SVELTEKIT • GETTING STARTED

# Project structure

ON THIS PAGE

A typical SvelteKit project looks like this:

```
my-project/
├ src/
│ ├ lib/
│ │ ├ server/
│ │ │ └ [your server-only lib files]
│ │ └ [your lib files]
│ ├ params/
│ │ └ [your param matchers]
│ ├ routes/
│ │ └ [your routes]
│ ├ app.html
│ ├ error.html
│ ├ hooks.client.js
│ ├ hooks.server.js
│ └ service-worker.js
├ static/
│ └ [your static assets]
├ tests/
│ └ [your tests]
├ package.json
├ svelte.config.js
├ tsconfig.json
└ vite.config.js
```

You'll also find common files like `.gitignore` and `.npmrc` (and `.prettierrc` and `eslint.config.js` and so on, if you chose those options when running `npx sv create`).

The `src` directory contains the meat of your project. Everything except `src/routes` and `src/app.html` is optional.

- `lib` contains your library code (utilities and components), which can be imported via the `$lib` alias, or packaged up for distribution using `svelte-package`
  - `server` contains your server-only library code. It can be imported by using the `$lib/server` alias. SvelteKit will prevent you from importing these in client code.
- `params` contains any param matchers your app needs
- `routes` contains the routes of your application. You can also colocate other components that are only used within a single route here
- `app.html` is your page template — an HTML document containing the following placeholders:
  - `%sveltekit.head%` — `<link>` and `<script>` elements needed by the app, plus any `<svelte:head>` content
  - `%sveltekit.body%` — the markup for a rendered page. This should live inside a `<div>` or other element, rather than directly inside `<body>`, to prevent bugs caused by browser extensions injecting elements that are then destroyed by the hydration process. SvelteKit will warn you in development if this is not the case
  - `%sveltekit.assets%` — either `paths.assets`, if specified, or a relative path to `paths.base`
  - `%sveltekit.nonce%` — a CSP nonce for manually included links and scripts, if used
  - `%sveltekit.env.[NAME]%` - this will be replaced at render time with the `[NAME]` environment variable, which must begin with the `publicPrefix` (usually `PUBLIC_`). It will fallback to `''` if not matched.
- `error.html` is the page that is rendered when everything else fails. It can contain the following placeholders:
  - `%sveltekit.status%` — the HTTP status

- `hooks.server.js` contains your server hooks
- `service-worker.js` contains your service worker

(Whether the project contains `.js` or `.ts` files depends on whether you opt to use TypeScript when you create your project. You can switch between JavaScript and TypeScript in the documentation using the toggle at the bottom of this page.)

If you added Vitest when you set up your project, your unit tests will live in the `src` directory with a `.test.js` extension.

## static

Any static assets that should be served as-is, like `robots.txt` or `favicon.png`, go in here.

## tests

If you added Playwright for browser testing when you set up your project, the tests will live in this directory.

## package.json

Your `package.json` file must include `@sveltejs/kit`, `svelte` and `vite` as `devDependencies`.

When you create a project with `npx sv create`, you'll also notice that `package.json` includes `"type": "module"`. This means that `.js` files are interpreted as native JavaScript modules with `import` and `export` keywords. Legacy CommonJS files need a `.cjs` file extension.

## svelte.config.js

This file contains your Svelte and SvelteKit configuration.

This file (or `jsconfig.json`, if you prefer type-checked `.js` files over `.ts` files) configures TypeScript, if you added typechecking during `npx sv create`. Since SvelteKit relies on certain configuration being set a specific way, it generates its own `.svelte-kit/tsconfig.json` file which your own config `extends`.

### vite.config.js

A SvelteKit project is really just a Vite project that uses the `@sveltejs/kit/vite` plugin, along with any other Vite configuration.

## Other files

### .svelte-kit

As you develop and build your project, SvelteKit will generate files in a `.svelte-kit` directory (configurable as `outDir`). You can ignore its contents, and delete them at any time (they will be regenerated when you next `dev` or `build`).

Edit this page on GitHub

PREVIOUS
Creating a project

NEXT
Web standards