# 开发规范

1. 目录结构的整齐划一统一化，可读性，分模块、组件构建，严禁构建杂乱无章，毫无可读性的项目目录。
2. 源文件目录、源文件名、类、方法、变量等要求使用英文字符命名，英文命名不够清晰的地方可以使用汉语拼音。

提示消息、表现层控件等给用户看的场景需使用中文。注释使用中文也可以让沟通更清晰易懂。

1. 所有纯文本文件都是用UTF-8编码，无 BOM。Windows要注意，使用系统自带的记事本无法做到这点，需要换个专业的文本编辑器，推荐Notepad++或者更专业的开发工具比如VS Code或者WebStorm。
2. 了解开发当前项目所需的框架，工具、插件，功能，兼容性，分辨率等问题做好准备工作，做到心中有数。
3. 代码风格以各语言官方文档为准
4. Java代码规范

<http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html>

1. Android代码规范

<http://source.android.com/source/code-style.html>

中文版本见《Code Conventions for Java》

1. iOS代码规范

《苹果 Cocoa 编码规范(中文版).pdf》

1. HTML和CSS和JavaScritp规范

Github上最火的：<https://github.com/airbnb/javascript>

中文版：https://github.com/sivan/javascript-style-guide

1. 变量命名要求遵循**见名知义**的原则，如果你的命名别人看不懂，或者过了2周自己也不知道这个变量是做什么的，那这个命名就失败鸟。

怎么做到见名知义呢？

1. 熟悉业务。只有熟悉了业务，才能准确定义业务和行为。
2. 业务区分。原来叫水果fruit，后来业务细分后，就得区分苹果apple和香蕉banana。
3. 根据语境加上适当的业务限定词。比如类名SpringBootApplication，他就跟Application、SpringApplication有区别。这就需要在语境里做适当的剪裁，以达到清晰不混淆的目的。
4. 英文要意译不要直译。”关注”动作译为follow，而不是focus或者attention。
5. 方法用动词开始，类名和属性名用名词。
6. 不要使用语音模糊、范围太大的词，例如：data, info等。
7. 尽量少地使用缩写，现代的屏幕和编辑器已经不在乎多几个字符，让别人理解意义是最重要的。地球人都知道的html这种缩写就可以使用。
8. 不要使用技巧性非常高的语法，或者很长的句子，可读性很差
9. 大量的命名，要使用前缀和后缀的方式，会使命名看起来很统一

//订单状态

typedef NS\_ENUM(NSInteger, GHSOrderStatus)

{

GHSOrderStatusPendingDelivery = 20,//待发货

GHSOrderStatusDelivering = 30, //已发货

GHSOrderStatusReceived = 39, //已签收

GHSOrderStatusFinished = 40, //交易成功

};

资源文件：

home\_logo.png

home\_item\_line.png

home\_icon\_search.png

home\_icon\_cancel.png

1. 养成使用常量和宏的好习惯，尽力避免使用magic number。如：

GHSPayTypeCashOnDelivery = 6, //货到付款

GHSPayTypeAlipay = 7 //支付宝

if(payType == 6) 这种代码别人是看不懂的

1. 跟开发、测试、生产环境相关的变量，禁止在项目中Hard Code，需要写到配置文件中，根据环境加载。比如：

服务器地址、小程序AppID、数据库用户名等

1. 业务逻辑尽量使用常量和宏，或者枚举，统一、见名知义、维护简单。

不要使用显示值来做业务逻辑，禁止下列行为：

~~if(status == “无货”)~~

~~{~~

~~disableBuyButton();~~

~~}~~

必须使用下列方式：

if( stock == 0 || status == STOCK\_EMPTY)

{

disableBuyButton();

}

1. 养成注释的好习惯，下列地方要求加注释
2. 业务逻辑复杂的地方
3. 变量需要释义和解释的地方
4. 暴露给别人调用的接口
5. 如果通过变量和命名已经很清晰业务逻辑，注释就是画蛇添足，例如：

if ( firstLaunch )

{

gotoRegister();

}

1. 尽量少地暴露，变量作用域越小越好。
2. 能用本模块完成的，就不用全局变量。
3. 只在本模块用的变量，不要定义在头文件里
4. 只在1个方法用到的变量，不要写成类变量
5. 尽量少使用全局变量
6. 代码封装，实现统一、容易维护、容易理解
7. 方法实现写50行以上，考虑一下是否有必要封装成另外一个方法
8. 一段代码被拷贝了3次，那就应该考虑封装成1个方法
9. 一个代码缩进了5层以上，那最好单独写个方法调用
10. 尽量使用所见即所得编辑器制作界面，下列情形才考虑使用代码实现界面
11. 编辑器无法实现的界面
12. 复用性很强的界面，也可以用部分代码部分
13. 界面制作使用恰当的占位图片和占位字符，尽量模仿真实数据。
14. 界面实现必须完全按照效果图来做，注意控件的坐标、大小尺寸，对齐、字体、颜色、阴影等
15. 界面上按钮可点击区域要大于44 x 44，在界面布局不允许这么大的情况下，尽量扩大可点击区域
16. 文本显示区域要尽可能地大，以避免文本显示不全，需要计算尺寸的地方，要进行计算然后调整相应的位置
17. 拿到界面要第一时间考虑适配问题，哪些空间是拉伸的，哪些空间是写死的，需要做到心中有数
18. 显示数据需要注意的地方
19. 视图加载完成后，清空占位数据，加载上个页面传过来的数据
20. 如果页面有缓存，需要加载缓存数据
21. 如果数据需要刷新，在视图将要显示时，根据刷新条件进行刷新
22. 显示数据建议封装成统一的方法比如showUser(User user)
23. 请求和提交数据时
24. 需要使用Loading框或者把按钮置为不可用，以免重复提交
25. 需要对数据进行非空和格式校验
26. 需要按照返回的错误代码进行恰当的错误提示
27. 类库和解决方案选择的原则
28. 使用语言、框架官方提供的方案
29. 使用自己公司封装好的类库和工具，统一成熟、有人测试有人维护
30. 使用github和stackoverflow等网站评分较高的开源方案
31. 使用Google搜索出来靠前方案
32. 使用活跃的开源社区提供的方案
33. 使用比较新的方案
34. 实在找不到了才考虑自己实现
35. 使用项目统一提供的基类、样式库、基础组件，这样可以在一个项目中统一样式，统一维护。