Coding Style Guidelines

By Group. Khaos

一．Coding Style

1. Indentation

因为 Tab 是 8 字符长度，所以每次缩进保持 8 字符长度。

2.Placing Braces

a).一般情况下。前括号放置于逻辑句的句尾，后括号空行行首：

```
if (x is true) {
        we do y
}
```

b).对于函数：则将前括号放在下行行首：

```
int function(int x)
{
        body of function
}
```

c). 对于连续逻辑语句，后括号后面紧跟后续语句：

```
do {
        body of do-loop
} while (condition);
```

和

```
if (x == y) {
        ..
} else if (x > y) {
        ..
} else {
        ....
}
```

3. Naming

对于全局变量，使用描述性命名：如 global_Variable 或 globalVariable；

局部变量，则使用简写：　　如 tmp, tmpvar。

4. Functions

函数的 screen size 保持 80x24 左右，超过之后则考虑是否能分为多个小函数；

函数内部局部变量变量保持不超过 5-10 个。

5. Commenting

在函数头部说明函数功能，pre-condition 和 post-condition 等，另外，可适当增加些原理的解释（但不宜太多）。

6. Data structures

尽量使用面向对象编程方式，增加代码可重用性，同时保证了代码的规范性，通过调用已有函数，增加界面的一致性。

二．Style consistency sponsorship

1.结对编程

对于一些重要，常用模块，进行结对编程，即一人主要编写，另一人旁观，提出问题和意见方案，同时可起到监督的作用，以保证代码的规范化和准确性！同时可以提高效率。

结对编程技术是一个非常简单和直观的概念:两位程序员肩并肩地坐在同一台电脑前合作完成同一个设计。同一个算法、同一段代码或同一组测试、与两位程序员各自独立工作相比．结对编程往往只需花费大约一半的时间就能编写出质量更高的代码， 但是，人与人之间的合作不是一件简单的事情——尤其当人们都早已习惯了独自工作的时候、实施结对编程技术将给软件项目的开发工作带来好处.只是这些好处必须经过缜密的思考和计划才能真正体现出来。而另一方面，两个有经验的人可能会发现配对编程里没有什么技能的转移，但是让他们在不同的抽象层次解决同一个问题会让他们更快地找到解决方案，而且错误更少。

结对编程还有其他多种好处:

a)、*直接的、连续的代码回顾。*

b)、*与别人工作会增加责任和纪律性。*

c)、*同时理解一个问题。*

d)、*在有人盯着的时候去偷懒要困难得多！*

2.Daily Check

由于 web 编程的特殊性，如 PHP 效果的即时可见性，可尝试建立公共 FTP，与 GIT 同步数据，

并以此为基础建立 HTTP 服务器，上传新页面后，成员都可以进行访问，并可即时查看对应源代码，查看代码是否符合规范。

周期：每天一次。

Daily Check．就是以天为单位，定期对代码，界面进行检测验收，有意见就进行提出，以期在较快的时间内对不合格的进行及时纠正，这样就可以防止不合格代码的泛滥，保证开发的高效，正确进行!

3.代码模块化重用

使用面向对象的方式，将特定功能模块化，规格化后，页面显示，数据访问等均采

用框架函数的方式，减少额外代码的使用。

模块化设计，简单地说就是程序的编写不是开始就逐条录入计算机语句和指令，而是首先用主程序、子程序、子过程等框架把软件的主要结构和流程描述出来，并定义和调试好各个框架之间的输入、输出链接关系。逐步求精的结果是得到一系列以功能块为单位的算法描述。以功能块为单位进行程序设计，实现其求解算法的方法称为模块化。模块化的目的是为了降低程序复杂度，使程序设计、调试和维护等操作简单化。

4.采用 MVC 架构，分离模块

根据 MVC 设计开发流程，分离 Model，View，Controller，协商之后，再分别进行设计建模开发，自顶向下逐步细化模块，同时保证代码的标准性。

MVC 是三个单词的缩写,分别为： 模型(Model),视图(View)和控制 Controller)。 MVC 模式的目的就是实现 Web 系统的职能分工。 Model 层实现系统中的业务逻辑，通常可以用 JavaBean 或 EJB 来实现。 View 层用于与用户的交互，通常用 JSP 来实现。 Controller 层是 Model 与 View 之间沟通的桥梁，它可以分派用户的请求并选择恰当的视图以用于显示，同时它也可以解释用户的输入并将它们映 射为模型层可执行的操作。