Node -v This  will check the current version of node.js

Node R e  p l:  we enter node reply by entering node to exit forn node write control c

cd to change directories

$ node index.js

Node.js is a **runtime environment** that allows you to run JavaScript **outside the browser**, typically on the server. It is built on Chrome's **V8 JavaScript engine** and is commonly used for backend development.

### ****Uses of Node.js:****

1. **Server-Side Development** – You can build fast and scalable backend services.
2. **REST APIs & Web Services** – Node.js is great for handling HTTP requests in web applications.
3. **Real-Time Applications** – Ideal for chat apps, gaming servers, and live notifications.
4. **Microservices** – Helps in creating lightweight, independent services.
5. **Streaming Applications** – Used for video/audio streaming platforms.
6. **Command-Line Tools** – You can create automation scripts and CLI applications.
7. **IoT (Internet of Things)** – Used for managing connected devices efficiently.
8. **Single-Page Applications (SPAs)** – Works well with frameworks like React, Vue, and Angular.

Since you already know JavaScript, learning Node.js will help you build **full-stack applications** by combining frontend and backend development. Let me know if you want a project suggestion to get started! 🚀

Npm

NPM (**Node Package Manager**) is a package manager for JavaScript. It is used to install, manage, and share JavaScript packages (also called modules or libraries).

### Key Features of NPM:

* **Manages dependencies**: Automatically installs and updates libraries your project needs.
* **Package registry**: Provides access to thousands of open-source JavaScript libraries.
* **Version control**: Helps manage different versions of packages.
* **Scripts automation**: Allows running scripts (e.g., starting a server, running tests, etc.).

### Common NPM Commands:

* npm init – Initializes a new project with a package.json file.
* npm install <package> – Installs a package locally.
* npm install -g <package> – Installs a package globally.
* npm update – Updates installed packages.
* npm uninstall <package> – Removes a package.

Since you're skilled in frontend development, you might use NPM with tools like React, Vue, or build tools like Webpack and Babel. Let me know if you need help setting up a project! 🚀

npm first we write the package npm init

npm install <something> we want to install in our package

This is the example of installing the packages npm i sillyname

In order to upgrade our projects we actuy need to go package.json

New method in node.js is go there in package

Also Write type:module

Default type is : commonjs

We chane it to type:module

{

  "name": "learning-npm",

  "version": "1.0.0",

  "description": "Im learning about NPM",

  "main": "index.js",

  "type":"module",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1"

  },

  "author": "Maaz",

  "license": "ISC",

  "dependencies": {

    "sillyname": "^0.1.0"

  }

}

Changing your package type to "module" in your package.json tells Node.js to treat your code as using **ECMAScript Modules (ESM)** instead of the older **CommonJS** system. Here’s why you might want to make that change:

### 1. ****Modern JavaScript Syntax****

* **ES Modules** allow you to use the standard import and export syntax, which is widely adopted in modern JavaScript development.
* This syntax is consistent with what’s used in browsers, making your code more portable and easier to understand across different environments.

### 2. ****Improved Static Analysis****

* ESM benefits from static analysis. Since the module structure is determined at compile time, tools and bundlers can perform better optimizations and catch errors early.

### 3. ****Better Interoperability with the Browser****

* As web applications increasingly rely on modules, using ESM on the backend (Node.js) can simplify sharing code between the client and server.

### 4. ****Future-Proofing Your Code****

* The ECMAScript module system is the standardized way of handling modules in JavaScript. Migrating to it helps future-proof your codebase as the ecosystem continues to evolve.

### 5. ****Consistency Across Your Projects****

* By using the same module system both in Node.js and in browser environments, you reduce context switching and potential errors related to module resolution.

### How to Change Your Package to Use ESM

In your package.json, add or update the "type" field:

{ "name": "your-project-name", "version": "1.0.0", "type": "module", "dependencies": { // your dependencies here } }