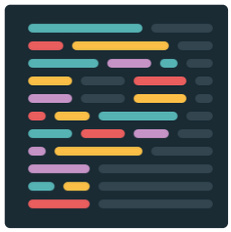

Prettier

Configuração

O **Prettier** é mais uma ferramenta que vamos utilizar para ajudar na padronização de código, ele consiste em várias configurações que são feitas para que o código seja formatado para seguir um padrão.

Alguns exemplos de formatações que ele faz é a quebra de linha quando ela tem mais de 80 caracteres, adicionar `;` no final das linhas dentre outras funcionalidades muito úteis para um projeto.

> 💡 ⚠️ Antes de começar a configuração é importante que você se certifique de remover a extensão **Prettier - Code Formatter** do seu VS Code, ela pode gerar incompatibilidades com as configurações que vamos fazer.

> 💡 As configurações são iguais para os projetos **NodeJS**, **ReactJS** e **React Native!**

A primeira coisa que vamos fazer para a configuração do **Prettier** é a instalação dos pacotes no projeto, e vamos isso executando:

```shell
yarn add prettier eslint-config-prettier eslint-plugin-prettier -D
```

Esse comando vai adicionar 3 dependências que serão as responsáveis por fazer a formatação do código e também integrar o **Prettier** com o **ESLint**.

Com a instalação feita vamos modificar o arquivo `.eslintrc.json` adicionando no `"extends"` as seguintes regras:

```
"prettier/@typescript-eslint", "plugin:prettier/recommended"
```

JSON

Nos `"plugins"` vamos adicionar apenas uma linha com:

```
"prettier"
```

JSON

E nas `"rules"` vamos adicionar uma linha indicado para o **ESLint** mostrar todos os erros onde as regras do **Prettier** não estiverem sendo seguidas, como abaixo:

```
"prettier/prettier": "error"
```

JSON

O arquivo final vai ficar assim:

```
{ ... "extends": [ ... "prettier/@typescript-eslint", "plugin:prettier/recommended" ],
... "plugins": [ ... "prettier" ], "rules": { ... }, ... }
```

JSON

E para resolver os conflitos entre as regras do **ESLint** e as regras do **Prettier** vamos criar um arquivo na raiz do projeto como `prettier.config.js` , nesse arquivo vamos adicionar 3 regras, sendo elas:

`singleQuote` para utilizar aspas simples (`'`);

`trailingComma` para adicionar vírgula (`,`) sempre ao final de um objeto que tenha sido quebrado em várias linhas

`arrowParens` para que não seja adicionado parênteses (`()`) quando uma Arrow Function tiver apenas um parâmetro

O arquivo no final vai ficar assim:

```
module.exports = { singleQuote: true, trailingComma: 'all', arrowParens: 'avoid', }
```

JavaScript

E para finalizar a configuração, vamos criar na raiz do projeto um arquivo `.eslintignore` e nele vamos adicionar a linha:

```
/*.js
```

JavaScript

E a configuração está finalizada, para garantir que o código seja formatado corretamente, você pode abrir os arquivos do projeto e salvar eles novamente.

Documentação

Para mais informações sobre as configurações do Prettier, você pode consultar a documentação oficial:

Options · Prettier

Prettier ships with a handful of customizable format options, usable in both the CLI and API. Specify the line length that the printer will wrap on. For

https://prettier.io/docs/en/options.html