

Day 18 - Asynchronous Programming in Node.js

Challenge 7: Callbacks

User Story

As a developer, I want to read a file and then display a confirmation message once it's done — using callbacks.

Code Snippet and Output

The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar with a folder named 'DAY18CODINGSOLUTION_ASY...' containing 'challenge7_callback.js' and 'data.txt'. The main editor area displays the following code:

```
JS challenge7_callback.js X data.txt
JS challenge7_callback.js > ...
1 const fs = require('fs');
2
3 console.log('Starting read (callbacks)...');
4
5 fs.readFile('data.txt', 'utf8', (err, data) => {
6   if (err) {
7     console.error('Error reading file:', err);
8     return;
9   }
10
11   // Show content (to demonstrate asynchronous completion)
12   console.log('File content:\n', data);
13
14   // Bonus: artificial delay before confirmation
15   setTimeout(() => {
16     console.log('Read operation completed');
17   }, 1000); // 1 second delay
18 });
19
20 // This log demonstrates that fs.readFile is non-blocking
21 console.log('This line runs before read callback finishes (non-blocking).');
```

At the bottom, the terminal window shows the execution of the script and its output:

```
PS D:\Wipro Training\OlympusAssignments\Day18CodingSolution_AsyncProgrammingInNode> node challenge7_callback.js
Starting read (callbacks)...
This line runs before read callback finishes (non-blocking).
File content:
Hello from data.txt!
This is example content.
Read operation completed
PS D:\Wipro Training\OlympusAssignments\Day18CodingSolution_AsyncProgrammingInNode>
```

Challenge 8: Promises

User Story

As a developer, I want to chain multiple async operations (read file → write to another file) using Promises.

Code Snippet and Output

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows a folder named "DAY18CO..." containing files: "challenge7_callback.js", "challenge8_promises.js" (selected), "data.txt", "input.txt", and "output.txt".
- Code Editor (Center):** Displays the content of "challenge8_promises.js":

```
JS challenge8_promises.js X input.txt
JS challenge8_promises.js > ...
1 const fs = require('fs').promises;
2
3 console.log('Starting promise chain...');
4
5 fs.readFile('input.txt', 'utf8')
6   .then((data) => {
7     // return the write Promise so we can chain
8     return fs.writeFile('output.txt', data);
9   })
10  .then(() => {
11    console.log('File copied successfully!');
12  })
13  .catch((err) => {
14    console.error('Error during file copy (promises):', err);
15  });

```

- Terminal (Bottom):** Shows the command "node challenge8_promises.js" being run, followed by the output:

```
PS D:\Wipro Training\OlympusAssignments\Day18CodingSolution_AsyncronousProgrammingInNode> node challenge8_promises.js
Starting promise chain...
File copied successfully!
PS D:\Wipro Training\OlympusAssignments\Day18CodingSolution_AsyncronousProgrammingInNode>
```

Challenge 9: Async/Await

User Story

As a developer, I want cleaner syntax for asynchronous operations using modern JavaScript.

Code Snippet and Output

The screenshot shows the Visual Studio Code interface. The left sidebar displays a file tree with files: challenge7_callback.js, challenge8_promises.js, challenge9_asyncawait.js (selected), data.txt, input.txt, output_async.txt, and output.txt. The main editor area contains the following code:

```
const fs = require('fs').promises;
async function copyFileWithDelay() {
  try {
    console.log('Starting async/await copy...');
    const data = await fs.readFile('input.txt', 'utf8');
    // Bonus: simulate slow operation
    await new Promise((res) => setTimeout(res, 1000)); // 1 second
    await fs.writeFile('output_async.txt', data);
    console.log('File copied successfully (async/await)!');
  } catch (err) {
    console.error('Error during async/await file copy:', err);
  }
}
copyFileWithDelay();
```

The bottom right panel shows the terminal output:

```
PS D:\Wipro Training\OlympusAssignments\Day18CodingSolution_AsyncronousProgrammingInNode> node challenge9_asyncawait.js
Starting async/await copy...
File copied successfully (async/await)!
```