Fastify, Vite, and TypeScript Setup Guide - Part 1

A complete walkthrough for setting up Fastify, Vite, TypeScript, including all the gotchas I encountered along the way.

October 26, 2024 · 5 min · 862 words · Me · 122 views

This is a recap of how I got Fastify, Vite, TypeScript, and Tailwind CSS working together. It was a bit like assembling a complicated jigsaw puzzle without the picture on the box, and I ran into a few unexpected gotchas along the way. Here, I've documented what worked, what didn't, and how I solved the problems that came up. Hopefully, this helps me revisit the setup later or assists anyone else going down the same path.

Setting Up Fastify and Vite

The first piece of this puzzle is getting Fastify and Vite running side by side. Fastify serves as our backend, and Vite as our modern front-end tooling. For a typical app, I like to keep these responsibilities separate but allow them to cooperate when necessary.

First, initialize your project:

```
mkdir fastify-vite-ts
cd fastify-vite-ts
pnpm init
```

Then install Fastify, Vite, and TypeScript, along with some development dependencies:

```
pnpm add fastify vite
pnpm add -D typescript ts-node @types/node @vitejs/plugin-vue
```

After setting up your `tsconfig.json` for TypeScript, I split the project into `src/backend` and `src/frontend` to keep things tidy. Fastify runs on port 3000 and Vite on 3001. To get Vite and Fastify to cooperate, you need a proxy setup in `vite.config.ts` to make API calls work during development:

```
import { defineConfig } from 'vite';
import vue from '@vitejs/plugin-vue';

export default defineConfig({
  plugins: [vue()],
  server: {
    port: 3001,
    proxy: {
      '/api': {
        target: 'http://localhost:3000',
        changeOrigin: true,
        secure: false
      }
    }
```

```
  }
});
```

This way, when Vite needs to make an API call to Fastify, it knows how to forward those requests properly. But this was only the beginning of the hiccups.

Project Structure

Here's the project structure I used to keep everything organized:

```
/fastify-vite-ts
  /src
    /backend
      index.ts
    /frontend
      index.html
      main.ts
  tsconfig.json
  vite.config.ts
  package.json
  postcss.config.js
  tailwind.config.js
```

- `/src/backend/index.ts` : Fastify backend server code.
- `/src/frontend/index.html` : Main HTML file for the front-end.
- `/src/frontend/main.ts` : Entry point for the front-end JavaScript/TypeScript.
- `tsconfig.json` : TypeScript configuration.
- `vite.config.ts` : Vite configuration for building and serving the front-end.

- `postcss.config.js` and `tailwind.config.js` : Configuration files for Tailwind CSS and PostCSS.

Gotcha #1: Serving HTML and 404s

Initially, I was hitting a `404` on `http://localhost:3001` . Turns out, Vite expects your `index.html` to be in the root directory, but I had mine sitting in `src/frontend/` . Moving `index.html` to the root fixed the problem. Alternatively, you can set the `root` property in `vite.config.ts` to point to your frontend folder:

```
1  root: 'src/frontend',
```

Adding Tailwind CSS

Next came Tailwind CSS. Installing Tailwind with Vite should be straightforward, right? Not quite. First, you need to install Tailwind, PostCSS, and Autoprefixer:

```
1  pnpm add -D tailwindcss postcss autoprefixer
```

Then initialize Tailwind:

```
1  npx tailwindcss init
```

This creates a `tailwind.config.js` file. Make sure to adjust the `content` property to point to your front-end files:

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    './index.html',
    './src/frontend/**/*.{vue,js,ts,jsx,tsx}'
  ],
  theme: {
    extend: {},
  },
  plugins: [],
};
```

Gotcha #2: PostCSS Config and ES Modules

When I first tried to run everything, I got a dreaded `ReferenceError: module is not defined in ES module scope`. This error happened because my project is using ES modules (`"type": "module"` in `package.json`), and `postcss.config.js` was still using the CommonJS export syntax (`module.exports`).

To fix this, I changed `postcss.config.js` to use the ES module syntax:

```
export default {
  plugins: {
    tailwindcss: {},
    autoprefixer: {},
  },
};
```

Once I made that change, everything played nicely, and Tailwind CSS was able to process

correctly.

Running Backend and Frontend Together

The next logical step was figuring out how to run both Fastify and Vite at the same time during development. I wanted something easy, without having to manually start two terminals every time.

I chose `concurrently`, which makes running multiple scripts a breeze:

```
pnpm add -D concurrently
```

Then I added a script in `package.json`:

```
"scripts": {
  "backend": "tsx src/backend/index.ts",
  "frontend": "vite",
  "dev": "concurrently -k \"pnpm run backend\" \"pnpm run frontend\""
}
```

Now, running `pnpm run dev` kicks off both servers at once, and I can see logs from both Fastify and Vite in one place. It's the little things like this that make development a lot smoother.

Conclusion

Setting up Fastify, Vite, TypeScript, and Tailwind CSS together was a journey with more gotchas than I expected. From file path issues to ESM/CommonJS quirks, there were a lot of bumps along the way. But now that it's working, it feels snappy and modern. I hope this guide helps you avoid some of the headaches I faced and gets you to that smooth developer experience faster.

What's Next: Part 2

In the next part, I'll be adding Shadcn for UI components and setting up proper routing. This will help create a more dynamic and well-structured front-end, so stay tuned for that if you're interested in taking this stack even further.

If you run into any other weird edge cases or just want to share your setup, I'd love to hear about it!

Javascript Vue Nodejs