SvelteKit Static Site Project Structure

Pre-render to static HTML. Deploy anywhere—Netlify, Vercel, GitHub Pages.

#sveltekit #svelte #typescript #static #ssg #jamstack

PNG PDF Copy Prompt

Project Directory

```
my-site/
  src/
    app.html
    app.css
    lib/
      index.ts
      components/
        Header.svelte
        Footer.svelte
        Nav.svelte
      utils/
        markdown.ts  Parse MD files
    routes/
      +layout.svelte
      +page.svelte
      about/
        +page.svelte
      blog/  Blog section
        +page.svelte  Post list
        +page.ts  Load all posts
        [slug]/
          +page.svelte
          +page.ts  Load single post
      docs/  Documentation
        +layout.svelte  Docs sidebar
        +page.svelte
        [...slug]/  Catch-all for n…
          +page.svelte
          +page.ts
  content/  Markdown content
    blog/
      hello-world.md
      getting-started.md
    docs/
      introduction.md
      installation.md
  static/
    favicon.png
    robots.txt
    images/
  svelte.config.js  adapter-static …
  vite.config.ts
  package.json
  tsconfig.json
  .gitignore
```

Why This Structure?

This structure uses `@sveltejs/adapter-static` to pre-render all routes to HTML at build time. Content lives in `content/` as Markdown files. Perfect for blogs, documentation, and marketing sites that need fast hosting on CDNs.

Key Directories

`content/` - Markdown files for blog posts and docs

`src/routes/blog/[slug]/` - Dynamic routes pre-rendered at build

`src/routes/docs/[...slug]/` - Catch-all for nested doc paths

`static/` - Copied as-is to build output

Static Generation

`+page.ts` - Universal load function—runs at build

`export const prerender` - Set to true in root +layout.ts

`entries()` - Return all paths for dynamic routes

`adapter-static` - Configure fallback and strict mode

Getting Started

```
npm create svelte@latest my-site
npm install @sveltejs/adapter-static
# In svelte.config.js:
# adapter: adapter({ fallback: '404.html' })
npm run build
# Output in /build
```

When To Use This

- Blogs and documentation sites
- Marketing and landing pages
- Portfolio and personal sites
- Sites hosted on CDNs or GitHub Pages

Trade-offs

`No server routes` - API endpoints not available

`Build-time data` - Content updates require rebuild

`No form actions` - Use external services for forms