What's new and what's next

Confoo.ca 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 2021-2022

BlueSky: @mhdawson.bsky.social

Twitter: @mhdawson1

GitHub: @mhdawson

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

Overview

- What’s Not New
- Recent survey
- What’s new
  - Features
  - Trends
- What’s next

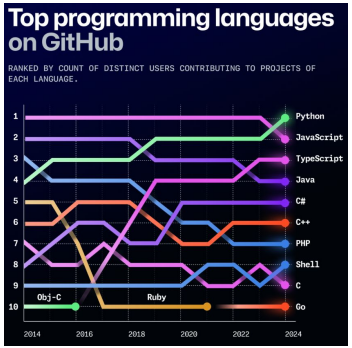
What's not new

* 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/

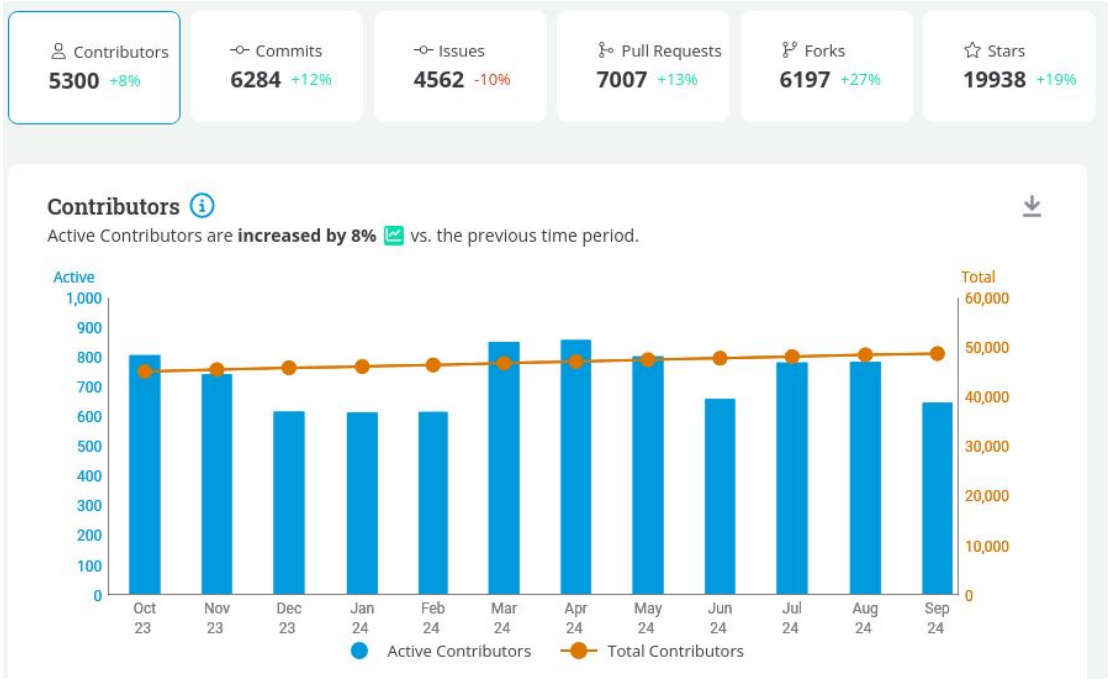


2 - JavaScript
3 - typeScript
Notes that combined still higher than Python

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

What's not new

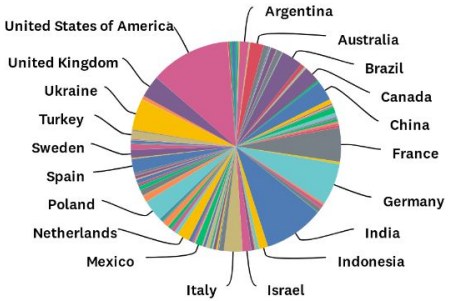
- Active and vibrant project

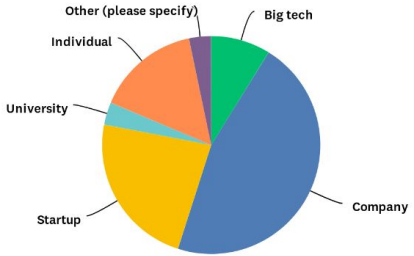


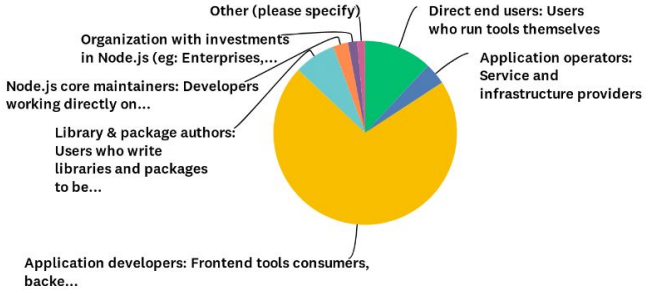
https://insights.lfx.linuxfoundation.org/foundation/ojsf/overview/github?project=ojsf-nodejs (Oct 1 2023 - Sep 30 2024)

Recent Survey

- 2978 respondents, 81% more than last year
- Average of 6 years using Node.js

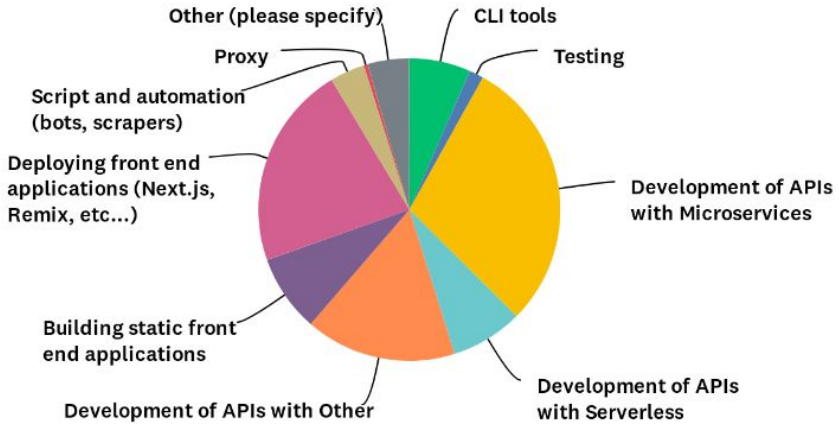






https://github.com/nodejs/next-10/blob/main/surveys/2024-04

Recent Survey

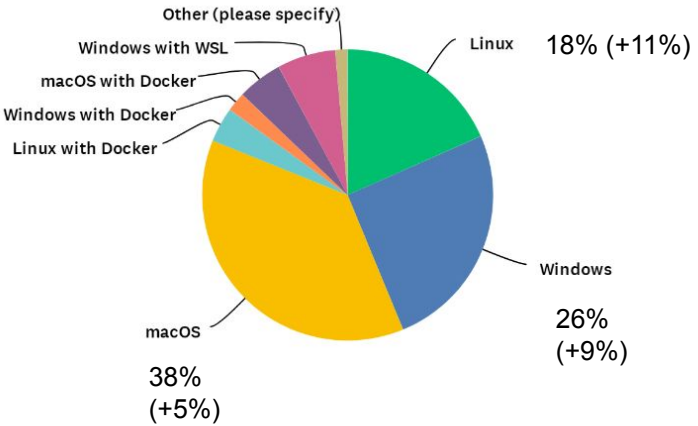


Use Cases

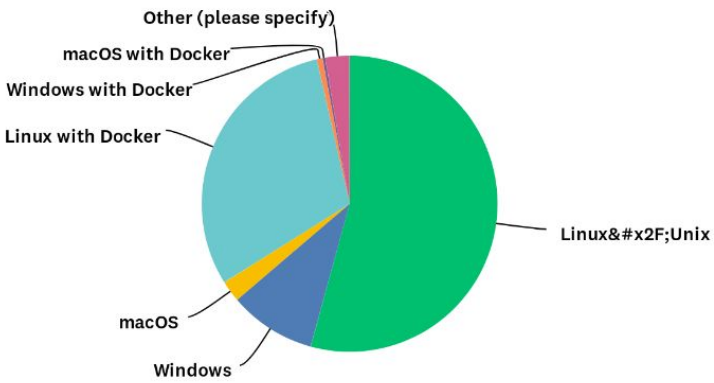
- 53% back end services
- 22% deploying front end apps
- 21% building, and scripting

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

Recent Survey



Local Development



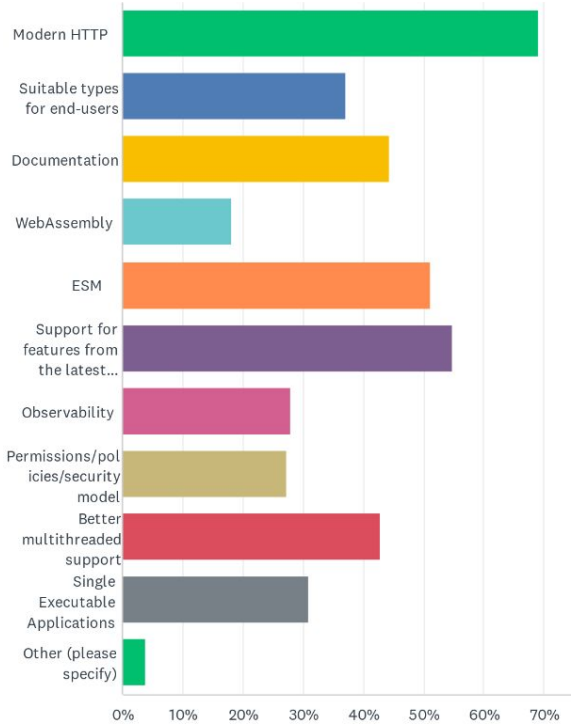
Production

x86 dominates but Arm is growing

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

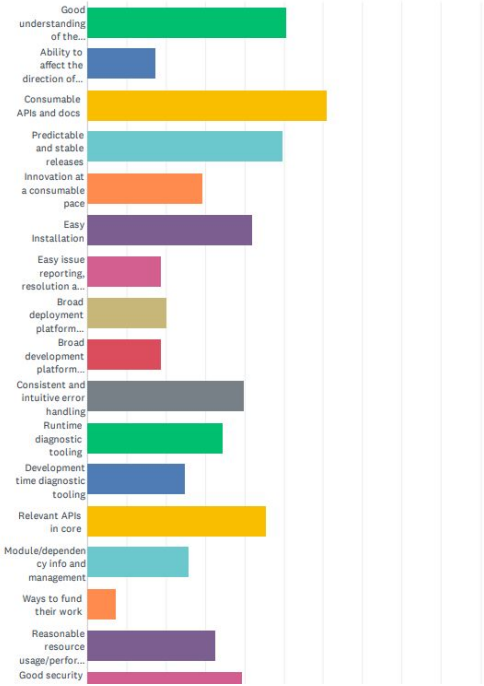
Recent Survey

Technical Priorities



What's important to you

(incomplete list shown)



Recent Survey

- Key Takeaways
  - Most common pain points
    - ESM integration
    - TypeScript
    - Docs
    - Monitoring/Diagnostics
    - Performance
  - ESM widely used, 71% reported code with ES module syntax
  - New stable features well used
  - Experimental features well used

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

What’s new - Features

- 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

What's new - Features

- 2024 was Exciting year
- Some Highlights
  - Progress on ESM compatibility
  - Experimental Limited TypeScript Support
  - Experimental Web Storage API
  - Experimental SQLite API
  - Continued progress on performance
  - Node run
- Regular Releases

Most common pain points
- ESM integration ✅
- TypeScript ✅
- Docs
- Monitoring/Diagnostics
- Performance ✅

Releases

Last Major Release

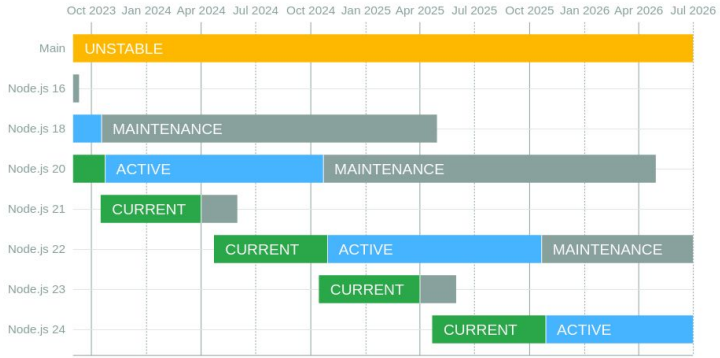
- 23.x in October 2024 (Curent)

Ongoing LTS Releases

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

Next current

- 24.x in April 2024



https://github.com/nodejs/Release

ESM Compatibility

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

Experimental Limited Typescript Support

- At a glance
  - --experimental-strip-types (enabled by default in v23.6.0)
    - replaces TypeScript with whitespace
      - no need for source maps
    - .ts, .mts, .cts files supported
  - --experimental-transform-types (not enabled by default)
- Key limitations
  - Only supports a subset of TypeScript
  - Does not handle files in node_modules (design choice)
  - Does not read tsconfig.json
  - type keyword needed in imports

https://nodejs.org/api/typescript.html - Stability: 1.1 - Active Development

Experimental Limited Typescript Support

hello.ts

```
import { GreetOptions } from
'./greeter.ts'
import greeter from './greeter.ts'

const message: string = 'World';
const greeting: string =
greeter(GreetOptions.Hello,
        message);

console.log(greeting);
```

greeter.ts

```
export enum GreetOptions {
  Hello,
  Goodbye
};

export default function (helloType:
GreetOptions, greeting: string) : string {
  let world: string;
  if (helloType === GreetOptions.Hello) {
    world = 'Hello ' + greeting;
    return world;
  };
  throw Error('Unknown GreetOption');
};
```

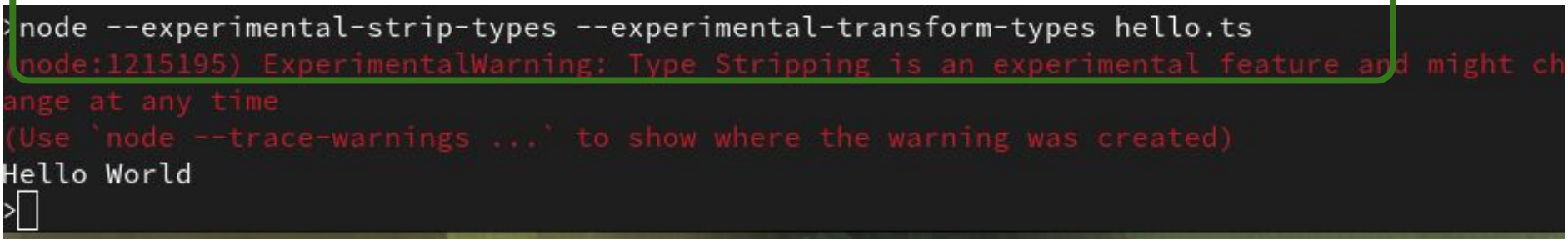
Experimental Limited Typescript Support (before)

```
midawson@midawson-virtualbox:~/test-typescript-with-nodejs$ node hello.ts
node:internal/modules/esm/get_format:218
  throw new ERR_UNKNOWN_FILE_EXTENSION(ext, filepath);
        ^

TypeError [ERR_UNKNOWN_FILE_EXTENSION]: Unknown file extension ".ts" for /home/midawson/test-typescript-with-nodejs/hello.ts
    at Object.getFileProtocolModuleFormat [as file:] (node:internal/modules/esm/get_format:218:9)
    at defaultGetFormat (node:internal/modules/esm/get_format:244:36)
    at defaultLoad (node:internal/modules/esm/load:122:22)
    at async ModuleLoader.load (node:internal/modules/esm/loader:570:7)
    at async ModuleLoader.moduleProvider (node:internal/modules/esm/loader:443:45)
    at async ModuleJob._link (node:internal/modules/esm/module_job:106:19) {
  code: 'ERR_UNKNOWN_FILE_EXTENSION'
}

Node.js v22.9.0
```

Experimental Limited Typescript Support

```
midawson@midawson-virtualbox:~/test-typescript-with-nodejs$ node --experimental-strip-types hello.ts
(node:1214933) ExperimentalWarning: Type Stripping is an experimental feature and might change at any time
(Use `node --trace-warnings ...` to show where the warning was created)
node:internal/process/promises:392
      new UnhandledPromiseRejection(reason);
      ^

UnhandledPromiseRejection: This error originated either by throwing inside of an async function without a catch
 block, or by rejecting a promise which was not handled with .catch(). The promise rejected with the reason "
x TypeScript enum is not supported in strip-only mode
   ,-[1:1]
 1 | ,-> export enum GreetOptions {
 2 | |     Hello,
 3 | |     Goodbye
 4 | `-> }
 5 |
 6 |     export default function (helloType: GreetOptions, greeting: string) : string {
 7 |       let world: string;
   `----
".
    at throwUnhandledRejectionsMode (node:internal/process/promises:392:7)
    at processPromiseRejections (node:internal/process/promises:475:17)
    at process.processTicksAndRejections (node:internal/process/task_queues:106:32) {
  code: 'ERR_UNHANDLED_REJECTION'
}

Node.js v22.9.0
```

Experimental Limited Typescript Support



Experimental Limited Typescript Support

- TSC ~~has~~/had trouble with .ts extensions in imports

```
>npx -p typescript tsc *.ts --declaration --allowJs --allowImportingTsExtensions --outDir
types
error TS5096: Option 'allowImportingTsExtensions' can only be used when either 'noEmit' or
 'emitDeclarationOnly' is set.
```

- Node.js Typescript team - nodejs / typescript
  - Members from typescript,Node.js collaborated on issue
  - PR landed to handle https://github.com/microsoft/TypeScript/pull/59767
    - --rewriteRelativeImportExtensions

Experimental Limited Typescript Support

- TypeScript 5.8 Beta - Option to target only subset enabled by default
  - --erasableSyntaxOnly
  - https://devblogs.microsoft.com/typescript/announcing-typescript-5-8-beta/#the---erasablesyntaxonly-option

```
midawson@midawson-virtualbox:~/confoo$ npm run build:strict

> build:strict
> tsc --erasableSyntaxOnly

src/greeter.ts:1:13 - error TS1294: This syntax is not allowed when 'erasableSyntaxOnly' is enabled.

1 export enum GreetOptions {
              ~~~~~~~~~~~~


Found 1 error in src/greeter.ts:1
```

Experimental Web Storage API

```
// persistent
localStorage.setItem('RuleToLiveBy', 'Never bet against Node.js');
console.log(localStorage.getItem('RuleToLiveBy'));

// in memory 10MB max
sessionStorage.setItem('RuleToLiveBy', 'I told you never bet against Node.js');
console.log(sessionStorage.getItem('RuleToLiveBy'));
```

```
>node --experimental-webstorage --localstorage-file=data.bin test-local-storage.js
Never bet against Node.js
I told you never bet against Node.js
(node:1267359) ExperimentalWarning: Web Storage is an experimental feature and might chang
e at any time
(Use `node --trace-warnings ...` to show where the warning was created)
>
```

Experimental SQLite API

```
const { DatabaseSync } = require ('node:sqlite');
const db = new DatabaseSync(':memory:');

db.exec(`
  CREATE TABLE data( key TEXT PRIMARY KEY, value TEXT) STRICT
`);

const insertStmt = db.prepare( 'INSERT INTO data (key,value) VALUES (?, ?)');

const readStmt = db.prepare( 'SELECT * FROM data ORDER BY key');

insertStmt.run('RULE TO LIVE BY 2', 'I told you never bet against Node.js');
insertStmt.run('RULE TO LIVE BY 1', 'Never bet against Node.js');

console.log(readStmt.all());
```

Experimental SQLite API

```
>node --experimental-sqlite test-sqlite.js
[
  [Object: null prototype] {
    key: 'RULE TO LIVE BY 1',
    value: 'Never bet against Node.js'
  },
  [Object: null prototype] {
    key: 'RULE TO LIVE BY 2',
    value: 'I told you never bet against Node.js'
  }
]
(node:1268528) ExperimentalWarning: SQLite is an experimental feature and
(Use `node --trace-warnings ...` to show where the warning was created)
```

Continued progress on performance

- Newer versions of V8
- On-disk caching to improve startup
  - NODE_COMPILE_CACHE, NODE_DISABLE_COMPILE_CACHE
- Buffer improvements
- FS improvements

Node --run

Package.json

```
{
  "name": "my-great-app",
  "version": "0.0.1",
  "scripts": {
    "start": "node doit.js"
  }
}
```

>node --run start

I ran

What’s new - Trends

- node-addon-api overtakes Nan
- Node.js/JavaScript support in AI libraries
- TypeScript usage

Node-api/Node-addon-api

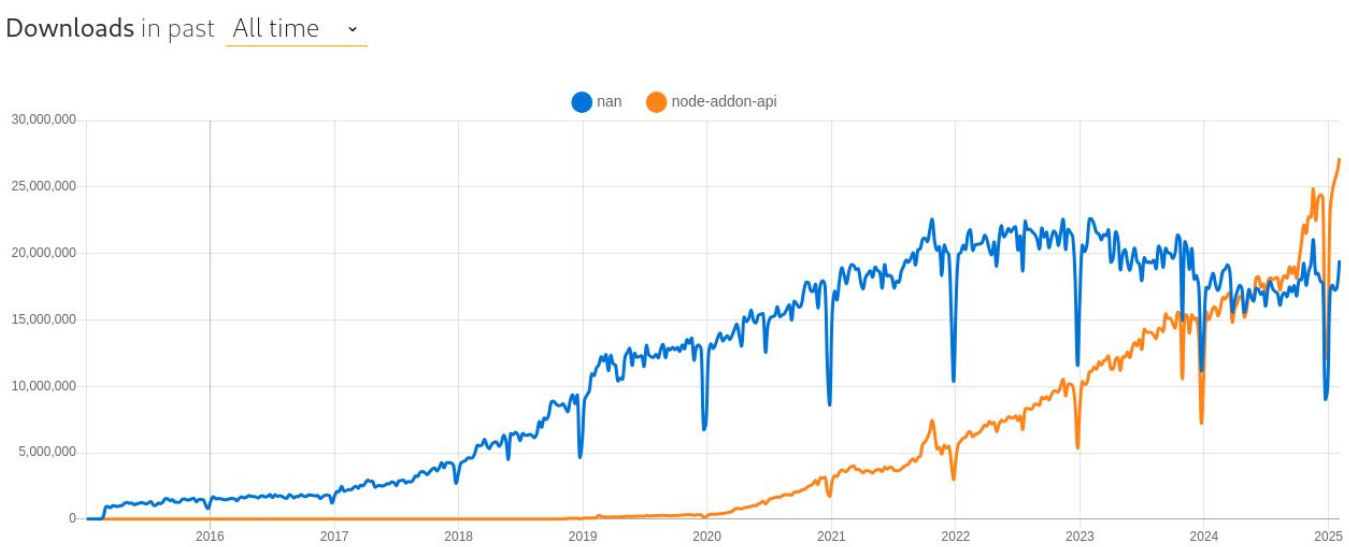
- ABI stable
  - No code changes needed for new Node.js versions
  - Build once, run with later Node.js versions
- Concepts and operations generally map to ideas specified in the ECMA262 Language Specification.
  - Cross runtime compatibility
  - Cross language support
  - Separation from Node.js itself

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)
```

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

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

Building addons like its 2025



Stats

	Stars	Issues	Version	Updated	Created	Size
nan	3,296	73	2.22.0	4 months ago	11 years ago	install size 1.09 MB
node-addon-api	2,215	14	8.3.0	3 months ago	8 years ago	install size 394 kB

80% use case
~ 58% use

https://npmtrends.com/nan-vs-node-addon-api

Trends - AI developer libraries

- JavaScript/Typescript 2nd language supported

ollama-js

llama-stack-client-typescript

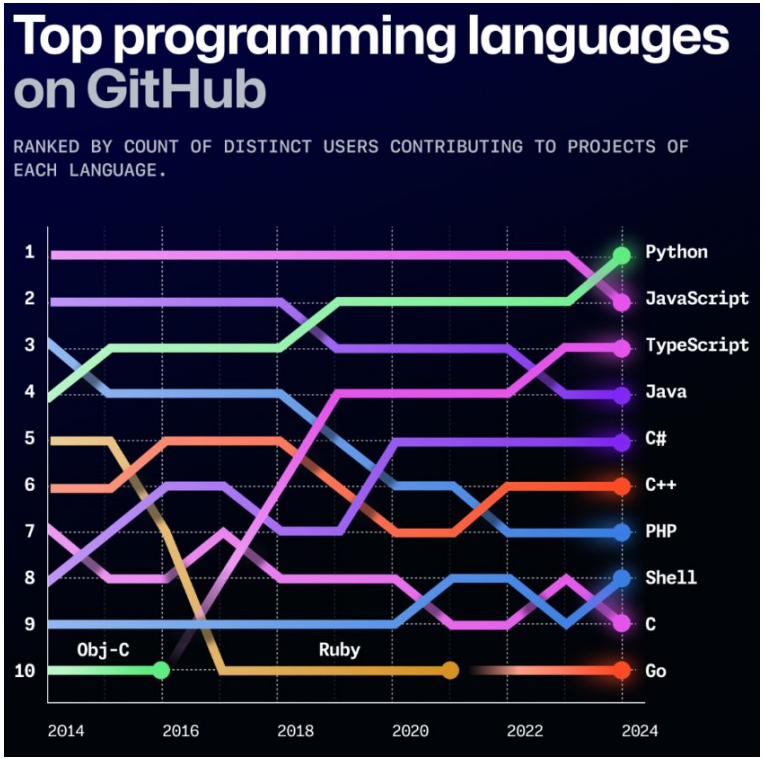
LlamaIndex

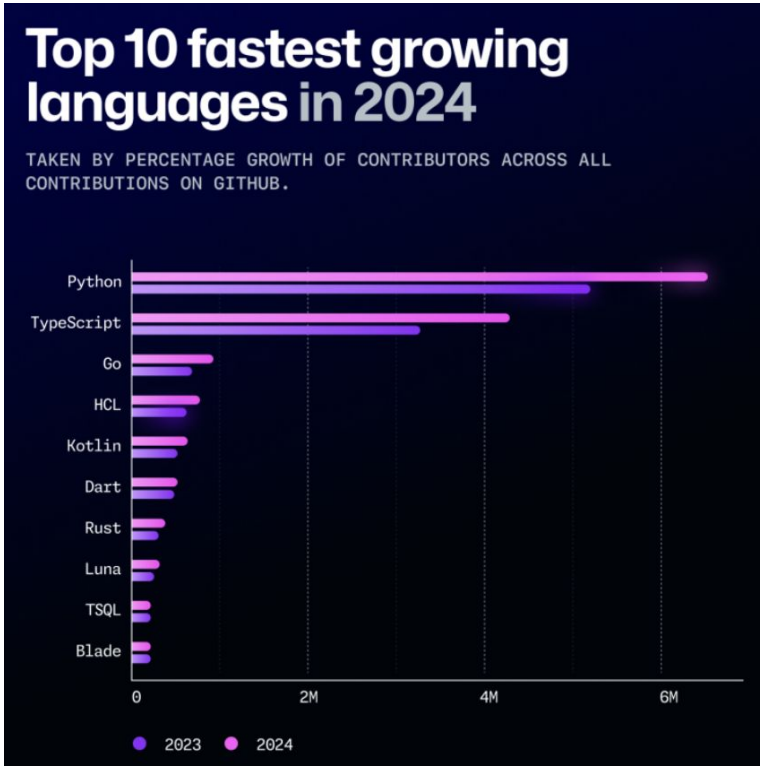
Trends - AI developer libraries

```
import {fileURLToPath} from "url";
import path from "path";
import { ChatPromptTemplate } from "@langchain/core/prompts";

////////////////////////////////
// GET THE MODEL
const __dirname = path.dirname(fileURLToPath(import.meta.url));
const modelPath = path.join(__dirname, "models", "mistral-7b-instruct-v0.1.Q5_K_M.gguf")
const { LlamaCpp } = await import("@langchain/community/llms/llama_cpp");
const model = await new LlamaCpp({ modelPath: modelPath });

////////////////////////////////
// CREATE CHAIN
const prompt =
  ChatPromptTemplate.fromTemplate(`Answer the following question if you don't know the answer say so:

Question: {input}`);

const chain = prompt.pipe(model);

////////////////////////////////
// ASK QUESTION
console.log(new Date());
let result = await chain.invoke({
  input: "Should I use npm to start a node.js application",
});
console.log(result);
console.log(new Date());
```

https://developers.redhat.com/blog/2024/11/20/essential-ai-tutorials-nodejs-developers

Trends - TypeScript usage





What’s Next ?

- Who decides what’s next ?
- Following what’s next
- Next-10

Who Decides what's next ?

~~TSC~~

~~Companies~~

~~An individual contributor~~

Who Decides what's next ?

All collaborators

Who Decides what's next ?

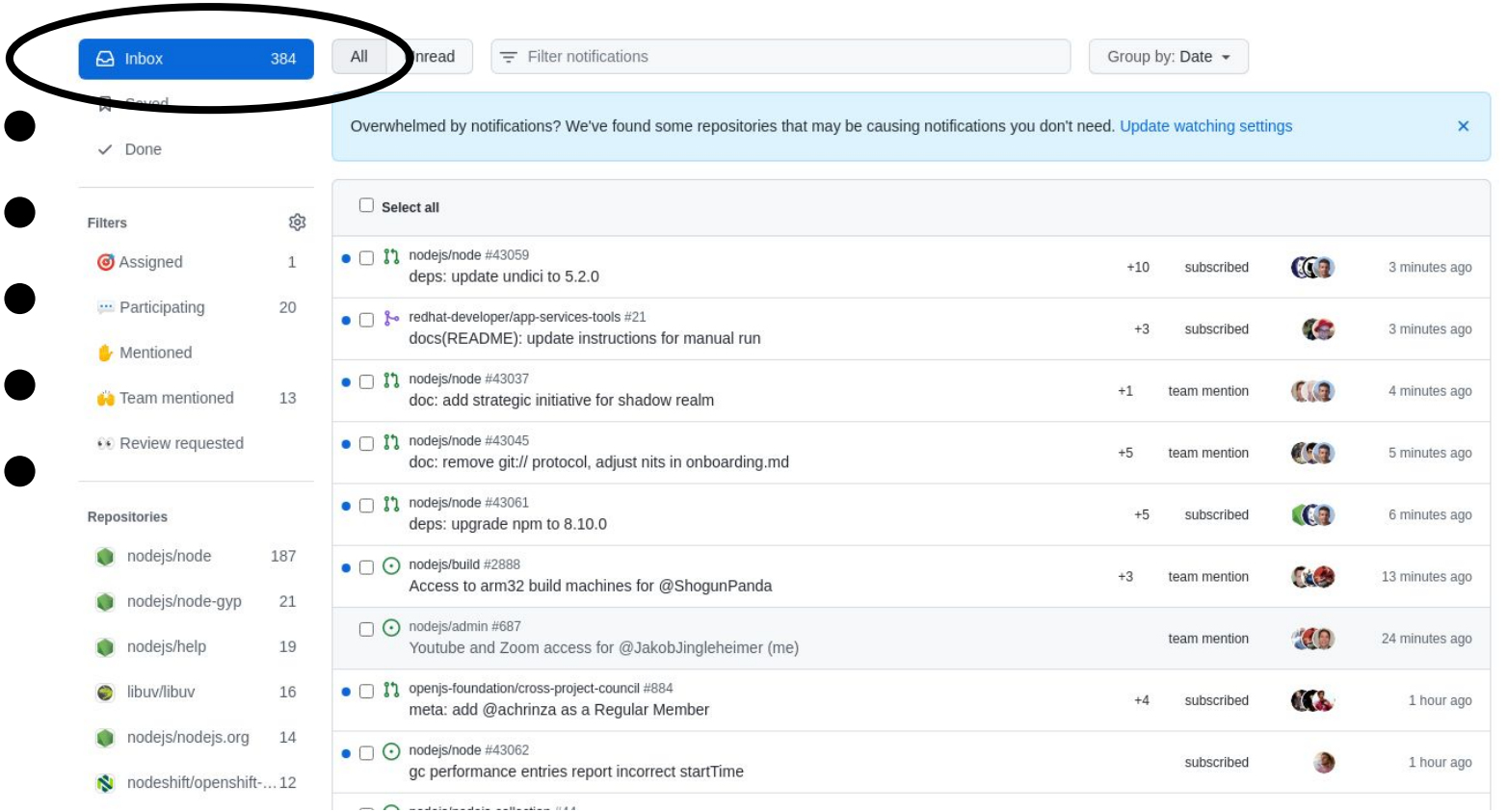
- All collaborators
  - Together
  - Real time
  - No pre-defined roadmap
- Driven by
  - Doer's
  - Reviewers/Approvers
  - Supported by Project's documented Priorities

What’s Next

What's Next

- 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 - https://github.com/nodejs/release
- Node.js news feed https://nodejs.github.io/nodejs-news-feeder/feed.xml

Following What's Next - from the Ground up



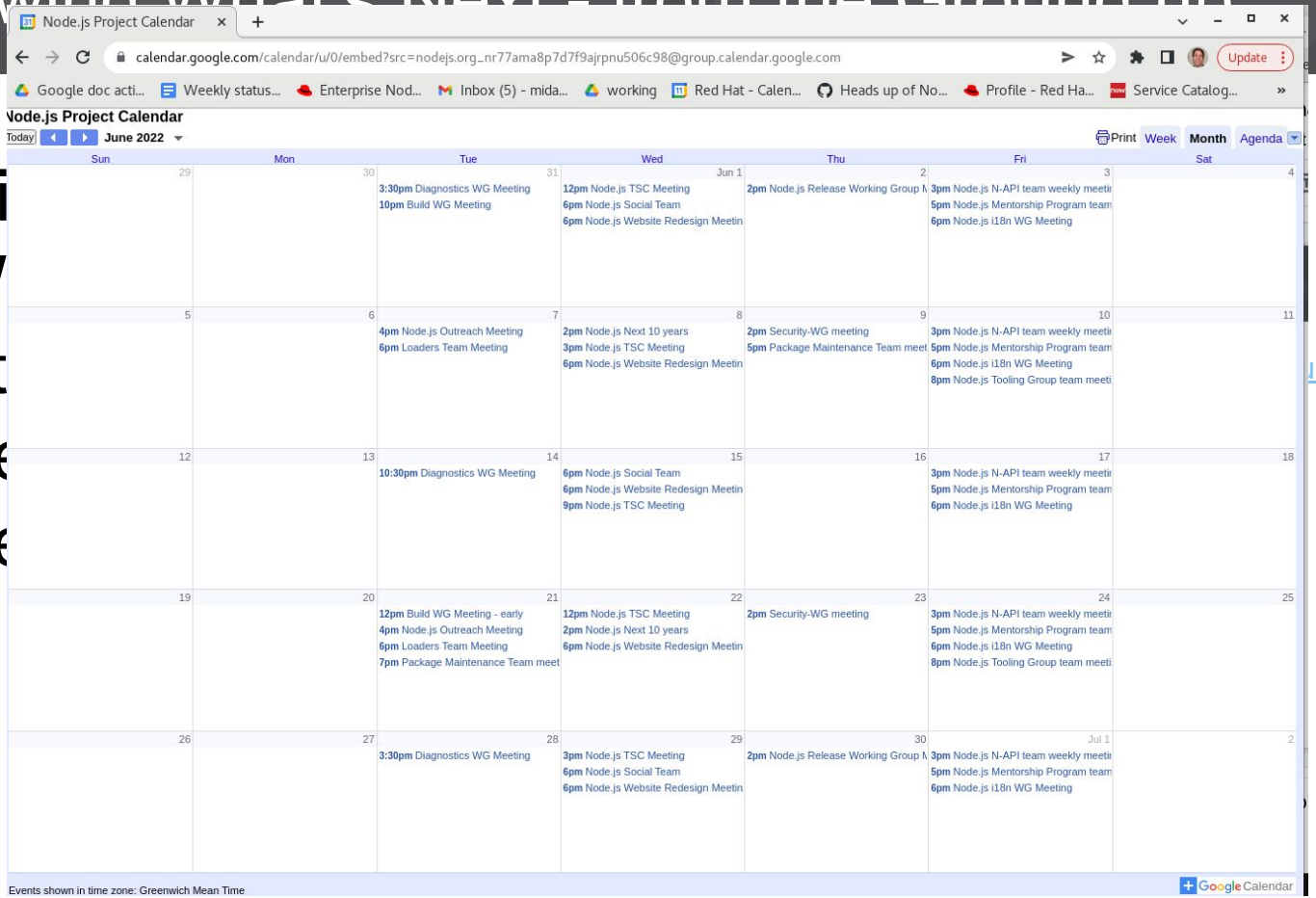
s/strategic-

What's Next

- 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 - https://github.com/nodejs/release
- Node.js news feed https://nodejs.github.io/nodejs-news-feeder/feed.xml

Following What's Next - from the Ground up

- Gi
- W
- St
- Ne
- Re



uides/strategic-

What's Next

- 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 - https://github.com/nodejs/release
- Node.js news feed https://nodejs.github.io/nodejs-news-feeder/feed.xml

Next 10 - Foundation

- Technical Values and Priorities - https://github.com/nodejs/node/blob/main/doc/contributing/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

N

master ▾ node / doc / guides / technical-values.md Go to file ···

Trott doc: use "Long Term Support" in technical values doc ··· Latest commit 0ba7da6 on May 29 History

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

- 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

- /nodejs/node/blob/master/doc/gu
ues.md
- STITUENCIES.md
- ob/main/CONSTITUENCY-NEEDS.md

174 lines (126 sloc) | 6.18 KB

Raw

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		
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**
- **WebAssembly**
- **ES Modules (ESM)**
- **Support for features from the latest ECMAScript spec**
- **Observability**
- **Better multithreaded support**
- **Single Executable Applications**
- **Serverless**
- **Small footprint**
- **Developers-first DX**
- **Package management**

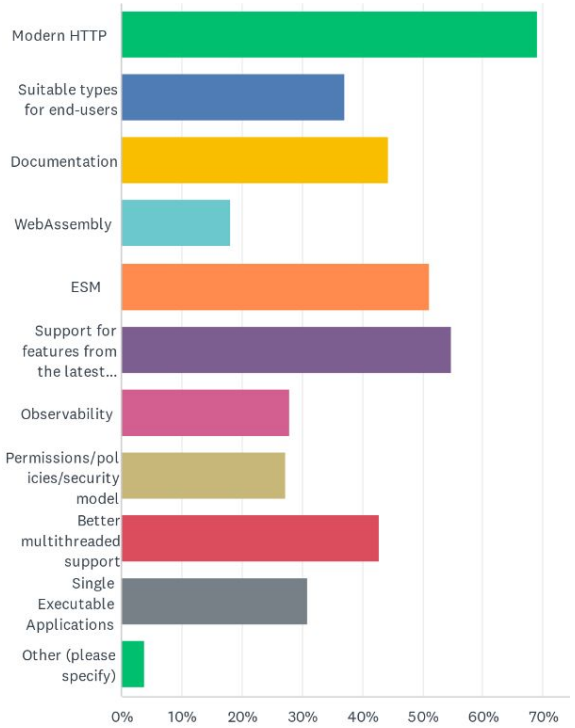
Next 10 team

- Technical priorities well defined
  - Aligned based on survey results
- Current focus areas less technical
  - Advocacy and promotion
    - Ambassador program
    - Corporate usage stories
    - News Feed - https://nodejs.github.io/nodejs-news-feeder/feed.xml
  - Funding
  - Collaborator Health

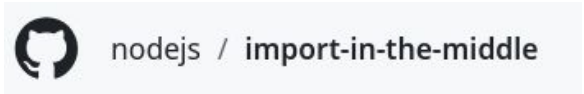
https://github.com/nodejs/node/blob/main/doc/contributing/advocacy-ambasador-program.md

What's next

- Continued work on
  - ESM
  - Typescript
  - Performance
  - QUIC
  - …



What's Next?

- What about other pain points ?
  - Docs Initiative - https://github.com/nodejs/next-10/issues/166
  - Monitoring/Diagnostics
    - diagnostics team
      - AsyncLocalStorage rewritten not to use Async Hooks
    - Key components moved into org
      - 

      - require-in-the middle ?

Most common pain points
- ESM integration
- TypeScript
- Docs ✅
- Monitoring/Diagnostics ✅
- Performance

Thank You

Questions?

Copyright and Trademarks