前

端

组

代

码

规

范

文

档

V1.0.0

目录

[1. 文档概述 1](#_Toc25169)

[2. 命名规范 1](#_Toc6945)

[2.1 命名严谨性 1](#_Toc31668)

[2.2 项目命名 1](#_Toc4582)

[2.3 目录命名 2](#_Toc23741)

[2.4 文件命名 2](#_Toc29794)

[3. 注释规范 2](#_Toc9130)

[3.1 HTML注释 2](#_Toc22297)

[3.2 CSS 注释 3](#_Toc19751)

[3.3 Javascript 注释 3](#_Toc32095)

[4. git规范 3](#_Toc11074)

[4.1 分支管理 3](#_Toc17639)

[4.2 git commit 提交 3](#_Toc12604)

[5. 图片规范 4](#_Toc5858)

[5.1 图片引入 4](#_Toc8874)

[5.2 图片格式 4](#_Toc21270)

[5.3 图片大小 4](#_Toc26810)

[5.4 图片质量 4](#_Toc3208)

[6. HTML规范 4](#_Toc9890)

[6.1 元素及标签闭合 5](#_Toc27831)

[6.2 书写风格 5](#_Toc11903)

[6.3 元素属性 5](#_Toc31101)

[7. CSS规范 5](#_Toc16943)

[7.1 代码格式化 5](#_Toc6918)

[7.2 选择器 5](#_Toc10215)

[8. JavaScript规范 5](#_Toc27271)

[8.1变量命名 5](#_Toc6292)

[8.2 代码格式 7](#_Toc22419)

[8.3 字符串 8](#_Toc32340)

[8.4 对象声明 8](#_Toc24902)

[8.5 使用 ES6+ 8](#_Toc27518)

[8.6 条件判断和循环最多三层 8](#_Toc18847)

[8.7 this 的转换命名 8](#_Toc14094)

[8.8 慎用 console.log 8](#_Toc7354)

[9. Vue规范 8](#_Toc20863)

[9.1 组件规范 8](#_Toc29385)

[9.2 Vue 项目目录规范 10](#_Toc15694)

[9.3 其他 11](#_Toc19205)

[10.代码规范 12](#_Toc12477)

[10.1 安装 VS Code 插件 12](#_Toc3818)

[10.2 配置 VS Code 12](#_Toc21141)

[10.3 修改Eslint的配置文件 12](#_Toc2162)

前端代码规范文档

1. 文档概述

规范的代码可以促进团队合作、降低维护成本、有助于 code review（代码审查）、养成代码规范的习惯，有助于程序员自身的成长

当团队的成员都严格按照代码规范来写代码时，可以保证每个人的代码看起来都像是一个人写的，看别人的代码就像是在看自己的代码（代码一致性），阅读起来更加顺畅。更重要的是我们能够认识到规范的重要性，并坚持规范的开发习惯。

详细见在线文档：[前端技术文档 (zs-fet-team.github.io)](https://zs-fet-team.github.io/ZS-FET-team/)

2. 命名规范

2.1 命名严谨性

正确命名方式

英文拼写

驼峰命名方法

遵循易于让阅读者理解的英文 （有道词典可以查询到为依据）

严禁出现命名方式

使用拼音与英文混合的方式

直接使用中文的方式

纯拼音命名方式

不符合规范的缩写方式

2.2 项目命名

驼峰命名方法

推荐：

PetroleumMiniProgram

避免：

petroleum\_mini\_program

petroleum-mini-program

2.3 目录命名

驼峰命名方法

有复数结构时，要采用复数命名法

缩写不用复数。

正例：

docs

demoDocs

反例：

demo-docs

Personal-center

demo\_docs

2.4 文件命名

驼峰命名方法

正例：

company-logo.png

demo-docs.html

Personal-center.vue

反例：

companyLogo.png

Company\_logo.png

3. 注释规范

3.1 HTML注释

模块注释

一般用于描述模块的名称以及模块开始与结束的位置

单行注释

一般用于简单的描述，如某些状态描述、属性描述等

嵌套注释

当模块注释内再出现模块注释的时候，注释写在模块结尾标签底部，单独一行。

3.2 CSS 注释

CSS注释不能嵌套

模块注释

单行注释

3.3 Javascript 注释

4. git规范

4.1 分支管理

master 为线上分支也可称为生产分支：不能在 master 分支上写代码。

dev 分支为开发汇总分支 ：个人不能在分支上写代码

name 为个人开发分支：需要个人从 dev 分支上切出分支，分支名需要以个人名命名缩写

feature为开发新功能分支：当团队成员开发新功能时，需要从 dev 上拉一个 feature-功能名称-开发姓名 分支进行开发，例如：feature-login-zst。开发完成后需要合并回 dev 分支。

部署：当 dev 分支通过测试后，就可以合并到 master 进行发布了。

4.2 git commit 提交

git 在每次提交时，都需要填写 commit message。

git commit -m '类型type ：描述subject'

git commit -m 'feat ：增加头部导航功能'

type commit 的类型：

feat: 新功能、新特性

fix: 修改 bug

perf: 更改代码，以提高性能（在不影响代码内部行为的前提下，对程序性能进行优化）

refactor: 代码重构（重构，在不影响代码内部行为、功能下的代码修改）

docs: 文档修改

subject commit 的描述：

commit message 的概述

1. 图片规范
   1. 图片引入

HTML 中图片引入不需添加 width、height 属性，alt 属性应该写上

* 1. 图片格式

内容图

内容图多以商品图等照片类图片形式存在，颜色较为丰富，文件体积较大

优先考虑 JPEG 格式，条件允许的话优先考虑 WebP 格式

尽量不使用 PNG 格式，PNG8 色位太低，PNG24 压缩率低，文件体积大

背景图

背景图多为图标等颜色比较简单、文件体积不大、起修饰作用的图片

PNG 与 GIF 格式，优先考虑使用 PNG 格式,PNG 格式允许更多的颜色并提供更好的压缩率

图像颜色比较简单的，如纯色块线条图标，优先考虑使用 PNG8 格式，避免不使用 JPEG 格式

图像颜色丰富而且图片文件不太大的（40KB 以下）或有半透明效果的优先考虑 PNG24 格式

图像颜色丰富而且文件比较大的（40KB - 200KB）优先考虑 JPEG 格式

条件允许的，优先考虑 WebP 代替 PNG 和 JPEG 格式

* 1. 图片大小

PC 平台单张的图片的大小不应大于 200KB。

移动平台单张的图片的大小不应大于 100KB。

* 1. 图片质量

蓝湖下载 Web @2X 切图

图片不应该出现肉眼可感知的失真区域

1. HTML规范
   1. 元素及标签闭合

所有具有开始标签和结束标签的元素都要写上起止标签，某些允许省略开始标签或和束标签的元素亦都要写上。

* 1. 书写风格

HTML 标签名、类名、标签属性和大部分属性值统一用小写

* 1. 元素属性

元素属性值使用双引号语法 元素属性值可以写上的都写上

1. CSS规范
   1. 代码格式化

统一使用展开格式书写样式

* 1. 选择器

尽量少用通用选择器

UI 组件选择器需注明修改情况

不使用 ID 选择器

不使用无具体语义定义的标签选择器

1. JavaScript规范

8.1变量命名

代码中的命名为驼峰式命名（constiableName）

方法名、参数名、成员变量、局部变量都遵从驼峰形式

其中 method 方法命名必须是 动词 或者 动词+名词 形式

增删查改，add / update / delete / detail / get 5 个单词，不得使用其他

8.1.1 函数方法常用的动词

get 获取/set 设置,

add 增加/remove 删除,

create 创建/destory 销毁,

start 启动/stop 停止,

open 打开/close 关闭,

read 读取/write 写入,

load 载入/save 保存,

begin 开始/end 结束,

backup 备份/restore 恢复,

import 导入/export 导出,

split 分割/merge 合并,

inject 注入/extract 提取,

attach 附着/detach 脱离,

bind 绑定/separate 分离,

view 查看/browse 浏览,

edit 编辑/modify 修改,

select 选取/mark 标记,

copy 复制/paste 粘贴,

undo 撤销/redo 重做,

insert 插入/delete 移除,

add 加入/append 添加,

clean 清理/clear 清除,

index 索引/sort 排序,

find 查找/search 搜索,

increase 增加/decrease 减少,

play 播放/pause 暂停,

launch 启动/run 运行,

compile 编译/execute 执行,

debug 调试/trace 跟踪,

observe 观察/listen 监听,

build 构建/publish 发布,

input 输入/output 输出,

encode 编码/decode 解码,

encrypt 加密/decrypt 解密,

compress 压缩/decompress 解压缩,

pack 打包/unpack 解包,

parse 解析/emit 生成,

connect 连接/disconnect 断开,

send 发送/receive 接收,

download 下载/upload 上传,

refresh 刷新/synchronize 同步,

update 更新/revert 复原,

lock 锁定/unlock 解锁,

check out 签出/check in 签入,

submit 提交/commit 交付,

push 推/pull 拉,

expand 展开/collapse 折叠,

enter 进入/exit 退出,

abort 放弃/quit 离开,

obsolete 废弃/depreciate 废旧,

collect 收集/aggregate 聚集

8.1.2 常量命名全部大写，单词间用下划线隔开，力求语义表达完整清楚， 不要嫌名字长

推荐：

MAX\_STOCK\_COUNT

不推荐：

MAX\_COUNT

8.2 代码格式

使用 2 个空格进行缩进

不同逻辑、不同语义、不同业务的代码之间插入一个空行分隔开来以 提升可读性

说明：任何情形，没有必要插入多个空行进行隔开。

8.3 字符串

统一使用单引号(‘)，不使用双引号(“)。这在创建 HTML 字符串非常有好处

8.4 对象声明

使用字面值创建对象

使用字面量来代替对象构造器

8.5 使用 ES6+

必须优先使用 ES6 + 中新增的语法糖和函数。

比如：箭头函数、await/async 、解构、let、for...of 等等

8.6 条件判断和循环最多三层

条件判断能使用三元运算符和逻辑运算符解决的，就不用使用条件判断

8.7 this 的转换命名

对上下文 this 的引用只能使用"that"来命名

8.8 慎用 console.log

console.log 使用完记得要删除

1. Vue规范

9.1 组件规范

9.1.1组件名为多个单词。

组件名应该始终是多个单词组成（大于等于 2），且命名规范为 KebabCase 格式。

这样做可以避免跟现有的以及未来的 HTML 元素相冲突，因为所有的 HTML 元素名称都是单个单词的。

9.1.2基础组件文件名

基础组件文件名为 base 开头，使用完整单词而不是缩写。

9.1.3组件的 data 必须是一个函数

当在组件中使用 data 属性的时候（除了 new Vue 外的任何地方），它的值必须是返回一个对象的函数。

因为如果直接是一个对象的话，子组件之间的属性值会互相影响

9.1.4 Prop 定义应该尽量详细

必须使用 camelCase 驼峰命名

必须指定类型

必须加上注释，表明其含义

必须加上 required 或者 default，两者二选其一

如果有业务需要，必须加上 validator 验证

9.1.5 为组件样式设置作用域

9.1.6 如果特性元素较多，应该主动换行。

9.1.7 模板中使用简单的表达式

组件模板应该只包含简单的表达式，复杂的表达式则应该重构为计算属性或方法。复杂表达式会让你的模板变得不那么声明式。我们应该尽量描述应该出现的是什么，而非如何计算那个值。而且计算属性和方法使得代码可以重用。

9.1.8 指令都使用缩写形式

指令推荐都使用缩写形式，(用 : 表示 v-bind: 、用 @ 表示 v-on: 和用 # 表示 v-slot:)

9.1.9 标签顺序保持一致

单文件组件应该总是让标签顺序保持为

9.1.10 必须为 v-for 设置键值 key

9.1.11 v-show 与 v-if 选择

如果运行时，需要非常频繁地切换，使用 v-show ；如果在运行时，条件很少改变，使用 v-if。

9.1.12 Vue Router 规范

9.1.13 页面跳转数据传递使用路由参数

页面跳转，例如 A 页面跳转到 B 页面，需要将 A 页面的数据传递到 B 页面，推荐使用 路由参数进行传参，而不是将需要传递的数据保存 vuex，然后在 B 页面取出 vuex 的数据，因为如果在 B 页面刷新会导致 vuex 数据丢失，导致 B 页面无法正常显示数据。

9.1.14使用路由懒加载（延迟加载）机制

9.1.15 router 中的命名规范

path、childrenPoints 命名规范采用 kebab-case 命名规范（尽量 vue 文件的目录结构保持一致，因为目录、文件名都是 kebab-case，这样很方便找到对应的文件）

name 命名规范采用 KebabCase 命名规范且和 component 组件名保持一致！（因为要保持 keep-alive 特性，keep-alive 按照 component 的 name 进行缓存，所以两者必须高度保持一致）

9.1.16 router 中的 path 命名规范

path 除了采用 kebab-case 命名规范以外，必须以 / 开头，即使是 children 里的 path 也要以 / 开头。如下示例

目的：

经常有这样的场景：某个页面有问题，要立刻找到这个 vue 文件，如果不用以/开头，path 为 parent 和 children 组成的，可能经常需要在 router 文件里搜索多次才能找到，而如果以/开头，则能立刻搜索到对应的组件

9.2 Vue 项目目录规范

9.2.1 基础

vue 项目中的所有命名一定要与后端命名统一。

比如权限：后端 privilege, 前端无论 router , store, api 等都必须使用 privielege 单词！

9.2.2. 使用 Vue-cli 脚手架

使用 vue-cli 来初始化项目，项目名按照上面的命名规范。

9.2.3 目录说明

目录名按照上面的命名规范，其中 components 组件用大写驼峰，其余除 components 组件目录外的所有目录均使用 kebab-case 命名。

9.2.4 api 目录

文件、变量命名要与后端保持一致。

此目录对应后端 API 接口，按照后端一个 controller 一个 api js 文件。若项目较大时，可以按照业务划分子目录，并与后端保持一致。

api 中的方法名字要与后端 api url 尽量保持语义高度一致性。

对于 api 中的每个方法要添加注释，注释与后端 swagger 文档保持一致。

9.2.4 assets 目录

assets 为静态资源，里面存放 images, styles, icons 等静态资源，静态资源命名格式为 kebab-case

9.2.5 components 目录

此目录应按照组件进行目录划分，目录命名为 KebabCase，组件命名规则也为 KebabCase

9.2.6 router 与 store 目录

这两个目录一定要将业务进行拆分，不能放到一个 js 文件里。

router 尽量按照 views 中的结构保持一致

store 按照业务进行拆分不同的 js 文件

9.2.7 views/pages 目录

命名要与后端、router、api 等保持一致

components 中组件要使用 PascalCase 规则

9.2.8 注释说明

整理必须加注释的地方

公共组件使用说明

api 目录的接口 js 文件必须加注释

store 中的 state, mutation, action 等必须加注释

vue 文件中的 template 必须加注释，若文件较大添加 start end 注释

vue 文件的 methods，每个 method 必须添加注释

vue 文件的 data, 非常见单词要加注释

9.3 其他

9.3.1 尽量不要手动操作 DOM

因使用 vue 框架，所以在项目开发中尽量使用 vue 的数据驱动更新 DOM，尽量（不到万不得已）不要手动操作 DOM，包括：增删改 dom 元素、以及更改样式、添加事件等。

9.3.2 删除无用代码

因使用了 git/svn 等代码版本工具，对于无用代码必须及时删除，例如：一些调试的 console 语句、无用的弃用功能代码。

10.代码规范

10.1 安装 VS Code 插件

ESLint

Vetur

Prettier - Code formatter

10.2 配置 VS Code

打开 vscode 软件 【文件】---【首选项】---【设置】按照下图点右上角那个小文件图标，会弹出一个 【settings.json】 文件。

settings.json

10.3 修改Eslint的配置文件

.eslintrc.js