|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 修订记录 | | | | | | | | | | | | |
| 版 本 | 修 订 描 述 | | | | | | | | | | 修订日期 | |
| A0 | 初次建立 | | | | | | | | | | 2020/1/18 | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
| 评审 | | | | | | | | | | | | |
| 部门 | | 评审 | | | 部门 | | 评审人 | | 部门 | | | 评审人 |
| 平台研发中心 | | 程绪伟 | | | TrackSolid 产品PDT经理 | | 陈新军 | | 有线定位产品PDT经理 | | | 张宁 |
| 项目管理中心 | | 段小权 | | | ISD产品PDT经理 | | 王胜 | | 无线定位产品PDT经理 | | | 王阳 |
| 规划中心 | | 郑旻琪 | | | Tracker产品PDT经理 | | 李胜如 | | 个人穿戴产品PDT经理 | | | 龚超 |
| 云平台智慧车联产品 | | 武宇飞 | | | DVR 产品PDT经理 | | 赵志文 | | 几米圈产品PDT经理 | | | 林时配 |
| 分发部门 | | | | | | | | | | | | |
| PDT管理 | | | 规划中心 | | | 项目管理中心 | | | | 国内营销--解决方案部 | | |
| 研发中心-测试中心 | | | 研发中心--云平台 | | | 国内营销-研发管理中心 | | | |  | | |
| 编制: 谢洪涛 | | | | 审核:－－ | | | | 批准:程绪伟 | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 日期 | 更新人员 | 更新内容 |
| V1.0 | 2018/05/26 | 李合飞 | 初始版本 |
| V1.1 | 2018/05/30 | 李合飞 | **修改**：缩进由一个tab改为四个空格  **新增**：css中禁止滥用无规律百分比尺寸  **新增**：js对象为一行时，每个值逗号后面需要加空格  **新增**：js函数每个参数值逗号后面需要加空格 |
| V1.2 | 2018/06/08 | 李合飞 | **新增**：SVN和git提交规范  **新增**：js中==与===的使用  **新增**：button标签需要添加type类型 |
| V1.3 | 2018/06/11 | 李合飞 | **新增**：文件目录规范  **新增**：静态资源目录规范 |
| V1.4 | 2018/9/4 | 李合飞 | **修改**：多行js代码注释说明以/\*\*/的方式  **新增**：JS函数类注释规范以JSDoc为标准  **新增**：vue项目目录结构规范 |
| V1.5 | 2018/10/30 | 李合飞 | **新增**：自动化测试约定规范 |
| V1.6 | 2018/11/07 | 李合飞 | **修改**：前端给自动化测试定制唯一属性 |
| V1.7 | 2019/04/17 | 李合飞 | **新增**：部分规范区分传统开发和前后端分离开发模式 |
| V1.8 | 2019/04/29 | 李合飞 | **新增**：vue规范说明  **新增**：【强制】【推荐】【参考】三种标识  **修改**：函数命名新增添加、删除、编辑、保存命名前缀  **修改**：变量声明禁止ES5、ES6混用  **修改**：空格、大括号覆盖更多使用场景  **修改**：js类示例  **修改**：if else条件应用优化  **修改**：vue目录结构优化  **修改**：代码示例效果优化  **删除**：提示框展示形式(根据产品及UI设计开发)  **删除**：分页展示形式(根据产品及UI设计开发) |
| V1.8.1 | 2019/04/30 | 李合飞(李真真、肖武参与) | **新增**：vue相关框架属性书写顺序 |
| V1.9 | 2019/05/07 | 李合飞 | **新增**：形参命名规范  **新增**：文档标签合理放置位置 |
| V1.9.1 | 2019/05/16 | 李合飞 | **新增**：形参命名请求参数为params  **修改**：函数jsdoc注释示例 |
| V1.9.2 | 2019/05/17 | 李合飞 | **新增**：表单元素命名规律新增保存 |

目录

[ 规范目的 4](#_Toc13838_WPSOffice_Level1)

[ 通用规范 4](#_Toc13418_WPSOffice_Level1)

[ 文件规范 4](#_Toc6078_WPSOffice_Level1)

[1. 文件名用英文单词，多个单词用驼峰命名法 4](#_Toc7031_WPSOffice_Level1)

[2. 图片文件命名多个单词使用下划线命名法 4](#_Toc26335_WPSOffice_Level1)

[3. 公用CSS文件为base.css，公用JS文件为common.js 4](#_Toc8669_WPSOffice_Level1)

[4. 压缩的CSS、JS文件名必须带上min区分(适用传统开发模式) 4](#_Toc30322_WPSOffice_Level1)

[5. 文件数量多的目录，必须划分更细 4](#_Toc4212_WPSOffice_Level1)

[6. requireJS模块化项目目录结构(适用传统开发模式) 4](#_Toc16857_WPSOffice_Level1)

[ HTML篇 5](#_Toc7005_WPSOffice_Level1)

[ CSS篇 22](#_Toc16876_WPSOffice_Level1)

[ JavaScript篇 31](#_Toc92_WPSOffice_Level1)

[ Vue篇 50](#_Toc3177_WPSOffice_Level1)

[1． 规范说明 50](#_Toc14618_WPSOffice_Level1)

[2． 项目目录结构 51](#_Toc2988_WPSOffice_Level1)

[ svn和git提交篇 55](#_Toc26742_WPSOffice_Level1)

[1． 提交必须填写说明，禁止内容为空 55](#_Toc31044_WPSOffice_Level1)

[ 自动化测试篇 55](#_Toc9032_WPSOffice_Level1)

* **规范目的**

为提升开发人员规范意识，减少因规范问题导致的低效开发，输出符合规范的高质量项目，

降低项目后期维护成本，特制订此文档，提高团队协作效率。本规范文档一经确认, 前端开

发人员必须按此规范进行项目开发，本文档内容如有错误或不适用，经团队商议确定后方可

升级修订。

* **通用规范**

1. 符合web标准，遵循结构、样式、行为分离的代码组织模式(适用传统开发模式)
2. 每个页面都必须有对应独立的同名css、js文件(适用传统开发模式)
3. 文档编码统一为UTF-8
4. 代码格式化，缩进统一为四个空格，保持干净整洁
5. 命名统一用英文，禁止使用拼音
6. 命名空间必须是有意义的，禁止使用无意义的命名
7. CSS、JS中的非注释类中文字符须转换成unicode编码使用,以避免编码错误时乱码显示
8. JavaScript函数、方法、类代码注释规范以JSDoc为标准

* **文件规范**

1. 【强制】文件名用英文单词，多个单词用驼峰命名法

index.html，aboutUs.js

1. 【强制】图片文件命名多个单词使用下划线命名法

body\_bg.jpg

1. 【强制】公用CSS文件为base.css，公用JS文件为common.js
2. 【强制】压缩的CSS、JS文件名必须带上min区分(适用传统开发模式)
3. 【强制】文件数量多的目录，必须划分更细
4. 【推荐】requireJS模块化项目目录结构(适用传统开发模式)

|  |
| --- |
| static  ├── index.html  ├── css 样式  │ ├── base.css  │ ├── xxxx.css  │ └── xxxx.css  ├── js 脚本  │ ├── app 业务模块  │ │ ├── common.js  │ │ ├── user (例子)  │ │ │ ├── add.js  │ │ │ └── edit.js  │ │ └── index.js  │ ├── lib 组件模块  │ │ ├── shim 非模块化组件  │ │ │ ├── bootstrap (例子)  │ │ │ │ ├── popper.js  │ │ │ │ ├── tooltip.js  │ │ │ │ └── modal.js  │ │ ├── map (例子)  │ │ │ ├── baidu.js  │ │ │ └── google.js  │ │ ├── xxx.js  │ │ └── xxx.js  │ ├── require.js  │ └── requireConfig.js  ├── img 图片  └── fonts 字体 |

* **HTML篇**

1. 【推荐】文档标签合理放置位置

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <!-- 放置meta标签开始 -->  <meta charset="utf-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">  <!-- 放置meta标签结束 -->  <!-- 放置与搜索引擎相关的标签开始 -->  <title>Examples</title>  <meta name="description" content="">  <meta name="keywords" content="">  <!-- 放置与搜索引擎相关的标签结束 -->  <!-- 放置样式标签开始 -->  <link href="" rel="stylesheet">  <style></style>  <!-- 放置样式标签结束 -->  </head>  <body>    <!-- 放置布局标签开始 -->  <header></header>  <main></main>  <footer></footer>  <!-- 放置布局标签结束 -->  <!-- 放置脚本标签开始 -->  <script></script>  <!-- 放置脚本标签结束 -->  </body>  </html> |

1. 【强制】**文档头声明统一使用HTML5，禁止使用HTML4或XHTML文档头**

|  |
| --- |
| <!-- 错误 -->  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  <!-- 正确 -->  <!DOCTYPE html> |

1. 【强制】**文档编码为UTF-8，禁止使用GB2312**

|  |
| --- |
| <!-- 错误 -->  <meta http-equiv="Content-Type" content="text/html; charset=gb2312"/>  <!-- 正确 -->  <meta charset="utf-8"> |

1. 【强制】 **设置IE采用支持的最高级标准模式渲染，如果有Chrome则强制使用****Chrome渲**

**染(适用传统开发模式)**

|  |
| --- |
| <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"> |

1. 【强制】**设置国产双核浏览器优先采用极速模式渲染**

|  |
| --- |
| <meta name="renderer" content="webkit"> |

1. 【强制】**移动端页面必须添加下面的meta标签**

|  |
| --- |
| <meta content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0" name="viewport"> |

1. 【推荐】**style和script标签不需要定义type属性**

|  |
| --- |
| <!-- 错误 -->  <style type="text/css">...</style>  <script type="text/javascript">...</script>  <!-- 正确 -->  <style>...</style>  <script>...</script> |

1. 【强制】**link、style标签必须放在<head>标签内，script标签优先放在</body>前面**

|  |
| --- |
| <!-- 错误 -->  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">  <title>Examples</title>  </head>  <link href="" rel="stylesheet">  <style></style>  <script></script>  <body>  <div></div>  </body>  </html>  <!-- 正确 -->  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">  <title>Examples</title>  <link href="" rel="stylesheet">  <style></style>  </head>  <body>  <div></div>  <script></script>  </body>  </html> |

1. 【强制】**标签名及属性统一用小写**

|  |
| --- |
| <!-- 错误 -->  <INPUT TYPE="button" ID="idName"></SPAN>  <!-- 正确 -->  <input type="button" id="idName"> |

1. 【强制】**标签属性值统一用双引号(")，禁止使用单引号(')**

|  |
| --- |
| <!-- 错误 -->  <span class='test'></span>  <!-- 正确 -->  <span class="test"></span> |

1. 【推荐】**HTML语义化**

* 尽可能少的使用无语义的标签div和span；
* 标题根据重要性用h\*(同一页面只能有一个h1)
* 段落标记用p, 列表用ul
* 内联元素中不可嵌套块级元素
* 在语义不明显时，既可以使用div或者p时，尽量用p, 因为p在默认情况下有上下间距，对兼容特殊终端有利；
* 不要使用纯样式标签，如：b、font、u等，改用css设置。
* 需要强调的文本，可以包含在strong或者em标签中（浏览器预设样式，能用CSS指定就不用他们），strong默认样式是加粗（不要用b），em是斜体（不用i）
* 使用表格时，标题要用caption，表头用thead，主体部分用tbody包围，尾部用tfoot包围。表头和一般单元格要区分开，表头用th，单元格用td；
* 每个input标签对应的说明文本都需要使用label标签，并且通过为input设置id属性，在lable标签中设置for=someld来让说明文本和相对应的input关联起来。

1. 【强制】**标签语义化，禁止滥用div和span标签**

|  |
| --- |
| <!-- 错误 -->  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">  <title>Examples</title>  </head>  <body>  <div class="header"></div>  <div class="main"></div>  <div class="footer"></div>  </body>  </html>  <!-- 正确 -->  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">  <title>Examples</title>  </head>  <body>  <header class="header"></header>  <main class="main"></main>  <footer class="footer"></footer>  </body>  </html> |

1. 【强制】**合理使用嵌套规则，禁止随意嵌套标签**

|  |
| --- |
| <!-- 错误 -->  <ul>  <div></div>  </ul>  <p>  <div></div>  </p>  <!-- 正确 -->  <ul>  <li></li>  </ul>  <div>  <p></p>  </div> |

1. 【强制】**标签中禁止出现行内样式，样式必须写在CSS文件中(适用传统开发模式)**

|  |
| --- |
| <!-- 错误 -->  <div class="class-name" style="font-size: 18px;"></div>  <!-- 正确 -->  <div class="class-name fs-18" ></div> |

1. 【强制】**标签必须成对出现有结束标签(自闭合标签除外)**

|  |
| --- |
| <!-- 错误 -->  <div>  <p>  <!-- 正确 -->  <div></div>  <p></p> |

1. 【推荐】**自闭合标签不需要写后面的斜杠**

|  |
| --- |
| <!-- 错误 -->  <input type="checkbox" />  <img src="..." />  <br/>  <hr/>  <!-- 正确 -->  <input type="checkbox">  <img src="...">  <br>  <hr> |

1. 【推荐】**尽可能减少标签嵌套层级**

|  |
| --- |
| <!-- 错误 -->  <div>  <div>  <div></div>  </div>  </div>  <!-- 正确 -->  <div>  <div></div>  </div> |

1. 【推荐】**特殊字符(如<>&空格)尽量使用实体引用**

|  |
| --- |
| <!-- 错误 -->  <div>< & ></div>    <!-- 正确 -->  <div>&lt;&nbsp;&amp;&nbsp;&gt;</div> |

1. 【强制】**重要的img标签需要添加alt属性说明**

|  |
| --- |
| <!-- 错误 -->  <img src="spreadsheet.png">    <!-- 正确 -->  <img src="spreadsheet.png" alt="Spreadsheet screenshot."> |

1. 【强制】**button元素必须指明type属性值**

|  |
| --- |
| <!-- 错误 -->  <button>确定</button>  <button>提交</button>    <!-- 正确 -->  <button type="button">确定</button>  <button type="submit">提交</button> |

1. 【强制】**设置截断或文本溢出隐藏的标签必须加上title属性**
2. 【推荐】**针对移动设备开发合理指定输入框的type属性**

|  |
| --- |
| <input type="number">  <input type="color"> |

1. 【推荐】**含有描述性表单元素（input，textarea）添加label。**

|  |
| --- |
| <p>  <label for="test">测试</label>  <input type="text" id="test" />  </p> |

1. 【强制】**主体结构必须写注释，内容两边必须包含一个空格，格式为<!-- 内容 -->**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">  <title>Examples</title>  </head>  <body>  <!-- header -->  <header class="header"></header>  <!-- main -->  <main class="main"></main>  <!-- footer -->  <footer class="footer"></footer>  </body>  </html> |

1. 【推荐】**格式化规则，每个块元素、列表元素和表单元素后，加上一新空白行。内联元素写在一行，块状元素另起一行**

|  |
| --- |
| <div>  <p><em></em><strong></strong></p>  </div>  <ul>  <li></li>  </ul> |

1. 【强制】**属性书写顺序，按照使用频率顺序书写，保证易读性**

* class
* id
* name
* data-\*
* src,for,type,href,value,max-length,max,min,pattern
* placeholder,title,alt
* aria-\*,role
* required,readnoly,disabled



|  |
| --- |
| <!-- 错误 -->  <div title="..." data-type="..." id="idName" class="class-name"></div>  <input disabled="disabled" max-length="20" value="1" type="text" name="testName">  <!-- 正确 -->  <div class="class-name" id="idName" data-type="..." title="..." ></div>  <input name="testName" type="text" value="1" max-length="20" disabled="disabled"> |

1. 【推荐】**标签布尔类型属性不需要赋值**

|  |
| --- |
| <!-- 错误 -->  <input type="checkbox" checked="checked">  <input type="text" disabled="disabled">  <select>  <option value="1" selected="selected">1</option>  </select>  <!-- 正确 -->  <input type="checkbox" checked>  <input type="text" disabled>  <select>  <option value="1" selected>1</option>  </select> |

1. 【强制】**属性命名必须为有意义的，禁止使用无意义的命名**

|  |
| --- |
| <!-- 错误 -->  <div class="aaa" id="bbb"></div>  <!-- 正确 -->  <div class="article-info" id="articleInfo"></div> |

1. 【参考】**功能模块命名**

|  |  |
| --- | --- |
| 结构 | |
| 容器 | container |
| 页头 | header |
| 内容 | content |
| 页面主体 | main |
| 页尾 | footer |
| 导航 | nav |
| 侧栏 | sidebar |
| 栏目 | column |
| 页面外围控制整体布局宽度 | wrapper |
| 左右中 | left right center |
|  |  |
|  |  |
| 导航 | |
| 导航 | nav |
| 主导航 | mainbav |
| 子导航 | subnav |
| 顶导航 | topnav |
| 边导航 | sidebar |
| 左导航 | leftsidebar |
| 右导航 | rightsidebar |
| 菜单 | menu |
| 子菜单 | submenu |
| 标题 | title |
| 摘要 | summary |
|  |  |
| 功能 | |
| 标志 | logo |
| 广告 | banner |
| 登录 | login |
| 登录条 | loginbar |
| 注册 | regsiter |
| 搜索 | search |
| 功能区 | shop |
| 标题 | title |
| 加入 | joinus |
| 状态 | status |
| 按钮 | btn |
| 滚动 | scroll |
| 标签页 | tab |
| 文章列表 | list |
| 提示信息 | msg |
| 当前的 | current |
| 小技巧 | tips |
| 图标 | icon |
| 注释 | note |
| 指南 | guide |
| 服务 | service |
| 热点 | hot |
| 新闻 | news |
| 下载 | download |
| 投票 | vote |
| 合作伙伴 | partner |
| 友情链接 | link |
| 版权 | copyright |
| 注册 | regsiter |
|  |  |

1. 【强制】**多个单词命名时class采用中划线连接，id和name命名采用驼峰命名法**

|  |
| --- |
| <!-- 错误 -->  <div class="class\_name" id="id-name"></div>  <input name="textname" type="text">  <!-- 正确 -->  <div class="class-name" id="idName"></div>  <input name="textName" type="text"> |

1. 【强制】**功能型的元素class命名规范格式是：元素简写-功能名(根据框架规则)**

|  |
| --- |
| <!-- 错误 -->  <button class="save"></button>  <!-- 正确 -->  <button class="btn-save"></button> |

1. 【强制】**与行为相关的class命名以js-为前缀，禁止用行为class定义样式**
2. 【强制】**如果标签需要自定义属性，使用data-xxx方式命名**

|  |
| --- |
| <div data-name="test"></div> |

1. 【强制】**重复内容必须使用模板，模板代码必须使用缩进规则**
2. 【强制】**同一页面，避免使用相同的name和id，因为在ie中会混淆这两个属性**
3. 【强制】**table表格标签上禁止直接定义表现,必须写在css里**

|  |
| --- |
| <!-- 错误 -->  <table border="0" cellpadding="0" cellspacing="0" ></table>  /\* 正确 \*/  table {  border: 0;  border-spacing: 0;  border-collapse: collapse;  } |

1. 【推荐】**不要指定引入资源所带的具体协议（http:, https:），除非这两者协议都不可用**

|  |
| --- |
| <!-- 错误 -->  src="http://www.google.com/js/gweb/analytics/autotrack.js"  background: url(http://www.google.com/images/example);  <!-- 正确 -->  src="//www.google.com/js/gweb/analytics/autotrack.js"  background: url(//www.google.com/images/example); |

1. 【推荐】**书写链接地址时避免重定向，即在URL地址后面加“/”**

|  |
| --- |
| href="http://www.kahn1990.com/" |

1. 【强制】**button标签必须有type类型**

|  |
| --- |
| <!-- 错误 -->  <button>按钮</button>  <!-- 正确 -->  <button type="button">按钮</button> |

* **CSS篇**

1. 【强制】**禁止使用ID或行为js-前缀的class选择器定义样式**

|  |
| --- |
| /\* 错误 \*/  #header {}  .js-btn-save {}  /\* 正确 \*/  .btn-default {} |

1. 【推荐】**不要使用元素标签名结合ID或class进行组合**

|  |
| --- |
| /\* 错误 \*/  ul.example {}  div.error {}  /\* 正确 \*/  .example {}  .error {} |

1. 【强制】**选择器与属性块中间留有一个空格**

|  |
| --- |
| /\* 错误 \*/  .example{}  /\* 正确 \*/  .example {} |

1. 【强制】**多个选择器公用属性块时每个选择器占一行**

|  |
| --- |
| /\* 错误 \*/  .example, .error, .success {}  /\* 正确 \*/  .example,  .error,  .success {} |

1. 【强制】**属性声明顺序**

* 盒子模型

box-sizing

* 内容

content

* 显示属性

display || visibility || opacity

* 位置属性

position top || right || bottom || left || z-index

float || clear

* 布局属性

flex-direction || flex-wrap || flex-flow || justify-content || align-items || align-content

grid-columns || grid-rows

* 多列属性

columns || column-count || column-width || column-span || column-fill || column-gap || column-rule

* 表格属性

table-layout || border-collapse || border-spacing || caption-side || empty-cell

* 大小

width || max-width || min-width

height || max-height || min-height

margin

padding

border || border-radius

outline

overflow || clip

* 文字系列

font

line-height

color

text-overflow

text-align

text-indent

text-transform

text-decoration

text-justify

text-outline

text-shadow

text-wrap

text-emphasis

vertical-align

word-spacing

white-space

letter-spacing

direction

* 计数器

counter-reset

counter-increment

counter()/counters()

* 背景

background

* 其他

list-style

user-select

resize

cursor

animation

transform

transition

perspective

box-shadow

1. 【强制】**声明样式每条单独一行。冒号后面留空格，最后的分号不能省略**

|  |
| --- |
| /\* 错误 \*/  .example {position:absolute;left:0;top:0;z-index:1;width:100px;height:100px}  /\* 正确 \*/  .example {  position: absolute;  left: 0;  top: 0;  z-index: 1;  width: 100px;  height: 100px;  } |

1. 【推荐】**写属性值的时候尽量使用缩写**

|  |
| --- |
| /\* 错误 \*/  .example {  padding-top: 10px;  padding-right: 5pxpx;  padding-top: 10px;  padding-left: 5px;  font-size: 14px;  font-family: Microsoft YaHei;  font-height: 1.8;  }  /\* 正确 \*/  .example {  padding: 10px 5px;  font: 14px/1.8 Microsoft YaHei;  } |

1. 【推荐】**属性值为0可省略px单位，为小数点可省略0**

|  |
| --- |
| .example {  margin: 0;  opacity: .5;  } |

1. 【推荐】**16进制色彩值两两相同的可以简写为三位**

|  |
| --- |
| /\* 错误 \*/  .example {  color: #66ccff;  }  /\* 正确 \*/  .example {  color: #6cf;  } |

1. 【强制】**CSS中使用双引号，禁止使用单引号**

|  |
| --- |
| /\* 错误 \*/  .example {  font-family: 'Microsoft YaHei';  }  /\* 正确 \*/  .example {  font-family: "Microsoft YaHei";  } |

1. 【推荐】**URL内的引号可以省略**

|  |
| --- |
| .example {  background: url(www.baidu.com) #fff;  } |

1. 【强制】**必须为大区块样式添加注释, 小区块适量注释**

|  |
| --- |
| /\* header \*/  .header {}  /\* main \*/  .main{}  /\* footer \*/  .footer{} |

1. 【推荐】**提高样式重复使用率**

|  |
| --- |
| /\* 错误 \*/  .example {  width: 100px;  height: 100px;  }  .box {  width: 100px;  height: 100px;  }  /\* 正确 \*/  .example,  .box {  width: 100px;  height: 100px;  } |

1. 【推荐】**选择器尽量简短，避免层级过多**

|  |
| --- |
| /\* 错误 \*/  .example ul li a span {}  /\* 正确 \*/  .example li span {} |

1. 【推荐】**减少使用影响性能的属性, 比如position:absolute || float**
2. 【强制】**尽量将媒体查询的规则靠近他们相关的规则，不要打包放在单一样式文件中或者放在文档底部**

|  |
| --- |
| .example {}  .example-avatar {}  .example-selected {}  @media (min-width: 480px) {  .example { ...}  .example-avatar { ... }  .example-selected { ... }  } |

1. 【推荐】**不要轻易在CSS中使用@import**
2. 【推荐】**不要轻易在CSS中使用!important**
3. 【推荐】**不要轻易在CSS中使用“\*”选择符**
4. 【推荐】**避免使用“\*”设置{ margin: 0; padding: 0; }**
5. 【强制】**禁止使用低效的选择器**

|  |
| --- |
| /\* 错误 \*/  body > \* {}  ul > li > a {} |

1. 【强制】**禁止使用无规律的百分比尺寸**

|  |
| --- |
| /\* 错误 \*/  div {  width: 37%;  height: 59%;  } |

* **JavaScript篇**

1. 【强制】**非特殊情况禁止使用双引号，统一用单引号**

|  |
| --- |
| /\* 错误 \*/  var str = "test";  /\* 正确 \*/  var str = 'test'; |

1. 【强制】**变量声明规范，必须以var、const、let声明，禁止ES5、ES6声明混用**

|  |
| --- |
| /\* 错误 \*/  var str = 'hello world';  let isRead = true;  len = 10;  /\* 正确 \*/  let str = 'hello world';  let isRead = true; |

1. 【强制】**变量命名规范，采用小驼峰式命名规则，前缀应当是名词，如果是boolean 类型的变量则使用 is 或 has 开头**

|  |
| --- |
| var maxCount = 10;  var isRead = false; |

1. 【参考】**函数命名规范，采用小驼峰式命名规则，前缀应当是动词，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 |
| get | 获取某个值，如果结果为数组，命名最后需加s表示复数 | 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 |
| count | 获取统计值 | 函数返回一个数值。 |
| add/insert | 插入某个值 | 无返回值或返回是否设置成功 |
| remove/delete | 删除某个值 | 无返回值或返回已删除的值 |
| update/edit | 更新某个值 | 无返回值或返回更新的值 |
| save | 保存某个值 | 无返回值或返回是否设置成功 |

|  |
| --- |
| // 是否可阅读  function canRead() {  return true;  }    // 获取名称  function getName() {  return this.name;  } |

1. 【强制】**常量命名规范，名称全部大写字母，使用下划线分割单词，力求语义表达完整清楚，不要嫌名字长**

|  |
| --- |
| var MAX\_COUNT = 10;  var URL = 'http://www.baidu.com'; |

1. 【强制】**构造函数命名规范，大驼峰式命名法，首字母大写**

|  |
| --- |
| function Student(name) {  this.name = name;  }  var st = new Student('tom'); |

1. 【强制】**形参命名规范**

|  |  |
| --- | --- |
| 类型 | 形参命名 |
| 字符串 | str或val |
| 普通对象 | obj |
| 数组对象 | arr |
| Json对象(混合) | data |
| 元素对象 | ele |
| 数字 | num |
| 布尔 | boo 或 flag |
| 函数 | fun 或 fn |
| 请求参数集合 | params |

|  |
| --- |
| function fn(str|val) { //字符串  }  function fn(obj) { //普通对象  }  function fn(arr) { //数组对象  }  function fn(data) { //json对象  }  function fn(ele) { //元素对象  }  function fn(num) { //数字  }  function fn(boo|flag) { //布尔  }  function fn(fun|fn) { //函数  }  function fn(params) { //请求传参  } |

1. 【强制】**类命名规范，与构造函数相同**

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。

|  |
| --- |
| class Person {  let \_name: 'zhangsan'; //私有属性  constructor() { }    getName() { // 公共方法  return this.\_name;  }    setName(name) { // 公共方法  this.\_name = name;  }  }  const person = new Person();  person.setName('mervyn');  person.getName(); |

1. 【强制】**不要在块内声明一个函数**

|  |
| --- |
| /\* 错误 \*/  if (x) {  function foo() {}  }  /\* 正确 \*/  if (x) {  var foo = function() {};  } |

1. 【强制】**单行代码注释说明规范，双斜线开始，双斜线与注释文字之间保留一个空格**

|  |
| --- |
| // 调用了一个函数；1)单独在一行  setTitle();    var maxCount = 10; // 设置最大量；2)在代码后面注释    setName(); // 3)注释代码 |

1. 【强制】**多行代码注释说明规范，以/\*开头，\*/结尾，**

|  |
| --- |
| /\* 这是多行数据的注释说明方式 \*/  var obj = {  a: 100,  b: function() {}  }; |

1. 【强制】**JSdoc函数(方法)注释**

|  |
| --- |
| /\*\*  \* 函数说明  \* @关键字  \*/ |

|  |  |  |  |
| --- | --- | --- | --- |
| 释名 | 语法 | 含义 | 示例 |
| @param | {类型} 参数名 描述 | 描述参数的信息 | @param {String} name 名称 |
| @return | {类型} 描述 | 描述返回值的信息 | @return {Boolean} true:可执行 |
| @author | 名称 (联系方式) | 描述此函数作者的信息 | @author 张三(zhangsan@xx.com) |
| @date | 日期 | 描述此函数更新时间 | @date 2019-05-16 12:33:32 |
| @version | 版本号 | 描述此函数的版本号 | @version 1.0.3 |
| @example | 示例代码 | 演示函数的使用 | @example setTitle('测试') |

* 组件/插件注释

|  |
| --- |
| /\*\*  \* @authors zhangsan (you@example.org)  \* @date 2019-05-16 12:33:32  \* @version 1.0.1  \*/ |

* 构造函数、类注释

|  |
| --- |
| /\*\*  \* Book类，代表一个书本.  \* @constructor  \* @param {string} title - 书本的标题.  \* @param {string} author - 书本的作者.  \*/  functionBook(title, author) {  this.title = title;  this.author = author;  }  Book.prototype = {  /\*\*  \* 获取书本的标题  \* @returns {string|\*} 返回当前的书本名称  \*/  getTitle: function() {  returnthis.title;  },  /\*\*  \* 设置书本的页数  \* @param pageNum {number} 页数  \*/  setPageNum: function(pageNum) {  this.pageNum = pageNum;  }  }; |

* 函数单层参数注释

|  |
| --- |
| /\*\*  \* 单层传参说明示例  \* @param {String} str 字符串  \* @param {Number} num 数字  \* @param {Array} arr 数组  \* @param {Boolean} boo 数组  \* @param {Element} ele 元素  \* @param {Function} fn 函数  \*/  function example(str, num, arr, boo, ele, fn) {  } |

* 函数多层参数注释

|  |
| --- |
| /\*\*  \* 多层传参说明示例  \* @param {String} params.str 字符串  \* @param {Number} params.num 数字  \*/  function example(params) {  var obj = {  name: params.str,  age: params.num  };  return obj;  } |

* 函数参数支持多种类型注释

|  |
| --- |
| /\*\*  \* 参数支持多种类型用|分割说明示例  \* @param {String} key 键名  \* @param {String|Number|Array} val 支持多种类型参数  \* @param {\*} all 支持所有类型参数  \*/  function example(key, val, all) {  } |

* 函数返回注释

|  |
| --- |
| /\*\*  \* 函数返回说明示例  \* @return {Object} 返回一个对象  \*/  function example(params) {  return {};  } |

1. 【强制】**分号的使用**

|  |
| --- |
| // 赋值必须加分号  var str = 'test';  var foo = function() {  };  // 函数调用必须加分号  foo();  // 函数声明结束禁止加分号  function test() {  }  // if、else、for、with、switch语句结束禁止加分号  If (true) {  } |

1. 【强制】**空格的使用**

|  |
| --- |
| // 赋值时等号两边要留空格  var str = 'test';  // 多个变量声明时逗号后面要用空格隔开  var str, test1, test2, test3;  // 对象键值对冒号后面要加空格  var obj = {  a: 1,  b: 2  };  //对象每个值逗号后面需要加空格  var arr = { a: 1, b: 2, c: 3, d: 4 };  //数组每个值逗号后面需要加空格  var arr = [1, 2, 3, 4, 5, 6];  //函数每个参数值逗号后面需要加空格  function foo(a, b, c) {}  //字符串拼接符号两边要加空格  var str = 'a' + 'bout';  //四则运算符两边要加空格  var num = 1 + 2 / 3 \* 4 - 5 % 6;  //逻辑运算符、位运算符两边要加空格  &&  || !  &  | ^ ~ >> << >>>  //if条件小括号两边及else两边需要加空格  var isRead = true;  if (isRead) {} else {}  // 函数名与小括号之间不能有空格，但小括号与花括号之间要有空格，  function foo() {} |

1. 【强制】**逗号的使用**

|  |
| --- |
| /\* 错误 \*/  // 数组如果最后一个值如果没有则删除逗号  var arr = [1, 2, 3, 4, 5, 6,];  // 对象最后一个键值对后面的逗号需删除  var obj = {  a: 1,  b: 2,  };  /\* 正确 \*/  var arr = [1, 2, 3, 4, 5, 6];  var obj = {  a: 1,  b: 2  }; |

1. 【强制】**大括号的使用**

如果是大括号内为空，则简洁地写成{}即可，不需要换行；

如果是非空代码块则： 1） 左大括号前不换行。 2） 左大括号后换行。 3） 右大括号前换行。 4） 右大括号后还有else等代码则不换行；表示终止的右大括号后必须换行。

在if/else/for/while/do语句中必须使用大括号，不可省略，即使只有一行代码。

|  |
| --- |
| /\* 错误 \*/  var obj = {  };  //分号会被隐式插入到代码中, 所以务必在同一行上插入大括号  if (something)  {  // ...  } else {  // ...  }  /\* 正确 \*/  var obj = {};  if (something){  // ...  } else {  // ...  } |

1. 【强制】**小括号的使用**

对于一元操作符(如delete, typeof 和void ), 或是在某些关键词(如return, throw,

case, new )之后, 不要使用括号

1. 【强制】**switch的使用**

在一个switch块内，每个case要么通过break/return等来终止，要么注释说明程序将继续执行到哪一个case为止；

在一个switch块内，都必须包含一个default语句并且放在最后，即使空代码。

1. 【强制】**jQuery对象变量命名必须带$符号**

|  |
| --- |
| var $btn = $('.btn'); |

1. 【强制】**eval函数不要使用，不安全、效率低，使用$.parseJSON()**
2. 【推荐】**多使用面向对象的方式编程，解决混乱问题**

|  |
| --- |
| /\* 错误 \*/  // 增加  function add() {};  // 移除  function remove() {};  // 编辑  function edit() {};  ...  /\* 正确 \*/  var operate = {  add: function() { //增加  },  remove: function() { // 移除  },  edit: function() { // 编辑  }  }; |

1. 【推荐】**如果一个函数内部代码非常复杂，则要考虑拆分成多个函数**
2. 【强制】**禁止频繁操作dom**

|  |
| --- |
| /\* 错误 \*/  for (var i = 0; i < 100; i++) {  $('#list').append('<li class="item-' + i + '">test</li>');  }  /\* 正确 \*/  var liStr = '';  for (var i = 0; i < 100; i++) {  liStr += '<li class="item-' + i + '">test</li>';  }  $('#list').append(liStr); |

1. 【推荐】**代码重复率很高就意味可以优化**
2. 【推荐】**当判断条件只有if else两种情况时可以用三元条件代替**

|  |
| --- |
| /\* 错误 \*/  if (a == 10) {  return 'success';  } else {  return 'failure';  }  /\* 正确 \*/  return a == 10 ? 'success' : 'failure'; |

1. 【推荐】**表达异常的分支时，少用if else方式**

|  |
| --- |
| if (!params.imei) {  alert('请输入imei');  } else if (!params.name) {  alert('请输入名称');  } else {  ajax({  ...  });  } |

这种方式可以改写成：

|  |
| --- |
| if (!params.imei) {  alert('请输入imei');  return;  }  if (!params.name) {  alert('请输入名称');  return;  }  ajax({  ...  }); |

1. 【强制】**用数组和对象字面量代替数组和对象构造器**

|  |
| --- |
| /\* 错误 \*/  var arr = new Array();  var obj = new Object();  /\* 正确 \*/  var arr = [];  var obj = {}; |

1. 【推荐】**True 和False 布尔表达式**

下面的布尔表达式都返回false:

* null
* undefined
* ' ' 空字符串
* 0 数字0

但小心下面的, 可都返回true:

* '0' 字符串0
* [] 空数组
* {} 空对象

|  |
| --- |
| /\* 错误 \*/  //比较糟糕的代码  while (x != null) {}  //检查字符串是否为null 或空  if (y != null && y != '') {}  /\* 正确 \*/  while (x) {}  if (y) {} |

* Boolean('0') == true
* '0' != true
* 0 != null
* 0 == []
* 0 == false
* Boolean(null) == false
* null != true
* null != false
* Boolean(undefined) == false
* undefined != true
* undefined != false
* Boolean([]) == true
* [] != true
* [] == false
* Boolean({}) == true
* {} != true
* {} != false

1. 【推荐】**&&和||**

|  |
| --- |
| /\* 错误 \*/  //写得太复杂  function foo(opt\_win) {  var win;  if (opt\_win) {  win = opt\_win;  } else {  win = window;  }  }  //写得太复杂  if (node) {  if (node.kids) {  if (node.kids[index]) {  foo(node.kids[index]);  }  }  }  /\* 正确 \*/  function foo(opt\_win) {  var win = opt\_win || window;  }  if (node && node.kids && node.kids[index]) {  foo(node.kids[index]);  }  或  var kid = node && node.kids && node.kids[index];  if (kid) {  foo(kid);  } |

1. 【推荐】**==与===的使用，相同类型的数据比较用===，不同数据类型比较用==**

|  |
| --- |
| /\* 错误 \*/  if (1 == 1) {}  /\* 正确 \*/  if (1 === 1) {}  if ('1' == 1) {} |

* **Vue篇**

1. 【强制】**规范说明**
2. **组件集成**

* 组件统一存放路径 ./src/components
* 组件命名统一使用小驼峰，例： loginForm
* 组件导出统一通过 index.js 文件导出 ，例如 loginForm 组件：

|  |
| --- |
| |-- loginForm  |-- index.js  |-- loginForm.vue  // index.js  import LoginForm from './loginForm.vue';  export default LoginForm; |

1. **插件集成**

* 插件统一存放路径 ./src/plugins
* 插件命名统一使用小驼峰(第三方插件用默认名称)，例： loginForm
* 插件导出统一通过 index.js 文件导出 ，例如 loading 组件

|  |
| --- |
| |-- loading  |-- index.js  |-- loading.vue |

1. 【强制】**项目目录结构**

|  |
| --- |
| Project  ├── build 构建脚本目录  │ ├── build.js 生产环境构建（编译打包）脚本  │ ├── check-version.js 检查node、npm等版本  │ ├── dev-client.js 开发服务器热重载脚本，主要用来实现开发阶段页面自动刷新  │ ├── dev-server.js 运行本地开发服务器  │ ├── utils.js 构建相关工具方法，主要用来处理css类文件的loader  │ ├── vue-loader.conf.js 处理vue中的样式  │ ├── vebpack.base.conf.js webpack基本配置  │ ├── vebpack.dev.conf.js webpack开发环境配置  │ └── vebpack.prod.conf.js webpack生产环境配置  │  ├── config 项目配置  │ ├── dev.env.js 开发环境变量  │ ├── index.js 项目配置文件  │ └── prod.env.js 生产环境变量  │  ├── dist 使用生产环境配置构建项目，构建好的目录放到该目录  │ ├── dev.env.js 开发环境变量  │ ├── index.js 项目配置文件  │ ├── prod.env.js 生产环境变量  │ └── test.env.js 测试环境变量  │  ├── node\_modules node的依赖包  │  ├── src 项目源码目录  │ ├── api  │ │ └── api.js 接口文件  │ ├── assets 资源目录，这里的资源会被webpack构建  │ │ ├── css 第三方css文件，全局css文件  │ │ ├── fonts 字体  │ │ ├── less 全局less  │ │ ├── sass 全局sass  │ │ └── images 图片  │ │  │ ├── components 组件目录  │ │ ├── loading  │ │ │ ├── index.js  │ │ │ └── loading.vue  │ │ └── trees  │ │ 　├── index.js  │ │ 　└── trees.vue  │ │  │ ├── directive 自定义指令  │ │  │ ├── http  │ │ └── http.js 接口请求封装  │ │  │ ├── libs 工具箱  │ │ └── util.js 常用工具类  │ │  │ ├── local 语言配置  │ │ ├── index.js  │ │ └── lang  │ │ 　├── en-US.js  │ │ 　├── zh-CN.js  │ │ 　└── zh-TW.js  │ │  │ ├── mock mock数据目录  │ │  │ ├── views 页面目录  │ │  │ ├── router 前端路由  │ │ └── index.js  │ │  │ ├── store vuex状态管理  │ │ ├── index.js  │ │ ├── actions.js 根级别的 action  │ │ ├── mutations.js 根级别的 mutations  │ │ └── modules  │ │ 　├── xxx.js xxx模块  │ │ 　└── yyy.js yyy模块  │ │  │ ├── App.vue 根组件  │ └── main.js 入口js文件  │  ├── static 纯静态资源，不会被webpack构建  │  ├── index.html 入口页面  │  ├── .babelrc ES6语法编译配置  │  ├── .editorconfig 定义代码格式  │  ├── .gitignore git 上传需要忽略的文件  │  ├── .postcssrc.js 指定使用的 css 预编译器，默认配置了 autoprefixer  │  ├── package.json 项目基本信息  │  └── README.md 项目介绍 |

1. 【强制】**属性书写顺序**

|  |
| --- |
| **1.注册**  ref="test"  class="test" :class="className"  **2. 显示隐藏**  v-for="(item, key) in data" || v-if="isShow" || v-else-if="isHide" || v-else  :key="key" || key="test"  **3.数据**  v-model="formInline.user" || :data="bindData"  **4.属性**  此属性数量较多，不强制顺序，可根据自己理解的重要级别排序  **5.方法-强制顺序**  // vue 方法  @click="handleClick"  // 自定义方法  @my-self-func="handleMySelfFunc"  // element 方法  @handle-change-page="handleChangePage"  **6.测试属性**  data-automation="test" |

* **svn和git提交篇**

1. 【强制】**提交必须填写说明，禁止内容为空**

* **自动化测试篇**

1. 【强制】**定制data-automation属性为自动化专用属性，其作用代替id或class，为保证其唯一性，命名风格采用驼峰命名法**

主体布局(头部、主体、底部、侧栏)、选项卡、表单、搜索、表格、翻页等功能模块增加全局唯一性标识；

|  |  |
| --- | --- |
| 头部 | data-automation="header" |
| 主体 | data-automation="main" |
| 底部 | data-automation="footer" |
| 侧栏 | data-automation="sidebar" |

更多命名规则详见HTML篇第28条‘功能模块命名’

1. 【强制】**表单元素命名规律统一**

|  |  |  |  |
| --- | --- | --- | --- |
| **按钮** | | **文本框/文本域** | |
| *格式* | *btn-xxx* | *格式* | *form-xxx* |
| 添加 | btn-add | 文本框 | form-control |
| 修改 | btn-edit | 文本域 | form-textarea |
| 删除 | btn-delete |  |  |
| 保存 | btn-save |  |  |
| 搜索 | btn-search |  |  |
| 重置 | btn-reset |  |  |

1. 【强制】**相同功能类型的组件结构和命名保持一致**

常见的设备树、客户树