# Electron Vite React Project Structure

Modern Electron stack with Vite, React, and TypeScript. Optimized for fast development and production builds.

#electron #react #vite #desktop #typescript #modern

PNG PDF Copy Prompt

## Project Directory

```
my-electron-app/
  package.json
  vite.config.ts  Renderer config
  vite.main.config.ts  Main process co…
  vite.preload.config.ts  Preload config
  electron-builder.json
  tsconfig.json
  src/
    main/
      index.ts  Main entry
      window.ts  Window factory
      menu.ts  App menu
      ipc/
        index.ts
        handlers.ts
      services/
        store.ts
        updater.ts
    preload/
      index.ts
      api.ts  Exposed APIs
    renderer/  React app
      index.html
      main.tsx
      App.tsx
      components/
        TitleBar.tsx
        Layout.tsx
      pages/
        Home.tsx
        Settings.tsx
      hooks/
        useElectron.ts
      store/  Zustand or Jotai
        index.ts
      styles/
        global.css
    shared/  Cross-process t…
      types.ts
      constants.ts
  resources/
    icon.icns
    icon.ico
    icon.png
```

## Why This Structure?

Optimized Vite setup for Electron. Separate Vite configs for main, preload, and renderer processes. Each compiles independently with proper targets. Shared types between processes ensure type-safe IPC.

## Key Directories

**`vite.*.config.ts`** - Separate Vite configs per process

**`src/shared/`** - TypeScript types shared across processes

**`src/renderer/hooks/`** - React hooks wrapping Electron APIs

**`src/renderer/store/`** - State management with Zustand/Jotai

## Getting Started

1. `npm create electron-vite -- --template react-ts`
2. `npm install`
3. Review Vite configs for each process
4. `npm run dev`

## React Hook

```ts
// src/renderer/hooks/useElectron.ts
import { useCallback } from 'react';

export function useElectron() {
  const readFile = useCallback(
    (path: string) => window.api.readFile(path),
    []
  );

  return { readFile };
}
```

## When To Use This

- Modern React desktop development
- Projects needing fast HMR
- Type-safe IPC requirements
- Production apps with optimized builds

## Best Practices

- Keep Vite configs separate for each process
- Use `src/shared/` for cross-process types
- Target 'node' for main, 'chrome' for renderer
- Use path aliases consistently across configs
- Enable source maps for development only

## Trade-offs

**`Config complexity`** - Three Vite configs to maintain

**`Learning curve`** - Understanding Electron + Vite integration

**`Updates`** - Keep Vite and Electron versions compatible