



Utoo WASM: Ultra-Fast Dev Toolkit for the AI Era

Dong Binghui · Ant Group



W3C AC meeting 2026
Hangzhou, China
20-22 April 2026





The AI Era: A New Paradigm

2

Intent-driven development

Developers express intent, AI generates code

Env setup is still the **bottleneck**

Coding barrier drops, but setup remains painful

Build speed is the new critical path

AI generates fast, toolchains can't keep up



Vibe Coding: Solutions Today

3

VM Backend — high server cost & latency

ESM Bundless — slow HMR, hard to adopt

WebContainer — poor perf, no upgrade path

WASI REPL — ops complexity too high



A Different Approach: Utoo WASM



Utoo — **Unified Toolchain**: Open & Optimized

5

Package Manager (utoo) — Rust, parallel, npm-compatible

Bundler (@utoo/pack) — Turbopack, PRs upstream

Browser (@utoo/web) — entire toolchain as WASM

One codebase: CLI · Node.js · Browser



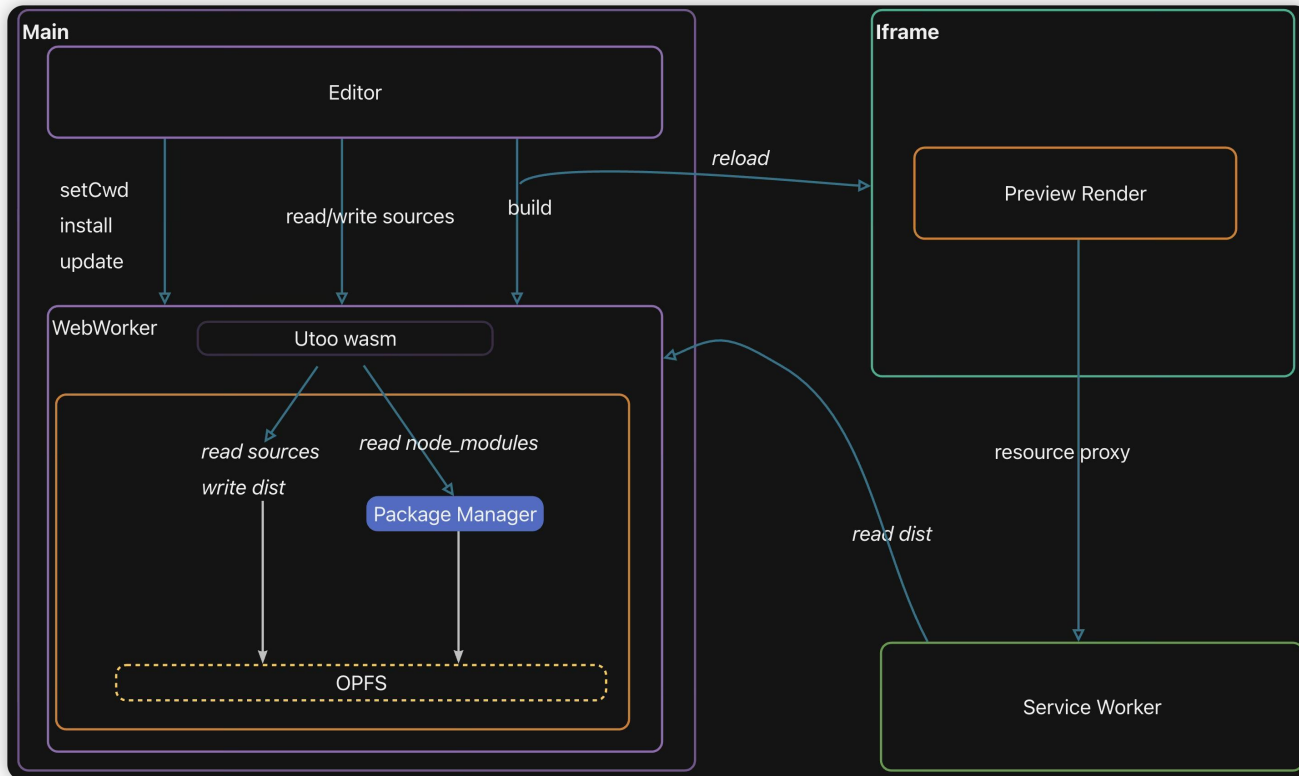
Demo: Bundling in the Browser

The screenshot displays a web development environment with three main panels: Project, Editor, and Preview.

- Project Panel:** Shows a file tree for a project named `/utooweb-demo`. The tree includes:
 - `node_modules` (expanded)
 - `src` (expanded)
 - `package-lock.json`
 - `package.json`
 - `utoopack.json`
- Editor Panel:** Shows a code editor with a single line of code: `1 |`.
- Preview Panel:** Displays a message: "No preview available" with a sub-message: "Click the Build button to build the project and see the preview."

At the bottom of the interface, there is a status bar with the text "Utoo REPL" on the left, "Powered by" in the center, and a timer showing "00:00" and a red button labeled "结束录制" (End Recording) on the right.

Architecture





Web Standards at Work

8

| Web Standard | How We Use It | Impact |
|-------------------|------------------------|---------------------------|
| WebAssembly | Rust core in browser | Near-native perf |
| SharedArrayBuffer | Shared memory threads | True parallelism |
| Web Workers | Multi-threaded Tokio | webpack + service + build |
| OPFS | Persistent dep & cache | Cross-project reuse |



Turbopack + WebWorker

Incremental

TurboTasks reactive engine

Only recompute **what changed**

Fine-grained **dep tracking**

Instant HMR feedback

Multi-threading

Tokio runtime in WASM

SharedArrayBuffer threads

13 parallel workers

Work-stealing **scheduler**



Performance

Fuse

Download **tgz** only

Extract **on-demand**

No `node_modules`

Deps downloaded **once**



10

Ruborist

npm resolution in WASM

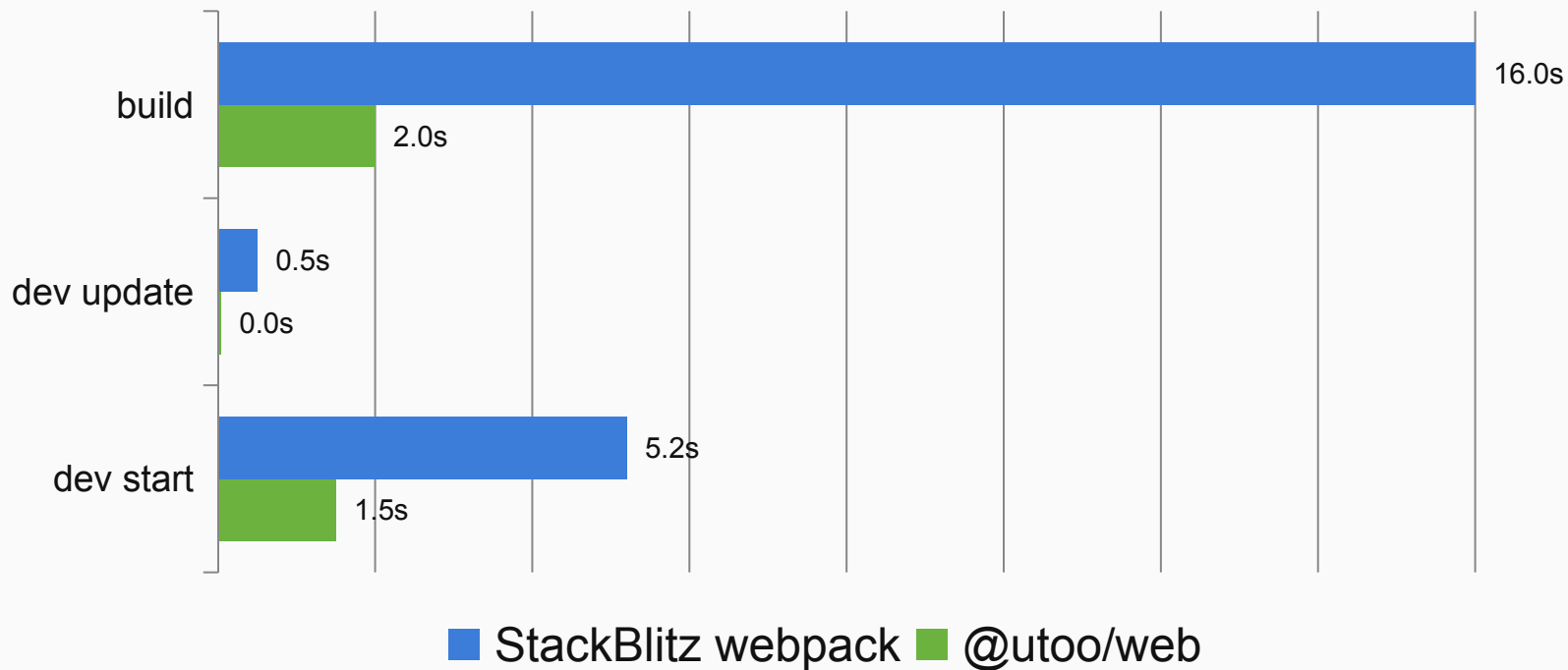
Direct **registry** requests

Client-side lock file

Cache: `mem/OPFS/net`



@ utoo/web vs StackBlitz Webpack





AI Needs **Fast Toolchains**

Intent → AI generates → **instant** compile & verify

Speed = interaction quality, seconds not minutes

Browser-native: zero setup for anyone



Takeaways

Unified, ultra-fast, AI-era toolchain

Full toolchain in the **browser** via WASM

Built on **Web standards**

<https://utoo.land>



<https://github.com/utooland/utoo>