

# 一. 开发规范

## 1. 工作目录构建规范

<https://segmentfault.com/a/1190000006031855>

```
├── assets
├── doc
├── └── user
├── i18n_import_data
├── └── project
├── logs
├── mock
├── public
├── src
├── └── css // 存放页面全局的css
├── └── scripts // 存放js
├── └── common
├── └── └── componentHolder
├── └── components // 基础组件
├── └── └── BigModal // 大弹框组件
├── └── └── BreadCrumb // 面包屑组件
├── └── └── DragSortingTable // 拖拽表格组件
├── └── └── GoodsDetail // 商品详情组件
├── └── └── GoodsPreview // 商品预览组件
├── └── └── LoadingComponent // loading组件
├── └── └── Panel // 折叠组件
├── └── └── PanelRetry //
├── └── └── Promotion // 促销表格组件
├── └── └── SideMenu // 侧边栏菜单组件
├── └── └── StepBox // 步骤条组件
├── └── └── TableCell // 表格单元组件
├── └── └── Title // Title组件
├── └── └── TopBar // 顶部信息条组件
├── └── └── UserSearchBox // 搜索框组件
```

## 2. 代码命名规范

### i. BEM命名方式

BEM(Block, Element, Modifier)是由Yandex团队提出的一种前端命名规范。其核心思想是将页面拆分成一个个独立的富有语义的块 (blocks) ,从而使得团队在开发复杂的项目变得高效, 并且十分有利于代码复用, 即便团队引入新成员, 也容易维护。在某种程度上, BEM和OOP是相似的。

BEM其实是块 (block) 、元素 (element) 、修饰符 (modifier) 的缩写, 利用不同的区块, 功能以及样式来给元素命名。这三个部分使用 `_` 与 `--` 连接 (这里用两个而不是一个是为了留下用于块儿的命名) 。命名约定的模式如下:

```
.block{}  
.block__element{}  
.block--modifier{}
```

- `block` 代表了更高级别的抽象或组件
- `block__element` 代表 `block` 的后代, 用于形成一个完整的 `block` 的整体
- `block--modifier` 代表 `block` 的不同状态或不同版本

```
<form class="site-search full">  
  <input type="text" class="field">  
  <input type="Submit" value ="Search" class="button">  
</form>
```

但是如果时用BEM规范去写, 代码如下:

```
<form class="site-search site-search--full">  
  <input type="text" class="site-search__field">  
  <input type="Submit" value ="Search" class="site-search__button">  
</form>
```

对比一下不难发现使用BEM可以使我们的代码可读性更高。

## ii. OOCSS

OOCSS不是一个框架，也不是一种技术，更不是一种新的语言，他只不过是一种方法，一种书写方法，换句话说OOCSS其核心就是用最简单的方式编写最整洁，最于净的CSS代码，从而使代码更具重用性，可维护性和可扩展性（把原本写在一起的样式，拆开多个class写，提高可复用性）

<https://v3.bootcss.com/components/#alerts>

```
1 <div class="alert alert-success" role="alert">...</div>
2 <div class="alert alert-info" role="alert">...</div>
3 <div class="alert alert-warning" role="alert">...</div>
4 <div class="alert alert-danger" role="alert">...</div>
```

### iii. Eslint

ESLint 这样的可以让你在编码的过程中发现问题，并且可以自己创建检测规则，保持代码编写风格的一致性

<https://www.cnblogs.com/my93/p/5681879.html>

## 二. 工作规范

1. 周报与日报
2. 邮件发送相关

## 三. Vue

### 1. 前端框架发展历史

### 2. 初始Vue.js

构建数据驱动的web应用开发框架

### 3. MV\*模式 (MVC/MVP/MVVM)

"MVC":Controller 薄, View 厚, 业务逻辑大都部署在 View。

- model view controller
  - views
  - model
  - controller

"MVVM":双向数据绑定, View的变动, 映射在 ViewModel, 反之一样

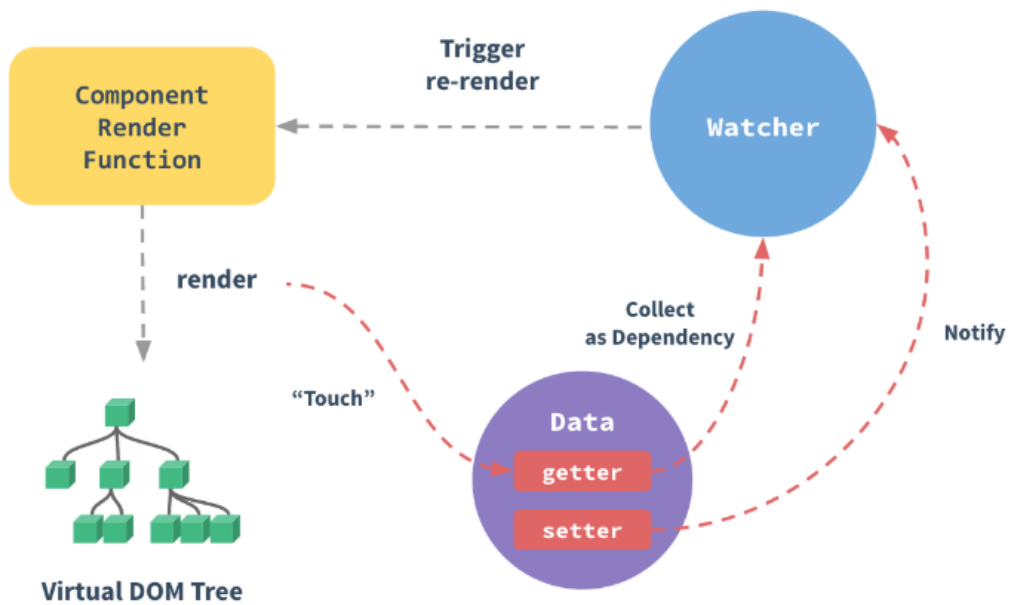
- model view viewmodel

"MVP":View 薄, 不部署任何业务逻辑, 称为"被动视图" (Passive View)

Presenter 厚, 逻辑都部署这里。

- model view presenter (android ,ios )

#### 4. Vue实现数据绑定的原理



<https://cn.vuejs.org/v2/guide/reactivity.html>