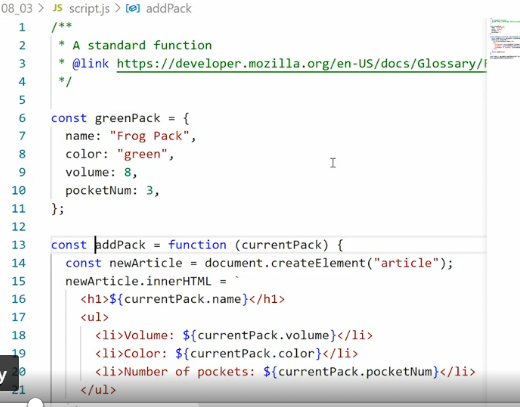
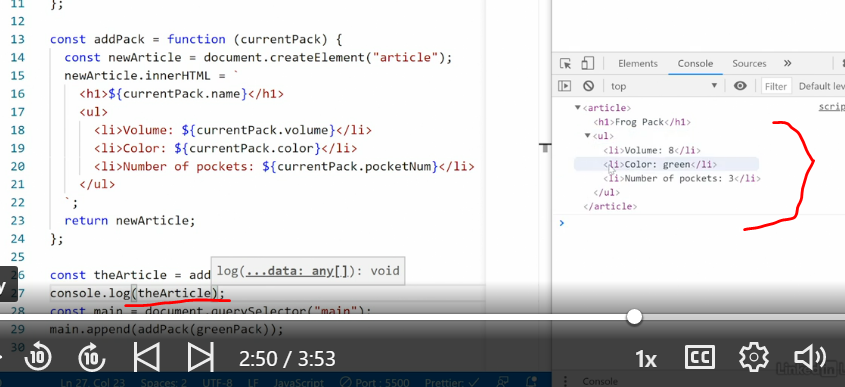
* - [Instructor] Looking at prototypical examples of functions like this, is useful but to fully understand functions and how they operate, you really need to see them in a real life scenario.
* So in the exercise files for this movie, I've set up such an example, and you can see right away that there's a lot more going on here.



* In this example, I've reduced our previous object demo to its core features.
* At the top, there is a standard object named 'greenPack' with some basic properties.
* Then towards the bottom, we have a constant called 'main,' containing the main element of the HTML document, and we use the append method to append some content to that main element.
* Graphical user interface, text, application, email

  Description automatically generated
* That content is displayed using this function expression called 'addPack.
* ' So now we can look at how the function interacts with the rest of the content.
* First off, this is a function expression, meaning we have set up a constant.
* We've given the constant a name, and then we've set its value equal to the function.
* *Here we have an anonymous function that has a single parameter, and that parameter is used throughout the functions.*
* You can see it says 'currentPack' here, and we are using that parameter throughout the function.
* And this is really important for our functions.
* When we have parameters, we can pass any value in, and then reuse the function over and over and over on different values.
* So in our circumstance, we are passing in the greenPack object.
* But we could in theory, pass in many different objects, and then the object's name is converted to this generic name when it's inside the function, so we can repeat the function over and over and over again.
* The actual functionality of the function is wrapped in the curly brackets.
* You can see my code editor is helping me to see where it starts and ends.
* And this is what's called the function body.
* This is where everything the function does actually happens, and this can be absolutely anything.
* So in our example, we are setting up a new constant using 'createElement' to create a new article, then repopulate that new article with some 'innerHTML,' and here we're using the 'currentPack' property and getting the name, the volume, the color, and the pocket number.
* And finally, we're returning the value back to where we called it.
* So right now, we are saying, main append, and then we're appending the returned value of the functions.
* We're calling the function, the function returns this new object that is an article, and that article has HTML, and that's what you're seeing in the browser.
* Interestingly, we can also populate a variable with the same value.
* So I can set up a new constant here, call it 'theArticle.
* ' Set it equal to and then I'll just copy addPack(greenPack) down here.
* And then we can console log out 'theArticle.



* ' Save that, go into the console.
* And down here, you'll see we now get the same thing returned to us.
* So here we have the article with all its contents.
* Only this time we're handling it as a variable.
* So using functions, you have a lot of options on how you want to handle things.
* *You can create a function that just does something and outputs content.*
* *You can create a function that returns content, and if you do so, you can choose what you want to do with that return.*
* You can place it in another variable or an object or an array, and then do something further with it, or you can just use it right away like we've done here to append it to an existing element or do something else.
* Now, functions quickly get super complex but this is the premise for all functions all the time.
* *The function will always have a name, either the variable that holds the function or the function itself.*
* It will always have parameters.
* Now the parameters may be nothing, so you may have just two parentheses with nothing inside but there will always be those parentheses.
* There will always be curly brackets wrapping around the function body.
* And some functions may return data back to where the function was called.