring it into the parameter called E L, and will replace the word bulb with the value inside of E L, which is short for element. Now let's run it and see if we've broken anything back toward browser Refresh. Click a couple of these from the script, and sure enough, everything is working. So that's it for loops. Next, let's take a look at a switch statement.

Exploring Switch Statements

Overview

[Autogenerated] Hey, I'm glad you're still with me because we're going to explore this switch statement. We'll start with the structure of a switch statement using strings and then show you how you would switch on numbers. Finally, we will see a fall through switch statement. A switch statement is basically a bunch of, if then statements all glued together into a single code block. Let's take a look at the parts. A swift statement has a single variable that it checks each possible matches, then set up is a case with a matching value. If there isn't match, then the statements following the match or executed in this case, If my variable waas morning, then the concert would like the message time for breakfast. Once a match is found, there is no need to keep checking. So you break out of the statement and jump to the end. Just in case there is not a match found. You should provide a default option. In addition to switching on variables with a string, you can also switch on a number. If the variable my name is assigned to the number three, you would be assigned to the Bear team. If the variable. My num is assigned to the number one. Then you would see an air message because mine, um, did not match any of the cases above. A switch statement can also have multiple cases with similar results. This is called a fall through switch. In this statement, a number of one or three or five would generate a message that these air odd numbers a number of two or four or six would generate a message that these are even numbers. Let's try this out using the start JavaScript file from the demos provided with this module.

Switch with Strings

[Autogenerated] Let's begin by creating a script that converts a number two a day name. I'm going to rename the start Js today practice. Drag it into my editor, and we will start with the HTML change this to day converter. The first thing we're going to have to have is a select. We're going to have to have seven options, one that represents each day. We'll give it a value and Sunday will be a you. It'll also be day number one. Change these in addition to having a select with several options. We also need a button to activate our script instead of using the consul log, which we've done before. We're going to actually send the result back to our HTML page. So we're going to put an empty division down here and we'll give it an I D. Then we can open up our JavaScript file. We need to have a function, and we'll just call it my function. The first thing we're going to do in this function is find out what option has been selected. So it was created new variable and once again, in between these quotes, we need to get the I. D of the select Let's jump back to the index. We have a select let's given an I D equals Choose me. We'll copy that exact value back to our JavaScript and put it in there. So we have a connection within to a new variable. We'll call it a date short. It is going to be equal to the element. Don't options because it's an array. We will use square brackets and it will get the value of the selected option. So now we need to come back here. We've got our element, and we want to ask for the selected index. Let's make sure this is working. Before we go any further, I'll take the date short and we will display Tour Council. Now that we have our function in place, we'll use the function name, which we have right here online. To copy it, Return to Index and here online 23 where we have on click well paced the name of the function. It's no test it and see if that's going to work. It's opener index file will inspect switch to the console and click run script, and we have you appearing, which is Day one which is Sunday. So Saturday Day seven should be an s Sunday, Monday, Tuesday, Wednesday should be Day four. And sure enough, there's a W Now that we have the console displaying the date short, let's now connect it so that it shows up here in the feedback division. So let's come down here and start by putting document, not get element by i d. The element that we're after is called feedback, so I'm just copy that to make sure I don't get any typos. And we're going to set the inner HTML vehicle to, and we'll grab this variable and paste it with a semicolon. Let's say that Refresh our page, run the script and I'm getting a you and w appearing here on the page as well as in the Consul. Now we're ready to do our switch statement. Here's the code for the switch statement. Switch open. Curly, close, Curly. We're going to be switching on a variable right here is the variable called date short, and then we need to set up several cases. First case is going to be Sunday, which is a you and with a colon. And then what happens if date short is actually you. Well, day name equals Sunday. Let's come appear. We have a variable element. We have a very both date Short list wouldn't create a new variable. Call it day name. So it's defined. And down here we can just assign it. If indeed the cases you will sign that variable and we're done, there's no need to continue with any other statements below. It would just \_\_\_\_\_ out of it. Now will come down to like 21. Instead of showing the letter of the day, we're actually going to show the full name, So let's save it it Refresh runner script. Day one does indeed equal Sunday. Now we haven't got anything else set up, so I'll do that next. The entire code block for the switch statement is found in the demos for this particular unit, so you don't have to type it out by hand if you don't want to. I'm just going to pace that in notice that Sunday is exactly what we had it. We simply included Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and we've included a default value in this case. It says the day name is out of range Let's save that refresher page. Let's try. A different day will choose four and it's Wednesday. Seven is Saturday, so we've now successfully converted a string, in this case, a single letter of the alphabet to a full name of a day of the week.

Switch with Numbers

[Autogenerated] So let's continue with our existing code and simply switch the case statement from letters to numbers appear at the top. We are interrogating the drop down element and asking for the selective Index, and we're getting the value. Remember, the value is represented right here as a strength. But there's also the number one, the number to the number three. We can change this from value to text. And now let's see what the consul log is going to give us. Let's refresh it, run the script. The concept gives us a one. Give us a three, give us a seven. So we are getting 13 and seven. But our switch statement is broke, of course, because the values air no longer u m t. They're now 123 So let's just come over here and put a one and a two etcetera and save it. Refresh the page, run the script and we're still getting an out of range error. So it's still coming down to the default value, even though I can clearly see a one and a four in my console log. It's not working as a number. While here's why. This dot text is pulling the text as a string. So even though it looks like a number one and number four, it's actually a string with the one in it and a foreigner. So the variable that short is a strength, so we need to force it to be a number. So we need to convert this very boat that short from a string to a number. So we'll simply put in you and be our left. Print the sea right parenthesis e moved eight short inside of that, save it it refresh runner script. And now we're getting Sunday from a one number five gives me Thursday. Seven should give me Saturday now just because in my switch statement, I've got my case. 123 That doesn't mean that's how they have to be. I can move these cases anywhere in any order, and they're still going to work so I could start with three and then go toe one and then two. I could move six. Let's test it. One is still Sunday to a still Monday. Three is still Tuesday. While as possible to put these cases in any order, sometimes it makes better sense to keep them in order because your day of the weeks we're gonna make more sense if it starts with Sunday and ends with Saturday, that if you have some spaghetti code where things were just randomly thrown in here

Switch with Fall Through

[Autogenerated] Now let's make a switch that has some fall through. Let's go back to our index file. And instead of having days one through seven, I'm going to change this toe fortunes once your 10. Now the code for this select is provided in the demos, so I'm just going to copy it and paste it. We now have Fortune one through 10 with values matching one through 10. Let's go back toward JavaScript. File. Line three is fine the way it is. We're no longer getting in date. Short. We're getting a fortune number and we'll grab the value instead of day name. So be fortune and was clean this out. So the fortune number. We want to drop into the consul to make sure it's working, and then we want to switch on a number, not a strain off the fortune number. So it's saved that, and let's put in our first case. If the user chooses a want colon, then our fortune message is going to equal. You will inherit a fortune cynical, and we're done. So let's break. Let's test it to see if it's working. Run my script and we can see in the console that Line 19 has an air because day name has not been to find well, I changed it from data fortune. So let's take the fortune message and display in our div. Get Refresh. Try it again. I get a one and I get a You inherit a fortune as my message case. One is inheriting a fortune. Let's do Case three as well. So if it's a one or into three, they're going to inherit a fortune. That's refresh. Fortune one and three are both of fortune. Great. Let's copy this. Let's do two and four. Maybe they will win a new car. That would be nice to have. And of course, we need to break. Save it. Refresh. Let's try to. There's the new car. Let's try four. It's also the new car. So the fall through in this case is that the one or the three? Give me this feedback message. The tour. The four. Give me another feedback message. Once again, I've provided the entire switch select in the Devils for this unit. I'm going to pace the completed code. I still got fortune number cascading all the way down through. I'm saying the fortune message which is displaying down here online. 34. Let's save it. See if everything is matching. Refresh the page. Run the script. You'll hear the fortune. Try number four. Got a new computer number. Eight. Uh, it's also Computer 10. You have three hours to live to finish this course. So that's how a fall through switch statement works. So that's it for a switch statement. Now let's explore interrogating the Dom.

Interrogating the DOM

Overview

[Autogenerated] Hello and welcome back. We're getting really close to the solar calculator. However, before we get there, we need to review how to access elements in the DOM. We will start by getting and changing text using five different methods of interrogation. Then we'll use a combination of these five methods to refine our interrogation techniques. The first one will be get element by D. Noticed that the word element is singular. This tells us that it returns a single object. Second, we will get elements by tag name. Elements in this case is plural, so we'll get a collection of items that we access using array notation. Third, we'll use get element by name, which also returns a collection. Next, we'll use get elements by class Name again. Ah, potential collection of multiple items. Then we will use query selector all which is really awesome and also return to collection finally will combine these two really target our inquiry. Make sure you have the start of file from this modules Demos. It contains lots of HTML code that will be using Let's get started

Element by Id

[Autogenerated] here on my desktop. I have the start file Module five, which contains an index file a CSS folder in the JavaScript folder. I also have the demos dot txt, which is a file that contains lots of source code that you can copy and paste while we're moving through these modules. Let's take a look at in nine case sublime text here is that start module five that you see over here. To the right, you can see it contains a CSS folder and an almost entirely empty CSS file. It also contains a main dot Js, which is entirely empty. And then, of course, the index file. Let's take a look at the index found this time in chrome. I've got my JavaScript console open on the right and I can see the HTML document on the left. The female is a standard Web page with the four basic parts. The first part is the header which contains an H one h two. Then the second part is a knave which contains an a murdered list with a bunch of list items. We then have the name which contains several sections. Each of the H ones is correlated with one of the main menu items, and you can see the link. Here is a pound sign, which references this i D. So it's a single page website, which simply scrolls down to the relative spot. This first H one has a couple of age two's with paragraphs noticed that they have names. We use those later on. The second H one Exploring Functions has a list. The third one also has an ordered list, and the fourth one hasn't undergird list. The Fifth Section Document Object model has a form with three inputs, each with a name which will use later on, and a select with three options, which will also use later on the final piece of the Web pages. The Footer This case, it has a class called Small Center. Let's go back to remain javascript file unless start by targeting and element by i d. So we're gonna create a new variable, and we're just gonna call it, found you. We're gonna set it equal to the document dot Get element by I d notice. In this case, that element is singular. Now let's go back to our main file and let's pick a section care first sections called Overview. Down Here Online 28. There's an I D called overview, so let's target that one. I'll just put it here between the quotes and now let's see what we found. Console dot log found you and I've got to fix that typo and that type of we'll save it. Let's refresh and we can see over here the H one with an idea. An overview. That's what's found is the single element, and it's no being displayed in our lock. Well, now that we've found the very well, let's take a look at just the text that's inside of it so we can do consul dot log found you dot inner text and, of course, semicolon. So we'll save that. Refresh it and we can see that Jobs is the text that's inside of that. So here's overview and there's overview access using JavaScript. Well, great. Now that we have the text, let's change it. So found you dot inner text equals, and I'm just gonna make it really obvious here. I changed you with the semicolon at the end. Save it it refresh, and we can see that there's the change made inside of our HTML document using line six of JavaScript. Now, quick side note. We have inter taxed. We also have HTML, so we can change that as well. That's another way of accessing and changing stuff, and the results, as you can see here, are identical.

Element by Tag Name

[Autogenerated] the next we're going to try is accessing elements by tag name. Now the tags on this page. There's lots of we've H ones, each one's H ones. If a couple of age twos, we have some paragraphs and all of those could be accessed using by tag name. So let's jump back toward JavaScript. Start off with a variable. Once again, we'll use found U equals document dot Get elements within s by tag name in here we simply less the tag name we're looking for. Let's start by looking for all the age ones, and then we'll console that log and we'll put found you in there and see what it comes up with. So let's save out it. Refresh. And here we can see h one h one h one of all kinds of H one in a collection. Because it's a collection, we can use a ray notation to access them So we'll do. Consul got log, found you left right, and we could put a number in here. Any number that's less than the total number of the collection. So there's lots of h one, so let's just put a to in there, save up so there's the second one exploring functions. But we can also ask for the dot inner text of this second item, which actually the third item because it's number two and now we get exploring functions. Which mats is this exploring functions? And if we were to tag number three, of course, that would be exploring loops and exploring loops. Appear those match. We can also look for all the paragraphs it refresh, and we can see a collection of paragraphs. We have an error here because, as you can see, there's 123 paragraphs which would be numbered 01 and two. But we're asking for number three, which doesn't exist, So it's back this off toe one. Try it. And sure enough, we now have the entire content of paragraph number two, which you can see there and over here Let's go back and do one more thing. Found you a ray notation. Let's do this. Let's do this second item, which is array Element number one don inner text equals. I changed you save it, it refresh, and we can see the results here under abstract. That paragraph has been changed and replaced with the text from the JavaScript

Element by Name

[Autogenerated] Now let's try accessing elements by their name, so we'll start with the variable found. U equals documents, not get noticed the \_\_\_ on elements by name. And we'll fill that in here in just a second. So let's jump back to our index Here. We have a name associated with this paragraph online. 30. We also have this same name associated with a paragraph online. 33. We go down to the bottom 70 71 72 also have names in this case, their unique first last and phone. So we should be able to go toward JavaScript and put first in here and then be able to track the results in our console, not log. Save it. Refresh. We can see that there's an input that it has found and noticed. That isn't list a collection. It says No. Blessed here has one element in it because there's only one with the name of first. Now let's try description. Remember, that was the paragraphs, this paragraph and this paragraph refresh. We now have a note list with two items paragraph zero and paragraph one in order to access the content of those because it's a collection will use a ray notation and asked for the inner HTML or text. In this case, let's start with the second one. It refresh. And there's the text from one or the second element in the array court. Changing that to a zero would show us the first one. And there's that once again. Now that we found something we can change, it found you which wondering after, Well, let's do the first element and set the inner html equal. Several Colin and I forgot the dot Right there. There we go. Save it. Refresh. And here we can see the results of our JavaScript acting on the first element of the found you collection.

Element by Class Name

[Autogenerated] Now let's take a look at accessing something by its class name in her index page at the very bottom in the Footer, we have a paragraph whose classes small center. Let's go back to our main Java script page, and let's access that one using it. Variable found U equals document, not get elements plural by classic and the class name that were after its small center. So copy that. Paste it in here and then we'll do a consul. Don't log found. You save it. Refresh. And there it is. It's a paragraph whose classes small and center, because there's only one element will be access. The rain notation is gonna have to be zero, But then we can ask for its dot inner text or inter HTML, and we can see the footer matches the text that we found. Once again, we can change it by taking this and setting it equal to new here. Daph Info. Save it and there's the new footer paragraph at the bottom of the page.

Query Selector All

[Autogenerated] Now let's take a look at the query selector. All this one is my favorite variable found You equals document dot query selector All Now we can look for any selector we want. So let's start by looking for all of the list items on the page and we'll do a console. Don't log found you save it. Take a look at it. We can see there is a lot of list items. 123456 K So every single list item from overview functions and loops down to all of these down to all of these, including all of these. So those are all list items they're showing up over here because it's a collection. We can read an individual one so we'll do a ray notation here. Let's grab the first one asked for its dot inner text. Save it and we get overview from the first item in our list. We can also change by setting the value equal to some string save refresh. When we can see that first menu item, it's now been switched to just text instead of a link. Let's now move on to look at a little bit more specific use of the Cory selector all. It cannot only find individual list items, as we have here, but we could look for a UN ordered list with a specific I d. Mr Case. Let's go back to our index. I scroll up and let's target this list of items. There's 123456 items in our function list, and here is the I. D. So this jump and put I under list. Who has a i d of function underscore, lest let's save that, it refresh. There's all of the text in that element. Now this is a little bit confusing, so let's track the length of what's being discovered here, so that will help you understand when we move onto the next phase. Let's reveal the console dot log and let's get the length I've found you don't length. We'll make it a little bit more clear here by adding something in front of it. Lee of Collection, Save out, get refresh, and we can see the length of the collection is one because it has found the owner would list. But if we go in here and we had a space which means a child off and now asked for all the list items. We're now getting a length of collection of six because there's six items in this exploring functions 100 less and we've chosen to change element zero off the six, which you can see the results of right here. But if we wanted to change number five, we can see that's now moved down here to number five. What if we wanted change number two? There it is in the middle. Yeah, we've masked with exploring functions. Let's go down a little bit and simply change this to a different I d or now under loops. Change that. We're still gonna look for an under list. We're still gonna ask for the list items in that undergird list. And now it's moved down to here under exploring loose

Combination 1

[Autogenerated] Now let's use a combination to do something similar to what we just did with the quarry selector. All we'll do a variable found you and said it equal to document. Don't get element by Ivy. In this case, we're going to get the S w underscore list. And of course you want to track it so that we can see what we've got. And there it is. 100 list we can see here. It has three different items. Now. We're going to get all of the Children of this, found you element and put him into a collection so set up a new variable because this next one is going to be a collection. I wouldn't call it Child Array. I'm going to set it equal to I found you don't get elements by tag name and we're going to look forward. The list items inside off this a Norden list which we can see right here is the result off the I d. Of course you want to track that. Let's hit refresh and we can see that the second consul log is a list of three items in a collection. Now that we have the child array. Let's go ahead and change the inter text of all three of those elements. Child array element zero dot inner text equals these items. Don't forget to semi Colon. Copy that. Do it two more times one and it to have just Ben changed. Save it. Refresh and there's the results and her switch statement off our inner text statements.

Combination 2

[Autogenerated] down at the bottom under document object model. We have three inputs underneath that we have a drop down, which cruelly says Honda Chevy four. We're going to use JavaScript to change the options of these two colors. So let's go back to our JavaScript and we'll start with setting up an array of colors because we're going to be using a loop. Let's start with the variable called new options. It will set it equal to in a rate collection quote, comma, quote. Call a quote and we'll put three colors in here. Next, we will grab the element by idea of the form. So said a new very well called found you and said it equal to document elements singular by i d. Let's go grab the I d from our html, my form. Save it, Consul, that log found you. Let's see what that gives us so far. Sure enough, we have a form. We can see that that form has 123 inputs and a select. In our case, we want to target the select, so we'll break this down further by doing another variable. And this is going to be a collection, so we'll call it the Select Array will set it equal to found you dot get elements plural by tag name The tag name were after is option. We can see right over here there's are three options that we're trying to get. Let's track those in the Consul, and sure enough, there's are three options length of three. Now that we have those in a collection, let's go and use a loop this time. Four. The three parts of a loop are the starting point. I equals zero. Then the ending point. I is less than select a raid dot length semicolon and then how quickly to advance where we want advanced one at a time. Let's start with the first element. Select a ray and inside the brackets we're going to put this I, which is changing value, starting at zero and then one and then two, and we want to set the inner text equal to and this is not what we're gonna end. Let's just put yet worked for now. Does that refresh and you can see I have three yet worked down here in my dropped out. We don't want that. We wanted to be orange and then red and then blue. So we're going to access this array using this same value of I. So be new options and it's an array and we'll start with zero. And then Goto wanted and go toe to see if that worked. Got orange, red and blue coming from our new options orange, red and blue.

Combination 3

[Autogenerated] Our third combination is going to allow us to access the values that the user types inside of these three inputs. Normally, those values would be entered by the user, and when the pages loaded they would be empty. But for the purpose of speed, I've pre filled those in my HTML by providing a value equals for each one of these three inputs. Let's go back toward JavaScript and let's start with the variable found. You said equal to document. Don't get element by i D. In this case, we're looking for my form. Let's track our progress and see what we've got. So we have a form showing up here once again. First last phone and the select. We just want the three inputs. So let's further refine it by saying a new variable, and we'll call it Select Array because it's a collection, it will set it equal to found. You don't get elements plural by tag name the tag name nor after is called input because we want these three inputs right here. Save that it refresh. And here we can see the results of this select a \_\_\_\_. Once again. Let's do a loop. We'll put in where to start, so I equals zero semicolon. Where to end I is less than and how quickly to move forward. Well, now do a console Not long. Programmers select array and will increment through each element using I, and we'll ask for the dot value. Where does value come from? Well, if I go back here, I can see that each input as a value and will be looking for Tim Oswald and a phone number. There we go, Tim aswell and a phone number be extracted from each one of these inputs and displayed to our consul Luck. So that's it for interrogating the dumb. Let's take everything we've learned so far and build a solar calculator.

Solar Calculator

Overview

[Autogenerated] congratulations on making it this far. It's time to put everything we've learned together to build the solar calculator. We'll start by looking at the flow of the JavaScript program, and then we'll actually build a calculator. There are three things we need to get from the homeowner. First is how much electricity that home uses. On average, this is calculated in kilowatts per hour. Second is how much sunlight they typically receive per day. Finally, the homeowner needs to choose a solar panel. Each pattern generates different amounts of energy solar panels, air rated by how maney watch they generate per hour, which is different than the kilowatt rating used by the home. So let's take a look at the program float. First, we gather the last 12 months of power usage from the form inputs and add them together. Then we divide by 365 days. This gives us the average kilowatts used per day. Next we find out what zone the homeowner lives in and get the number of hours of sunshine per day. Then we divide the homes daily need by the number of sunshine hour to see what we need to generate per hour. Then we have to adjust for bad weather that blocks the sun, so we increase the need by 25%. Then we multiply the kilowatts needed by 1000 to get watt hours. Remember, this is how panels air rated. Then we find out which panel the homeowner would like to use. This tells us how many watts that panel can generate in a single hour of sun. Finally, we divide to see how many panels air needed to offset 100% of the electrical needs for the home. Let's see this in code.

Calculate Home Usage

[Autogenerated] make sure that you have the solar start folder from the demos for this particular unit. I'm going to open up in sublime text. I've got my index and my JavaScript file open. As you can see, the JavaScript is completely empty. At this point, we're gonna start by calculating the total annual usage for this particular home. You can see her in the index file that there is an input for January February March all the way through December. I also have assigned a value that's a testing thing. Onley. Once I'm done building this form, I'm going to take those values out so that when the user sees it, each of these is empty. But for now, I don't want to enter 12 values every single time we test this page, that would drive me absolutely insane. So I have provided some defaults online. 30. You can see that we have a field set. It has an I D. So the first thing we're going to do is get element by I d. Once we have that, then we're going to access all of the inputs as an array. So there's 12 of those. So then we use get element by tag name to put those in the list. Inner JavaScript. We're going to start by creating the four variables that we're going to need. We have an annual kilowatt usage which will then be converted to a daily kilowatt usage. We're also going to have very well I and a variable X which we use in a minute. As I said earlier, we're going to do get element by i d and then get the tags within that The element name was monthly power consumption, or MPC. And the tag that we're looking for is input Ellis. Track it and see if we've got it correct. After saving it, we will hit, refresh, and we can see an HTML collection here of the 12 different elements in that array. So so far, so good. So here's the basic structure of our loop. We're going to start I off a zero. We're going to go to the length of the months which we saw over here was 12 and then we're gonna increment one of the times we get every single month. Now we're gonna use that variable X. We're going to set it equal two months because it's an array. We use a square record I don't value in this case months I dot values going to return a string we need to know, convert it to a number so we'll start with number rapid in parentheses and with a semicolon. Now that it's a number we can use our next variable, which is our annual kilowatt usage. It will do plus equals X. That's the same as annual use. Kilowatts equals, and you will use kilowatts plus X. When we're all done, let's do a console and we'll track the value off our kilowatts. Let's save it, refresh it and we can see that 9631 is the sum of all these months. So now that we have our annual kilowatt usage, we can convert that to daily kilowatt usage and appear. Is that variable? So I'll copy it and paste it, and I simply said it equal to the annual usage divided by 365 days per year. And now let's display the daily usage to the consul to make sure that's working correctly. So let's refresh, and there it is. 26.38 is our daily kilowatt usage for this particular set of values. Millet that is working. Let's wrap it inside of a function and we'll call the function add months. So it describes what's happening open currently, and we'll close it down here. If you run it now, nothing's there because the function is not being called. So let's come down here. Weighs on. Let's call the function, Save it, refresh it and now it's being called. Well, here we have a hard coded value. Forget element by I D. But instead of using that, we're going to pass the value from the function call. So let's take the name of this I d. Here. I'll remove it, and I'll put it down here in the function call. That means in Line four, where we add months. We have to bring in the name of the element, and I'll just make up my own word called E L E M. Once this has a value, we can then use that value here in or get element by i D. And that should give us the same result. And there is once again 26 down here. Let's create another variable called daily use in kilowatts and we're gonna set it equal to the results of this ad months function call. While that means that instead of displaying the daily usage out to the console log, which doesn't go anywhere, we need to return that value. So now if the function is done, you'll return daily kilowatt usage and it was sign it to daily use kilowatts. And to test it, we need to now claim so luck. I'm gonna remove line seven. It was just there for testing purposes, and I'll re use it down here to display the daily use kilowatts. Let's save it. Refresh. And now we have just the results from line 27 which we can see here. So the clean this up a little bit, our first function toe add up, the months is complete, it returns the value David Kilowatt usage and we have that value now assigned to a variable which we can use later in our master calculations.

Calculate Number of Sun Hours

[Autogenerated] The next piece of the puzzle is to gather the amount of sunshine used from this Sunshine Zone graphic, and the user selects the zone that's appropriate to where they live. If I lived in the state of Florida, that would be Zone four here in her HTML. We're going to access the zone that is selected by the user. So let's take a look at Thea. Drill down. For this. We'll start with the document. The document contains a form the name of this form, and actually the only form is called solar form. And then if we scroll down, we can see that we have a select, which also has a name called Zone. And then we're going to ask for the selected index. In other words, which one of these is currently selected in our JavaScript? Let's come down and create a new variable, and we'll call it in this zone, and we'll set it equal to document that forms that solar form Done zone not selected index semicolon for part two. We're also going to need a variable, and we'll just call it H R s, which stands for the hours of sun. The zone is referencing a selected index, which, if you remember from a previous units, is an array and starts at zero. But if they choose own one, I don't want to return to zero. That doesn't make any sense. I'm going to offset this by one. So the zone plus equals one. So now, if they choose his own one, the value of the zone is one. So we haven't even bench. Now we're going to use a switch case to convert the zone number into hours of sunshine, so we're going to switch on his own. Let's quit and set the first one up. Let's start with the case of one. If they've chosen Zone one than the hours of sunshine is equal to \_\_\_. And since we found a match will disquieting break and jump out of the switch statement. If it's not Zone one, that's do a case of to once again assigned hours equal to in this case is 5.5 and we're done. So here's all the rest of them in the switch statement. It's on one all the way through zone \_\_\_. Now the default. For some reason, something goes wrong. What is to sign hours to zero, but because it's a drop down, it's virtually impossible for them not to choose something. And when we're done, let's consul dot log the value of ours. Save it. We can see that Zone one is the selected index, and it's returning a value of six. The problem is choosing a different zone. We now need to activate the JavaScript, and we haven't got a button set up to do that yet. But what we do want to do is take this code block with the switch statement and turn it into its own function again. So we'll make a new function. I'm gonna call it son hours, and we'll end that function down here with an end function. Now, instead of doing a console log which once again goes into oblivion, we'll return the value hours and then we need to come down here and call it here. Online, 52 will create a new very well. Call it son hours per day, and we'll set it equal to the results of this function call, which is sun hours. I'm not gonna pass anything, but we do want to do a console dot log and show the scent hours per day to make sure that it's working correctly. We'll say that hit, refresh and notice Now that we have a six showing up after we have the data kilowatt usage showing up. Now let's hook up this calculate sore needs button so that we can actually change the zone and make sure that we're getting the results here in the consul lock. Let's go back to remain Js. I would drop these function calls inside of their very own function, and we'll call it Calculate Solar, and it will end down here. So let me run that calculates solar function. It will find the daily use kilowatts, and it will find the sun hours per day. It was jump back now to our HTML here online 74. We have a buttoned, and when the user on clicks, we're going to run the calculate solar function. Let's save that. Take a look at it. Nothing's over here in the log. Click the button, and sure enough, it works. Change the zone. His own four gives us 4.5 hours of Sunshine Zone six. We only get 3.5 hours of sunshine

Additional Calculations

[Autogenerated] So at this point we have the months being calculated from this series of inputs. We also have the sunshine per day being calculated from the user's choice. Now we need to run some calculations on the back end. Let's jump to remain Js file down here in the calculate solar function is where will make this happen. We have the daily use kilowatts, so we know how much energy is needed per day. We also know how many hours of sunshine per day, which is value between, say, three and six. We need to divide those in order to get the minimum kilowatts needed. So let's create a new variable. Call it men kilowatts needs. We're going to set it equal to the daily, use kilowatts and divide it by the hours of sunshine per day. And, of course, you want to see the results. So one do another console log and will take this minimum. Kill what needs and paste it out. It refresh. Jews own three. We can see here that the minimum kill, what needs is 5.277 Now remember that their days when there's clouds there's days when there's snow and rain so we can't generate just 5.277 We have to increase that by 25% in order to account for bad weather. Let's create a new very well call the rial kilowatt needs and we'll set it equal to the minimum kilowatt needs and multiply it by 1.25 which is an increase of 25%. And then we want to do a console and will show the rial kilowatt needs in our lock. Refresh it this time when shoes own one, we can see that 4.397 kilowatts increased by 25% is now 5.49 kilowatts. Well, in order to compare the home usage which is in kilowatts to the panels which are rated in watts, we need to multiply by 1000. So our next step here is to create a new variable and we'll call it Riel. What needs as opposed to kill what needs and we'll take a riel, kill what needs and multiply it by 1000 to get the rial watt needs. Once again, we'll do a consulate out log and well paced the rial. What needs in there So we can see it in the lock. Refresh, choose own to weaken. See that the 5.99 kilowatts is now equal to 5996 watts.

Choose the Solar Panel

[Autogenerated] now this next part is a little bit tricky. What we need to dio is figure out which panel the user has selected and the number of Watts per hour that it generates. So over here in my select online 65 you can see I have three options. The name of the panel. But then the value is actually the watts that are generated by that panel. So this first one generates 250 watts per hour. The second one generates 275 watts per hour, and the third one generates 260 watts per hour. So let's access this panel in Sardar JavaScript above the calculate Solar will make space here for another block of code, creating new variable called user choice. It will set it equal to the document form solar form by name panel, which is panel here. And then we'll ask for the selected index, which is going to give us a zero a one or a two, and it will put the user choice into this variable. Next, we need to extract of these options into their own array, but a new variable called panel options, and we'll set it equal to the document forms solar forms panel. But instead of selected index, we're gonna do options. So now we need the two parts. There's the Watts generated. And then there's the name of the panel. So creative variable called Power and we'll set it equal to panel options. Dot value and the panel options isn't array, and so we used the user choice very well to pull out the value of the correct one. Next one is very similar this time instead of power. It's the name of the panel. Most of the value. It's the text that's inside of the select. Finally, we put those two together inside of their own variable, so variable acts is going to be equal to an array. This array is going to have two things inside of it. The power generated comma in the name of the panel Consulate Out log. Let's see what X gives us. Save it your fresh And here it is two and 50 watts, and it's that a X I t. X panels selected No, once again because it's not inside of our function calls when we hit Calculate solar. That doesn't refresh, so we need to wrap this thing inside of its own function. I'm gonna call this function calculate panel, come down here and I will end it. Clean this up a little bit. Instead of logging out the value of acts, I'm going to return it. We'll return it to who we need to know. Call the calculate panel. Close. Come down here inside of our master calculates Solar where we've called all of these other functions come down here will create a very well called panel info, and we'll set it equal to this calculate panel function. So the value of X now is going to put into panel info. Remember, X is an array. It has two values inside of it, a power and name. We need to extract those into their own two variables. So variable camera output is going to be equal to the panel info. And we're going to get the first element, which is zero second variable called panel name. Well, say equal to panel info because it's an array will get the second value, which is one semicolon. And now we need a consul. Lock both of those to make sure everything is working correctly. Save it never fresh. Let's choose the Canadian one. There's the Canadian name and generates 275 watts, which is the first one. It's a 250 watt panel, and there's the name of the panel now that we have the rial what's needed and we have the panel that spent selected by the user weaken do some math to find out how many panels air needed for the total home. So it's created a variable. We'll call it panels need it, and we'll set it equal to the rial. Want needs and divided by the panel's capacity, which is panel output semicolon. And let's do a consul that log and find out what the result is. Save it. Switch to our browser calculator needs, and we can see that it requires 21.9885 panels to run this house. Well, no one's going to sell you 0.988 parts of a panel, so we need to round it up so we can take this calculation right here and wrap it inside of math, dot ceiling and then close the print to see their now save it and refresh. We can see that spent up to 22 panels for the home

Provide Feedback to Customer

[Autogenerated] Inter html page. We're going to replace this information using JavaScript with the results of all of the work that we've done. So the best way to do this is the first of all right out a script without any variables in it. So here's mine. I've created variable called feedback notice I'm using plus equals. So it adds to the script each time. So line one based on your average daily use of kilowatts per hour, and this is going to be replaced by a variable you will need to purchase once again a made up number 99 and then the brand of the panel, Goffstown. 100% of your electrical bill. Fine. And then we're gonna throw in an age to from this one another paragraph. Your average daily electricals consumption is and we'll get this is a made up number, but we know how many kilowatts per day because it's up here. Same thing, average sunshine per hour. We know the number of hours from our switch statement. We also know that based on clouds and rain, we need to increase this by 25%. So we'll show them that there and then we're going to say Finally, the name of the panel, which is the name that they selected, generates so many watts per hour, and then we're gonna take all of that feedback we're gonna drop it. Is the inner html off the feedback I D. So let's save that. That's refresh calculate, and here's our nonsense. Data right now fits beautifully on her page. It just doesn't have the correct information in it yet start by replacing this average daily use of so many kilowatt hours. Let's get rid of the exes. I will put a quote and then a plus plus, and between the plus plus is where the variable goes. So based on your average daily use off in this case, it's daily used. Kilowatt. Save it fresh calculate, and there it is 26.38 I don't think anybody wants to know that level of detail, So let's round that thing down. Math dot round It will take that variable and simply rapid inside of parentheses. Save it, refresh and now we can see it's a normal 26 kilowatt hours. Next thing is the number of panels. We've calculated that so let's delete it quote, quote plus plus between them will put the variable. In this case, it's panels needed. We also need to replace the brand, which is the name of the panel. So delete. Quote plus plus in between it, Let's sandwich the panel name, Save it fresh calculate to know we conceive that 26 kilowatt hours you'll need to purchase 22. And there's the name of the panel just the way it came off of this descriptor right here. So we chose that Canadian ones takes 20 Canadian panels to run your home. And if we chose the third one, I take 22 of those. So first part is done. Now we just need to come down and do the same thing with these additional details. The process is exactly the same. Your average daily electrical consumption. What would delete that? Put in quotes. Put in a couple of pluses between them. We'll put the rounded value of the daily use kilowatts. Now we're looking at the average sunshine per day, which is generated from that map quote plus plus paste sunshine hours per day. The value and increasing by 25% is the rounded value of riel. What needs the name of our panel is panel name the What's generated his panel output. Save it, refresh it. I've got it. Air in there somewhere. I'll get two pluses here online. 90 to save that. Try it again. There we go. So let's choose the Canadian solar panel. Calculate it years all of our nice information average data used with numbers instead off empty placeholders. At this point, we no longer need all of this log information so we can come back and comment out. I think that's all of them. It's a refresh. Sure enough, there's no more data spewing out here on the console log. We also need to return to our HTML page and set all these values, which were pre defined 20 so that the user enters their information instead of working with these values that they might get confused as important. So we'll strip out that that should return everything to its default state. So these are all empty zone one, So I would now have to completely fill these out with my own numbers. I would have to choose my zone. I would have to choose my panel calculator and it would work beautifully. Well, that concludes this solar calculator. Thanks for working through these hands on activities with me as we have built this Web application together and look at everything we've covered. Hopefully you can use it to build something awesome in Java script.