Create React App comes with a bunch of tools that improve the editing experience - if configured correctly. Here's a few tips to maximize your productivity:

Syntax highlighting

To configure the syntax highlighting in your favorite text editor, head to the relevant Babel documentation page and follow the instructions. Some of the most popular editors are covered.

Displaying Lint Output in the Editor

> *Note: this feature is available with* `react-scripts@0.2.0` *and higher.*

> *It works out of the box for newly created projects with* `react-scripts@2.0.3` *and higher.*

> *It also only works with npm 3 or higher.*

Some editors, including Sublime Text, Atom, and Visual Studio Code, provide plugins for ESLint.

They are not required for linting. You should see the linter output right in your terminal as well as the browser console. If you prefer the lint results to appear right in your editor, please make sure you install an ESLint plugin/extension.

Note that even if you customise your ESLint config, these changes will **only affect the editor integration**. They won't affect the terminal and in-browser lint output. This is because Create React App intentionally provides a minimal set of rules that find common mistakes.

If you want to enforce a coding style for your project, consider using Prettier instead of ESLint style rules.

Extending or replacing the default ESLint config

You can extend our base ESLint config, or replace it completely if you need.

There are a few things to remember:

1. We highly recommend extending the base config, as removing it could introduce hard-to-find issues.
2. When working with TypeScript, you'll need to provide an `overrides` object for rules that should *only* target TypeScript files.
3. It's important to note that any rules that are set to `"error"` will stop the project from building.

In the below example:

- the base config has been extended by a shared ESLint config,
- a new rule has been set that applies to all JavaScript and TypeScript files, and
- a new rule has been set that only targets TypeScript files.

```
{
  "eslintConfig": {
    "extends": ["react-app", "shared-config"],
    "rules": {
      "additional-rule": "warn"
    },
    "overrides": [
      {
        "files": ["**/*.ts?(x)"],
```

```
        "rules": {
          "additional-typescript-only-rule": "warn"
        }
      }
    ]
  }
}
```

Debugging in the Editor

This feature is currently only supported by Visual Studio Code and WebStorm.

Visual Studio Code and WebStorm support debugging out of the box with Create React App. This enables you as a developer to write and debug your React code without leaving the editor, and most importantly it enables you to have a continuous development workflow, where context switching is minimal, as you don't have to switch between tools.

Visual Studio Code

You need to have the latest version of VS Code and VS Code Chrome Debugger Extension installed.

Then add the block below to your `launch.json` file and put it inside the `.vscode` folder in your app's root directory.

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Chrome",
      "type": "chrome",
      "request": "launch",
      "url": "http://localhost:3000",
      "webRoot": "${workspaceFolder}/src",
      "sourceMapPathOverrides": {
        "webpack:///src/*": "${webRoot}/*"
      }
    }
  ]
}
```

> *Note: the URL may be different if you've made adjustments via the HOST or PORT environment variables.*

Start your app by running `npm start`, and start debugging in VS Code by pressing `F5` or by clicking the green debug icon. You can now write code, set breakpoints, make changes to the code, and debug your newly modified code—all from your editor.

Having problems with VS Code Debugging? Please see their troubleshooting guide.

WebStorm

You need to have WebStorm and JetBrains IDE Support Chrome extension installed.

In the WebStorm menu `Run` select `Edit Configurations...`. Then click `+` and select `JavaScript Debug`. Paste `http://localhost:3000` into the URL field and save the configuration.

> *Note: the URL may be different if you've made adjustments via the HOST or PORT environment variables.*

Start your app by running `npm start`, then press `^D` on macOS or `F9` on Windows and Linux or click the green debug icon to start debugging in WebStorm.

The same way you can debug your application in IntelliJ IDEA Ultimate, PhpStorm, PyCharm Pro, and RubyMine.

Formatting Code Automatically

Prettier is an opinionated code formatter with support for JavaScript, CSS and JSON. With Prettier you can format the code you write automatically to ensure a code style within your project. See Prettier's GitHub page for more information, and look at this page to see it in action.

To format our code whenever we make a commit in git, we need to install the following dependencies:

```
npm install --save husky lint-staged prettier
```

Alternatively you may use `yarn`:

```
yarn add husky lint-staged prettier
```

- `husky` makes it possible to use githooks as if they are npm scripts.
- `lint-staged` allows us to run scripts on staged files in git. See this blog post about lint-staged to learn more about it.
- `prettier` is the JavaScript formatter we will run before commits.

Now we can make sure every file is formatted correctly by adding a few lines to the `package.json` in the project root.

Add the following field to the `package.json` section:

```
+  "husky": {
+    "hooks": {
+      "pre-commit": "lint-staged"
+    }
+  }
```

Next we add a 'lint-staged' field to the `package.json`, for example:

```
  "dependencies": {
    // ...
  },
+ "lint-staged": {
+   "src/**/*.{js,jsx,ts,tsx,json,css,scss,md}": [
+     "prettier --write"
+   ]
+ },
  "scripts": {
```

Now, whenever you make a commit, Prettier will format the changed files automatically. You can also run `./node_modules/.bin/prettier --write "src/**/*.{js,jsx,ts,tsx,json,css,scss,md}"` to format your entire project for the first time.

Next you might want to integrate Prettier in your favorite editor. Read the section on Editor Integration on the Prettier GitHub page.