

VUE项目相关规范

2018-07-05

[1. VUE项目目录结构 2](#_Toc518900555)

[2. 主题配置与切换 5](#_Toc518900556)

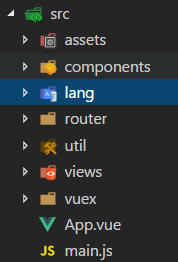
[3. 国际化配置及调用 5](#_Toc518900557)

[4. 一些通用代码命名规范 7](#_Toc518900558)

[5. VUE的一些规范 10](#_Toc518900559)

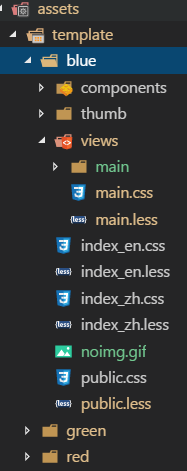
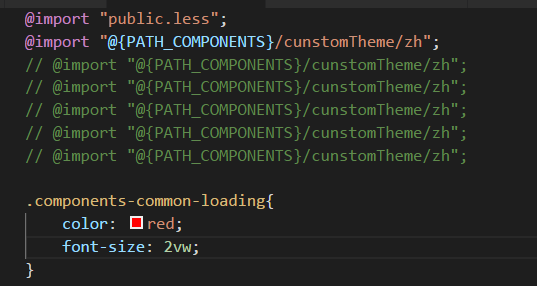
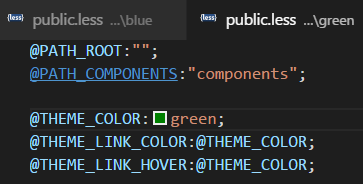
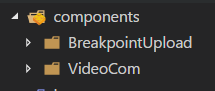
[6. 前端Mock 数据 11](#_Toc518900560)

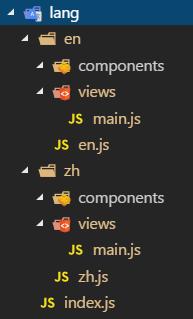
## VUE项目目录结构

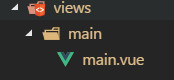
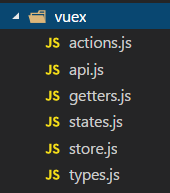


* Src 项目root目录 代码和资源统一放置在 目录下



* Assets 目录存放所有资源文件 主要就是 所有 图片 字体 样式 等，一套主题一个目录。
  + 
  + **组件的样式 按语言命名；图片则放在 img 目录下**
  + **Views** 里的资源直接按 Vies 下的二级目录名称命名
  + 每一个 Views 包含 img 和 less 两个目录 ，如果部分图片需要配合国际化则 分语言文件夹存放；
  + 全局 less 的入口文件 为 index\_语言缩写.less
  + 
  + 
  + Thumb 目录放美工出的设计图
* Components 目录存放所有抽象通用组件的VUE文件
  + \* 这个目录下的每个 VUE 组件都要满足 足够抽象 可复用的原则 复制到其他项目可直接使用 ，否则就是个Views 请放到 Views下
  + 
* Lang 存放项目的语言文件，详情见 国际化配置及调用



* Views 目录 存放所有视图vue文件，可按导航结构组织目录。
  + 
* Vuex 目录维护一些全局状态，可一些跨视图数据。
  + 

## 主题配置与切换

* 配置主题
  + 进入 src/vuex/states.js 中 ，修改 skins 对象数组 ：template 的值为 assets/ template/目录名称；color 的值为 index\_语言.less 中的 @THEME\_COLOR 保持一致
* 静态更换主题
  + 修改 target\_template 为 skins 数组索引值



## 国际化配置及调用

* 语言切换下拉菜单

公司可注册一个 npm 账号，将一些公共组件发布到npm中，方便管理和跨项目。

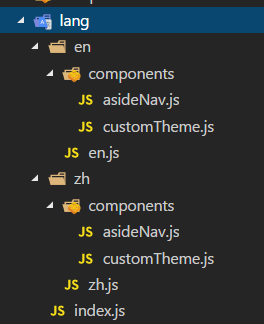
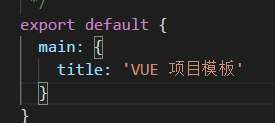
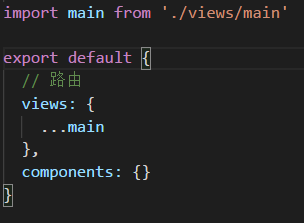
npm i vue-language-switch

组件使用说明：

<https://www.npmjs.com/package/vue-language-switch>

* 组织语言文件

lang 目录 所有 语言文件 每种语言有一个主入口文件 例如 EN/EN.JS 存放着所有子文件的引用，可直接在这里 跳转响应的子文件。这里一般有两个目录 Components 和 Views 其他科按需增加。

* + 
  + 开发人员自行添加语言文件的方法：
  + 先在相应目录下建立自己组建的 语言文件，名称为 对应的VUE文件名(一般也是组件或View名称).js 导出默认对象：
  + 
  + 在入口文件 import ， 然后把 变量 加到响应位置中，位置参考代码所在的目录结构
  + VUE 里调用
  + 

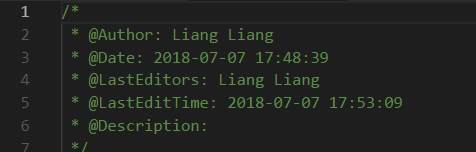
## 一些通用代码命名规范

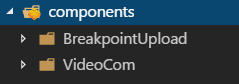
项目开启eslint语法检查，最简单的一个例子，你的个人代码或者公司的代码展示给别人看起码看起来是舒服的，是标准的，优雅的，统一的，不是混乱无章的，既可以提高你的个人形象同时也维护了公司形象。项目不是一个人在写代码，想怎么写就怎么写，自己写自己维护。项目终究需要团队协作，别人会读你的代码，你也会有可能去修改别人的代码，如果每个人都按自己的习惯写，你看别人的可能会头疼，别人一样可能会有这种感觉。所以引入一个公认的基础的代码规范，让自己逐步适应，一点点规范自己的书写习惯，对自己对他人都是一件非常有利的事。



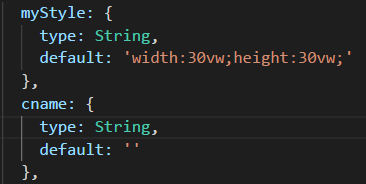
* VSCODE 设置 自动eslint格式化：
  + 1. 安装 vetur + prettier + eslint 3个plugin
  + 2. userconfig 配置文件里添加
  + // 强制单引号
  + "prettier.singleQuote": true,
  + // 尽可能控制尾随逗号的打印
  + "prettier.trailingComma": "all",
  + // 开启 eslint 支持
  + "prettier.eslintIntegration": true,
  + // 保存时自动fix
  + "eslint.autoFixOnSave": true,
  + // 添加 vue 支持
  + "eslint.validate": [
  + "javascript",
  + "javascriptreact",
  + {
  + "language": "vue",
  + "autoFix": true
  + }
  + ],
  + // 使用插件格式化 html
  + "vetur.format.defaultFormatter.html": "js-beautify-html",
  + // 格式化插件的配置
  + "vetur.format.defaultFormatterOptions": {
  + "js-beautify-html": {
  + // 属性强制折行对齐
  + "wrap\_attributes": "force-aligned",
  + }
  + }
* 命名的方法通常有以下几类：
  + Camel驼峰命名法，形如thisIsAnApple
  + Pascal帕斯卡命名法，形如ThisIsAnApple
  + 下划线命名法，形如this\_is\_an\_apple ·
  + 中划线命名法，形如this-is-an-apple
  + JS：
  + .变量名：必须使用 下划线命名法
  + .参数名：必须使用camel命名法
  + .函数名：必须使用camel命名法
  + .方法/属性：必须使用camel命名法
  + .私有（保护）成员：必须以下划线 \_ 开头
  + .常量名：必须使**用全部大写的下划线命名法**，如IS\_DEBUG\_ENABLED
  + .类名，构造函数名：必须使用pascal命名法
  + .枚举名：必须使用pascal命名法
  + .枚举的属性：必须使用全部大写的下划线命名法
  + .命名空间：必须使用camel命名法 新项目建议启动 SEDU 这个命名空间，所有公共方法都加到这个命名空间下
  + 全局变量： 全部加到命名空间下
  + boolean类型的应当使用is、has等起头,表示其类型
  + CSS
  + 选择器必须单词全字母小写,多个单词情况下使用中划线分割
  + class选择器必须代表相应模块或部件的内容或功能，不得以样式信息进行命名
  + id 选择器必须保证项目唯一 ，命名加目录结构前缀
  + 同一页面，应避免使用相同的 name 与 id VUE项目推荐 ref属性

## VUE的一些规范

* VUE 组件：
  + 文件顶部加入文档注释
  + 
  + 每个组件一个目录，命名 帕斯卡方式，每个组件要包含一个README.md，把说明加进去



* + 统一加入 style class 接口，方便个性化局部组件。 cname – 自定义class， myStyle – 行内 style。

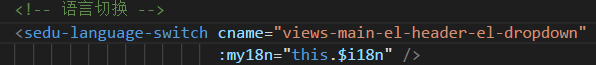


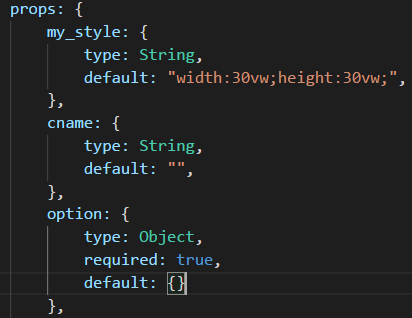
组件内的 <style> 标签删除！！！ 统一在 template目录下书写样式，如果需要默认样式，则把组件默认样式写入canme的default中。

Views 文件中调用时 公司内部组件import 时统一加入 sedu 前缀， 驼峰 命名

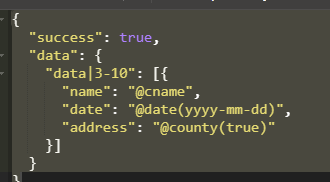


Template 里使用的标签 就统一为 <sdue-xxxxx-xxxxx>



* VUE 参数 Props 方法注意点
  + 使用 data 里的变量时请先在 data 里面初始化；
  + props 指定类型，type ，如果是必须的设置required: true, 设置 默认值 default
  + 能够根据业务规则合并自动化路由和通知
  + 

## 前端Mock 数据

* 优势：
  + 前后端分离，团队并行开发，加快效率，避免前端等后端的情况发生吗，降低前后端耦合。
  + 可以无需后端代码就可做 系统演示
  + 一些简单的测试用例非常方便实现
* 工具： Easy-mock <https://easy-mock.com/>
  + 新建一个项目
  + 
  + 编写接口和响应的返回数据
  + 
  + 点 进入接口数据生成规则编辑界面，支持 mockjs 语法自动产生返回json数据
  + 
  + 团队开发 ，进入项目的设置界面，可以把其他项目组员的ID加到 邀请成员 里，就可以一些参与接口的编辑了。
  + 