Adding React and TypeScript

Continuing from the last lesson, we are going to add both React and TypeScript to our project in this lesson. We'll also create the root React component as well.

WE'LL COVER THE FOLLOWING

Adding React

We can install React into our project by running the following command in the terminal:

```
npm install react react-dom
```

This installs the core React library as well the React library for interacting with the browser.

Adding TypeScript

TypeScript can be installed in our project as a development dependency by running the following command in the terminal:

```
npm install --save-dev typescript
```

Why do you think we have installed TypeScript as a development dependency rather than a full dependency that is available at runtime?

Show Answer

TypeScript is configured with a file called `tsconfig.json`. Let's create this file in the root of our project with the following content:

```
{
  "compilerOptions": {
    "lib": [
      "dom",
      "dom.iterable",
      "esnext"
    ],
    "allowJs": true,
    "allowSyntheticDefaultImports": true,
    "skipLibCheck": true,
    "esModuleInterop": true,
    "strict": true,
    "forceConsistentCasingInFileNames": true,
    "moduleResolution": "node",
    "resolveJsonModule": true,
    "isolatedModules": true,
    "noEmit": true,
    "jsx": "react"
  },
  "include": [
    "src"
  ]
}
```

We are only going to use TypeScript in our project for type checking. We are going to use a tool called Babel later in this lesson to do the code transpilation. So, the compiler options in our `tsconfig.json` are focused on type checking, not code transpilation.

Here's an explanation of the settings we have used:

- `lib`: The standard typing to be included in the type checking process. In our case, we have chosen to use the types for the browsers DOM as well as the latest version of ECMAScript.
- `allowJs`: Whether to allow JavaScript files to be compiled.
- `allowSyntheticDefaultImports`: This allows default imports from modules with no default export in the type checking process.

- `skipLibCheck`: Whether to skip type checking of all the type declaration files (`*.d.ts`).
- `esModuleInterop`: This enables compatibility with Babel.
- `strict`: This sets the level of type checking to very high. When this is true, the project is said to be running in *strict mode*.
- `forceConsistentCasingInFileNames`: Ensures that the casing of referenced file names is consistent during the type checking process.
- `moduleResolution`: How module dependencies get resolved which is `node` for our project.
- `resolveJsonModule`: This allows modules to be in `.json` files which are useful for configuration files.
- `noEmit`: Whether to suppress TypeScript generating code during the compilation process. This is `true` in our project because Babel will be generating the JavaScript code.
- `jsx`: Whether to support JSX in `.tsx` files.
- `include`: These are the files and folders for TypeScript to check. In our project, we have specified all the files in the `src` folder.

More information on all the compiler options can be found here.

Let's also add the React types into our project as a development dependency by running the following npm command in the terminal:

```
npm install --save-dev @types/react @types/react-dom
```

Adding a root React component

We are going to create the root React component now and the code to inject it into `index.html`. Let's add an `index` file into the `src` folder. Can you remember what the file extension for TypeScript files are that contain JSX?

Show Answer

Let's add the following content into the file:

```
import React from 'react';
```

```
import ReactDOM from 'react-dom';


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

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

Our root component is a function component called `App,` which contains a heading. We use the `render` function from React to inject this into the `div` in `index.html`.

Wrap up

The project is coming along nicely.

In the next lesson, we'll continue by adding a tool that will transpile both the React and TypeScript code.