React with TypeScript

Goal

In this lab, you will covert an existing React application to use TypeScript.

Your mission

This is lab you will convert a simple working React application created with Create-React-App and written in JavaScript to TypeScript.

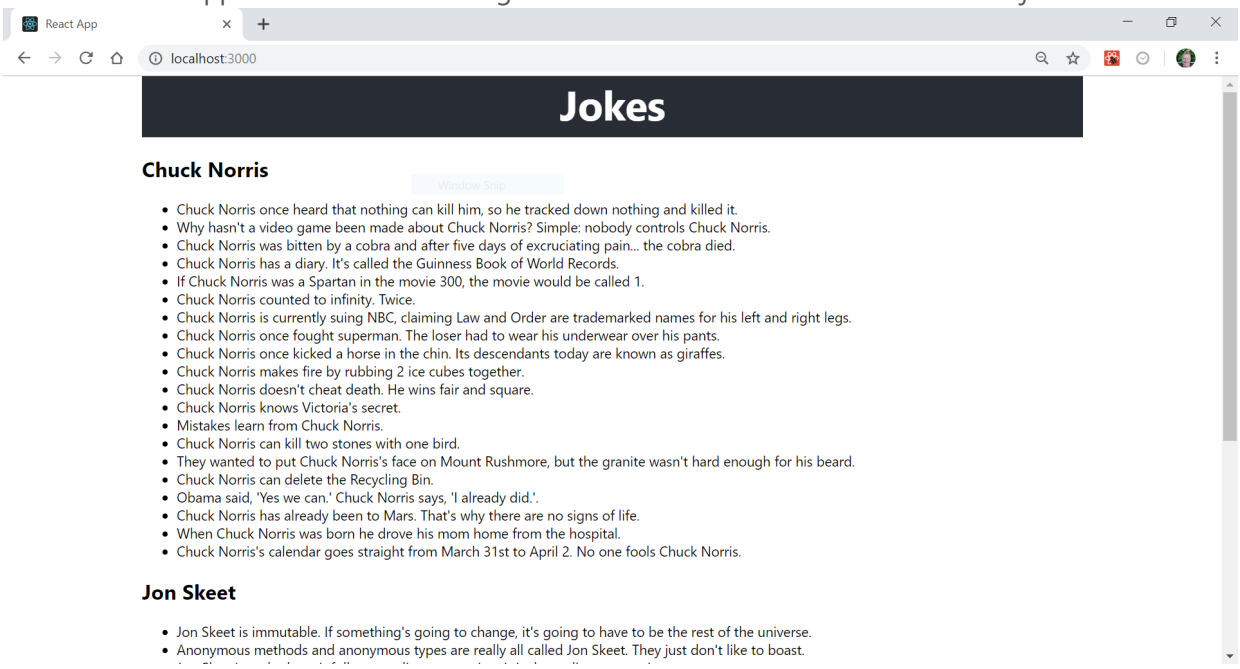
Type it out by hand?

> Typing it drills it into your brain much better than simply copying and pasting it. You're forming new neuron pathways. Those pathways are going to help you in the future. Help them out now.

Add TypeScript support

In this section you will add TypeScript support to an existing application build with Create React App.

1. Open the **begin** folder.
2. Run `npm install` to install all required NPM packages and `npm start` to start the application. You should see the application below showing a number of Chuck Norris and Jon Skeet jokes.



3. Stop the application from running. Install the TypeScript compiler and the type definitions for React, Node and Jest. You can do that with the following command: `npm install --save typescript @types/node @types/react @types/react-dom @types/jest`
4. Rename `jokes.js` in the `components` folder to `jokes.tsx`.
5. Restart the application using `npn start`. Note that a `tsconfig.json` is created automatically with

a number of default settings.

6. Create a file name `types.ts` in the `components` folder and add the type definition of Joke to it. The type should have an `id` property of type `number` and a `joke` property of type `string`. Make sure to export this type.

7. Open `jokes.tsx` and add a type definition `JokesProps` with a `jokes` property of type `Joke[]`. Update the `Joke` component to receive a `props` of type `JokesProps`.

8. Rename `jon-skeet.js` to `jon-skeet.tsx` and restart application using the `npm start` command to solve the error.

9. Open `jon-skeet.tsx` and notice that `jokes` variable is of type `never[]` and `setJokes()` function expects a type of `never[]` as input argument. Import the `Joke` type from `types.ts` and use the generic argument of `useState()` to define the `jokes` variable and `setJokes()` function as `Joke[]`.

10. Rename `App.js` to `App.tsx` and restart the application with the `npm start` command. Notice the code which is partially TypeScript and partially JavaScript works just fine.

11. Rename `chuck-norris.js` to `chuck-norris.tsx` and restart the application with the `npm start` command. Add an empty type `ChuckNorrisProps` to represent the incoming props. Add the type `ChuckNorrisState` with a `jokes` property of type `Joke[]`. Update the `ChuckNorris` class based component by adding the two types to the generic `Component` class.

Bonus Exercise: Convert the rest of the application to TypeScript

Part of the application is now written in TypeScript and part in JavaScript.

1. Rename all remaining `.js` files to `.ts` or `.tsx` depending on whether they contain JSX code.

2. The file `serviceWorker.ts` will initially contain a number of compilation errors. Add a `Config` type with the required properties and use it to fix the compilation errors for the `config` parameter. The `swUrl` parameter can be defined as type `string`.

Solution

The Solution can be found in **complete** folder