|  |
| --- |
| 蜂巢（Comb）开发规范 |
| |  |  |  | | --- | --- | --- | | 文件状态：  ☐ 草稿  ☐ 正在修改   已发布 | 文件标识 |  | | 当前版本 | V1.0 | | 作者 | 朱向飞 | | 完成日期 | 2017年5月24日 | | 审核人 |  | | 审核日期 |  |  |  |  |  | | --- | --- | --- | | 修改人 | 修改日期 | 修改内容 | |  |  |  | |  |  |  | |  |  |  | |
|  |

概述 2

目的 2

适用范围 2

蜂巢项目使用方式 2

*项目GitLib地址：* 2

*分支原则：* 2

*分支开发原则：* 2

基本原则 2

*文件及资源命名* 2

*注释* 3

*关注点分离* 3

*代码检查* 4

eslint部分 4

*命名规则* 4

*直接量* 5

*注释* 7

CSS部分 9

### 概述

#### 目的

提升蜂巢（Comb）代码质量，提高阅读性、复用性，加强团队合作能力，提升项目效率。

此文档不定期更新，会从更多的优秀开源项目中，汲取更优的编码方案，逐步提升个人及团队能力。

#### 适用范围

所有蜂巢（Comb）项目参与者需遵守此约定。

TIPS： 为能获得更好的阅读体验，阅读本文档时请关闭word的拼写和语法检查！

### 蜂巢项目使用方式

##### *项目GitLib地址：*

<http://172.28.20.6/Front/Comb>

使用工号登陆（RTX）

##### *分支原则：*

master分支为最终发布版本分支，只有拥有者（ower）有权限；

develop为开发分支，只有master有权限；

##### *分支开发原则：*

参与者需从develop上checkout出自己的分支进行修改或新组建的添加，完成后提交合并请求，经过code review后，合并至develop分支；

代码开发原则请遵从以下原则；

### 基本原则

##### *文件及资源命名*

资源文件命名全部使用小写字母加中横线（-）方式。在某些对大小写字母敏感的操作系统中，当文件通过工具压缩混淆后，或者人为修改过后，大小写不同而导致引用文件不同的错误，很难被发现。

js文件压缩后的min、或cdn混淆后的前缀等，使用点分割（.）

// 可以这样  
my-script.js  
my-camel-**case**-name.css  
i-love-underscores.html  
thousand-and-one-scripts.js  
my-file.min.css  
  
// 别这样  
MyScript.js  
myCamelCaseName.css  
i\_love\_underscores.html  
1001-scripts.js  
my-file-min.css

##### *注释*

请在编写js、scss文件时，在各文件开头部分按以下格式编写当前文件的说明信息。信息包含两个部分，一是对当前文件（或函数）进行功能（参数、返回）的描述，二是创建人（修改人）及时间信息；

*/\*\*  
 \* test.js  
 \* 上面是文件名称或模块名称, 这里进行模块的描述说明.  
 \* -- Created by Dio Zhu. on 2016.10.10  
 \*  
 \* 修改了某个功能, 添加了某些方法.  
 \* -- Modified by Dio Zhu. on 2016.10.11  
 \*/*

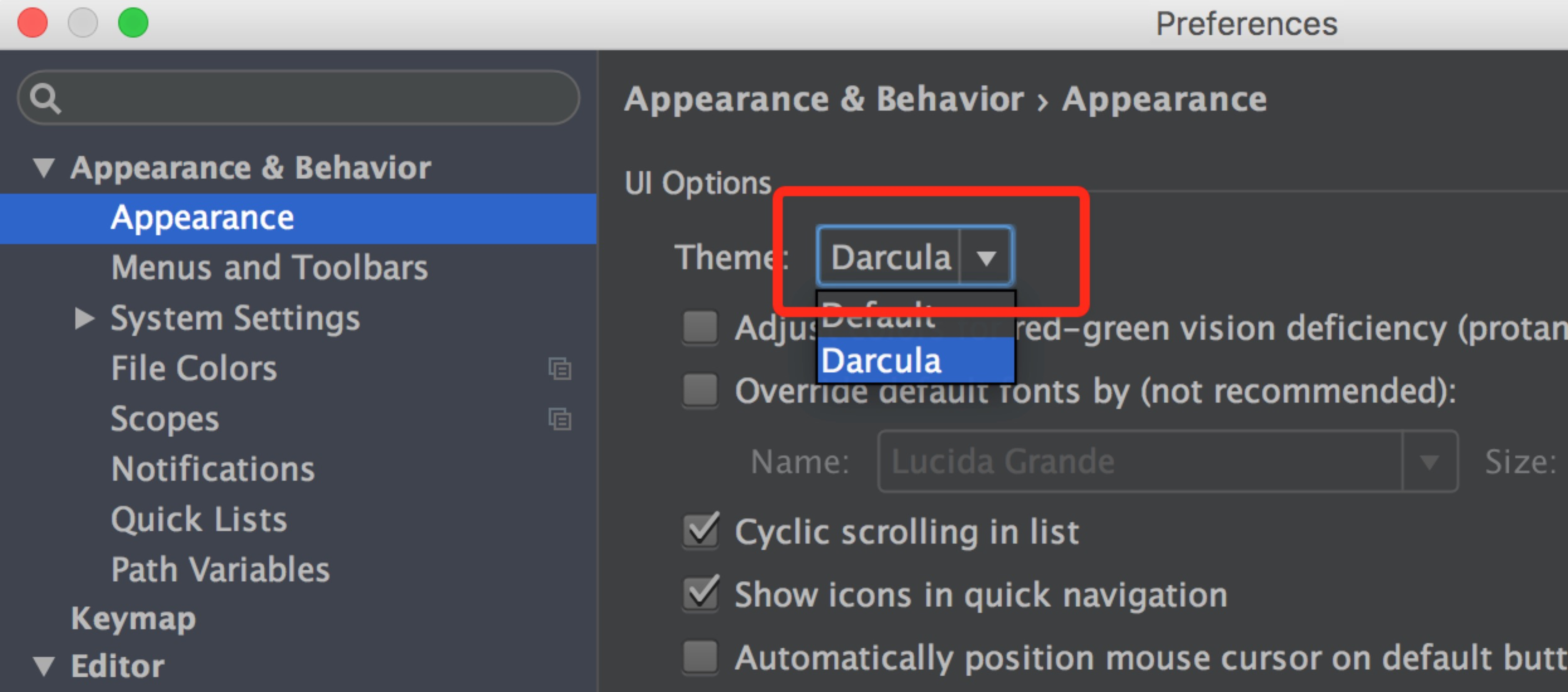
##### *关注点分离*

这里的关注点主要指的是：信息（HTML 结构）、外观（CSS）和行为（JavaScript），严格地保证三者分离。

* 不使用行内样式（<style>.no-good {}</style>）
* 尽量不在元素上使用 style 属性（<hr style="border-top: 5px solid black">）（动画的transform除外）
* 不使用行内脚本（<script>alert('no good')</script>）
* 不使用表象元素（<b>, <u>, <center>, <font>, <b>）
* 不使用表象 class 名（red, left, center）
* 所有非公共class必须限制在当前.page中，如：当前有<div class=“page page-demo”>，所有当前页的class一定放在.page-demo{}中，防止污染全局；

##### *代码检查*

建议调整webstorm的背景色为黑色，这样编码时的提示更容易引起注意。



### eslint部分

项目中使用了eslint进行约束，所有项目参与者需遵守其规则，不得私自修改相关配置。

特殊情况如：对外部js引用后，生成了自定义对象，可直接在上下文中使用实例，如果eslint报出“未定义”错误，推荐两种方式解决：

1. 修改引入的js文件，使其支持export方式；
2. 如是在线的js，比如埋点、地图等，可加入“/eslint-disable-line”进行例外处理；

##### *命名规则*

Camel Case 和 Pascal Case 均表示“驼峰式大小写”；

区别为Camel Case命名法是由小写字母开始，后续每个单词首字母大写；

Pascal Case以大写字母开头，后续每个单词首字母大写；

* 1. 变量：Camel Case，命名请不要使用动词
  2. 函数：Camel Case，命名请以动词开头

|  |  |
| --- | --- |
| **动词** | **含义** |
| can、has、is | 函数返回一个布尔值 |
| get | 函数返回一个非布尔值 |
| set | 函数保存一个值 |

// 可以这样: 名词表示的变量, 动词开头表示的函数  
**var** count = 10,  
 myName = 'Dio Zhu',  
 tag = **true**;  
  
**function** getMyName() {  
 **return** myName;  
}  
  
// 别这样: 看起来像函数的变量, 以及看起来像变量的函数  
**var** getCount = 10,  
 getMyName = 'Dio Zhu',  
 isTag = **true**;  
  
**function** theName() {  
 **return** getMyName;  
}

* 1. 常量：请以全部大写的形式

**var** MAX\_COUNT = 10,  
 URL = "http://www.syswin.com";

* 1. 构造函数：Pascal Case，名词形式

通过此约定可帮助我们快速定位，如果遇到以此形式命名的函数，前面一定要有new运算符。

**function** Person(name) {  
 **this**.name = name;  
}  
  
**var** user = **new** Person('Dio');

##### *直接量*

* 1. 字符串：使用双引号（单双引号并无不同，此约定是为保持整体风格）

**var** str = "This's my test project...";

* 1. 数字：不要省略整数部分或小数部分

// 可以这样:  
**var** count = 10,  
 price = 10.0;  
  
// 别这样:  
**var** shopPrice = 10.,  
 goodPrice = .9;

* 1. null：请区分与undefined区别，不要使用null检测未初始化的变量以及传入的参数

// 可以这样  
**var** person = **null**;  
  
// 可以这样  
**function** getPerson() {  
 **if** (tag) {  
 **return new** Person("Dio Zhu");  
 } **else** {  
 **return null** }  
}  
  
// 可以这样  
**var** user = getPerson();  
**if** (user !== **null**) {  
 doSomething();  
}  
  
// 不要这样  
**var** someone;  
**if** (someone !== **null**) {  
 doSomething();  
}  
  
// 不要这样  
**function** doIt(arg1, arg2, arg3) {  
 **if** (arg1 != **null**) {  
 doSomething();  
 }  
}

* 1. undefined：尽量避免使用undefined

如无特殊情况，不推荐项目中使用undefined校验。如需判断变量，请使用null方式占位。

* 1. 对象直接量：

// 可以这样  
**var** book = {  
 title: "A Book",  
 author: "Dio Zhu"  
};  
  
// 别这样  
**var** book2 = **new** Object();  
book2.title = "A Book";  
book2.author = "Dio Zhu";

* 1. 数组直接量：

// 可以这样  
**var** arr = [1, 2, 3];  
// 别这样  
**var** arrs = **new** Array(1, 2, 3);

##### *注释*

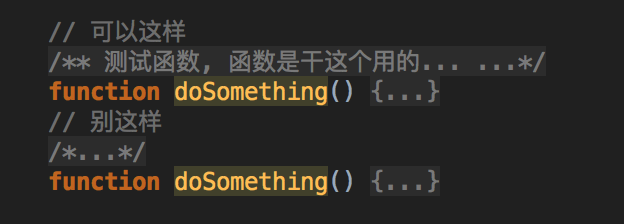
* 1. 模块注释：模块文件首行必须有说明注释， 目的是说明当前模块的描述和依赖关系等。

*/\*\*  
 \* test.js  
 \* 上面是文件名称或模块名称, 这里进行模块的描述说明.  
 \* -- Created by Dio Zhu. on 2016.10.10  
 \*  
 \* 修改了某个功能, 添加了某些方法.  
 \* -- Modified by Dio Zhu. on 2016.10.11  
 \*/*

* 1. 单行注释：请在复杂逻辑或关键逻辑节点进行标注，注意缩进与代码保持一致
  2. 多行注释：方法前使用“ /\*\* \*/“说明函数，便于折叠后能看到每个方法的简要说明。

// 可以这样  
*/\*\*  
 \* 测试函数, 函数是干这个用的...  
 \* -- Created by Dio Zhu. on 2016.10.10  
 \*/***function** doSomething() {  
 // 根据标识为判断路径...  
 **if** (arguments[0] === **true**) {  
 comeHere(); // 执行xxx  
 } **else** {  
 goThere(); // 否则xxx  
 }  
}  
// 别这样  
/\*  
 \* 测试函数, 函数是干这个用的...  
 \* -- Created by Dio Zhu. on 2016.10.10  
 \*/  
**function** doSomething() {  
// 错误的缩进  
 **if** (arguments[0] === **true**) {  
 comeHere();// 分号后面要有一个空格分隔, 这样影响阅读  
 } **else** {  
 goThere(); /\* 不要把多行注释用在行尾 \*/  
 }  
}

阅读代码时，我们可以采用“ctrl+shift+-“来进行全文语法折叠，这样我们能看到更多的函数，使用”/\*\* \*/“折叠后的代码能看到简述，这样便于我们更快的阅读代码。



### CSS部分

项目使用了autoprefixer进行兼容性语法转换，已经设置了兼容区间，原则上我们的代码不需要考虑兼容性写法，组件会帮我们进行处理。

但请注意，进项目的功能组件，必须经过android4.4版本、ios8.4版本及以上的机型验证过才可提交。

命名:请避免使用难以理解的class名称

/\* 可以这样 \*/  
.heavy {  
 font-weight: 800;  
}  
.important {  
 color: red;  
}  
  
/\* 别这样 \*/  
.fw-800 {  
 font-weight: 800;  
}  
.red {  
 color: red;  
}

不要使用ID，不仅是因为复用性，使用ID后会影响选择器的权重，使得代码变得很奇怪

/\* 这个选择器权重高于下面的选择器 \*/  
#content .title {  
 color: red;  
}  
html body div.content.news-content .title.content-title.important {  
 color: blue;  
}

请尽可能精确的使用选择器

/\* 可以这样 \*/  
.content {  
 > .title {  
 font-size: 2rem;  
 }  
}  
/\* 尽量别这样 \*/  
.content {  
 .title {  
 font-size: 2rem;  
 }  
}

尽量使用缩写语句，提升代码效率和可读性

div {  
 /\* 可以这样 \*/  
 border-top: 0;  
 font: 100%/1.6 palatino, georgia, serif;  
 padding: 0 1em 2em;  
  
 /\* 别这样 \*/  
 border-top-style: none;  
 font-family: palatino, georgia, serif;  
 font-size: 100%;  
 line-height: 1.6;  
 padding-bottom: 2em;  
 padding-left: 1em;  
 padding-right: 1em;  
 padding-top: 0;  
}

如果值为0，请不要加单位

div {  
 /\* 可以这样 \*/  
 padding-bottom: 0;  
 margin: 0;  
  
 /\* 别这样 \*/  
 padding-bottom: 0px;  
 margin: 0em;  
}

尽量使用3个字符的十六进制表示法

div {  
 /\* 可以这样 \*/  
 color: #f3a;  
 background-color: #fff;

/\* 别这样 \*/  
 color: #FF33AA;  
 background-color: #ffffff;  
}

命名分隔符只能使用中横线“-”

/\* 可以这样 \*/  
#video-id {}  
.ads-sample {}  
  
/\* 别这样 \*/  
.demoimage {}  
.error\_status {}

声明顺序：

这是一个选择器内书写CSS属性顺序的大致轮廓。这是为了保证更好的阅读体验。

作为最佳实践，我们应该遵循以下顺序：

1. 结构性属性：
   1. display
   2. position, left, top, right etc.
   3. overflow, float, clear etc.
   4. margin, padding
2. 表现性属性：
   1. background, border etc.
   2. font, text

/\* 可以这样 \*/  
.box {  
 display: block;  
 position: absolute;  
 left: 30%;  
 right: 30%;  
 overflow: hidden;  
 margin: 1em;  
 padding: 1em;  
 background-color: #eee;  
 border: 3px solid #ddd;  
 font-family: 'Arial', sans-serif;  
 font-size: 1.5rem;  
 text-transform: uppercase;  
}

/\* 别这样 \*/  
.box {  
 font-family: 'Arial', sans-serif;  
 border: 3px solid #ddd;  
 left: 30%;  
 position: absolute;  
 text-transform: uppercase;  
 background-color: #eee;  
 right: 30%;  
 display: block;  
 font-size: 1.5rem;  
 overflow: hidden;  
 padding: 1em;  
 margin: 1em;  
}

sass中请使用选择器嵌套，但请尽量避免空选择器的嵌套。并注意嵌套中空行的处理

/\* 别这样 \*/  
.content {  
 display: block;  
}  
  
.content > .news-article > .title {  
 font-size: 1.2em;  
}  
  
/\* 不是很好 **/  
.content {  
 display: block;  
  
 > .news-article {  
 /\*这里是个空选择器。。。**\*/  
 > .title {  
 font-size: 1.2em;  
 }  
 }  
}  
  
/\* 可以这样 \*/  
.content {  
 display: block;  
  
 > .news-article > .title {  
 font-size: 1.2em;  
 }  
}

不要以姓名缩写命名class

DOM结构不要超过8层，css结构尽量不要超过3层