Node.js - Day 4: NPM Basics + Debugging + Mini-project: Command-line Task Manager

■ Learning Objectives

- Understand what npm is and its purpose.
- Initialize and manage Node.js projects with package.json.
- Install, use, and remove npm packages.
- Practice debugging Node.js scripts.
- Build a command-line task manager using Node.js.

■ Session 1 (45 min) — NPM Basics + Debugging

1. What is NPM?
NPM (Node Package Manager) is the tool used to install and manage JavaScript libraries and dependencies.
It allows developers to:
• Install packages from the npm registry
• Manage versions and dependencies
• Run project scripts
npm init -y npm install package-name npm uninstall package-name

■ 2. Example: Installing Packages

Install chalk:
npm install chalk Use it in code:
const chalk = require('chalk'); console.log(chalk.green('Task added successfully!')); **Install nodemon (global):**
npm install -g nodemon Run automatically:
nodemon index.js

■ 3. Debugging Techniques

Console Logs:
Use strategically to inspect variables.
console.log('Current tasks:', tasks);
Using Debugger:
Add keyword:
debugger; Run in terminal:
node inspect index.js Commands:
• n → next line
• c → continue

• repl → inspect variables
VS Code Debugging:
1. Open project in VS Code.
2. Add a debug configuration for Node.js.
3. Set breakpoints and run the debugger.

■ 4. Exercise

Create a file named **color-test.js**, install **chalk**, and print two messages in different colors. const chalk = require('chalk'); console.log(chalk.green('Success message!')); console.log(chalk.red('Error message!'));

■ Session 2 (45 min) — Mini Project: Command-Line Task Manager

Goal:
Build a simple CLI app to manage tasks (add, list, delete).
Tasks are saved in a JSON file for persistence.

■ Step 1 — Setup

mkdir task-manager cd task-manager npm init -y npm install chalk Create files:
• index.js
• tasks.json (with an empty array: [])

■ Step 2 — Write the Logic

const fs = require('fs'); const chalk = require('chalk'); const command = process.argv[2]; const task = process.argv[3]; const loadTasks = () => { try { return JSON.parse(fs.readFileSync('tasks.json', 'utf8')); } catch { return []; } }; const saveTasks = (tasks) => { fs.writeFileSync('tasks.json', JSON.stringify(tasks, null, 2)); }; if (command === 'add') { const tasks = loadTasks(); tasks.push(task); saveTasks(tasks); console.log(chalk.green(`Task added: ${task}`)); } else if (command === 'list') { const tasks = loadTasks(); console.log(chalk.yellow('Your Tasks:')); tasks.forEach((t, i) => console.log(`${i + 1}. ${t}`)); } else if (command === 'delete') { const index = parseInt(task) - 1; const tasks = loadTasks(); if (index >= 0 && index < tasks.length) { const removed = tasks.splice(index, 1); saveTasks(tasks); console.log(chalk.red(`Deleted: ${removed}`)); } else { console.log(chalk.red('Invalid task number.')); } } else { console.log(chalk.cyan('Usage:')); console.log('node index.js add "Task name"'); console.log('node index.js list'); console.log('node index.js delete 1'); }

■ Step 3 — Run Commands

node index.js add "Finish Node.js class" node index.js add "Go to gym" node index.js list node index.js delete 1 Example Output:
Task added: Finish Node.js class Task added: Go to gym Your Tasks: 1. Finish Node.js class 2. Go to gym Deleted: Finish Node.js class

■ Debugging Challenge

1. Introduce an intentional error (like a wrong file name).
2. Use console.log() and debugger to trace and fix it.
3. Verify task deletion and file saving logic.

■ Recap Discussion

- What is npm and why is it important?
- How does package.json help manage projects?
- How can you debug Node.js code efficiently?
- What features would you add next to the Task Manager?

■ Homework

Enhance the Task Manager app:
- Add a 'mark as done' feature.
- Display completed tasks in gray using chalk.
Example output:
1. [■] Finish project 2. [ ] Study for test