Platformatic

View online

Download PDF

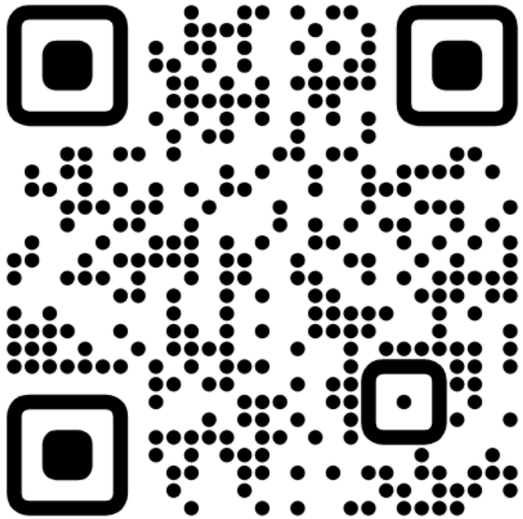
The alleged "end" of Node.js is much ado about nothing

Paolo Insogna

Node.js TSC, Principal Engineer

How can Platformatic help?

Platformatic gives you the complete out-of-the-box primitive to deploy, optimize, autoscale, and secure all your Node.js apps. No rewrites, no overhauls. **Just plug and play.**



https://qr.link/uCLwdt

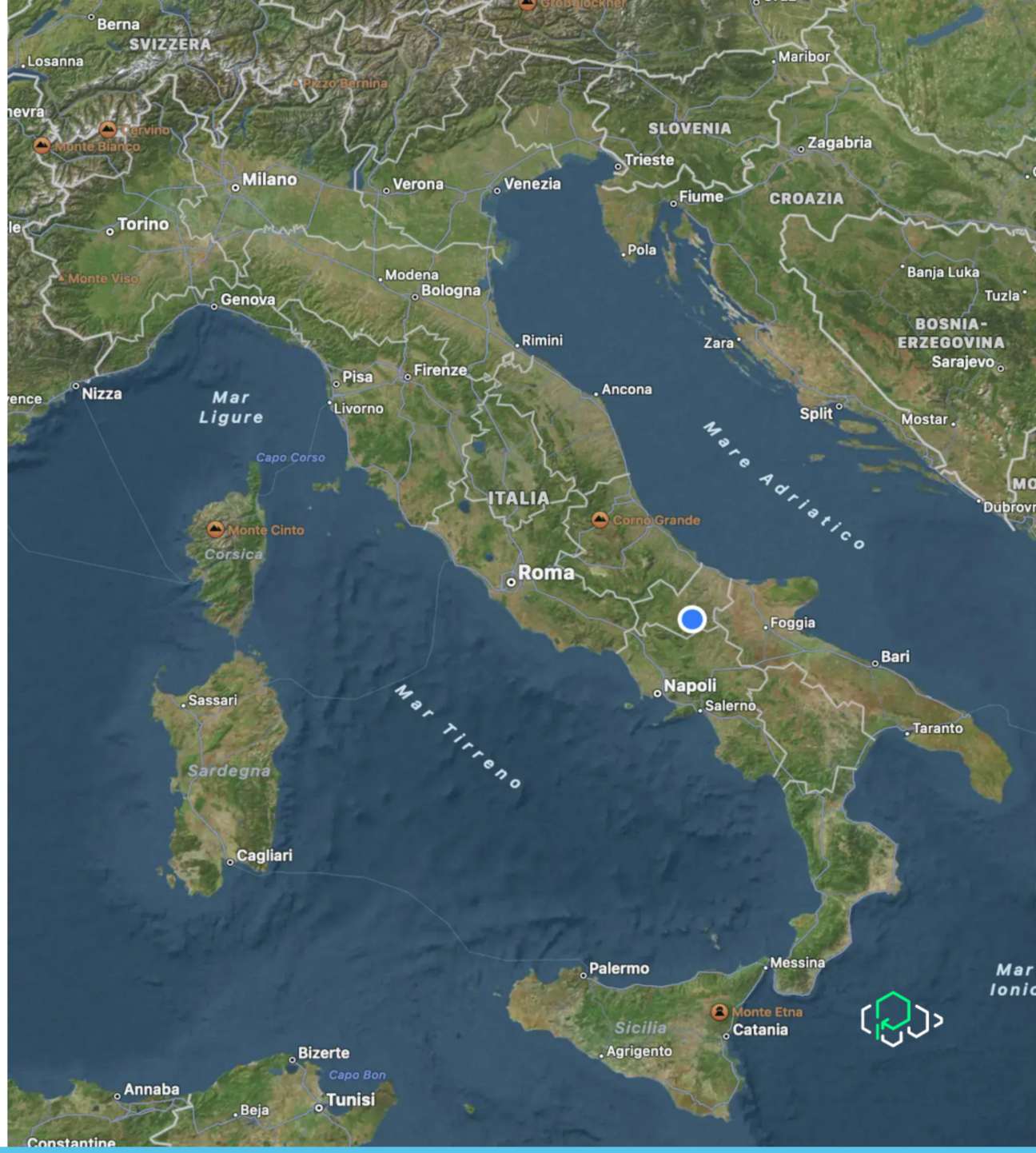
Don't count your chickens before they hatch.

Hello, I'm **Paolo**!

Node.js Technical Steering Committee Member

Platformatic Principal Engineer

`paoloinsogna.dev` `ShogunPanda` `p_insogna` `pinsogna`



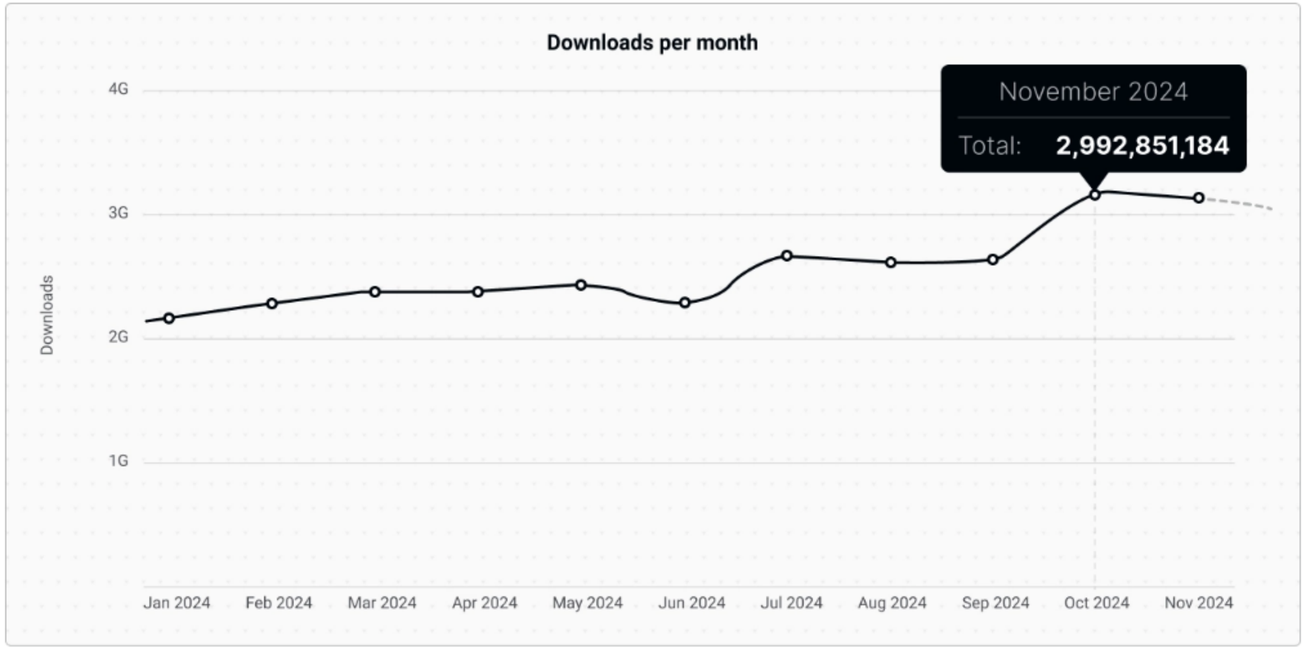
First of all, let's give credits!

This talk has originally been authored by Platformatic CTO and friend of mine, the only **Matteo Collina**.

Whatever goes wrong today, please complain directly to him on Twitter!

@matteocollina

I think we should start by looking at numbers.



30 Billion downloads per year

Usually we ready this all over the internet...

“ZYX will flip the default backend JavaScript runtime from Node.js”

Random excited user #1

… or, actually, something like this.

“XYZ will destroy Node.js”

Random excited user #2

This talk is about a question we are trying to answer...


Is Node.js dead yet?


NO
Period.

To all of you that "aim" to destroy other technologies...

...let me tell you a secret!

A technology born in 1959 is on the RAISE!

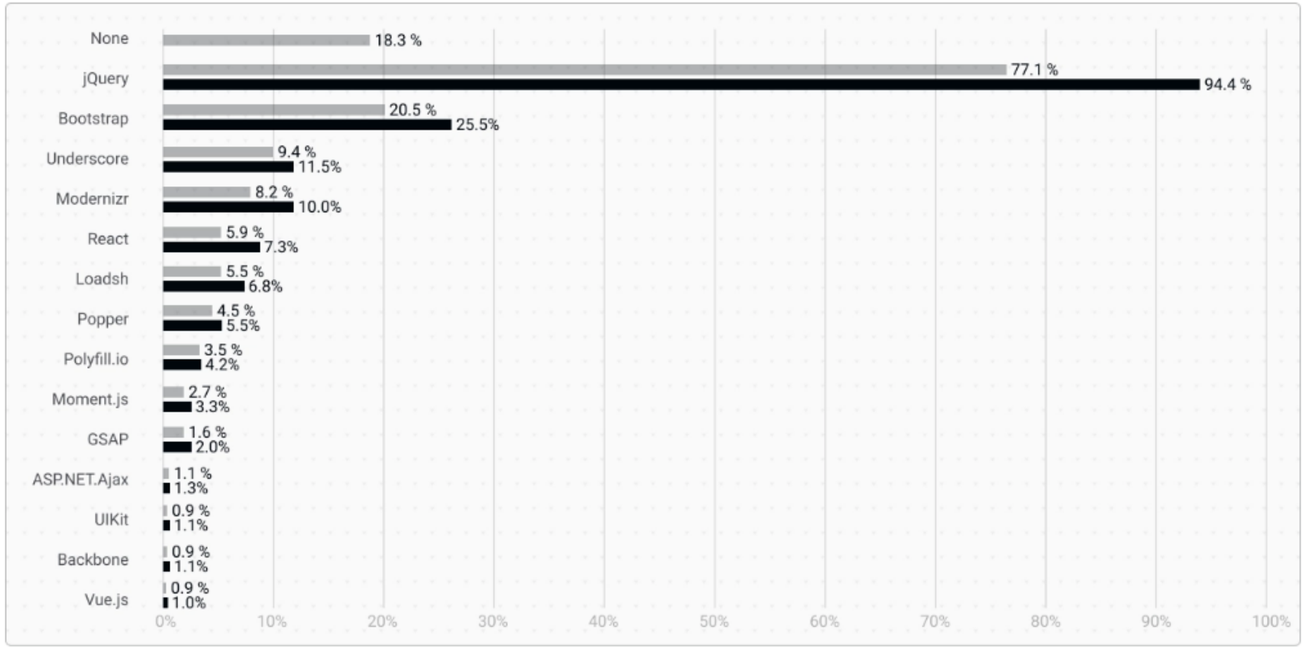
Have you have heard about **COBOL**?

FEB 2024	FEB 2023	CHANGE	PROGRAMMING LANGUAGE	RATINGS	CHANGES
15	18	▲		1.18 %	+ 0.42 %
16	15	▼		1.16 %	+ 0.23 %
17	33	⇈		1.07 %	+ 0.76 %
18	20	▲		1.05 %	+ 0.35 %
19	**30**	⇈		**1.01 %**	**+ 0.60 %**
20	16	⇊		0.99 %	+ 0.17 %

Source: **https://www.tiobe.com/tiobe-index/**

jQuery is used by 94.4% of the JS-enabled sites

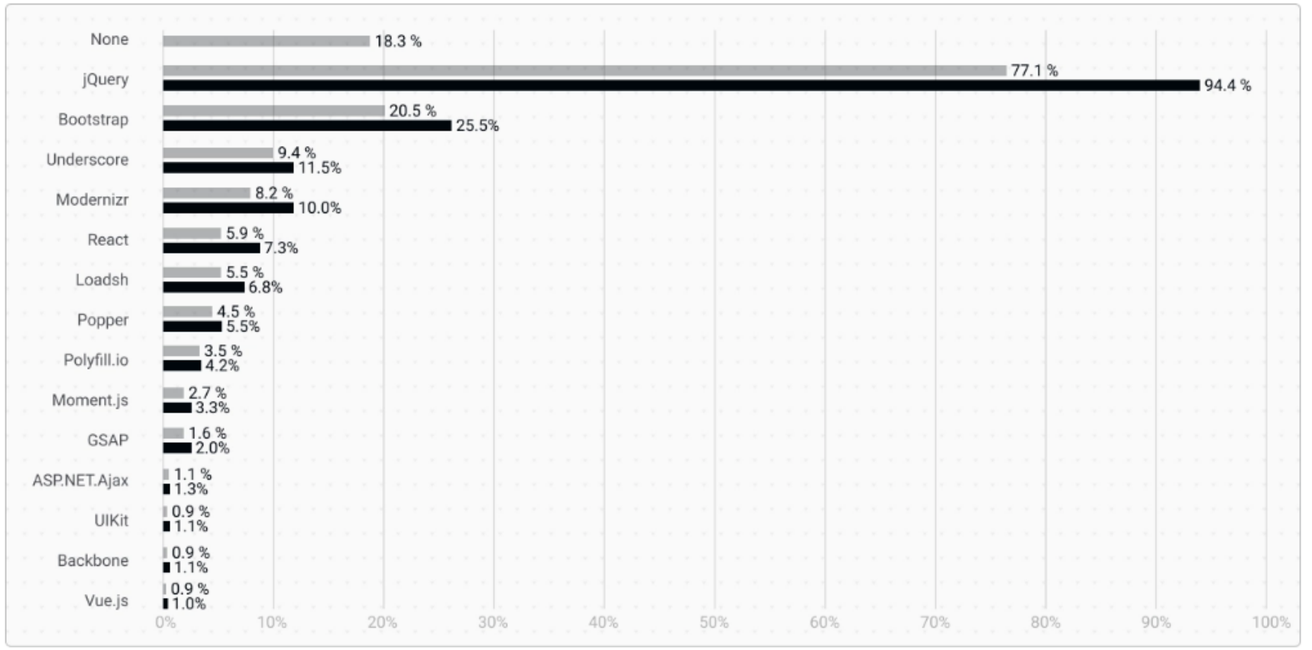
And v4 was just released!



Source: **https://w3techs.com/technologies/overview/javascript_library**

What about Node.js?

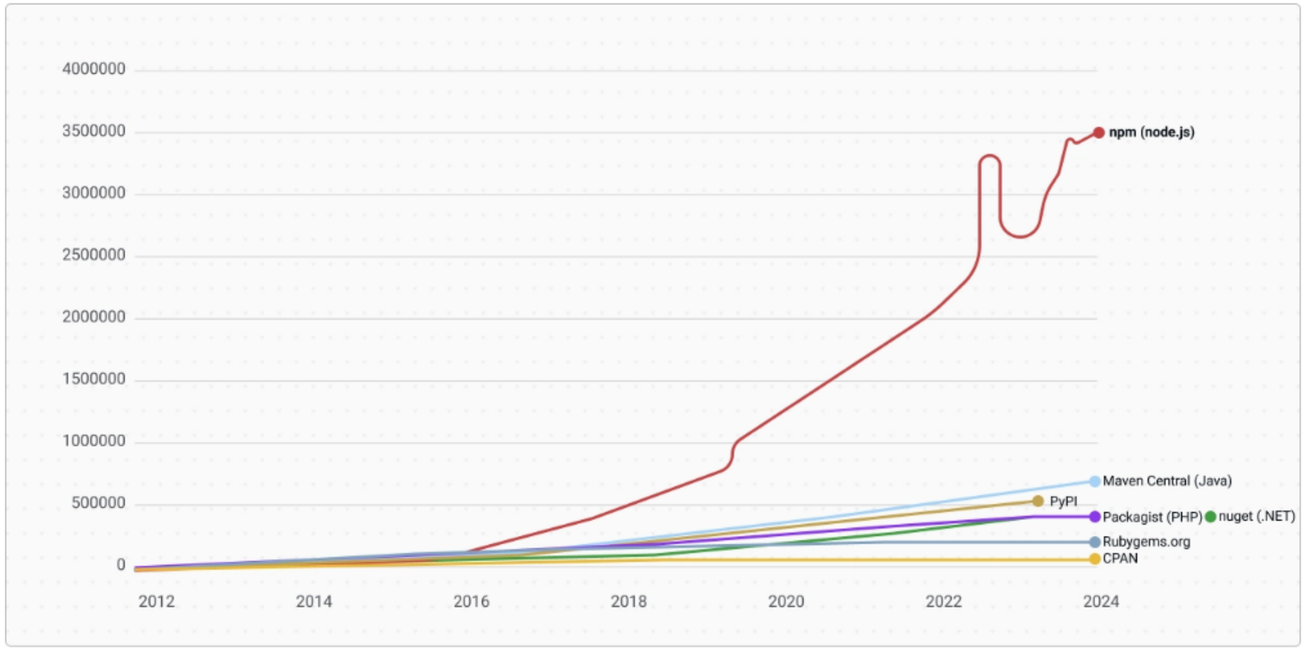
Node.js is the most popular technology according to StackOverflow



Source: **https://survey.stackoverflow.co/2023/#technology-most-popular-technologies**

No, bundling npm and Node.js was not a mistake

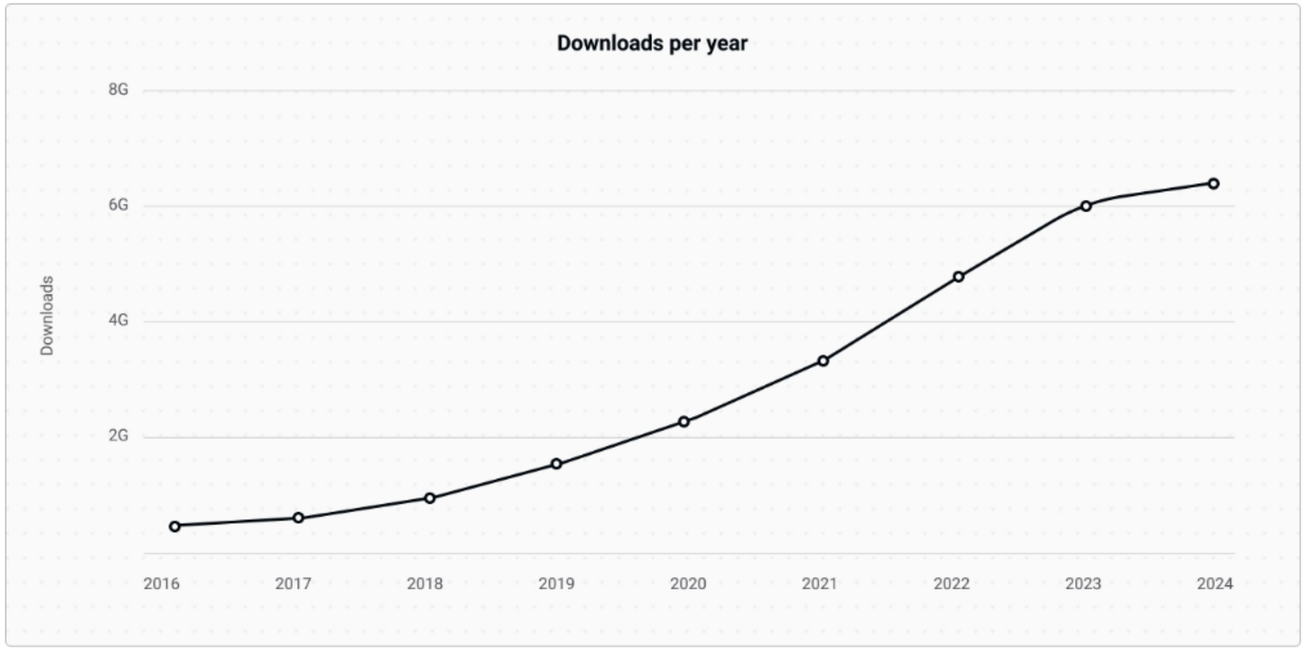
A combo that solve software reuse, at scale. No one else did.



Source: **http://www.modulecounts.com/**

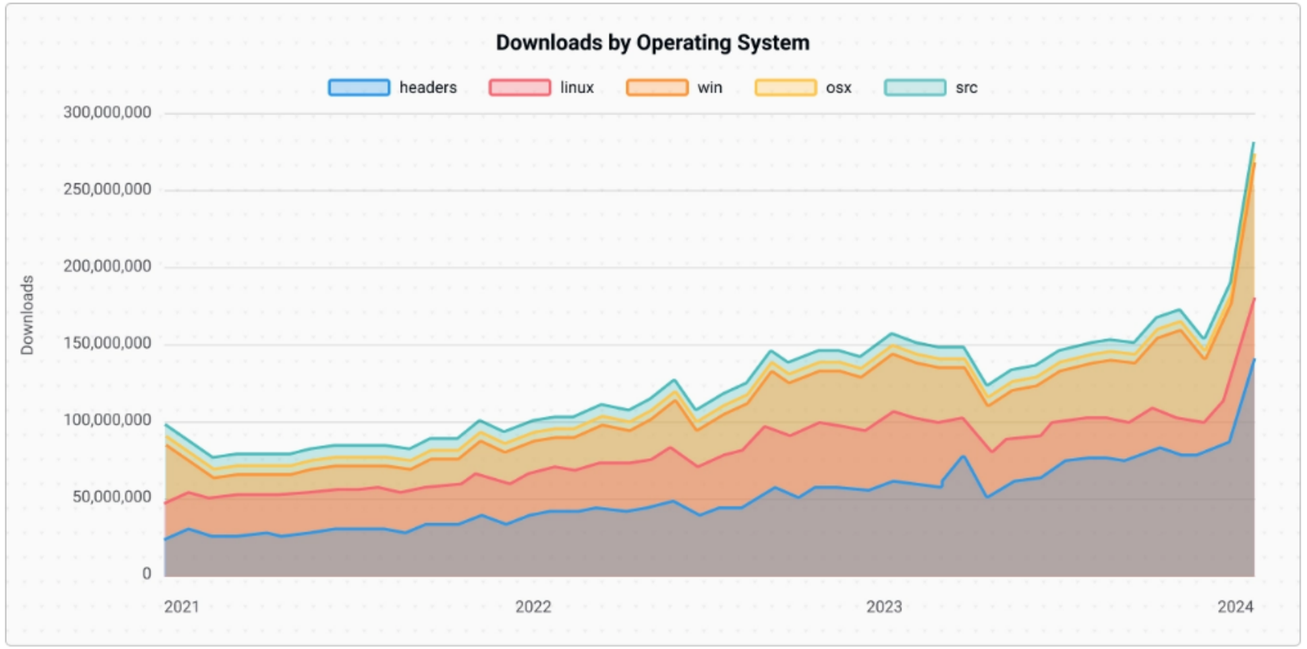
Module usage also grew

Downloads of readable-stream (4th most downloaded module on npm)



The "usage" of Node.js is growing roughly **+50% year over year**, or doubling every 2 years

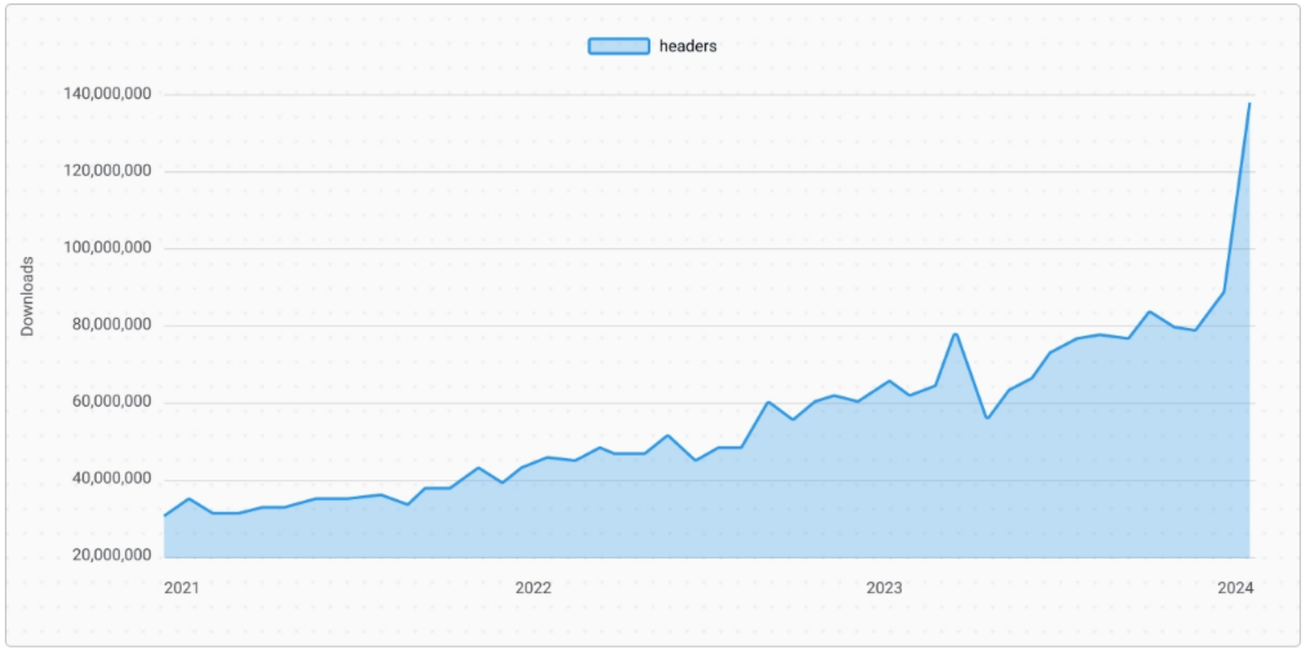
Half of Node.js downloads are the headers files



Source: **https://nodedownloads.nodeland.dev/**

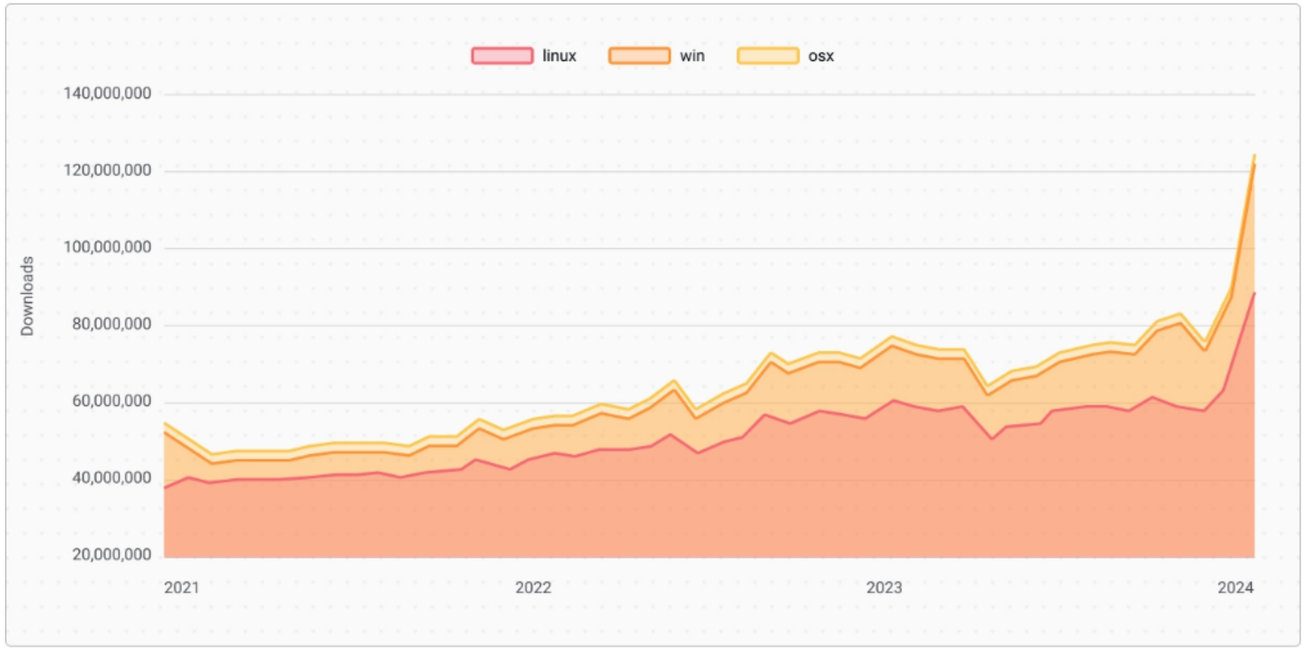
What are "headers" downloads?

Headers are downloaded whenever an "npm i" needs to compile binary addons.
Then they are cached to disk.



Downloads of the Node.js binary, per OS

We believe Linux is the default for CI. OSS projects often tests on WIndows for CI as well.



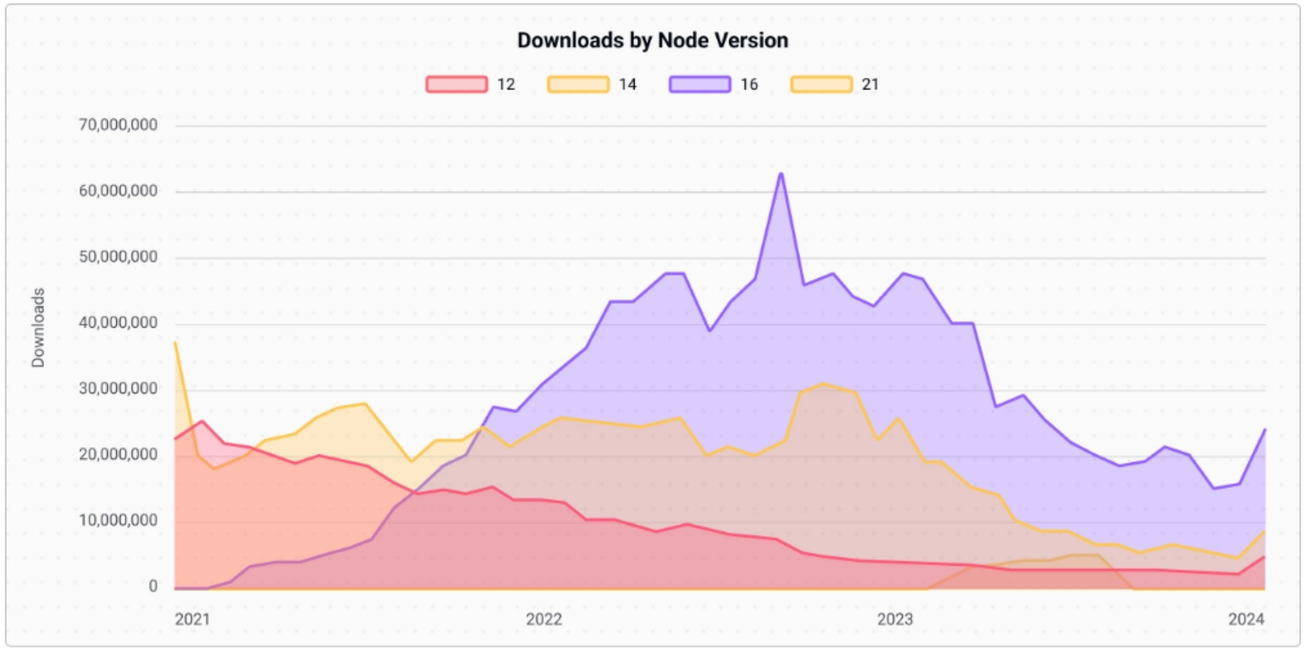
30

million monthly downloads in 2021

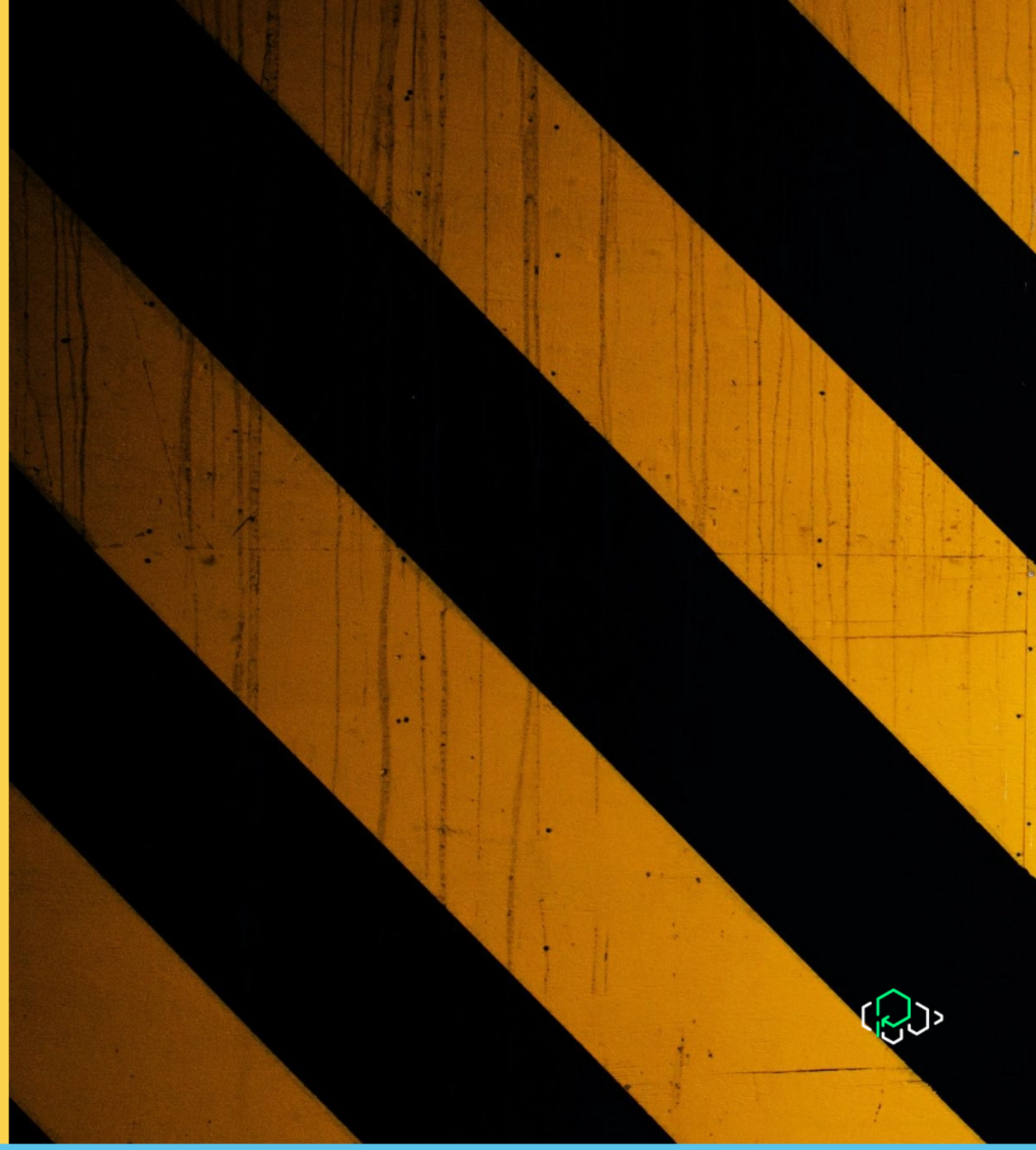
60

million monthly downloads in 2024

Node.js v16, v14, v12 are massively popular, while they all have KNOWN VULNERABILITIES

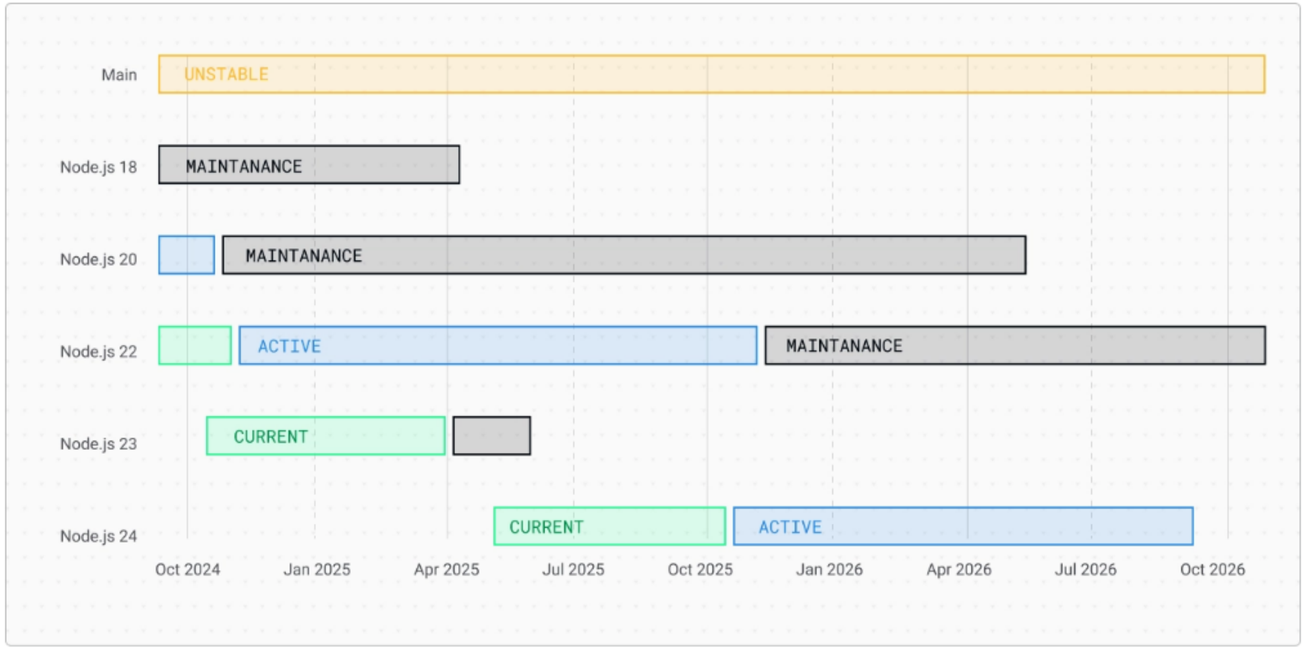


If you are not updating Node.js...

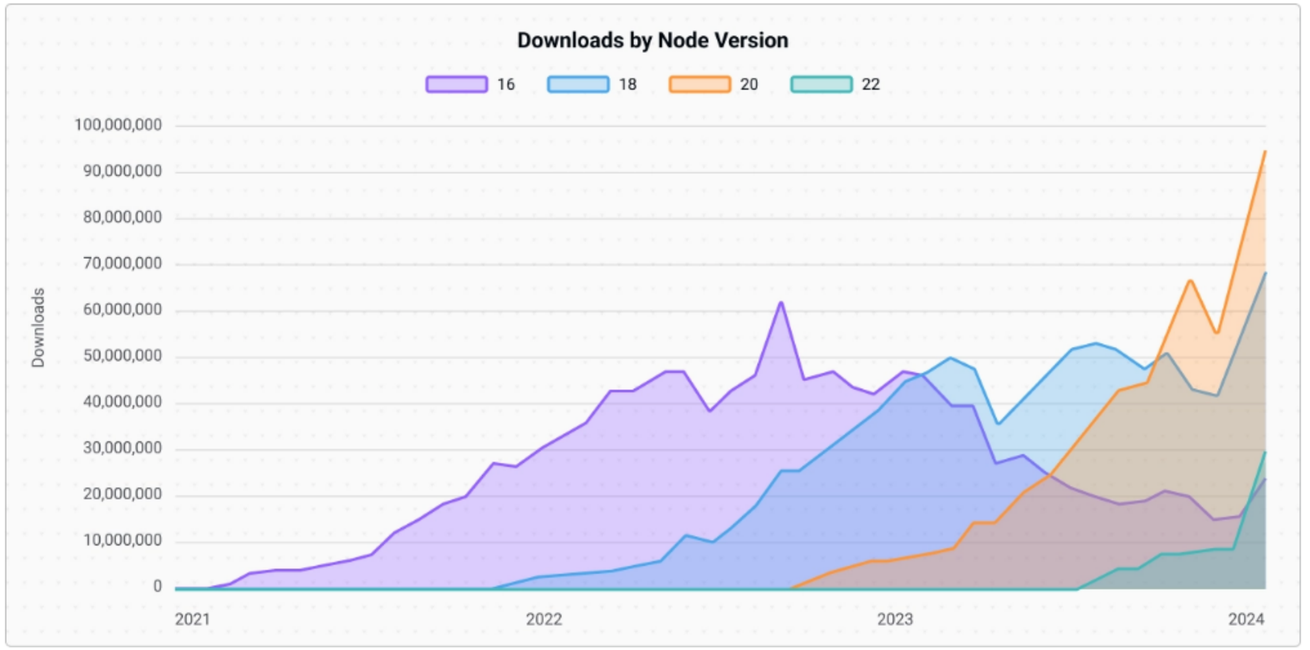


...you are putting yourself at risk!

Long Term Support (LTS) Schedule



Most teams update their Node.js version every 2 LTS releases



Organization Activity from Nov 2023 to Nov 2024

Source: https://next.ossinsight.io/analyze/nodejs?period=past_12_months#overview

19,917

Star Earned

+ 6.61 %

5,261

Pull Requests

+ 8.72%

28,438

Reviews

- 2.71 %

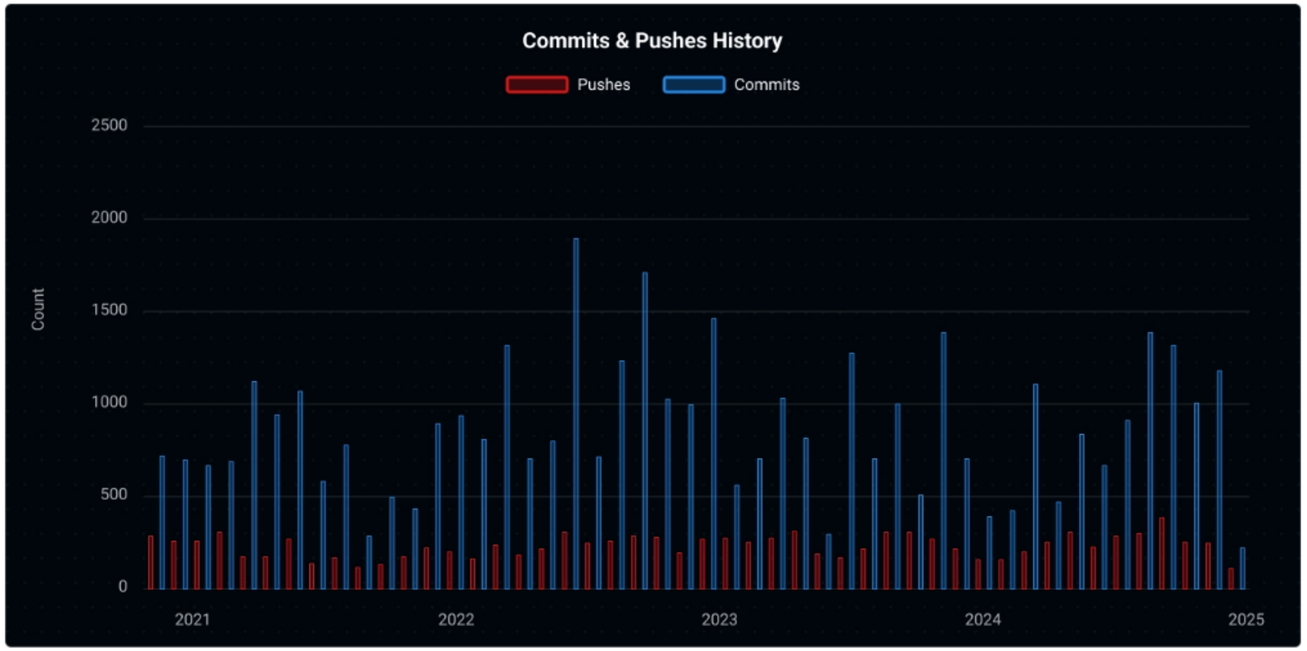
4,499

Issues

- 3.69 %

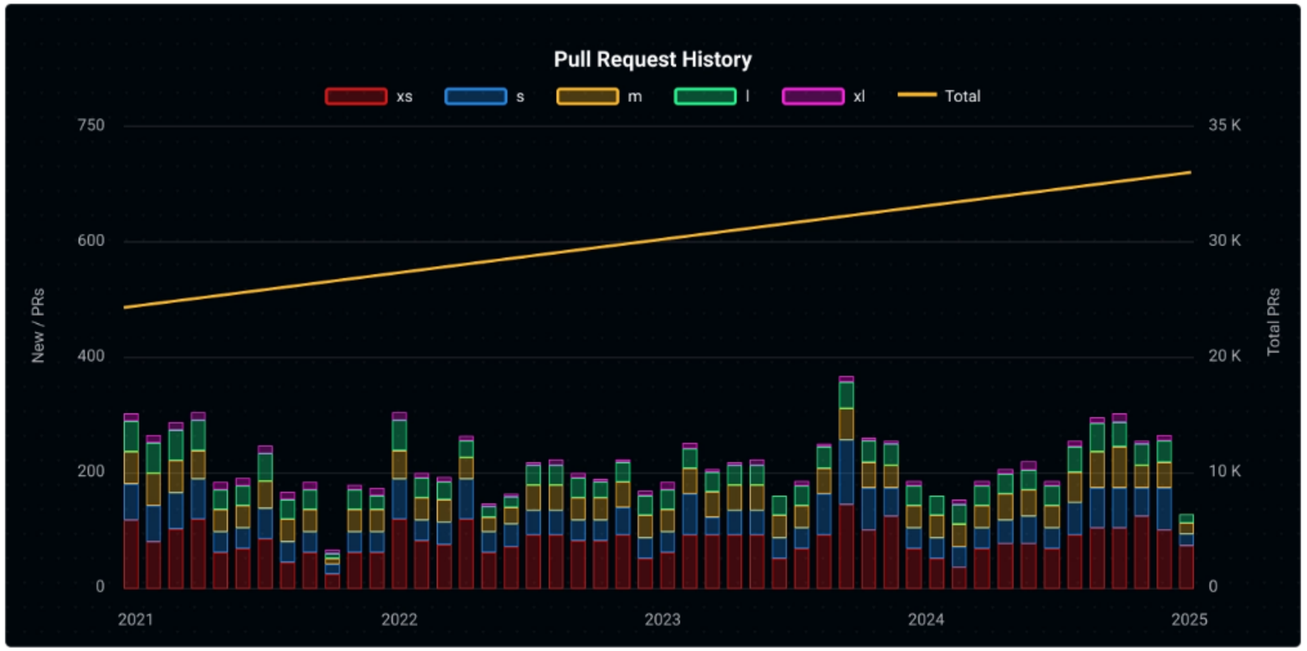
Commits & Pushes to Node.js Core

Source **https://ossinsight.io/analyze/nodejs/node**



The number of pull requests is (mostly) stable

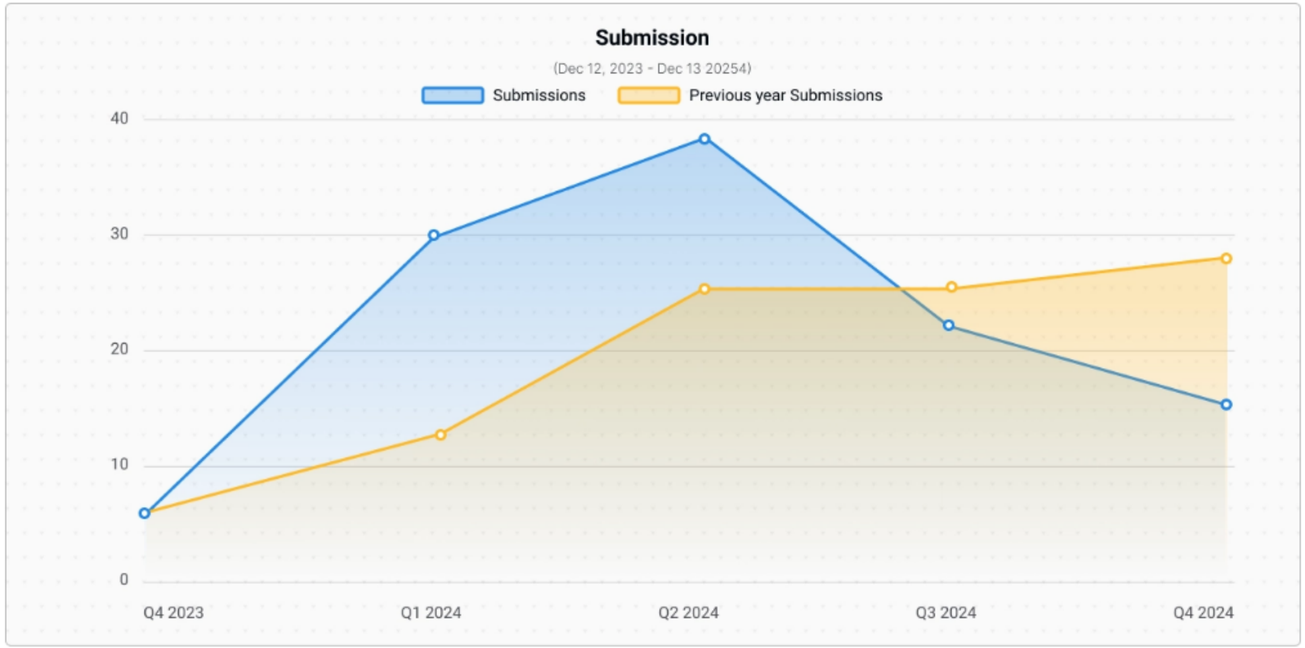
Source https://ossinsight.io/analyze/nodejs/node



We work hard to keep you safe!

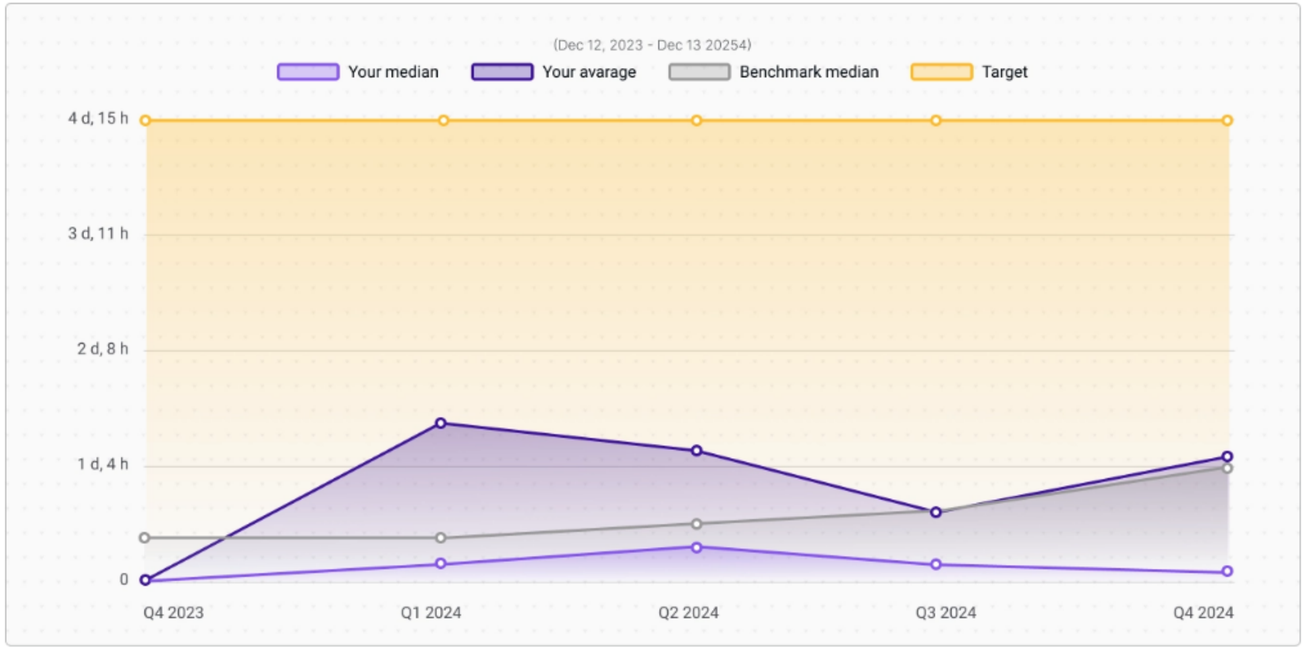
Node.js Security Submissions

Submissions are not turned into vulnerabilities, but they need to be handled by the team



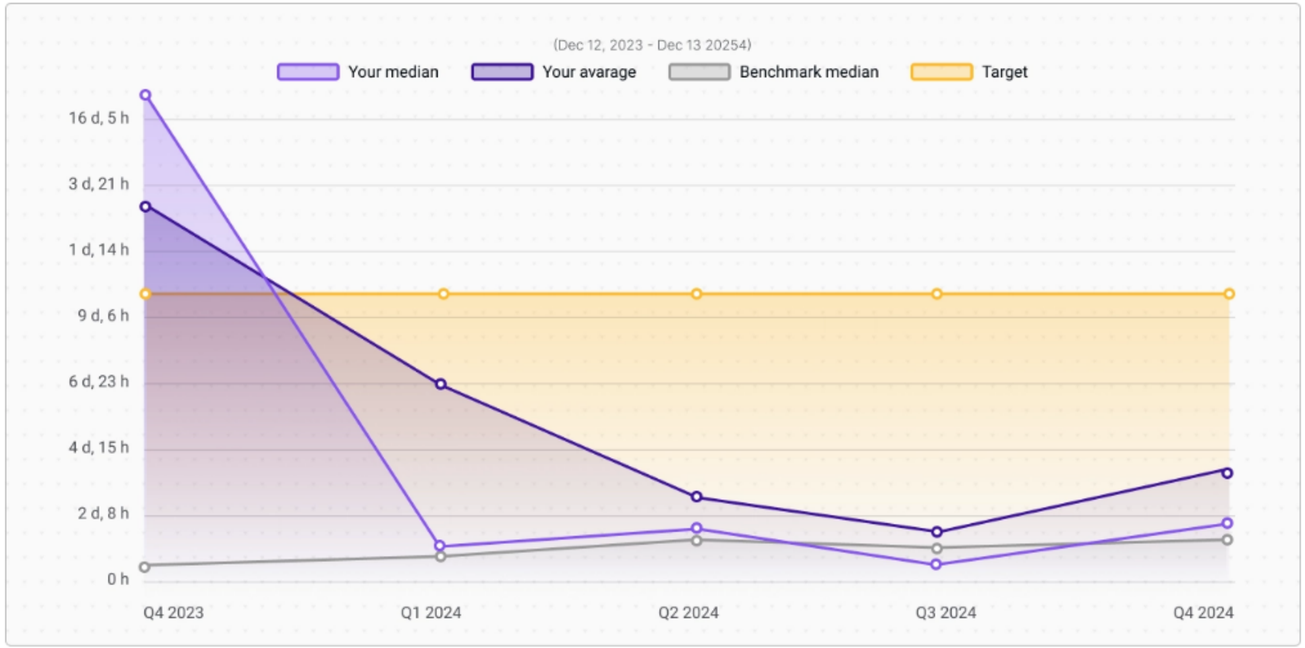
Average time to first response

Submissions are not turned into vulnerabilities, but they need to be handled by the team



Average time to triage

Submissions are not turned into vulnerabilities, but they need to be handled by the team

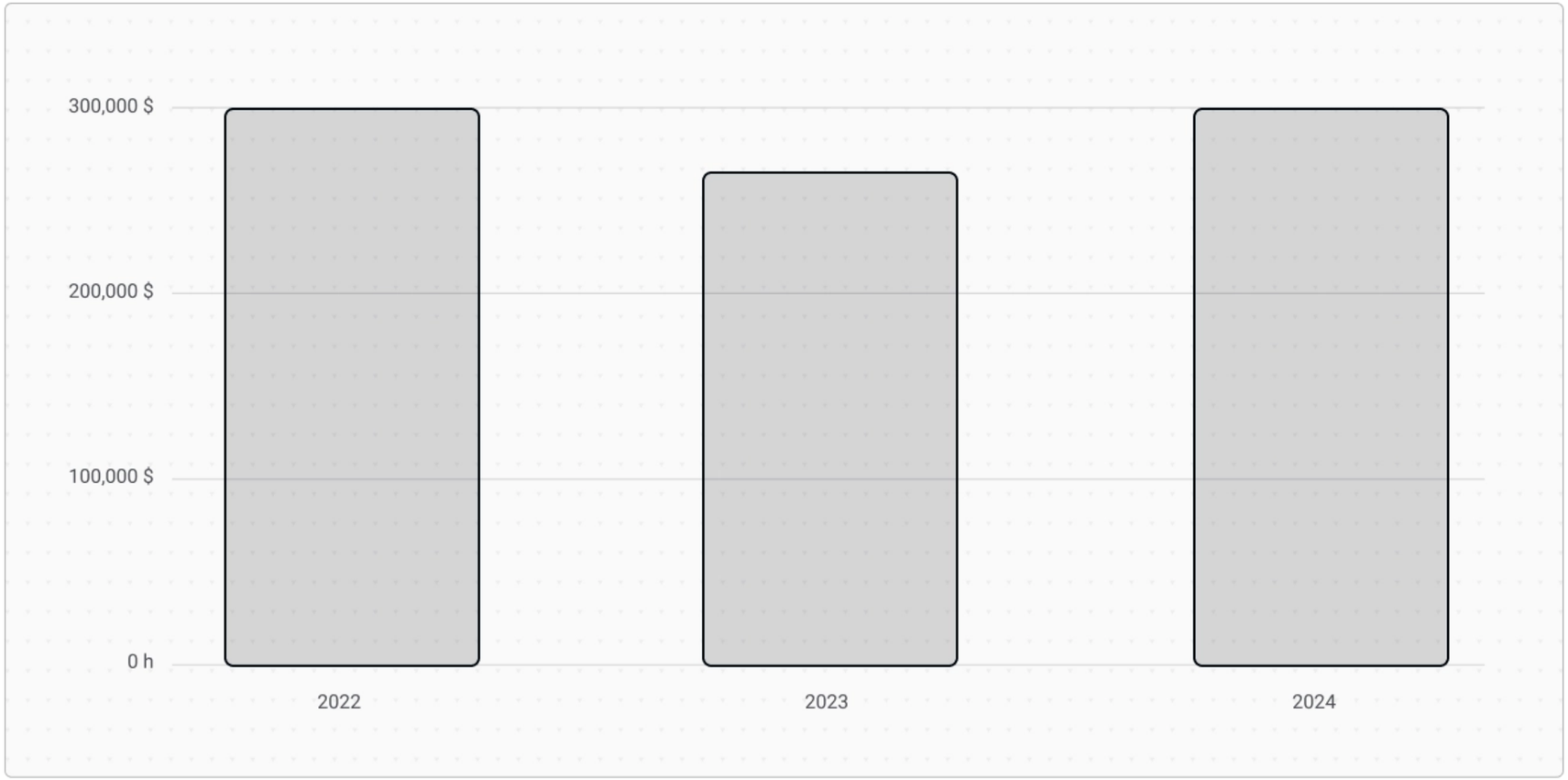


Node.js was one of the first project sponsored by

Alpha-Omega is an associated project of the OpenSSF, established in February 2022, funded by Microsoft, Google, and Amazon, and with a **mission to protect society by catalyzing sustainable security improvements** to the most critical open source software projects and ecosystems.

The project aims to build a world where critical open source projects are secure and where security vulnerabilities are found and fixed quickly.

Total funding for Security work



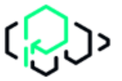
What did we ship in the last few years?

ESM

```
// addTwo.mjs
function addTwo(num) {
    return num + 2
}

export { addTwo }

// app.mjs
import { addTwo } from './addTwo.mjs'

// Prints: 6
console.log(addTwo(4))
```

Threads

```
import {
  Worker,
  isMainThread,
  setEnvironmentData,
  getEnvironmentData,
} from 'node:worker_threads'

if (isMainThread) {
  setEnvironmentData('Hello', 'World!')
  const worker = new Worker(import.meta.filename)
} else {
  console.log(getEnvironmentData('Hello')) // Prints 'World!'
}
```



Fetch

```
const res = await fetch('https://example.com/data')
const json = await res.json()
console.log(json)
```

Web Platform Compatibility

- `fetch()` and Response
- Web Streams
- Web Crypto
- `FormData`
- `structuredClone()`
- `TextEncoder` and `TextDecoder`
- `Blob` and `File`
- `EventTarget`
- Initial `Symbol.dispose`

... and there's more!

Promises

```
import { readFile } from 'node:fs/promises'

try {
  const filePath = new URL('./package.json', import.meta.url)
  const contents = await readFile(filePath, { encoding: 'utf8' })
  console.log(contents)
} catch (err) {
  console.error(err.message)
}
```

node: **only core modules**

```
import assert from 'assert'
import test  from 'test'
```

```
import assert from 'node:assert'
import test  from 'node:test'
```

Watch mode

Few simple CLI flags

`--watch` command line flag, `--watch-path` customizes watched files

Watches imported files and restarts Node.js when they change

Works with test runner to rerun tests

Limited platform support

macOS and Windows only

AsyncLocalStorage

Without it, we would not have had **React Server Components**!

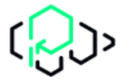
```
import http from 'node:http'
import { AsyncLocalStorage } from 'node:async_hooks'

const asyncLocalStorage = new AsyncLocalStorage()

function logWithId(msg) {
  const id = asyncLocalStorage.getStore()
  console.log(`${id !== undefined ? id : '-'}: `, msg)
}

let idSeq = 0
http.createServer((req, res) => {
  asyncLocalStorage.run(idSeq++, () => {
    logWithId('start')

    // Imagine any chain of async operations here
    setImmediate(() => {
      logWithId('finish')
      res.end()
    })
  })
}).listen(8080)

http.get('http://localhost:8080')
http.get('http://localhost:8080')
```

WebCrypto

```
import { webcrypto } from 'node:crypto'

const subtle = webcrypto.subtle

async function main() {
  const key = await webcrypto.subtle.generateKey(
    { name: 'HMAC', hash: 'SHA-256', length: 256 },
    true,
    [ 'sign', 'verify' ]
  )

  const enc = new TextEncoder()
  const message = enc.encode('I love cupcakes')
  const sign = await subtle.sign({ name: 'HMAC' }, key, message)

  return Buffer.from(sign).toString('hex')
}

console.log(await main())
```

util.parseArgs()

```
import { parseArgs } from 'node:util'

const args = ['-f', '--bar', 'b']

const options = {
  foo: {
    type: 'boolean',
    short: 'f'
  },
  bar: {
    type: 'string'
  }
}

const { values, positionals } = parseArgs({ args, options })

console.log(args)
// Prints: [Object: null prototype] { foo: true, bar: 'b' } []
```



Single Executable Applications

Inject application code into Node.js binary.
Originally developed by Postman Labs

Single CommonJS file is currently supported
Binaries can be distributed without Node/npm installation.

Node.js project offers a helper module named postject
Similar tools exist. They should work but have not been tested.

Permission System

Simple to use CLI flags

`--experimental-permissions`, `--allow-fs-read`, `--allow-fs-write`, `--allow-child-process`, and `--allow-worker`.

`process.permissions.has` API

`process.permissions.deny()` has been removed but may be added again in the future.

Test Runner

Supports subtests, skipping, only, lifecycle hooks

`node:test` module, `--test` flag, `test` on npm.

Supports function and timer mocking.

Module mocking soon.

Code coverage via `--experimental-test-coverage`

Reporters via `--test-reporter` and `--test-reporter-destination`.

Good reporting defaults

Defaults to **`spec`** or **`tap`** on stdout based on TTY.

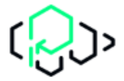
Test Runner

```
import assert from 'node:assert'
import test from 'node:test'

test('synchronous passing test', (t) => {
  // This test passes because it does not throw an exception.
  assert.strictEqual(1, 1)
})

test('asynchronous failing test', async (t) => {
  // This test fails because the Promise returned by the async
  // function is rejected.
  assert.strictEqual(1, 2)
})
```

WebSocket

Thanks **Matthew Aitken** for implementing this!

```
shogun@panda:~/example$ node
Welcome to Node.js v22.12.0.
Type ".help" for more information.
> WebSocket
[class WebSocket extends EventTarget] {
  CONNECTING: 0,
  OPEN: 1,
  CLOSING: 2,
  CLOSED: 3
}
```

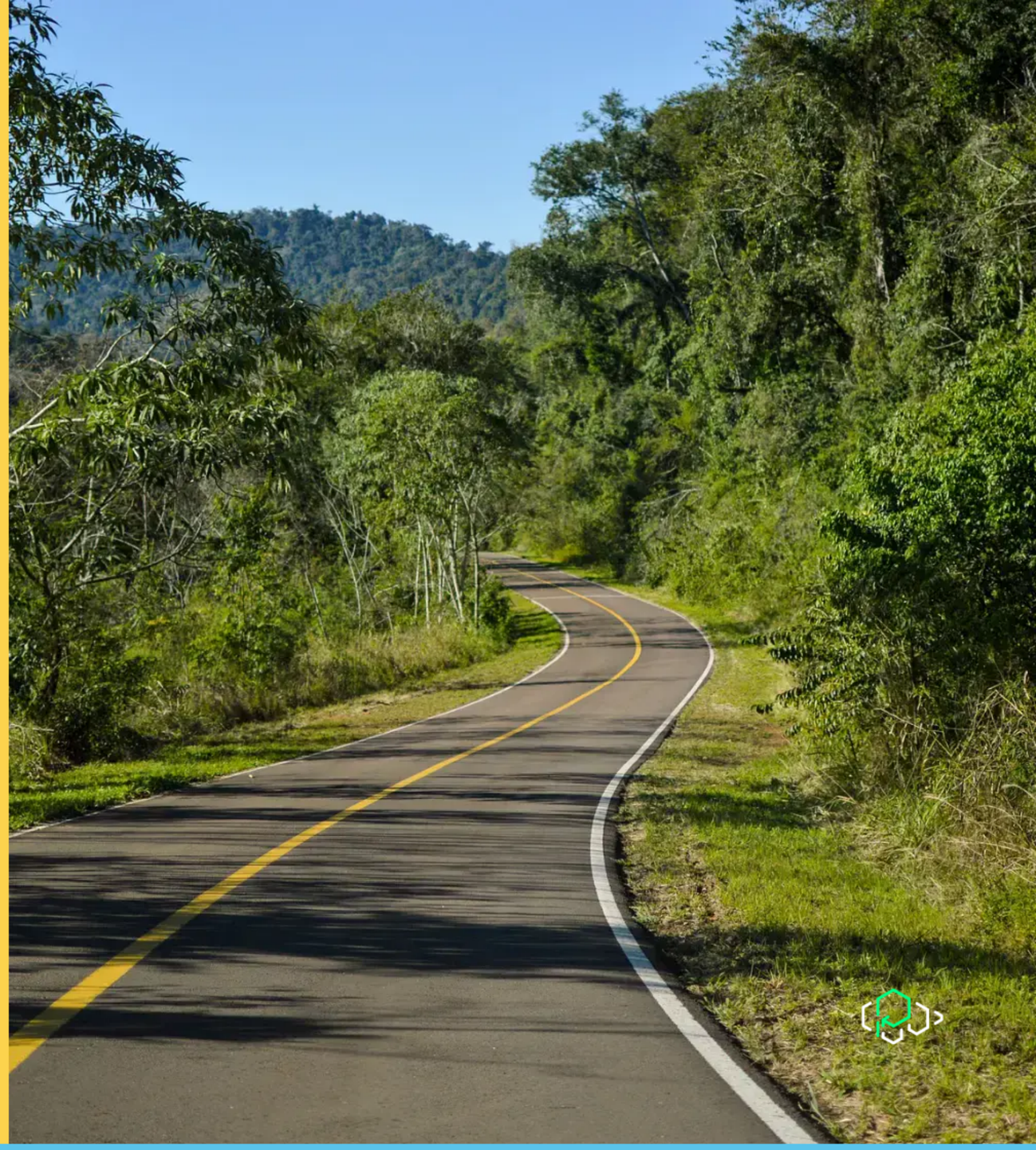


Are you using the latest Node.js features?

Or are you still stuck in 2015?

What's coming?

Let’s look at the two
most exciting things in v22!



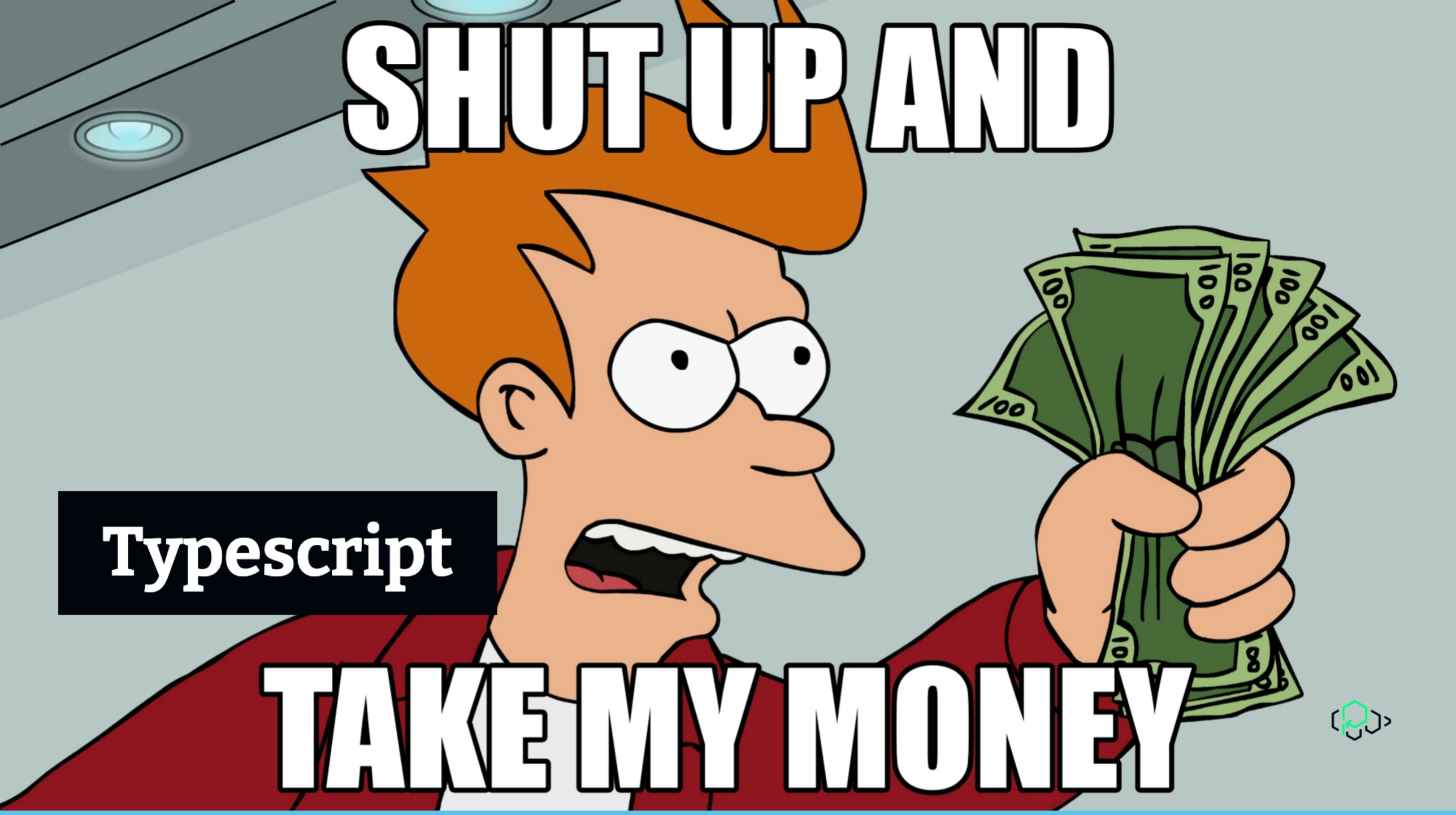

SHUT UP AND
require(esm)
.mjs is not needed anymore
TAKE MY MONEY

require(esm)

Learn more on Joyee's blog:
https://joyeecheung.github.io/blog/2024/03/18/require-esm-in-node-js/

Thanks to **Geoffrey Booth** for remove the need of `.mjs`.

```
'use strict'

const { answer } = require('./esm.mjs')
console.log(answer)
```


SHUT UP AND
Typescript
TAKE MY MONEY

Automatic TypeScript support

Thanks **Marco Ippolito** for implementing this!

```ts
// hello.ts
import {
  setTimeout as sleep
} from 'node:timers/promises'

function hello(who: string) {
  console.log(`Hello ${who}!`)
}

await sleep(1000)
hello('world')
```

```
shogun@panda:~/example$ node hello.ts
(node:30248) ExperimentalWarning: Type Stripping is
an experimental feature and might change at any time
(Use `node --trace-warnings ...` to show where
the warning was created)
Hello world!
```

Node.js is not always relaxing...

...because we need you!

Project Governance

OpenJS Foundation

Node.js core collaborators maintain the nodejs/node GitHub repository

@nodejs/collaborators have commit access to the nodejs/node repository and access to the Node.js continuous integration (CI) jobs.

Both collaborators and non-collaborators may propose changes to the Node.js source code. The mechanism to propose such a change is a GitHub pull request (PR).

Collaborators review and merge (*land*) pull requests.

The review process

Two collaborators must approve a pull request before the pull request can land.
(*One collaborator approval is enough if the pull request has been open for more than 7 days.*)

Approving a pull request indicates that the collaborator accepts responsibility for the change.
Approval must be from collaborators who are not authors of the change.

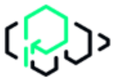
If a collaborator opposes a proposed change, then the change cannot land.
The exception is if the TSC votes to approve the change despite the opposition.

Involving the TSC is usually unnecessary: often, discussions or further changes **result in collaborators removing their opposition**.

The Node.js Technical Steering Committee

The TSC voting members are responsible for all technical development within the Node.js project, including:

- Setting release dates.
- Release quality standards.
- Technical direction.
- Project governance and process.
- GitHub repository hosting.
- Conduct guidelines.
- Maintaining the list of additional Collaborators.
- Development process and any coding standards.
- Mediating technical conflicts between Collaborators or Foundation projects.



In case of disagreements, the TSC votes

TSC members can be either regular members or voting members.

Regular members can attend meetings and participate in TSC discussions, but do not vote. Voting members can do everything regular members can do, and also have the ability to cast votes when consensus is not reached on an issue.

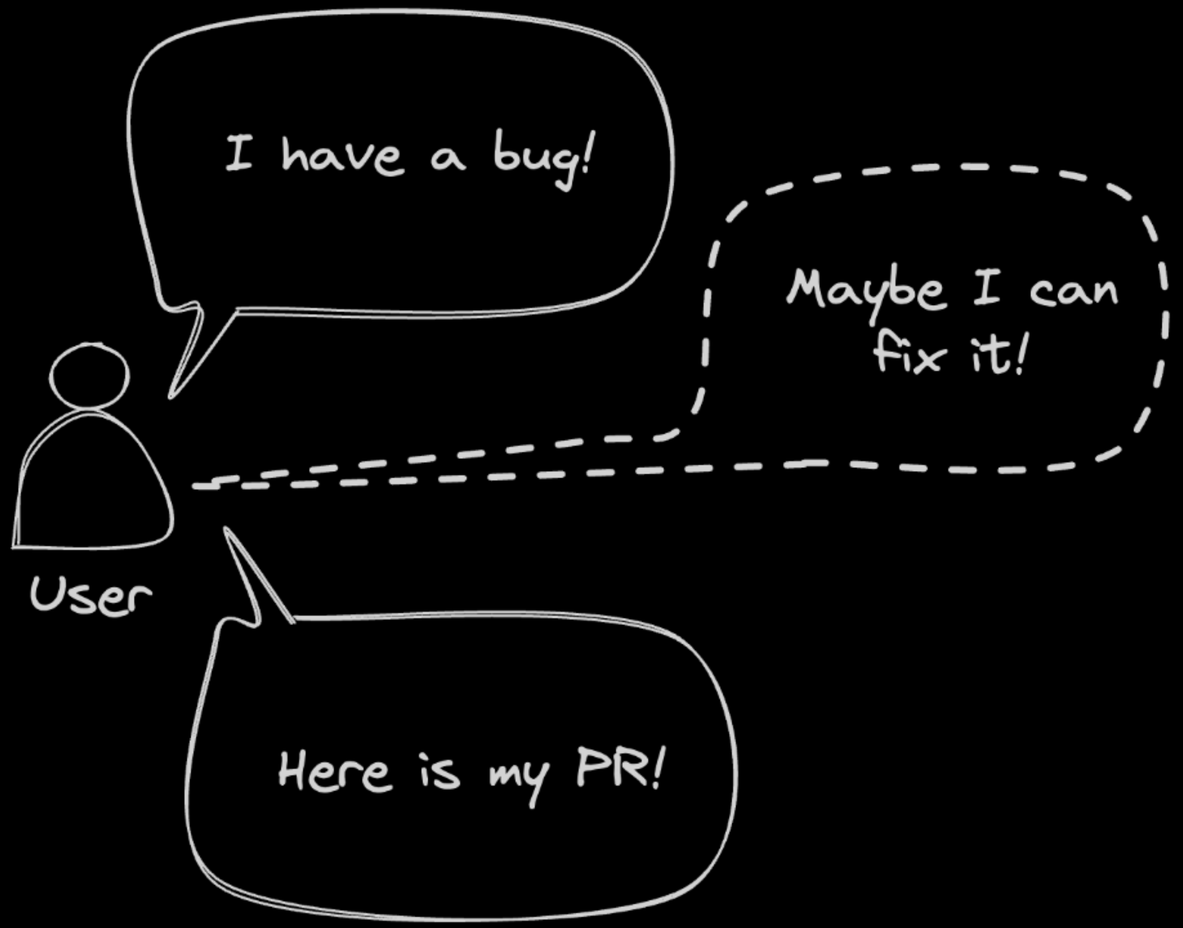
TSC memberships are not time-limited and there is no maximum size of the TSC. The TSC must have at least four voting members.

No more than one-fourth of the TSC voting members may be affiliated with the same employer at any time.


No one can control Node.js

We all have to
to COMPROMISE
to achieve our
objectives

Do you want to have a say in the future of Node.js?



Start contributing!

One last thing™

“It is not how old you are but how you are old.”

Jules Renard

Thank you!

Paolo Insogna

Node.js TSC, Principal Engineer

@p_insogna

paolo.insogna@platformatic.dev

Platformatic