



What is Node.js?

Node.js is an open-source **JS runtime** that allows you to execute JavaScript code on the server side. It's built on Chrome's V8 JavaScript engine.

Code - <https://github.com/nodejs/node>

Runtime?

The environment where JavaScript code is executed. It could be

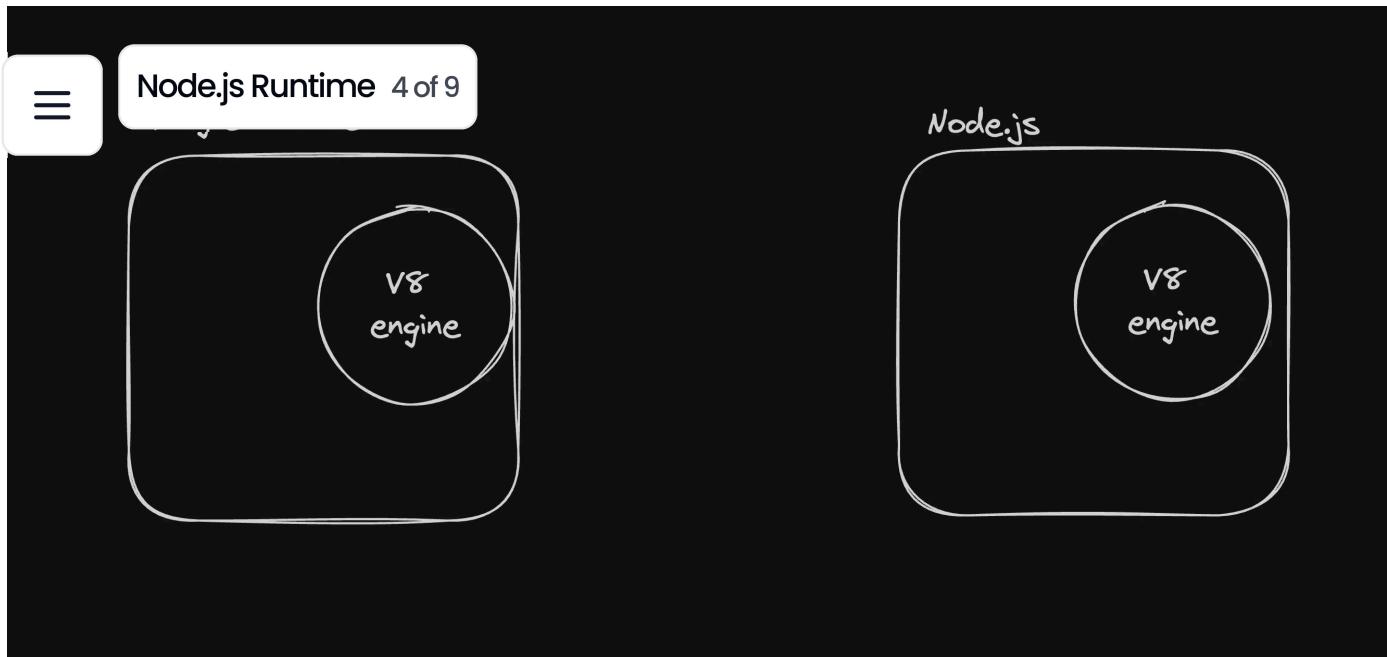
1. On the server
2. In the browser
3. On a small watch

...

V8 engine?

The V8 engine is an open-source JavaScript engine developed by Google. It is used to execute JavaScript code in various environments, most notably in the Google Chrome web browser.





Mozilla has their own JS engine – **SpiderMonkey**

Safari – **JavaScriptCore**

Installing Node.js

<https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-ubuntu-22-04>

- Build from source
- Using a package manager (brew, chocolat)

What is bun?

Like Node.js, Bun is a JavaScript runtime that allows you to execute JavaScript code on the server side.

The screenshot shows the Bun.js Runtime page. At the top, it says "Node.js Runtime 4 of 9". Below that is the title "Bun is a fast JavaScript test runner". A text block explains that Bun is an all-in-one runtime designed for speed, mentioning development, testing, bundling, and package management. It includes a "Server-side rendering React" benchmark chart comparing Bun, Deno.serve(), and Node.js. The chart shows Bun at 71,375 requests per second, Deno.serve() at 38,445, and Node.js at 14,525. Below the chart is an "Install Bun v1.1.26" section with "Linux & macOS" and "Windows" buttons, and a command-line install link. There's also a "View source" link.

Runtime	Version	HTTP requests per second (Linux x64)
Bun	v1.1	71,375
Deno.serve()	v1.42.1	38,445
Node.js	21.7.1	14,525

Installing bun

Linux

```
curl -fsSL https://bun.sh/install | bash
```



Mac

```
powershell -c "irm bun.sh/install.ps1 | iex"
```



Starting a Node.js project

- Run the following command -

Node.js Runtime 4 of 9

npm init -y



- Exploring package.json

- Writing some code

```
let firstName = "Harkirat Singh"  
console.log(firstName)
```



- Run the code

```
node index.js
```



- Add a `script` in `package.json`

```
"scripts": {  
  "start": "node index.js"  
},
```



- Run `npm run start`

npm

The package manager for modern JavaScript.

It is a package manager for JavaScript, primarily used for managing dependencies in Node.js projects. NPM allows developers to easily install, update, and manage packages of reusable code

 package managers are an important concept in programming languages/runtimes.

For eg the package manager of rust is cargo

Uses of npm

- Initializing a project

npm init



- Running scripts

npm run test



- Installing external dependencies

npm install chalk



- Write some code

```
const chalk = require('chalk');
```



```
console.log(chalk.blue('Hello, world!'));
```

```
console.log(chalk.red.bold('This is an error message.'));
```

```
console.log(chalk.green.underline('This is a success message.'));
```



packages

Node.js provides you some **packages** out of the box. Some common ones include

1. fs - Filesystem
2. path - Path related functions
3. http - Create HTTP Servers (we'll discuss this tomorrow)

fs package

The fs (Filesystem) package is used to read, write, update contents on the filesystem.

```
const fs = require('fs');
const path = require('path');

const filePath = path.join(__dirname, 'a.txt');

fs.readFile(filePath, 'utf8', (err, data) => {
  if (err) {
    console.log(err);
  } else {
    console.log(data);
  }
});
```



Why use the **path** library?

1. Cross platform joins (Windows has Users\kirat\dir , linux has Users/kirat/dir)
2. Gives you a bunch of helper functions (dirname)



External packages

These are packages written and maintained by other people. You just use their work in your project.

For example

1. Express

2. chalk

You can read more about them on their [npm page](#) -

<https://www.npmjs.com/package/chalk>

Sometimes they are open source as well -

<https://github.com/chalk/chalk>

Semantic Versioning Format

Every external package is updated incrementally. A specific version looks something like follows -

"chalk": "^5.3.0"



The format is as follows - **MAJOR.MINOR.PATCH**

Major version numbers indicate significant updates or breaking

- MINOR - Minor version changes signify the addition of new features or backward-compatible manner.
- PATCH - Patch version changes include backward-compatible bug fixes or minor improvements that address issues without adding new features or causing breaking changes.

Usage in `package.json`

- `"chalk": "^5.3.0"` - npm will install any version that is compatible with `5.3.0` but less than `6.0.0`. This includes versions like `5.3.1`, `5.4.0`, `5.5.0`, etc.
- `"chalk": "5.3.0"` - Will install the exact version
- `"chalk": "latest"` - Will install the latest version

package-lock.json

The `package-lock.json` records the exact versions of all dependencies and their dependencies (sub-dependencies) that are installed at the time when `npm install` was run.

Consistency: By locking down these versions, `package-lock.json` ensures that every time someone installs dependencies (e.g., by running `npm install`), they get the exact same versions of packages. This prevents discrepancies that can arise from different versions being installed in different environments.



Node.js Runtime 4 of 9

<https://github.com/code100x/cms/blob/main/package-lock.json>

Dailycode

<https://github.com/code100x/daily-code/blob/main/yarn.lock>

Assignments #1 – Create a cli

Create a **command line interface** that lets the user specify a file path and the nodejs process counts the number of words inside it.

Input – node index.js /Users/kirat/file.txt



Output – You have 10 words in this file

Library to use – <https://www.npmjs.com/package/commander>

▼ What's wrong with this code



Node.js Runtime 4 of 9



```
const { require } = process;  
const { Command } = require('commander');  
const program = new Command();  
  
program  
  .name('counter')  
  .description('CLI to do file based tasks')  
  .version('0.8.0');  
  
program.command('count')  
  .description('Count the number of lines in a file')  
  .argument('<file>', 'file to count')  
  .action((file) => {  
    fs.readFile(file, 'utf8', (err, data) => {  
      if (err) {  
        console.log(err);  
      } else {  
        const lines = data.split('\n').length;  
        console.log(`There are ${lines} lines in ${file}`);  
      }  
    });  
  });  
  
program.parse();
```

Assignment #2

Node.js Runtime 4 of 9

Filesystem based todo list.

Create a `cli` that lets a user

1. Add a todo
2. Delete a todo
3. Mark a todo as done

Store all the data in files (`todos.json`)