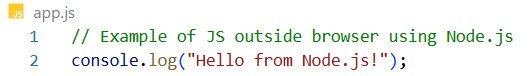
Namaste NodeJS - Ep 1: Introduction to NodeJS

What is Node.js?

* Node.js is a JavaScript runtime built on Chrome’s V8 JavaScript Engine.
* It allows developers to run JavaScript outside the web browser - such as on servers, command-line tools, etc.



Who built Node.js?

* Creator: Ryan Dahl
* Year: 2009
* Maintained by: Initially Joyent, now the OpenJS Foundation

Key Concept: Run JavaScript Everywhere

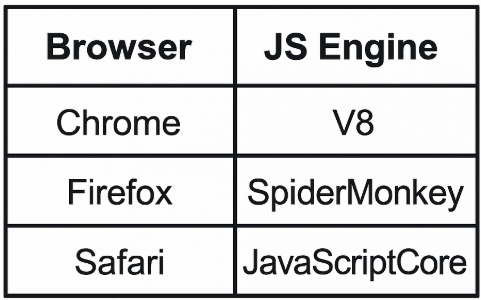
JavaScript is the language of the browser. But with Node.js, we can run JavaScript anywhere.

That’s why Node.js is often associated with the phrase: "Run JavaScript Everywhere"

Why do we need Node.js?

To run JavaScript, we need a JavaScript Engine.

Every browser already has one.



Node.js uses V8 engine from Chrome to execute JavaScript outside browsers.

How Node.js Started (Ryan Dahl’s Journey)

* Ryan started Node.js using SpiderMonkey (Firefox's engine)
* Within 2 days, he switched to V8 because of its performance and stability.
* He neverlooked back - V8 became the core engine of Node.js.

### **Role of Joyent in Node.js**

* **Ryan Dahl started Node.js by himself.**
* A company called **Joyent** was working on similar ideas at the same time.
* **Joyent hired Ryan** and gave him support to continue working on Node.js.
* **They gave money and resources** to help build and improve the project.
* **Joyent made Node.js open for everyone (open-source)** and helped share it with the developer community.
* Thanks to **Joyent’s support**, Node.js became **popular and successful**.

Original Name of Node.js

* Initially called web.js
* Later renamed to Node.js
  + Because its scope extended beyond web servers
  + It supported scripting, file operations, CLI tools, etc.

### **Blocking Server (Apache server)**

* Traditional servers like **Apache servers** handle **one request at a time.**
* If one user’s request takes time (e.g., reading a file or calling a database),  
  👉 the server pauses and **waits** until it finishes.
* This **blocks other users,** even if their requests are quick.

Think of it like - "You're standing in a queue. If one person takes too long, everyone behind them has to wait."

### **Non-blocking Server (like Node.js)**

* Node.js is **non-blocking** meaning it works in a smarter way. It does not wait for one task to finish.
* Instead it says Okay, I’ll start this task... but meanwhile, I’ll handle other users too.

Think of it like - "You place an order at a food counter. While your food is being prepared, the server takes the next customer’s order. No waiting!"

## How does Node.js Work ?

### 1. ****Uses Non-blocking I/O****

#### **What does that mean?**

* I/O means **input/output tasks** like reading files, calling APIs, or connecting to a database.
* **Non-blocking** means Node.js **doesn’t wait** for those tasks to finish.

#### **How does it help?**

* While one task is running in the background, Node.js **moves on** to handle the next task.
* This keeps the app **fast and responsive.**

#### **Real-life example -**

You put a pizza in the oven and **don’t wait there staring at it**.  
Instead, you clean the room, check your phone, or make a salad.  
When the pizza is done, **the oven beeps**, and you go get it.

That’s **non-blocking**. Node.js does the same - it gives work to someone else (like the oven) and keeps going.

### 2. ****Uses Events and Callbacks****

#### **What is an event?**

* An **event** is something that happens - like a file being read, a button clicked, or data arriving from a server.
* Node.js listens for these events.

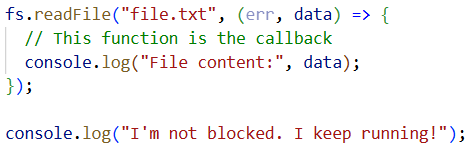
#### **What is a callback?**

A **callback** is a function that runs **after the event is finished.**

#### **Why use it?**

Instead of waiting, Node.js says - Hey, when this work is done (when the event is triggered), just **call this function back** to handle the result.

***Example -***



### *What's happening here?*

1. readFile("file.txt", callback) is called.
2. Node.js starts reading the file **in the background** (non-blocking).
3. While it reads, the rest of the code (like console.log("I'm not blocked")) runs **immediately**.
4. When the file is finally read:
   * That’s the **event** → File is ready
   * Node.js triggers the **callback function** you gave.

#### **Real-life example:**

You order food and tell the restaurant.  
Call me when it’s ready.  
You don’t wait. You do your own stuff.  
Later, they call you - that’s the **callback**.

In Node.js, callbacks help it **respond** once tasks are done - without freezing.

### 3. ****Runs on a Single Thread, but Handles Many Tasks****

#### **What is a thread?**

* A **thread** is like a single worker doing one job.
* Many programming languages use **multiple threads** (many workers).
* Node.js uses **just one main thread.**

#### **How does it still handle many users**?

* It uses an internal mechanism called the **event loop.**
* The event loop **jumps between tasks** - always checking if something is done and ready to continue.
* Node.js has many tasks running in the background (like file reading, API calls, timers, etc.)
* The **event loop keeps looping** through them (like going around in a circle).
* At each step, it asks: Is any task finished now? If yes, it says: Okay, now I’ll run the callback linked to that finished task.”

***What is the Event Loop?***

It's a watcher or traffic controller inside Node.js.

It monitors all the background tasks (like file reading, API calls, database requests). When any of those tasks are done, it tells Node.js: Hey, this task is ready - now run its callback!

#### **Benefit -**

Less memory needed, fewer bugs, and faster for I/O-heavy apps.

#### **Real-life example -**

One smart waiter (single thread) is:

* Taking orders
* Delivering food
* Talking to customers  
  ...by quickly rotating between them without getting stuck on one customer.

Node.js is like that waiter - one thread, many tasks.

### **Why Node.js is Fast & Scalable ?**

### Node.js uses non-blocking I/O and callbacks, so it never waits. It runs on one thread, but thanks to the event loop, it smartly manages many tasks at the same time - that’s what makes it super-fast and scalable.

* Handles **thousands of requests** without waiting
* Perfect for **real-time apps** like:
  + Chat apps
  + Live updates
  + Streaming
  + APIs

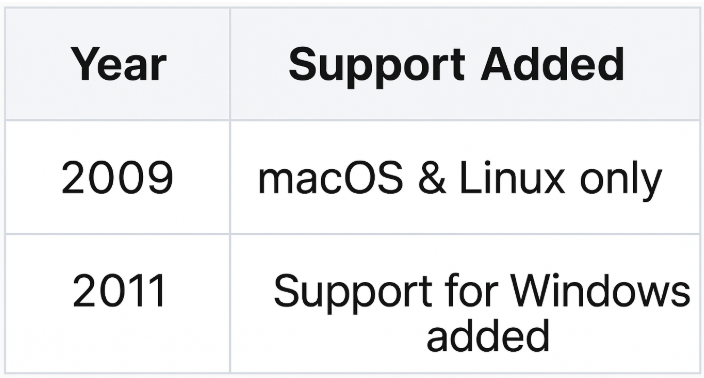
NPM – Node Package Manager

* Introduced in 2010
* Official package manager for Node.js
* Manages reusable packages (called modules)
* Node.js wouldn't have been so successful without NPM

You can install packages like this -



Platform Support



Node.js Maintainers & Fork History

***2012: Isaac Z. Schlueter***

* Started maintaining Node.js.
* Also, the creator of NPM.

***2014: io.js Fork***

* Developer Fedor Indutny forked Node.js to create io.js.
* Reason: Disagreements within the Node.js team.
* Result: Controversy and fragmentation in community.

***2015: Reunification***

* Developers from Node.js and io.js merged.
* Formed the Node.js Foundation.

***2019: Foundation Merge***

* JS Foundation and Node.js Foundation merged.
* Created the OpenJS Foundation.
* This new foundation now officially maintains Node.js.