**前端开发规范**

**序言**

程序员生涯中，Coding时期应该是比较简单和快乐的时间。优雅的代码就像诗句，精简却不失风味。多人团队开发中，不免出现代码风格迥异的现象。这种现象非常不利于日后代码维护，也容易出现模块交叉开发时，开发效率过低的情况。由此开发规范显得异常重要，此篇规范主要解决以下问题：

1. 提高团队人员的项目开发效率
2. 规范项目代码风格，增强代码可读性
3. 提高项目日后维护的灵活性
4. 降低Bug出现率

**目录**

注释篇 ·················································· 3

命名篇 ·················································· 5

项目结构篇 ··················································

HTML篇 ··················································

JS篇 ··················································

CSS篇 ··················································

模块篇 ··················································

组件篇 ··················································

插件篇 ··················································

**注释**

1. 单行注释（//）

单独一行，双斜杠与注释文字之间保留一个空格；

代码后注释，代码与双斜杠之间保留一个空格，双斜杠与注释文字之间保留一个空格；

注释代码，代码与双斜杠之间保留一个空格。

示例：

// 页面初始化

this.init()

1. 多行注释（/\* 注释说明 \*/）

多行注释至少为3行，以/\*开头，以\*/结尾，并且注释文字或代码与\*之间保留一个空格

示例：

/\*

\* 清空userData的数据

\*/

this.userData = {}

1. 函数、方法注释

当函数、方法中带有形参和需要返回值时，需要进行参数和返回值进行说明，此时就需要采用多行注释，格式略区别于普通多行注释

示例：

/\*

\* 将用户信息存储至sessionStorage中

\* @key {String} 存储的信息名称

\* @val {Object} 存储的信息值

\* @return {Boolean} 存储是否成功

\*/

setSessionStorage (key, val) {

……

}

1. 注释区域

不易理解的Data、Props、Computed数据变量；

Methods、Filters、Directive中的函数方法；

注册的自定义组件；

引入的第三方和自定义的插件；

更改第三方组件、插件的源码逻辑区域

Vue模板及HTML中的不易理解的代码块；

Vuex中定义的状态以及更改状态的方法；

复杂的业务逻辑代码；

封装的模块代码。

**命名**

1. 驼峰命名法

大写驼峰命名，首字母大写，后单词首字母亦大写。

小写驼峰命名，首字母小写，后单词首字母采用大写。

示例：

AddressDialog.vue

this.setDataUser()、this.userData

1. 文件命名

组件文件采用大写驼峰命名法；

其它文件统一采用小写命名，单词之间使用-（中横杠）分割；

引用资源时采用相对资源路径。

示例：

DoctorDialog.vue

store-methods.js

\_var.scss

../../components/dialog/Address.vue

1. 变量、常量命名

变量采用小写驼峰命名法；

常量采用全大写命名，单词之间使用\_（下划线）分割。

示例：

showInfo、showInfo.saveLoading

SET\_USER\_DATA

1. 函数、类命名

普通函数采用小写驼峰命名；

构造函数、类采用大写驼峰命名。

示例：

getUserData () { …… }

Person () { …… }

1. 样式命名

Class统一使用小写命名，单词之间使用-（中横杠）分割；

Sass中使用的函数和变量采用变量、函数命名方式。

示例：

common.scss、form-info.scss

**项目结构篇**

1. 基础介绍

build -> webpack配置目录，可对开发环境文件以及线上打包文件进行配置处理，对依赖引入进行配置

config -> 文件配置目录，可对开发、生产环境进行变量配置，以此达到多环境开发打包目的

src -> 项目主要开发目录，包含业务逻辑代码,

dist -> 项目打包后文件，最终放至服务器上解析

static -> 静态文件目录，不参与项目打包