**Introduction to Node.js**

**Author:** Apurva Sachin Kulkarni

**Date:** 28th August, 2025

**Introduction**

In today’s software development world, applications demand speed, scalability, and real-time interactions. Traditional server-side technologies often struggled with handling large numbers of concurrent requests efficiently. To solve this problem, Node.js was introduced as a lightweight, fast, and event-driven runtime environment that allows developers to run JavaScript on the server-side. Built on Chrome’s V8 JavaScript engine, Node.js has transformed the way developers build applications by enabling the use of a single programming language—JavaScript—for both frontend and backend development. With its non-blocking I/O model, single-threaded event loop, and a massive ecosystem of modules via npm, Node.js has become one of the most popular technologies for modern web development. This documentation provides a detailed overview of Node.js, its features, architecture, modules, and use cases, making it a great starting point for beginners who want to understand the fundamentals of Node.js.

**Introduction to Node.js**

**1. What is Node.js?**

* Node.js is an open-source, cross-platform, runtime environment that allows you to run JavaScript code outside of a web browser.
* It is built on Chrome’s V8 JavaScript engine, making it fast and efficient.
* It is especially popular for building server-side applications, REST APIs, and real-time applications like chat apps.

In simple words: Node.js lets you use JavaScript for backend development, not just frontend scripting in the browser.

**2. Why Node.js?**

* **Single Language**: Developers can use JavaScript for both frontend and backend.
* **Non-blocking I/O**: Uses an event-driven, asynchronous model, which makes it lightweight and efficient.
* **Fast execution**: Powered by Google’s V8 engine.
* **Scalable**: Perfect for applications that need to handle thousands of connections at once.
* **Huge ecosystem**: Through npm (Node Package Manager), developers get access to over a million packages.

**3. History of Node.js**

* Created in 2009 by Ryan Dahl.
* Goal: To build scalable network applications that can handle large numbers of simultaneous connections.
* Written in C, C++, and JavaScript.

**4. Features of Node.js**

* **Asynchronous & Event-driven**
  + All APIs are non-blocking.
  + A Node.js server never waits for an API to return data.
* **Single-threaded but highly scalable**
  + Uses an **event loop** instead of creating new threads for every request.
* **Fast execution**
  + Runs on Google V8 engine → compiles JS into machine code.
* **Cross-platform**
  + Runs on Windows, macOS, Linux, etc.
* **NPM (Node Package Manager)**
  + Largest ecosystem of open-source libraries.

**5. Node.js Architecture**

1. **Client Request** → comes from the browser/app.
2. **Event Queue** → request stored here.
3. **Event Loop** → continuously listens for events.
4. **Worker Threads / APIs** → process the request asynchronously.
5. **Response** → sent back to the client.

This makes Node.js **non-blocking** and highly efficient.

**6. Installing Node.js**

* Download from [nodejs.org](https://nodejs.org/).
* Installation includes:
  + **node** → to run JavaScript files.
  + **npm** → package manager.

Check installation:

node -v

npm -v

**9. Node.js Modules**

* **Core modules** (built-in): http, fs, path, os, events.
* **Custom modules** (your own files).
* **Third-party modules** (via npm, e.g., express, mongoose).

**10. Node Package Manager (npm)**

* Default package manager for Node.js.
* Used to install libraries.

**11. Use Cases of Node.js**

* Web servers & REST APIs.
* Real-time chat applications.
* Streaming services.
* IoT applications.
* Single Page Applications (SPAs).

**12. Advantages of Node.js**

1. High performance.
2. Large community support.
3. Same language (JavaScript) for both client & server.
4. Great for real-time apps.

**13. Limitations of Node.js**

1. Not suitable for CPU-intensive tasks (e.g., video processing).
2. Callback hell (though reduced by Promises & async/await).
3. Still evolving compared to older backend technologies.

**14. Node.js Modules**

In Node.js, modules are reusable blocks of code whose existence does not accidentally impact other code. They help in structuring code, improving maintainability, and avoiding code duplication.

**Types of Modules**

**a) Core Modules (Built-in)**

Node.js comes with several modules already included.  
Examples:

* **http** – to create servers and handle HTTP requests.
* **fs (File System)** – to work with files (create, read, update, delete).
* **path** – to handle file and directory paths.
* **os** – to get OS-related info.
* **events** – to handle custom events.

**b) Local (User-defined) Modules**

These are modules that you create yourself.

**c) Third-party Modules (npm Modules)**

Installed via **npm (Node Package Manager)**.

**Module.exports and require()**

* module.exports → used to **export** functions, objects, or variables from a module.
* require() → used to **import** them into another file.

**ES6 Modules (Modern Syntax)**

Node.js (v13+) supports **ES6 module syntax**.  
Instead of require and module.exports, you can use import and export.

**Conclusion**

Node.js has grown from a simple runtime environment into a powerful platform that drives some of the world’s most popular applications. Its asynchronous, event-driven model makes it ideal for real-time, scalable, and data-intensive applications, while its vast npm ecosystem provides developers with countless tools and libraries to accelerate development. By understanding the basics of Node.js—its features, architecture, and module system—developers can unlock the full potential of JavaScript in both client-side and server-side environments. Whether you’re building APIs, chat applications, or enterprise-level solutions, Node.js offers the flexibility, speed, and scalability required for modern web development. In summary, Node.js is more than just a technology; it is a modern approach to building fast, scalable, and efficient applications, making it an essential skill for every aspiring developer.

\*\*\*