

## Day 17 - Introduction to Node.js & Basics

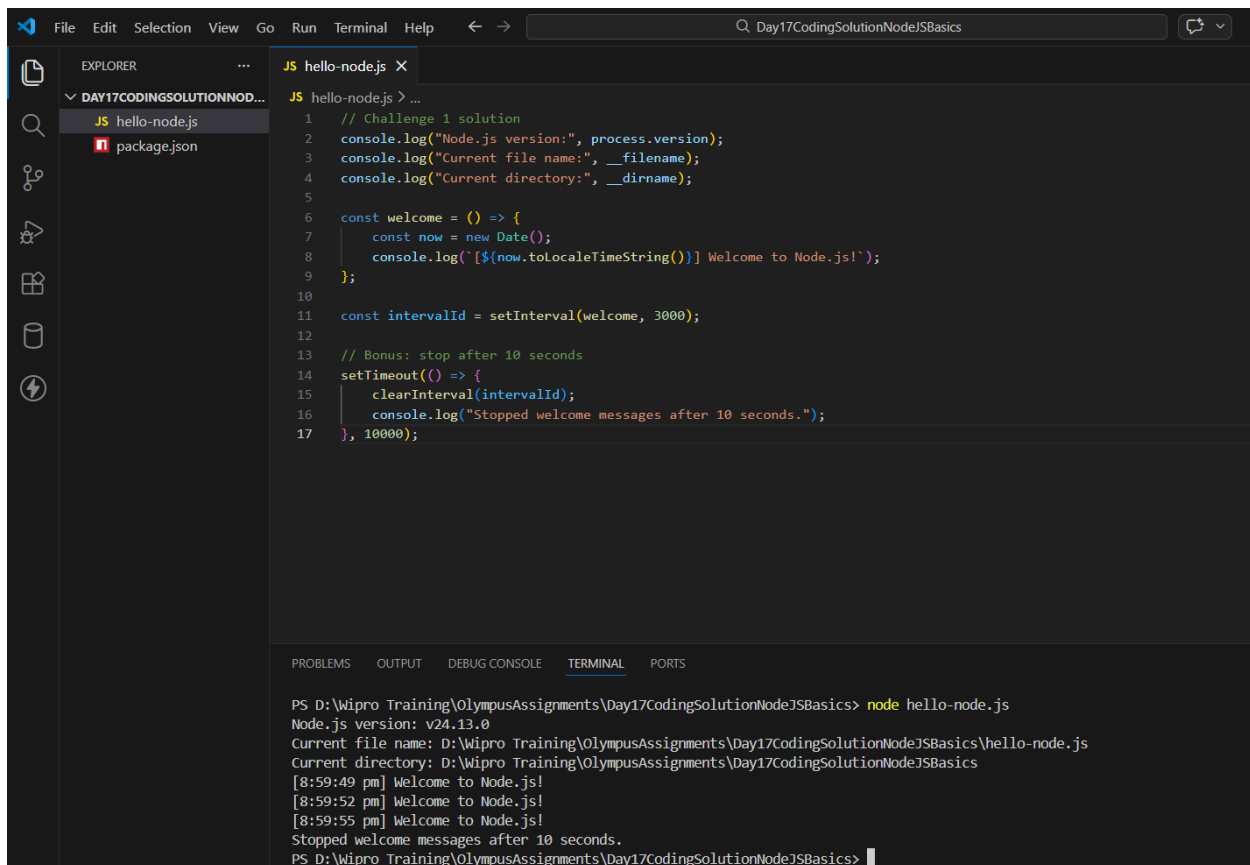
This report summarizes the three coding challenges completed as part of the Day 16 Node.js assignment. Each challenge demonstrates a key Node.js concept such as global objects, timers, CLI arguments, and external npm packages.

### Challenge 1: Hello Node

This challenge demonstrates Node.js fundamentals. The program prints the Node version, file name, and directory path. It uses `setInterval()` to display a welcome message every 3 seconds and `clearInterval()` to stop the execution after 10 seconds.

#### Approach:

1. Used built-in global objects (`__filename`, `__dirname`).
2. Implemented `setInterval()` for periodic message display.
3. Used `setTimeout()` to clear the interval automatically.



The screenshot shows the Visual Studio Code editor interface. The Explorer panel on the left shows a project named 'DAY17CODINGSOLUTIONNOD...' with two files: 'hello-node.js' and 'package.json'. The main editor window displays the code for 'hello-node.js'. The code is as follows:

```
1 // Challenge 1 solution
2 console.log("Node.js version:", process.version);
3 console.log("Current file name:", __filename);
4 console.log("Current directory:", __dirname);
5
6 const welcome = () => {
7   const now = new Date();
8   console.log(`[${now.toLocaleTimeString()}] Welcome to Node.js!`);
9 };
10
11 const intervalId = setInterval(welcome, 3000);
12
13 // Bonus: stop after 10 seconds
14 setTimeout(() => {
15   clearInterval(intervalId);
16   console.log("Stopped welcome messages after 10 seconds.");
17 }, 10000);
```

The Terminal panel at the bottom shows the output of running the command `node hello-node.js` in the directory `D:\Wipro Training\OlympusAssignments\Day17CodingSolutionNodeJSBasics`. The output is:

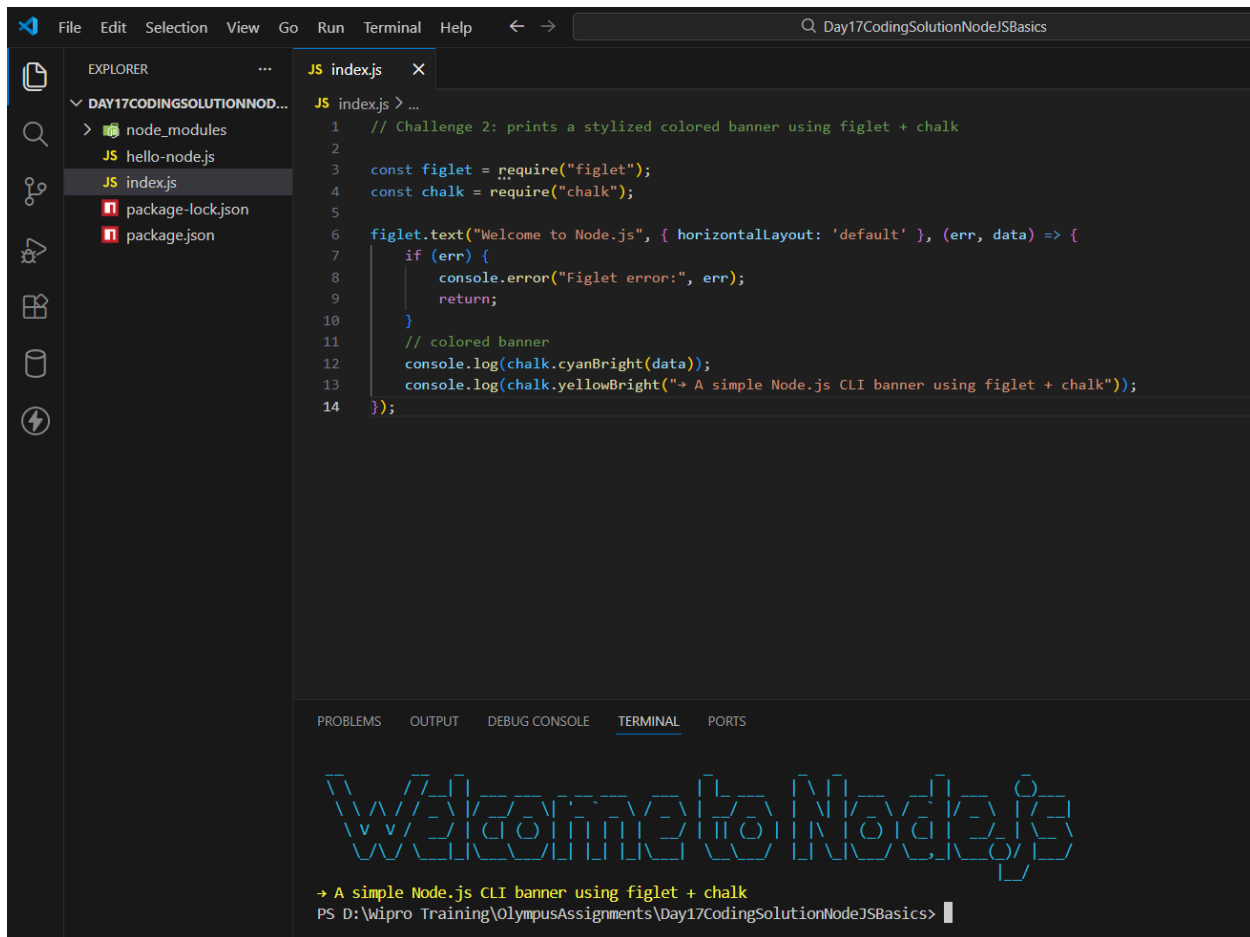
```
PS D:\Wipro Training\OlympusAssignments\Day17CodingSolutionNodeJSBasics> node hello-node.js
Node.js version: v24.13.0
Current file name: D:\Wipro Training\OlympusAssignments\Day17CodingSolutionNodeJSBasics\hello-node.js
Current directory: D:\Wipro Training\OlympusAssignments\Day17CodingSolutionNodeJSBasics
[8:59:49 pm] Welcome to Node.js!
[8:59:52 pm] Welcome to Node.js!
[8:59:55 pm] Welcome to Node.js!
Stopped welcome messages after 10 seconds.
PS D:\Wipro Training\OlympusAssignments\Day17CodingSolutionNodeJSBasics>
```

## Challenge 2: Colorful Banner App

This challenge prints a colorful banner on the terminal using the figlet and chalk libraries. It demonstrates how to install and use external npm packages and how to create a custom npm start script.

### Approach:

1. Imported figlet to generate stylized ASCII text.
2. Used chalk to apply colors to terminal output.
3. Configured an npm start script in package.json to execute index.js.



The screenshot shows the Visual Studio Code editor with a project named 'DAY17CODINGSOLUTIONNOD...'. The Explorer sidebar on the left shows the file structure with 'index.js' selected. The main editor window displays the code for 'index.js', which uses 'figlet' to generate ASCII art and 'chalk' to color it. The code is as follows:

```
1 // Challenge 2: prints a stylized colored banner using figlet + chalk
2
3 const figlet = require("figlet");
4 const chalk = require("chalk");
5
6 figlet.text("Welcome to Node.js", { horizontalLayout: 'default' }, (err, data) => {
7   if (err) {
8     console.error("Figlet error:", err);
9     return;
10  }
11  // colored banner
12  console.log(chalk.cyanBright(data));
13  console.log(chalk.yellowBright("→ A simple Node.js CLI banner using figlet + chalk"));
14 });
```

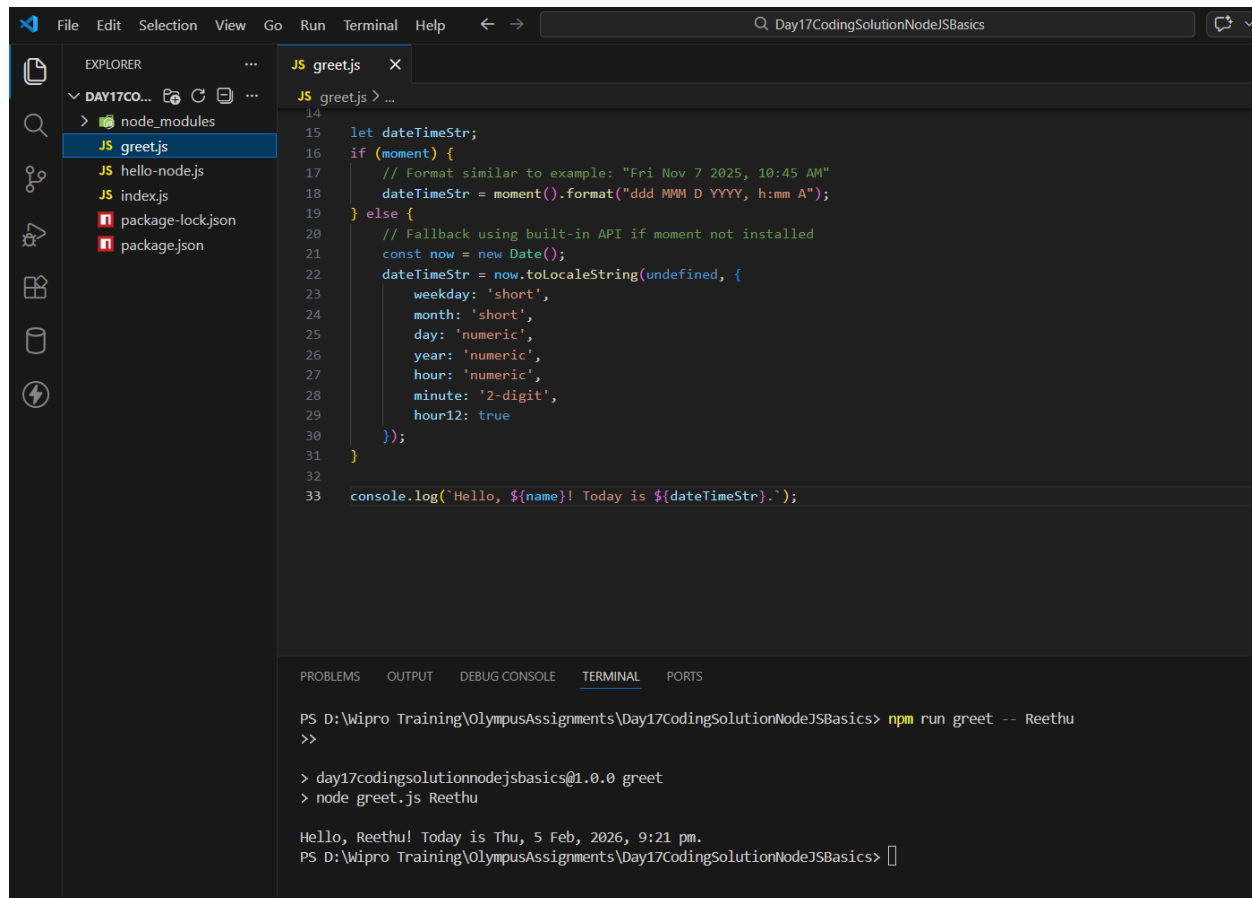
Below the code editor, the TERMINAL panel is active, showing the output of the application. It displays the ASCII art 'Welcome to Node.js' in cyan, followed by the message '→ A simple Node.js CLI banner using figlet + chalk' in yellow. The terminal prompt is 'PS D:\Wipro Training\OlympusAssignments\Day17CodingSolutionNodeJSBasics>'.

### Challenge 3: CLI Greeting App

This challenge takes a user's name as a command-line argument and greets the user along with the current date and time. It uses the `process.argv` array to capture input and the `moment` library to format timestamps.

#### Approach:

1. Retrieved user input using `process.argv`.
2. Formatted date and time using `moment()`.
3. Provided a fallback using JavaScript's built-in `Date` object if `moment` was not available.



The screenshot shows a Visual Studio Code editor with a file explorer on the left and a code editor in the center. The file explorer shows a project named 'DAY17CO...' with a 'node\_modules' directory and files 'greet.js', 'hello-node.js', 'index.js', 'package-lock.json', and 'package.json'. The code editor shows the content of 'greet.js'.

```
14
15 let dateTimeStr;
16 if (moment) {
17   // Format similar to example: "Fri Nov 7 2025, 10:45 AM"
18   dateTimeStr = moment().format("ddd MMM D YYYY, h:mm A");
19 } else {
20   // Fallback using built-in API if moment not installed
21   const now = new Date();
22   dateTimeStr = now.toLocaleString(undefiend, {
23     weekday: 'short',
24     month: 'short',
25     day: 'numeric',
26     year: 'numeric',
27     hour: 'numeric',
28     minute: '2-digit',
29     hour12: true
30   });
31 }
32
33 console.log(`Hello, ${name}! Today is ${dateTimeStr}.`);
```

Below the code editor, the terminal window shows the execution of the script:

```
PS D:\Wipro Training\OlympusAssignments\Day17CodingSolutionNodeJSBasics> npm run greet -- Reethu
>>
> day17codingsolutionnodejsbasics@1.0.0 greet
> node greet.js Reethu

Hello, Reethu! Today is Thu, 5 Feb, 2026, 9:21 pm.
PS D:\Wipro Training\OlympusAssignments\Day17CodingSolutionNodeJSBasics>
```

### Conclusion

All three challenges successfully demonstrate the fundamentals of Node.js, including working with global objects, timers, npm packages, and command-line interaction. This assignment provided practical experience in executing and managing Node.js scripts.