In Node.js, callbacks are a common pattern used to handle asynchronous operations. A callback is a function that is passed as an argument to another function and gets invoked once the asynchronous operation completes or encounters an error. Let's dive into a detailed explanation with a coding example:

```
// Asynchronous function that takes a callback
function fetchData(callback) {
  // Simulating an asynchronous operation (e.g., fetching data
from a database or making an API request)
  setTimeout(() => {
    const data = 'This is the fetched data';
    const error = null;
    callback(error, data);
  }, 2000);
}

// Callback function to handle the fetched data
function handleData(error, data) {
  if (error) {
    console.error('Error:', error);
  } else {
```

```
    console.log('Data:', data);
  }
}

// Call the asynchronous function with the callback
fetchData(handleData);
```

In this example, we have an fetchData function that simulates an asynchronous operation using setTimeout. After a delay of 2 seconds, it invokes the callback function with two arguments: error and data.
The handleData function is the callback function we pass to fetchData. It takes two parameters: error and data. Inside handleData, we can perform actions based on whether an error occurred or the data was successfully fetched.
Finally, we call the fetchData function and pass the handleData callback as an argument. When the asynchronous operation completes, the fetchData function invokes the handleData callback, passing the error (if any) and the fetched data as arguments.
If an error occurs during the asynchronous operation, the handleData callback prints an error message to the console. Otherwise, it logs the fetched data.
This pattern allows us to handle the result of an asynchronous operation without blocking the execution of other code. The callback function acts as a notification mechanism, executing the appropriate code when the asynchronous operation completes.
Callbacks are a foundational concept in Node.js, but they can lead to callback hell or the "pyramid of doom" when dealing with

multiple asynchronous operations. To mitigate this issue, there are alternative patterns like Promises and async/await, which provide more readable and maintainable code. However, understanding callbacks is still important as many Node.js libraries and APIs use them extensively.