Adding npm scripts

In this lesson, we will finally get to run our project after implementing a couple of npm scripts.

WE'LL COVER THE FOLLOWING

- Adding the scripts
- Running the app in development mode
- Producing a production-ready build
- Wrap up

Adding the scripts

We are going to leverage npm scripts to start our app in development mode and build a production version of our app. Let's add a `scripts` section to `package.json` with the following scripts:

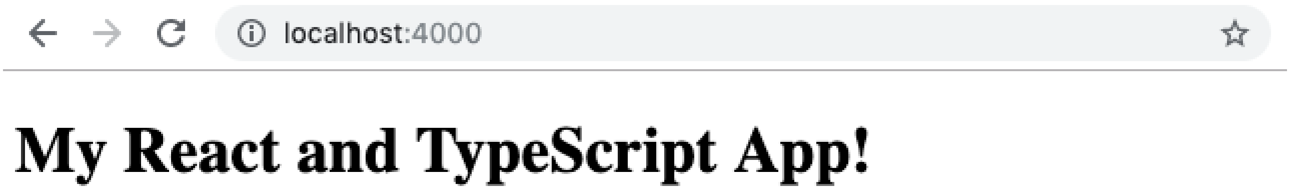
```
...,
"scripts": {
  "start": "webpack-dev-server",
  "build": "webpack --mode production"
},
...
```

Running the app in development mode

Let's run the following command in a terminal to start the app in development mode:

```
npm start
```

After a few seconds, the Webpack development server will start. In our browser, if we browse to `http://localhost:4000/` we'll see our web app:



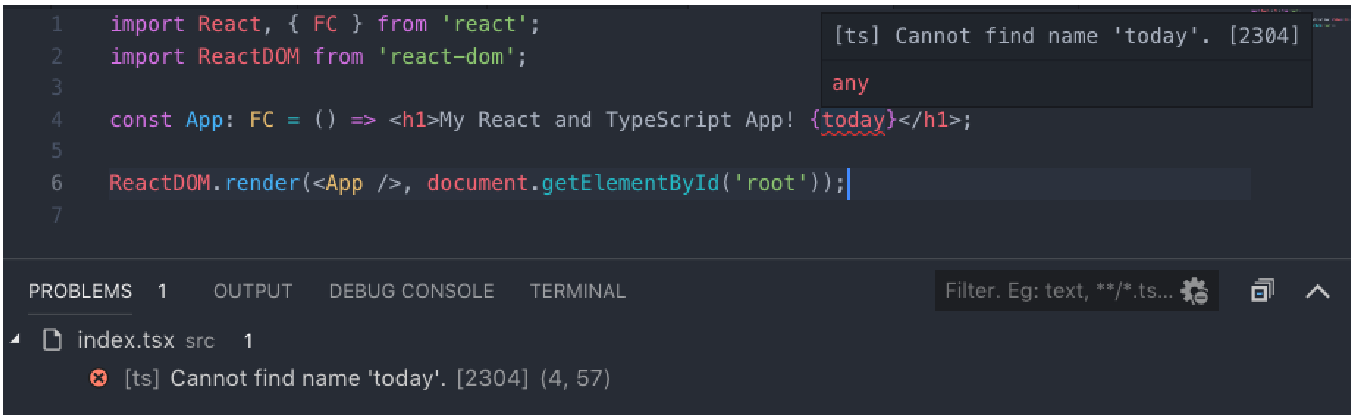
My React and TypeScript App!

We’ll leave the app running.

Let’s make a change to the heading that is rendered in `index.tsx`. Let’s reference a variable called `today` in the heading:

```
...
const App: FC = () => <h1>My React and TypeScript App! {today}</h1>;
...
```

Of course, this is an error because we haven’t declared and initialized `today` anywhere. Our code editor should inform us of this:

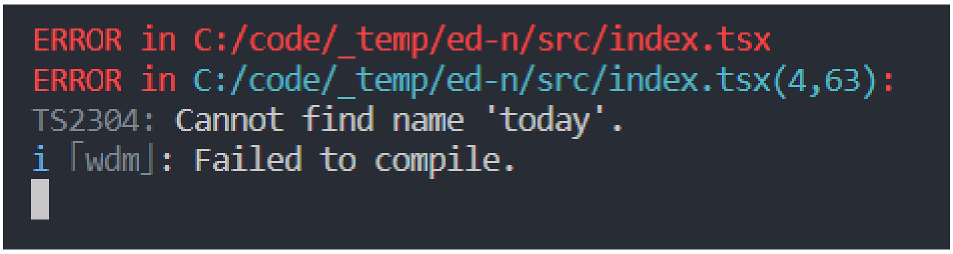


```
import React, { FC } from 'react';
import ReactDOM from 'react-dom';

const App: FC = () => <h1>My React and TypeScript App! {today}</h1>;

ReactDOM.render(<App />, document.getElementById('root'));

```

```
[ts] Cannot find name 'today'. [2304]
any
```

```
[ts] Cannot find name 'today'. [2304] (4, 57)
```

Webpack will also raise this type error in the terminal:

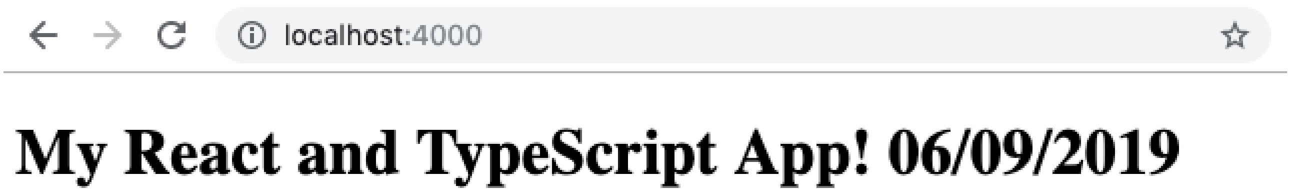


```
ERROR in C:/code/_temp/ed-n/src/index.tsx
ERROR in C:/code/_temp/ed-n/src/index.tsx(4,63):
TS2304: Cannot find name 'today'.
i ｢wdm｣: Failed to compile.
```

Let’s resolve this now by changing the rendered header to reference something that is valid:

```
const App: FC = () => <h1>My React and TypeScript App! {new Date().toLocal
eDateString()}</h1>;
```

The type errors will vanish, and the running app will have been updated to include today's date:



My React and TypeScript App! 06/09/2019

Producing a production-ready build

Let's move on now to try the final npm script. This produces a production build of our app. Let's run the following command in the terminal:

```
npm run build
```

Where can we find the bundled JavaScript file in our project?

Show Answer

Will any type checking occur in this build process?

Show Answer

Wrap up

That's it! We've managed to create a project by manually installing and configuring all the dependencies. Our project doesn't include all the dependencies that Create React App does, but we have gained a good appreciation for how React with TypeScript projects can work with tools like Babel, Webpack, and ESLint.

Next up is a quiz to check what we have learned in this chapter.