Developing a SolidJS Component Library with Vite, Tailwind 4, and a Dev App

When building a reusable SolidJS component library (like `NetworkSimulator`), it's important to test and develop in isolation—without needing to publish or rebuild constantly.

This guide outlines a clean development setup using:

- **Vite** for bundling and dev server
- **Tailwind CSS 4** for styling
- **A symlinked dev app** for live testing
- **SolidJS** as the framework

Why This Setup?

Vite's `library mode` is great for final builds, but not for development:

- No hot module reload (HMR)
- Slow feedback loop
- Not meant for interactive testing

To address that, we introduce a **dedicated `dev/` folder**, acting as a standalone test app. This allows us to develop against the **real source code**, with **live reloading**, **Tailwind scanning**, and **HMR**.

Project Structure (Simplified)

```
.
├── dev/
│   ├── index.html          # Dev app entry point
│   ├── main.jsx            # Mounts the app
│   ├── App.jsx             # Test wrapper for component
│   └── src -> ../src       # Symlink to source (important!)
│
├── src/                    # Library source code
├── dist/                   # Built output (ignored during dev)
├── vite.config.js          # Dev + build config
├── package.json
└── tsconfig.json
```

Why the Symlink?

We symlink `src` into the `dev/` directory like this:

```
ln -s ../src dev/src
```

This is crucial because:

- Tailwind 4 only scans **inside the Vite root** (which is `dev/` in dev mode)
- Vite treats code **outside root** as external (no HMR, no scanning)
- The symlink makes `src/` look like part of the root—solving both issues

---

Vite Config (Dual Mode)

```
// vite.config.js
import { defineConfig } from 'vite';
import solidPlugin from 'vite-plugin-solid';
import tailwindcss from '@tailwindcss/vite';

const isLib = process.env.BUILD_LIB === 'true';

export default defineConfig({
  plugins: [tailwindcss(), solidPlugin()],
  root: isLib ? '.' : 'dev',
  build: isLib
    ? {
        target: 'esnext',
        outDir: 'dist',
        lib: {
          entry: 'src/index.jsx',
          name: 'NetworkSimulator',
          fileName: 'index',
          formats: ['es', 'cjs'],
        },
        rollupOptions: {
          external: ['solid-js'],
        },
      }
    : {
        outDir: 'dist-dev',
      },
  server: {
    port: 3000,
  },
});
```

- `BUILD_LIB=true vite build` builds the library.
- Default `vite` runs the dev app from `dev/`.

Dev App Entry – `dev/main.jsx`

```
/* @refresh reload */
import { render } from 'solid-js/web';
import './index.css';
import App from './App';

const root = document.getElementById('root');

if (import.meta.env.DEV && !(root instanceof HTMLElement)) {
  throw new Error('Root element not found. Check your index.html.');
}

render(() => <App />, root);
```

- Enables HMR and JSX refresh
- Renders a test wrapper (`App.jsx`) for development

Dev Test Harness – `dev/App.jsx`

```
import NetworkSimulator from './src/NetworkSimulator';

function App() {
  return (
    <div class="flex items-center justify-center min-h-screen bg-gray-800">
      <div class="bg-gray-800 p-4">
        <h1 class="text-2xl font-bold text-gray-100 m-4">Network Simulator Test</h1>
        <NetworkSimulator />
      </div>
    </div>
  );
}

export default App;
```

- Uses Tailwind classes for styling
- Mounts the actual package export (`NetworkSimulator`)

`package.json`

```json
{
  "name": "network-simulation",
  "version": "0.1.0",
  "type": "module",
  "main": "dist/index.js",
  "module": "dist/index.js",
  "exports": {
    ".": "./dist/index.js"
  },
  "files": [
    "dist"
  ],
  "scripts": {
    "dev": "vite",
    "build": "BUILD_LIB=true vite build",
    "serve": "vite preview"
  },
  "dependencies": {
    "@tailwindcss/vite": "^4.1.2",
    "solid-js": "^1.9.3",
    "tailwindcss": "^4.1.2"
  },
  "devDependencies": {
    "vite": "^6.0.0",
    "vite-plugin-solid": "^2.11.1"
  }
}
```

- dev runs the test app
- build bundles the library
- `files` and `exports` ensure clean npm packaging

`jsconfig.json`

```json
{
  "compilerOptions": {
    "strict": true,
    "target": "ESNext",
    "module": "ESNext",
    "moduleResolution": "node",
    "allowSyntheticDefaultImports": true,
    "esModuleInterop": true,
    "jsx": "preserve",
    "jsxImportSource": "solid-js",
    "types": ["vite/client"],
    "noEmit": true,
    "isolatedModules": true
  }
}
```

Supports JSX, Vite environment, and works well with Solid’s compiler.

Why This Works

- **Live reloads** your actual source files
- Tailwind 4 works perfectly due to symlink in Vite root
- Super fast dev loop with no rebuilds required
- Clean separation of dev vs. production
- Focused, real-world testing of your library

Let me know if you’d like this turned into a downloadable README or formatted Markdown file!