

Why Vite



- [The Origins](#)
 - [Growing with the Ecosystem](#)
 - [A Unified Toolchain](#)
 - [Where Vite is Heading](#)
-

As web applications have grown in size and complexity, the tools used to build them have struggled to keep up. Developers working on large projects have experienced painfully slow dev server startups, sluggish hot updates, and long production build times. Each generation of build tooling has improved on the last, but these problems have persisted.

Vite was created to address this. Rather than incrementally improving existing approaches, it rethought how code should be served during development. Since then, Vite has evolved through multiple major versions, each time adapting to new capabilities in the ecosystem: from leveraging native ES modules in the browser, to adopting a fully Rust-powered toolchain.

Today, Vite powers many frameworks and tools. Its architecture is designed to evolve with the web platform rather than lock into any single approach, making it a foundation you can build on for the long term.

The Origins

When Vite was first created, browsers had just gained wide support for [ES modules](#) (ESM), a way to load JavaScript files directly, without needing a tool to bundle them into a single file first. Traditional build tools (often called *bundlers*) would process your entire application upfront before anything could be shown in the browser. The larger the app, the longer you waited.

Vite took a different approach. It split the work into two parts:

- **Dependencies** (libraries that rarely change) are [pre-bundled](#) once using fast native tooling, so they're ready instantly.
- **Source code** (your application code that changes frequently) is served on-demand over native ESM. The browser loads only what it needs for the current page, and Vite transforms each file as it's requested.

This meant dev server startup was nearly instant, regardless of application size. When you edited a file, Vite used [Hot Module Replacement](#) (HMR) over native ESM to update just that module in the browser, without a full page reload or waiting for a rebuild.

In a bundle-based dev server, the entire application is bundled before it can be served.

In an ESM-based dev server, modules are served on-demand as the browser requests them.

Vite was not the first tool to explore this approach. [Snowpack](#) pioneered unbundled development and inspired Vite's dependency pre-bundling. [WMR](#) by the Preact team inspired the universal plugin API that works in both dev and build. [@web/dev-server](#) influenced Vite 1.0's server architecture. Vite built on these ideas and carried them forward.

Even though unbundled ESM works well during development, shipping it in production is still inefficient due to additional network round trips from nested imports. That's [why bundling is still necessary](#) for optimized production builds.

Growing with the Ecosystem

As Vite matured, frameworks began adopting it as their build layer. Its [plugin API](#), based on Rollup's conventions, made integration natural without requiring frameworks to work around Vite's internals. [Nuxt](#), [SvelteKit](#), [Astro](#), [React Router](#), [Analog](#), [SolidStart](#), and others chose Vite as their foundation. Tools like [Vitest](#) and [Storybook](#) built on it too, extending Vite's reach beyond app bundling. Backend frameworks like [Laravel](#) and [Ruby on Rails](#) integrated Vite for their frontend asset pipelines.

This growth was not one-directional. The ecosystem shaped Vite as much as Vite shaped the ecosystem. The Vite team runs [vite-ecosystem-ci](#), which tests major ecosystem projects against every Vite change. Ecosystem health is not an afterthought. It is part of the release process.

A Unified Toolchain

Vite originally relied on two separate tools under the hood: [esbuild](#) for fast compilation during development, and [Rollup](#) for thorough optimization in production builds. This worked, but maintaining two pipelines introduced inconsistencies: different transformation behaviors, separate plugin systems, and growing glue code to keep them aligned.

[Rolldown](#) was built to unify both into a single bundler: written in Rust for native speed, and compatible with the same plugin API the ecosystem already relied on. It uses [Oxc](#) for parsing, transforming, and minifying. This gives Vite an end-to-end toolchain where the build tool, bundler, and compiler are maintained together and evolve as a unit.

The result is one consistent pipeline from development to [production](#). The migration was done carefully: a [technical preview](#) shipped first so early adopters could validate the change, ecosystem CI caught compatibility issues early, and a compatibility layer preserved existing configurations.

Where Vite is Heading

Vite's architecture continues to evolve. Several efforts are shaping its future:

- **Full bundle mode:** Unbundled ESM was the right tradeoff when Vite was created because no tool was both fast enough and had the HMR and plugin capabilities needed to bundle during dev. Rolldown changes that. Since exceptionally large codebases can experience slow page loads due to the high number of unbundled network requests, the team is exploring a mode where the dev server bundles code similarly to production, reducing network overhead.
- **Environment API:** Instead of treating "client" and "SSR" as the only two build targets, the [Environment API](#) lets frameworks define custom environments (edge runtimes, service workers, and other deployment targets), each with their own module resolution and execution rules. As where and how code runs continues to diversify, Vite's model expands with it.
- **Evolving with JavaScript:** With Oxc and Rolldown closely collaborating with Vite, new language features and standards can be adopted quickly across the entire toolchain, without waiting on upstream dependencies.

Vite's goal is not to be the final tool, but to be one that keeps evolving with the web platform, and with the developers building on it.

🔗 #2	
📄 📄	📄 2026-10-31 13:15:16 UTC
📄 📄	📄 2026-10-31 13:15:16 UTC