Adding Babel

We will continue to create our project in this lesson by adding a dependency that will transpile both the React and TypeScript code to JavaScript.

Installing Babel

As mentioned in the last lesson, our project is going to use **Babel** to convert our TypeScript code to JavaScript. Let's install Babel with the necessary plugins as a development dependency:

```
npm install --save-dev @babel/core @babel/preset-env @babel/preset-react @
babel/preset-typescript @babel/plugin-transform-runtime @babel/runtime
```

Here's an explanation of the packages we have just installed:

- `@babel/core`: As the name suggests, this is the core Babel library.
- `@babel/preset-env`: This is a collection of plugins that allow us to use the latest JavaScript features but still target browsers that don't support them.
- `@babel/preset-react`: This is a collection of plugins that enable Babel to transform React code into JavaScript.
- `@babel/preset-typescript`: This is a collection of plugins that enable Babel to transform TypeScript code into JavaScript.
- `@babel/plugin-transform-runtime` and `@babel/runtime`: These are plugins that allow us to use the `async` and `await` JavaScript features.

Configuring Babel

Babel is configured in a file called `.babelrc`. Let’s create this file in the root of our project with the following content:

```
{
  "presets": [
    "@babel/preset-env",
    "@babel/preset-react",
    "@babel/preset-typescript"
  ],
  "plugins": [
    [
      "@babel/plugin-transform-runtime",
      {
        "regenerator": true
      }
    ]
  ]
}
```

This configuration tells Babel to use the plugins we have installed.

Wrap up

We are now able to transpile the code in our project to JavaScript but there is still plenty to do. We’ll continue with adding linting and code autoformatting in the next lesson.