Bun is killing Node.js

Code For Real
@codeforreal

Code For Real
@codeforreal

Give a follow

Swipe to see detail explanation f Bun & why it might replace Node.js

Dont' forget to save it

Save

Bun is a fast, all-in-one toolkit for running, building, testing, and debugging **JavaScript** and **TypeScript**, from a single file to a full-stack application. It ships as a single executable called bun. At its core is the Bun runtime, a fast JavaScript runtime designed as a drop-in replacement for **Node.js.** It's written in Zig and powered by **JavaScriptCore** under the hood, dramatically reducing startup times and memory usage.

Bun version **1.0** is stable and finally production-ready.

Why Bun?

Bun's goal is simple: eliminate slowness and complexity without throwing away everything that's great about JavaScript. Your favorite libraries and frameworks should still work, and you shouldn't need to unlearn the conventions you're familiar with.

Bun is a drop-in replacement for **Node.js**, so you don't need:

- node
- npx — bunx is 5x faster
- nodemon — Bun has a built-in watch mode!
- dotenv, cross-env — Bun reads .env files by default

Bun is an npm-compatible package manager with familiar commands. It reads your package.json and writes to node_modules, just like other package managers, so you can replace:

- npm, .npmrc, package-lock.json
- yarn, yarn.lock
- pnpm, pnpm.lock, pnpm-workspace.yaml
- lerna

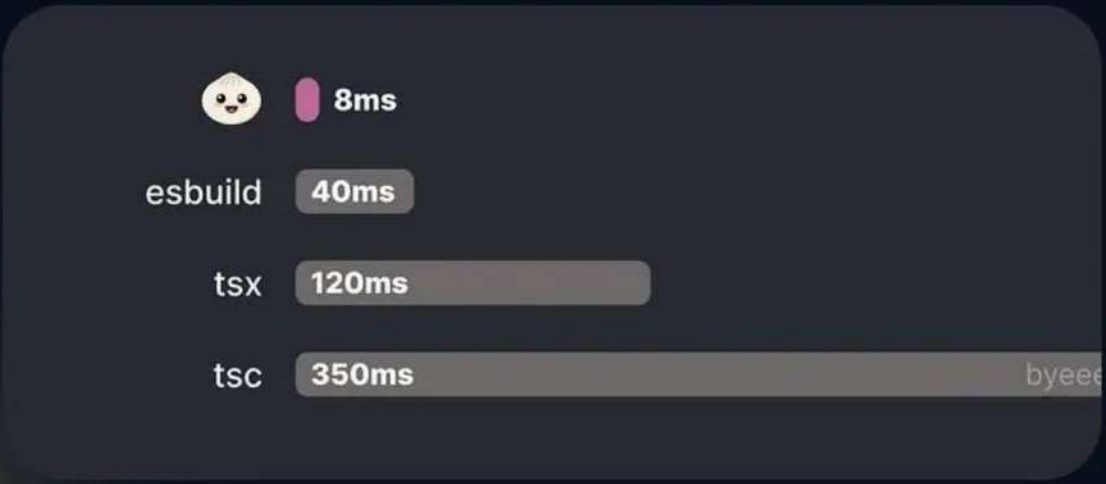
Bun is a JavaScript bundler with best-in-class performance and an esbuild-compatible plugin API, so you don't need:

- esbuild
- webpack
- parcel, .parcelrc
- rollup, rollup.config.js

Bun is tested against test suites of the most popular Node.js packages on npm. Server frameworks like Express, Koa, and Hono just work. As do applications built using the most popular full-stack frameworks. Collectively, these libraries and frameworks touch every part of Node.js's API surface that matters.

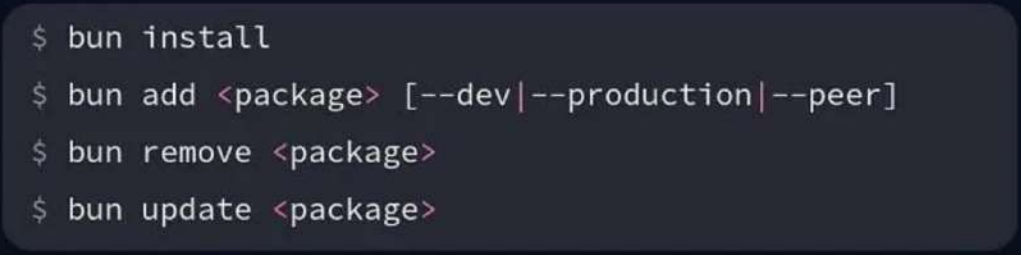
BUN is FAST!

Bun is fast, starting up to 4x fasterthan Node.js. This difference is only magnified when running a TypeScript file, which requires transpilation before it can be run by Node.js.

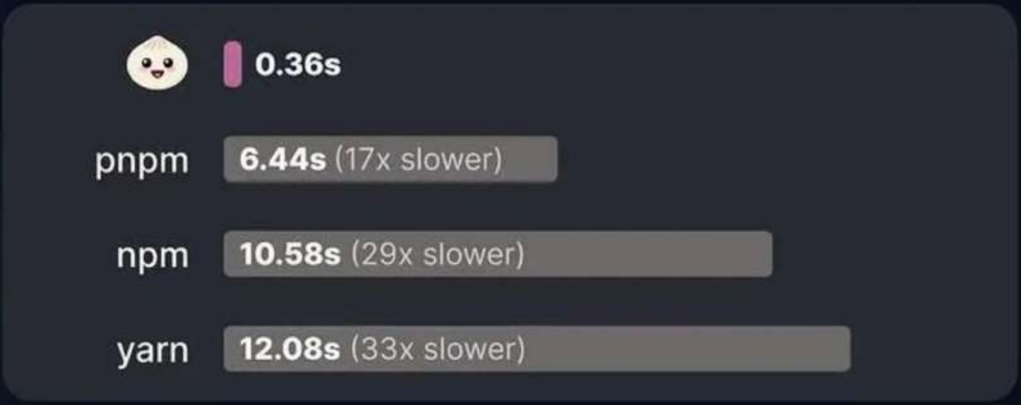


Package Manager

Even if you don't use Bun as a runtime, Bun's built-in package manager can speed up your development workflow. Gone are the days of staring at that npm spinner as your dependencies install.
Bun may look like the package managers you're used to —



```
$ bun install
$ bun add <package> [--dev|--production|--peer]
$ bun remove <package>
$ bun update <package>
```



ESM & CommonJS

The transition from CommonJS to ES modules has been slow and full of terrors. After ESM was introduced, Node.js took 5 years before supporting it without an --experimental-modules flag. Regardless, the ecosystem is still full of CommonJS.

Bun supports both module systems, all the time. You can even use import and require(), in the same file. It just works.

```
import lodash from "lodash";
const _ = require("underscore");
```

i post about Tech, Coding and Career

Follow for more content like this

@alighouri