

前言

代码规范是前端工程化落地的基石，在其中扮演着监督者的角色，它主要用于约束团队成员的编码规范与编码风格。应用代码规范有三点好处。

- 强制规范团队编码规范可让新旧成员编码习惯一样
- 增加项目代码的可维护性与可接入性，新成员能快速适应项目的架构与需求
- 保障项目整体质量，可减少无用代码、重复代码、错误代码和漏洞代码的产生几率

这一切的实现都离不开代码校验工具中代码格式化的功能。众所周知，基本所有编辑器都会配置代码校验工具(以下简称 Lint)检测代码中的错误或漏洞，根据提供的修复方案格式化出正确代码，让代码更严谨。本章将带领你部署VSCode的代码格式化，实现保存代码时一键校验，统一所有项目的编码规范与编码风格，养成规范的码字习惯，减少不必要的错误与隐患。

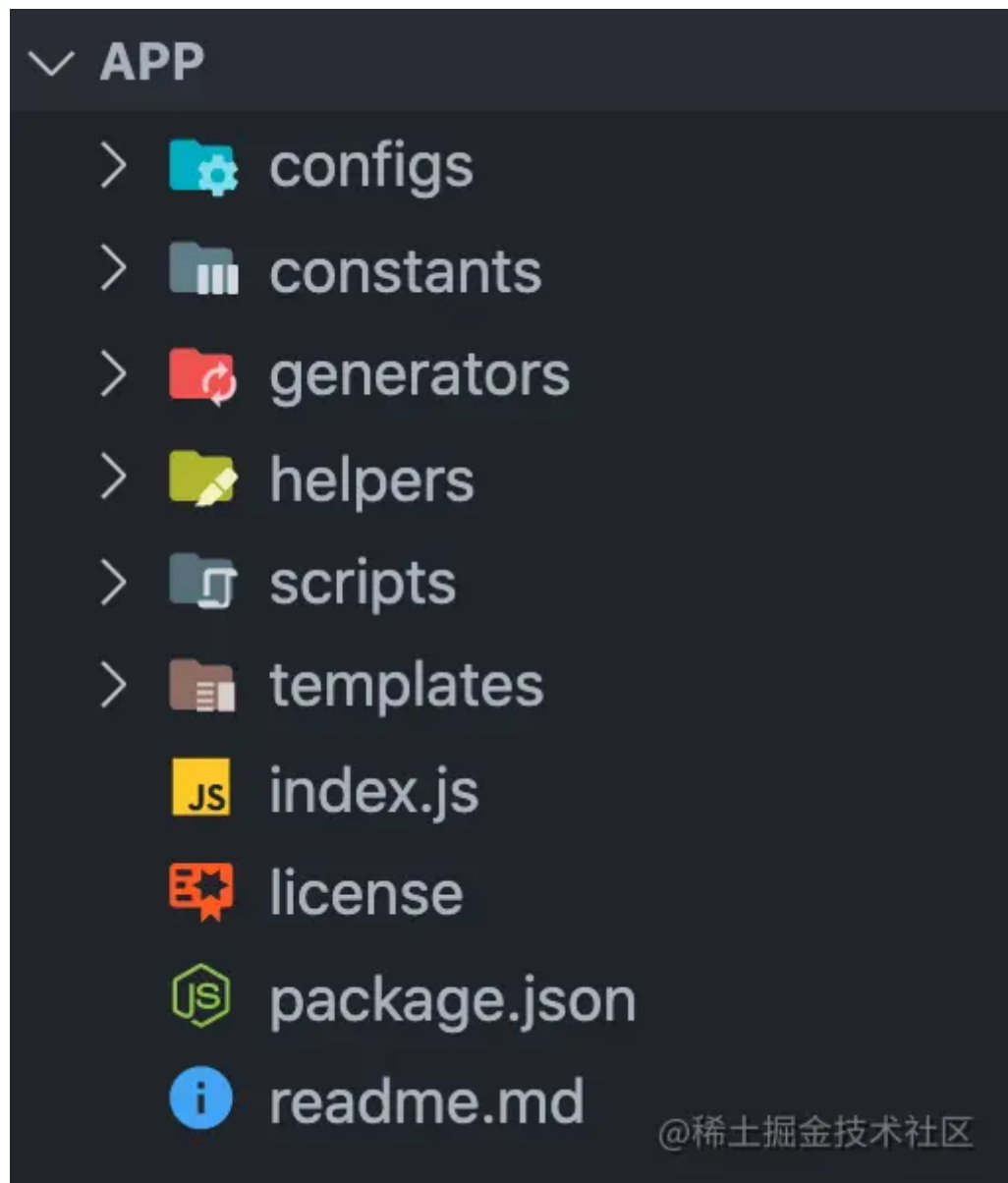
背景：一次部署全局运行

Lint 其实就是编辑器中运行的一个脚本进程，将代码解析为抽象语法树，遍历抽象语法树并通过预设规则做一些判断与改动，再将新的抽象语法树转换为正确代码。整个校验过程都跟抽象语法树相关，若暂未接触过抽象语法树可阅读babel源码或eslint源码了解其工作原理。

如何配置 Lint，百度谷歌一搜一大堆，在此不深入讲述了。在保存文件时触发 Lint 自动格式化代码，该操作当然不能 100% 保证将代码格式化出最正确的代码，而是尽可能根据修复方案格式化出正确的代码。简而言之，可能存在部分代码格式化失败，但将鼠标移至红色下划线上会提示修复方案，可根据修复方案自行修正代码。

我很喜欢严谨的代码逻辑与优雅的编码风格，所以也很喜欢格式化代码，但我不想为每个项目配置 Lint。这些重复无脑的复制粘贴让我很反感，所以更多时候我只想对 Lint 做到一次部署全局运行。

代码规范 很适合使用 一次部署全局运行 这样的方式处理。若使用该方案，相信能将所有项目的 `stylelint/eslint/tslint/prettier` 相关依赖与配置文件全部删除，使项目结构变得超级简洁。



在部署 代码规范 前，我想要强调以下情况。

- `Tslint`官方 已宣布废弃 `tslint`，改用 `eslint` 代替其所有校验功能
- `eslint` 部分配置与 `prettier` 部分配置存在冲突且互相影响，为了保障格式化性能就放弃接入 `prettier`

所以 `VSCode` 只需安装 `Stylelint` 与 `Eslint` 两个插件。为了方便理解，统一以下名词。

- 以下提到的 `Stylelint` 与 `Eslint` 都为 `VSCode`插件
- 以下提到的 `stylelint` 与 `eslint` 都为 `Npm`模块

方案：部署VSCode的代码格式化

整体配置相比单独为项目接入 `stylelint` 与 `eslint` 更简单。

安装依赖

为了搞清两个插件集成哪些依赖，以下区分安装 `stylelint` 与 `eslint` 及其相关依赖。不过，先别急着安装，后面我有更简单的解决方案。我有个习惯：喜欢将依赖更新到最新版本，在享受新功能的同时也顺便填坑。

```
# stylelint及其依赖
npm i stylelint stylelint-config-standard stylelint-order postcss-html postcss-scss postcss
```

```
# eslint及其依赖
npm i @babel/core @babel/eslint-parser @babel/preset-react eslint eslint-config-standard
```

```
# typescript-eslint及其依赖
npm i -D @typescript-eslint/eslint-plugin @typescript-eslint/parser typescript eslint-config
```

安装完毕需配置多份文件，**CSS类型**有 `html/css/scss/less/vue`文件，**JS类型**有 `html/js/ts/jsx/tsx/vue`文件。查看插件文档，`Stylelint` 的配置文件可同时校验 `html/css/scss/less/vue`文件，`Eslint` 需配置不同文件分别校验 `html/js/ts/jsx/tsx/vue`文件。两个插件可在 `settings.json` 中通过指定字段覆盖默认配置。

`settings.json`是VSCode的配置文件，用户可通过插件暴露的字段自定义编辑器功能。

txt

因为配置文件太多不好管理，我将这些配置文件整合起来，封装为 [@yangzw/bruce-std](#)。这样就能跳过上述的依赖安装，执行以下命令安装 `@yangzw/bruce-std`。

```
npm i -g @yangzw/bruce-std
```

`@yangzw/bruce-std` 包括以下文件夹。

- **demo文件夹**
 - ✓ [demo](#): 随便捣鼓几个示例用于测试格式化代码
- **stylelint文件夹**
 - ✓ [stylelinttrc.js](#): 校验 [html/css/scss/less/vue](#)文件
- **eslint文件夹**: 校验 [html/js/jsx/vue](#)文件
 - ✓ [eslinttrc.js](#): 校验 [html/js](#)文件
 - ✓ [eslinttrc.react.js](#): 校验 [html/jsx](#)文件
 - ✓ [eslinttrc.vue.js](#): 校验 [html/vue](#)文件
- **tslint文件夹**: 校验 [html/ts/tsx/vue](#)文件 ([tslint](#) 已弃用, 使用 [eslint](#) 代替)
 - ✓ [tsconfig.json](#): 配置 [TypeScript](#)
 - ✓ [tslinttrc.js](#): 校验 [html/ts](#)文件
 - ✓ [tslinttrc.react.js](#): 校验 [html/tsx](#)文件
 - ✓ [tslinttrc.vue.js](#): 校验 [html/vue](#)文件

配置文件中的 [rules](#) 可根据自身 [编码规范](#) 与 [编码风格](#) 适当调整, 在此不深入讲述了。推荐使用 [@yangzw/bruce-std](#) 默认规则, 若校验规则不喜欢可自行调整。

- 配置 [Stylelint](#) 可查看[Stylelint规则](#)
- 配置 [Eslint](#) 可查看[Eslint规则](#)
- 配置 [TypeScriptEslint](#) 可查看[TypeScriptEslint规则](#)
- 配置 [VueEslint](#) 可查看[VueEslint规则](#)

配置插件

到了配置插件这一步, 其实操作不复杂, 直接把过程罗列出来, 跟着我一步一步完成。

- 打开 [VSCode](#)
- 选择左边 [工具栏](#) 的 [插件](#), 搜索并安装 [Stylelint](#) 与 [Eslint](#), 安装完毕重启 [VSCode](#)
- 选择 [文件](#) → [首选项](#) → [设置](#), [设置](#) 中可选 [用户](#) 或 [工作区](#)
 - **用户**: 配置生效后会作用于全局项目(若很多项目都是单一的[React](#)应用或[Vue](#)应用推荐使用全局配置)
 - **工作区**: 配置生效后只会作用于当前打开项目
- 点击 [设置](#) 右上角中间图标 打开设置(json), 打开的对应文件是 [settings.json](#)
- 加入以下内容并重启 [VSCode](#): 为了保障每次改动都能正常格式化代码, 必须重启 [VSCode](#)

```
{
  // 默认自定义配置
  "css.validate": false,
  "less.validate": false,
  "scss.validate": false,
  "editor.codeActionsOnSave": {
    "source.fixAll.eslint": true,
    "source.fixAll.stylelint": true
  },
  // 扩展自定义配置
  "eslint.nodePath": "path/@yangzw/bruce-std/node_modules",
  "eslint.options": {
    "overrideConfigFile": "path/@yangzw/bruce-std/eslint/eslintrc.js" // 可
  },
  "stylelint.configBasedir": "path/@yangzw/bruce-std",
  "stylelint.configFile": "path/@yangzw/bruce-std/stylelint/stylelintrc.js",
  "stylelint.customSyntax": "postcss-scss", // 可变
  "stylelint.stylelintPath": "path/@yangzw/bruce-std/node_modules/stylelint",
  "stylelint.validate": ["html", "css", "scss", "less", "vue"]
}
```

上述配置的 `path` 为 `@yangzw/bruce-std` 模块所在的**Npm根目录**，可执行 `npm config get prefix` 获取**Npm根目录**并替换 `path`。

- 执行 `npm config get prefix` 获取**Npm根目录**，例如是
`E:/Node/prefix/node_modules`
- 将上述配置的 `path` 替换为 `E:/Node/prefix/node_modules`

请保持**VSCode**及其所有插件都为**最新版本**，防止因为版本问题导致 `Stylelint` 与 `Eslint` 失效。

选择配置

首先配置 `Stylelint`。校验不同类型代码需实时修改 `stylelint.customSyntax` 的值。

- ☑ **CSS/SCSS**: `postcss-scss`
- ☑ **CSS/LESS**: `postcss-less`
- ☑ **HTML/VUE**: `postcss-html`

其次配置 **Eslint** 。校验不同类型代码需实时修改

`eslint.options.overrideConfigFile` 的值。

- ☑ **JS**: `path/@yangzw/bruce-std/eslint/eslintrc.js`
- ☑ **React**: `path/@yangzw/bruce-std/eslint/eslintrc.react.js`
- ☑ **Vue**: `path/@yangzw/bruce-std/eslint/eslintrc.vue.js`
- ☑ **TS**: `path/@yangzw/bruce-std/tslint/tslintrc.js`
- ☑ **React TS**: `path/@yangzw/bruce-std/tslint/tslintrc.react.js`
- ☑ **Vue TS**: `path/@yangzw/bruce-std/tslint/tslintrc.vue.js`

以上述路径 `E:/Node/prefix/node_modules` 为例。在默认情况下，**Eslint** 只能校验普通**JS**。若校验其他**JS**类型 需将 `eslint.options.overrideConfigFile` 的 `path` 改成以下路径。

- **React**: `E:/Node/prefix/node_modules/@yangzw/bruce-std/eslint/eslintrc.react.js`
- **Vue**: `E:/Node/prefix/node_modules/@yangzw/bruce-std/eslint/eslintrc.vue.js`
- **TS**: `E:/Node/prefix/node_modules/@yangzw/bruce-std/tslint/tslintrc.js`
- **React TS**: `E:/Node/prefix/node_modules/@yangzw/bruce-std/tslint/tslintrc.react.js`
- **Vue TS**: `E:/Node/prefix/node_modules/@yangzw/bruce-std/tslint/tslintrc.vue.js`

示例：一键修复并格式化代码

上述步骤操作完毕就可愉快地敲代码了。每次保存文件可自动格式化 **CSS**代码 与 **JS**代码，该功能不仅将代码根据规范 整理 与 排序，甚至尽可能根据修复方案格式化出正确的代码。

这样就无需为每个项目配置 **Lint**，将所有项目的

`stylelint/eslint/tslint/prettier` 相关依赖与配置文件全部删除，使项目结构变得超级简洁。值得注意，若项目太复杂或太老旧，就不要折腾了，要保持对旧代码的敬畏之心！

```
index.css • index.scss
demo > index.css > #app
You, 2 minutes ago | 1 author (You)
1 * {
2   padding: 0;
3   margin: 0;
4 }
5 #app {
6   background-color: #ffffff;
7   opacity: 0.5;
8 }
```

@稀土掘金技术社区

html/js/ts/jsx/tsx/vue文件

```
index.js • index.jsx
demo > index.js > ...
You, a minute ago | 1 author (You)
1 const name = 'JowayYoung'
2 document.getElementById('app').innerText = "Hello, " + name
```

@稀土掘金技术社区

答疑：VSCode细节问题

更新eslint到v6+就会失效

很多同学反映 `eslint v6+` 在 `VSCode` 中失效，最高使用版本只能控制在 `v5.16.0`。其实这本身就是配置问题，跟版本无关。`@yangzw/bruce-std` 的 `eslint` 使用 `v8`，只要配置正确就能正常使用。

上述安装行为使用了 `Npm`，那 `settings.json` 的 `eslint.packageManager` 必须配置为 `npm` (小写)，但最新版本 `Eslint` 已默认此项，所以无需配置。若执行 `yarn global add @yangzw/bruce-std`，那必须在 `settings.json` 中加入以下内容。

```
{  
    "eslint.packageManager": "yarn"  
}
```

首次安装Eslint并执行上述配置就会失效

首次安装 `Eslint` 可能会在 `js/ts/jsx/tsx/vue` 文件的控制台中看到以下警告。

```
Eslint is disabled since its execution has not been approved or denied yet. Use the ligh
```

该警告说明 `Eslint` 被禁用。虽然配置中无明确的禁用字段，但还是被禁用了。移步到 `VSCode` 右下角的工具栏会看到 `禁用图标+ESLINT` 的标红按钮，单击它就会弹出一个弹框，选择 `Allow Everywhere` 就能启用 `Eslint` 的校验功能了。

为何不考虑使用Prettier

首先 `Eslint` 是保障 `JS类型代码` 的质量工具，该工具必须配置。其次 `Vscode` 现在可基于 `Eslint` 格式化出正确的代码。既然一个编辑器已内置了格式化代码的功能，为何还要额外引入一个 `Prettier`？

使用过 `Eslint` 与 `Prettier` 的同学都知道，两者的标准配置存在冲突状态，那修改 `Eslint` 与 `Prettier` 哪个配置？反正都得改，改完还得确认。

若基于 `Eslint` 使用 `Vscode` 内置的格式化功能，那只需专注 `eslint配置`，完全无必要花时间维护 `Eslint` 与 `Prettier` 的冲突配置啊！

这些事情仁者见仁智者见智，相信很多同学使用了本方案会喜欢更纯净的 `eslint` 配置，而不是 `Eslint` 与 `Prettier` 合并还要修改冲突这样的配置。

总结

若准备接手一个离职同事的代码，而他的代码完全是想写就写无任何规范可言，此刻的你是哪种心情？接着如何解决？

后续遇到的工程化问题也可借鉴本章思路，将一些具备全局特征的配置分离出来统一管理，利用 `模块化` 的特性分离部分构建代码以减少项目结构的复杂度，使得项目更具条理性。

有些同学可能一时适应不了 `Lint` 带来的强制性操作会在编辑器中关闭项目所有校验功能，这种自私行为会带来很严重的后果。

若上传无任何校验痕迹的代码块，当其他成员将该代码块更新合并到原有代码后，因为编辑器一直配置着 `Lint`，导致被拉下来的代码块立即报错甚至产生冲突。

不仅让其他成员花费更多时间解决这些额外问题，还浪费了团队的精力。若团队无任何规范可随意编码，若已认可团队规范就请努力遵守，不给团队带来麻烦。

本章内容到此为止，希望能对你有所启发，欢迎你把自己的学习心得打到评论区！

☑ 示例项目：[fe-engineering](#)


☑ 正式项目：[bruce](#)

留言

输入评论（Enter换行，Ctrl + Enter发送）

发表评论

全部评论（61）

qingkooo 

前端开发 1月前

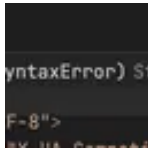
老师，我们如何维护自己的配置呢？

如果团队clone一份@yangzw/bruce-std改吧改吧，然后也发布一个npm包。后续每次修改完毕都要重新发布、重新安装新版本吗？有更加简便的方法吗？

👍 点赞 💬 回复

qingkooo  前端开发 1月前

配置完毕后，测试html时发现stylelint对<!doctype html>这样的语法标红了。
我还是想保留html行内样式的格式检查，请问下该怎么改哪条stylelint的规则呢？



👍 3 💬 回复



oldfu  前端开发 1月前

公司里没法统一编辑器，各有各的爱好，各有各的使用习惯。

👍 1 💬 回复



Willon  猪肉佬 @ 某肉联厂 2月前

如果编辑器配置文件的path能指向网络资源的配置就更好了，连安装的那步都省了🐵

👍 点赞 💬 1

JowayYoung  (作者) 2月前

我也希望这样

👍 点赞 💬 回复



fakership  前端 @ null 2月前

其实prettier跟eslint冲突的事情 有eslint-plugin-prettier去解决 它的解决方案就是禁掉冲突rules 放到prettier里去做这件事

👍 1 💬 回复



juln  前端工程师 @ 虎扑 3月前

我认为像代码校验这种流程应该放到工作流的某个环节或者某个钩子上，这样才能确保无遗漏。最好是在commit或push之前的钩子，然后ci的时候再去校验一次。我心中理想的方案是搞个专门用于校验团队代码的cli工具（包含团队的lint规则），然后用husky和lint-stage去触发这个命令

👍 2 💬 3

juln 3月前

commit前校验是为了保证每次提交的代码都是对的，ci的时候再次校验是为了保证commit没有用-n跳过校验，且保证当前项目的代码都符合最新的lint规则

👍 点赞 💬 回复

juln 3月前

本地配置最多只是用来辅助开发，不适合做确保工作

👍 点赞 💬 回复

查看更多回复 ▾



大花花 JY.3 API测试工程师 3月前

值

👍 点赞 💬 1

JowayYoung (作者) 3月前

谢谢支持

👍 1 💬 回复



刘小灰 LV.4 JY.4 前端开发工程师 @ 码... 3月前

受本章节的启发，写了个工具更加方面的进行项目规范化配置 github.com

👍 2 💬 1

JowayYoung (作者) 3月前

不错，学以致用

👍 点赞 💬 回复



EEEEEEEEEE JY.4 前端 3月前

可以理解为 中心化管理配置文件, 易于维护和管理, 且所有项目都使用统一的配置, 而无需单独配置?

👍 点赞 💬 1

JowayYoung (作者) 2月前

是的，就是中心化管理

👍 点赞 💬 回复



WangShuXian6 JY.3 3月前

这里显得作者很不专业。eslint是代码质量校验，Prettier是代码格式校验。明明可以通过专有插件很容易解决冲突规则。编辑器应当始终使用项目的配置文件取格式化代码，项目是统一的，编辑器确实五花八门。

👍 点赞 💬 5

JowayYoung (作者) 3月前

这只是一种解题思路，可取可不取，那请你说出专业且兼容全部情况的解决方案

👍 2 💬 回复



aComputer... 1月前

请教您务必说下专业的，有落地实践的方法 谢谢

👍 1 💬 回复

查看更多回复 ▾



cnelf JY.3 前端开发 3月前

学习了，有个疑问是 @yangzw/bruce-std 可以直接安装到项目里边吗？然后使用工作区的 settings.json，感觉让团队成员手动去安装依赖配置不太现实。

👍 点赞 💬 1

JowayYoung (作者) 3月前

不要安装到项目，要安装到全局，然后全局配置，对某些项目可使用工作区配置覆盖。针对这个问题“感觉让团队成员手动去安装依赖配置不太现实”，我觉得@yangzw/bruce-std已经很友善了，总比你手动配置stylelint和eslint强吧

👍 点赞 💬 回复



ticktock76323 JY.4 3月前

请教一下这种配置方式是不是只能全部人都是用vscode呀？这样对于有使用webstorm的团队来说是不是就不太友好？

👍 点赞 💬 1

JowayYoung (作者) 3月前

是的，没有探究过ws

👍 点赞 💬 回复



听话不听话 JY.2 前端 4月前

开发一个vscode插件，统一eslint 配置可以？也可以一键升级

👍 点赞 💬 1



小彩机 2月前

这个我不确定，有大佬可以解答下吗？

👍 点赞 💬 回复



AlexRen94

LV.1

JY.3



前端工程师

4月前

关于eslint和prettier冲突的问题，有个eslint-plugin-prettier插件可以解决。同意楼上有一位说的，格式化和代码检查各司其职，eslint在某些场景，比如单行过长和jsx换行格式化上没有prettier好看，个人更偏好二者同时使用，分别有侧重

👍 4 💬 回复

console_man

JY.3

Web前端

4月前

这种方式对于升级lint简直不要太方便

👍 1 💬 1

JowayYoung



(作者)

4月前

一件升级所有lint配置，别的方案都做不到

👍 点赞 💬 回复

console_man

JY.3

Web前端

4月前

哇 这个思路好神奇 佩服

👍 2 💬 1

JowayYoung



(作者)

4月前

是呀，可以帮忙推广下吗

👍 点赞 💬 回复



furfurJiang

JY.4

前端 @ 程序江

4月前

遇到问题记录：

“npm config get prefix获取Npm根目录并替换path”

path替换的时候遇到问题，使用npm config get prefix命令行获取的是C:/Program Files/nodejs，

但是配置后vscode提示找不到，自己区文件夹查看发现需要加上/node_modules，才能找到全局按照的@yangzw包...

[展开](#)

👍 点赞 💬 1

JowayYoung



(作者)

4月前

不错的话可以帮忙推广下这种lint代码的方式，哈哈

👍 点赞 💬 回复



馍馍汉堡

JY.4



4月前

可以用editorconfig代替prettier

👍 1 💬 2

JowayYoung (作者) 4月前

能用vscode做的功能就不使用其他工具了，工具太多开发起来也要维护很多东西

👍 点赞 🗨 回复



馍馍汉堡 回复 JowayYoung 4月前

editorconfig就是一个vscode插件😂

“能用vscode做的功能就不使用其他工具了，工具太多开发起来也要维护...”

👍 1 🗨 回复



丶从头喜欢你 JY.4 前端工程师 4月前

请问怎么修改eslint的配置呢，不喜欢末尾加分号。

👍 点赞 🗨 1

JowayYoung (作者) 4月前

刚才查了一下文档，可以尝试通过vscode的settings.json配置vscode eslint.rules.customizations处理下，如果不行的话Eslint这个插件暂时没有相关字段配置覆盖的rules了，只能在@yangzw/bruce-std中修改rules了

👍 点赞 🗨 回复



Ivrice_start JY.2 web 前端 4月前

Eslint 和Prettier，还是需要配置的。eslint 是保证代码的质量工具，prettier 是格式化代码工具，遵循单一原则，专干专事。

如果两者工具有冲突的情况，不应该是简单的把 eslint 规则关掉，而是把问题分类，再解决。

个人观点，请指教 😊

👍 2 🗨 6

JowayYoung (作者) 4月前

来了来了。首先eslint是保证代码的质量工具这个完全赞同，其次是vscode现在可以基于eslint的配置去格式化代码，输出正确的代码。既然一个编辑器已内置了格式化代码的功能，为何还要额外去引入一个工具prettier呢？

使用过prettier与eslint的都知道，它们两者的标准配置都存在冲突，那这时改prettier与eslint哪个好，反正都得改是吧，改完还要去确认。。。...

[展开](#)

👍 3 🗨 回复



doho 4月前

不懂就问，提交前的代码整理（prettier）能通过vscode解决吗

 点赞  回复

查看更多回复 

查看全部 61 条回复 