|  |
| --- |
| **友云音前端开发规范** |
|  |
|  |
| 命名规范 HTML规范 CSS规范 JS规范 多屏适配规范 性能优化规范 测试规范 |
|  |
| **友云音前端组** |
| **2017/11/13** |
|  |

**解释权**

文档摘要

|  |  |
| --- | --- |
| 文件名 | 友云音前端开发规范.docx |
| 最新版本 | 1.0 |
| 文档创建者 | 袁凯 |
| 文档维护者 |  |
| 文档评审者 |  |
| 定稿日期 |  |
| 文档目标 |  |

版本管理

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **描述** | **日期** | **作者** |
| 1.0 | 创建 | 2017-11-03 | 袁凯 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1. 规范目的 4](#_Toc498371170)

[2. 基本准则 4](#_Toc498371171)

[3. 命名规范 4](#_Toc498371172)

[3.1 HTML文件命名 4](#_Toc498371173)

[3.2 CSS文件命名 4](#_Toc498371174)

[3.3 Javascript文件命名 4](#_Toc498371175)

[4. HTML规范 5](#_Toc498371176)

[4.1 语法 5](#_Toc498371177)

[4.2 doctype 5](#_Toc498371178)

[4.3 meta标签 5](#_Toc498371179)

[4.4 字符编码 5](#_Toc498371180)

[4.5 IE兼容模式 5](#_Toc498371181)

[4.6 引入CSS JS 6](#_Toc498371182)

[4.7 属性顺序 6](#_Toc498371183)

[4.8 注释 7](#_Toc498371184)

[5. CSS规范 7](#_Toc498371185)

[5.1 语法 7](#_Toc498371186)

[5.2 注释 7](#_Toc498371187)

[5.3 命名 7](#_Toc498371188)

[5.4 属性顺序声明 8](#_Toc498371189)

[5.5 属性简写 9](#_Toc498371190)

[5.6 SASS相关 9](#_Toc498371191)

[6. JS规范 9](#_Toc498371192)

[6.1 语法 9](#_Toc498371193)

[6.2 注释 10](#_Toc498371194)

[6.3 变量 10](#_Toc498371195)

[6.4 函数 10](#_Toc498371196)

[6.5 图表 10](#_Toc498371197)

[7. 屏幕适配规范 10](#_Toc498371198)

[8. 测试规范 11](#_Toc498371199)

[8.1 浏览器兼容 11](#_Toc498371200)

[8.2 完成度测试 12](#_Toc498371201)

[9. 协议规范 12](#_Toc498371202)

[10. 性能优化规范 12](#_Toc498371203)

[10.1 JS层： 12](#_Toc498371204)

[10.2 CSS层： 13](#_Toc498371205)

[10.3 HTML层 14](#_Toc498371206)

[10.4 配置层 14](#_Toc498371207)

[11. 参考文献 14](#_Toc498371208)

[12. 常用组件命名 14](#_Toc498371209)

# 规范目的

为提高团队协作效率， 便于后台人员添加功能及前端后期优化维护， 输出高质量的文档， 特制订此文档。 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。 本文档如有不对戒者不合适的地方请及时提出，经讨论决定后方可更改。

# 基本准则

符合web标准，语义化html， 结构、表现、行为分离， 兼容性优良。 页面性能方面， 代码要求简洁明了有序， 尽可能的减小服务器负载， 保证最快的解析速度。基本的代码编写规范使用ESLint Standard标准进行规范。其余条目按照友云音专有规范进行编写。

# 命名规范

## HTML文件命名

英文命名，后缀为“.html”。同时将对应UE设计稿放于同目录中，方便测试人员使用。

## CSS文件命名

英文命名，后缀为”.css”。

共用： base.css

首页： home.css

框架样式：layoout.css

其它页面依实际模块需求命名。

## Javascript文件命名

英文命名。

共用：app.js

其他依实际模块需求命名

# HTML规范

## 语法

1. 所有缩进使用tab进行缩进，不允许存在空格。

2. 对于属性的定义，确保全部使用双引号，绝不要使用单引号。

3. 所有编码均遵循xhtml标准，所有标签必须闭合，包括br (<br />) 、hr(<hr />)等。

4. 充分利用无兼容性问题的html 自身标签，比如span 、em 、strong、optgroup、label 等等。

5. 语义化html， 如 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。

6. 在页面中尽量避免使用style 属性，即style=”…”。

7. 给区块代码及重要功能（比如循环）加上注释，方便后台添加功能 。

8. 特殊符号使用：尽可能使用代码替代，比如 <( &lt; )、>( &gt; )、空格( &nbsp; )、»( &raquo; ) 等等。

## doctype

1. 使用 HTML5 的 doctype 来启用标准模式，建议使用大写的 DOCTYPE。

## meta标签

1. meta 标签http-equiv属性的顺序：

Content-Type -> Content-Language -> ROBOTS -> keywords -> descrition

## 字符编码

1. 编码统一为：<metahttp-equiv=”Content-Type” content=”text/html; charset=utf-8” />。

## IE兼容模式

1. 启用 IE Edge 模式：<meta http-equiv="X-UA-Compatible" content="IE=Edge">

## 引入CSS JS

1. 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。

<!-- External CSS -->

<link rel="stylesheet" href="code-guide.css">

<!-- In-document CSS -->

<style>

/\* ... \*/

</style>

<!-- JavaScript -->

<script src="code-guide.js"></script>

2. 引入JS 库文件，文件名须包含库名称及版本号及是否为压缩版，比如jquery-1.4.1.min.js；引入插件，文件名格式为库名称+插件名称，比如jQuery.cookie.js。

3.不在页面内部通过内联方式添加CSS。即禁止使用style标签添加CSS。

## 属性顺序

1. 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

<a class="..." id="..." data-modal="toggle" href="#">

Example link

</a>

<input class="form-control" type="text">

<img src="..." alt="...">

## 注释

# CSS规范

## 语法

1. 十六进制值应该全部小写，尽量使用全写形式的十六进制值。例如，#ffffff。

2. 避免为0值指定单位。例如，用 margin: 0; 代替 margin: 0px;。

3. 大的框架统一采用“wrapper”、“header”、“footer”、“content”、“ sidebar” 。避免使用中文拼音。

4. 规避class与id命名。

5. 使用table标签时(尽量避免使用table标签)，不要用width、height、cellspacing、cellpadding等table属性直接定义表现。

6. 减少使用影响性能的属性，比如position:absolute || float

7. 省略“0”值后面的单位。不要在0值后面使用单位，除非有值。

## 注释

1. 必须为大区块样式添加注释，小区块适量注释。对于较长的注释，务必书写完整的句子；对于一般性注解，可以书写简洁的短语。

/\* Bad example \*/

/\* Modal header \*/

.modal-header {

...

}

/\* Good example \*/

/\* Wrapping element for .modal-title and .modal-close \*/

.modal-header {

...

}

## 命名

1.命名风格使用bootstrap前缀式类名。如.btn .btn-success .panel-base。

2.基于最近的父class或基本（base）class作为新class的前缀。如.btn-base-subbtn。

/\* Bad example \*/

.t { ... }

.red { ... }

.header { ... }

/\* Good example \*/

.tweet { ... }

.important { ... }

.tweet-header { ... }

## 属性顺序声明

1.相关的属性声明应当归为一组，并按照下面的顺序排列：

Positioning

Box model

Typographic

Visual

1. .declaration-order {
2. /\* Positioning \*/
3. position: absolute;
4. top: 0;
5. right: 0;
6. bottom: 0;
7. left: 0;
8. z-index: 100;
9. /\* Box-model \*/
10. display: block;
11. float: right;
12. width: 100px;
13. height: 100px;
14. /\* Typography \*/
15. font: normal 13px "Helvetica Neue", sans-serif;
16. line-height: 1.5;
17. color: #333;
18. text-align: center;
19. /\* Visual \*/
20. background-color: #f5f5f5;
21. border: 1px solid #e5e5e5;
22. border-radius: 3px;
23. /\* Misc \*/
24. opacity: 1;
25. }

## 属性简写

1. 避免过度任意的简写。如.btn代表button，但是.s不能代表任何含义。

## SASS相关

1.公共模块使用@include来调用具有相同名称的mixin模块。



# JS规范

## 语法

1. 每行代码结束必须有分号。

2. 当需要缓存this时必须推荐使用self变量进行缓存。

## 注释

1. 单行注释使用C++风格： // 这儿是单行注释

2. 多行注释使用C风格 /\* 这儿有多行注释 \*/

## 变量

1. 变量命名：驼峰式命名，原生JavaScript变量要求是纯英文字母，首字母须小写。

2. jQuery变量要求首字符为“\_”。

3. 要求变量集中声明，避免全局变量。

4. 私有属性、变量和方法以下划线开头。var \_privateProperty。

5. 常量, 使用全部字母大写，单词间下划线()分隔的命名方式,通常用于ajax请求url，和一些不会改变的数据var PLACE\_TYPE = 'hotel'。

6. 总是使用 var（如果用ES6 则let ） 来声明变量。

## 函数

1. 函数命名：首字母小写驼峰式命名。

## 图表

1. 各个页面引入图表时，先取内置的公共图表样式，然后根据页面需要扩展配置进行生成。

# 屏幕适配规范

适配主要满足两件事：

1.各种分辨率的PC端屏幕显示一致。

2.窗口进行缩放时显示一致。

友云音项目中目前涉及两类。

1. 宽度适配

采用的方式为bootstrap网格适配。



所有布局列出所有网格。比如



2. 图表适配

后续图表通过调用统一方法，获取该图表grid内设置，然后根据不同分辨率进行适配显示。

# 测试规范

## 浏览器兼容

建议测试顺序：FireFox –> IE11 –> IE10 –> IE9 –> Chrome–> Opera，建议安装firebug及IE Tab Plus插件。

浏览器兼容优先顺序：IE11 –> IE10 –> IE9 –> FireFox –> Chrome –> Opera。



## 完成度测试

1. 页面完成后需要UE配合进行当前页面的完成度测试，给出评估。

# 协议规范

统一采用网关访问协议。URL使用相对路径。

# 性能优化规范

## JS层：

1. 尽量避免使用存在兼容性及消耗资源的方法戒属性，比如eval( ) & innerText。

2. 避免不必要的 DOM 操作

浏览器遍历 DOM 元素的代价是昂贵的。最简单优化 DOM 树查询的方案是，当一个元素出现多次时，将它保存在一个变量中，就避免多次查询 DOM 树了。

// Recommended

**var** myList = "";

**var** myListHTML = document.getElementById("myList").innerHTML;

**for** (**var** i = 0; i < 100; i++) {

myList += "<span>" + i + "</span>";

}

myListHTML = myList;

// Not recommended

**for** (**var** i = 0; i < 100; i++) {

document.getElementById("myList").innerHTML += "<span>" + i + "</span>";

}

3. 缓存数组长度

循环无疑是和 JavaScript 性能非常相关的一部分。通过存储数组的长度，可以有效避免每次循环重新计算。

注: 虽然现代浏览器引擎会自动优化这个过程，但是不要忘记还有旧的浏览器。

**var** arr = **new** Array(1000),

len, i;

// Recommended - size is calculated only 1 time and then stored

**for** (i = 0, len = arr.length; i < len; i++) {

}

// Not recommended - size needs to be recalculated 1000 times

**for** (i = 0; i < arr.length; i++) {

}

## CSS层：

1. 对于通用元素使用 class选择器 ，这样利于渲染性能的优化。

2. 对于经常出现的组件，避免使用属性选择器（例如，[class^="..."]）。浏览器的性能会受到这些因素的影响。

3. 选择器要尽可能短，并且尽量限制组成选择器的元素个数，建议不要超过 3 。

/\* Bad example \*/

span { ... }

.page-container #stream .stream-item .tweet .tweet-header .username { ... }

.avatar { ... }

/\* Good example \*/

.avatar { ... }

.tweet-header .username { ... }

.tweet .avatar { ... }

## HTML层

1. 减少标签数量

/\* Bad example \*/

<div class=”box”><div class=”welcome”>欢迎访问XXXX：<div class=” integration”>9527 </div></div></div>

/\* Good example \*/

<div class=”box”>

<p>欢迎访问XXXXX网站<span>9527</span></p>

2.避免浏览器发生回流。

## 配置层

1. 路由配置时，按需自定义路由模块，将路由表进行拆分，提高解析速度。

# 参考文献

1. <https://wenku.baidu.com/view/b613ecd476a20029bd642d9c.html> 常见规范

2. <http://www.runoob.com/bootstrap/bootstrap-css-codeguide-html.html> BootstrapCSS规范

3. <http://www.runoob.com/bootstrap/bootstrap-html-codeguide.html> BootstrapHTML规范

4. <http://ued.lanrentuku.com/wd/css-guifan-qianduan.html>

5. <https://www.w3cschool.cn/webdevelopment/ow89nozt.html>

6. <http://alloyteam.github.io/CodeGuide/> 腾讯alloy前端规范

7. <https://github.com/styleguide> github前端规范

8. <https://github.com/ecomfe/spec> 百度前端规范

# 常用组件命名

页面结构：

* 容器: container
* 页头：header
* 内容：content/container
* 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
* 菜单：menu
* 子菜单：submenu
* 标题: title
* 摘要: summary

功能：

* 标志：logo
* 广告：banner
* 登陆：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：guild
* 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