**Vite2 + Vue3 + TypeScript + Pinia搭建开发脚手架**

一、搭建前准备

1、**Vscode**: 前端人必备写码神器

2、**Chrome**：对开发者非常友好的浏览器(反正我是很依赖它的)

3、**Nodejs** & **npm**：配置本地开发环境，安装 Node 后你会发现 npm 也会一起安装下来

4、**Vue-devtools**：浏览器调试插件

5、**Vue Language Features (Volar)**：Vscode 开发 vue3 必备插件，提供语法高亮提示，非常好用

6、**Vue 3 Snippets**：vue3 快捷输入

二、使用 vite 快速创建脚手架(node >= 12.0.0)

根据框架构建模板，命令如下：

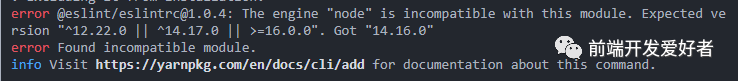


三、约束代码风格

1、**ESLint支持**: 安装如下包



**重点注意：**如果eslint安装报错，执行以下命令

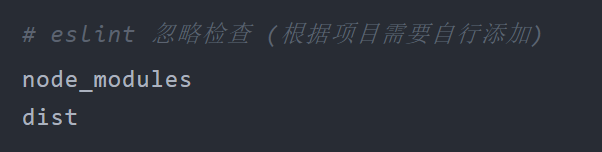


**尝试运行如下命令：**yarn config set ignore-engines true

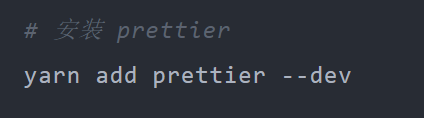
运行成功以后再次安装eslint

2、**项目下新建.eslintrc.js**：配置eslint校验规则（下图为部分截图，详细配置查看vue3\_pc模板中内容）

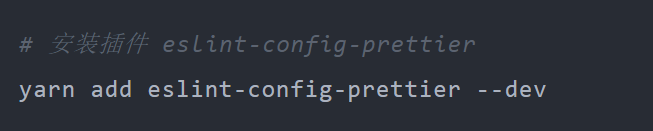
3、**项目下新建.eslintignore：**配置eslint检查忽略



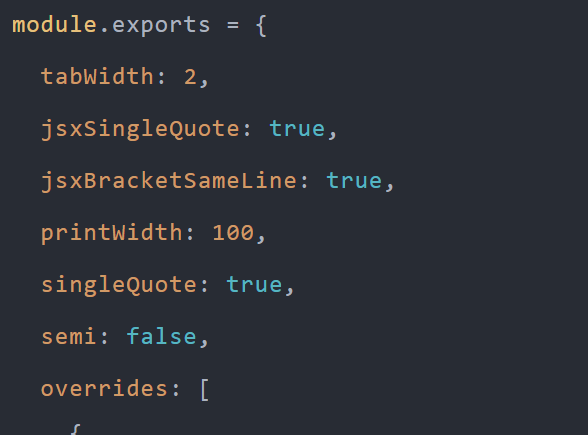
4、**安装prettier支持**



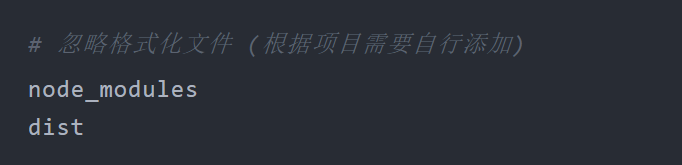
5、**解决eslint和prettier冲突：**解决eslint中的样式规范和prettier中样式规范冲突，以prettier的样式规范为准，使ESLint中的样式规范自动失效



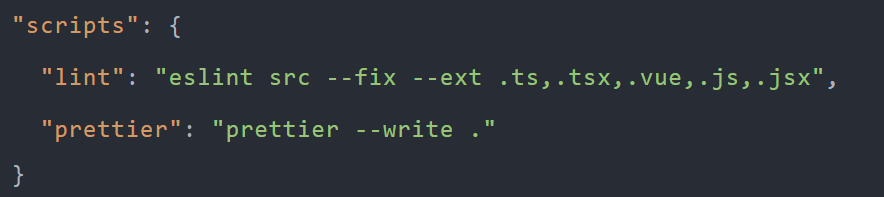
6、**项目下新建.prettier.js：**配置prettier格式化规则



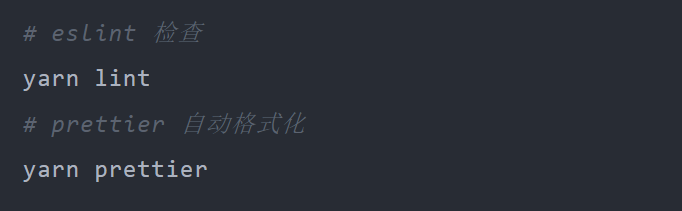
7、**项目下新建.prettierignor：**忽略格式化文件



8、**配置package.json（scripts内增加lint和prettier）**



上面配置完成后，可以运行以下**命令**测试下代码检查个**格式化**效果



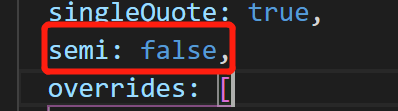
重点问题：ESLint与Prettier分号冲突问题，解决方案如下：

1、配置.eslintrc.js

配置分号，则在rules字段增加配置：

'semi': ["error", "always"] - eslint检测必须有分号，否则报错  
'semi': ["warn", "never"] - eslint检测不能有分号，否则报警  
'semi': 0 - eslint不检测分号，即分号可加可不加

2、配置.prettier.js



四、配置husky + lint-staged

注释：使用husky + lint-staged助力团队编码规范, husky & lint-staged安装推荐使用mrm,它将根据package.json依赖项中的代码质量工具来安装和配置husky和lint-staged，因此请确保在此之前安装并配置所有代码质量工具，如Prettier和ESlint等

1. **husky含义**

husky 是一个为 git 客户端增加 hook 的工具。安装后，它会自动在仓库中的 .git/ 目录下增加相应的钩子；比如 pre-commit 钩子就会在你执行 git commit 的触发。

那么我们可以在 pre-commit 中实现一些比如 lint 检查、单元测试、代码美化等操作。当然，pre-commit 阶段执行的命令当然要保证其速度不要太慢，每次 commit 都等很久也不是什么好的体验。

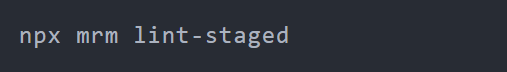
2、**lint-staged含义**

lint-staged，一个仅仅过滤出 Git 代码暂存区文件(被 git add 的文件)的工具；这个很实用，因为我们如果对整个项目的代码做一个检查，可能耗时很长，如果是老项目，要对之前的代码做一个代码规范检查并修改的话，这可能就麻烦了呀，可能导致项目改动很大。

所以这个 lint-staged，对团队项目和开源项目来说，是一个很好的工具，它是对个人要提交的代码的一个规范和约束

3、**首先安装nrm：** 

4、**安装lint-staged：mrm**安装**lint-stated**会**自动**把**husky**一起安装



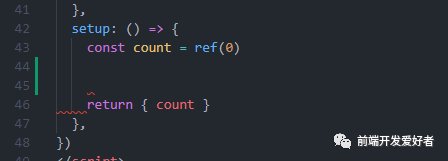
因为我们要结合 **prettier** 代码格式化,所有修改一下配置:



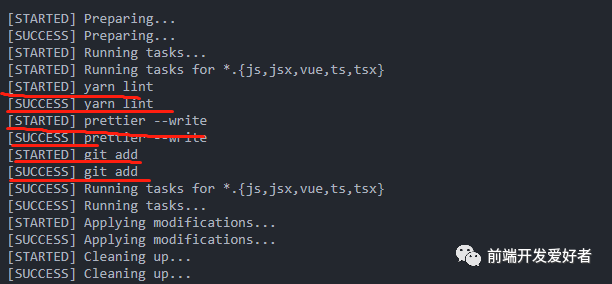
到此，代码格式化配置基本大功告成了!!!

可以修改部分代码尝试git commit，代码将自动格式化：

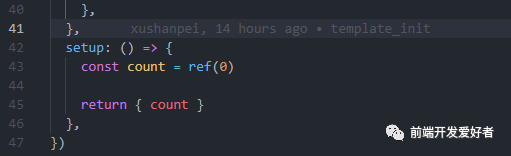
提交前(爆红 – 错误)



执行commit操作，控制台可以看到走了哪些流程



commit后，代码被格式化了

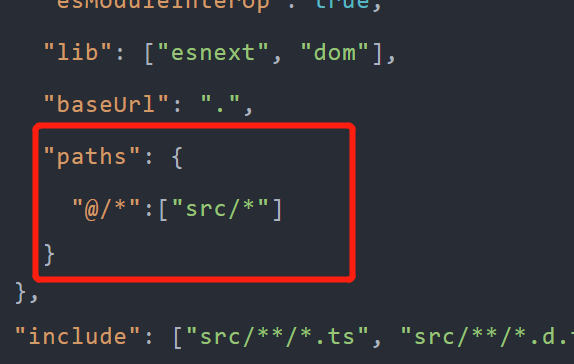


五、配置文件引用别名alias

注释：直接修改vite.config.ts文件配置



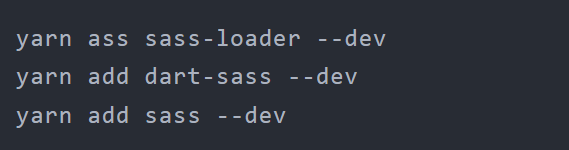
修改tsconfig.json文件



六、配置css预处理器scss

注释：虽然vite原生支持less/sass/scss/stylus，但是必须收到安装它们的预处理器依赖

1、安装依赖



1、配置全局scss样式文件

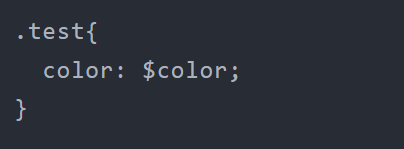
在**src/assets**下新增**css**文件夹，用于存放全局样式文件，新建**main.scss**, 设置一个用于测试的颜色**变量 $color: #c00;**

配置vite**，**项目中**全局注入main.scss**文件



2、组件中使用

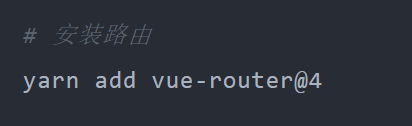
不需要任何引入可以直接使用全局scss定义的变量



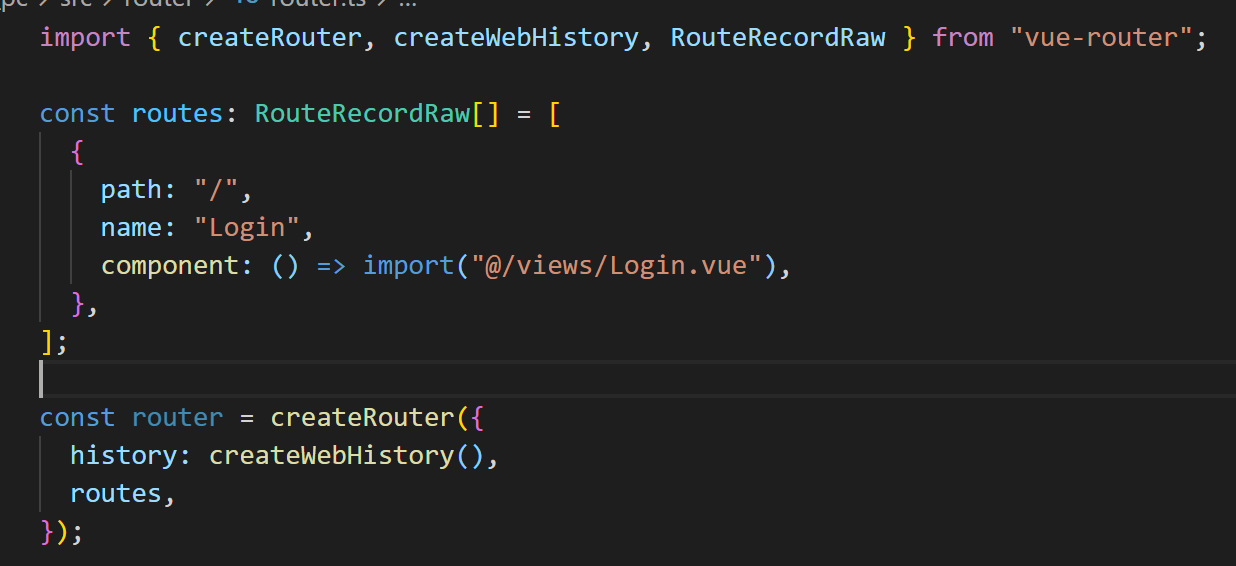
重点提示：组件<style lang=”scss” scoped>，一定要加lang=”scss”

七、路由

1、安装路由



2、在**src**文件下新增**router**文件夹及**index.ts**文件

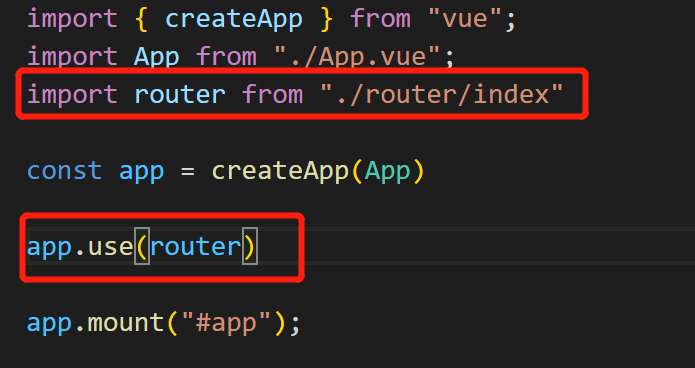


# 重点问题：报错Delete `␍`eslint prettier/prettier

# 解决方案：

# 

3、在**main.ts**文件引入**router**/**index.ts**文件



八、统一请求封装

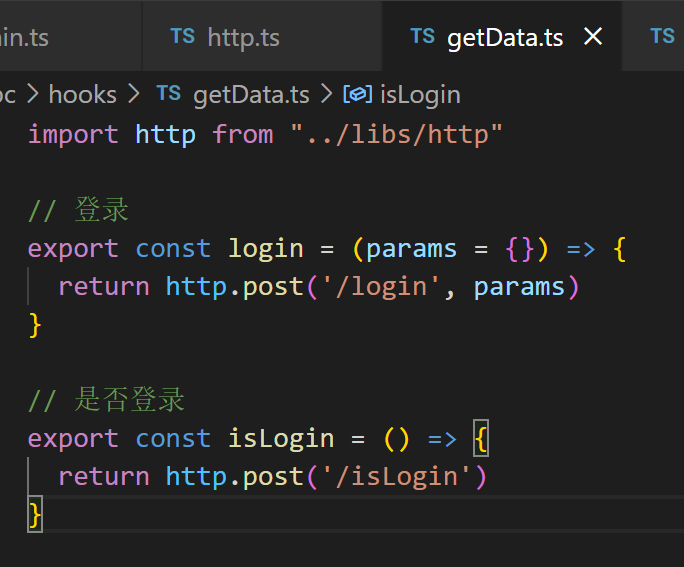
1. 安装Axios & nprogress(进度条及loading 可自定义loading)



2、新增libs文件夹，libs下新增 http.ts 文件，封装axios(详细内容查看模板文件http.ts文件)

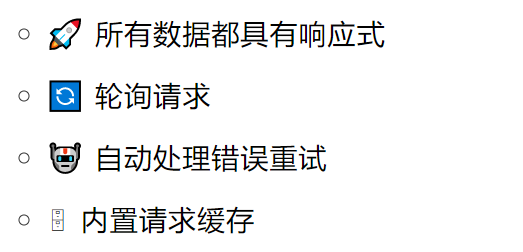
注释：实际使用中可以根据项目修改，比如RESTful api中可以自行添加put和delete请求,ResType也可以根据后端的通用返回值动态的去修改

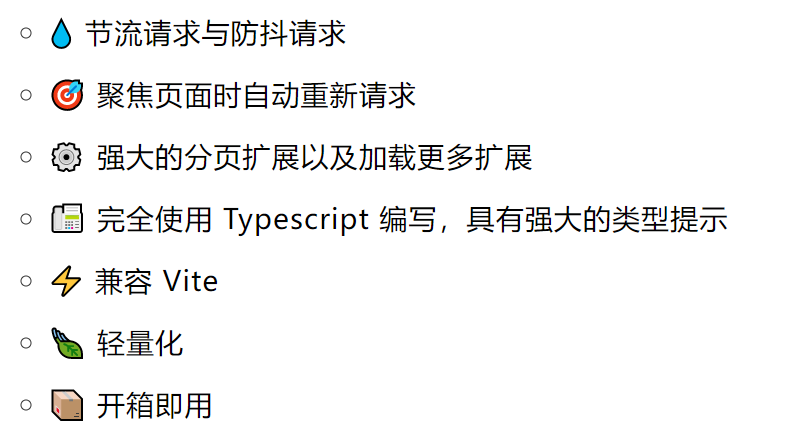
3、新增hooks文件夹，hooks下新增 getData.ts（所有接口请求）和methods（公共方法）文件



重点知识：除了手动封装axios，还推荐一个vue3请求库，VueRequest非常好用,下面来看看 VueRequest有哪些比较好用的功能!!!（功能很强大）

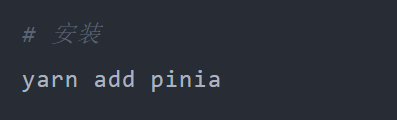
官方链接：https://www.attojs.com/





九、状态管理pinia（如有疑问，查看之前pinia教程）

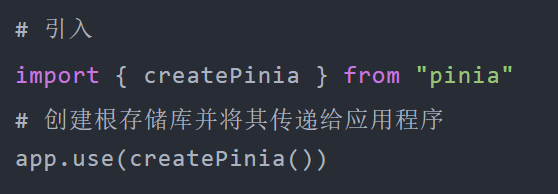
1、安装Pinia



2、创建store/index.ts文件



3、入口main.ts引入pinia



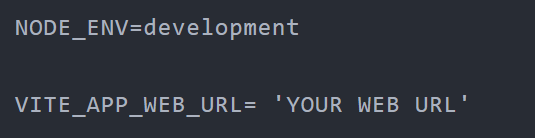
4、组件中引入store/index.ts文件



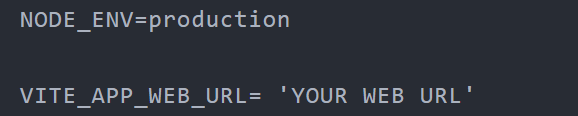
十、环境变量配置

注释：Vite提供两种模式：开发环境（development）和生产环境（production）

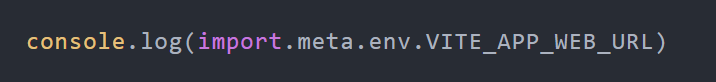
1、项目根目录新建：.env.development



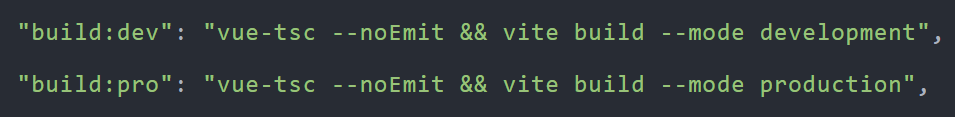
2、项目根目录新建：.env.production



3、组件中使用：



4、配置package.json：打包区分开发环境和生产环境

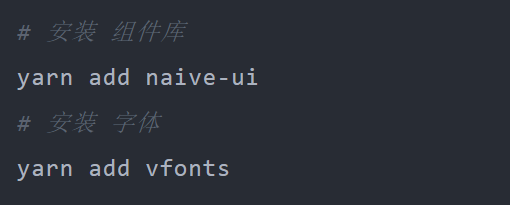


十一、使用组件库Naive UI

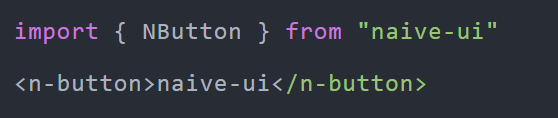
注释：Naive UI组件库，尤大大力推的vue组件库

官网地址：https://www.naiveui.com/zh-CN/os-theme

1、安装Naive UI

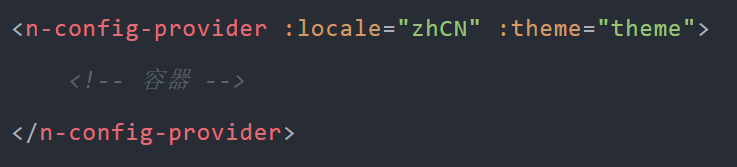


2、组件按需引入



3、全局配置Config Provider

全局化配置设置内部组件的主题、语言和组件卸载于其他位置的DOM的类名



十二、Vite.config.js常用配置

注释：详细配置查看模板中 vite.config.ts 文件