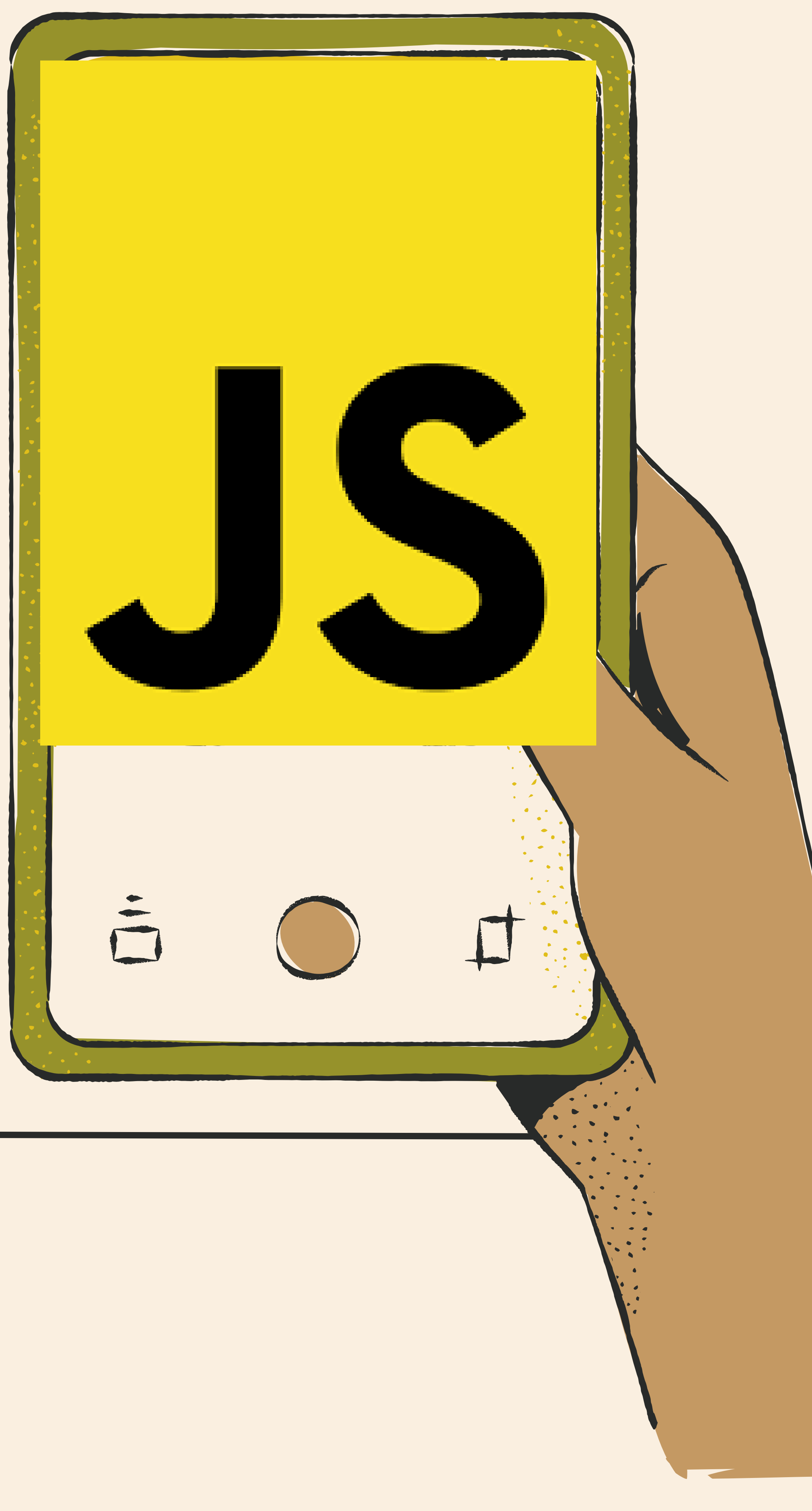
Callback and higher order functions



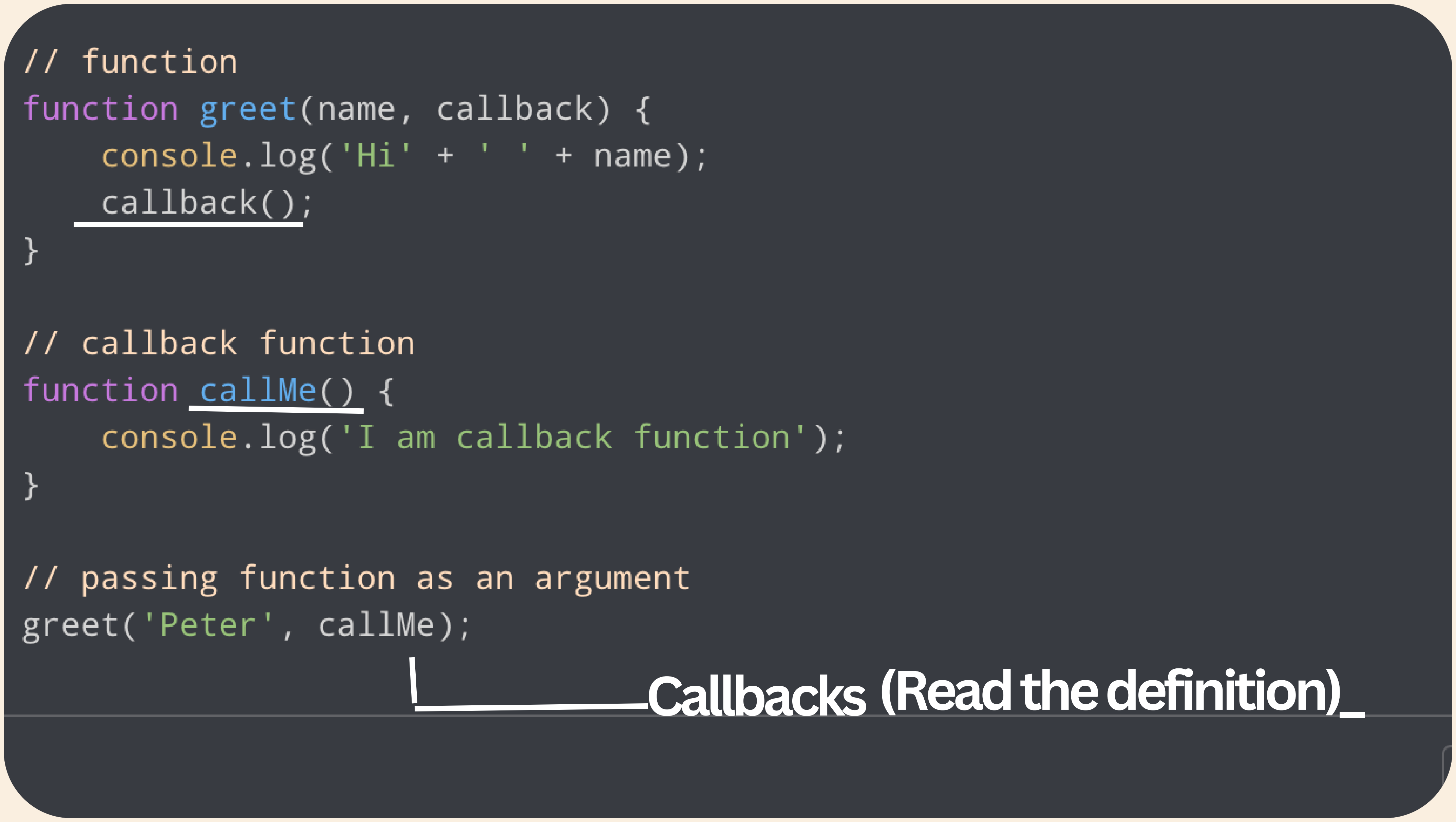
Give a space to learn

Call Back functions

A callback function is a function that is passed to another function with the expectation that the other function will call it.

how callbacks are called?
why callbacks are needed?

How callbacks are called?



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

// callback function
function callMe() {
    console.log('I am callback function');
}

// passing function as an argument
greet('Peter', callMe);
```

Callbacks (Read the definition)_

Explanation

- The greet() function is called with arguments , where function callMe is passed as an argument to another function .
- now greet() function will be executed and after execution, callback function will be triggered (callMe())
- Now it will execute callMe() function which is a call back function in our case.

you understood how to write callbacks but

Do you know why we need callbacks?

Why we need callbacks?

- JavaScript runs code sequentially in top-down order. However, there are some cases that code runs NON sequentially. This is called asynchronous programming.

Non sequential code example

```
//  program that shows the delay in execution

function greet() {
    console.log('Hello world');
}

function sayName(name) {
    console.log('Hello' + ' ' + name);
}

// calling the function
setTimeout(greet, 2000);
sayName('John');
```

Output:

```
Hello John
Hello world
```

above code is an example of non sequential run. As set timeout which is async , has changed top-down approach of code

In few cases that's not a problem ,

- But think of a scenario where two api calls has to be made based on each other.

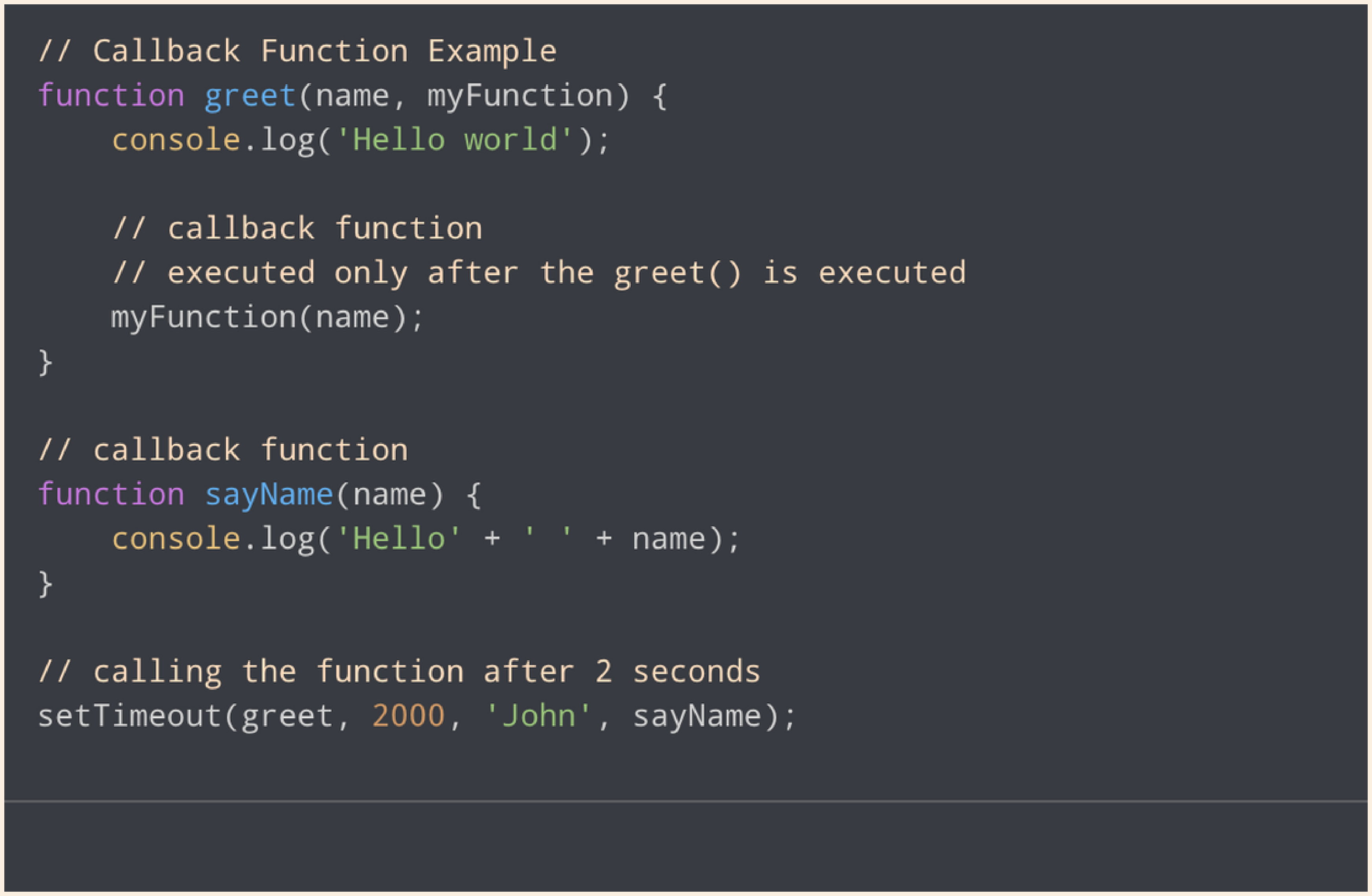
- api1 should be called first and api2 follows as there is connection between two api's

- Here due to delay response, api2 called first and api1 is called second . Here probably issues arise in application .

- To solve this we use Callbacks.

- Callbacks make sure that a function is not going to run before a task is completed .

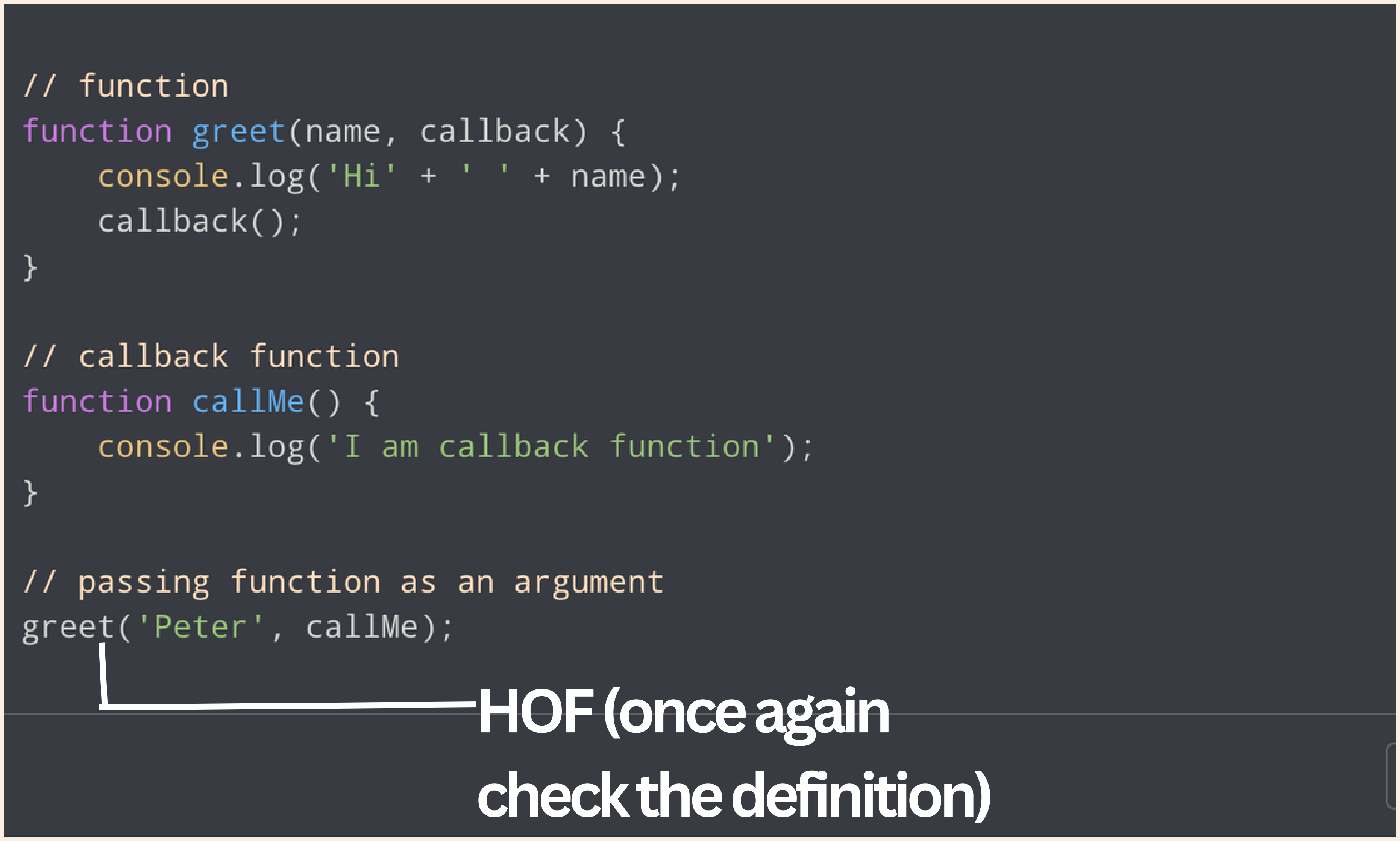
- But will run right after the task has completed. It helps us develop asynchronous JavaScript code and keeps us safe from problems and errors.



```javascript
// Callback Function Example
function greet(name, myFunction) {
    console.log('Hello world');

    // callback function
    // executed only after the greet() is executed
    myFunction(name);
}

// callback function
function sayName(name) {
    console.log('Hello' + ' ' + name);
}

// calling the function after 2 seconds
setTimeout(greet, 2000, 'John', sayName);
```

Higher order functions

A <u>higher-order function</u> is a function that takes another function(s) as an argument(s) and/or returns a function .



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

// callback function
function callMe() {
    console.log('I am callback function');
}

// passing function as an argument
greet('Peter', callMe);
```

HOF (once again check the definition)

Why we need higher order functions?

We will understand the problem solved by HOF , for better understanding of HOF

```javascript
function filterEven(arr) {
  const filteredArr = [];
  for (let i = 0; i < arr.length; i++) {
    if (arr[i] % 2 == 0) {
      filteredArr.push(arr[i]);
    }
  }
  return filteredArr;
}
console.log(filterEven(arr));

// Output:
// [ 2, 4, 6, 8, 10 ]
```

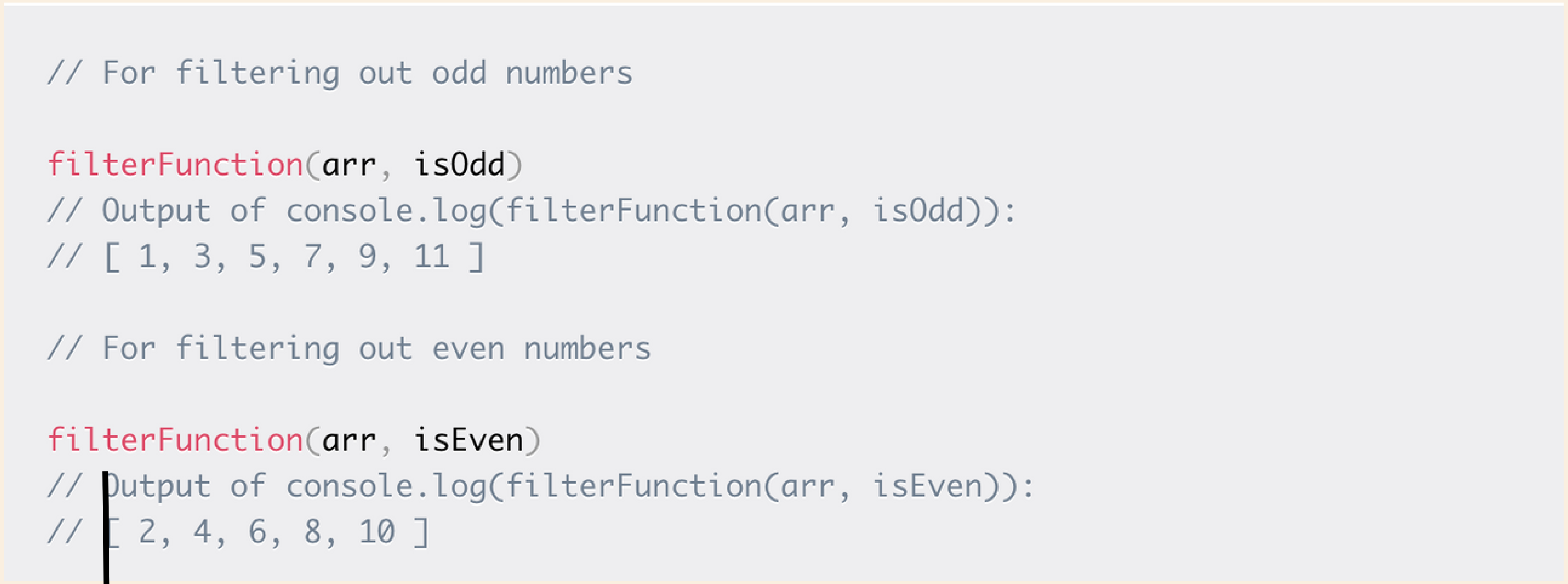
```javascript
const arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11];

function filterOdd(arr) {
  const filteredArr = [];
  for (let i = 0; i < arr.length; i++) {
    if (arr[i] % 2 !== 0) {
      filteredArr.push(arr[i]);
    }
  }
  return filteredArr;
}
console.log(filterOdd(arr));

// Output:
// [ 1, 3, 5, 7, 9, 11 ]
```

We see lot of similar code to get even and odd numbers

Here comes in Higher order functions where we can write similar code function and call even and odd number functions from the similar function

```javascript
function filterFunction(arr, callback) {
  const filteredArr = [];
  for (let i = 0; i < arr.length; i++) {
    callback(arr[i]) ? filteredArr.push(arr[i]) : null;
  }
  return filteredArr;
}
```

```javascript
// Function containing logic for filtering out odd numbers

function isOdd(x) {
  return x % 2 != 0;
}

// Function containing logic for filtering out even numbers

function isEven(x) {
  return x % 2 === 0;
}
```



```
// For filtering out odd numbers

filterFunction(arr, isOdd)
// Output of console.log(filterFunction(arr, isOdd)):
// [ 1, 3, 5, 7, 9, 11 ]

// For filtering out even numbers

filterFunction(arr, isEven)
// Output of console.log(filterFunction(arr, isEven)):
// [ 2, 4, 6, 8, 10 ]
```

Here filterfunction is a function which accepts another function (isEven) to perform the task and this are called HOF

NOTE: To be clear all higher oder functions have callbacks but not all callback functions are not higher order functions