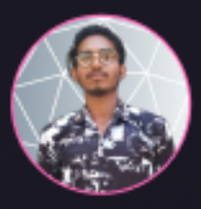
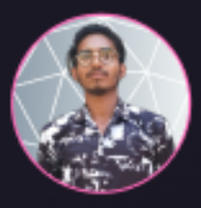


@Code.Clash

Callback

JavaScript



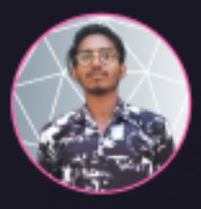
@Code.Clash

Hey Everyone 👋

JavaScript is everywhere. Millions of webpages are built on JS.

A few examples will help you understand the JavaScript Callback function in this post.

Do Like, save and Share This Post If You Found This Helpful.



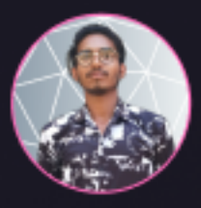
Function

A Function is a block of code that performs a certain task when called.

```
function greet(name) {
    console.log('Hi' + ' ' + name);
}

greet('Imtiyaz');
// Hi Imtiyaz
```

- In the above program, a string value is passed as an argument to the greet() function.

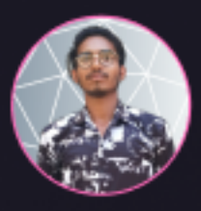
- In JavaScript, you can also pass a function as an argument to a function.



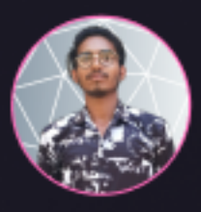
Callback Function

This function that is passed as an argument inside of another function is called a callback function.

```
// callback funtion
function myDisplayer(sum) {
  console.log("Total sum is: "+sum);
}
// function
function myCalculator(num, myCallback) {
  let sum = num + 5;
  myCallback(sum);
}
// passing myDisplayer funtion as an argument to the
// myCalculator funtion
myCalculator(8, myDisplayer);

/*  Output
    Total sum is: 13
*/
```



- In this example, the myDisplayer function is the callback.
- It is passed as an argument to the myCalculator function.
- When the myCalculator function is called, it calculate the sum of num and 5, and then calls the myDisplayer function, passing the sum as an argument.
- The myDisplayer function then log the sum.

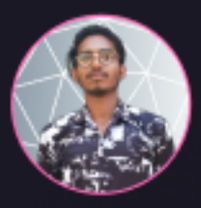


Benefits

1. Asynchrony: Callbacks allow you to execute a function after a certain task is completed, without blocking other code from executing in the meantime.

2. Reusability: Callback functions can be defined once and then passed as arguments to be used multiple times.

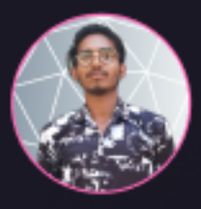
3. Flexibility: Callback functions can be added or removed at runtime, providing greater flexibility in your code.



@Code.Clash

Example Of Async. Task

```
// A function that simulates a time-consuming task
function heavyTask(callback) {
  setTimeout(() => {
    console.log("Heavy task finished!");
    callback();
  }, 2000);
}

// A function that is called after the heavy task is
// completed
function done() {
  console.log("Done!");
}

console.log("Starting...");

// Calling the heavy task and passing the "done" function
// as a callback
heavyTask(done);

console.log("Continuing with other work...");

// Output:
// Starting...
// Continuing with other work...
// Heavy task finished!
// Done!
```



@CODE.CLASH

Here is an Example of using a callback function to perform an asynchronous task in JavaScript.

In this example, heavyTask simulates a time-consuming task that takes 2 seconds to complete.

The done function is passed as a callback to heavyTask, and is executed when the heavy task is finished.

Meanwhile, the rest of the code continues to execute, allowing for other tasks to be performed while the heavy task is in progress.

@Code.Clash

Check Out My Profile For Such Excellent Posts On Web Development.

Imtiyaz Nandasaniya