Introduction Of Node.js

What is Node.js?

- Node.js is an open-source, cross-platform JavaScript runtime environment.
- It executes JavaScript code outside a web browser.
- Built on Chrome's V8 JavaScript engine.
- Allows developers to use JavaScript for server-side programming.

What is Node.js?

- Node.js is an **open-source, cross-platform JavaScript runtime environment** that allows developers to run JavaScript code **outside of a web browser**. Traditionally, JavaScript was used only inside browsers (like Chrome, Firefox, Edge) to make websites interactive. But with Node.js, developers can use JavaScript for **server-side programming** as well. This means the same language can be used for both **frontend** and **backend** development, making development faster and more efficient.

Open Source Means

- When we say Node.js is **open-source**, it means:
- Its **source code is publicly available** for anyone to view, modify, and contribute.
- A large community of developers around the world contributes to improving Node.js by fixing bugs, adding new features, and enhancing performance.
- It is maintained by the **OpenJS Foundation**, which ensures its long-term development and stability.
- Open-source software is usually **free to use**, making it accessible to individuals, startups, and enterprises.

Cross-Platform

- Node.js is **cross-platform**, meaning it works on multiple operating systems such as:
- Windows
- macOS
- Linux
  This allows developers to write applications once and run them on different systems without rewriting code for each platform.

Chrome's V8 JavaScript Engine

- At the heart of Node.js is **Google's V8 engine**, which is the same engine used in the Chrome browser.
- The **V8 engine** is an open-source JavaScript engine developed by Google.
- Its main job is to **convert JavaScript code into machine code** that your computer's CPU can understand.
- Unlike traditional interpreters that read code line by line, V8 **compiles JavaScript into highly optimized machine code**, making it extremely fast.
- This is why Node.js is known for **high performance** and can handle a large number of requests efficiently.

History of Node.js

- **The Beginning (2009)**
- **Creator**: Node.js was created by **Ryan Dahl**, an American software engineer.
- **Reason for creation**: At the time, web servers (like Apache) followed a **multi-threaded blocking model**, which wasn't efficient for handling thousands of concurrent connections.
- Ryan Dahl wanted to build a lightweight, efficient system that could handle **asynchronous, non-blocking I/O operations**, making it ideal for **real-time, scalable applications**.
- **First release**: Node.js was released in **May 2009**.

History of Node.js

- **Core Technology**
- Built on **Google Chrome's V8 JavaScript Engine**, which compiles JavaScript into machine code for high performance.
- Introduced an **event-driven, non-blocking I/O model**, making it efficient and scalable for real-time applications like chat servers, APIs, and streaming platforms.
- This design made Node.js very different from traditional server-side technologies like PHP, Ruby on Rails, or Java-based servers.

History of Node.js

- **Early Adoption (2010–2012)**
- Node.js quickly gained popularity among developers because it allowed them to use **JavaScript on both the client and server side**.
- **npm (Node Package Manager)** was introduced in **2010** by Isaac Schlueter, which made Node.js even more powerful.
  - npm provided a massive ecosystem of open-source packages that developers could use instead of building everything from scratch.
- By **2011**, companies like LinkedIn and Uber adopted Node.js for their applications due to its scalability.

History of Node.js

- **Corporate Support & Joyent (2011–2014)**
- **Joyent**, a cloud computing company, hired Ryan Dahl and took over stewardship of Node.js.
- Dahl later stepped away from the project (2012), and **Isaac Schlueter** and others continued the development.
- Node.js kept growing, but as more developers joined, issues began to appear:
  - **Slow release cycles**.
  - Lack of transparency in decision-making.
  - Frustration in the open-source community about governance

History of Node.js

- **The io.js Fork (2014)**
- In **December 2014**, due to disagreements with Joyent's management of Node.js, a group of developers forked the project and created **io.js**.
- **io.js goals**:
  - Faster and more transparent release cycles.
  - Use of the latest features of the V8 engine (e.g., ES6/ES2015 support).
  - More open governance.
- io.js quickly gained popularity and active contributors.

History of Node.js

- **The Merger (2015)**
- The split between Node.js and io.js confused the developer community.
- To resolve this, the **Node.js Foundation** (under the Linux Foundation) was formed in **2015** to manage the project with an open governance model.
- In **September 2015**, io.js officially **merged back into Node.js**, combining the strengths of both projects.
- The project adopted a **neutral foundation governance model**, ensuring that no single company controlled Node.js.

History of Node.js

- **OpenJS Foundation (2019 – Present)**
- In **2019**, the **Node.js Foundation** merged with the **JS Foundation** to create the **OpenJS Foundation**.
- The OpenJS Foundation is now responsible for maintaining and growing Node.js, along with other popular JavaScript projects like jQuery, Electron, webpack, and more.
- Today, Node.js is one of the **most widely used back-end technologies**, supported by major companies like Google, Microsoft, IBM, and PayPal.

History of Node.js

- **Timeline of Node.js**
- **2009 (May):** Ryan Dahl creates Node.js, first release.
- **2010:** npm launched by Isaac Schlueter.
- **2011:** Joyent takes stewardship, LinkedIn adopts Node.js.
- **2012:** Ryan Dahl steps back from Node.js.
- **2014:** io.js fork created due to governance issues.
- **2015:** io.js merges back → Node.js Foundation formed.
- **2019:** Node.js Foundation + JS Foundation merge → OpenJS Foundation.
- **2025 (Today):** Node.js remains one of the most important technologies in web development.

Features of Node.js

- Asynchronous and event-driven.
- Fast execution with V8 engine.
- Single-threaded but handles multiple requests using event loop.
- Rich package ecosystem via npm (Node Package Manager).
- Cross-platform support.

Frameworks in Node.js

- Express.js – Minimal and flexible web application framework.
- NestJS – Progressive Node.js framework for building efficient apps.
- Koa.js – Lightweight and modern web framework.
- Meteor.js – Full-stack framework for real-time apps.
- Sails.js – MVC framework for Node.js.

Benefits of Node.js

- High performance for real-time applications.
- Scalability with non-blocking I/O.
- Single programming language (JavaScript) for frontend and backend.
- Large and active community.
- Rich ecosystem of libraries and tools.

Use Cases of Node.js

- Real-time chat applications.
- Streaming applications.
- Single-page applications (SPAs).
- APIs and microservices.
- IoT applications.

Why Use Node.js?

- **Fast Performance** → Thanks to the V8 engine.
- **Same Language for Client & Server** → Developers don't need to switch between languages.
- **Scalability** → Suitable for real-time apps like chat applications, gaming servers, and streaming services.
- **Large Ecosystem** → Node.js has **npm (Node Package Manager)**, which contains millions of reusable packages and libraries.

Installing Node.js

- Step 1: Visit official website https://nodejs.org
- Step 2: Download the LTS (Long Term Support) version.
- Step 3: Run the installer and follow installation steps.
- Step 4: Verify installation using command: node -v and npm -v

Running a Basic Node.js Program

- 1. Open a text editor (e.g., VS Code).
- 2. Create a file named app.js
- 3. Write code: console.log('Hello Node.js');
- 4. Open terminal and navigate to file location.
- 5. Run command: node app.js
- 6. Output will be displayed in terminal.

Variables in Node.js

- Variables are containers for storing data values.
- Declaration keywords: var, let, const
- Example: let name = 'John';
- Best practice: Use let and const (avoid var).

Data Types in Node.js

- Primitive Types: String, Number, Boolean, Undefined, Null, Symbol, BigInt.
- Non-Primitive: Object, Array, Function.
- Example: let num = 10; let arr = [1,2,3]; let obj = {name: 'John'};

Operators in Node.js

- Arithmetic Operators: +, -, *, /, %
- Assignment Operators: =, +=, -=, *=, /=
- Comparison Operators: ==, ===, !=, >, <, >=, <=
- Logical Operators: &&, ||, !
- Type Operators: typeof, instanceof

Difference: JavaScript vs Node.js

- JavaScript is mainly used for client-side scripting in browsers.
- Node.js is a runtime environment to run JavaScript on the server.
- JS has access to DOM, Node.js does not.
- Node.js provides modules like fs, http, path for backend tasks.
- JS runs in browser; Node.js runs on server-side.

Global Variables in JavaScript

- In JavaScript (browser):
- - window is the global object.
- - this refers to window object in global scope.
- - Example: console.log(this) → window.

Global Variables in Node.js

- In Node.js:
- - global is the global object (not window).
- - this refers to module.exports in global scope.
- - __dirname: path of current directory.
- - __filename: file name with path.
- - process: provides info about current Node.js process.