**前端开发规范文档**

**前言**

适当的规范和标准不是为了消灭小姐姐小哥哥们代码内容的创造性、优雅性，而是限制过度个性化，以一种普遍认可的统一方式一起做事，提升协作效率。所谓前人栽树后人乘凉，提高我们代码的可读性和可维护性，是我们共同努力的目标。(以下涉及的规范可参考亩均模块化产品项目一起享用)

**手册**

1. **项目技术栈**

电脑端：vue + element-ui + sass/less ；

移动端（钉钉/微信）：vue + vant/mint ui + sass/less ；

微信小程序：uni-app + uni ui /uni衍生的插件市场 + sass/less ；

项目涉及的可视化模块：地图部分使用高德地图，图表部分使用ECharts可视化库。

1. **前端编码规范**
2. **文件规范**
3. **文件存放位置规范**：
4. **一般项目的目录结构：**

src

* assets 存放静态图片
* common 存放公共方法，接口，样式和变量等

1. icons 存放icon样式库 ( iconfont/svg等 )
2. styles 存放公共样式
3. api 存放公共接口
4. utils 存放公用方法
5. constant 存放公共的变量

* components 存放公用组件
* directive 存放全局的指令
* filters 存放全局过滤器
* layout 全局布局的组件
* router 存放全局的路由列表
* store 存储应用全局的所有组件状态
* views 所有的功能组件
* vue.config.js vue-cli项目配置文件

1. **各目录存放规则：遵循语义化规范**
2. utils方法文件：

方法按照处理类型分类；

比如validate表示存放所有验证的方法，time处理所有涉及时间的方法，以此类推；

所有的方法文件都在constant/index.js中引入并导出；

调用时通过@utils调用即可。

1. constant变量文件：

按照常规和非常规常量进行分类；

常规的都放在commonConstant；

非常规可新建文件，比如地图相关的所有放mapConstant,图表相关放echartsConstant；

所有的变量文件都在utils/index.js中引入并导出；

调用时通过@constant调用即可。

1. icons图标库:

iconfont引入阿里图标库；

svg是项目的svg图标存放位置（具体使用可查看components中SvgIcon组件）

1. styles样式文件：

一般使用sass语言；

若要使用其他语言可在该文件下建文件夹，比如新建less文件夹，内部放置less文件。

1. api接口文件：

放置所有的接口文件；

按照模块名新建文件夹，存放该模块下的所有子模块接口文件；

接口文件命名规则遵循跟views的文件名一致，方便后人查看翻阅；

调用规则遵循 @api/模块名/子模块名。

1. router路由文件:

所有功能的路由文件都在router内；

在router/modules/下新建模块的路由文件；

文件命名规则遵循跟views的文件名一致；

所有的路由文件都在router/index.js中引入并导出为路由对象；

其中constantRoutes为固定路由，asyncRoutes为动态路由。

1. **文件命名规则：在vue官方规范基础上结合大家的编码习惯独家定制**
2. **Component**

所有的Component文件都是以大写开头 (PascalCase)，包括views下的Component文件，但除了index.vue。

例如：

* @/components/BackToTop/index.vue
* @/components/Charts/Line.vue
* @/views/example/components/ButtonItem.vue

1. **Js文件**

所有的.js文件遵循驼峰命名(kebabCase)。

例如：

* @utils/handleData.js
* @/views/svgIcons/requireIcons.js
* @/components/MarkdownEditor/defaultOptions.js

1. **Views文件**

views下的**文件夹和单组件文件**都遵循驼峰命名(kebabCase)

例如：

* @/views/organization/logManage/index.vue
* @/views/entFile
* @/views/entFile/search/entDetail.vue

1. **新建模块涉及的一般操作（老筒子们请自动忽略）**
2. **要新建一个企业管理模块**：
3. 在 @/views 新建模块文件夹： entManage
4. 在 @/router/module下新建路由文件 ：entManage.js
5. 在@/common/api下新建接口文件：

若企业管理只有一个模块则在 @/common/api下新建接口文件 : api/ entManage /index.js；

如若企业管理有多个子模块，则建立和views文件名相同的js文件接口：api/entManage/entFileSet.js和api/entManage/entLibrary.js。

1. **在views文件或其他地方要调用方法，接口等的方式**

在vue.config.js文件的alias中已经有如下配置项

alias: {

'@': resolve('src'),

'@api': resolve('src/common/api'),

'@utils': resolve('src/common/utils'),

'@styles': resolve('src/common/styles'),

'@icons': resolve('src/common/icons'),

'@components': resolve('src/common/components'),

'@constant': resolve('src/common/constant'),

}

所以在调用时只需使用@XX的形式调用即可，当然你也可以根据自己需要自行添加，开心就好。

例如：

* @import "~@styles/variables.scss";
* import { typeList } from '@constant';
* import {notEmpty,notSelect} from '@utils';
* import {findById} from '@api/organition/user.js;

1. **一般代码规范**
2. **文本缩进：**一次缩进两个空格
3. **对象命名规范**

**常量**：采用全大写并且单词以\_分割(KEBAB\_CASE)

**函数名**：

1. 类（构造函数）：以大写开头的驼峰命名格式 (PascalCase)。
2. 函数：首字母小写，驼峰命名（kebabCase）。

普通函数：使用 (动词 / 动词+名词形式)。

内部函数：使用\_+动词+名词，内部函数需在函数最后定义。

例如:涉及返回逻辑的函数用is,has等，fnHasClass()，fnIsString;

或者内部函数 \_fnAddSome()。

函数方法常见的动词：

get 获取/set 设置,

add 增加/remove 删除

create 创建/destory 移除

start 启动/stop 停止

open 打开/close 关闭,

load 载入/save 保存,

create 创建/destroy 销毁

download 下载/upload 上传,

refresh 刷新/synchronize 同步

update 更新/revert 复原,

submit 提交/commit 交付

**变量名**：首字母小写，驼峰命名(kebabCase)，匈牙利命名。

匈牙利命名法语法：变量名＝变量类型＋对象描述。例如：

起名类型 命名前缀 举例

Array 数组 a aList， aGroup

Boolean 逻辑 b bChecked，bHasLogin

Function 函数 f fGetHtml，fInit

Integer 数字 n nPage，nTotal

Object 对象 o oButton，oDate

Regular E正则 r rDomain，rEmail

String 字符 s sName，sHtml

如此遵循语义化的规则。

1. **代码注释**

原则上：如无必要，勿增注释，如有必要，尽量详尽。以此提高代码可读性和可维护性。注释分为单行注释：// 和多行注释：/\*\*/。

1. **顶部文档注释：**

/\*\*

\* @description: 中文说明

\* @author: name

\* @update: name(xxxx-xx-xx)

\*/

1. **JS函数注释：**

/\*\*

\* @func

\* @todo 这个函数需要优化

\* @desc 一个带参数的函数

\* @param {string} a - 参数a

\* @param {number} b=1 - 参数b默认值为1

\* @param {string} c=1 - 参数c字符串

\* @param {object} d - 参数d为一个对象

\* @param {string} d.e - 参数d的e属性

\* @param {string} d.f - 参数d的f属性

\* @param {object[]/Array} g - 参数g为一个对象数组

\* @param {string} g.h - 参数g数组中一项的h属性

\* @returns {boolean} 返回值为true

\*/

参考文档：[**JSDoc Guide**](http://yuri4ever.github.io/jsdoc/)

1. **代码检查：**
2. 建议VSCode配置ESlint插件：每次保存，VSCode就能标红不符合 eslint 规则的地方，同时还会做一些简单的自我修正。另外运行npm run lint -- --fix，eslint会自动修复一些简单错误。
3. 一般所有的配置项都存放在 [.eslintrc.js](https://github.com/PanJiaChen/vue-element-admin/blob/master/.eslintrc.js) 中，如果觉得配置过于严格可自行修改内部配置；原则上为了减少代码不必要的错误使用最严格的plugin:vue/recommended来校验代码。
4. 配置步骤：
5. VSCode扩展插件中搜索安装eslint插件；
6. 在根目录配置[.eslintrc.js](https://github.com/PanJiaChen/vue-element-admin/blob/master/.eslintrc.js)文件；
7. VSCode 进行扩展设置，依次点击 文件 > 首选项 > 设置

Settings.json添加配置项如下：

{

"files.autoSave": "off",

"eslint.validate": [

"javascript",

"javascriptreact",

"vue-html",

{

"language": "vue",

"autoFix": true

}

],

"eslint.run": "onSave",

"eslint.autoFixOnSave": true

}

参考文档：[ESlint配置](https://panjiachen.github.io/vue-element-admin-site/zh/guide/advanced/eslint.html#%E9%85%8D%E7%BD%AE%E9%A1%B9)

若ESlint经常出现error/warn请查看：[目前主流代码规范](https://github.com/BingKui/javascript-zh)

1. **Git代码提交规范**

每次提交，Commit message 都包括三个部分：Header，Body 和 Footer。其中，Header 是必需的，Body 和 Footer 可以省略。而Header部分只有一行，包括两个字段：type（必需）和subject（必需）。type用于说明 commit 的类别，只允许使用下面9个标识，subject是 commit 目的的简短描述，不超过50个字符。type如下：

feat: 新功能（feature）

fix: 修补bug

docs: 文档（documentation）

style: 格式（不影响代码运行的变动）

refactor: 重构（即不是新增功能，也不是修改bug的代码变动）

chore: 构建过程或辅助工具的变动

revert: 撤销，版本回退

perf: 性能优化

test：测试

improvement: 改进

build: 打包

ci: 持续集成