**Array methods**

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

- [Instructor] Arrays are used a lot in JavaScript. So there's a whole array of array methods to do things with arrays and their contents. All of these methods are heavily documented with examples in the MDN web docs. You can find them under static methods and instance methods. And this whole page is well-worth a read once you're finished with the course. In the meantime, let's look at some of these methods in more detail. In the exercise files for this movie, I've set up a basic array called backpackContents. It contains three items: piggy, headlamp, and pen, and when we output it, you see them over here in the browser. All right. So far, nothing new. A very common thing we want to do with an array is just output all of these values in the array as a string. This can be done using the join method and the join method is added directly to the object itself. So I'll just say .join, and then call it as a method by putting the parentheses at the end. Save. And now we get a comma-separated list of each of the items. The problem is, this looks weird, right? Because there's no space between here. So I can pass in a parameter saying what type of separator I want. So if I want a comma-separated list but I want some space, I say comma space. Save. And now we have comma space. Of course, I can pass in something else, too. Space pipe space. So here you have full control over how you want to output those items. Also note, the join method outputs a string. So once we've done this, we now have a string that we can work with just like any other string. Another common task is to add one or more new items to the end of the array. Now, in the previous movie, you saw we can do this by grabbing the array length and then just appending more content. But we can also use a method to do this and the method is more secure because we're not risking overriding anything in the array. That method is called push because we're pushing more items onto the array. So we'll grab our array, backpackContents.push and then we provide a comma-separated list of all the items we want to add. So here I want to add a pencil and I want to add the number five. Save. And now my array has five items: piggy headlamp, pen, pencil and the number five. Push adds new items to the end of the array. If I wanted to add the new items to the front of the array instead, I use a method called unshift. Now, it sounds a little weird, unshift, and I'll explain why it's called unshift in a second, but let's just see what happens if I do this first. Unshift... Sorry, spell it correctly. Unshift, save. And now we can see the pencil and five are added to the top of the array, the front of the array, slot number zero and number one. So why is this method called unshift? Well, there's a simple reason. The shift method takes the first item off the array. So right now, the array has the piggy as the first item. If I say backpackContents shift and save it, piggy is lost. We've taken piggy off the front of the array. If I want to take something off the back of the array, so the last item, in this case pen, I use a method called pop instead cause I'm literally popping the item off the array. Now we have piggy and headlamp. Now, all of these methods are fairly straightforward. We either just apply them directly to the array object or we apply them and pass on some parameters to do something. In addition, many array methods allow us to do more advanced things like apply functions to each item within the array. So let's see how that works. What I want to do here is add an L-I around each of the items and then output them to the console. To do that, I'm going to wipe out this example and then copy in this commented-out section down here and then take off the comments. Here, I'm going to use the For Each loop. So the For Each loop will grab each item and do something to that item. The For Each loop doesn't do anything to the array itself; it just loops through the array to do something. So here we say backpackContents for each item. Then we run a function on each of those items, passing the item itself in as the parameter. And what we're doing here is setting up the item equal to list item plus the item plus end list item. And then we just console log that out in the browser. Save that. And now we get a list. So, piggy with L-I before and after, head lamp with L-I before and after, and so on. So the For Each loop gives us the ability of manipulating the contents within the array and then doing something with those contents outside of the array without touching the array itself. You'll see down here, the array is still the same. We didn't touch it. There's another example here. If we copy that out and paste it in and un-comment it, this one is a little bit different. I'm just going to comment out my For Each example, as well. Here, we're doing something else and even more advanced. We're saying we'll create a new let and then we go to backpackContents to find the first item that meets our parameters and the parameters are defined inside a custom function. The function grabs each item in turn, and then it says, if this item has a length that is bigger than or equal to five, meaning there are five or more characters in that item, then return the item. And this will return the first item that meets this requirement. Save. And we get in return piggy. We can then test it by saying, what if we just call it pig, save, and this time we get headlamp in return. So here you can see, using some of these array functions, we can apply new functions to the individual items within the array, or we can customize the array in pretty much any way we want. As I mentioned, there are a ton of methods for arrays, and you can see a list of all of them in the MDN web docs, where you can also see code example for how to use each of them.