An introduction to node

POWERUp 2025

About Michael Dawson

Node.js lead for Red Hat and IBM

Active Node.js community member

- Node.js Collaborator, Node.js Technical Steering Committee,
- Active in a number of Working group(s)

Active OpenJS Foundation member

- Voting Cross Project Council Member
- Community Director 2020-2022

Twitter: @mhdawson1

GitHub: @mhdawson

Linkedin: https://www.linkedin.com/in/michael-dawson-6051282

Agenda

- What is Node.js
- Why Node.js?
- Node.js components
- Key Node.js fundamentals
- Node.js Reference Architecture
- Platform Support
- Where to get Node.js

What is it?

- JavaScript != Java

- Node.js = Server-side JavaScript
  - Event-Oriented
  - Non-blocking
  - Asynchronous

History - over 15 years of growth/maturity

- The early years
  - 2009 – written by Ryan Dhal
  - 2010 - npm, Joyent sponsors Node.js
  - 2011 – windows support added
  - 2012 – 2014 – Hand over to Isaac Schlueter, then Timothy J. Fontaine
  - 2014 – io.js fork
  - 2015 – Node.js Foundation created, Node.js 4.x unites io.js/node.js 0.12.x lines
- 2015 -2025
  - 2 Majors every year
  - LTS every october
  - 2019 - Node.js Foundation merged into OpenJS foundation

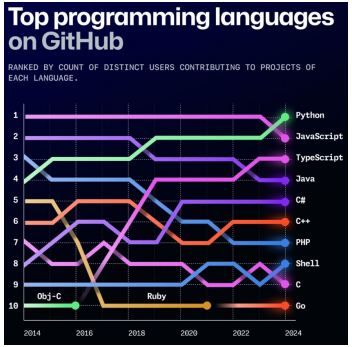
Widely Used

* Even after docker activities pushing people to cache

- Massive usage in last year
  - Over 1B downloads from node.org
  - Almost 800M docker pulls* dockerhub

- Tops recent surveys

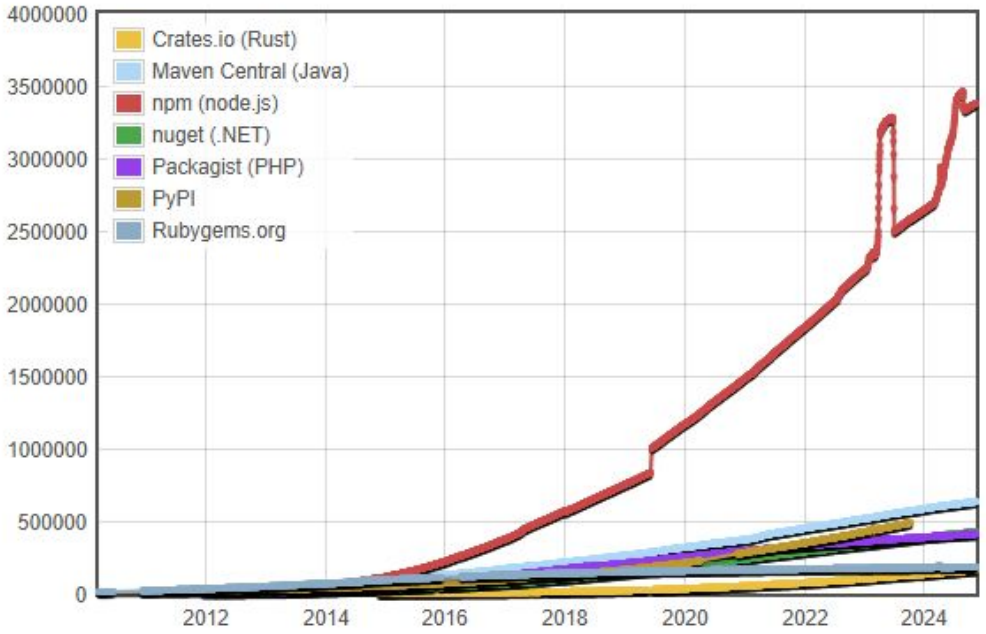
https://survey.stackoverflow.co/2024/technology/



2 - JavaScript
3 - typeScript
Notes that combined still higher than Python

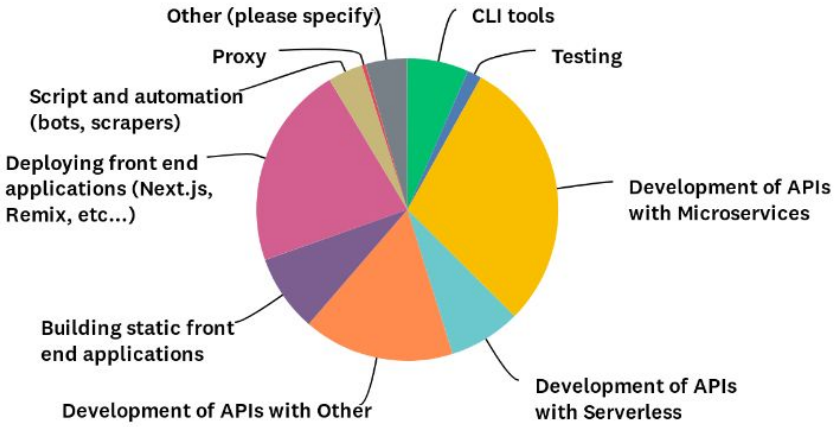
https://github.blog/news-insights/octoverse/octoverse-2024/

Broad Ecosystem



http://www.modulecounts.com/

What is it used for?



Use Cases

- 53% back end services
- 22% deploying front end apps
- 21% building, and scripting

https://github.com/nodejs/next-10/blob/main/surveys/2024-04

Why? - Productivity and Performance

Productivity

'Took ½ the time with less people, 33% fewer lines of code, 40% fewer files' - **PayPal**

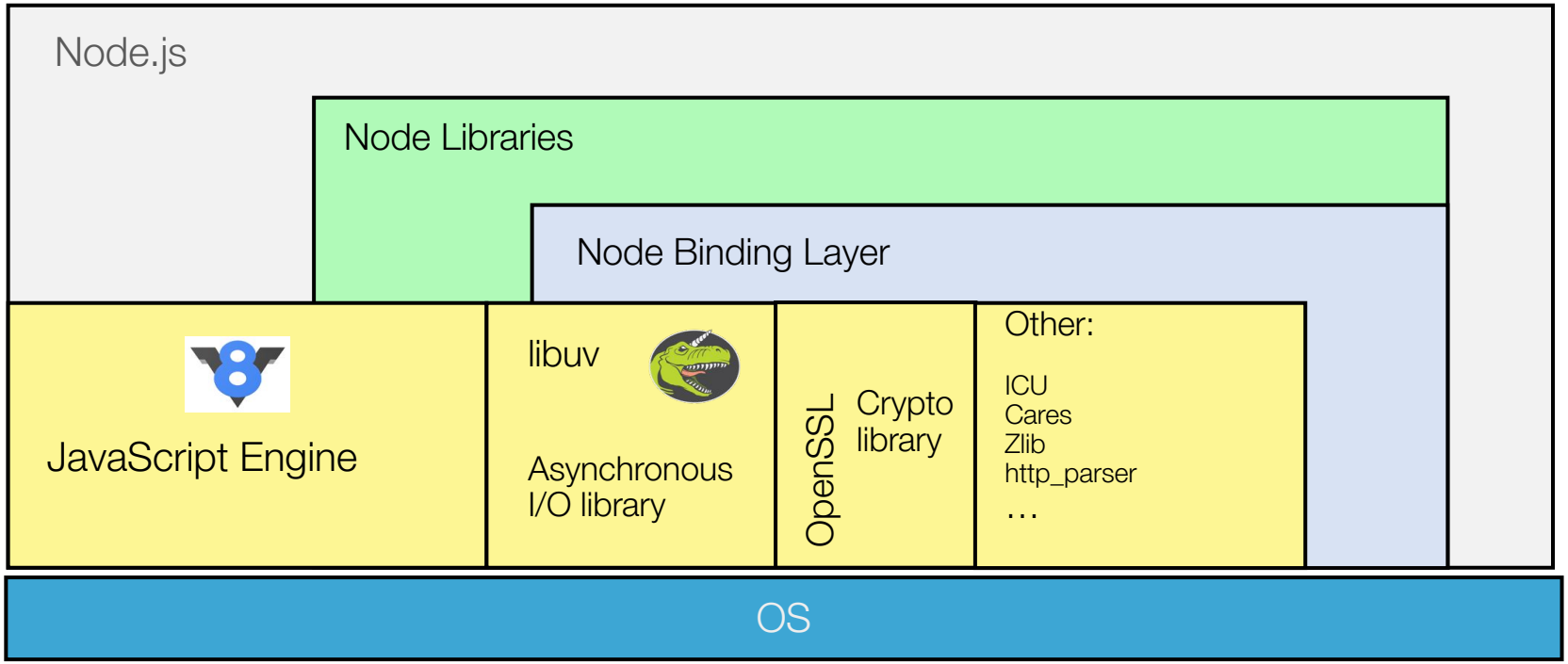
'We're used to working in JavaScript all day long. Having Node just makes it feel like a very natural extension of our work environment' - **Netflix**

Performance

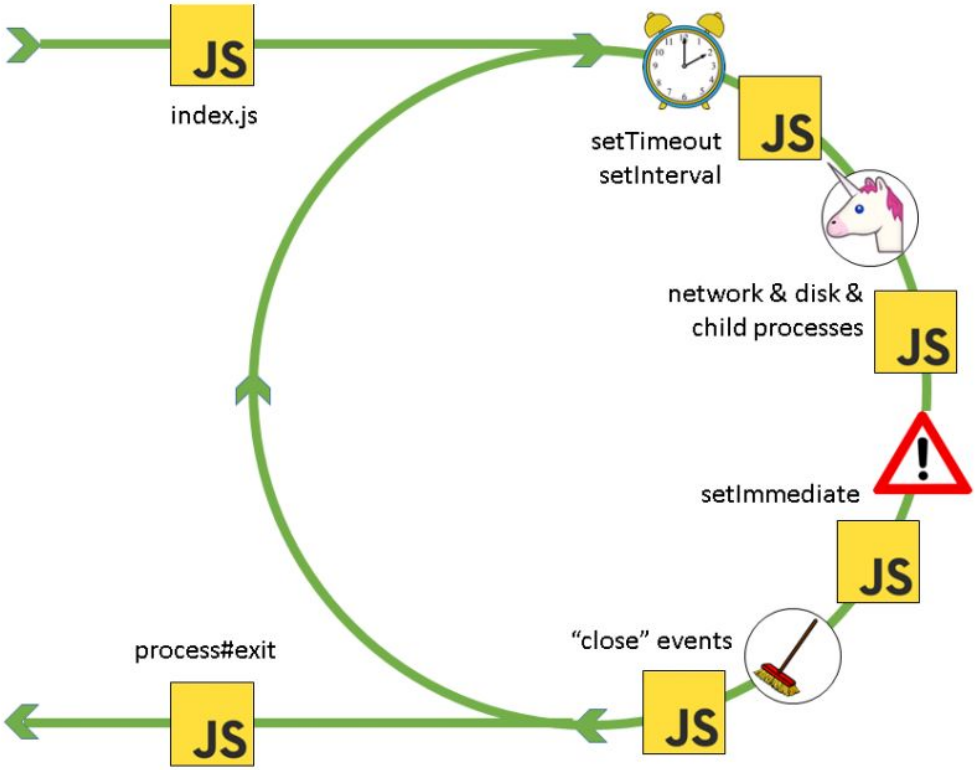
'Double the number of request per second, response time 35% lower' - **PayPal**

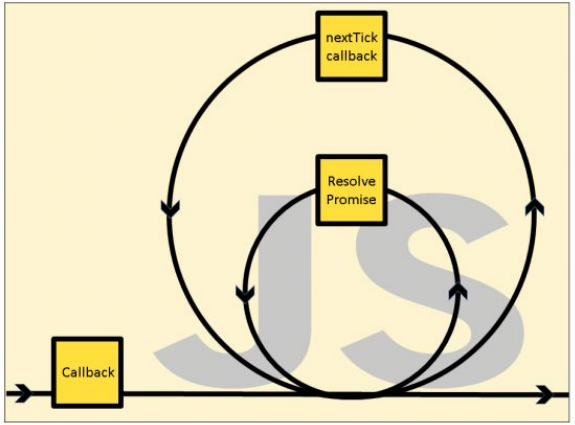
'Reduced page load time by 50%' - **Groupon**

Building blocks

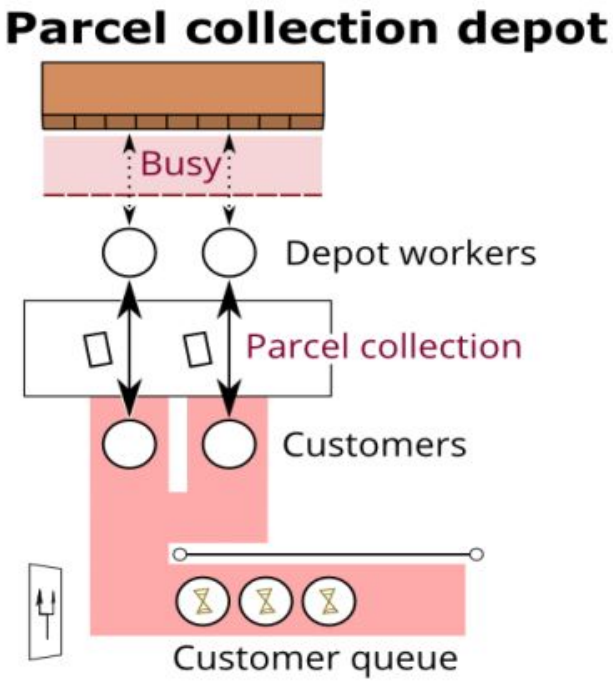


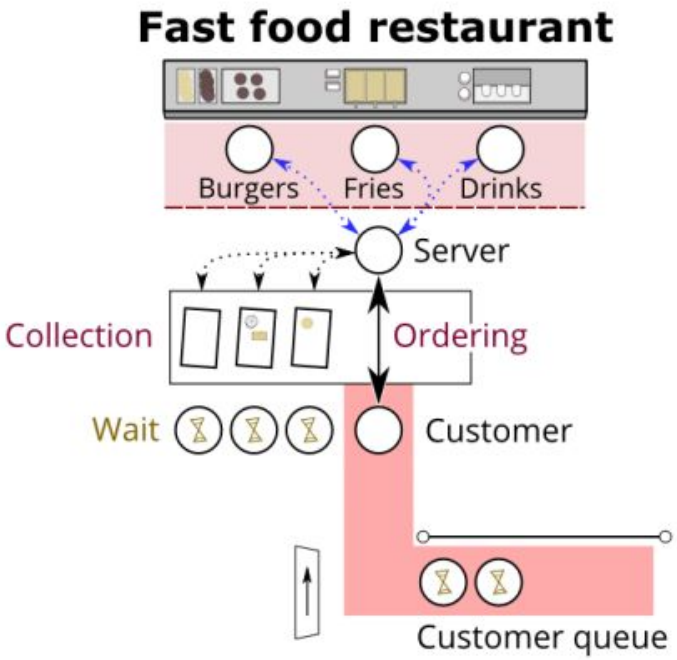
The event loop





Unlocking parallelism

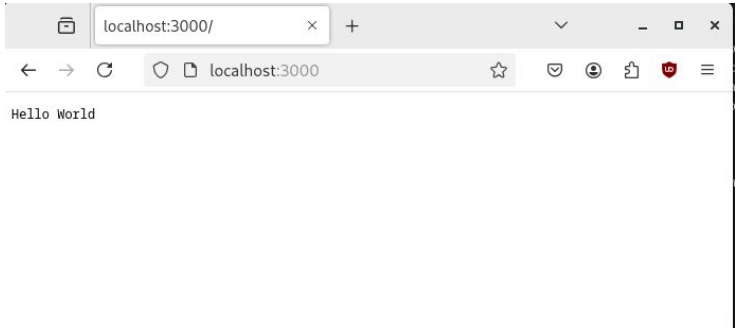




Unlocking productivity

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

const server = http.createServer(function(request, response) {
  response.end('Hello World');
});

server.listen(3000);
```

```
midawson@midawson-virtualbox:~/presentations$ node hello-server.js
```



Callbacks

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

const server = http.createServer(function(request, response) {
  response.end('Hello World');
});

server.listen(3000);
```

Callback Hell

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

const server = http.createServer((request, response) => {
  fs.stat('newfile.txt', (err, stats) => {
    fs.writeFile('newfile.txt', 'the file content', (err) => {
      response.end('Hello World' + ' : ' +  err);
    });
  });
});

server.listen(3000);
```

Promises

```
const thePromise = new Promise((resolve, reject) => {
  setTimeout(() => {
    resolve('done');
  }, 100);
});

thePromise.then((data) => {
  console.log('Fulfilled:' + data);
}, (err) => {
  console.log('Error:' + err);
});
```

```
midawson@midawson-virtualbox:~/presentations$ node promise.js
Fulfilled:done
```

Await

```
const thePromise = new Promise((resolve, reject) => {
  setTimeout(() => {
    resolve('done');
  }, 100);
});

async function doit() {
  try {
    const data = await thePromise;
    console.log('Fulfilled:' + data);
  } catch (err) {
    console.log('Error:' + err);
  }
}

doit();
```

```
midawson@midawson-virtualbox:~/presentations$ node await.js
Fulfilled:done
```

Async/Await

```
async function waitAndSucceed() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve('done');
    }, 100);
  });
};

async function doit() {
  try {
    const data = await waitAndSucceed();
    console.log('Fulfilled:' + data);
  } catch (err) {
    console.log('Error:' + err);
  };
};

doit();
```

```
midawson@midawson-virtualbox:~/presentations$ node async-await.js
Fulfilled:done
```

CJS/ESM

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

```
midawson@midawson-virtualbox:~/presentations$ node express.js
Example app listening on port 3000
```

CJS/ESM

```
import express from 'express';
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World - ESM!')
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

```
midawson@midawson-virtualbox:~/presentations$ node express.mjs
Example app listening on port 3000
```

CJS/ESM Interoperability

hello.js

```
const greeter = require('./greeter.js').default;

const greeting = greeter('World');
console.log(greeting);
```

CJS

greeter.js

```
import util from 'node:util';
export default function (greeting) {
  return 'Hello ' + greeting;
};
```

ESM

It just works!

```
>node hello.js
Hello World
```

Backported to 22.x in 22.12.0

The Fine Print:

Depends on two features that are still experimental but stability is “release candidate”, and no experimental warnings.

Options to turn of in case of issues

--no-experimental-require-module

-> ESM must be synchronous

--no-experimental-detect-module

-> depends on syntax detection - https://nodejs.org/api/packages.html#syntax-detection

-> may load file twice

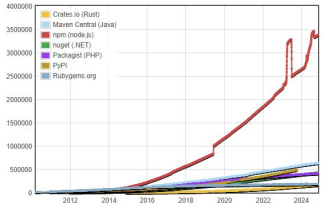
-> recommended to use "type": "module" in package.json where possible

Package Management

```
midawson@midawson-virtualbox:~/presentations$ node express.mjs
node:internal/modules/esm/resolve:844
  throw new ERR_MODULE_NOT_FOUND(packageName, fileURLToPath(base), null);
        ^

Error [ERR_MODULE_NOT_FOUND]: Cannot find package 'express' imported from
/home/midawson/presentations/express.mjs
    at packageResolve (node:internal/modules/esm/resolve:844:9)
    at moduleResolve (node:internal/modules/esm/resolve:901:20)
    at defaultResolve (node:internal/modules/esm/resolve:1121:11)
    at ModuleLoader.defaultResolve (node:internal/modules/esm/loader:396:12)
    at ModuleLoader.resolve (node:internal/modules/esm/loader:365:25)
    at ModuleLoader.getModuleJob (node:internal/modules/esm/loader:240:38)
    at ModuleWrap.<anonymous> (node:internal/modules/esm/module_job:85:39)
    at link (node:internal/modules/esm/module_job:84:36) {
  code: 'ERR_MODULE_NOT_FOUND'
}

Node.js v20.10.0
```

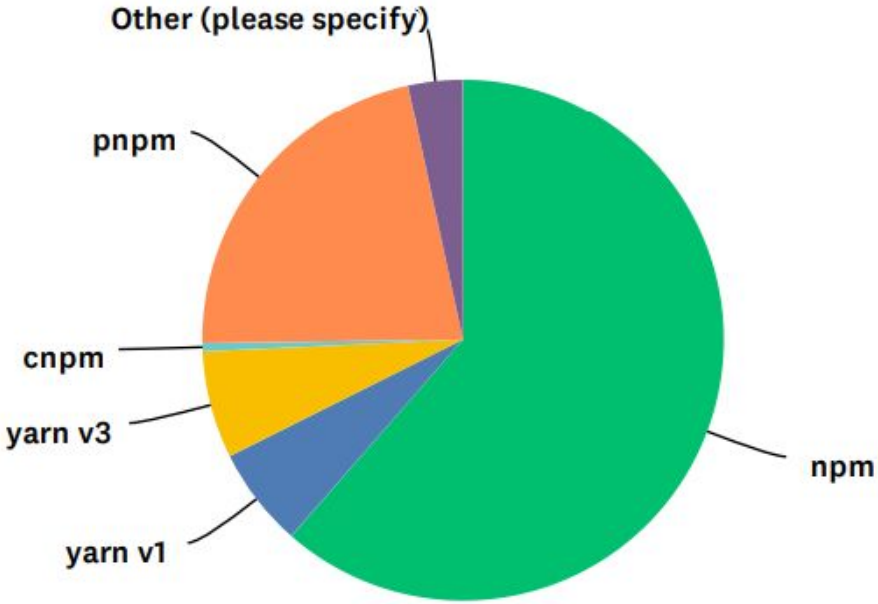
Package Management

```
midawson@midawson-virtualbox:~/presentations$ npm install express

added 66 packages, and audited 67 packages in 1s

14 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```




```
midawson@midawson-virtualbox:~/presentations$ cat package.json
{
  "dependencies": {
    "express": "^5.1.0"
  }
}
```

Package Management



*npm bundled with Node.js

Worker Threads

- Useful for doing CPU intensive work
  - Don’t block the even loop
  - Can take advantage of multiple CPUs
- Not much help with I/O intensive work
- Can share memory
  - transferring ArrayBuffer
  - SharedArrayBuffer

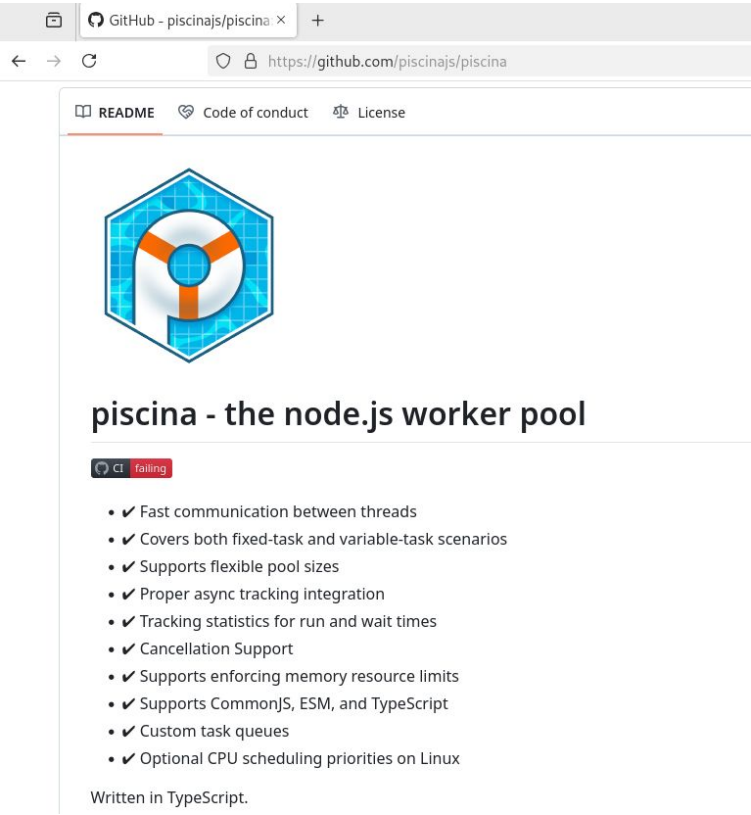
Worker Threads

```
const { Worker, isMainThread, parentPort } =
  require('node:worker_threads');

if (isMainThread) {
  const worker = new Worker(__filename);
  worker.once('message', (message) => {
    console.log(message);
  });
  worker.postMessage('Hello, world!');
} else {
  // When a message from the parent thread is received, send it back:
  parentPort.once('message', (message) => {
    parentPort.postMessage(message + ' From worker');
  });
}
```

Derived from https://nodejs.org/api/worker_threads.html#workerparentport

Worker Threads

- Now you are managing
  - Thread Pools
  - Concurrency
- piscina



What are native addons

- What
  - JavaScript functions, objects, etc. that are not actually written in JavaScript !
- Why
  - **Re-use:** Lots of existing code written in other languages
    - sharp, bcrypt, sqlite3, etc.
  - **Speed:** Some things run faster in other languages
  - **Access:** Some resources are not available from JavaScript natively
    - serialport

Building C/C++ addons - node-api

```c
#include <assert.h>
#include <node_api.h>

static napi_value Method(napi_env env, napi_callback_info info) {
  napi_status status;
  napi_value world;
  status = napi_create_string_utf8(env, "world", 5, &world);
  assert(status == napi_ok);
  return world;
}

#define DECLARE_NAPI_METHOD(name, func)                                          \
  { name, 0, func, 0, 0, 0, napi_default, 0 }

static napi_value Init(napi_env env, napi_value exports) {
  napi_status status;
  napi_property_descriptor desc = DECLARE_NAPI_METHOD("hello", Method);
  status = napi_define_properties(env, exports, 1, &desc);
  assert(status == napi_ok);
  return exports;
}

NAPI_MODULE(NODE_GYP_MODULE_NAME, Init)
```

```bash
var addon = require('bindings')('hello');

console.log(addon.hello()); // 'world'
```

Building C/C++ addons - node-addon-api

```cpp
#include <napi.h>

Napi::String Method(const Napi::CallbackInfo& info) {
  Napi::Env env = info.Env();
  return Napi::String::New(env, "world");
}

Napi::Object Init(Napi::Env env, Napi::Object exports) {
  exports.Set(Napi::String::New(env, "hello"),
              Napi::Function::New(env, Method));
  return exports;
}

NODE_API_MODULE(hello, Init)
```

```bash
var addon = require('bindings')('hello');

console.log(addon.hello()); // 'world'
```

Cross language support

Node-API bindings for other languages

Project	Programming language
napi-rs	Rust
napi-sys	Rust
nodejs-sys	Rust
neon	Rust

+

Early stage

Project	Programming language
napi-cs	C#
node-api-dotnet	C#
swift-napi-bindings	Swift
swift-node-addon-examples	Swift
napi-nim	Nim
zig-napigen	Zig
zig-nodejs-example	Zig
go-node-api	Go

Other core concepts

Index | Node.js v23.11.0 D

https://nodejs.org/docs/latest/api/

Node.js

About this documentation

Usage and example

Assertion testing

Asynchronous context tracking

Async hooks

Buffer

C++ addons

C/C++ addons with Node-API

C++ embedder API

Child processes

Cluster

Command-line options

Console

Crypto

Debugger

Deprecated APIs

Diagnostics Channel

DNS

Domain

Errors

Events

File system

Node.js v23.11.0 documentation

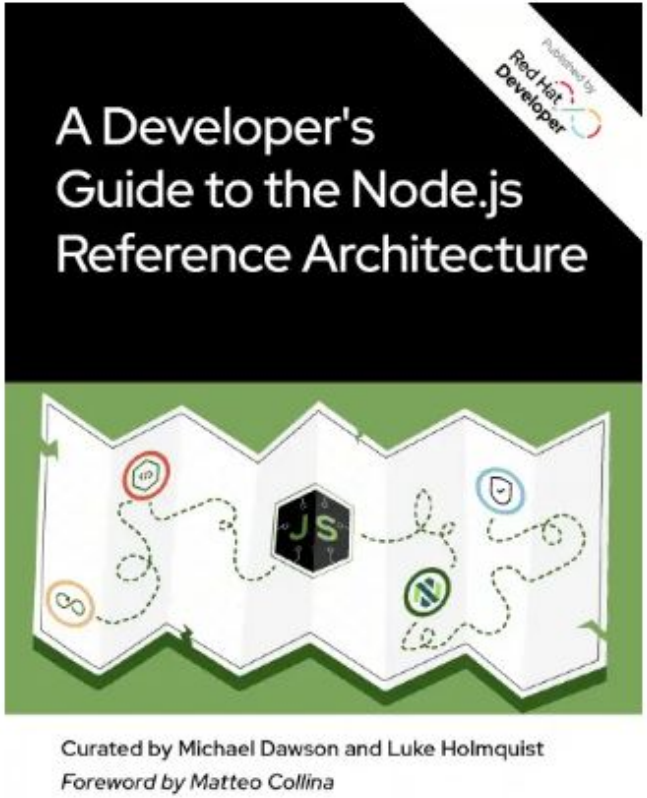
▸ Other versions | ▸ Options

- About this documentation
- Usage and example

- Assertion testing
- Asynchronous context tracking
- Async hooks
- Buffer
- C++ addons
- C/C++ addons with Node-API
- C++ embedder API
- Child processes
- Cluster
- Command-line options
- Console
- Crypto
- Debugger
- Deprecated APIs
- Diagnostics Channel
- DNS
- Domain
- Errors
- Events
- File system
- Globals
- HTTP
- HTTP/2
- HTTPS
- Inspector

nodejs.org/docs/latest/api

Building Enterprise Applications - Ref Arch



Components

The reference architecture covers the following components (currently a sections having recommendations):

- Functional Components
  - Web Framework
  - Template Engines
  - Message Queuing
  - Internationalization
  - Accessibility
  - API Definition
  - Databases
  - Authentication and Authorization

A Developer's Guide to the Node.js Reference Architecture

https://github.com/nodeshift/nodejs-reference-architecture

Upstream Contribution - Enterprise Focus

- Stable and Predictable releases
- Platform support
- Security
- Diagnostics
- Performance
- Code quality and safety net
- Key Features

https://youtu.be/Y0H4ki4bWN0

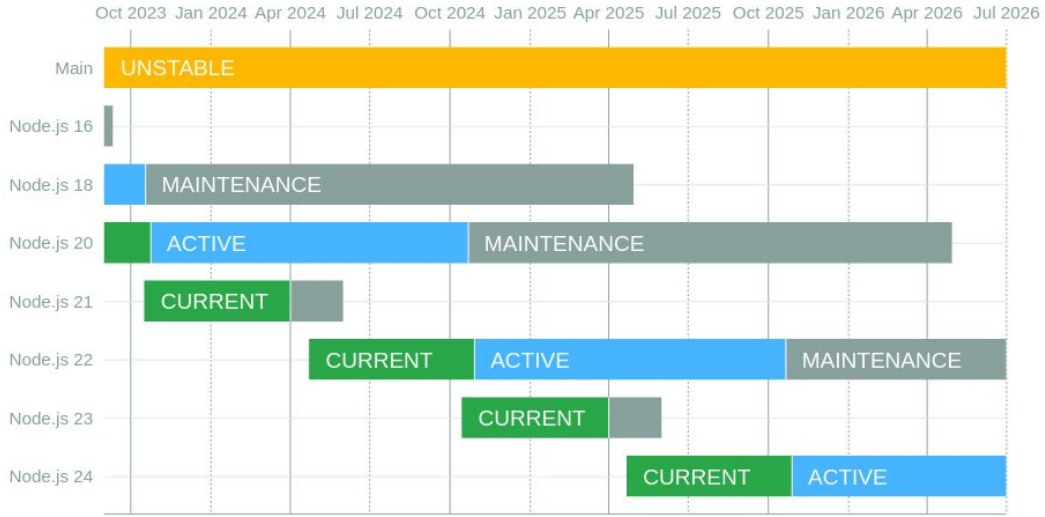
Releases Schedule

Last Major Community Release

- 24.x in April 2025 (Current)

Ongoing LTS Releases

- ~~Node.js 18.x - ends April 2025~~
- Node.js 20.x - ends April 2026
- Node.js 22.x - ends April 2027
- Node.js 24.x - ends April 2028



Broad Platform Support

x64

IBM i installation

- IBM i
  - yum install **nodejs${version}**
  - For example **yum install nodejs22**

Other Operating Systems

nodejs.org/en/download

Download Node.js®

Get Node.js® v22.14.0 (LTS) for Linux using nvm with npm

```bash
# Download and install nvm:
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.2/install.sh | bash

# in lieu of restarting the shell
\. "$HOME/.nvm/nvm.sh"

# Download and install Node.js:
nvm install 22

# Verify the Node.js version:
node -v # Should print "v22.14.0".
nvm current # Should print "v22.14.0".

# Verify npm version:
npm -v # Should print "10.9.2".
```

Bash Copy to clipboard

"nvm" is a cross-platform Node.js version manager. If you encounter any issues please visit **nvm's website**

Or get a prebuilt Node.js® for Linux running a x64 architecture.

Windows
macOS
Linux
AIX

Standalone Binary (.xz)

Read the changelog or blog post

Learn more about Node.js releases, se schedule and LTS status.

Learn how to verify signed SHASUMS.

Looking for Node.js source? Download a signed Node.js source tarball.

Check out our nightly binaries or all previous releases or the unofficial binaries for other platforms.

IBM i Integrations

- Database:
  - Mapepire - https://mapepire-ibmi.github.io/guides/usage/nodejs/
  - Node ODBC - https://www.npmjs.com/package/odbc
  - Node.js idb-connector - https://www.npmjs.com/package/idb-connector
- Running CL Commands, Service Programs / Programs:
  - itoolkit - https://www.npmjs.com/package/itoolkit

Thank You

Thank You

Questions?

Copyright and Trademarks