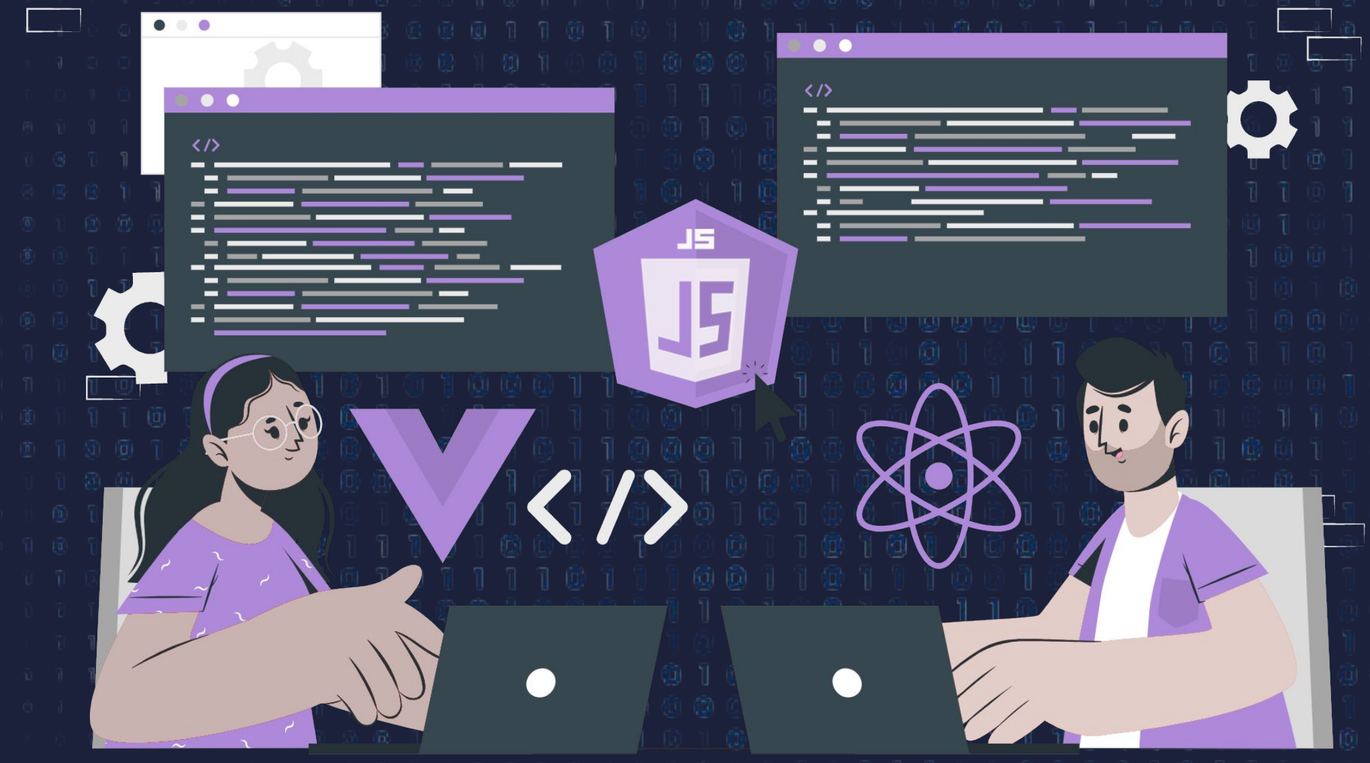
PW SKILLS

Callback



Lecture CheckList

1. Introduction.
2. What is the need for a callback?
3. Synchronous callbacks.
4. Asynchronous callbacks.
5. Nested callbacks.
6. Callback Hell.

Introduction

A callback is a function that is passed as an argument to another function and is executed once the main function has finished executing. The purpose of a callback is to allow a program to perform actions asynchronously. This is especially useful when dealing with tasks that take a long time to complete or when you want to execute multiple tasks at the same time.

Callbacks can be used in many different scenarios in programming, such as event handling, network requests, and user interactions. They provide a flexible and efficient way to execute code after a certain task has been completed, and can be a powerful tool for writing clean and efficient code. However, it's important to use callbacks carefully, as they can quickly become difficult to manage and lead to what is known as "callback hell."

We will be looking into all of these concepts in this lecture.

What is the need for a callback?

Javascript is a single-threaded language, which means, it executes the code sequentially one line after another. However, there are some cases where multiple tasks or operations can be executed concurrently without waiting for each other to complete them. This is asynchronous programming.

- Handling Asynchronous Operations.
- Event Handling.
- Fetching Data.

Synchronous callbacks.

In synchronous programming, tasks or operations are executed one at a time, in a sequential manner, and each task must complete before the next one can begin. In other words, the program waits for each task to finish before moving on to the next one.

Asynchronous callbacks.

There are some cases where multiple tasks or operations can be executed concurrently without waiting for each other to complete them. This is asynchronous programming.

Nested Callbacks

Nested callbacks are a common pattern in asynchronous programming, where a callback function is called inside another callback function.

This pattern is used when you need to execute a series of asynchronous tasks, where each task depends on the output of the previous task.

Callback Hell

Callback hell is a situation in asynchronous programming where multiple levels of nested callbacks make the code difficult to read, understand, and maintain. This can occur when you have to execute a series of asynchronous tasks, where each task depends on the output of the previous task, and you need to pass the results of each task to the next one.

When you use a lot of nested callbacks, the code can become difficult to read and maintain. Additionally, if there are any errors in the code, it can be difficult to pinpoint where the errors are occurring.

To avoid callback hell, there are several techniques you can use in JavaScript, such as Promises and async/await functions. We will be looking into them in further lectures. These techniques can help you write cleaner, more maintainable code when dealing with complex asynchronous operations.

THANK YOU