Mastering Node.js: A Comprehensive Crash Course

Introduction to Node.js

- **Open-source, server-side** JavaScript **runtime**.
- Allows you to run **JavaScript** on your **machine** or **servers**.
- Built on the **V8 JavaScript engine** for speed.
- Robust ecosystem with **npm**

Open Source??

Open-source means that the **source code of Node.js is publicly available.**

Anyone can:

- **View**: You can see how Node.js is built under the hood.
- **Modify**: If you want to customize or improve Node.js, you can do it.
- **Contribute**: Developers around the world actively contribute to improving Node.js.

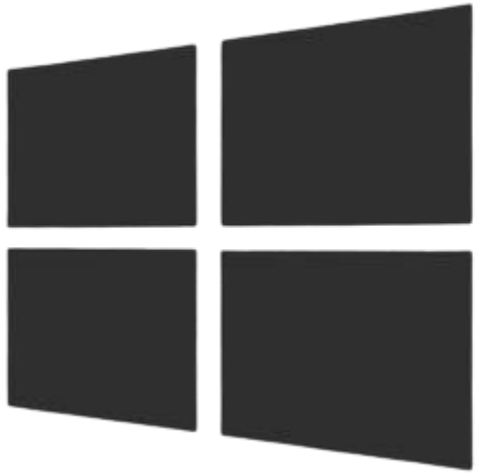
THAPA TECHNICAL – NODE JS SERIES YOUTUBE

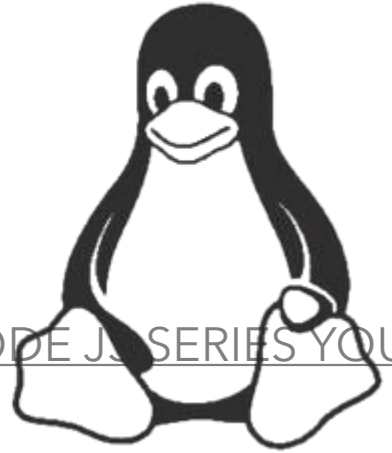
Cross-Platform??

Cross-platform means Node.js can **run on multiple operating systems** without needing major changes.

Supported platforms:

Windows **macOS** **Linux**





JavaScript Runtime Environment??

JavaScript runtime environment refers to the **environment where your JavaScript code runs.**

In browsers: JavaScript typically runs in the browser (like Chrome or Firefox) to handle frontend tasks.

With Node.js:

Node.js allows JavaScript to **run outside the browser, on the server**.

It provides tools to interact with the system, like:

File system (read/write files).

Network (handle HTTP requests).

Database (connect to databases like MongoDB or MySQL).

Why is Node.js Special as a Runtime?

V8 Engine:

Node.js uses Google Chrome's V8 engine to compile JavaScript into machine code, making it lightning-fast.

Built-in APIs:

Node.js comes with built-in APIs (like fs for file systems or http for servers) so you can build powerful applications without extra libraries.

Why is Node.js Special as a Runtime?

V8 Engine:

Node.js uses Google Chrome's V8 engine to compile JavaScript into machine code, making it lightning-fast.

Built-in APIs:

Node.js comes with built-in APIs (like fs for file systems or http for servers) so you can build powerful applications without extra libraries.

Why Do We Need Node.js?

Single Language for Full Stack Development:

- Developers can use JavaScript for both frontend and backend, reducing complexity.
- For example, a MERN stack project (MongoDB, Express, React, Node).

High Performance:

- Thanks to the V8 engine, Node.js is super fast.
- It can handle thousands of simultaneous connections with ease.

Event-Driven and Non-Blocking I/O:

- Unlike traditional servers that block a thread for each request, Node.js uses an asynchronous model, making it highly efficient for handling multiple tasks.

THAPA TECHNICAL – NODE JS SERIES YOUTUBE

Scalable for Modern Applications:

Ideal for real-time applications like chat apps, streaming platforms, or online games.

Where Is Node.js Used?

Examples of Companies Using Node.js:

Netflix: To deliver faster streaming experiences.

LinkedIn: Their backend uses Node.js to handle massive amounts of data.

PayPal: To handle secure transactions and ensure scalability.

Uber: For real-time driver and rider data syncing.

Trello: For managing asynchronous updates in real-time boards.

THAPA TECHNICAL – NODE JS SERIES YOUTUBE

Million Dollar Question

Is it a Language or a Framework?

Node.js is not a programming language or a framework, but rather a JavaScript runtime environment that allows developers to run JavaScript outside of a browser

History and Evolution

- **JavaScript's Initial Role:** Initially, JavaScript was only used for client-side tasks in web browsers.
- **Creation of Node.js (2009):** Ryan Dahl created Node.js by taking the V8 JavaScript engine (used in Google Chrome) and running it outside the browser. This innovation allowed JavaScript to handle server-side development.
- **Introduction of Modules:** Node.js came with built-in modules like http, which made it popular for creating backend applications.

There has been lots of drama behind Node.js and its evolution. If you want to learn more, watch this [video](https://www.youtube.com/watch?v=LB8KwiiUGy0)

- **npm (2010):** Node.js launched its package manager (npm) in 2010, revolutionizing JavaScript development by simplifying package sharing and dependency management.
- **io.js and Node.js Merger (2015):** In 2015, Node.js merged with io.js, reuniting the ecosystem and bringing in community-driven improvements.
- **Node.js Foundation (2015):** The Node.js Foundation was established to support collaboration, innovation, and sustainable development within the ecosystem.

There has been lots of drama behind Node.js and its evolution. If you want to learn more, watch this [video](https://www.youtube.com/watch?v=LB8KwiiUGy0) https://www.youtube.com/watch?v=LB8KwiiUGy0

Some Famous npm Packages

- **Express.js:** A popular framework for building web applications and APIs.
- **Mongoose:** A package for interacting with MongoDB databases using an object data modeling (ODM) approach.
- **Axios:** A promise-based HTTP client for making API requests.
- **React (and React-DOM):** Though originally developed for the browser, React can be installed via npm for building modern web applications.
- **Socket.IO:** Enables real-time communication between clients and servers.
- **Passport.js:** Authentication middleware for Node.js.

Merger of Node.js and io.js:

What happened:

- io.js was a fork of Node.js created by developers who wanted faster updates and improvements. The merger brought io.js's community-driven innovations back into the main Node.js project.
- The actual merger took place in September 2015, resulting in the release of Node.js v4.0. This version combined the best features of both projects, including V8 ES6 support from io.js, and marked the beginning of a Long-Term Support (LTS) release cycle for Node.js.
- This ensured faster progress, stability, and a unified ecosystem for developers.

THAPA TECHNICAL – NODE JS SERIES YOUTUBE

Prerequisites to Learn Node.js

Prerequisites to Learn Node.js

HTML

CSS

JavaScript

- Basic & Intermediate
- ES6 & Above