Adding code linting and autoformatting

We will continue to create our React and TypeScript project in this lesson by adding linting and autoformatting.

WE'LL COVER THE FOLLOWING

- Adding linting
- Adding autoformatting
- Wrap up

Adding linting

We touched on linting with **ESLint** when we created a project using Create React App. We are going to use ESLint in this project as well. So, let's install ESLint along with the plugins we are going to need as development dependencies:

```
npm install --save-dev eslint eslint-plugin-react eslint-plugin-react-hooks @typescript-eslint/parser @typescript-eslint/eslint-plugin
```

Below is an explanation of the packages that we just installed:

- `eslint`: This is the core ESLint library.
- `eslint-plugin-react`: This contains some standard linting rules for React code.
- `eslint-plugin-react-hooks`: This contains some linting rules for React hooks code.
- `@typescript-eslint/parser`: This allows TypeScript code to be linted.
- `@typescript-eslint/eslint-plugin`: This contains some standard linting rules for TypeScript code.

Can you remember what file we use to configure the linting rules?

Show Answer

Let's create the configuration file containing the following:

```
{
  "parser": "@typescript-eslint/parser",
  "parserOptions": {
    "ecmaVersion": 2018,
    "sourceType": "module"
  },
  "plugins": [
    "@typescript-eslint",
    "react-hooks"
  ],
  "extends": [
    "plugin:react/recommended",
    "plugin:@typescript-eslint/recommended"
  ],
  "rules": {
    "react-hooks/rules-of-hooks": "error",
    "react-hooks/exhaustive-deps": "warn",
    "react/prop-types": "off"
  },
  "settings": {
    "react": {
      "pragma": "React",
      "version": "detect"
    }
  }
}
```

So, we have configured ESLint to use the TypeScript parser, and the standard React and TypeScript rules as a base set of rules. We've explicitly added the two React hooks rules and suppressed the `react/prop-types` rule because prop types aren't relevant in React with TypeScript projects. We have also told ESLint to detect the version of React we are using.

If we are using Visual Studio Code as our editor, hopefully, we still have the ESLint extension installed from the last lesson. We also need to tell VS Code to lint TypeScript code. Can you remember how to do this?

Show Answer

Adding autoformatting

We touched on autoformatting code with **Prettier** when we created a project using Create React App. Let's install Prettier into this project with the plugins that allow it to play nice with ESLint:

```
npm install --save-dev prettier eslint-config-prettier eslint-plugin-prettier
```

Let's configure Prettier to autoformat our code with the following rules:

- Semicolons should be automatically added at the end of statements.
- Single quotes should be used around strings.
- Lines should wrap at 100 characters.
- Indentation level will be two spaces.
- A trailing comma should be added to multi-line arrays and objects.

Show Answer

We also need to add the following to the `.eslintrc.json` file for Prettier to play nice with ESLint:

```
{
  ...,
  "plugins": [
    ...,
    "prettier"
  ]
  "extends": [
    ...,
    "prettier"
  ],
  "rules": {
    ...,
    "prettier/prettier": "error"
  }
}
```

The last task is to get VS Code to automatically format code in a file when we save it by adding the `editor.formatOnSave` flag to our settings.json file:

```
{
  ...,
  "editor.formatOnSave": true
}
```

Wrap up

The code in the project is now being formatted and linted automatically.

We will continue setting up the project in the next lesson by using a tool to bundle all the JavaScript together into a single file.