{JS} MASTERY

ESLint, Prettier, & VSCode Guide for Next & React

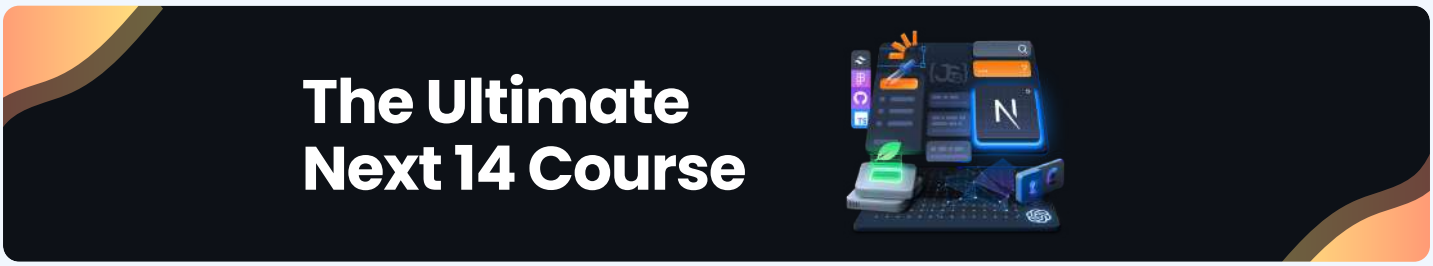
Created by **JS Mastery**

Visit *jsmastery.pro* for more

...before you go

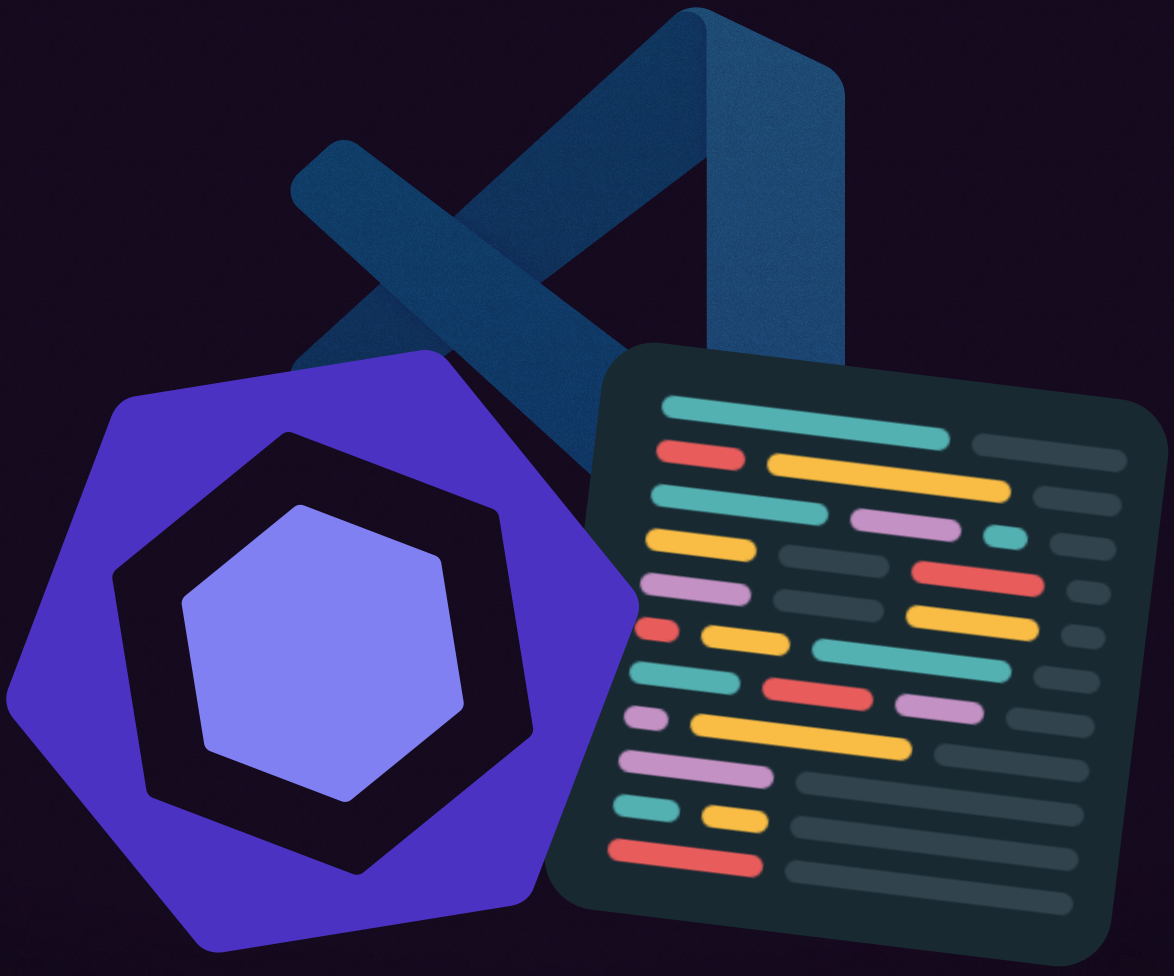
While our ESLint, Prettier guide is fantastic for mastering linting and taking the initial steps into code quality, imagine how much cooler it would be to apply that knowledge to the latest tech stack like Next.js & leveraging ESLint, and witness the remarkable impact it can have on code consistency and cleanliness in your projects.

If you're eager to dive deep into something this specific and build substantial projects, our special course on Next.js has got you covered.



It teaches everything from the ground up, providing a hands-on experience that goes beyond just Docker.

Check it out and take your skills to the next level 🚀



Introduction

Introduction

Introduction to Linting, ESLint, and Prettier

Linting is the process of analyzing code for potential errors, bugs, and stylistic issues. It helps maintain a consistent code style, improves code quality, and catches common mistakes early in development. Two popular tools for linting modern web development code are ESLint and Prettier.

What is Prettier?

Prettier is a code formatter that focuses on code formatting and style consistency. It automatically formats your code according to predefined rules, making it more readable and eliminating debates over code style.

A code formatter is primarily concerned with the visual appearance and consistent code formatting. It automatically analyzes the structure of the code and applies predefined formatting rules to ensure a uniform style throughout the codebase.

Code formatters focus on aspects such as indentation, line breaks, spacing, & the placement of braces & parentheses

Introduction

The goal of a code formatter like Prettier is to standardize the appearance of code, making it more readable and enhancing code consistency within a project or team.

What is ESLint?

ESLint is a linter that allows you to define custom linting rules and enforce coding standards. It helps identify and fix potential errors, enforces consistent coding styles, and promotes best practices in your codebase.

A code linter goes beyond code formatting and focuses on the correctness, quality, and potential issues within the code. Linters analyze the code for potential errors, bugs, stylistic inconsistencies, and violations of coding standards and best practices.

They check for issues such as unused variables, missing semicolons, incorrect function usage, possible memory leaks, and adherence to coding conventions.

Introduction

Code linters like ESLint provide warnings, errors, or suggestions to developers about problematic code areas, helping them identify and fix potential issues early in the development process. By enforcing consistent coding styles and detecting errors, linters improve code quality and maintainability and reduce the likelihood of bugs.

Best of Both Worlds

Combining ESLint and Prettier in your development workflow can greatly improve code quality, maintainability, and collaboration within a team. ESLint ensures your code adheres to specific rules and catches potential issues, while Prettier enforces consistent code formatting.

With ESLint and Prettier configured in your development environment, you can focus on writing clean, high-quality code while enjoying automated code formatting and helpful linting feedback.

Let's get started!

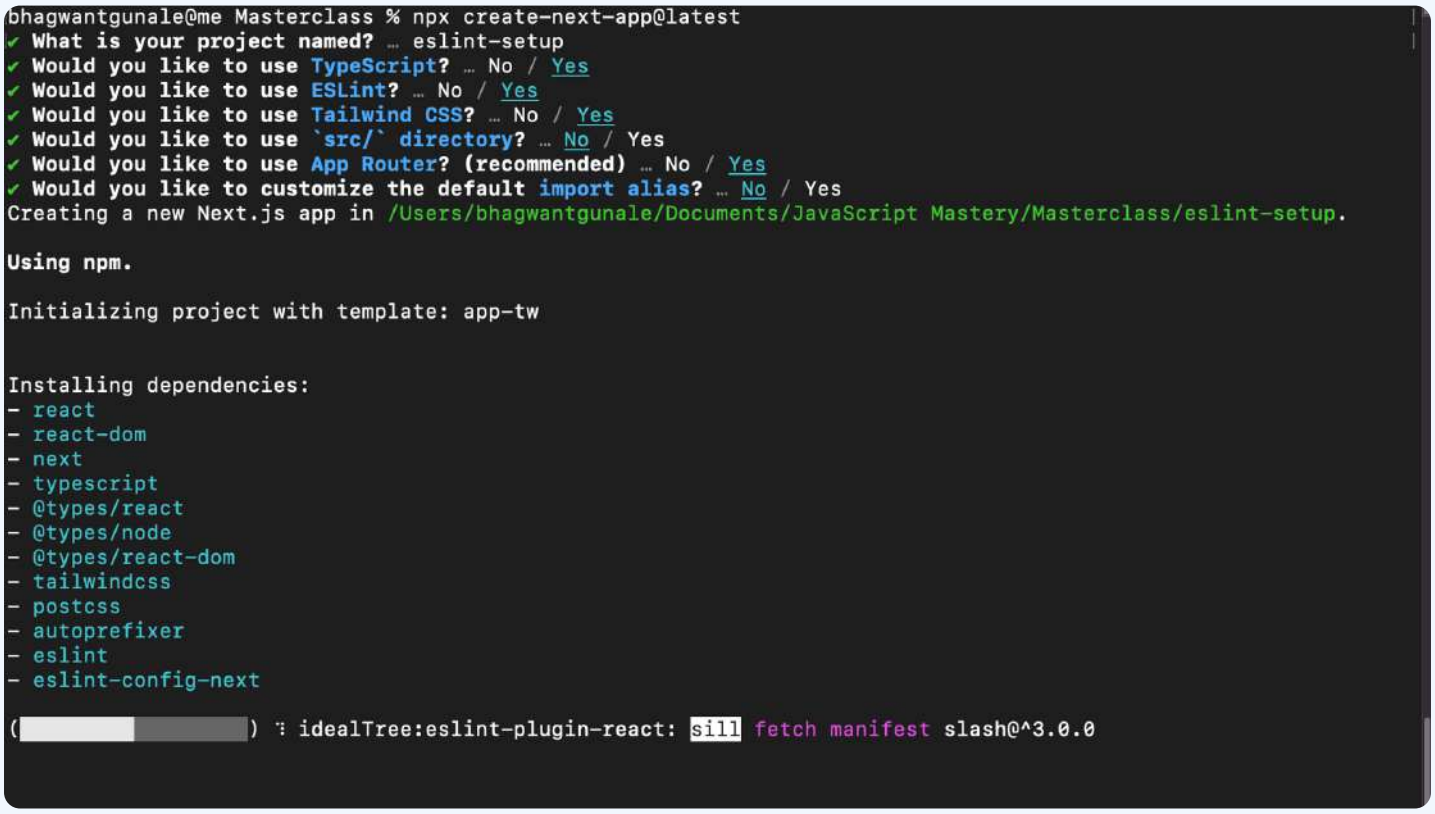
Next.js Setup

Next.js 14 Setup

Create Next.js Project using,

```
npx create-next-app@latest
```

Select yes when asked about adding ESLint to your project:



Once this is done, open up the project in VSCode.

Add an ESLint Config

If you look at the `.eslintrc.json` file at the root of your new Next.js project, you'll see that there is already an ESLint configuration called ***`next/core-web-vitals`*** and standard. It's important to keep it. It will warn you when you write code in your Next.js project that affects core web vitals.

We still need to add another ESLint configuration for consistent code styling, as the default Next configuration doesn't do much to help you improve the quality of your codebase.

Let's install the standard configuration. Here are its **rules**.

> *Protip: learning the rules will allow you to improve your code quality immediately.*

Now, run:

```
npm install eslint-config-standard
```

Add an ESLint Config

This will install the standard ESLint configuration in your project. Then, add it to your `.eslintrc.json` file:

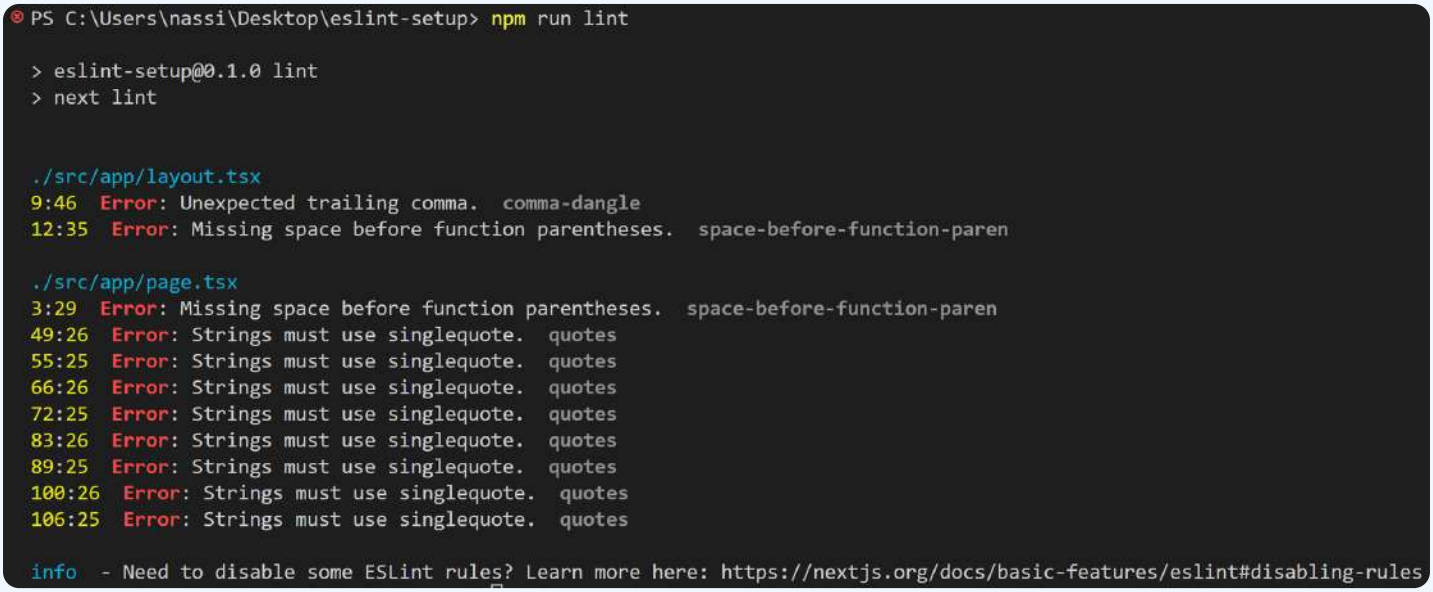
```
{
  "extends": ["next/core-web-vitals", "standard"]
}
```

Testing ESLint in the Terminal

Now that we have added a new ESLint configuration let's run it with

```
npm run lint
```

to ensure it's working. Later on, I'll show you how to get it to work in VSCode automatically as soon as you save a file.



Here you see a bunch of errors. Exactly what we needed 😅.

For example, the standard code style wants us to use single quotes rather than double quotes for strings.

Adding an ESLint Configuration for TailwindCSS

Having our class names organized logically in Tailwind makes the code easier to read and review. We can install another ESLint config that takes care of that to do this automatically.

```
npm install eslint-plugin-tailwindcss
```

and add it to your **.eslintrc.json** file like this:

```
{
  "extends": ["next/core-web-vitals", "standard",
              "plugin:tailwindcss/recommended"]
}
```

Prettier VS ESLint potential conflicts

ESLint sometimes likes to enter into conflict with Prettier. 🥊

To avoid these situations, you should run:

```
npm install eslint-config-prettier
```

This will install a package that removes all ESLint rules that could conflict with Prettier. Once installed, add “prettier” to your **.eslintrc.json** file.

```
{
  "extends": ["next/core-web-vitals", "standard",
      "plugin:tailwindcss/recommended", "prettier"]
}
```

Install Prettier

Don't forget to install prettier before proceeding; otherwise, things will not work, and you won't know why. 🤦

Run:

```
npm install prettier
```

Setting things up for VSCode

Now that everything is set up command-line-wise, it's time to integrate ESLint and Prettier with VSCode. For that, we will need to modify the **`settings.json`** file or create a config specific to our project that you can share with others if you'd like. We'll go with the latter.

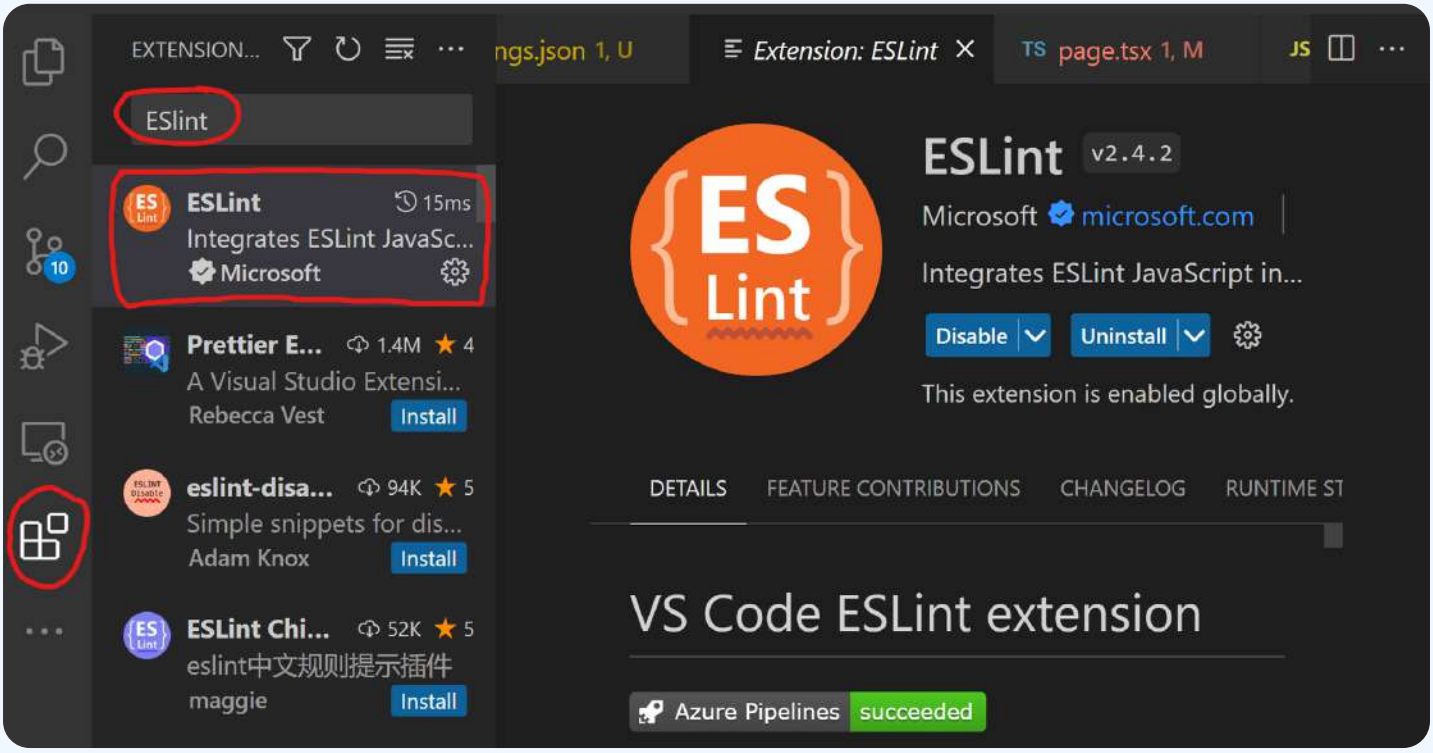
In your Next.js project at the root, create a **`.vscode`** folder and within it a **`settings.json`** file with the following code:

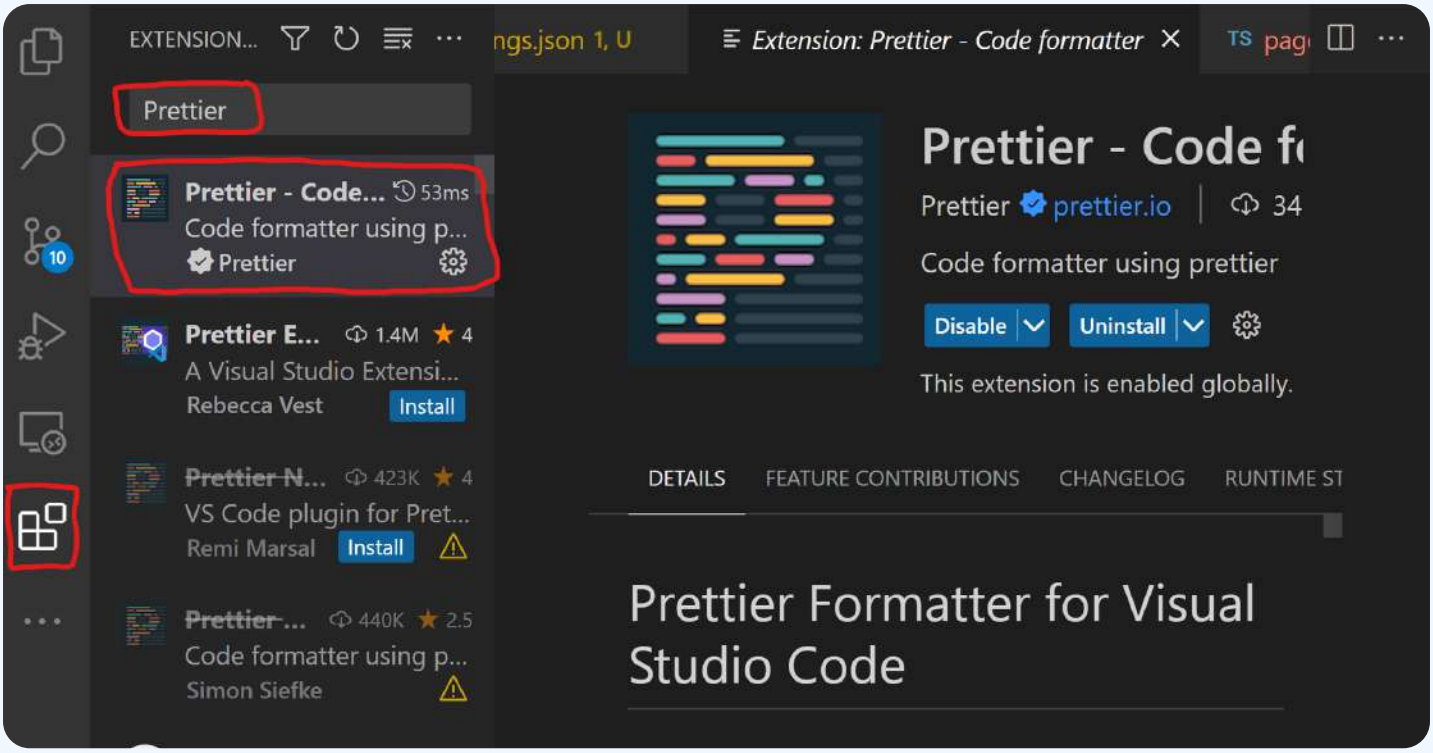
Install Prettier

```
{
  "editor.defaultFormatter": "esbenp.prettier-vscode",
  "editor.formatOnSave": true,
  "editor.codeActionsOnSave": {
    "source.fixAll.eslint": true,
    "source.addMissingImports": true
  },
  "prettier.tabWidth": 2,
  "prettier.useTabs": false,
  "prettier.semi": true,
  "prettier.singleQuote": false,
  "prettier.jsxSingleQuote": false,
  "prettier.trailingComma": "es5",
  "prettier.arrowParens": "always",
  "[typescriptreact]": {
    "editor.defaultFormatter": "esbenp.prettier-vscode"
  }
}
```

This sets up ESlint and Prettier to work within VSCode. Every time you hit save, both Prettier and ESLint will run. But to test it in action, there's one final thing we have to do...

Install ESLint and Prettier from the Extensions Marketplace



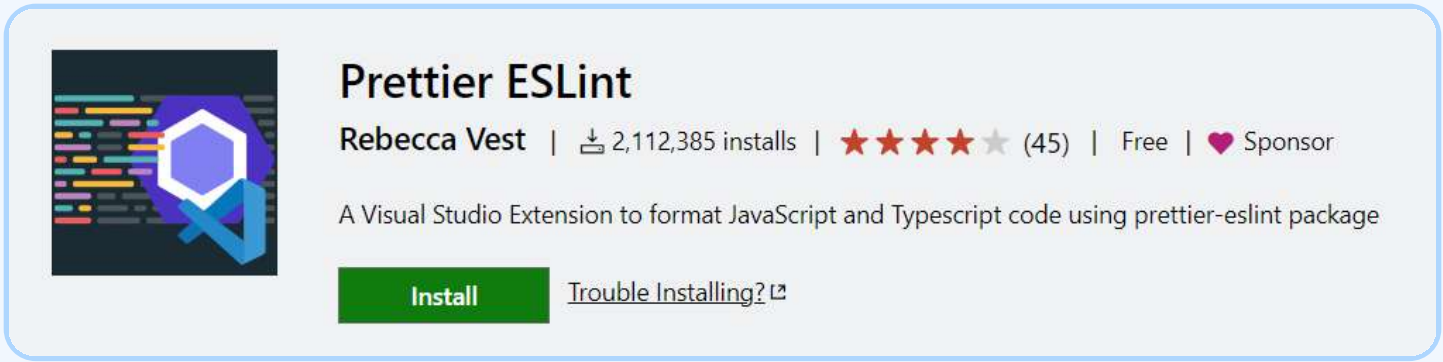


Install Prettier

Once both are installed, restart your Visual Studio Code. Now try to edit and save some files. You should see that Prettier and ESLint are now working.

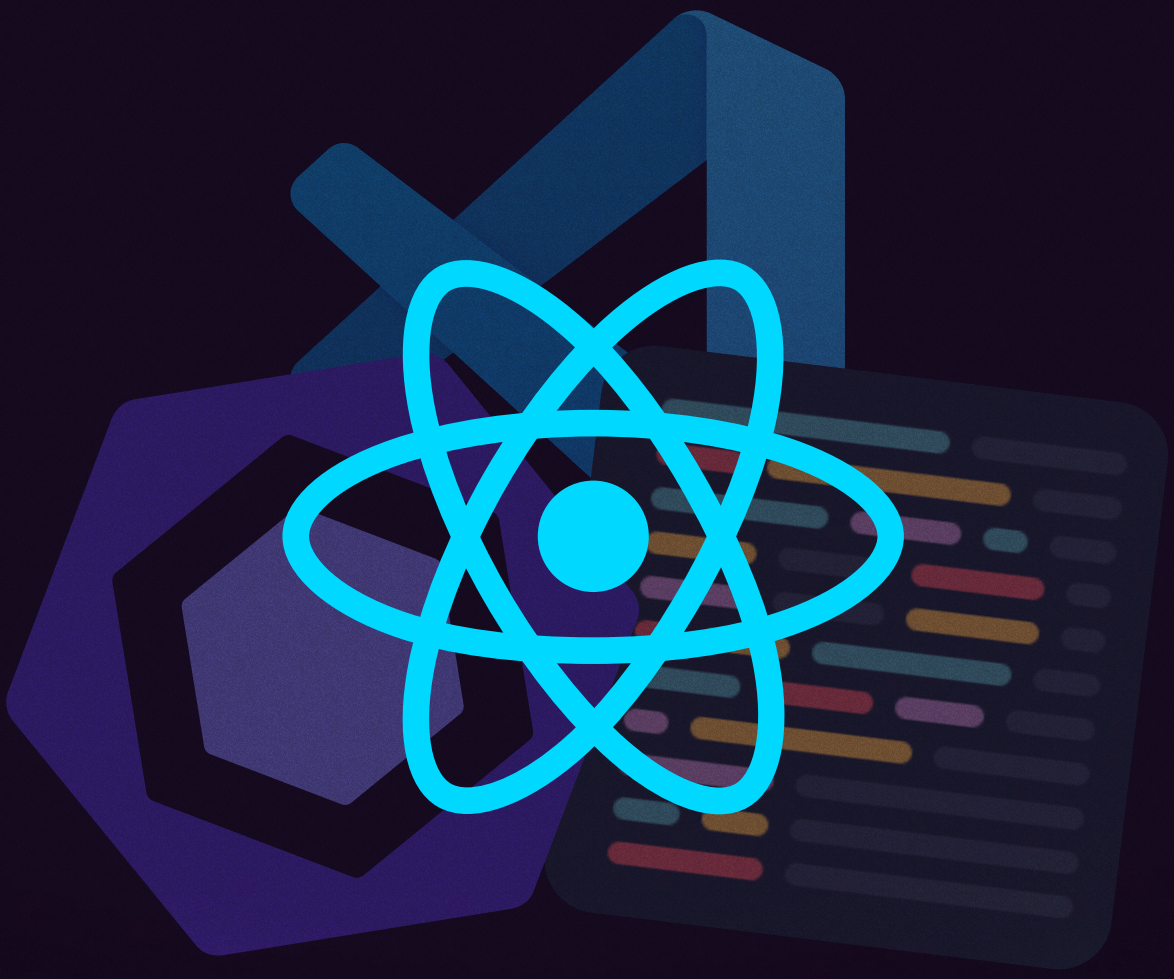
To test that the Tailwind CSS plugin is also working. Try to put flex class names as the last class member and see if it is returned to the front when saving.

IMPORTANT: Install the Prettier Eslint Package too (a third one!).



Conclusion

You can finally start working on your project rather than pulling your hair out with configurations. 😄



React.js Setup

React.js Setup

Start a React project by running the following command in your terminal,

```
npm create vite@latest
```

Select React and TypeScript when asked.

Go inside created project, run install packages. Finally, open the project in VSCode. By default ESLint is included in the Vite project so you don't have to reinstall ESLint.

```
cd project_name

npm install
```

Finally, open the project in VSCode. By default ESLint is included in the Vite project so you don't have to reinstall ESLint.

React.js Setup

1. Remove the **`.eslintrc.cjs`** file at the root of the project and replace it with a **`.eslintrc.json`** file with the following content:

```
{
  "extends": [
    "standard",
    "plugin:react/recommended",
    "plugin:tailwindcss/recommended",
    "prettier"
  ],
  "rules": {
    "max-len": [2, 250],
    "no-multiple-empty-lines": [
      "error",
      {
        "max": 1,
        "maxEOF": 1
      }
    ],
    "object-curly-newline": 0
  }
}
```

React.js Setup

2 Install the required packages

```
npm install eslint-config-standard eslint-plugin-
tailwindcss eslint-config-prettier
```

- eslint-config-standard is a package for enforcing the standard coding style. Look at the rules here.
- eslint-plugin-tailwindcss is a package for enforcing logical ordering of Tailwind CSS classes.
- eslint-config-prettier is a package to avoid conflict sbetween ESLint and Prettier.

3 Install prettier package

```
npm install prettier
```

4 Set VSCode to work with ESLint and Prettier. Follow the steps outlined previously in Next.js Setup for VSCode

The End

Congratulations on reaching the end of our guide! But hey, learning doesn't have to stop here.

If you're craving a more personalized learning experience with the guidance of expert mentors, we have something for you — Our Masterclass.

JSM Masterclass Experience

In this special program, we do not just teach concepts – offering hands-on training, workshops, one on one with senior mentors, but also help you build production-ready applications in an industry-like environment, working alongside a team and doing code reviews with mentors. It's almost a real-world experience simulation, showcasing how teams and developers collaborate.

If this sounds like something you need, then don't stop yourself from leveling up your skills from junior to senior.

Keep the learning momentum going. Cheers! 🚀