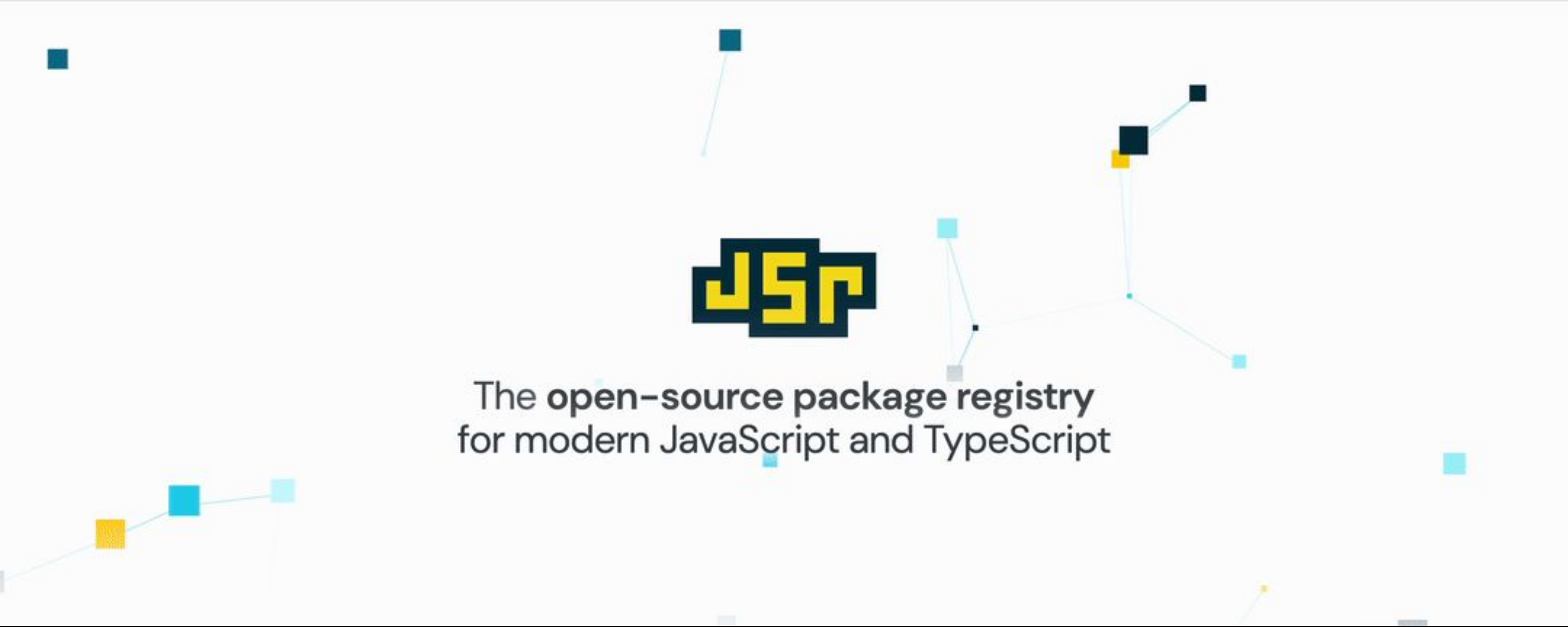


The **open-source package registry** for modern JavaScript and TypeScript

JSR

Behind The Scenes & native support in the ecosystem

WEH 2025/06/02

What is JSR?

JSR is a modern JavaScript registry that:

- supports TypeScript out of the box
- does ***not*** replace npm, it lives alongside it
- compatible across different runtimes
- completely open-source and free
- Has built-in support by yarn and pnpm (but not npm)

Why?

Complexity

Publishing to NPM is **too complicated** nowadays.

Getting TypeScript to work well in npm packages is way too difficult.

Innovation

NPM has **stopped innovating**.

The JavaScript ecosystem is known for rapid change and progress, but NPM doesn't follow suit.

Our Values

Unlike **NPM which is owned by Microsoft**, **JSR is open*.**

Open source, community, and open governance are core principles.

Additional features

Tokenless auth

Allows publish packages automatically from CI without you having to configure any static secrets. This is done with OIDC tokens.

Provenance

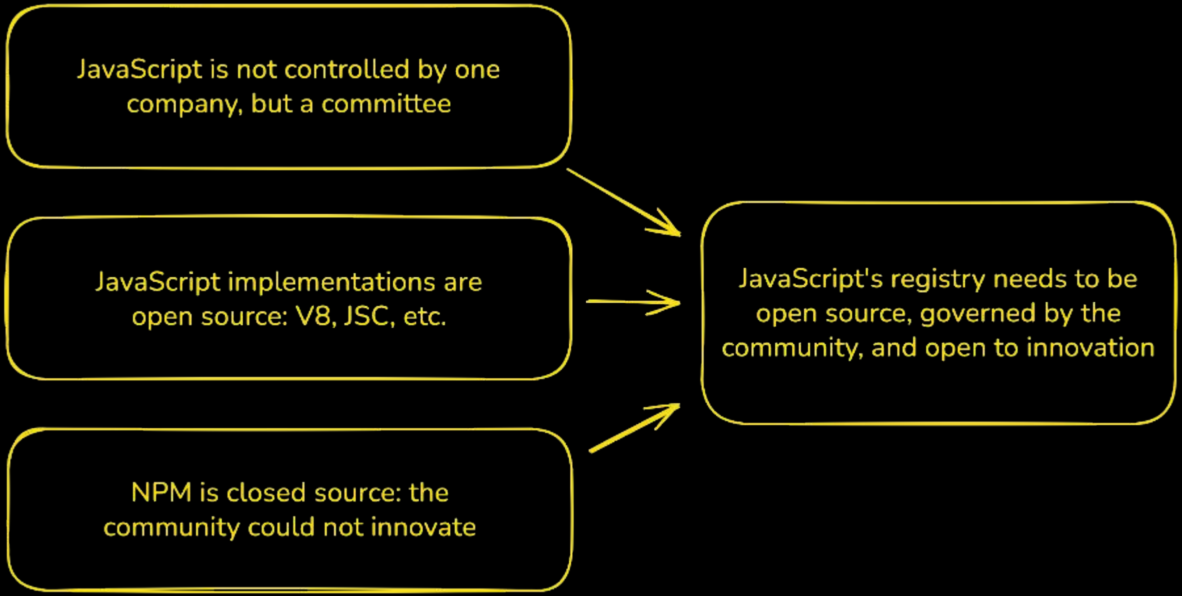
If you use GitHub Actions, JSR can generate package public provenance statements, which further proves the integrity of published packages.

Documentation

JSR automatically generates documentation for your package from its source code and types; no need to set up a separate system or docs site.

Governance

Why is JSR open?



JavaScript is not controlled by one company, but a committee

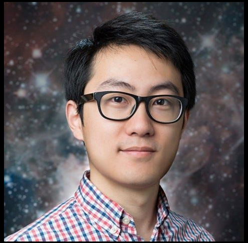
JavaScript implementations are open source: V8, JSC, etc.

NPM is closed source: the community could not innovate

JavaScript's registry needs to be open source, governed by the community, and open to innovation

Board Members

Evan You	creator of Vue.js and Vite, founder of VoidZero
Isaac Schlueter	creator of npm, cofounder of vlt.sh
James Snell	Node.js TSC member, Principal System Engineer at Cloudflare
Luca Casonato	software Engineer at Deno, TC39 representative
Ryan Dahl	creator of Node.js and Deno



What does it do?

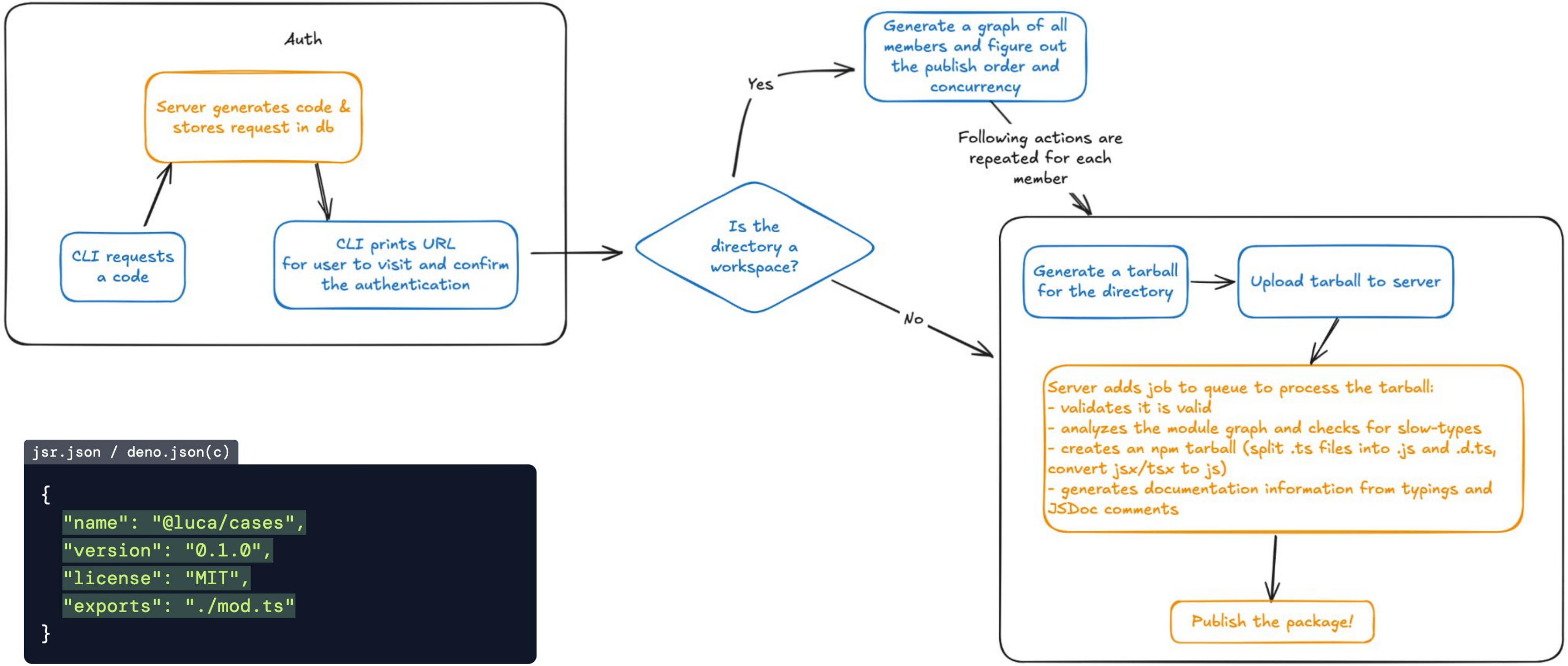
- Overseeing JSR's move to a foundation
- Setting the general direction of the project
- Making decisions on behalf of the project when necessary
- Determining how the governance board is to be elected in the future
- Determining how to review open source contributions to the project

Moderation Group

Oliver Medhurst	creator of porffor
Augustin Mauroy	contributor to NodeJS
Doctor/BlackAsLight	contributor to JSR and Deno

Behind The Scenes

The CLI & publishing



Auth

Server generates code & stores request in db

CLI requests a code

CLI prints URL for user to visit and confirm the authentication

Is the directory a workspace?

Yes

Generate a graph of all members and figure out the publish order and concurrency

Following actions are repeated for each member

No

Generate a tarball for the directory

Upload tarball to server

Server adds job to queue to process the tarball:
- validates it is valid
- analyzes the module graph and checks for slow-types
- creates an npm tarball (split .ts files into .js and .d.ts, convert jsx/tsx to js)
- generates documentation information from typings and JSDoc comments

Publish the package!

jsr.json / deno.json(c)

```
{
  "name": "@luca/cases",
  "version": "0.1.0",
  "license": "MIT",
  "exports": "./mod.ts"
}
```

How do package managers work with JSR?

- For npm compatible clients, npm.jsr.io
- Npm: you need to use .npmrc to specify the registry (JSR cli does this for you)
- Yarn, PNPM and vlt: they understand jsr: entries in dependencies field in package.json, internally point to the JSR registry
- Deno: gets the source directly

Module loading

Deno currently does the following:

- We discover all the jsr specifiers first before we start resolving them
- Get https://jsr.io/@std/collections/meta.json (contains information about the versions available in this package)
- Get https://jsr.io/@std/collections/1.1.1_meta.json (contains the package manifest, exports, and module graphs for all the entrypoints)
- Get the requested entrypoint by the user, and any of its dependencies through regular HTTP imports
- Load the files as normal, it's just an import

What’s next?

- Continuous work on moving to a foundation
- See to it that the NPM cli adds native support JSR
- Allow browsers to import JSR packages via https
- Get popular NPM packages moved over to JSR

Thank you

We have bi-weekly open working group meetings / office hours.

deno.co/jsr-meeting

jsr.io

github.com/jsr-io/jsr