大觅网—开发规范

大觅网项目架构从两个方向进行设计：前端架构和后端架构，将前端和后端完全分离。整个项目开发过程由项目组的前端团队与后端团队共同协作完成，需要遵循开发规范如下：

1.1 前端开发规范

1.1.1 规范目的

- 提高团队协作效率，实现代码一致性
- 通过代码风格的一致性，降低维护代码的成本以及改善多人协作的效率
- 方便新进的成员快速上手
- 输出高质量的代码
- 同时遵守最佳实践，确保页面性能得到最佳优化和高效的代码

本规范文档一经确认，前端开发人员必须按本文档规范进行前台页面开发。本文档如有不对或者不合适的地方请及时提出，经讨论决定后可以更新此文档。

1.1.2 基本原则

1.1.2.1 结构、样式、行为分离

尽量确保文档和模板只包 HTML 结构，样式都放到样式表里，行为都放到脚本里。

1.1.2.2 缩进

统一 4 个空格缩进或者 Tab 缩进，不要使用 Tab 和空格混搭。

1.1.2.3 文件编码

使用不带 BOM 的 UTF-8 编码;
在 HTML 中指定编码 <meta charset="utf-8"> 。

1.1.2.4 一律使用小写字母

```
<!-- Recommended -->
```

```
<img src="google.png" alt="Google">
<!-- Not recommended -->
<A HREF="/">Home</A>
```

1.1.3 命名

1.1.3.1 CSS 命名

- 使用语义化、通用的命名方式；
- 使用连字符 - 作为 ID、Class 名称界定符，不要驼峰命名法和下划线；
- 避免选择器嵌套层级过多，尽量少于 3 级；
- 避免选择器和 Class、ID 叠加使用；

出于性能考量，在没有必要的情况下避免元素选择器叠加 Class、ID 使用。

元素选择器和 ID、Class 混合使用也违反关注分离原则。如果 HTML 标签修改了，就要再去修改 CSS 代码，不利于后期维护。

```
/* Not recommended */
.red {}
.box_green {}
.page .header .login #username input {}
 ul#example {}
/* Recommended */
#nav {}
.box-video {}
#username input {}
#example {}
```

1.1.3.2 JavaScript 命名

1）变量命名

变量，使用 Camel 命名法。

```
var loadingModules = {};
```

私有属性、变量和方法以下划线 _ 开头。

```
var _privateMethod = {};
```

常量，使用全部字母大写，单词间下划线分隔的命名方式。

```
var HTML_ENTITY = {};
```

2）函数

函数，使用 Camel 命名法。

函数的参数，使用 Camel 命名法。

```
function stringFormat(source) {}
function hear(theBells) {}
```

3）类

类名，使用名词。

```
function Engine(options) {}
```

4）boolean 类型的变量

boolean 类型的变量使用 is 或 has 开头。

```
var isReady = false;
var hasMoreCommands = false;
```

5）对象命名

Promise 对象用动宾短语的进行时表达。

```
var loadingData = ajax.get('url');

loadingData.then(callback);
```

1.1.3.3 接口命名规范

- 可读性强，见名晓义；
- 尽量不与 jQuery 社区已有的习惯冲突；
- 尽量写全。不用缩写，除非是下面列表中约定的；（变量以表达清楚为目标，uglify 会完成压缩体积工作）

常用词	说明
options	表示选项，与 jQuery 社区保持一致，不要用 config, opts 等
active	表示当前，不要用 current 等
index	表示索引，不要用 idx 等
trigger	触点元素
triggerType	触发类型、方式
context	表示传入的 this 对象

常用词	说明
object	推荐写全，不推荐简写为 o, obj 等
element	推荐写全，不推荐简写为 el, elem 等
length	不要写成 len, l
prev	previous 的缩写
next	next 下一个
constructor	不能写成 ctor
easing	动画平滑函数
min	minimize 的缩写
max	maximize 的缩写
DOM	不要写成 dom, Dom
.hbs	使用 hbs 后缀表示模版
btn	button 的缩写
link	超链接
title	主要文本
img	图片路径（img 标签 src 属性）
dataset	html5 data-xxx 数据接口
theme	主题
className	类名
classNameSpace	class 命名空间

1.1.4 注释

1.1.4.1 HTML 注释

- 模块注释

```
<!-- 文章列表列表模块 -->
<div class="article-list">
...
</div>
```

- 区块注释

```
<!--
@name: Drop Down Menu
@description: Style of top bar drop down menu.
@author: Ashu(Aaaaaashu@gmail.com)
-->
```

1.1.4.2 CSS 注释

- 组件块和子组件块以及声明块之间使用一空行分隔，子组件块之间三空行分隔

```
/*
============================================================================
  组件块

============================================================================ */
/* 子组件块

============================================================================
*/.selector {
  padding: 15px;
  margin-bottom: 15px;}


/* 子组件块

============================================================================
*/.selector-secondary {
  display: block; /* 注释*/
}
.selector-three {
  display: inline;
}
```

1.1.4.3 JavaScript 注释

- 单行注释

  必须独占一行。

// 后跟一个空格，缩进与下一行被注释说明的代码一致。

- 多行注释

  多行注释以/*开始，以*/结束。

  多行注释总是出现在将要出现的代码段之前，注释与代码之间没有空行间隔。

  多行注释之前应当有一个空行，且缩缩进层级与器描述的代码保持一致。

- 文件注释用于告诉不熟悉这段代码的读者这个文件中包含哪些东西。 应该提供文件的大体内容，它的作者，依赖关系和兼容性信息。如下:

```
/**
 * @fileoverview Description of file, its uses and information
 * about its dependencies.
 * @author user@meizu.com (Firstname Lastname)
 * Copyright 2015 Meizu Inc. All Rights Reserved.
 */
```

1.1.5 Less 规范

1.1.5.1 代码组织

代码按一下顺序组织：
1. @import
2. 变量声明
3. 样式声明

```
@import "mixins/size.less";
@default-text-color: #333;
.page {
  width: 960px;
  margin: 0 auto;}
```

1.1.5.2 @import 语句

@import 语句引用的内容需要写在一对引号内,.less 后缀不得省略。引号使用 ’ 和 ″ 均可，但在同一项目内需统一。

```
/* Not recommended */
@import "mixins/size";@import 'mixins/grid.less';
```

```
/* Recommended */
@import "mixins/size.less";@import "mixins/grid.less";
```

1.1.5.3 混合（Mixins）

在定义 mixins 时，如果 mixins 名称不是一个需要使用的 className，必须加上括号，否则即使不被调用也会输出到 CSS 中。

如果混合的是本身不输出内容的 mixins，需要在 mixins 后添加括号（即使不传参数），以区分这是否是一个 className。

```
/* Not recommended */
.big-text {
  font-size: 2em;
}

h3 {
  .big-text;
  .clearfix;
}

/* Recommended */
.big-text() {
  font-size: 2em;
}

h3 {
  .big-text(); /* 1 */
  .clearfix(); /* 2 */
}
```

1.1.5.4 避免嵌套层级过多

- 将嵌套深度限制在 2 级。对于超过 3 级的嵌套，给予重新评估。这可以避免出现过于详实的 CSS 选择器。
- 避免大量的嵌套规则。当可读性受到影响时，将之打断。推荐避免出现多于 20 行的嵌套规则出现。

1.1.6 vue 框架开发

1.1.6.1 mock 数据模拟接口

由于后端提供的接口请求方式统一都是使用 post，并且按页面顺序开发。所以前端使用 mock.js 模拟数据，接口名称统一使用 api_<页面名称（或者缩写）>_<功能描述>。

如图 1 所示。

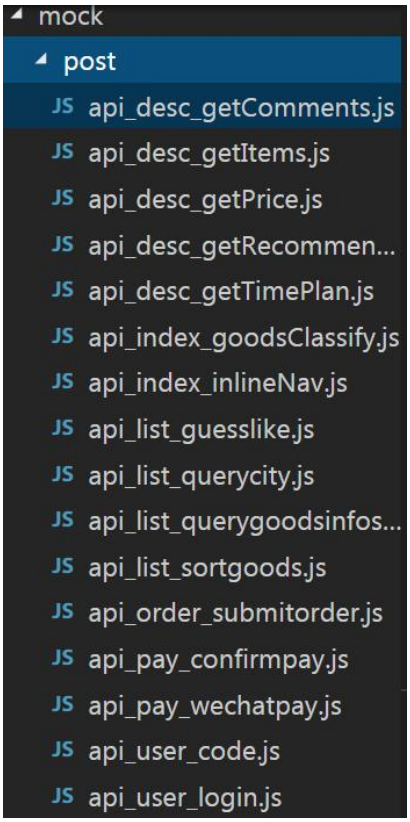

图 1

在项目中调用接口具体如图 2 所示。

```
//·获取注册验证码
export·function·postUserCode·(phone)·{
··return·request('post',·'/api/user/code',·{·phone·})
}

//·使用手机号注册
export·function·postUserRegister·(phone,·password,·vcode)·{
··return·request('post',·'/api/user/register',·{·phone,·password,·vcode·})
}
```

图 2

1.1.6.2 接口请求

由于页面比较繁多，请求接口也比较多，如果分散在不同页面中编写，后期和后端交互修改起来比较耗时，所以统一管理项目的所有接口，统一放在 src/http 下面，具体如图 3 所示。

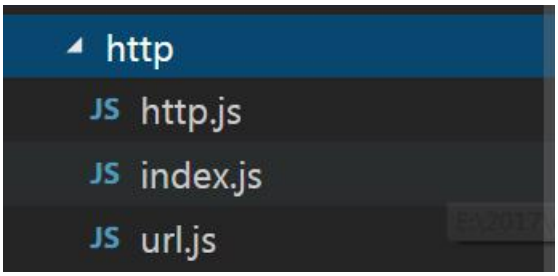

图 3

1.1.6.3 项目模块

项目的主要路线是根据页面来划分了，打开项目的时候首先展示的也是首页，开发的时候页面统一放在 pages 文件夹下面，如图 4 所示。

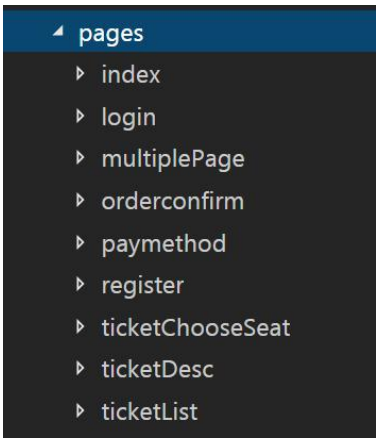

图 4

1.1.6.4 项目组件

Vue 项目开发中可复用的部分一般都会抽离成为组件，那么组件也不是想放哪里就放哪里的，项目中统一将组件放在 components 文件夹下面，然而组件的命名是根据该组件所赋予的功能来决定的。每个组件都是一个独立的，为了其他开发同伴更好的复用代码，也为了保证组件的完整性，每个组件中所有用到的图片、CSS 或者是 JS 代码都放在该组件的更目录下。这样便于更好的维护。组件里的 vue 文件一律使用 index.vue。具体如图 5 所示。引用该组件的时候只需要引用到该文件夹名称即可，具体如图 6 所示。

例如：引用 header 组件下的 index.vue 文件。

```
import header from "@/components/header/index.vue"; //  非简写
```

import header from "@/components/header";　　//简写，提倡使用

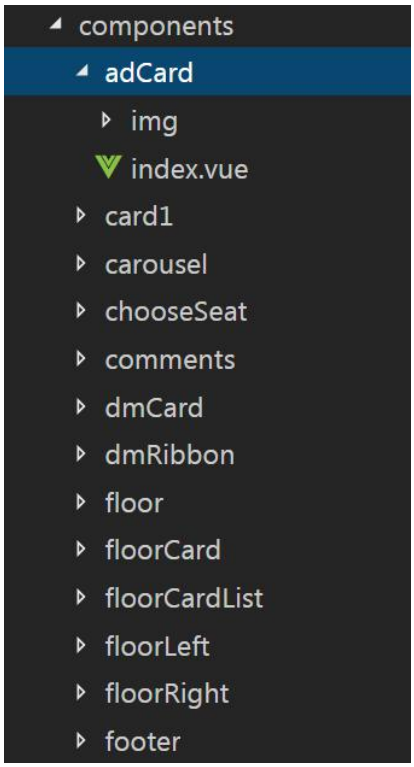

图 5

```
<script>
import header from "@/components/header";
import footer from "@/components/footer";
import carousel from "@/components/carousel";
import goodsClassify from "@/components/goodsClassify";
import card from "@/components/card1";
import floor from "@/components/floor";
```

图 6

1.1.6.5　项目公共部分

每个项目开发中必然都少不了会有些公共的代码，项目中 CSS 公共的样式代码放在 static 文件夹中。一些 JavaScript 的公共方法或者功能放在 common 文件夹中。

1.3 前后端交互规范

1. 【强制】前后端开发前必须先定义 schema
2. 【强制】API 不能干涉产品，只提供数据
3. 【强制】前后端边界处理
4. 【强制】开发环境下，后端开发人员必须保证 API 本地测试通过，统一使用 API 测试工具：Postman
5. 【强制】开发环境下，后端开发人员须保证使用 Swagger 生成 API 文档对接口描述清晰明确