JavaScript Callback Functions–What are Callbacks in JS and How to Use Them

freecodecamp.org/news/javascript-callback-functions-what-are-callbacks-in-js-and-how-to-use-them

March 17, 2020

If you're familiar with programming, you already know what functions do and how to use them. But what is a callback function? Callback functions are an important part of JavaScript and once you understand how callbacks work, you'll become much better in JavaScript.

So in this post, I would like to help you to understand what callback functions are and how to use them in JavaScript by going through some examples.

What is a Callback Function?

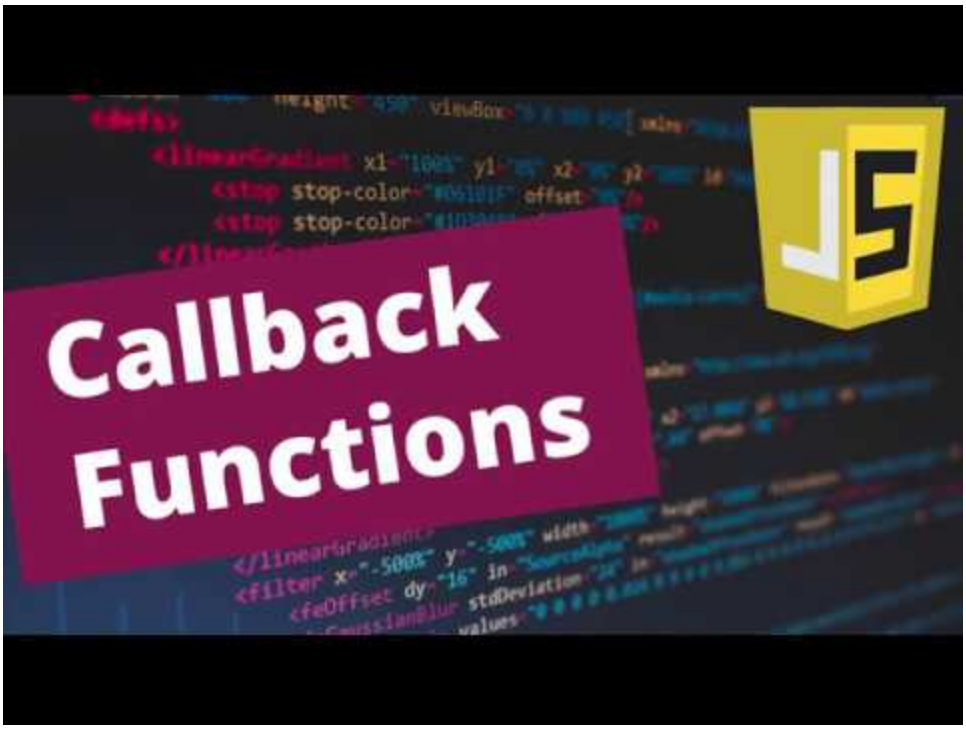
In JavaScript, functions are objects. Can we pass objects to functions as parameters? Yes.

So, we can also pass functions as parameters to other functions and call them inside the outer functions. Sounds complicated? Let me show that in an example below:

```
function print(callback) {
    callback();
}
```

The print( ) function takes another function as a parameter and calls it inside. This is valid in JavaScript and we call it a "callback". So a function that is passed to another function as a parameter is a callback function. But that's not all.

You can also watch the video version of callback functions below:



Watch Video At: https://youtu.be/qtfi4-8dj9c

Why do we need Callback Functions?

JavaScript runs code sequentially in top-down order. However, there are some cases that code runs (or must run) after something else happens and also not sequentially. This is called asynchronous programming.

Callbacks make sure that a function is not going to run before a task is completed but will run right after the task has completed. It helps us develop asynchronous JavaScript code and keeps us safe from problems and errors.

In JavaScript, the way to create a callback function is to pass it as a parameter to another function, and then to call it back right after something has happened or some task is completed. Let's see how...

How to create a Callback

To understand what I've explained above, let me start with a simple example. We want to log a message to the console but it should be there after 3 seconds.

```
const message = function() {
    console.log("This message is shown after 3 seconds");
}

setTimeout(message, 3000);
```

There is a built-in method in JavaScript called “setTimeout”, which calls a function or evaluates an expression after a given period of time (in milliseconds). So here, the “message” function is being called after 3 seconds have passed. (1 second = 1000 milliseconds)

In other words, the message function is being called after something happened (after 3 seconds passed for this example), but not before. So the message function is an example of a callback function.

What is an Anonymous Function?

Alternatively, we can define a function directly inside another function, instead of calling it. It will look like this:

```
setTimeout(function() {
    console.log("This message is shown after 3 seconds");
}, 3000);
```

As we can see, the callback function here has no name and a function definition without a name in JavaScript is called as an “anonymous function”. This does exactly the same task as the example above.

Callback as an Arrow Function

If you prefer, you can also write the same callback function as an ES6 arrow function, which is a newer type of function in JavaScript:

```
setTimeout(() => {
    console.log("This message is shown after 3 seconds");
}, 3000);
```

What about Events?

JavaScript is an event-driven programming language. We also use callback functions for event declarations. For example, let’s say we want users to click on a button:

```
<button id="callback-btn">Click here</button>
```

This time we will see a message on the console only when the user clicks on the button:

```
document.queryselector("#callback-btn")
    .addEventListener("click", function() {
      console.log("User has clicked on the button!");
});
```

So here we select the button first with its id, and then we add an event listener with the addEventListener method. It takes 2 parameters. The first one is its type, “click”, and the second parameter is a callback function, which logs the message when the button is clicked.

As you can see, callback functions are also used for event declarations in JavaScript.

Wrap up

Callbacks are used often in JavaScript, and I hope this post helps you understand what they actually do and how to work with them easier. Next, you can learn about JavaScript Promises which is a similar topic that I've explained in my new post.

If you want to learn more about web development, feel free to follow me on Youtube!

Thank you for reading!

Cem Eygi

Front-end Developer // Follow Me on Youtube: https://bit.ly/3dBiTUT