前端开发规范与说明

WEB客户端开发自成体系， 主要用于智能终端（iPhone、Android手机、iPad、Android Pad）和传统PC的开发。JS规范、HTML规范和CSS规范对客户端开发进行全方位指导，统一编码规范、提高可读性、降低维护成本。

一般规则

应用在 HTML, JavaScript 和 CSS上的通用规则。

1. 文件、资源命名
   * 以可读性而言，中划线用来分隔文件名
   * 确保文件命名总是以字母开头而不是数字
   * 特殊含义的文件，需要对文件增加前后缀或特定的扩展名（比如 .min.js, .min.css），抑或一串前缀（比如 all.main.min.css）。使用点分隔符来区分这些在文件名中带有清晰意义的元数据
2. 文本缩进:一次缩进4个空格
3. 代码检查
   * 对于前端JavaScript这种比较宽松自由的编程语言来说，严格遵循编码规范和格式化风格指南极为重要。前端开发人员需严格遵循开发规范。如果需要，公司会使用自动代码检查工具（如eslint）降低语法错误，确保代码正确执行。

**HTML** 规范

1. 文档类型 **HTML5 docType**  
   使用 HTML5的文档类型申明： <!DOCTYPE html>  
   html5不基于SGML,因此不需要对DTD进行引用，但是需要doctype来规范浏览器的行为（让浏览器按照他们应该的方式来运行）而HTML4.01基于SGML，所以需要对DTD进行引用，才能告知浏览器文档所使用的文档类型。
2. **media** 标签  
   <meta name="viewport" content="width=device-width,initial-scale=1.0,maximum-scale=1.0,user-scalable=no">  
   &lt;meta name="format-detection" content="telephone=no" /&gt; // 禁止数字识自动别为电话号码
3. 语言属性（**Language attribute**）  
   强烈建议为 html 根元素指定 lang 属性，从而为文档设置正确的语言。这将有助于语音合成工具确定其所应该采用的发音，有助于翻译工具确定其翻译时所应遵守的规则等等。更多关于 lang 属性的知识可以从 此规范 中了解。  
   [HTML 语言代码参考手册](http://www.w3school.com.cn/tags/html_ref_language_codes.asp)上的文章可以获得更多有用的信息。
4. 字符编码  
   通过明确声明字符编码，能够确保浏览器快速并容易的判断页面内容的渲染方式。这样做的好处是，可以避免在 HTML 中使用字符实体标记（character entity），从而全部与文档编码一致（一般采用 UTF-8 编码）。
5. **IE**兼容模式  
   IE 支持通过特定的 标签来确定绘制当前页面所应该采用的 IE 版本。除非有强烈的特殊需求，否则最好是设置为 edge mode，从而通知 IE 采用其所支持的最新的模式。  
   &lt;meta http-equiv=“X-UA-Compatible” content=“IE=edge”&gt; "IE=edge""IE=11""IE=EmulateIE11""IE=10""IE=EmulateIE10""IE=9""IE=EmulateIE9"IE=8""IE=EmulateIE8""IE=7""IE=EmulateIE7""IE=5"
6. 注释  
     
   <!-- header -->
7. 引入**CSS**和**JAVASCRIPT**  
   【推荐】根据 HTML5 规范，在引入 CSS 和 JavaScript 文件时一般不需要指定 type 属性，因为 text/css 和 text/javascript 分别是它们的默认值。
8. 语法
9. 属性顺序  
   【参考】HTML 属性应当按照以下给出的顺序依次排列，确保代码的易读性。
   * class
   * id, name
   * data-\*
   * src, for, type, href
   * title, alt
   * aria-\*, role
10. 【参考】class 用于标识高度可复用组件，因此应该排在首位。id 用于标识具体组件，应当谨慎使用（例如，页面内的书签），因此排在第二位。
11. 语义化标签  
    【推荐】根据元素（有时称作“标签”）其被创造出来时的初始意义来使用它,有根据有目的地使用 HTML 元素，对于可访问性、代码重用、代码效率来说意义重大。
12. 多媒体回溯**:**  
    【参考】对页面上的媒体而言，像图片、视频、canvas 动画等，要确保其有可替代的接入接口
13. 关注点分离**:**  
    web 中的关注点包括信息（HTML 结构）、外观（CSS）和行为（JavaScript）。为了使它们成为可维护的干净整洁的代码，必须将它们分离开。严格地保证结构、表现、行为三者分离，并使三者之间没有太多的交互和联系.就是说，尽量在文档和模板中只包含结构性的 HTML；而将所有表现代码，移入样式表中；将所有动作行为，移入脚本中;在此之外，为使得它们之间的联系尽可能的小，在文档和模板中也尽量少地引入样式和脚本文件
    * 【推荐】合并样式，不引用过多样式表
    * 【推荐】合并脚本，不使用过多脚本
    * 【推荐】不使用行内样式
    * 【推荐】不在元素上使用style 属性
    * 【推荐】不使用行内脚本
    * 【推荐】不使用表象元素
    * 【推荐】不使用表象 class 名（red, left, center）
14. **type**属性  
    【推荐】省略样式表与脚本上的 type 属性。鉴于 HTML5 中以上两者默认的 type 值就是 text/css 和text/javascript，所以 type 属性一般是可以忽略掉的。在老旧版本的浏览器中这么做也是安全可靠的。
15. **ID**和锚点  
    【参考】在利用锚点提高用户体验方面，一个比较好的做法是将页面内所有的头部标题元素都加上 ID。页面 URL 的 hash 中带上对应的 ID 名称，即形成描点，方便跳转至对应元素所处位置。  
    【参考】例如，在浏览器中输入URL（带有锚点）时，浏览器将定位至锚点对应元素位置。
16. **HTML**引号  
    【强制】使用双引号(“”)而不是单引号(‘’)
17. 实用为王  
    【推荐】尽量遵循 HTML 标准和语义，但是不要以牺牲实用性为代价。任何时候都要尽量使用最少的标签并保持最小的复杂度。

**CSS** 规范说明

文件规范

1. 所有文件均归档至约定的目录中：
2. 业务类的CSS是指和具体产品相关的文件，放在如下目录中：
   * 多模块根据页面模块创建文件夹，
   * 只有一个模块情况可以放入CSS文件夹
3. 文件名、文件编码及文件大小
4. React 架构中 所有css(less)文件放入同一个文件夹style下面

**CSS**注释规范

1. 【推荐】文件顶部注释  
   /\*\* @description:中文说明\* @author：name\* @update 2017-01-01 14:00\*/
2. 【推荐】模块注释：模块注释必须单独写在一行  
   /\* module: module2 by 张三 \*/Codes/\* module: module2 by 张三 \*/
3. 【推荐】单行注释与多行注释,单行注释可以写在单独一行，也可以写在行尾，注释中的每一行长度不超过40个汉字，或者80个英文字符。
4. 【推荐】特殊注释：用于标注修改、待办等信息  
   /\* TODO: xxxx by name 2013-04-13 18:32 \*/Codes/\* BUGFIX: xxxx by name 2012-04-13 18:32 \*/
5. 【参考】区块注释  
   /\* Header \*//\* Footer \*//\* leftside \*/

**CSS**命名规范

ID和class(类)名使用可以反应元素目的和用途的名称，或其他通用名称。使用具体且反映元素目的的名称，这些是最容易理解的，而且发生变化的可能性最小。使用连字符（中划线）分隔命名中的单词。为了增强课理解性，在选择器中不要使用除了连字符（中划线）以为的任何字符（包括没有）来连接单词和缩写。另外，作为该标准，预设属性选择器能识别连字符（中划线）作为单词[attribute|=value]的分隔符。

1. 【强制】尽可能提高代码模块的复用，样式尽量用组合的方式
2. 【强制】命名避免使用中文拼音，应该采用更简明有语义的英文单词进行组合,应该用意义命名，而不是样式显示结果命名;不要用抽象的晦涩的命名.
3. 【强制】规则命名中，一律采用小写加中划线的方式，不允许使用大写字母或\_、不允许通过1、2、3等序号进行命名
4. 【推荐】命名注意缩写，但是不能盲目缩写
5. 【推荐】ID命名要注意明确性及唯一性,不要随意新建id
6. 【推荐】class命名要注意通用性及复用性,命名必须言简意赅
7. 【推荐】避免class与id重名

声明顺序

1. 【参考】一，Positioning定位：可以使一个元素脱离正常文本流，并且覆盖盒模型相关的样式
2. 【参考】二，Box model 盒模型：决定了一个组件的大小和位置
3. 【参考】三，Typographic 排版：
4. 【参考】四，Visual 外观：

**CSS**代码格式

1. 排版规范
   * 【强制】使用2个空格，而不使用tab或者混用空格+tab作为缩进
   * 【强制】规则可以写成单行，或者多行，但是整个文件内的规则排版必须统一
   * 【强制】多个selector共用一个样式集，则多个selector必须写成多行形式 ；
   * 【强制】每一条规则结束的大括号 } 必须与规则选择器的第一个字符对齐；
   * 【推荐】写成单行时每一条规则的大括号 { 前后加空格，每一条规则结束的大括号 } 前加空格；
   * 【推荐】属性名冒号之前不加空格，冒号之后加空格；
   * 【推荐】每一个属性值后必须添加分号; 并且分号后空格；
2. 规则书写规范
   * 【强制】使用单引号，不允许使用双引号;
   * 【强制】除16进制颜色和字体设置外，CSS文件中的所有的代码都应该小写;
   * 【强制】除了重置浏览器默认样式外，禁止直接为html tag添加css样式设置;
   * 【强制】每一条规则应该确保选择器唯一，禁止直接为全局.nav、.header、.body等类设置属性;
3. 代码性能优化
   * 【推荐】合并margin、padding、border的-left/-top/-right/-bottom的设置，尽量使用短名称。
   * 【推荐】选择器应该在满足功能的基础上尽量简短，减少选择器嵌套，查询消耗。但是一定要避免覆盖全局样式设置。
   * 【推荐】注意选择器的性能，不要使用低性能的选择器。
   * 【推荐】禁止在css中使用\*选择符。
   * 【推荐】除非必须，否则，一般有class或id的，不需要再写上元素对应的tag。
   * 【推荐】0后面不需要单位，比如0px可以省略成0，0.8px可以省略成.8px。
   * 【推荐】如果是16进制表示颜色，则颜色取值应该大写，如果可以，颜色尽量用三位字符表示，例如#AABBCC写成#ABC 。
   * 【推荐】如果没有边框时，不要写成border:0，应该写成border:none 。
   * 【推荐】尽量避免使用AlphaImageLoader
   * 【推荐】在保持代码解耦的前提下，尽量合并重复的样式。
   * 【推荐】background、font等可以缩写的属性，尽量使用缩写形式 。
4. **CSS Hack**的使用  
   请不用动不动就使用浏览器检测和CSS Hacks，先试试别的解决方法吧！考虑到代码高效率和易管理，虽然这两种方法能快速解决浏览器解析差异，但应被视为最后的手段。在长期的项目中，允许使用hack只会带来更多的hack，你越是使用它，你越是会依赖它！
   * 区别属性： IE6 -- \_property:value IE6/7 -- \*property:value IE6/7/8/9 -- property:value\9
   * 区别规则： IE6 -- \* html select {...} IE7 -- \*:fist-child+html selector {...} 非IE6 -- html>body selector {...} firefox only -- @-moz-document url-prefix() {...}
5. 字体规则
   * 为了防止文件合并及编码转换时造成问题，建议将样式中文字体名字改成对应的英文名字，如：黑体(SimHei) 宋(SimSun) 微软雅黑 (Microsoft Yahei，几个单词中间有空格组成的必须加引号)
   * 为了对font-family取值进行统一，更好的支持各个操作系统上各个浏览器的兼容性，font-family不允许在业务代码中随意设置

**CSS**编码技巧

1. 【参考】尽量减少代码重复
2. 【参考】合理使用简写
3. 【参考】是否应该使用预处理器？
4. 【参考】层级(z-index)必须清晰明确，页面弹窗、气泡为最高级（最高级为999），不同弹窗气泡之间可在三位数之间调整；普通区块为10-90内10的倍数；区块展开、弹出为当前父层级上个位增加，禁止层级间盲目攀比。

**CSS** 常用工具

1. W3C CSS validator：http://jigsaw.w3.org/css-validator/
2. CSS Lint：http://csslint.net/
3. CSS Usage：https://addons.mozilla.org/en-us/firefox/addon/css-usage/

**JS** 规范

**JS**文件规范

1. 文件编码统一UTF-8
2. 使用严格模式（use strict）编写代码:ECMAScript 5 引入严格模式('strict mode')概念。通过严格模式，在函数内部选择进行较为严格的全局或局部的错误条件检测，使用严格模式的好处是可以提早知道代码中的存在的错误，及时捕获一些可能导致编程错误的ECMAScript行为。在开发中使用严格模式能帮助我们早发现错误。设立"严格模式"的目的，主要有以下几个：错误检测、规范、效率、安全、面向未来.
   * 消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为;
   * 消除代码运行的一些不安全之处，保证代码运行的安全；
   * 提高编译器效率，增加运行速度
   * 为未来新版本的Javascript做好铺垫

**JS**注释规约

1. 【强制】类，类属性，类方法使用/\*\*内容\*/格式，不得使用// xxx方式
2. 【强制】方法内部单行注释，在被注释语句上方另起一行，使用//注释。方法内部多行注释使用/\* \*/注释，注意与代码对齐。
3. 【推荐】代码修改同时，注释也要进行相应修改，尤其是参数、返回值、核心逻辑等的修改。说明：代码与注释更新不同步，就像路网与导航软件更新不同步一样，如果导航软件严重滞后，就失去了导航的意义。
4. 【参考】谨慎注释掉代码。在上方详细说明，而不是简单地注释掉。如果无用，则删除;说明：代码被注释掉有两种可能性：1）后续会恢复此段代码逻辑。2）永久不用。前者如果没有备注信息，难以知晓注释动机。后者建议直接删掉（代码仓库保存了历史代码）。
5. 【参考】对于注释的要求：
   * 第一、能够准确反应设计思想和代码逻辑
   * 第二、能够描述业务含义，使别的程序员能够迅速了解到代码背后的信息。完全没有注释的大段代码对于阅读者形同天书，注释是给自己看的，即使隔很长时间，也能清晰理解当时的思路；注释也是给继任者看的，使其能够快速接替自己的工作。
6. 【参考】好的命名、代码结构是自解释的，注释力求精简准确、表达到位。避免出现注释的一个极端：过多过滥的注释，代码的逻辑一旦修改，修改注释是相当大的负担
7. 【参考】特殊注释标记，请注明标记人与标记时间。注意及时处理这些标记，通过标记扫描，经常清理此类标记。线上故障有时候就是来源于这些标记处的代码。
   * 待办事宜（TODO）:（ 标记人，标记时间，[预计处理时间]） 表示需要实现，但目前还未实现的功能。
   * 错误，不能工作（FIXME）:（标记人，标记时间，[预计处理时间]） 在注释中用FIXME标记某代码是错误的，而且不能工作，需要及时纠正的情况。

**JS**命名规范

1. 【强制】文件夹统一使用全小写
2. 【强制】代码中命名不能以下划线或美元符开始，也不能以下划线或美元符结束
3. 【强制】代码中严禁使用拼音与英文混合的方式，更不允许直接使用中文方式。纯拼音命名方式也要避免采用（国际通用的名称可视为英文，如：taobao，alibaba等）
4. 【强制】类名使用UpperCamelCase风格
5. 【强制】方法名、参数名、成员变量、局部变量都统一使用lowerCamelCase风格，必须遵从驼峰形式 如：localValue，getMessage()
6. 【强制】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
7. 【强制】杜绝完全不规范的缩写，避免望文不知义
8. 【推荐】为了达到代码自解释的目标，任何定义编程元素在命名时使用尽量完整单词组合来表达其意

**JS**代码风格规范

1. 【强制】大括号使用约定:如果大括号内容为空则简洁的写成{}即可，不需要换行；如果非空代码块则：
   * 左大括号前不换行
   * 左大括号后换行
   * 右大括号前换行
   * 右大括号后还有else等代码则不换行
2. 【强制】左小括号和字符之间不出现空格；同样，右小括号和字符之间也不出现空格
3. 【强制】if/for/while/switch/do等保留字与括号之间都必须加空格
4. 【强制】任何二目、三目运算符的左右两边都需要加一个空格；如：settings = settings ? settings : {}; if (a && flag == 1) {}
5. 【强制】注释的双斜线与注释内容之间有且仅有一个空格; 如：// 注释内容，注意在//和注释内容之间有一个空格
6. 【强制】单行字符数限不超过 120 个，超出需要换行，超出需要换行时遵循如下原则：
   * 第二行相对第一行缩进2空格，从第三行开始，不再继续缩进
   * 运算符与下文一起换行
   * 方法调用是，多个参数需要换行时，在逗号后进行
   * 在括号前不要换行
7. 【强制】方法参数在定义和传入是，多个参数逗号后面加空格;如：Function Sum(a, b, c){} Sum(1, 2, 3);

**JS**常量定义规范

1. 【强制】不允许任何魔法值（即未经定义的常量）直接出现在代码中
2. 【推荐】不要使用一个常量类维护所有常量，按常量功能进行归类，分开维护
3. 【推荐】常量复用层次，公共常量、模块常量、功能页面常量

**JS**控制语句规范

1. 【强制】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，即是他什么代码也没有。
2. 【强制】在if/else/for/while/do语句中必须使用大括号。即使只有一行代码，避免采用单行的编码方式：if (condition) statements;
3. 【推荐】表达异常的分支时，少用if-else方式，这种方式可以改写成，如果非得使用避免后续代码维护困难，请勿超过3层。
4. 【推荐】不要在条件判断中执行其它复杂的语句，将复杂逻辑判断的结果赋值给一个有意义的布尔变量名，以提高可读性（很多if语句内的逻辑相当复杂，阅读者需要分析条件表达式的最终结果，才能明确什么样的条件执行什么样的语句，那么，如果阅读者分析逻辑表达式错误呢？ ）
5. 【推荐】循环体中的语句要考量性能