Getting Started

Overview

Vite (French word for "quick", pronounced `/vit/` , like "veet") is a build tool that aims to provide a faster and leaner development experience for modern web projects. It consists of two major parts:

- A dev server that provides rich feature enhancements over native ES modules, for example extremely fast Hot Module Replacement (HMR).
- A build command that bundles your code with Rollup, pre-configured to output highly optimized static assets for production.

Vite is opinionated and comes with sensible defaults out of the box, but is also highly extensible via its Plugin API and JavaScript API with full typing support.

You can learn more about the rationale behind the project in the Why Vite section.

Browser Support

The default build targets browsers that support native ES Modules, native ESM dynamic import, and `import.meta` . Legacy browsers can be supported via the official @vitejs/plugin-legacy - see the Building for Production section for more details.

Trying Vite Online

You can try Vite online on StackBlitz. It runs the Vite-based build setup directly in the browser, so it is almost identical to the local setup but doesn't require installing anything on your machine. You can navigate to `vite.new/{template}` to select which framework to use.

The supported template presets are:

JavaScript	TypeScript
vanilla	vanilla-ts
vue	vue-ts
react	react-ts
preact	preact-ts
lit	lit-ts
svelte	svelte-ts

Scaffolding Your First Vite Project

Compatibility Note

Vite requires **Node.js** version 14.18+, 16+. However, some templates require a higher Node.js version to work, please upgrade if your package manager warns about it.

With NPM:

```sh
$ npm create vite@latest
```

With Yarn:

```sh
$ yarn create vite
```

With PNPM:

```sh
$ pnpm create vite
```

Then follow the prompts!

You can also directly specify the project name and the template you want to use via additional command line options. For example, to scaffold a Vite + Vue project, run:

```sh
# npm 6.x
npm create vite@latest my-vue-app --template vue

# npm 7+, extra double-dash is needed:
npm create vite@latest my-vue-app -- --template vue

# yarn
yarn create vite my-vue-app --template vue

# pnpm
pnpm create vite my-vue-app --template vue
```

See create-vite for more details on each supported template: `vanilla` , `vanilla-ts` , `vue` , `vue-ts` , `react` , `react-ts` , `preact` , `preact-ts` , `lit` , `lit-ts` , `svelte` , `svelte-ts` .

Community Templates

create-vite is a tool to quickly start a project from a basic template for popular frameworks. Check out Awesome Vite for community maintained templates that include other tools or target different frameworks. You can use a tool like degit to scaffold your project with one of the templates.

```sh
npx degit user/project my-project
```

```
cd my-project

npm install
npm run dev
```

If the project uses `main` as the default branch, suffix the project repo with `#main`

```
npx degit user/project#main my-project
```

sh

`index.html` and Project Root

One thing you may have noticed is that in a Vite project, `index.html` is front-and-central instead of being tucked away inside `public`. This is intentional: during development Vite is a server, and `index.html` is the entry point to your application.

Vite treats `index.html` as source code and part of the module graph. It resolves `<script type="module" src="...">` that references your JavaScript source code. Even inline `<script type="module">` and CSS referenced via `<link href>` also enjoy Vite-specific features. In addition, URLs inside `index.html` are automatically rebased so there's no need for special `%PUBLIC_URL%` placeholders.

Similar to static http servers, Vite has the concept of a "root directory" which your files are served from. You will see it referenced as `<root>` throughout the rest of the docs. Absolute URLs in your source code will be resolved using the project root as base, so you can write code as if you are working with a normal static file server (except way more powerful!). Vite is also capable of handling dependencies that resolve to out-of-root file system locations, which makes it usable even in a monorepo-based setup.

Vite also supports multi-page apps with multiple `.html` entry points.

Specifying Alternative Root

Running `vite` starts the dev server using the current working directory as root. You can specify an alternative root with `vite serve some/sub/dir`.

Command Line Interface

In a project where Vite is installed, you can use the `vite` binary in your npm scripts, or run it directly with `npx vite`. Here are the default npm scripts in a scaffolded Vite project:

```json
{
  "scripts": {
    "dev": "vite", // start dev server, aliases: `vite dev`, `vite serve`
    "build": "vite build", // build for production
    "preview": "vite preview" // locally preview production build
  }
}
```

You can specify additional CLI options like `--port` or `--https`. For a full list of CLI options, run `npx vite --help` in your project.

Using Unreleased Commits

If you can't wait for a new release to test the latest features, you will need to clone the vite repo to your local machine and then build and link it yourself (pnpm is required):

```sh
git clone https://github.com/vitejs/vite.git
cd vite
pnpm install
cd packages/vite
pnpm run build
pnpm link --global # you can use your preferred package manager for this step
```

Then go to your Vite based project and run `pnpm link --global vite` (or the package manager that you used to link `vite` globally). Now restart the development server to ride on the bleeding edge!

Community

If you have questions or need help, reach out to the community at Discord and GitHub Discussions.

Suggest changes to this page

Previous page
Why Vite

Next page
Features