**Solution: Build and modify an array**

Selecting transcript lines in this section will navigate to timestamp in the video

(upbeat music) - [Instructor] Welcome back, here's how I solved the "Build and Modify an Array Challenge." You'll find my solution in the exercise files for this movie. It's sitting in script.js. And if you look closely, you'll see I've commented out all of my solutions so that we can walk through them step-by-step. I've also provided links to the relevant array methods for each of the solutions to each of the sub tasks. So you can go check out the examples and the documentation for each of these methods on your own time. All right, let me just collapse this so it's easier to see what's going on. At the top I defined a new array called desk array, and that just has a bunch of things that are on my desk right now. Pen, camera, phone, notebook, headphones, light bulb, and a USB drive. And then I console log out this original array. Looks like this in the browser, and we have seven items and these are the seven items. Now pay close attention here because the last item has a capitalized USB at the top. This will come into play later. All right the first challenge was to remove the last item. For this we use desk array pops. So the pop method takes the last item off and returns it to us. If I uncommon these two lines and save the page, you'll see here we have the new array that just has six items and USB drive is missing. I know I said this method pop returns to us the last item. So if I do this console log and then ramp desktop pop here, you'll see in the browser we now get USB drive as an element. This will come into play in our next code example. So comment out these two lines and go down to number two here, add last item as the first item on the array. So what we need to do here is grab the last item and take it off the array. That's what we used the pop method for, and then use the unshift method to push the item back to the front of the array. So here is what I was saying in the setup to this challenge that you can nest methods inside other methods. So I'm saying desk array unshift because I want to put something to the front of the array. And what I want to put on the front of the array is desk array pop. That would be the item that gets returned when we pull it off the end of the array. So desk array pop pulls the item off the end of the array and returns it to us. Then we use desk array unshift to put it back onto the front of the array. So we're moving an item inside the array save this, go in the browser, and now you see we've reorganized it. So in the original array USB drive is at the end, in the new array USB drive is at the beginning, cool. All right, the next challenge is to sort the array in alphabetical order. Now here I use the sort method. I'm just going to use it the way it is because by default it will sort based on alphabetical order. So I say desk array sort and output that array. Now look at this. Here we get a sorted array except USB drive is at the top. Hm that's weird. 'Cause U is definitely later than C in the alphabet, right? So technically this should probably be towards the end. Well, remember how I said USB drive has capitalized letters and that comes into play later. It comes into play here. Capitalized letters are sorted before lowercase letters. I'll show you what I mean. So if I go in here and change, let's say light bulb to capitalize L save it. Now you see it says light bulb, USB drive, and then all the other ones. So that means if I want USB drive to actually be at the end of my array, I need to change it to lower case like this. And then it'll be alphabetically at the end of my array. This is one of those idiosyncrasies you'll see a lot of when you work with Java scripts and this is why it's so important to unit test all your code, meaning passing any possible information that could go into your code to make sure everything is working properly. So the sort function works the way its supposed to, just that it does something you may not have been aware of it doing. The next challenge is to find a specific item. So here I wanted to look for the item with the name notebook. So that would be the actual value that sits in the item. Now you can find items in different ways in arrays. You can look for them by their index number, or you can look for them by their contents. So that's what I want to do here for that purpose. I use the find method. Now here I've created a constant called found item and then set it equal to desk array find, and then inside the fine method, I set up a callback function that grabs each item in turn, and then it loops through all the items until it finds an item that matches my query. In this case, an item that is equal to XX equals to notebook. Then it returns only that item into my constant and then output my constant, save that. And here we have just notebook as the item. Now if I want to, I can test this with some other values. I'll say cat, I don't have any cats in my array. So if I save this, I get an undefined in return because there is no item for that. So you can see that the function works as intended. The final challenge is to find and remove an item. So here I want to find an item based on its name, but I then need to remove it as well. And here I need to use two different methods. So what I did was set up a let first call it remove, and then put in the value I wanted to remove. This is just so that I can test it later and so that it's easy to work with. Then I first use desk array index of to find the index number for the remove item. This will give me a number in return and it matches the slot. The item I'm looking for is currently sitting in. The item I'm looking for in this case is notebook. So if I put quotes notebook in here, it would do the same thing, but I'm using it as a variable instead. Now that I have that index value, I pass the index value into the splice method. The splice method takes two or three arguments. The first argument is the index value for the item you want to start looking at. So if you want to go through the whole array, it would just be zero. In my case, I want to look for, I want to make a splice out of specific points. So I pass in the index number for the current value I'm looking at. Then the next value number one here is the number of items I want to remove from the array. So in this case, I want to remove one item and it is the item I'm pointing at. Then we could, if we wanted to add a comma here and then add another value and that value would be what we add to the array in addition, but that's not what I want to do here. I just want to remove an item. So I want to remove the notebook item, save this, go back in the browser. And here you can see I have the array with notebook removed. So again, I'm using this nest thing ability to nest one method inside another, to get everything to work. Now that you've seen how all of these different solutions work, I encourage you to go check out the links that I provided in the solution to look at the documentation for each of these methods and see how it's set up and what you can do with them.