**The arrow function**

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

- [Instructor] In modern JavaScript, it's more and more common to use arrow functions in place of regular function expressions. Arrow functions are a shorter way of writing function expressions, and they also have some special properties, we can take advantage of. The MDN web docs, have a good breakdown of a prototypical arrow function, you can see it down here. So here we have a traditional function. We declare the function and then we have some properties and then we have the curly brackets and the function body. Looking at the arrow function below you can immediately see it is different. Instead of declaring function, we just use the parenthesis and then we use this arrow, it's an equal symbol and then larger than symbol that literally looks like a big fat arrow, and it points at the curly bracket. Everything else inside the functions of the function body is exactly the same. So the arrow function refers to this declaration here, parentheses with parameters inside and then an equal symbol and the bigger than symbol. Let's see what that would look like in our code example. So right now we have a function expression where we say const, addPack, function, and then the currentPack and then the curly bracket. If I want to transform this into an arrow function, all I need to do is take away the function declaration. So we have just the parentheses. And you'll notice immediately the code editor is saying, "Hey, hey, hey, you're doing something wrong here." So if I then hover over this error, it says, arrow expected because it now says, "Oh, you're trying to make an arrow function, well you need to actually add the arrow for me." So, I'll add in the arrow, save. And now everything works exactly as it did before, except we're no longer explicitly saying, "This is a function," instead we're using this arrow function syntax. Now you'll often see arrow function syntax be simplified even more because a lot of the time we're using them when we're just passing an anonymous function that doesn't have parameters. So you'll often see this syntax here, just parentheses and then the arrow and then a curly bracket. And in some cases, if you're just passing one parameter, you can also just name the parameter itself. So I could say currentPack, and then arrow, and then this curly bracket. And this actually works the same way as if I had the parentheses around. But notice what happens if I do this and then save the file. Prettier, we'll go, "Hey, hey, hey, that's not easy to read." So I'm going to add the parentheses back for you so that people can actually understand what you're doing, because I want your code to be formatted properly. So this in a nutshell is the arrow function. So why do we have two different ways of doing the same thing? On the surface, arrow functions are just a simpler way of writing anonymous functions. And as you'll see later when we work with events, they produce a lot cleaner code. But there are some subtle differences worth knowing about. First of all, function declarations can be hoisted, meaning you can call the function before it is declared in JavaScript. Now this is improper coding practice, but it does work. Arrow functions on the other hand can only be called after they have been declared. So we are impulsing a stricter way of writing our code by using arrow functions. Second, you can't use arrow functions when declaring methods in an object. Inside an object if you have a method you need to use a proper anonymous function declaration. The arrow function simply doesn't work in that context and we'll talk a bit about scope and this later, and that will make a lot more sense. Oh, there's one more thing worth noting about arrow functions. You can reduce and simplify the arrow function syntax to the point where it becomes really hard to understand what's going on. And some developers like doing this, personally I like to keep it verbose so I can actually see, but here in the MDN web docs, you can see the examples. So we have our regular arrow function here with the parentheses, the parameter, the arrow and then the curly bracket. Then there's a reduced model down here where we just have the parameter and the arrow and it points directly as the output. So in this case, there's very little going on inside the function so we don't have the curly brackets. And then we can also take away the argument parentheses or the parentheses that go around the parameter. So we just have a, arrow, a plus 100. Looking at this there's a good chance you go, "What is happening here?" That's what happens when I see it, I'm always like, what? I don't understand this. So like I said, I like formatting it properly or verbosely so that I can clearly see what's going on. And I think that's just good coding practice. That said, you will encounter this super short formats