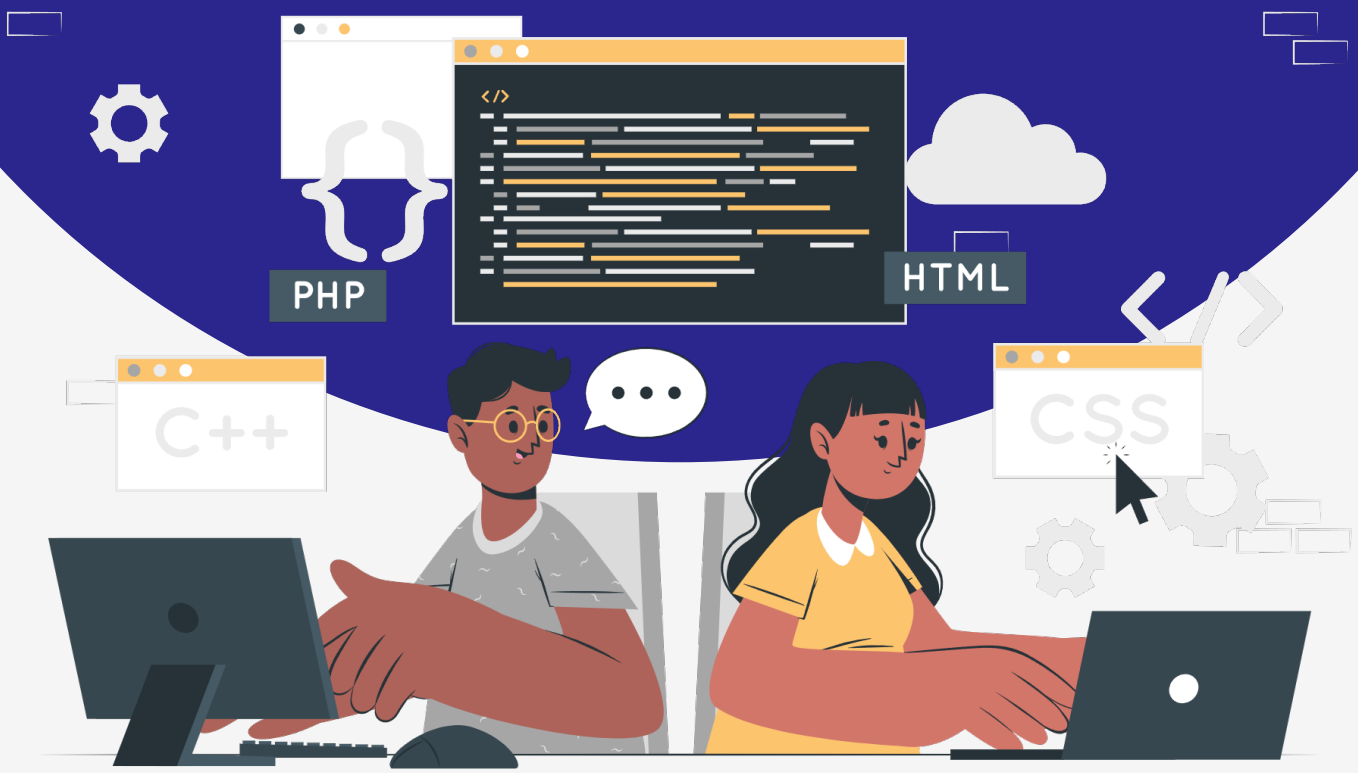
Lesson:

Callback, Returning function



PW SKILLS

Topics

1. What is callback
2. What is callback hell?
3. Returning function

What is a callback?

A callback function is a function that is passed as an argument to another function and is intended to be called by that function at a later time or in response to a specific event.
This is one of the ways we can handle asynchronous operations

EXAMPLE

```
function hiFunction(name, callback) {
  console.log('Hi' + ' ' + name + ' I am the actual
function');
  callback();
}

// callback function
function exampleFunction() {
  console.log('I come from the callback function');
}

// passing function as an argument
hiFunction('Folks', exampleFunction);
```

OUTPUT:

```
Hi Folks I am the actual function
I come from the callback function
```

In this example, we have created two functions viz hiFunction() and exampleFunction(). While calling the hiFunction() function, two arguments (a string value name and a function) are passed. This is the function from where the execution starts.
The exampleFunction() function is a callback function and starts it execution on encountering the callback()

The benefit of using a callback function is that you can wait for the result of a previous function call and before executing another function call.

Callback Hell

It is nested callbacks stacked below one another forming a pyramid kind of structure. Every callback here, waits for the previous callback, thereby affecting the readability and maintainability of the code.

Example

```
getProduct(18, (user) => {
  console.log("Get Products", user);
  getUserInfo(user.username, (name) => {
    console.log(name);
    getAddress(name, (item) => {
      console.log(item);
      // this goes on and on...
    }
})
```

In the example, getUserInfo takes in an argument which is dependent or needs to be extracted from the result produced by getProducts which is inside user. The same dependency can be observed with getAddress also. This is callback hell.

There are a few ways to avoid these callback hells which we will discuss in the lectures ahead.

Returning function

As we already know, when a return statement is encountered in a function body, the execution of the function stops. If specified, a given value is returned to the function caller.

Till now we have seen examples of functions returning/not returning a value. However, in higher-order function concepts, we saw that functions can also return a function.

Let us look at the example below for a better understanding:

Example 1:

Demonstrates a simple function that returns a function.

```
multiplyBy7() {
  return function(x) { return x * 7; };
}

const changedValue = multiplyBy7();
console.log(changedValue(25));
```

OUTPUT:

```
175
```

Here, as you can observe, the function multiplyBy7() returns another function that in itself does some operation before returning the result.

Example 2:

The example below shows an implementation where both the parent function and returned functions are taking parameters.

```
function myAdder(n1) {
  return function(n2) { return n1 + n2;};
}
console.log( myAdder(5)(3) );
```

OUTPUT:

```
8
```

Here, you can see that the function myAdder() takes a parameter n1 and returns the addition of n1 and n2 (n2 being a parameter passed to the function return function returned by myAdder().

Question 1: Explain what a callback function is in JavaScript and provide an example of its usage.

Answer: A callback function in JavaScript is a function that is passed as an argument to another function and is executed after the completion of a specific task. It allows for asynchronous programming and is commonly used in scenarios like event handling or making AJAX requests.

Question 2: Explain what is meant by "Callback Hell" in JavaScript.

Answer: Callback Hell, also known as the "Pyramid of Doom," refers to the situation where multiple nested callback functions create an indentation pyramid that makes the code hard to read and maintain. This commonly occurs in asynchronous JavaScript operations.

```
getUserData((userData) => {
  getProfile(userData.id, (profile) => {
    getPosts(userData.id, (posts) => {
      // More nested callbacks...
    });
  });
});
```

PW SKILLS

THANK YOU !