**一、规范目的**

为提高团队协作效率，便于前端后期优化维护，输出高质量的文档。

**二、基本准则**

* 符合web标准，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。
* 项目的维护和二次开发可能是直接或间接的团队合作，所以创建易维护的代码是一个项目成功与否的关键，易维护的代码意味着具有如下特性：
* 阅读性好：如良好的注释和命名规范，有文档
* 具有一致性：看起来如同一个人编写
* 代码的松耦合，高度模块化：将页面内的元素视为一个个模块，相互独立，尽量避免耦合过高的代码，从html,css,js三个层面都要考虑模块化
* 严格按照规范编写代码

**三、命名规范**

1. 目的

提高代码可预测性和可维护性的方法是使用命名约定，这就意味着采用一致的方法来对变量和函数进行命名。

2. 变量名

变量名包括全局变量，局部变量，类变量，函数参数

3. 构造函数（类）命名

首字母大写，驼峰式命名。

JS中没有类，但是可以用new调用构造函数：var man = new Person();

4. 普通变量命名

首字母小写，驼峰式命名，匈牙利命名

如：nCheckCount 表示整形的数值

5. 匈牙利命名法

匈牙利命名法语法：变量名＝类型＋对象描述

类型指变量的类型

对象描述指对象名字全称或名字的一部分，要求有明确含义，命名要容易记忆容易理解。

提示: 虽然JavaScript变量表面上没有类型，但是JavaScript内部还是会为变量赋予相应的类型

| **JavaScript变量起名类型** | **变量命名前缀** | **举例** |
| --- | --- | --- |
| Array 数组 | a | aList，aGroup |
| Boolean 逻辑 | b | bChecked，bHasLogin |
| Function 函数 | f | fGetHtml，fInit |
| Integer 数字 | n | nPage，nTotal |
| Object 对象 | o | oButton，oDate |
| Regular Expression 正则 | r | rDomain，rEmail |
| String 字符 | s | sName，sHtml |

6. 例外情况

以根据项目及团队需要，设计出针对项目需要的前缀规范，从而达到团队开发协作便利的目的。

作用域不大临时变量可以简写，比如：str，num，bol，obj，fun，arr。

循环变量可以简写，比如：i，j，k等。

某些作为不允许修改值的变量认为是常量，全部字母都大写。例如：COPYRIGHT，PI。常量可以存在于函数中，也可以存在于全局。必须采用全大写的命名，且单词以\_分割，常量通常用于ajax请求url，和一些不会改变的数据。

7. 函数命名

**普通函数**：首字母小写，驼峰式命名，统一使用动词或者动词+名词形式

例如：fnGetVersion()，fnSubmitForm()，fnInit()；涉及返回逻辑值的函数可以使用is，has，contains等表示逻辑的词语代替动词，例如：fnIsObject()，fnHasClass()，fnContainsElment()。

**内部函数**：使用\_fn+动词+名词形式，内部函数必需在函数最后定义。

例如：

function fnGetNumber(nTotal) {

if (nTotal < 100) {

nTotal = 100;

}

return \_fnAdd(nTotal);

function \_fnAdd(nNumber) {

nNumber++;

return nNumber;

}

}

alert(fGetNumber(10)); //alert 101

**对象方法与事件响应函数**：对象方法命名使用fn+对象类名+动词+名词形式；

例如： fnAddressGetEmail()，

**事件响应函数**：fn+触发事件对象名+事件名或者模块名

例如：fnDivClick()，fnAddressSubmitButtonClick()

**函数方法常用的动词：**

get 获取/set 设置,

add 增加/remove 删除

create 创建/destory 移除

start 启动/stop 停止

open 打开/close 关闭,

read 读取/write 写入

load 载入/save 保存,

create 创建/destroy 销毁

begin 开始/end 结束,

backup 备份/restore 恢复

import 导入/export 导出,

split 分割/merge 合并

inject 注入/extract 提取,

attach 附着/detach 脱离

bind 绑定/separate 分离,

view 查看/browse 浏览

edit 编辑/modify 修改,

select 选取/mark 标记

copy 复制/paste 粘贴,

undo 撤销/redo 重做

insert 插入/delete 移除,

add 加入/append 添加

clean 清理/clear 清除,

index 索引/sort 排序

find 查找/search 搜索,

increase 增加/decrease 减少

play 播放/pause 暂停,

launch 启动/run 运行

compile 编译/execute 执行,

debug 调试/trace 跟踪

observe 观察/listen 监听,

build 构建/publish 发布

input 输入/output 输出,

encode 编码/decode 解码

encrypt 加密/decrypt 解密,

compress 压缩/decompress 解压缩

pack 打包/unpack 解包,

parse 解析/emit 生成

connect 连接/disconnect 断开,

send 发送/receive 接收

download 下载/upload 上传,

refresh 刷新/synchronize 同步

update 更新/revert 复原,

lock 锁定/unlock 解锁

check out 签出/check in 签入,

submit 提交/commit 交付

push 推/pull 拉,

expand 展开/collapse 折叠

begin 起始/end 结束,

start 开始/finish 完成

enter 进入/exit 退出,

abort 放弃/quit 离开

obsolete 废弃/depreciate 废旧,

collect 收集/aggregate 聚集

1. 变量命名例子

* 为什么需要这样强制定义变量前缀？正式因为javascript是弱语言造成的。在定义大量变量的时候，我们需要很明确的知道当前变量是什么属性，如果只通过普通单词，是很难区分的。

**普通代码**

var checked = false;

var check = function() {

return true;

}

if(check) {//已经无法很确切知道这里是要用checked还是check()从而导致逻辑错误

//do some thing

}

**规范后代码**

var bChecked = false;

var fnCheck = function() {

return true;

}

if(bChecked) {

// do some thing

}

if(fnCheck()) {

// do other thing

}

1. **编写注释**

为代码编写注释是非常重要的。通常人们在深入思考一个问题时，会非常清楚这段代码的工作原理。但是当过一周后再次回到该代码时，可能会花上很长时间来回想起那段代码到底是干什么的。

* 公共组件维护者和各栏目WD都需要在文件头部加上注释说明：

/\*\*

\*文件用途说明

\*作者姓名、联系方式

\*制作日期

\*\*/

* 大的模块注释方法：

//================

// 代码用途

//================

* 小的注释：

//代码说明

* 注释单独一行，不要在代码后的同一行内加注释。例如：

//姓名

var name = “abc”; V

var name =”abc”; //姓名 X

**五、引号的使用，单引号' ' 优先（如果不是引号嵌套，不要使用双引号）**