**Arrow functions and "this"**

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

- [Instructor] We need to take a quick sidebar here and talk about the arrow function scope and the 'this' keyword. Earlier, we saw that 'this' keyword used when we created an object constructor and you may remember, this keyword can be used inside any object to refer back to the object itself. So for example, here, we have a method inside our object called 'new volume.' New volume contains a function. And inside that function, we console log out the text, this volume in the method. And then we use 'this' keyword to refer to the volume property of the current object. You can see over here that when we use the functions we call it down here, console log(greenpack, newvolume), when we use the function we get the volume in the method, which is eight. So that's what's declared up here. And then afterwards, I use this volume to re-assign a new value to this attribute. And that's the value that's passed in through the new volume method. So it gets captured up here and then assigned. And therefore the second entry says this volume after update is five. So we're passing five in to the method, five is then reassigned to the value. And we can refer back to this because this has now been changed. The important thing here is, that this keyword works fine inside a regular function inside an object. But, if I call in a function inside that function, weird things start happening. So to understand this example which admittedly is a bit convoluted, I need to explain exactly what's going on. By the way, this is convoluted because these issues only arise in convoluted circumstances. So this is really the easiest example I could come up with that fully encapsulates what's going on. Below here, I've added an immediately invoked function expression which is an anonymous function. So you can see that we're running a function inside our method. And in this function, I say this volume in nested function. And then we, again, I'll cut this volume. Now looking at this, you would say, well, we're doing this right after we re-declare the volume number, right? So we should get the same value as the one above here. But if I save this, the output we get is 20. Why, well that's because, if you look all the way at the top of the file, we've declared the window object volume to 20. Now remember, I said, if you use a function declaration that function declaration is hoisted to the global scope. So what's happening here is this function is hoisted out of the object and up to the global scope. And suddenly, this is pointing at the window object and we're getting the volume of value from the window object, which is not what we want. This is where the arrow function comes in. And this is really important. So I'll copy this and paste it in. And then instead of using a function expression, I'll use an arrow function. So I'll say parentheses and then an arrow and then save it again. Now look over here, you see that the function expression outputs 20. So it's hoisted up to the global scope. Whereas the arrow function stays within the current scope and output five. The interesting thing is this is happening because an arrow function does not have its own 'this.' It does not know what this means and it will refer to the closest available scope which in this case is the object. And that's why we're able to get a hold of it. This also explains why we can't use an arrow function as the declaration for a method, because if we did that, the arrow function wouldn't know what scope to use. So we would refer back to the global scope instead of the method scope and therefore nothing would work properly inside that arrow function. So the bottom line is, if you're using 'this' in a method within an object, and you then get an odd result, try turning the function into an arrow function to see if that solves the problem. Most likely, you're dealing with the wrong scope and an arrow function will help you get the correct scope because it doesn't carry its own scope with it. For more on this, you can read the very long and detailed article on MDN that explores 'this' and all its interesting parameters in an object. But I recommend you do that after you're finished with this course.