A Real-World Project: Typescript, Express and React

How I structure my Typescript + Express + React projects

Dirk Hoekstra Follow
Dec 28, 2019 · 4 min read ★

Aquaduct Structure — Photo by Paul-Louis Pröve on Unsplash

In this article, I will show you how I set up and structure my Express — React projects.

Folder structure

When I set up a React Express app I always use the following folder structure.

```
├─app
├─build
├─frontend
```

- The `app` directory will hold the Express backend application.
- The `build` directory will hold the production build of the frontend and backend application
- The `frontend` directory will hold the React frontend application.

Note that you are free to use any other folder structure that you like, this is simply my preferred way of doing things.

Creating the React app

Let's begin with creating the React app. I'm going to use the `create-react-app` npm package for this.

You can run npm packages without installing them using the `npx` tool.

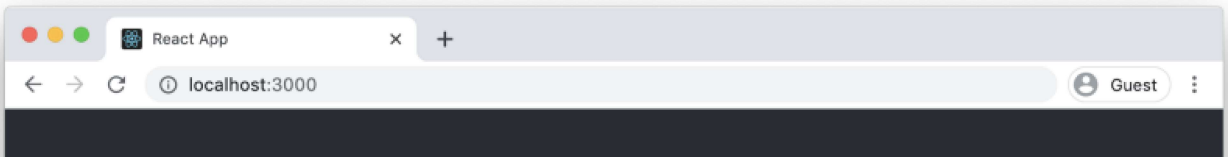
```
npx create-react-app frontend
```

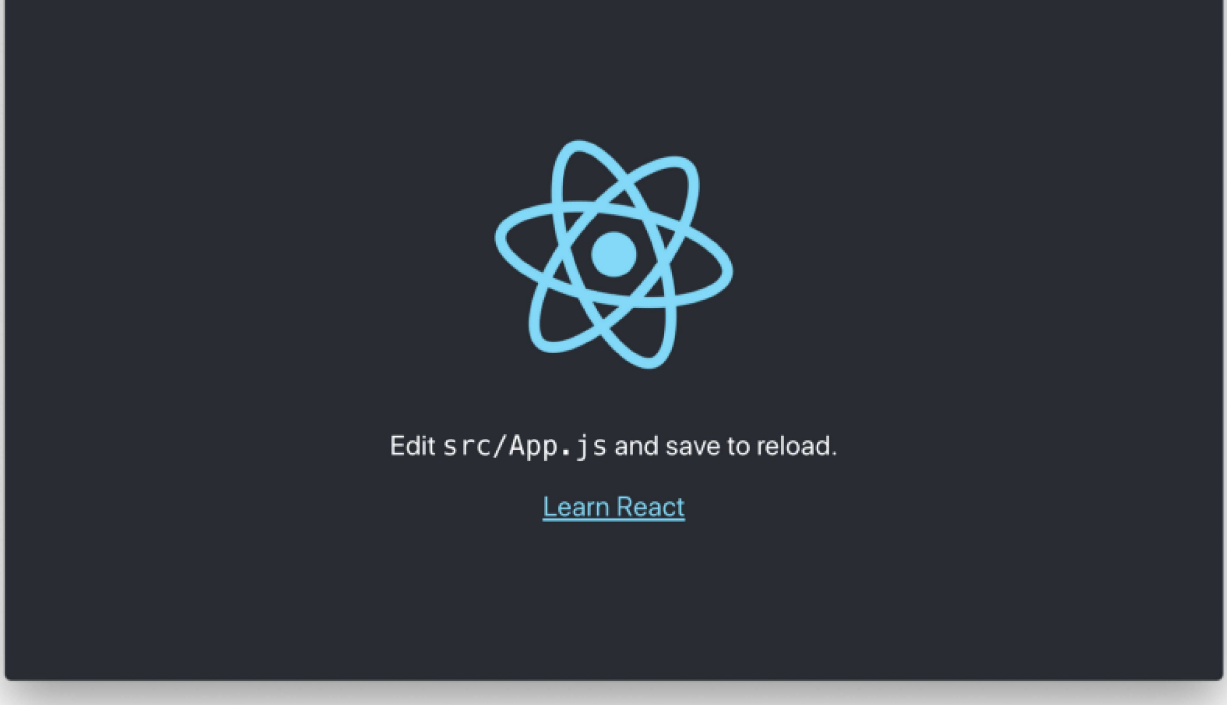
The react app will be created in the `frontend` folder.

Let's test if the installation went correctly.

```
cd frontend
yarn run start
```

The `yarn run start` command will run the React development server. Whenever you make changes to a file it will automatically recompile the react app and reload the browser! 🚀





The `create-react-app` package will also initialize a git repository in the `frontend` directory. However, I want to have a single git repository in the project root directory.

To remove the git repository in the `frontend` directory I simply remove the `.git` directory.

```
rm -rf .git
```

Creating the Express app

We now have a working frontend application, now it's time to set up the backend Typescript Express app.

I start by creating a new package in the project root directory.

Then I add the Express and Typescript dependencies and finally, I create the `app` directory.

```
yarn init
yarn add express @types/express typescript
mkdir app
```

Next, I create a pretty standard `tsconfig.json` file. This file contains the settings for compiling Typescript to Javascript.

I only want to use Typescript in the backend — at least for now. That is why I exclude the `frontend` directory.

In the `app` directory I will create a `Server.ts` that will contain a Server class.

This class will receive the `Express` app in the constructor and initialize the application routes.

In the real world, I would probably create another class `Router.ts` that will hold all the application routes, but that is out of scope for this article.

To initialize this server I create a `index.ts` file in the application root directory. All this does is create a new Server class and start the server.

To start the backend server I add the following snippet to the `package.json` file. It will use the `ts-node` package to directly run Typescript code.

This way you won't have to worry about compiling the Typescript to Javascript as it is all done for you.

That why I can start the server running the following command.

```
yarn run start
```

As a bonus, you can use Nodemon to automatically restart `ts-node` when a file changes.

Building the React app

Let's make a production build of the React app.

I will make a change to the `frontend/package.json` file. Because after building the React application I want to move the build files to the `/build/frontend` folder.

Find the `"scripts"` and update the `"build"` line.

Let's run the `yarn run build` command and see if it works! 🤲

```
mkdir build
cd frontend
yarn run build
```

If you navigate to the `/build/frontend` directory you will see the production-ready React app!

Building the Express app

Let's add a new `"build"` script to the `package.json` file.

This will simply call the Typescript compiler package `tsc` to compile — *or transpile if you prefer* 💁 — the Typescript to Javascript.

Run the build command to test if it works!

```
yarn run build
```

If all went correctly your build directory should look like this.

```
build/
├─app/
├─frontend/
├─index.js
```

Connecting Express and React

We can develop the backend and frontend applications and build them. However, we should also connect Express to React.

For example, if I browse to `localhost:8080/` I should get to see the React application.

If I browse to `localhost:8080/api` I should get to see the API message.

To do this I update the `constructor` of the `Server` class.

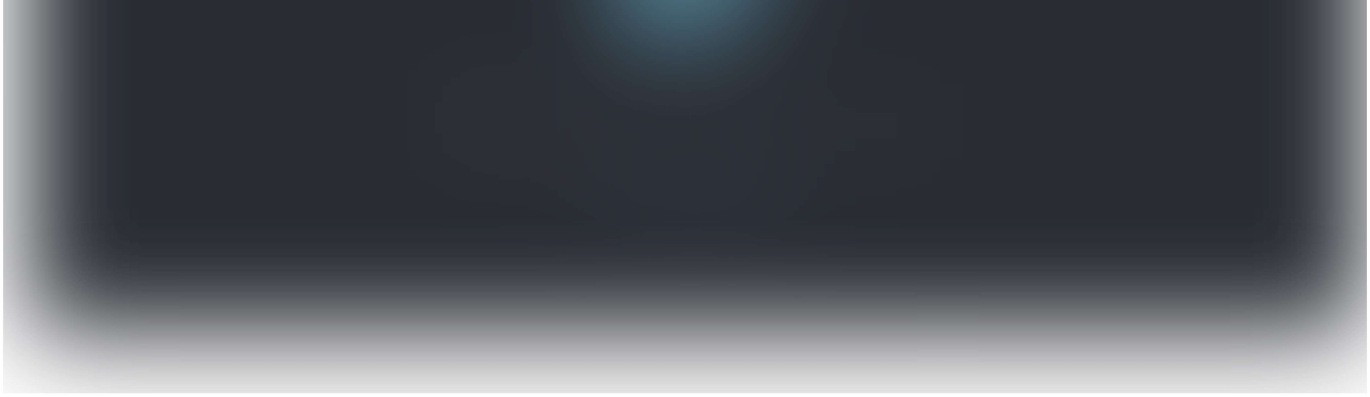
First I mark the `build/frontend` directory as a static asset directory. This means that Express will automatically serve the files in that directory.

Next, I add a wildcard `*` route and send those all to the `index.html` file. This is the file that holds the React frontend application.

Let's rerun the backend application.

```
yarn run start
```

When navigation to `localhost:8080` I get to see the React application 🎉



When navigating to `localhost:8080/api` I get to see the API message 🔥

That's it! You can find the source code here on Github 🚀

JavaScript Front End Development React Web Development