JS编写规范

(讨论稿)

版 本 号：V1.0

发布时间：2008年01月

# 历史修订记录

修订记录：

| 作者 | 版本 | 修订日期 | 内容描述 | 章节 | 页面 |
| --- | --- | --- | --- | --- | --- |
|  | V1.0 |  | 初始版本 | 目录大纲 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

复核记录：

| 姓名 | 职务/修订摘要 | 发表日期 | 版本 |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

审批/签收记录：

| 姓名 | 角色 | 日期 | 签字 |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

# JS编写规范纲领

## 1.1规范制订的必要性

代码质量在软件质量中占很大比例。在软件生命周期里，一个程序会被许多人接手。如果一个程序可以很好的表达自己的结构和特性，则在不久的将来修改它时就会减少程序崩溃的可能。代码规范可以帮助减少程序的脆弱性。我们所有的JavaScript代码都直接发布给公众，它应该具有发布质量

## 1.2参考资料

# 2、JS编写规范细则

## 2.1 总体命名规范

### 2.1.1命名基本规范

1）.变量名称 必须为 小写字母。

2）.类的命名使用骆驼命名规则，例如： Account, EventHandler

3）.常量 必须 在对象（类）或者枚举变量的前部声明。枚举变量的命名必须要有实际的意义，并且其成员 必须 使用骆驼命名规则或使用大写：

var NodeTypes = {

Element : 1,

DOCUMENT: 2

}

4）.简写单词 不能使用 大写名称作为变量名：

getInnerHtml(), getXml(), XmlDocument

5）.方法的命令 必须 为动词或者是动词短语： obj.getSomeValue()

6）.公有类的命名 必须 使用混合名称（mixedCase）命名。

7）.CSS 变量的命名 必须 使用其对应的相同的公共类变量。

8）.私有类的变量属性成员 必须 使用混合名称（mixedCase）命名，并前面下下划线（\_）。例如：

var MyClass = function(){

var \_buffer;

this.doSomething = function(){

};

}

9）.变量如果设置为私有，则前面 必须 添加下划线。

this.\_somePrivateVariable = statement;

10）.通用的变量 必须 使用与其名字一致的类型名称：

setTopic(topic) // 变量 topic 为 Topic 类型的变量

11）.所有的变量名 必须 使用英文名称。

12）.变量如有较广的作用域（large scope），必须使用全局变量；此时可以设计成一个类的成员。相对的如作用域较小或为私有变量则使用简洁的单词命名。

13）.如果变量有其隐含的返回值，则避免使用其相似的方法：

getHandler(); // 避免使用 getEventHandler()

14）.公有变量必须清楚的表达其自身的属性，避免字义含糊不清，例如：

MouseEventHandler，而非 MseEvtHdlr。

请再次注意这条规定，这样做得的好处是非常明显的。它能明确的表达表达式所定义的含义。例如：

dojo.events.mouse.Handler // 而非 dojo.events.mouse.MouseEventHandler

15）.类/构造函数 可以使用 扩展其基类的名称命名，这样可以正确、迅速的找到其基类的名称：

EventHandler

UIEventHandler

MouseEventHandler

基类可以在明确描述其属性的前提下，缩减其命名：

MouseEventHandler as opposed to MouseUIEventHandler.

### 2.1.2特殊命名规范

1）.术语 "get/set" 不要和一个字段相连，除非它被定义为私有变量。

2）.前面加 "is" 的变量名 应该 为布尔值，同理可以为 "has", "can" 或者 "should"。

3）.术语 "compute" 作为变量名应为已经计算完成的变量。

4）.术语 "find" 作为变量名应为已经查找完成的变量。

5）.术语 "initialize" 或者 "init" 作为变量名应为已经实例化（初始化）完成的类或者其他类型的变量。

6）.UI （用户界面）控制变量应在名称后加控制类型，例如：

leftComboBox, TopScrollPane。

7）.复数必须有其公共的名称约定（原文：Plural form MUST be used to name collections）。

8）.带有 "num" 或者 "count" 开头的变量名约定为数字（对象）。

9）.重复变量建议使用 "i", "j", "k" （依次类推）等名称的变量。

10）.补充用语必须使用补充词，例如：

get/set, add/remove, create/destroy, start/stop, insert/delete, begin/end, etc.

11）.能缩写的名称尽量使用缩写。

12）.避免产生歧义的布尔变量名称，例如：

isNotError, isNotFound 为非法

13）.错误类建议在变量名称后加上 "Exception" 或者 "Error"。

14）.方法如果返回一个类，则应该在名称上说明返回什么；如果是一个过程，则应该说明做了什么。

## 2.2详细命名规范

### 2.2.1布局

1）.缩进请使用 4 个空白符的制表位。

2）.表达式的缩进与变量声明应为一致的。

3）.代码折叠必须看起来是完成并且是合乎逻辑的：

4）.函数的参数应采用明确的缩进，缩进规则与其他块保持一致。

### 2.2.2变量

1.变量必须在声明初始化以后才能使用，即便是 NULL 类型。

2.变量不能产生歧义。

3.相关的变量集应该放在同一代码块中，非相关的变量集不应该放在同一代码块中。

4.变量应该尽量保持最小的生存周期。

5.循环/重复变量的规范：

1).只有循环控制块的话，则必须使用 FOR 循环。

2).循环变量应该在循环开始前就被初始化；如使用 FOR 循环，则使用 FOR 语句初始化循环变量。

3)."do ... while" 语句是被允许的。

4)."break" 和 "continue" 语句仍然允许使用（但请注意）。

6.条件表达式

1).应该尽量避免复杂的条件表达式，如有必要可以使用临时布尔变量。

2).The nominal case SHOULD be put in the "if" part and the exception in the "else" part of an "if" statement.

3. 应避免在条件表达式中加入块。

7.杂项

1).尽量避免幻数（Magic numbers），他们应该使用常量来代替。

2).浮点变量必须指明小数点后一位（即使是 0）。

3).浮点变量必须指明实部，即使它们为零（使用 0. 开头）。

### 2.2.3语句

1.普通代码段 应该 看起来如下：

while (!isDone){

doSomething();

isDone = moreToDo();

}

2.IF 语句 应该 看起来像这样：

if (someCondition){

statements;

} else if (someOtherCondition){

statements;

} else {

statements;

}

3.FOR 语句 应该 看起来像这样：

for (initialization; condition; update){

statements;

}

4.WHILE 语句 应该 看起来像这样：

while (!isDone) {

doSomething();

isDone = moreToDo();

}

5.DO ... WHILE 语句 应该 看起来像这样：

do {

statements;

} while (condition);

6.SWITCH 语句 应该 看起来像这样：

switch (condition) {

case ABC:

statements;

// fallthrough

case DEF:

statements;

break;

default:

statements;

break;

}

7.TRY ... CATCH 语句 应该 看起来像这样：

try {

statements;

} catch(ex) {

statements;

} finally {

statements;

}

8.单行的 IF - ELSE，WHILE 或者 FOR 语句也 必须 加入括号，不过他们可以这样写：

if (condition){ statement; }

while (condition){ statement; }

for (intialization; condition; update){ statement; }

9. 操作符 建议 使用空格隔开（包括三元操作符）。

10. 下面的关键字 避免使用 空白隔开：

o break

o catch

o continue

o do

o else

o finally

o for

o function （如果为匿名函数，例如：var foo = function(){}; ）

o if

o return

o switch

o this

o try

o void

o while

o with

11. 下面的关键字必须使用空白隔开：

o case

o default

o delete

o function （如果为申明，例如：function foo(){}; ）

o in

o instanceof

o new

o throw

o typeof

o var

12.逗号（,） 建议 使用空白隔开。

13.冒号（:） 建议 使用空白隔开。

14.点（.） 在后部 建议 使用空白隔开。

15.点（.） 避免 在前部使用空白。

16.函数调用和方法 避免 使用空白，例如： doSomething(someParameter); // 而非 doSomething (someParameter)

17.逻辑块 之间使用空行。

18.声明 建议 对齐使其更容易阅读。

### 2.2.4函数

1.根命名空间和子命名空间

var figcn={};

//重复定义命名空间会清空该命名空间内的所有代码，所以根命名空间的命名最好跟公司或项目代号相关，最好不要使用com、cn等命名，这跟java的包命名方式是有区别的。

figcn.io={};

2.类

figcn.util=function(){};

3.带构造函数的类

figcn.util=function()

{

if(arguments.length==0)

this.name="default";

else

this.name=arguments[0];

};

4.类的静态字段

figcn.util.name="figcn.util";

5.类的静态函数

figcn.util.showmsg=function(){alert(arguments[0]);};

6.类的成员字段

figcn.util.prototype.classname="util";

//可以在类的成员函数中以this.classname引用

figcn.util.prototype.namespace="figcn";

7.类的成员函数

figcn.util.prototype.getFullName=function()

{

return this.namespace+"."+this.classname;

};

figcn.util.prototype.showFullName=function()

{

alert(this.getFullName());

};

### 2.2.4注释

1.生涩的代码就 没有必要 添加注释了，首先您需要重写它们。

2.所有的注释请使用英文或中文。

3.从已解决的方案到未开发的功能，注释必须与代码相关。

4.大量的变量申明后必须跟随一段注释。

5.注释需要说明的是代码段的用处，尤其是接下来的代码段。

6.注释 没有必要每行都添加。