



Node.js What's Next?

Catalyzing change in the Node.js ecosystem

About Jean



Platform Tech Lead @Sanofi



Professor @Supinfo

Node.js community member

Member of few OSS projects



Twitter: @shepsheplu

GitHub: @sheplu

Linkedin: <https://www.linkedin.com/in/jeanburellier>

About Michael



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

- Following What's Next
- Recent Features
- Next-10
- New teams and initiatives
- How to get involved

Agenda

- **Following What's Next**
- Recent Features
- Next-10
- New teams and initiatives
- How to get involved

Following What's Next - from the Ground up

- Releases - <https://github.com/nodejs/release>
- GitHub Notifications
- Working Groups and Teams - <https://nodejs.org/calendar>
- Strategic Initiatives - <https://github.com/nodejs/node/blob/master/doc/guides/strategic-initiatives.md>
- Next-10 - <https://github.com/nodejs/next-10>

Releases

Last Major Release

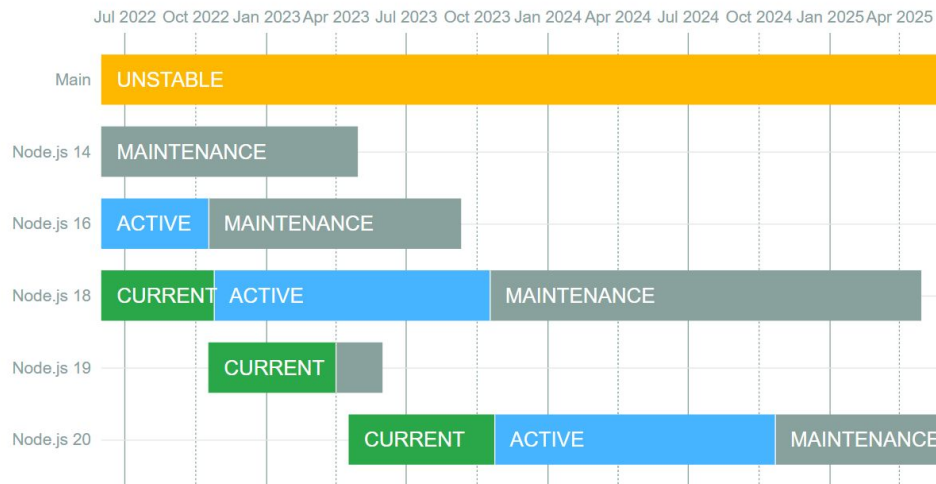
- 20.x in April 2023

Ongoing LTS Releases

- ~~Node.js 14.x ended April 2023~~
- Node.js 16.x - ends **Sept 2023**
- Node.js 18.x - ends **April 2025**

Current

- 20.x in April



Following What's Next from the Ground up

411 lines (351 sloc) | 53.4 KB

<> [icon] Raw Blame [icon] [icon]

2022-04-19, Version 18.0.0 (Current), @BethGriggs

Node.js 18 is here! Highlights include the update of the V8 JavaScript engine to 10.1, global fetch enabled by default, and a core test runner module.

Initially, Node.js 18 will replace Node.js 17 as our 'Current' release line. As per the release schedule, Node.js 18 will be the 'Current' release for the next 6 months and then promoted to Long-term Support (LTS) in October 2022. Once promoted to long-term support the release will be designated the codename 'Hydrogen'. Node.js 18 will be supported until April 2025.

Notable Changes

Deprecations and Removals

- (SEMVER-MAJOR) **fs**: runtime deprecate string coercion in `fs.write`, `fs.writeFileSync` (Livia Medeiros) [#42607](#)
- (SEMVER-MAJOR) **dns**: remove `dns.lookup` and `dnsPromises.lookup` options type coercion (Antoine du Hamel) [#41431](#)
- (SEMVER-MAJOR) **process**: runtime deprecate `multipleResolves` (Benjamin Gruenbaum) [#41896](#)
- (SEMVER-MAJOR) **stream**: remove thenable support (Robert Nagy) [#40773](#)
- (SEMVER-MAJOR) **tls**: move `tls.parseCertString` to end-of-life (Tobias Nießen) [#41479](#)

fetch (experimental)

An experimental fetch API is available on the global scope by default. The implementation is based upon [undici](#), an HTTP/1.1 client written for Node.js by contributors to the project.

```
const res = await fetch('https://nodejs.org/api/documentation.json');
if (res.ok) {
  const data = await res.json();
  console.log(data);
}
```

Through this addition, the following globals are made available: `fetch`, `FormData`, `Headers`, `Request`, `Response`.

Disable this API with the `--no-experimental-fetch` command-line flag.

[des/strategic-](#)

Following What's Next - from the Ground up

Inbox384

Allunread

Filter notifications

Group by: Date

Overwhelmed by notifications? We've found some repositories that may be causing notifications you don't need. [Update watching settings](#)

Select all

nodejs/node #43059

deps: update undici to 5.2.0

+10

subscribed

3 minutes ago

redhat-developer/app-services-tools #21

docs(README): update instructions for manual run

+3

subscribed

3 minutes ago

nodejs/node #43037

doc: add strategic initiative for shadow realm

+1

team mention

4 minutes ago

nodejs/node #43045

doc: remove git:// protocol, adjust nits in onboarding.md

+5

team mention

5 minutes ago

nodejs/node #43061

deps: upgrade npm to 8.10.0

+5

subscribed

6 minutes ago

nodejs/build #2888

Access to arm32 build machines for @ShogunPanda

+3

team mention

13 minutes ago

nodejs/admin #687

Youtube and Zoom access for @JakobJingleheimer (me)

team mention

24 minutes ago

openjs-foundation/cross-project-council #884

meta: add @achrinza as a Regular Member

+4

subscribed

1 hour ago

nodejs/node #43062

gc performance entries report incorrect startTime

subscribed

1 hour ago

Filters

Assigned1

Participating20

Mentioned

Team mentioned13

Review requested

Repositories

nodejs/node187

nodejs/node-gyp21

nodejs/help19

libuv/libuv16

nodejs/nodejs.org14

nodeshift/openshift-...12

s/strategic-

Following What's Next - from the Ground up

Node.js Project Calendar

Today April 2023

Print Week Month Agenda

Sun	Mon	Tue	Wed	Thu	Fri	Sat
26	27	28	29	30	31	Apr 1
		6pm Loaders Team Meeting 7pm Package Maintenance Team meeting	2pm Node.js Next 10 years 4pm Node.js TSC Meeting 6pm Node.js uvvival team meeting	2pm Security-WG meeting	3pm Node.js N-API team weekly meeting	
2	3	4	5	6	7	8
5pm Node.js Performance Team Meeting	3:30pm Diagnostics WG Meeting		2pm Node.js Release Working Group Meeting	3pm Node.js N-API team weekly meeting		
9	10	11	12	13	14	15
	12pm Build WG Meeting - early 6pm Loaders Team Meeting	1pm Node.js TSC Meeting 2pm Node.js Next 10 years 6pm Node.js uvvival team meeting	2pm Security-WG meeting 5pm Package Maintenance Team meeting	3pm Node.js N-API team weekly meeting		
16	17	18	19	20	21	22
5pm Node.js Performance Team Meeting		3pm Node.js TSC Meeting		3pm Node.js N-API team weekly meeting		
23	24	25	26	27	28	29
	6pm Loaders Team Meeting 7pm Package Maintenance Team meeting	2pm Node.js Next 10 years 6pm Node.js uvvival team meeting 7pm Node.js TSC Meeting	2pm Security-WG meeting	3pm Node.js N-API team weekly meeting		
30	May 1	2	3	4	5	6
5pm Node.js Performance Team Meeting	3:30pm Diagnostics WG Meeting	1pm Node.js TSC Meeting 10pm Build WG Meeting	2pm Node.js Release Working Group Meeting	3pm Node.js N-API team weekly meeting		

Events shown in time zone: Greenwich Mean Time

Google Calendar

[es/strategic-](#)

nodejs.org/calendar

Following What's Next - from the Ground up

Current initiatives

Initiative	Champion	Links
Core Promise APIs	Antoine du Hamel	nodejs/TSC#1094
QUIC / HTTP3	James M Snell	https://github.com/nodejs/quic
Shadow Realm	Chengzhong Wu	#42528
Startup Snapshot	Joyee Cheung	#35711
V8 Currency	Michaël Zasso	
Next-10	Michael Dawson	https://github.com/nodejs/next-10
Single executable apps	Darshan Sen	#43432
Performance	Yagiz Nizipli	https://github.com/nodejs/performance

[alendar](#)

[/doc/contributing/strat](#)

Agenda

- Following What's Next
- **Recent Features**
- Next-10
- New teams and initiatives
- How to get involved

Features - what's new/interesting ?

- Major Releases
 - Boring.....
- SemVer Minor is where its at!
 - features (but not all) flow into Current/LTS lines
- Majors still a good time to party
 - Talk about recent new features
 - Promotion to LTS

Features - Old News - Keep in mind for upgrade to 18.x

- OpenSSL 3
- Default DNS Resolution

OpenSSL 3 (17.x)

- Biggest impact
 - Users of smaller key sizes
 - Legacy algs no longer available by default
 - Native modules - Deprecation warnings

OpenSSL Migration Guide -

https://www.openssl.org/docs/man3.0/man7/migration_guide.html



<https://www.openssl.org/blog/blog/2021/09/07/OpenSSL3.Final/>

Default DNS resolution

- Node.js no longer prefers IPv4 over IPV6
- `--dns-result-order=ipv4first`

Features - Baking some likely to be stable in 20.x LTS

- Fetch
- WASI
- -- watch

Fetch - Experimental (Added in v17.5.0, no flag in 18.0.0)

<https://fetch.spec.whatwg.org/>

```
const resp = await fetch(url)
resp.text().then(text => console.log(text));
```

Technical Priority - Modern HTTP

WASI - Experimental (Added in v13.3.0, v12.16.0)

```
import { readFile } from 'node:fs/promises';
import { WASI } from 'wasi';
import { argv, env } from 'node:process';

const wasi = new WASI({
  version: 'preview1',
  args: argv,
  env,
  preopens: {
    '/sandbox': '/some/real/path/that/wasm/can/access',
  },
});

const wasm = await WebAssembly.compile(
  await readFile(new URL('./demo.wasm', import.meta.url)),
);
const instance = await WebAssembly.instantiate(wasm, wasi.getImportObject());

wasi.start(instance);
```

WASI - Experimental (Added in v13.3.0, v12.16.0)

```
import { readFile } from 'node:fs/promises';
import { WASI } from 'wasi';
import { argv, env } from 'node:process';
```

```
const wasi = new WASI({
  version: 'preview1',
  args: argv,
  env,
  preopens: {
    '/sandbox': '/some/real/path/that/wasm/can/access',
  },
});
```

```
const wasm = await WebAssembly.compile(
  await readFile(new URL('./demo.wasm', import.meta.url)),
);
const instance = await WebAssembly.instantiate(wasm, wasi.getImportObject());

wasi.start(instance);
```

New in 20.0.0

Command line flag no longer necessary

--watch - Experimental (Added in v18.11.0, v16.19.0)

- Variants
 - --watch
 - --watch-path
 - --watch-preserve-output
- Changes in watched files -> process restarts
- macOS/Windows only

Features - Hot off the press

- Argument parser
- Test Runner - stable
- Single Executable Applications
- Permissions Model
- Diagnostic Channel
- Support for ARM64 Windows

Argument Parser (added in v18.30, v16.17.0, stable in 20)

- Higher level API for command-line versus `process.argv`
- Collaboration from **yargs** and **Commander** maintainers

Argument Parser (Stable in 20)

```
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(values, positionals);
```

{ foo: true, bar: 'b' }

<https://nodejs.org/api/util.html#utilparseargsconfig>

Test Runner (Stable in 20)

- Mocks added in v18.13.0, v19.1.0
- Reporters added in v19.6.0
- Coverage added in v19.7.0

Test Runner (Added in 18.0.0, stable in 20)

```
const test = require('node:test');  
const assert = require('node:assert');
```

```
test('test1', (test) => {  
  assert.strictEqual('yes', 'no');  
});
```

<https://nodejs.org/api/test.html>

Tap Output



```
[midawson@drx-hemera pres]$ node --test test1.js  
TAP version 13  
not ok 1 - /home/midawson/pres/test1.js  
...  
duration_ms: 0.069108223  
failureType: 'subtestsFailed'  
exitCode: 1  
stdout: |  
  TAP version 13  
  not ok 1 - test1  
...  
duration_ms: 0.002767144  
failureType: 'testCodeFailure'  
error: |  
  Expected values to be strictly equal:  
  
  'yes' !== 'no'  
  
code: 'ERR_ASSERTION'  
stack: |  
  TestContext.<anonymous> (/home/midawson/pres/test1.js:5:10)  
  Test.runInAsyncScope (node:async_hooks:202:9)  
  Test.run (node:internal/test_runner/test:340:20)  
  Test.start (node:internal/test_runner/test:292:17)  
  Test.test (node:internal/test_runner/harness:126:18)  
  Object.<anonymous> (/home/midawson/pres/test1.js:4:1)  
  Module._compile (node:internal/modules/cjs/loader:1105:14)  
  Module._extensions.js (node:internal/modules/cjs/loader:1159:10)  
  Module.load (node:internal/modules/cjs/loader:981:32)  
  Module._load (node:internal/modules/cjs/loader:827:12)  
  ...  
  1..1  
  # tests 1  
  # pass 0  
  # fail 1  
  # skipped 0  
  # todo 0  
  # duration_ms 0.055707066  
  
stderr: |  
  (node:3039014) ExperimentalWarning: The test runner is an experimental feature. This feature could change at any time  
  (Use 'node --trace-warnings ...' to show where the warning was created)  
  
error: 'test failed'  
code: 'ERR_TEST_FAILURE'  
...  
  1..1  
  # tests 1  
  # pass 0  
  # fail 1  
  # skipped 0  
  # todo 0  
  # duration_ms 0.133823902
```

Technical Value - Developer experience

Single Executable Applications (Experimental Added in v19.7.0)

- Wins
 - Easier distribution
 - Reduced surface area
- Bundles app into Existing Node.js binary
 - Fuse to eliminate overhead when not used
 - Looks for a segment within the executable that holds bundled code.
 - Runs the bundled code when such a segment is found.
- Injection done though <https://github.com/nodejs/postject>
- Details in <https://github.com/nodejs/node/blob/main/doc/contributing/maintaining-single-executable-application-support.md>

Process Based Permissions (Experimental, added in v19.7.0)

```
node --experimental-permission app.js
```

- Limits some features by default
 - file system access (read/write)
 - spawning processes
 - use of node:worker_threads
 - Native modules
- Runtime API to check

Tracing Channel - Experimental (Added in v19.9.0)

- Why ?
 - **Simplify** generation/consumption of Diagnostic Channel info
- Generating info
 - `tracingChannel.traceSync(fn[, context[, thisArg[, ...args]])`
 - `tracingChannel.tracePromise(fn[, context[, thisArg[, ...args]])`
 - `tracingChannel.traceCallback(fn[, position[, context[, thisArg[, ...args]])`
- Consuming info
 - `tracingChannel.subscribe(subscribers)`
 - `tracingChannel.unsubscribe(subscribers)`

Tracing Channel - Experimental (Added in v19.9.0)

```
const diagnostic_channel = require('node:diagnostics_channel');  
const channels = diagnostic_channel.tracingChannel('my-package-info');
```

```
// info consumer  
channels.subscribe({  
  start(message) { console.log('start:' + message.ctx_data) },  
  end(message) { console.log('end:' + message.ctx_data) },  
  error(message) { console.log('error:' + message.ctx_data) },  
});
```

```
// info source  
channels.traceSync(() => {  
  console.log('function ran');  
}, { ctx_data: 'data' });
```

```
[midawson@drx-hemera channel]$ node test.js  
start:data  
function ran  
end:data
```

ARM64 support for Windows (Added in 19.9.0)

Downloads

Latest Current Version: 20.0.0 (includes npm 9.6.4)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS
Recommended For Most Users

Current
Latest Features

 Windows Installer <small>node-v20.0.0-x64.msi</small>	 macOS Installer <small>node-v20.0.0.pkg</small>	 Source Code <small>node-v20.0.0.tar.gz</small>
--	--	---

Windows Installer (.msi)
Windows Binary (.zip)
macOS Installer (.pkg)
macOS Binary (.tar.gz)
Linux Binaries (x64)
Linux Binaries (ARM)
Source Code

32-bit	64-bit	ARM64
32-bit	64-bit	ARM64
64-bit / ARM64		
64-bit		ARM64
	64-bit	
ARMv7		ARMv8
node-v20.0.0.tar.gz		

Additional Platforms

Docker Image

Linux on Power LE Systems

Linux on System z

AIX on Power Systems

Official Node.js Docker Image	
	64-bit
	64-bit
	64-bit

- Signed SHASUMS for release files (How to verify)
- All download options
- Installing Node.js via package manager
- Previous Releases
- Nightly builds
- Unofficial builds
- Building Node.js from source on supported platforms
- Installing Node.js via binary archive
- Install on Windows Subsystem for Linux (WSL)

Agenda

- Following What's Next
- Recent Features
- **Next-10**
- New teams and initiatives
- How to get involved

Next 10 - Foundation

- Technical Values and Priorities - <https://github.com/nodejs/node/blob/master/doc/guides/technical-values.md>
- Constituencies - <https://github.com/nodejs/next-10/blob/main/CONSTITUENCIES.md>
- Constituencies Needs - <https://github.com/nodejs/next-10/blob/main/CONSTITUENCY-NEEDS.md>



Trott doc: use "Long Term Support" in technical values doc ...

Latest commit 0ba7da6 on May 29



4 contributors



74 lines (63 sloc) | 2.77 KB

Raw

Blame



Technical values and priorities

The project uses these technical values to establish priorities and guide collaboration.

These are the shared values as of this writing and will evolve. We hope they are useful to people new to the project in order to better understand which contributions will be aligned with the current direction and as thinking points when trading off between conflicting goals.

The factors influencing every discussion/decision are different and priority 1 does not always trump priority 2 and so on.

Values and priority level

- Priority 1 - Developer experience
- Priority 2 - Stability
- Priority 3 - Operational qualities
- Priority 4 - Node.js maintainer experience
- Priority 5 - Up to date technology and APIs

Value descriptions

1 - Developer experience

We value ensuring that developers are productive and enjoy developing with Node.js. Some key elements of this include:

</doc/gu></-NEEDS.md>

Next 10 - Foundation

- Technical Values and Priorities - <https://github.com/nodejs/node/blob/master/doc/guides/technical-values.md>
- Constituencies - <https://github.com/nodejs/next-10/blob/main/CONSTITUENCIES.md>
- Constituencies Needs - <https://github.com/nodejs/next-10/blob/main/CONSTITUENCY-NEEDS.md>

Next 10

- Tech
- Cons
- Cons

List of constituencies

- Direct end users
 - Users who run tools themselves (install Node.js, run tool, deal with errors including writing/using scripts)
- Application operators
 - Users who interact with existing, running applications (regardless if they wrote the application or not)
 - Service and infrastructure providers
- Application Developers
 - Front-end tool consumers
 - Back-end server authors
 - Hobby developers
 - Professional developers
 - Tool authors
- Library/package authors
 - Users who write libraries and packages to be included on other applications
- Node.js contributors
 - Developers working directly on [nodejs/node](#)
 - Individuals participating in Working Groups and teams
- Organizations with investments in Node.js (eg: Enterprises, Government bodies, startups, non-profits, standards groups like TC39)
 - C level executives (CTO, CEO, etc.)
 - Planning/program offices
 - Managers
- Education
 - Teachers
 - Students
 - Organizations who help people learn Node.js (colleges, University, boot camps, online learning resources, etc.)
 - Programs to help people demonstrate capability (certification programs, etc.)
- Security Practitioners
 - People involved in the life-cycle of handling security issues including:
 - Penetration testing
 - Incident response
 - CVE scanning/remediation,
 - Legal, Public relations

[/master/doc/gu](#)

[TUENCY-NEEDS.md](#)

Next 10 - Foundation

- Technical Values and Priorities - <https://github.com/nodejs/node/blob/master/doc/guides/technical-values.md>
- Constituencies - <https://github.com/nodejs/next-10/blob/main/CONSTITUENCIES.md>
- Constituencies Needs - <https://github.com/nodejs/next-10/blob/main/CONSTITUENCY-NEEDS.md>

Needs of the Node.js Constituencies

Need	Users	Ops	AppDev	LibDev	Orgs	NodeMaint
Good understanding of the direction of the project	X	X	X	X	X	X
Ability to affect the direction of the project	X	X	X	X	X	X
Consumable APIs and docs	X		X	X		X
Predictable and stable releases	X	X	X	X	X	
Innovation at a consumable pace	X		X	X	X	X
Easy Installation	X					
Easy issue reporting, resolution and collaboration	X	X	X	X	X	
Broad deployment platform support	X	X			X	
Broad desktop platform support	X		X	X		X
Consistent and intuitive error handling	X	X	X	X		

[/nodejs/node/blob/master/doc/gu
jes.md](#)

[;ITUENCIES.md](#)

[ob/main/CONSTITUENCY-NEEDS.md](#)

•
•
•

Ability to embed and bundle the Node.js runtime			X			
A well maintained and secure standard library			X	X	X	
Assets that show Node.js is a good choice	X		X	X	X	

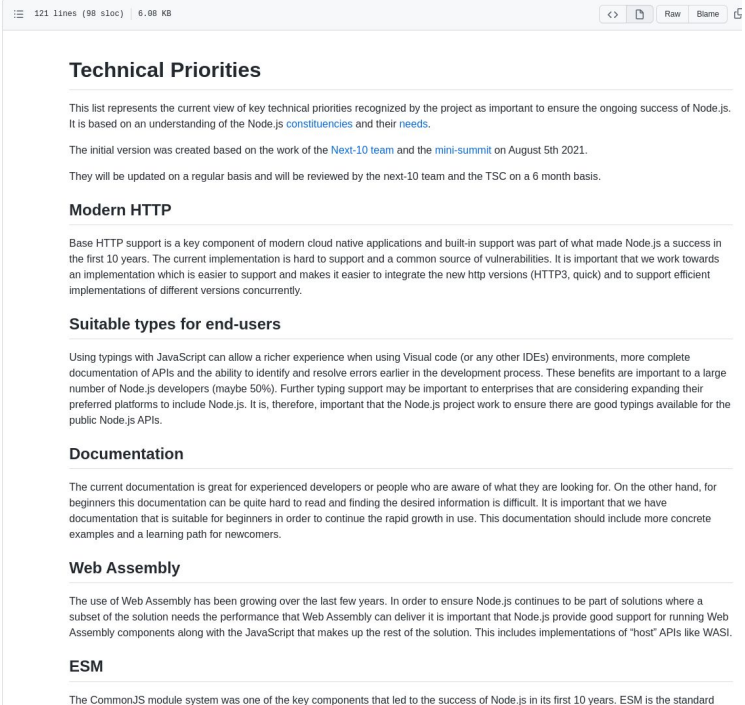
Good understanding of the direction of the project

- An understanding of the work happening in the Node.js project (what features do I have to look forward to, what's the project direction). Changes between releases. Info on how new changes affect end users/developers

Technical Priorities

<https://github.com/nodejs/node/blob/master/doc/contributing/technical-priorities.md>

- Modern HTTP
- Suitable types for end-users
- Documentation
- Web Assembly
- ESM
- Observability
- Permissions/policies/security model
- Better multithreaded support
- Single Executable Applications
- Support for features from the latest ECMAScript spec



The screenshot shows a GitHub file viewer for the file `technical-priorities.md` in the `nodejs/node` repository. The file is 121 lines long, 98 sloc, and 6.08 KB. The content of the file is as follows:

Technical Priorities

This list represents the current view of key technical priorities recognized by the project as important to ensure the ongoing success of Node.js. It is based on an understanding of the Node.js [constituencies](#) and their [needs](#).

The initial version was created based on the work of the [Next-10 team](#) and the [mini-summit](#) on August 5th 2021.

They will be updated on a regular basis and will be reviewed by the next-10 team and the TSC on a 6 month basis.

Modern HTTP

Base HTTP support is a key component of modern cloud native applications and built-in support was part of what made Node.js a success in the first 10 years. The current implementation is hard to support and a common source of vulnerabilities. It is important that we work towards an implementation which is easier to support and makes it easier to integrate the new http versions (HTTP3, quick) and to support efficient implementations of different versions concurrently.

Suitable types for end-users

Using typings with JavaScript can allow a richer experience when using Visual code (or any other IDEs) environments, more complete documentation of APIs and the ability to identify and resolve errors earlier in the development process. These benefits are important to a large number of Node.js developers (maybe 50%). Further typing support may be important to enterprises that are considering expanding their preferred platforms to include Node.js. It is, therefore, important that the Node.js project work to ensure there are good typings available for the public Node.js APIs.

Documentation

The current documentation is great for experienced developers or people who are aware of what they are looking for. On the other hand, for beginners this documentation can be quite hard to read and finding the desired information is difficult. It is important that we have documentation that is suitable for beginners in order to continue the rapid growth in use. This documentation should include more concrete examples and a learning path for newcomers.

Web Assembly

The use of Web Assembly has been growing over the last few years. In order to ensure Node.js continues to be part of solutions where a subset of the solution needs the performance that Web Assembly can deliver it is important that Node.js provide good support for running Web Assembly components along with the JavaScript that makes up the rest of the solution. This includes implementations of "host" APIs like WASI.

ESM

The CommonJS module system was one of the key components that led to the success of Node.js in its first 10 years. ESM is the standard

Mini Summits/Collaborator Summits

- Nov 2021 - Suitable types/Single executable applications
- Jan 2022 - Modern HTTP/Documentation
- Apr 2022 - WASM/Security model and permissions
- June 2022 (collab summit)
ESM/Observability

Next 10 - Current Initiatives

- Future mini-summits
 - Serverless
 - HTTP
 - TypeScript / Hooks

Next 10 - Current Initiatives

- Survey
 - Feedback from the community (~1700)
 - Idea for new initiatives
 - Community needs
 - How is Node.js used

Next 10 - Current Initiatives

- Survey

- Feedback

- Ideas

- Comments

- How to improve

Modern HTTP

Support for features from the latest ECMAScript spec

Documentation

Better multithreaded support

ESM

Permissions/policies/security model

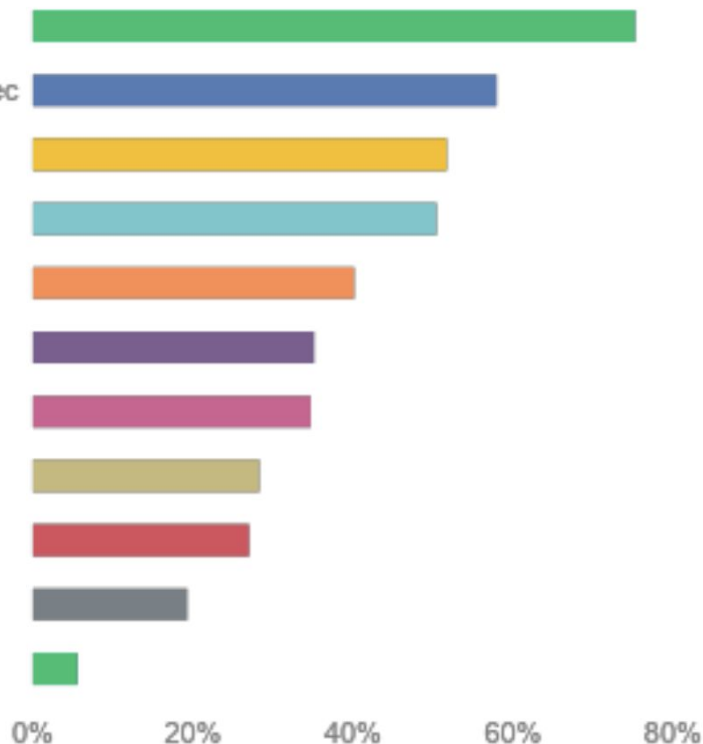
Suitable types for end-users

Observability

Single Executable Applications

WebAssembly

Other (please specify)



Next 10 - Current Initiatives

- Survey

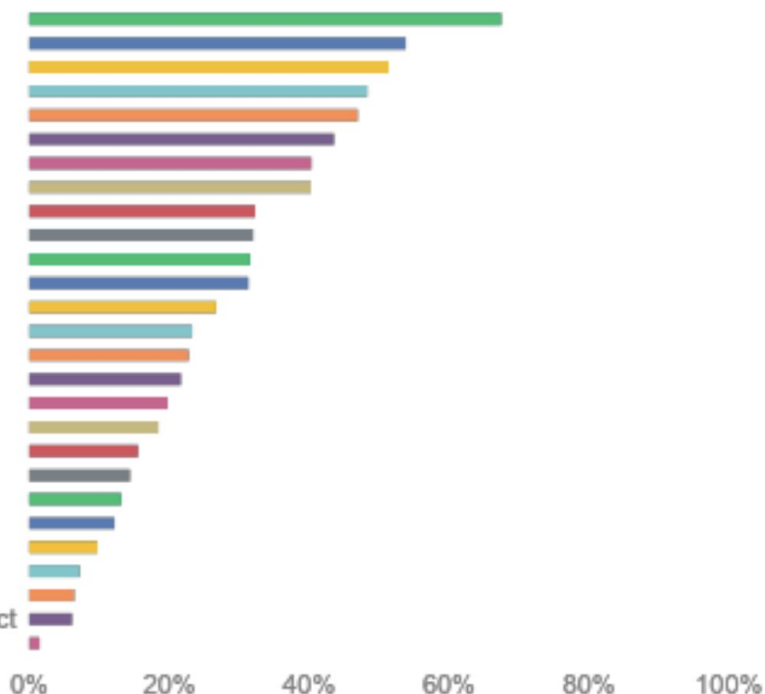
- Feedback

- Ideas

- Comments

- How

Consumable APIs and docs
Predictable and stable releases
A well maintained and secure standard library
Relevant APIs in core
Good understanding of the direction of the project
Runtime diagnostic tooling
Consistent and intuitive error handling
Easy Installation
Reasonable resource usage/performance
Module/dependency info and management
Development time diagnostic tooling
Good security and CVE practices in the project
Good CI infrastructure and experience in the project
Innovation at a consumable pace
Broad deployment platform support
Assets that show Node.js is a good choice
Ability to embed and bundle the Node.js runtime
Easy issue reporting, resolution and collaboration
Ability to affect the direction of the project
Easy contribution workflow
Better CVE management in the ecosystem
Broad desktop platform support
Better ways to build consensus in the project
Ways to fund work
Ability to assess impact of changes they make
Supportive Collaborators and Environment in the project
Other (please specify)



Next 10 - Current Initiatives

- Documentation proposal
 - Easier to use
 - Easier to maintain
 - Searchable
 - Generation of JSON structured data

Next 10 - Current Initiatives

- **Sharing project news**

- The project aims to make it easy to share project news with those who publish newsletters that cover Node.js.
- The next-10 team will be reaching out to periodicals like Node.js weekly to point them at discussion item created in core and issues created by teams across the project.

<https://github.com/nodejs/node/blob/main/doc/contributing/sharing-project-news.md>

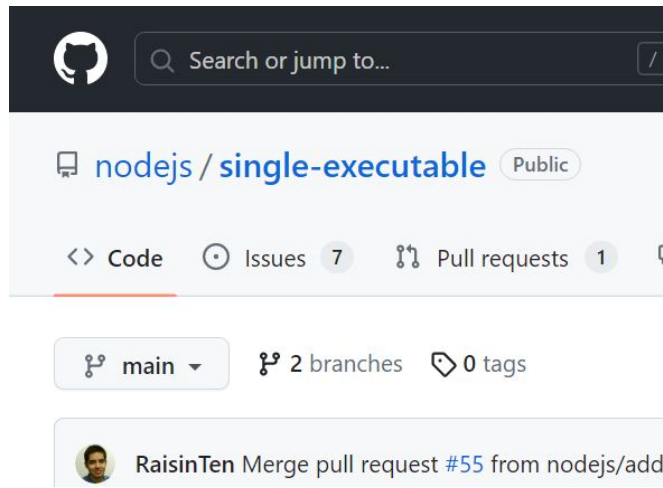
Agenda

- How to follow what's going on in the project
- Recent Features
- Next-10
- **New teams and initiatives**
- How to get involved

New teams and initiatives

- Single Executable Applications
 - Package Node.js into existing Node.js binary

Many thanks to:
Darshan
Joyee



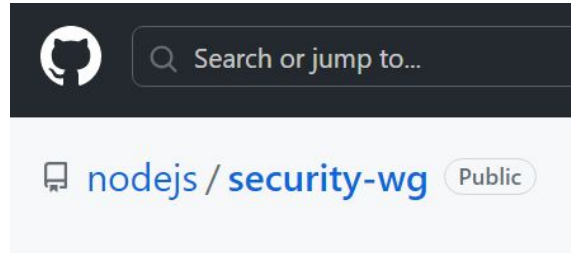
New teams and initiatives

- Security
 - Funding from OSSF
 - Some current initiatives
 - Scorecards related to security
 - Automation of dependency updates
 - Automation of security releases
 - Continued work on permission model



<https://openssf.org/>

Many thanks to:
Rafael



New teams and initiatives

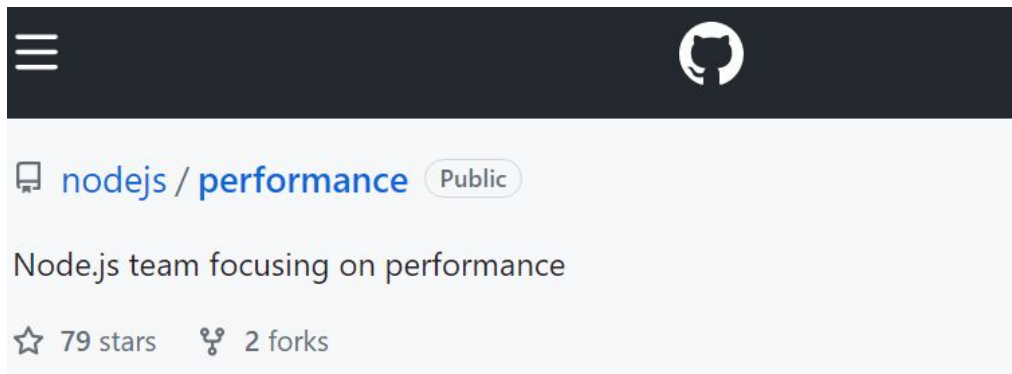
- Performance
 - Improvements to URL, fetch() and EventTarget
 - Ada fast spec compliant URL parser

Many thanks to:



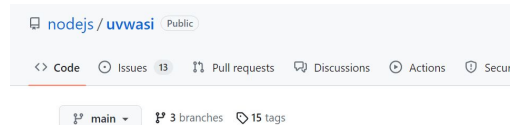
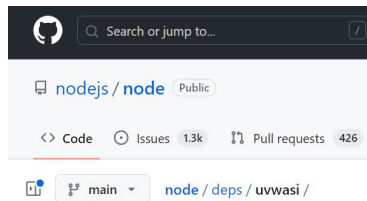
Yagiz Nizipli ●

Node.js Core Collaborator,
OpenJS Foundation Member
he/him



New teams and initiatives

- uvwasi
 - Web assembly system interface (WASI) implementation in Node.js
 - Used in other runtimes like grain
 - Current focus
 - Complete socket functions in preview1
 - Planning for preview2
 - Refactoring to ease use as shared library



Many thanks to:
Colin

Agenda

- Following What's Next
- Recent Features
- Next-10
- New teams and initiatives
- **How to get involved**

How to get Involved

- You are all welcomed
- Lot of different initiatives
 - Participating to meetings
 - Opening issues
 - Helping on doc / translation

Copyright and Trademarks

© Red Hat, IBM. All Rights Reserved

Red Hat, the Red Hat logos are trademarks or registered trademarks of Red Hat

IBM, the IBM logo, ibm.com are trademarks or registered trademarks of International Business Machines Corp.,

registered in many jurisdictions worldwide.

A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml

Node.js is an official trademark of Joyent. IBM SDK for Node.js is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

Java, JavaScript and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

npm is a trademark of npm, Inc.

Other trademarks or logos are owned by their respective owners.