Software Studio
軟體設計與實驗

Create React App with TypeScript

Hung-Kuo Chu
Department of Computer Science
National Tsing Hua University

CS2410

Using TypeScript

- To use TypeScript in your web app, you have the following options:
  - Install the TypeScript compiler and manually compile your JavaScript code.
  - Use a framework that has TypeScript out of the box. (Most common!)
    - Create React App, Next.js, Gatsby, Vue.js, etc.
- We will demonstrate the **Create React App** usage, as it is the most convenient.

Create React App

- A convenient npm package that manages the long list of dependencies needed for developing a modern React application.
  - React.js, Webpack, Babel, HMR, etc.
- All you need is just one line!

```
npx create-react-app [app-name]
```

- Your actual project will be under the [app-name] directory.

What's Create React App?

Create React App with TS

- The default setup for Create React App is for a JavaScript application.
- We can start with a setup that has TypeScript out of the box by using the TypeScript template.

```
npx create-react-app [app-name] --template typescript
```

Create React App with TS

- For an existing Create React App project, you can install the TypeScript dependencies explicitly instead.

```
npm install --save typescript @types/node
@types/react @types/react-dom @types/jest
```

Compiling TypeScript

- The TypeScript compiler will be used automatically to compile your TypeScript code when you run **build** or **start**.
- The **build** script (npm run build) will build your project and put the files in the **public** folder.
- The **start** script (npm start) will build and then start a localhost server with HMR.

Edit `src/App.tsx` and save to reload.

Learn React

Let’s add a component!

```
// ./MyButton.tsx
import React, { MouseEventHandler } from "react";
interface MyButtonProps{
    onClick?: MouseEventHandler;
    children: string;
}
export default function MyButton(props?:
MyButtonProps){
    if(!props) return (<button/>);
    else return (
        <button
onClick={props.onClick}>{props.children}</button>
    );
}
```

Let’s add a component!

my-app > src > TS App.tsx > ...

```tsx
import React from 'react';
import logo from './logo.svg';
import './App.css';
import MyButton from './MyButton';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.tsx</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
        <MyButton onClick={() => alert("Hello!")}>Click me!</MyButton>
      </header>
    </div>
  );
}

export default App;
```

Import the component…

And put it here.

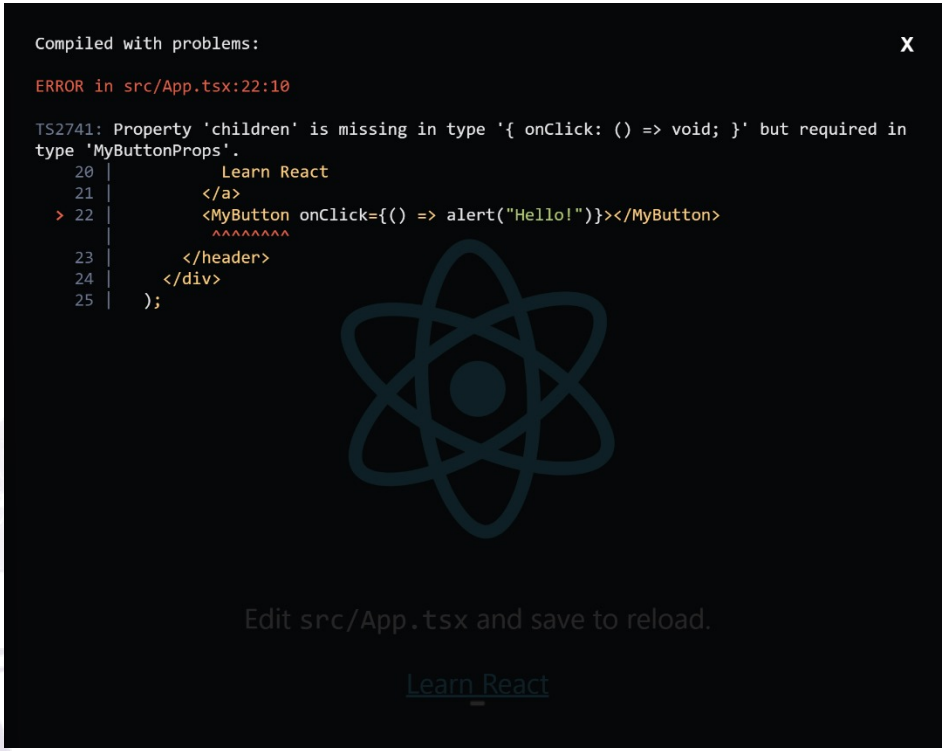
Type Checking

- We can use TypeScript to enforce **rules** on how we want our component to be used.
- Because our button looks weird when it has no string inside, we can require users to always put something inside by changing the props' **interface**.

```
interface MyButtonProps{
    onClick?: MouseEventHandler; // A callback onClick is optional.
    children: string; // You must put a string inside the button!
}
```

Compile Error

- You will see compile error now if you remove the inner text from the button. View it in the command line or on your localhost.



Compile Error

- TypeScript will catch incorrect types as well!

```
Compiled with problems:                                                    X

ERROR in src/App.tsx:22:51

TS2322: Type 'Element' is not assignable to type 'string'.
    20 |             Learn React
    21 |           </a>
  > 22 |           <MyButton onClick={() => alert("Hello!")}><div>Hello!</div></MyButton>
       |                                                     ^^^^^^^^^^^^^^^^^^
    23 |         </header>
    24 |       </div>
    25 |     );
```

Compile Error

- Notice that the webpage still “works” even with compile error.
- A lot of times, TypeScript compiler can generate working code **even with compile error.**
- But since you’re using TypeScript, you should not ignore any compile error even if it “works”!

Deploying

- When deploying to a hosting service (e.g., Firebase Hosting), you need to **build** your project first, and then **deploy**.
- You can add a script in **package.json** to automate the process.

Automating Deployment

- When you execute “**npm run deploy**”, Node.js will always run “**predeploy**” for you first!

```
"scripts": {
    "predeploy": "npm run build",
    "deploy": “firebase deploy",
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
```

Configuring TS Compiler

- You can modify the settings used by the TypeScript compiler when compiling your source files by editing **tsconfig.json**.
- Refer to https://www.typescriptlang.org/docs/handbook/compiler-options.html for a full list of options.


thank you!
Question?