# 前端开发规范手册【VUE版本】

项目目录结构

├── build # 构建相关

├── mock # 项目mock 模拟数据

├── plop-templates # 基本模板

├── public # 静态资源

│ │── favicon.ico # favicon图标

│ └── index.html # html模板

├── src # 源代码

│ ├── api # 公共类请求

│ ├── assets # 主题 字体等静态资源

│ ├── components # 全局公用组件

│ ├── directive # 全局指令

│ ├── filters # 全局 filter

│ ├── icons # 项目所有 svg icons

│ ├── lang # 国际化 language

│ ├── layout # 全局 layout

│ ├── router # 路由

│ ├── store # 全局 store管理

│ ├── styles # 全局样式

│ ├── utils # 全局公用方法

│ ├── vendor # 公用vendor

│ ├── views # views 所有页面

│ ├── appSystem # 业务大类文件夹

│ ├── AppVue # 业务模块文件夹

│ ├── api.js # 业务类子模块请求文件

│ ├── component # 业务类子模块组件文件夹

│ └── index.vue # 业务模块主界面

│ ├── App.vue # 入口页面

│ ├── main.js # 入口文件 加载组件 初始化等

│ └── permission.js # 权限管理

├── tests # 测试

├── .env.xxx # 环境变量配置

├── .eslintrc.js # eslint 配置项

├── .babelrc # babel-loader 配置

├── .travis.yml # 自动化CI配置

├── vue.config.js # vue-cli 配置

├── postcss.config.js # postcss 配置

└── package.json # package.json

### 一、编码规范

#### 1、样式文件命名说明

（1）重置样式：reset.stylus

（2）基础布局样式：base.stylus

（3）公共样式：public.stylus

（4）变量样式：variable.stylus

（5）icon样式：icon.stylus

（6）混合类型(函数样式)：mixin.stylus

#### 2、样式规范化

（1）连字符CSS选择器命名规范

长名称或词组可以使用“中横线”来为选择器命名。 不建议使用“\_”下划线来命名CSS选择器，例：class=”page page-index”。

（2）使用16进制表示颜色值，除非表示的是透明度，例：#F62AB5

（3）以下常用的css命名规则；

page-hd、page-bd、page-ft、tit、cnt、desc、info、left-bar、search-bar

#### 3、文件目录

（1）存放其它图片文件夹命名规范：imgaes

（2）存放广告图的文件夹banner

（3）文件名应该全部小写，多个单词以下划线“\_”分开

（4）JS文件：驼峰命名方式，首字母小写，例：getUserInfo.js

（5）确保文档或模板中只包含html，把用到的样式都写到样式表文件中，把脚本都写到js文件中，采用外链引入形式，使页面的结构与行为分离。

#### 4、 Javascript

（1）因vue脚手架有自动编译的功能，为了代码的整洁，每行js代码不必带“；”作为结束，但需遵循一个语句一行的编码方式；

#### 5、命名规范

（1）文件命名：以英文命名，后缀为.js，例：（共用）common.js，其他命名可根据模块需求命名；

（2）变量命名：驼峰式命名，原生JavaScript变量要求是纯英文字母， 首字母须小写，变量集中声明， 避免全局变量  
（3）类命名：首字母大写， 驼峰式命名。eg：StudentInfo、UserInfo、ProductInfo；

（4）函数命名：首字母小写驼峰式命名。eg：getUserInfo；

（5）命名语义化，尽可能利用英文单词或其缩写。

（6）常量：必须采用全大写的命名，且单词以\_分割，常量通常用于ajax请求url，和一些不会改变的数据；

命名规范：使用大写字母和下划线来组合命名，下划线用以分割单词。

例：const MAX\_LENGTH = 20;const URL = 'http://www.star.com'

（7）变量：必须采用小驼峰式命名法，例：let maxCount = 10;

#### 6、项目命名

全部采用小写方式， 以下划线分隔。例：my\_project\_name

#### 7、函数

命名方法：小驼峰式命名法；

命名规范：前缀应当为动词；

命名建议：可使用常见动词约定。

|  |  |  |
| --- | --- | --- |
| 动词 | 含义 | 返回值 |
| can | 判断是否可执行某个动作(权限) | 函数返回一个布尔值。true：可执行；false：不可执行 |
| has | 判断是否含有某个值 | 函数返回一个布尔值。true：含有此值；false：不含有此值 |
| is | 判断是否为某个值 | 函数返回一个布尔值。true：为某个值；false：不为某个值 |
| get | 获取某个值 | 函数返回一个非布尔值 |
| set | 设置某个值 | 无返回值、返回是否设置成功或者返回链式对象 |
| load | 加载某些数据 | 无返回值或者返回是否加载完成的结果 |

#### 8、类 & 构造函数

命名方法：大驼峰式命名法，首字母大写；例：Person

命名规范：前缀为名称。

示例：

class Person {

  public name: string;

  constructor(name) {

    this.name = name;

  }

}

const person = new Person('mevyn');

#### 9、类的成员

类的成员包含：

公共属性和方法：跟变量和函数的命名一样。

私有属性和方法：前缀为\_(下划线)，后面跟公共属性和方法一样的命名方式。

示例：

class Person {

  private \_name: string;

  constructor() { }

  // 公共方法

  getName() {

    return this.\_name;

  }

  // 公共方法

  setName(name) {

    this.\_name = name;

  }

}

const person = new Person();

person.setName('star');

person.getName(); // ->'star'

#### 10、注释规范

js 支持三种不同类型的注释：行内注释、单行注释和多行注释：

行内注释

说明：行内注释以两个斜线开始，以行尾结束。

语法：code // 这是行内注释

使用方式：//(双斜线)与代码之间保留一个空格，并且//(双斜线)与注释文字之间保留一个空格。

命名建议：

// 用来显示一个解释的评论  
// -> 用来显示表达式的结果，  
// >用来显示 console 的输出结果，

示例：

function test() { // 测试函数

  console.log('Hello World!'); // >Hello World!

  return 3 + 2; // ->5

}

单行注释

说明：单行注释以两个斜线开始，以行尾结束。

语法：// 这是单行注释

使用方式：单独一行：//(双斜线)与注释文字之间保留一个空格。

示例：

// 调用了一个函数；1)单独在一行setTitle();

多行注释

说明：以 /\* 开头， \*/ 结尾

语法：/\* 注释说明 \*/

使用方法：若开始/和结束/都在一行，推荐采用单行注释。若至少三行注释时，第一行为/，最后行为/，其他行以开始，并且注释文字与保留一个空格。

示例:

/\*\* 代码执行到这里后会调用setTitle()函数\* setTitle()：设置title的值

\*/

setTitle();

函数(方法)注释

说明：函数(方法)注释也是多行注释的一种，但是包含了特殊的注释要求，参照JSDoc

语法：

/\*\* \* 函数说明 \* @关键字

\*/

常用注释关键字：(只列出一部分，并不是全部)

|  |  |  |  |
| --- | --- | --- | --- |
| 注释名 | 语法 | 含义 | 示例 |
| @param | @param 参数名 {参数类型} 描述信息 | 描述参数的信息 | @param name {String} 传入名称 |
| @return | @return {返回类型} 描述信息 | 描述返回值的信息 | @return {Boolean} true:可执行;false:不可执行 |
| @author | @author 作者信息 [附属信息：如邮箱、日期] | 描述此函数作者的信息 | @author 张三 2015/07/21 |
| @version | @version XX.XX.XX | 描述此函数的版本号 | @version 1.0.3 |
| @example | @example 示例代码 | 演示函数的使用 | @example setTitle(‘测试’) |

/\*\*\* 合并Grid的行

\* @param grid {Ext.Grid.Panel} 需要合并的Grid

\* @param cols {Array} 需要合并列的Index(序号)数组；从0开始计数，序号也包含。

\* @param isAllSome {Boolean} ：是否2个tr的cols必须完成一样才能进行合并。true：完成一样；false(默认)：不完全一样

\* @return void

\* @author polk6 2015/07/21

\* @example

\* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_                             \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\* |  年龄 |  姓名 |                              |  年龄 |  姓名  |

\* -----------------      mergeCells(grid,[0])  -----------------

\* |  18   |  张三 |              =>             |        |  张三  |

\* -----------------                              -  18   ---------

\* |  18   |  王五 |                              |        |  王五  |

\* -----------------                              ----------------------

\*/

function mergeCells(grid: Ext.Grid.Panel, cols: Number[], isAllSome: boolean = false) {

  // Do Something

}

#### 11、手机设计稿规范

设计稿的宽度一般情况下为宽度1920px,在PC端页面编写的过程，将页面的最大宽度设置为1200px，最小宽度为1200px，宽度为100%；即min-width:1200px；max-width:1920px；width100%；这样一来的话宽度在1200到1920之间是自适应的，当页面小于1200时出现水平滚动条，当页面大于1920时页面居中布局。