Academy

Module 4: BACKEND DEVELOPMENT

17/12/2025

Introduction to Node.js & NPM

Tutor: Onyekachi Obute

Topics

01 What Node.js is and how it differs from browser JavaScript

02 Understanding the Node.js runtime environment

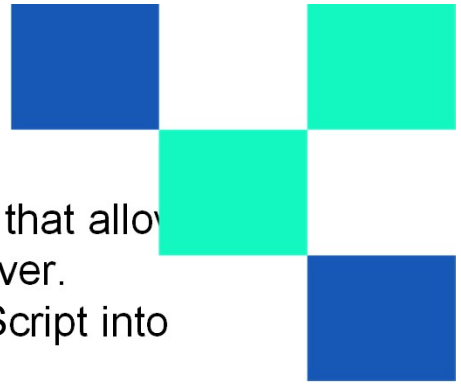
03 Introduction to NPM (Node Package Manager)

04 Installing and managing packages

05 Understanding package.json

06 Built-in Node modules: fs, path, http

What is Node.js?



Node.js is an open-source, cross-platform JavaScript runtime environment that allo
developers to run JavaScript code outside the browser, typically on the server.
It is built on Google Chrome's V8 JavaScript engine, which compiles JavaScript into
fast machine code.

Why Node.js Exists
Before Node.js:
•JavaScript was mainly used only in browsers
•Backend development required other languages like Java, PHP, or C#
With Node.js:
•JavaScript can be used for both frontend and backend
•Developers can build full-stack applications using one language

- Node.js is a JavaScript runtime built on Chrome's V8 engine
- Allows JavaScript to run outside the browser
- Used for building fast, scalable backend applications

Node.js vs Browser JavaScript

Although both use **JavaScript**, they run in **very different environments** and serve **different purposes**.

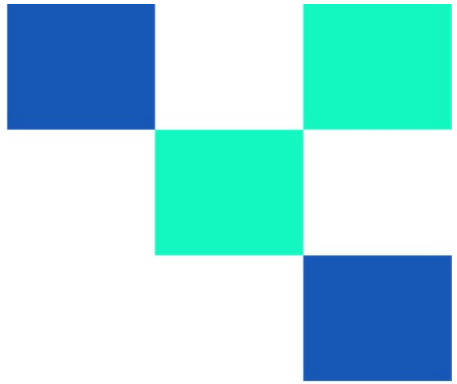
- Node.js runs on servers, browsers run on client machines
- Node.js has access to file system and OS resources
- Browser JavaScript interacts mainly with the DOM

Execution Environment

Browser JavaScript
- Runs inside a web browser (Chrome, Firefox, Edge, Safari)
- Executes in a sandboxed environment for security
- Mainly used for user interface and interactions

Node.js
- Runs on the server or local machine
- Not sandboxed like browsers
- Used for backend logic, APIs, and system-level tasks

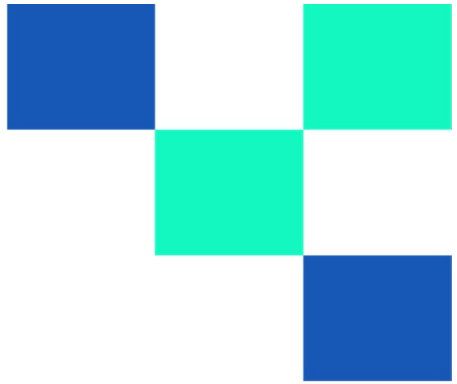
Node.js Runtime Environment



The **Node.js runtime environment** is the system that allows **JavaScript code to run outside a web browser**, typically on a server or local machine. It provides all the **tools, libraries, and mechanisms** needed to execute JavaScript, manage system resources, and handle asynchronous operations efficiently.

Single-threaded but highly scalable
Uses event-driven, non-blocking I/O
Ideal for APIs, real-time apps, and microservices

Introduction to NPM

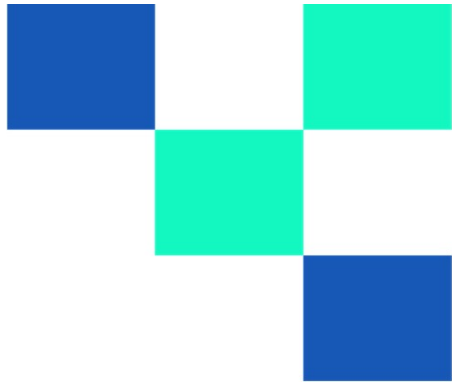


NPM (Node Package Manager) is the default tool that comes with Node.js and is used to **install, manage, share, and update JavaScript packages** (libraries and tools) used in Node.js applications.

What is NPM Used For?
NPM helps developers:
•Reuse existing code instead of writing everything from scratch
•Manage project dependencies
•Maintain consistent versions of libraries
•Run project scripts (build, start, test, etc.)

How NPM Works
•NPM connects to the NPM Registry, a large online repository of open-source packages
•Packages are downloaded and stored in the node_modules folder
•Dependency information is tracked in the package.json file

Installing & Managing Packages



Installing and managing packages with **NPM** allows developers to easily add, update, and remove libraries required for a Node.js application. This helps keep projects **organized, scalable, and maintainable**.

- npm install <package-name>
- npm uninstall <package-name>
- npm update <package-name>
- Global vs local package installation

Understanding package.json

package.json is the essential metadata file for Node.js/JavaScript projects, acting as a manifest that describes your project, lists its dependencies (libraries it needs), defines scripts for automation (like starting or testing), and holds configuration for npm/yarn. It's a JSON file in the project's root, vital for managing packages, running tasks, and ensuring consistent environments across different machines by installing the correct library versions with npm install

- Contains project metadata
- Lists dependencies and scripts
- Controls project configuration
- Automatically created with npm init

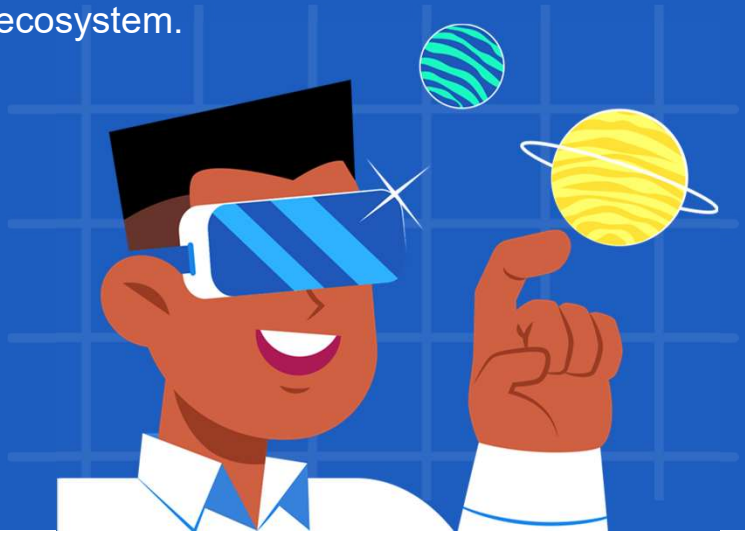
Built-in Node.js Modules

Node.js includes numerous built-in modules that provide core functionalities like file system operations, networking, and operating system information, without requiring separate installation. These modules are compiled into the Node.js binary and can be accessed using the require() function in CommonJS modules or import statements in ECMAScript modules.

- fs – File system operations
- path – File and directory paths
- http – Creating web servers
- No installation required

Why Learn Node.js?

Learning Node.js is valuable primarily because it allows developers to use a single language (JavaScript) for both front-end and back-end development, enabling the creation of fast, scalable applications with a large, supportive ecosystem.



- High performance and scalability
- Uses JavaScript for frontend and backend
- Strong ecosystem and community
- Widely used in modern backend development

Assignments

Exporting and Importing Functions
This will be submitted on GitHub

ANY QUESTIONS?


THANK YOU