# 项目编码规范

### 前端开发规范

代码一致性和最佳实践性。通过代码风格的一致性，降低维护代码的成本以及改善多人协作的效率。同时遵守最佳实践，确保页面性能得到最佳优化和高效规范。

编码规范就是编写出简洁、可维护、可靠、可测试、高效、可移植的代码，提高产品代码的质量。

1、函数

函数设计的要点：编写整洁的函数，同时把代码有效地组织起来。

函数整洁的要求：代码简单直接、不隐藏设计者的意图，用干净利落的抽象和直截了当的控制语句将函数有机组织起来。

原则：

a、一个函数仅能完成一个功能；

b、重复代码应该尽可能提炼成函数；

原则：

a、避免函数过长；

b、避免函数的代码块嵌套过深，嵌套不超过4层；

c、可重入函数应该少用或者尽量避免使用共享变量；

d、检查参数合法性，避免程序瘫痪；

e、对函数的错误返回值要全面处理；

f、对整个小程序无用的代码要及时删除；

建议：

a、在app文件，即源文件中声明和定义所有函数；

b、函数应尽量避免使用全局变量、静态局部变量和I/O操作，不可避免的地方应集中使用；

c、检查函数所有非参数输入的有效性，如数据文件、公共变量等；

d、函数参数不宜过长，控制在5个以下；

2、结构、样式、行为分离

结构和样式都放在样式文件中，即CSS文件；行为都放在脚本文件中JS和WXML文件中。

3、识符命名与定义

特别注意标识符的命名，统一、明确的标识符命名可以增强程序的可读性，也有利于程序的维护。

1）、 通用命名规则

常用命名规范：

a、划线“\_”分割，如text\_userinfo;

b、大小写字母混用，驼峰式命名法，如ReadRFCText。

规则：

a、标识符的命名要清晰、明了，有明确含义，同时使用完整的单词或大家基本可以理解的缩写，不得使用汉语拼音，避免使人产生误解，可读性不强，影响程序的整体效果；

b、项目组内部应保持统一的命名风格即使用英文单词统一命名，不得出现模棱两可的标识符；

建议：

a、用反义单词词表示具有互斥意义的变量或相反动作的函数;

b、尽量避免名字中出现数字编号;

c、标识符前不应添加模块、项目、产品、部门的名称作为前缀;

d、重构或者修改部分代码时，应保持和原有代码的命名风格一致。

2）、 文件命名规则

文件命名统一采用小写字符。

3）、变量命名规则

全局变量十分危险，通过前缀使得全局变量更加醒目， 促使开发人员对这些变量的使用更加小心。

规则：

a、全局变量增加“g”前缀，静态变量增加“s”前缀;

b、禁止使用单字节命名变量，但允许定义i、 j、 k作为局部循环变量;

c、使用名词或者形容词＋名词方式命名变量。

4）、 函数命名规则

a、函数命名应以函数要执行的动作命名，一般采用动词或者动词＋名词的结构;

b、函数指针除了前缀，其他按照函数的命名规则命名。

5）、宏的命名规则

a、对于数值或者字符串等等常量的定义，建议采用全大写字母，单词之间加下划线"\_"的方式命名（枚举同样建议使用此方式定义）;

b、除了头文件或编译开关等特殊标识定义，宏定义不能使用下划线"\_"开头和结尾。

4、变量

原则：

a、一个变量只有一个功能;

b、结构功能单一；

c、不用或者少用全局变量。

规则：

a、防止局部变量与全局变量同名;

b、严禁使用未经初始化的变量作为右值;

建议：

a、构造仅有一个模块或函数可以修改、创建，而其余有关模块或函数只访问的全局变量，防止多个不同模块或函数都可以修改、创建同一全局变量的现象;

b、使用面向接口编程思想，通过API访问数据：如果本模块的数据需要对外部模块开放， 应提供接口函数来设置、获取，同时注意全局数据的访问互斥;

c、在首次使用前初始化变量，初始化的地方离使用的地方越近越好;

d、明确全局变量的初始化顺序，避免跨模块的初始化依赖;

e、尽量减少没有必要的数据类型默认转换与强制转换。

5、常量

规则：

a、定义表达式时，要使用完备的括号;

eg：#define RECTANGLE\_AREA(a, b) ((a) \* (b))

b、将宏所定义的多条表达式放在大括号中;

c、使用宏时，不允许参数发生变化;

#define SQUARE(a) ((a) \* (a))

int a = 5;

int b;

b = SQUARE(a);

a++;

建议：

a、除非必要，应尽可能使用函数代替宏;

b、常量建议使用const定义代替宏;

c、宏定义中尽量不使用return、 goto、 continue、 break等改变程序流程的语句。

6、注释

原则：

a、注释的内容要清楚、明了，含义准确，防止注释二义性;

b、注释解释代码难以直接表达的意图，而不是重复描述代码；

规则：

a、修改代码时，维护代码周边的所有注释，以保证注释与代码的一致性。不再有用的注释要删;

b、文件头部应进行注释，注释必须列出：版权说明、版本号等；

c、函数声明处注释描述函数功能、性能及用法，包括输入和输出参数、函数返回值、可重入的要求等；定义处详细描述函数功能和实现要点，如实现的简要步骤、实现的理由、 设计约束等;

d、全局变量要有较详细的注释，包括对其功能、取值范围以及存取时注意事项等的说明;

e、注释应放在其代码上方相邻位置或右方，不可放在下面；

f、避免在注释中使用缩写；

g、同一产品或项目组统一注释风格。

建议：

a、避免在一行代码或表达式的中间插入注释;

b、文件头、函数头、全局常量变量、类型定义的注释格式采用工具可识别的格式。

7、排版与格式

规则：

a、程序块采用缩进风格编写， 每级缩进为2个空格;

b、相对独立的程序块之间、变量说明之后必须加空行;

c、一条语句不能过长，如不能拆分需要分行写；

d、一行只写一条语句;

e、在两个以上的关键字、变量、常量进行对等操作时，它们之间的操作符之前、之后或者前后要加空格； 进行非对等操作时，如果是关系密切的立即操作符（如－>），后不应加空格;

### 后端开发规范

1、数据库命名

a、 数据库表、视图、存储过程命名按照如下格式：

表：T\_Student

视图：V\_Student

存储过程：P\_Student

主键：PK\_StudentId

外键：FK\_StudentSorce

索引：IX\_StudentNum

规则为开头简写大写字母加下划线加全称英文单词，英文单词首字母大写。

b、数据库语句中SQL关键字，系统默认参数、变量全部大写。

c、 多表交叉查询语句不使用表别名，使用完整的限定名。

d、 查询语句中的字段大小写于数据表设计时使用的大小写一样。

e、多表查询统一使用join关联查询语句；

2、代码缩进

a、使用Tab键来缩进，每个Tab键长度设置为2个空格；

b、程序当中不允许出现空格缩进；

3、关键字 和 True/False/Null

PHP的关键字，必须小写，boolean值：true，false，null 也必须小写。

4、代码注释标签

函数注释、变量注释等，常用标签有 @package、@var

5、业务模块

a、涉及到多个数据表 更新/添加 操作时，最外层建议要用事务，保证数据库操作的原子性；

b、只做简单的数据表的增删改查；

c、控制器层不能出现SQL操作语句；

6、码样式风格

a、命名空间(namespace)的声明后面必须有一行空行；

b、所有导入(use)声明必须放在命名空间(namespace)声明的下面；

c、一句声明中，必须只有一个导入(use)关键字；