**What are Callbacks in JavaScript? –**

[Codesphere](https://medium.com/@codesphere-cloud?source=post_page-----921e0111fe66--------------------------------) Jul 30, 2021

Anyone familiar with programming, already knows what functions do and how to use them.

***But, what exactly is a callback function?***

Let me walk you through a few examples that will help you understand what callback functions are and how to use them in JavaScript.

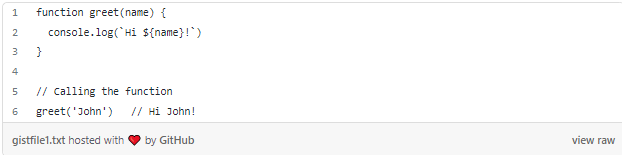
**Functions**

Let’s start with what a function is.

A function is a block of code that can be executed repeatedly, as needed. Functions are really useful because you write the code once and then you can run it multiple times.

To run the code inside a function, simply call the function.

Here’s an example that defines a function and then calls it ‒



In the code sample above, we created a function called greet() (lines 1 ‒ 3) that has a parameter called name. This parameter serves as a placeholder so that when the function is called, a value can be passed to the parameter.

In line 6, we called the greet() function and we passed the string value ‘John’ as an argument.

# Callback Functions

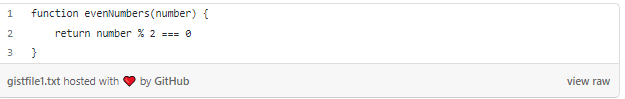
A callback function is simply a function that is passed into another function as an argument, which is then triggered when the other function is executed.

The following is an example of using a simple callback function.

Let’s start by creating an array of numbers that we will use in our example ‒

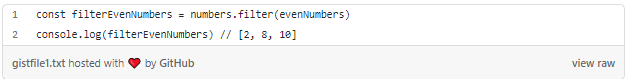


Here’s a function that checks for even numbers ‒



To find all the even numbers in the array that we created above, we can use the JavaScript filter() method, which returns a new array that only contains the numbers that match the function’s condition, meaning that they are even.

Now, we can pass the evenNumbers() function, described above, to the filter() method and assign it to the filteredEvenNumbers variable, as shown below ‒



In this example, the evenNumbers() function is a**callback function**, because it is being passed as an argument to another function.

# Synchronous Callback Functions

By default, JavaScript runs code sequentially from top to bottom by completing one command line before starting the next.

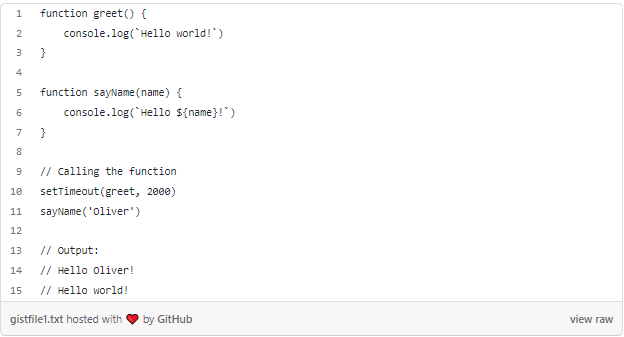
The evenNumbers() function we created is an example of a synchronous callback function.

# Asynchronous Callback Functions

Asynchronicity means that JavaScript does not wait for the operation of a command line to complete before it starts executing the next line of the code.

**If you are familiar with setTimeout, then you’ve been using callback functions all along!**

In the following example, we are going to use the setTimeout() method that only executes after 2 seconds ‒



In the example above, we used the setTimeout() method and called the greet() function to execute 2 seconds later. JavaScript starts counting 2 seconds. But meanwhile, it proceeds to the next command, which immediately displays ‘Hello Oliver!’. Then after the 2 seconds are over, the greet() function is run, and displays ‘Hello world!’. Therefore, even though the greet() function was called before the sayName() function, they were executed in reverse order.

# The Next Step

I hope this tutorial has helped you understand what callback functions are and how to use them as easily as functions!

There is still much to learn about callback functions, so I encourage you to continue practicing and learning.

To practice using callbacks, check out [Codesphere](https://link.codesphere.com/BX" \t "_blank), an online code editor in which you can build, test, and deploy your apps all within a single tab of your browser.