# 苏宁易购javascript编码规范

版本：1.0

修订日期：2013-04-01

# 1.JavaScript 语言规范

- 声明变量必须加上 `var` 关键字.
    - 当你没有写 `var`, 变量就会暴露在全局上下文中, 这样很可能会和现有变量冲突. 另外, 如果没有加上 `var` , 很难明确该变量的作用域是什么, 变量也很可能像在局部作用域中, 很轻易地泄漏到 `Document` 或者 `Window` 中, 所以务必用 `var` 去声明变量.
- 常量必须使用大写字符,如 `NAMES_LIKE_THIS` , 并用下划线分隔.

```
var PI = 3.14
```

- JavaScript代码不应该被包含在HTML文件中,除非这是段特定只属于此部分的代码。

```
//特例：  宽窄屏切换必须放在HTML文件内
<script type="text/javascript">
    /*
    * 宽窄屏切换
    * */
    var bigscreen = false;
    if ( screen.width>=1200 ) {
    bigscreen = true;
    var bodyTag = document.getElementsByTagName("body")[0],
    bodyClassName = bodyTag.getAttribute("className") || bodyTag.getAttribute("class");
    bodyClassName = bodyClassName ? bodyClassName+" " : "";
    bodyTag.className = bodyClassName+"root1200";
    }
</script>
```

- 所有的变量必须在使用前进行声明

```
//错误的!
if(sn){
    return data;
}
var sn = true;
```

- 所有的函数在使用前进行声明

- 即使只有一条语句也不能省略{}

```
if(boolean){
    return ;
}
```

- 每个语句结束必须使用分号
- 使用{}代替new Object()。使用[]代替new Array()

```
var arr = [], obj = {};
```

- eval是JavaScript中最容易被滥用的方法。避免使用它。（只用于解析序列化串）

```
$.get(url, function(sn_data_json){
    eval(sn_data_json)
})
```

- 类(构造函数)的必须首字母大写

```
function Class(){


}
```

- 不要使用with
- this仅在对象构造器,方法,闭包中使用
- 不要修改内置对象的原型
- 不要使用 `for-in` 遍历数组
- 使用规范有意义的变量名
    - 使其他成员可以更好的读懂你写的代码
- 尽量降低代码与XHTML的耦合性
    - 使代码更具有通用性
- 一个函数应该返回统一的数据类型

```
//错误示例，应该返回同样的数据类型
function getUserName(userID){
     if (data[userID]){
```

```
        return data[userID];
    } else {
        return false;
    }
}
```

- 不重复定义其他团队成员已经实现的方法

---

# 2.JavaScript 编码风格

- 私有变量名用下划线开头
- 变量用下划线分隔方式或者驼峰方式

```
var className = "suning";
var class_name = "suning";
```

- 为全局代码使用命名空间

```
var SN = {}
SN.tab = function(){};
SN.show = function(){};
```

- 复合语句是被包含在{ }(大括号)的语句序列，{(左大括号)应在复合语句其实行的结尾处，}(右大括号)应与{(左大括号)的那一行的开头对齐
- 明确命名空间所有权
    - 当选择了一个子命名空间, 请确保父命名空间的负责人知道你在用哪个子命名空间, 比如说, 你为工程 'SNProduct' 创建一个 'lottery' 子命名空间, 那确保 "SNProduct" 团队人员知道你在使用 SNProduct.lottery
- 尽量不要让每行超过 80 个字符;
- 总是使用分号.
    - 保证代码压缩后不出错
    - 能清楚哪里是语句的起止
- 语句中的必要空格和缩进，缩进的单位为四个空格

```
function somefunction(){
```

```
    if(true){
        return 1;
    } else {
        return 0;
    }
}
```

- 重命名那些名字很长的变量,提高可读性

```
var SNProduct = some.long.className.MyclassName;
```

- "="前后需要跟空格,建议数值操作符(如,+/-/*/% 等)两边留空;

```
var sn = "suning";
```

- for 循环条件中,分号后留一空格;

```
for( var i = 0; i < 10; i++){
    //TO DO...
}
```

- 变量声明语句,数组值,对象值及函数参数值中的逗号后留一空格;

```
var a = 0, b = 1, c = 2;
var arr = [1, 2, 3, 4];
var obj = {a: 1, b: 2, c: 3};
function(a, b, c){}
```

- 空行不要有空格;行尾不要有空格;
- 逗号和冒号后一定要跟空格;点号前后不要出现空格;
- 空对象和数组不需要填入空格;
    - 不要吝啬空行.尽量使用空行将逻辑相关的代码块分割开,以提高程序的可读性.
- 格式化对象参数

```
var obj = {
    getName: "1",
    setName: "0"
}
```

- 代码需要格式化
  - 大括号

```
if (suning) {
  // ...
} else {
  // ...
}
```

  - 数组和对象的初始化
    - 如果初始值不是很长,就保持写在单行上:

```
var arr = [1, 2, 3];
var obj = {a: 1, b: 2, c: 3};
```

    - 初始值占用多行时,缩进4个空格.

```
// 对象初始化
var inset = {
    top: 10,
    right: 20,
    bottom: 15,
    left: 12
};

// 对象参数初始化
SNCore.UI("Tab", {
    tabBtn: "#id",
    tabBtnClass: "cur",
    tabBox: "#box",
    tabType: "click",
    tabBoxClass: ".className"
});
```

    - 比较长的标识符或者数值,不要为了让代码好看些而手工对齐.如:

```
//不推荐
var aaaa     = 1,
    b        = 2,
```

```
    ccccccc = 3;
```

- 逻辑上独立的代码块使用空行分隔

```
doSomethingTo(x);
doSomethingElseTo(x);
andThen(x);

nowDoSomethingWith(y);

andNowWith(z);
```

- 二元和三元操作符

```
// 在同一行
var x = a ? b : c;

// 一个缩进
var y = a ?
    longButSimpleOperandB : longButSimpleOperandC;

// 两个缩进.
var z = a ?
        moreComplicatedB :
        moreComplicatedC;
```

- 括号
  - 不要滥用括号, 只在必要的时候使用它.对于一元操作符(如delete, typeof 和 void ), 或是在某些关键词(如 return, throw, case, new )之后, 不要使用括号.
- 字符串
  - 单引号 (') 优于双引号 ("). 当你创建一个包含 HTML 代码的字符串时就知道它的好处了.

```
var string = '<div class="demo"></div>';
```

- 建议每个重要函数定义之前尽量添加注释以说明此函数的主要信息，包括：函数的用途，创建者，创建时间以及函数的实现原理，如果后人修改这个函数，也要在注释中添加修改信息：包括修改内容，修改人以及修改时间等。这些信息有助于函数的使用以及函数代码的维护。

- 注释
  - 参照jsDoc
  - 顶层/文件注释

```
// Copyright 2013 Suning.com All Rights Reserved.
/**
 * @fileoverview 易购通用JS方法库
 * about based on jQuery
 * @author 1205****
 * @version 1.0.0
 */
```

  - 类注释

```
/**
 * 类的作用
 * @param  {Array}  a  原数组
 * @param  {Object} b  操作的对象
 * @param  {Number} c  数组索引
 * @return {Array}  新数组
 */

function Class(a, b, c){

}
```

  - 不要在一行注释

```
//不要像下面这样注释
var a = 0; //初始数值
```

  - 对于一些简单的,不带参数的 getters,说明可以忽略.

```
/**
 * @return {Element} The element for the function.
 */
suning.someFunction.getArr = function() {
    return this.element;
};
```

- 保持一致性.
    - 当你在编辑代码之前, 建议先花一些时间查看一下现有代码的风格. 如果他们给算术运算符添加了空格, 你也应该添加. 如果他们的注释使用一个个星号盒子, 那么也请你使用这种方式.

---

# 建议

1. 永远不要忽略代码优化工作

2. 不要吝啬注释。及时更新注释

---