24120111119, Computer Science, Year 2

Technologies: Node.js, TypeScript, Inquirer.js, Chalk, File System (fs), ES Modules

Project Details

Designed to let consumers create, update, remove, search, tag, and analyse tasks using a basic interactive terminal program, this project is a Command-Line Interface (CLI) Task Manager. The effort developed from simple JavaScript into a more solid TypeScript application.

Things I learned:

1. Node.js Fundamentals
    - Making a CLI with inquirer for user input
    - adding colour and formatting with chalk
    - modularising code into reusable commands (add, remove, etc.) into command files

2. File System Management
    - Reading and writing JSON files via an FS module.
    - File path handling using path and URL modules guarantees data durability.

3. TypeScript
    - Incorporating type safety via TaskProps.
    - Converting .js files to.ts for easier upkeep.
    - Knowing the differences between CommonJs and ES Modules (import).

4. Asynchronous Development in Programming
    - Asynchronous file operations and user input managed with async/await.
    - Using try/catch, handle mistakes precisely.

5. Project Structure & Design
    - Organizing code into commands/ for greater modularity.
    - Using index.ts or index.js as the primary entry point.
    - Creating reusable tools for statistics, storage, and task filtering.

6. Advanced CLI Features
    - Filtering and tagging tasks.
    - Exporting tasks to CSV.
    - Examining task statistics (such as total, completed, overdue).
    - Searching jobs using keyword matching

Issues encountered and resolved

- Correct application of JavaScript extensions in import statements during compilation.
- Controlling variations between production builds (tsc) and ts-node development.
- TypeScript module resolution issues.
- Global as opposed to local implementation of nodemon, typeScript, and ts-node utilities.

This project was a practical introduction to creating a real-world Node.js CLI utility using contemporary development techniques such modular design, static typing, and user-friendly UX.

It confirmed:

Core JavaScript and TypeScript grammar

Correct error handling

Collaborating with asynchronous files I/O.

Strong typing and modularity provide several advantages.