**Day 42**





**“Web Development + Security”**

**Backend, Node.js & npm:**

**What is Backend? (Basic Definition)**

* The backend is the server-side part of a web application.
* It handles logic, databases, authentication, APIs, and communication with the frontend.
* Example: When you submit a form, backend code processes and stores the data in a database.

Frontend (client) → sends request → Backend (server) → sends response back

**What is Node.js?**

* Node.js is a JavaScript runtime environment that lets you run JS outside the browser (like on your computer or server).
* It uses Google Chrome’s V8 engine internally.
* With Node.js, you can build:
  + Web servers
  + APIs
  + Command-line tools
  + Real-time apps (like chat or streaming)

**What is npm (Node Package Manager)?**

* npm is the default package manager for Node.js.
* It lets you install, update, or manage external libraries (modules) for your project.
* Every Node project uses a file called package.json — which stores project details and dependencies.

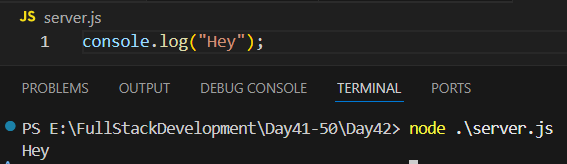
**Difference between running JS in the browser console vs. in Node.js (terminal)**:

**Environment Difference**

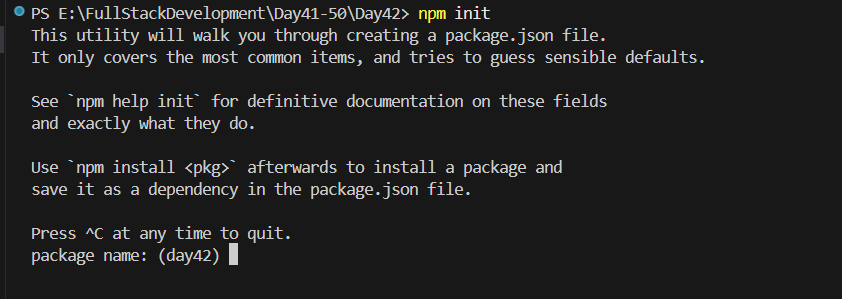
| **Feature** | **Browser Console (Frontend JS)** | **Node.js (Backend JS)** |
| --- | --- | --- |
| **Environment** | Runs inside a web browser (like Chrome) | Runs inside the Node.js runtime on your computer/server |
| **Scope** | Attached to the window object | Attached to the global object |
| **APIs available** | Has access to DOM, window, document, alert() | Has access to file system (fs), path, http, process) |
| **Use case** | Used for frontend logic, testing, UI manipulation | Used for backend logic, servers, APIs, file handling |

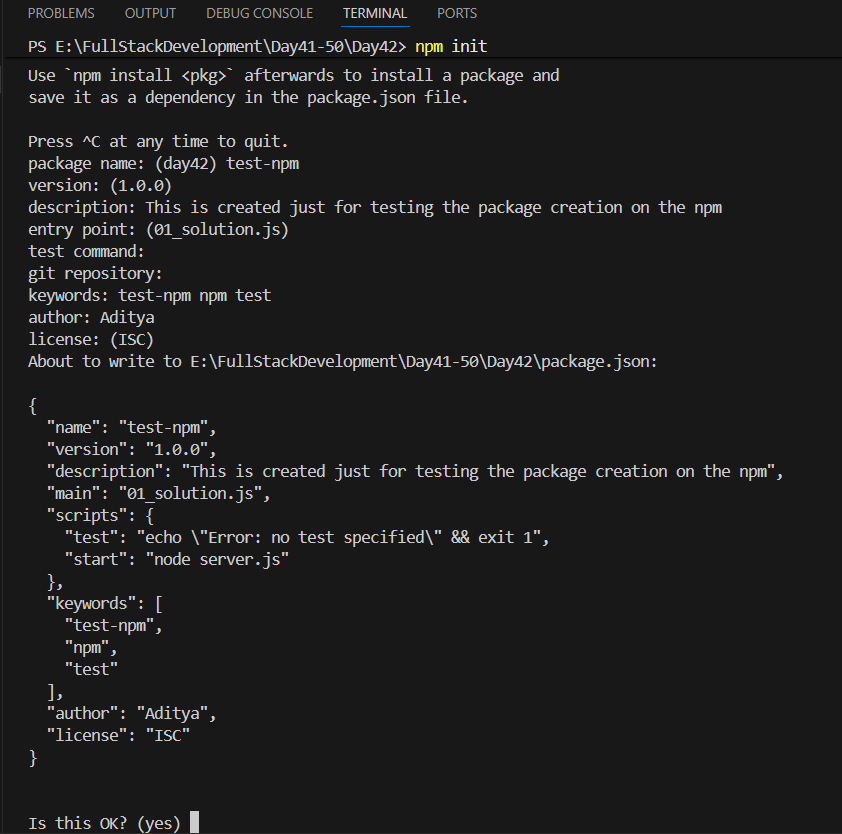
Now, let’s see a basic example where we used node to run the javascript code:

Server.js:



Now, we are initialising the npm in this: for which we will use “npm init”. Following text will come in the terminal, fill as per your wish:

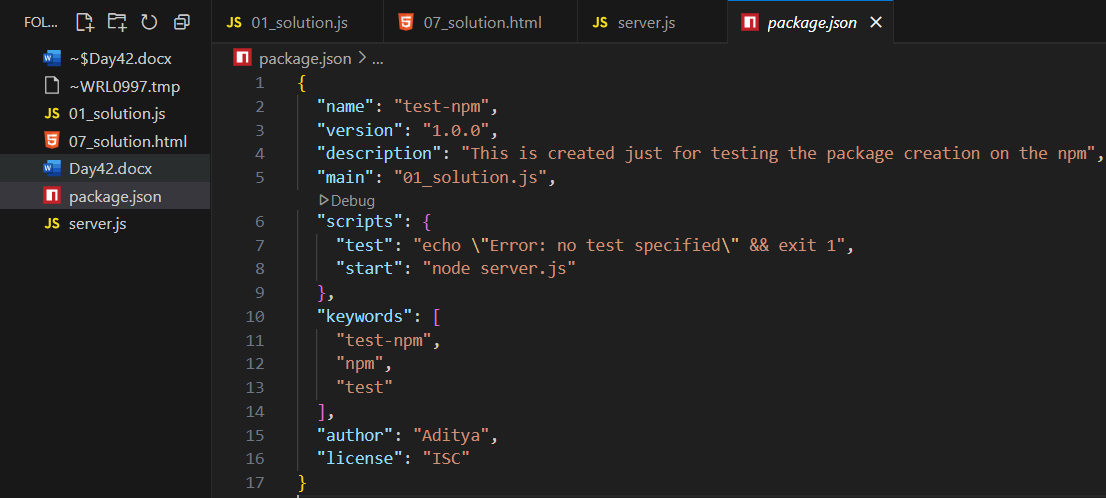




Write yes to confirm: following package.json will automatically get generated.

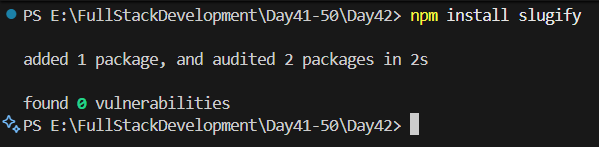


We can see the content of it:

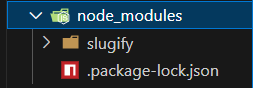


Now, installing packages using the npm: npm install <package-name>

For example:



Clearly, we can see this in the folder:



**CommonJs Vs EcmaScript Modules:**

**What Are Modules in JavaScript?**

A module means splitting code into multiple files, so each file handles one specific part — making code reusable and organized.

Example idea:

* math.js → handles calculations
* server.js → runs server logic

In JS, there are two main module systems:  
-> CommonJS (CJS) — older, used in Node.js (by default)  
-> ECMAScript Modules (ESM) — newer, standard in modern JS

**CommonJS (CJS)**

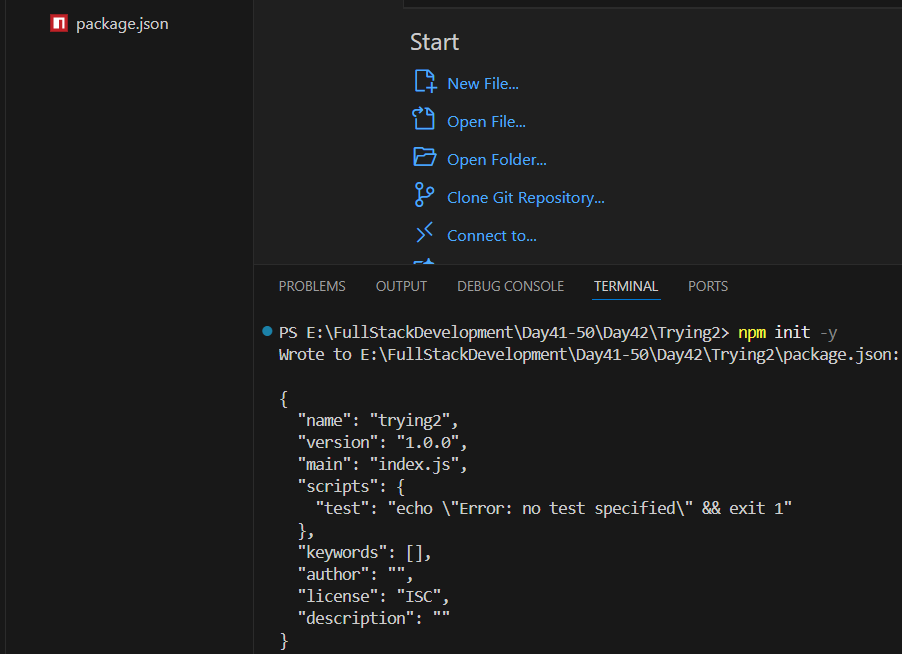
* Used in: Node.js (default)
* File extension: .js
* Uses: require() to import, module.exports to export

**ECMAScript Modules (ESM)**

* Used in: Modern JavaScript (Frontend + Node 14+)
* File extension: .mjs (or .js with "type": "module" in package.json)
* Uses: import and export keywords

Let’s start from basics:

We can do this to get package.json without specifying the questions: clearly, package.json gets created.

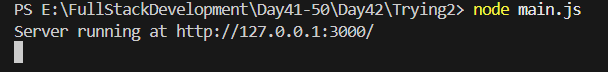


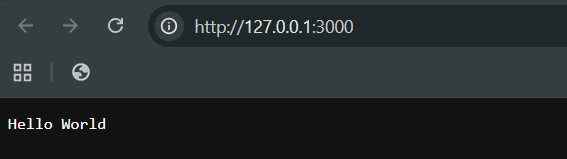
Now, creating a basic server using the node.js: just copy paste the code

Main.js:



Terminal:



Output:  


Now, let’s make some changes in the code: just add <h1> there

Main.js:



Output:

