Remix SPA Mode Project Structure

Client-side SPA with Remix. Deploy to any static host without a server.

#remix #react #typescript #spa #static

PNG PDF Copy Prompt

Project Directory

```
my-remix-spa/
  app/
    root.tsx  App shell and p…
    entry.client.tsx  SPA entry point
    routes/
      _index.tsx
      about.tsx
      dashboard.tsx  Layout for dash…
      dashboard._index.tsx
      dashboard.settings.tsx
      dashboard.profile.tsx
    components/
      ui/
        button.tsx
        input.tsx
        card.tsx
      layout/
        header.tsx
        nav.tsx
    lib/
      api.ts  API client func…
      utils.ts
      constants.ts
    hooks/
      use-api.ts
      use-auth.ts
    context/  React context p…
      auth-context.tsx
      theme-context.tsx
    tailwind.css
  public/
    favicon.ico
    images/
  vite.config.ts  SPA mode config
  package.json
  tailwind.config.ts
  tsconfig.json
  .env  VITE_API_URL
  .gitignore
```

Why This Structure?

SPA Mode renders entirely on the client—no server required. You get Remix routing, nested layouts, and data patterns while deploying to any static host. Ideal when your backend is a separate API and you want Remix DX without Node.js hosting.

Key Directories

app/routes/ - Client-side routes with nested layouts

app/hooks/ - Custom hooks for API calls and state

app/context/ - React context for global state

app/lib/api.ts - API client for external backend

Client Data Loading

Without server loaders, use `clientLoader` for data fetching. It runs on the client and receives the same context. For shared data, fetch in parent layouts and access via `useRouteLoaderData`. Consider React Query or SWR for caching.

Getting Started

```
npx create-remix@latest --template remix-run/remix/tem t
cd my-remix-spa
npm run dev
npm run build  # Outputs to build/client
```

When To Use This

- Backend is a separate API service
- Deploying to static hosts (Netlify, Vercel static)
- Migrating existing SPAs to Remix routing
- Teams wanting Remix DX without server hosting
- Internal dashboards consuming external APIs

Trade-offs

No SSR - Content not available without JavaScript

No server loaders - Data fetching on client only

SEO limitations - Search engines may not index content

When To Upgrade

- SEO becomes important for the project
- Need server-side auth or sessions
- Want progressive enhancement benefits
- Performance requires server rendering