# **Vue开发规范**

## **代码格式化规范**

### 背景

团队项目开发过程中和代码交接时，因个人编码习惯的不同往往出现代码风格不一致的情况，造成不必要的代码维护成本，有时甚至大于新功能的开发成本。

### **安装prettier**

// 使用yarn：

yarn add prettier --dev --exact

// 全局安装

yarn global add prettier

//使用npm：

npm i -D --save-exact prettier

// 全局安装

npm i --global prettier

### **配置文件**



后续会同步各个工程的配置文件。

.prettierrc.js 代码格式配置文件

.eslintrc.js 代码规则配置文件

.eslintignore 第三方引入插件在此处添加路径取消代码格式检验。

\*项目使用过程中，以上（.prettierrc.js，.eslintrc.js）两个属性需要新增或修改的请提出统一修改

### 提交

请在开发过程中修正控制台的报错， 否则将会提交失败。

推荐安装 prettier 插件 ****Prettier - Code formatter****  
可以使用shift+alt+f（vsCode默认快捷键）来格式化代码，可以对单个文件格式化  
或者修改vsCode设置，实现保存既格式化  
"editor.formatOnSave": true

## **1.2. 语法规范**

### **1.2.1. 组件名**

组件名为多个单词，并且用连接线（-）连接，避免与HTML标签冲突，并且结构更加清晰

// 反例

export default {

name: 'item'

}

// 正例

export default {

name: 'page-article-item'

}

### **1.2.2. 文件名命名** Vue、Js、css、json文件，函数采用小驼峰式命名(camelCase)

### **1.2.3. v-for**

在执行v-for遍历的时候，总是应该带上key值使更新DOM时渲染效率更高。  
v-for应该避免与v-if在同一个元素上使用，因为v-for的优先级比v-if更高，为了避免无效计算和渲染，应该尽量将v-if放到容器的父元素之上。

// 反例

<ul>

<li v-for="item in list" :key="item.id" v-if="showList">

{{ item.title }}

</li>

</ul>

// 正例

<ul v-if="showList">

<li v-for="item in list" :key="item.id">

{{ item.title }}

</li>

</ul>

### **1.2.4. 指令缩写**

为了统一规范始终使用指令缩写，使用v-bind，v-on并没有什么不好，这里仅为了统一规范。v-bind用":"，v-on用"@"  
<input :value="mazeyUser" @click="verifyUser">

### 1.2.5. 对象

属性值简写

{height: height, width: width} => {height, width}

### **1.2.6. <style></style>**

为了避免样式冲突，整个项目全都使用scss语言，scoped特性  
<style lang="scss" scoped></ style>

### **1.2.7. vue文件顶级元素顺序**

为了统一和便于阅读，应该按<template>、<script>、<style>的顺序放置

### **1.2.8. var/let/const**

不再使用 var，而使用 let / const，优先使用const。

### **1.2.9. 引号**

建议不再使用双引号，静态字符串使用单引号，动态字符串使用反引号衔接。

// 反例

const foo = "bd"

const bar = foo + "，前端工程师"

// 正例

const foo = 'bd'

const bar = `${foo}，前端工程师`

### **1.2.10. <script></script>内声明顺序**

* name 常用
* components 常用
* props
* data 常用
* computed 常用
* watch 常用
* created
* mounted 常用
* activited
* update
* beforeRouteUpdate
* methods 常用
* filter

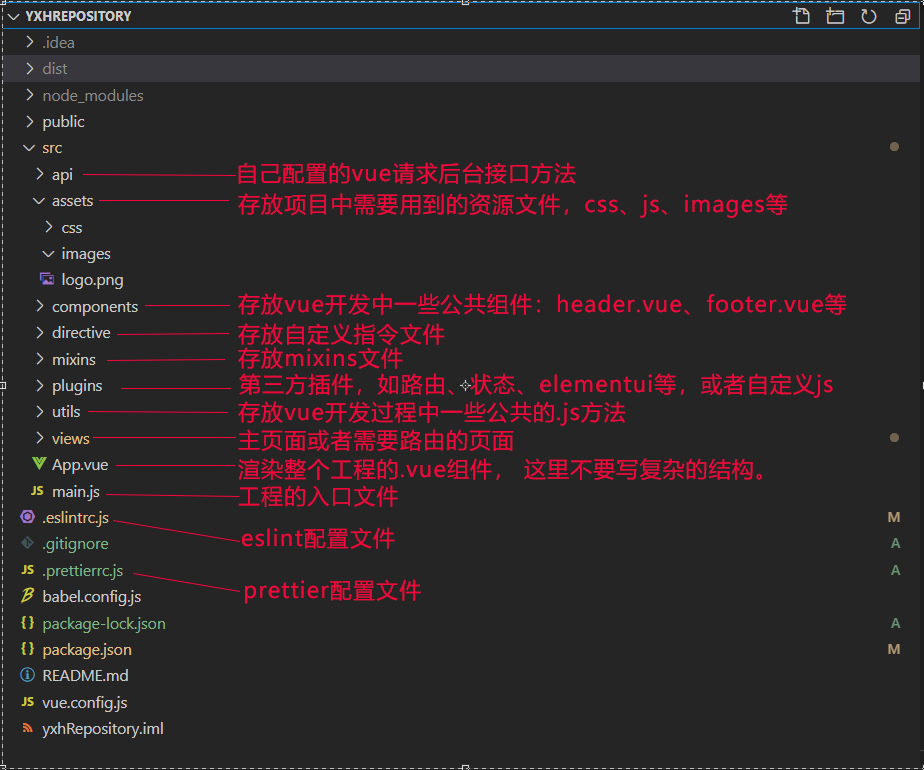
### **1.2.11. 调试信息**

由于现有工程中发现很多调试信息debugger,console使用后未及时删除，导致浏览器控制台输出大量无用信息，并还会阻塞正常流程。在eslint中将这两种调试信息设置为错误级别，提交代码时将会强制开发人员删除，如要异常抛出请使用console.error和console.warn，请不要滥用。

### **1.2.12. 统一样式处理**

* 为了统一样式，查询框及按钮、表格等统一使用默认size，表格内部按钮使用size=”mini”
* **查询按钮**统一使用primary， **“重置”按钮**使用默认+plain，工具栏按钮默认使用class=”tool-btn”。
* **分页控件**放置在右下角。

## **1.3. 目录结构规范**



## 1.4. 公共组件库

### 背景

多个项目使用同一个组件时，需要从一个项目复制粘贴至其他项目，如果后面组件更新的话，要修改多个项目，这种情况耗费时间，有时还会忘记哪个项目的组件是最新的。

代码地址： <https://gitlab-sc.fxnotary.com/UI/notary-ui>

### **1.4.1. 发布流程**

1. 修改 package.json 版本，version 加一
2. 打包 npm run lib
3. 切换到仓库 http://nexus-nc.fxnotary.com/repository/npmjs-hosted/
4. 发布 npm publish

### **1.4.2. 安装使用**

1. npm install notary-ui -S
2. 在 main.js 中 import UI from 'notary-ui'；
3. 然后再 Vue.use(UI);
4. 工具类使用 this.$utils.propertyName()；
5. 组件类使用 k-componentname

具体使用方式请参见：

https://gitlab-sc.fxnotary.com/UI/notary-ui/wikis