**Return values from a function**

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

- [Narrator] The tip calculator function executes code and outputs it directly into the console. In some cases, this is what we want a function to do, just immediately output content. In other cases, we want a function to return some content to where the function was originally called so that we can process that content further. So in this movie, we'll look at how to do that using the return keyword. If we look at the output from the tip calculator function right now, you'll notice we're using the punctuation mark to delineate the decimals in the currency. This is correct for some currencies in some languages but not for all of them. We're also not showing the correct currency symbols here or rather we can pass it in but we don't really know what they are. So we're not getting the correct output here. I want to upgrade my function to output the correct currency, formatting and symbols for whatever currency we are targeting. And to do that, we'll create a new helper function to transform these numbers for us. In the exercise files for this movie, I've reduced our example down to the bare minimum again, because we don't need to pass in a currency symbol and then say before or after, instead we're going to have an automated function do that for us. Now, remember how I said, JavaScript has a bunch of global objects that we can use to perform common functions. One of those functions is to transform a number into an internationalized number format. So we can say which language it should be and then it'll output the number in the correct formatting for that language. We're going to use this international number formatting object, and it's a constructor for objects that enable language sensitive number formatting, which is exactly what we want. In fact, the very first example here is an example of what we want to do. You pass in a number and then you get out the correct formatting for a specific currency. The first step of this process is to create a new function. I'll call it formatter. So const formatter, I'll set this equal to arrow function. And inside this arrow function, I'll set up a new let called formatted value, and then I'll return formatted value. So now when we call formatter, we'll get whatever is sitting inside the formatted value returned to us and then we can process that number further. What I want to do now is pass the values from down here up into formatter and do something with them. So I'll set up a parameter for formatter here and say value. And then down here in the function, I'll say formatter and then wrap it around the value we want to format. And I'll do the same down here for tip and for total. So what happens now is when I call tip calculator, tip calculator will then take each of these values, the sum, the tip and the total, pass them up to the formatter function then do something to those numbers and pass them back out or return them back out and display that down here. Now we need to actually do something with these values 'cause right now we're capturing the values but not using them. That's where the international number format object constructor comes in. And we're actually just going to copy out this entire code example here 'cause it's exactly what we want. That's often the case when you work with JavaScript objects. Then we'll paste that in here as we'll say let formatted value equals what I just copied. I'll save it so that it gets formatted properly. Then we get an error over here in the console and that's fine 'cause we're not done yet. Then we need to grab the value and pass it down into our new object, that's done here. So now the value that's passed up from the formatter goes up here and then gets returned back. And we do this and save, we get a new output here. And this output is formatted for Germany, the German language and the Euro. And if we then go in and change this to something else, let's say en CA, which is Canada, and then CAD which is Canadian dollar and save, we'll get a different output. So that means our object is working. However, we need to be able to pass the locale and the currency in manually. So I'm going to add in two more parameters here, locale and currency. And then say that locale is this thing here. So we'll pass in that as a variable or parameter. And then the currency is down here. Now that we have these two new parameters, we need to pass them up from where we call formatter. So I'll just paste them in here. So we're passing locale, currency and sum, locale, currency and tip, and locale, currency and total. Of course we don't have these values inside tip calculator yet, so we need to add them here as well. So we'll say, locale, currency, and then like that. So now tip calculator has some percentage locale and currency, and that means we need to pass in locale and currency down here where we're calling tip calculator as well. So we'll say the locale is DE-DE like our example and the currency is EUR four Euro. Save this and we get the correct output over here in the console again. So now we can see what happens. We call tip calculator and pass along the four arguments. The value, the percentage, the locale and the currency. These are all captured as parameters up here. The sum, percentage, locale, and currency. Then we do the regular tip percentage math. Then we go into the console log and say we want to pass some values up to the formatter. We pass the locale, the currency and the sum. So that's the locale, the currency, and the sum that was generated here up into the formatter who then uses the international number format object to create a new object using the locale, the currency and the value, and returns it back to tip calculator. And that's the value that we get output in the browser. So now we can pass any value we want here. So if we want Norwegian, NO and then NOK. And save it, you'll see the format changes again. We want US dollars. We say, EN US and USD. Save, and we get a different output. Now you see how important the return keyword is. Using the return keyword, we can pass information back and forth between different functions, and use functions again and again to do different things like in this example where we're using the formatter function three times inside the tip calculator function to get different results for different purposes. That way, we can create helper functions and we don't have to do everything inside one function.