Callback functions in javascript | JavaScript Tutorial In Hindi #37

In the previous lecture, we have discussed what functions do and how to use them. But now the question is, what is a **callback function**? Callback functions are one of the most important parts of JavaScript, and once we understand how it works, we will become much better in JavaScript. So, in this tutorial, we will understand what callback functions are and how to use them in JavaScript.

What is a Callback?

A callback is a function that is executed after another function has finished executing. As we have studied earlier that functions are objects. Functions can take functions as arguments and can be returned by other functions. Functions that take another function as an argument are called higher-order functions. *So, the callback function can also be defined as any function that is passed as an argument is called a* ***callback function****.*

Why do we need Callbacks?

As we know, instead of waiting for a response, JavaScript will keep executing while listening for other events. Here is an example:

```
function fun1(){
   console.log("Code With Harry");
 }
function fun2(){
   console.log("JavaScript Tutorial");
 }
first();
second();
```

The function fun1 is executed first, and the function fun2 is executed second. JavaScript runs code from top to bottom. However, there are some cases that code

must run after something else happens and also not using a top-down approach.
Callbacks are used to make sure that a function is not going to execute before a task is completed but will execute right after the task has completed. It helps us develop asynchronous JavaScript code and keeps us safe from future 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.

How to create a Callback:-

Suppose we want to log a message to the console but it should be there after 5 seconds.

```
function myMessage (str) {
    setTimeout(() => {
        // script to download the picture here
        console.log(`Code With Harry`);
    }, 5000);
}
```

There is a built-in method called **"setTimeout"**, which calls a function after a given period of time. So here, the myMessage function is being called after 5 seconds. **(1 second = 1000 milliseconds)**. In other words, the myMessage function is being called after something happened (after 5 seconds passed for this example), but not before. So, the myMessage function is an example of a callback function.
Here is an example of callback functions in JavaScript:

```
function addition(x, y , callback){
setTimeout(() => {
document.write(`The sum of ${x} and ${y} is ${x+y}.`);
callback();
 }, 5000); }
 // display() function is called just after the ending of addition() function
function display(){
document.write('Numbers added!');
 }
 // Calling addition() function
addition(5,5,display);
```

Here are the two functions – addition(x, y, callback) and display(). Here addition() is called after 5 seconds with the display() function, i.e. passed in as the third argument to the addition function along with two numbers x and y. As a result, the addition() is invoked with 1, 2 and the display() which is the callback. The addition() prints the addition of the two numbers(x+y), and as soon as that is done, the callback function is executed.

Output:-

```
The sum of 5 and 5 is 10.
Numbers added!
```

Code ajax.html as described/written in the video

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXC
    <title>Document</title>
</head>
<body>
    <div class="container">
        <h1 id="heading" class='yourhead rhia is'> Welcome to Code With Harry</h1>
        <div id="myfirst" class="child red good" id="first">child 1

            <ul class="this" id='myul'>
                <li class="childul" id='fui'>this</li>
                <li class="childul">is</li>
                <li class="childul">a</li>
                <li class="childul">list </li>
                <li class="childul" id='lui'>of my dreams</li>
        </div>
        <div class="child">child 2</div>
        <div class="child red">child 3</div>
        <div class="child">child 4</div>
        <form action="none.html" method="post">
            <a href="//codewithharry.com">Go to Code With Harry</a>
            <br>
            <br>
            Search this website: <input type="text" name="Hello" id="">
            <button id="btn">Submit form</button>
            <!-- <input type="button" id='btn' value="submit"> -->
        </form>
```

```
    </div>
    <br>
    <div class="no">this is a dummy div1</div>
    <div class="no">this is a dummy div2</div>
    <div class="no">this is a dummy div3</div>
    <div class="container">
    <h1>Student list</h1>
    <ul id="students"></ul>

        <button id="myBtn" class="btn btn-primary">Your Button</button>
        <button class="btn btn-primary">Fetch Data</button>
        <div id="content"></div>
    </div>
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhz
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ
</body>
<!-- <script src="js/tut12.js"></script> -->
<!-- <script src="js/tut14.js"></script> -->
<!-- <script src="js/tut15.js"></script> -->
<!-- <script src="js/tut16.js"></script> -->
<!-- <script src="js/tut17.js"></script> -->
<!-- <script src="js/tut18.js"></script> -->
<!-- <script src="js/tut20.js"></script> -->
<!-- <script src="js/tut21.js"></script> -->
<!-- <script src="js/tut23.js"></script> -->
<!-- <script src="js/tut24.js"></script> -->
<!-- <script src="js/tut25.js"></script> -->
<!-- <script src="js/tut27.js"></script> -->
<!-- <script src="js/tut28.js"></script> -->
<!-- <script src="js/tut30.js"></script> -->
<!-- <script src="js/tut31.js"></script> -->
<!-- <script src="js/tut32.js"></script> -->
<!-- <script src="js/tut34.js"></script> -->
<!-- <script src="js/tut37.js"></script> -->
<!-- <script src="js/tut39.js"></script> -->
<!-- <script src="js/tut39b.js"></script> -->
<!-- <script src="js/tut41.js"></script> -->
```

```
<!-- <script src="js/tut43.js"></script> -->
<script src="js/tut44.js"></script>
</html>
```

Code ajax.js as described/written in the video

```
console.log("This is tutorial 37");

// Pretend that this response is coming from the server
const students = [
    {name: "harry", subject: "JavaScript"},
    {name: "Rohan", subject: "Machine Learning"}
]


function enrollStudent(student, callback){
    setTimeout(function() {
        students.push(student);
        console.log("Student has been enrolled");
        callback();
    }, 1000);
}

function getStudents(){
    setTimeout(function() {
        let str = "";
        students.forEach(function(student){
            str += `<li> ${student.name}</li>`
        });
        document.getElementById('students').innerHTML = str;
        console.log("Students have been fetched");
    }, 5000);
}

let newStudent = {name:"Sunny", subject:"Python"}
enrollStudent(newStudent, getStudents);
// getStudents();
```

Previous Next

CodeWithHarry