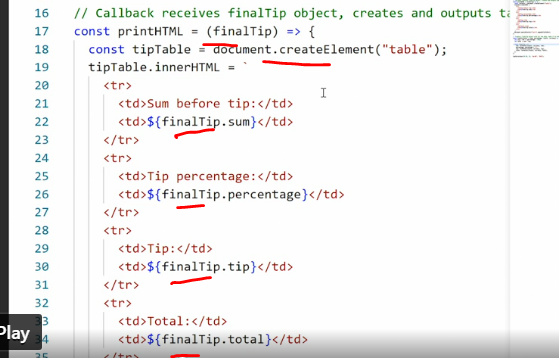
* - [Instructor] Sometimes we have two functions and we want *to control the sequence* at which they execute.
* This typically happens if one function relies on the output of another function for data or when one function has to wait for another function to complete before doing something.
* One way of **handling this type of function sequencing** is through what's known as a **callback function** and this is a traditional method for handling this type of problem and you will come across it all the time.
* In the exercise files for this movie I've updated our tip calculator example a little bit.

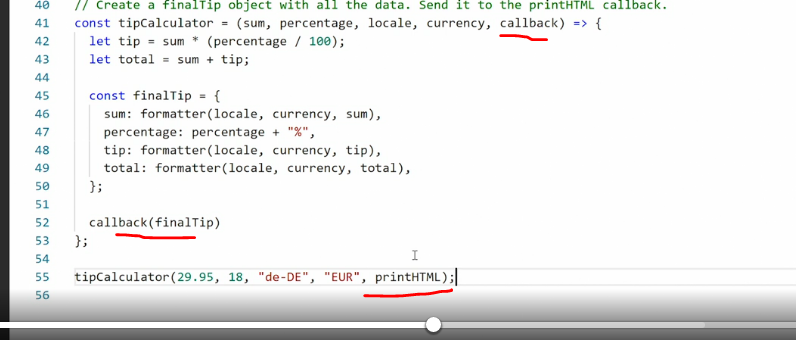
Graphical user interface, text, application

Description automatically generated

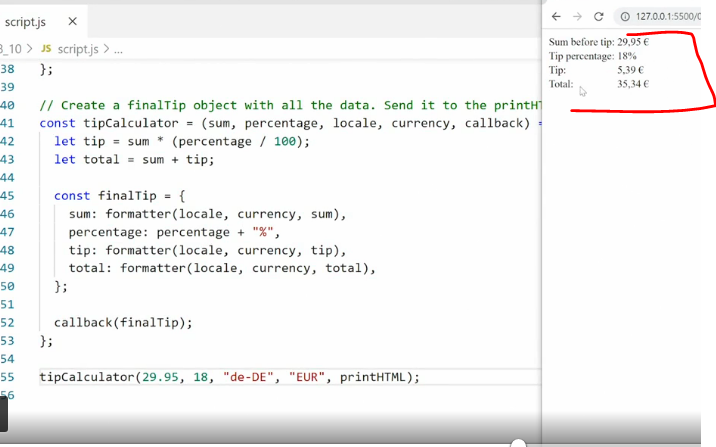
* So from the bottom, ***the tip calculator function now creates an object named final tip with all of the data***.
* This object uses the format or function we created previously to format the number values and nothing is new here, except for that object.
* **What is new** is directly above here ***is this new callback function called print HTML that receives the final tip object***.
* *The idea here is I want this print HTML callback function to output an HTML table that appends to the main and the document so we can see the results of the tip calculator in the browser over here.*
* To do that I need to make sure that things happen in the right order.
* So a) I want to pass my values into the tip calculator and b) only when everything is done do c) I want to call in the new print HTML function and d) then that function will output the content into the browser.
* This is where we'll use **a callback function** and **call it callback**.
* It can be called whatever you want but I'll just call it, call back cause that makes it really clear what's going on.



* So expand this to the full view so you can see what's going on.



* So callback is here then at the bottom here, I'll say call back and inside call back I'll pass final tip then inside the callback I'll pass final tips so that we're throwing the final tip object to the function in question.
* Now the functioning question is print HTML up here but it doesn't say print HTML anywhere so far, right? Well, what we're going to do now is pass a call to the callback from tip calculator down here.
* So we'll say the name of the function we want to run which in this case is print HTML, looking at this and thinking back to when we were talking about function expressions and function declarations I think you might be able to figure out what's going on.
* Remember how ***if we have a function and we call it using the parentheses then we are executing the function*** but **if we are calling a function expression without adding those parentheses, we're actually calling the entire function and passing it forward.**
* So what's happening here is when we call print HTML without the parentheses in the end, we grab this entire constant here with the print HTML function inside it and then we pass the entire function up into tip calculator as a value that value the function gets pat captured in this parameter callback and down here we then place that entire function inside the callback and append final tip.
* So that way we're effectively running print HTML inside the tip calculator function instead of sending values to it.



* So when I save this and go look in the browser, you see we get the output we want inside the browser.
* If I reduce this again you can see if we pass different values here.
* So we have nb-No and NOK, save and we get the updated information over here.
* So everything is working the same way except I'm now using a callback function.
* Looking at this, you may be asking an obvious question which is **why can't I just call the function itself**? I mean, I could just call print HTML down here right? And I would get the same output or we can try.
* So I'll say print HTML save, yeah, you're right we get the same output.
* So why on earth would I want to do this callback thing, right? Pass the entire function up in and then use it.
* Well, there's a simple reason.
* ***There's a good chance when you set up some more complex code that you may have different callback functions you want to use for different purposes***.
* ***So in this particular circumstance we want to use the print HTML function but there could be several different versions of the print HTML function and then you want to use them for different purposes and by calling a callback like this, you can pass in exactly the function you want into the other function***.
* So we're effectively saying here is the precise function I want you to use once you're done processing your information right now but later it could be a different function.
* Like I said callbacks are a traditional way of handling this and in modern code, we also have more modern tools like **promises** that can do something similar but you'll see this pattern all the time when you work with JavaScript where you call in a function or call back a function and then get a value.
* You'll also see this pattern and other types of features.
* For example, if you are working with a set timeout function you are in fact working with a call back function or if you're working with an event listener which is something we'll do later on in the course you're also working with a callback function.
* **So this format of throwing a function into another function and then getting it to execute is a standard pattern you see all the time in JavaScript.**