# JavaScript开发规范

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 撰写人/修改人 | 日期 | 备注 |
| ver 1.0 | 李志晓 | 2016-03-04 | 新建 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 脚本嵌入规则

JavaScript在项目中的使用包含两种方式。第一种直接嵌入在HTML代码中，第二种建立单独的脚本文件，然后由html页面调用。

## 1.1文件方式

文件名全为小写字母，单词以下划线分割，后缀名使用\*.js。通用代码放在项目根目录下的js目录中，模块专属代码放在模块目录的js目录中。

/js/common.js // 通用的脚本库文件路径

/mms/js/mms\_add.jsp // 彩信增加模块所用的js文件路径

文件编码使用UTF-8。

文件方式的调用只使用type和src属性，如下示例：

<script type="text/javascript" src="js/filename.js"></script>

## 1.2脚本段方式

当某页面所使用的脚本只在本页面中使用，在项目的其他模块中没有详细代码时，且方法并不是很多，长度并不是很长（总长度不超过200行）的情况下可以采用脚本段方式，如果脚本过长或在项目中多次调用，请把脚本归并到.js文件在页面中调用。

把脚本段放在<head>标签中包含。尽量不要重复划分多个<script>脚本段。以下是嵌入方式的例子：

<script type=”text/javascript”>

alert(’sample’);

</script>

# 命名规范

脚本变量及方法的使用避免缩写。

## 2.1脚本段下的命名

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **结构** | **规则** | **说明** |
| 1 | 方法 | mixedCase | 多个单词组成复合词、首单词小写、其他单词首字母大写 |
| 2 | 方法内变量 | mixedCase | 多个单词组成复合词、首单词小写、其他单词首字母大写 |

### 2.1.1变量命名

1. 常量以及全局变量必须全部使用大写字母，单词之间用下划线分隔。

var MMS\_FRAME, LIMIT\_TIME, ALERT\_MESSAGE

1. 方法内变量使用多个单词组成复合词，首单词小写，其他单词首字母大写。

var arrSampleList, objSubmitButton

1. 对于指代DOM对象的变量名，尽量使用elm、obj或elm、obj开头命名。

var objSubmitButton = document.getElementById(’btn\_submit’);

function getCheckElement(name) {

var obj = document.getElementById(name); // 没有歧义的方法内部对象命名直接使用obj

return obj;

}

1. 对于指代字符串对象的变量名，尽量使用str或str开头命名。

var strMaxsizeLimit = ’已超过最大限制’;

function alertMessage(message) {

var str = message; // 没有歧义的方法内部字符串变量命名直接使用str

alert(str);

}

1. 对于指代数字的变量名，尽量使用num或num开头命名。

var numMaxCount = 300;

function isMaxCount(size) {

var num = 1000; // 没有歧义的方法内部数字命名直接使用num

return (num > size);

}

1. 布尔值的变量命名前面加"is"，同理还可以为 "has", "can" 或者 "should"。

var isChecked, hasTowApple, canDoit;

1. 循环变量使用"i", "j", "k" （依次类推）等名称的变量。

for(var i=0;i<10;i++) {

....

}

1. 用于返回结果变量使用result变量。

function getName(){

var result; // 或者使用 retName

result='sina';

return result;

}

1. 其他一些建议的约定，例如临时变量使用tmp或以tmp开头的命名；横向纵向坐标x,y；捕获的错误变量e；等等。
2. 表示数组的变量在单词前加arr进行修饰。

var arrCity=['南京','上海'];

1. 避免使用否定的布尔变量名称，例如：

isNotError,isNotFound 为非法

### 2.1.2方法命名

1. 方法命名使用多个单词组成复合词，首单词小写，其他单词首字母大写。

function openWin(){}

1. 方法命名的第一个单词尽量使用动词。例如： get/set, add/remove, create/destroy, start/stop, insert/delete, begin/end, 等等。

function getName() {}

function beginLoop() {}

## 2.2 组件下的命名

API命名风格：

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **结构** | **规则** | **说明** |
| 1 | Module（模块） | lowercase | 小写的一个英文单词，不要使用多个单词组成 |
| 2 | Class（类） | CamelCase | 多个单词组成复合词、每个单词首字母都大写 |
| 3 | method（方法） | mixedCase | 多个单词组成复合词、首单词小写、其他单词首字母大写 |
| 4 | var（变量） | mixedCase | 多个单词组成复合词、首单词小写、其他单词首字母大写 |

结构内部命名风格：

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **结构** | **规则** | **说明** |
| 1 | Module（模块） | \_mixedCase | 小写的一个英文单词，不要使用多个单词组成 |
| 2 | Class（类） | \_mixedCase | 多个单词组成复合词、每个单词首字母都大写 |
| 3 | method（方法） | \_mixedCase | 多个单词组成复合词、首单词小写、其他单词首字母大写 |
| 4 | var（变量） | \_mixedCase | 多个单词组成复合词、首单词小写、其他单词首字母大写 |

### 2.2.1 常用规范

1. 类名称必须为名词并使用骆驼命名规则：

Account, EventHandler

1. 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

}

1. 当变量名为一个缩写词时，不要使用全大写方式：

getInnerHtml(), getXml(), XmlDocument

1. 方法的命名应该为一个动词或者动词短语：

obj.getSomeValue()

1. 公有类的命名必须使用混合名称（mixedCase）命名。
2. 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（\_）：

var MyClass = function() {

var \_buffer;

this.doSomething = function() {

};

}

1. 变量如果设置为私有或者被有意定义为私有，则前面必须添加下划线：

this.\_somePrivateVariable = statement;

1. 通用的变量一般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

1. 所有的变量名使用英文名称。
2. 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。
3. 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()

1. 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

1. 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

### 2.2.2特殊规范

1. 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。
2. 错误类建议在变量名称后加上 "Exception" 或者 "Error"。
3. 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。

# 变量的定义及使用

1. 变量必须声明后再使用，必须使用var声明变量。

错误

name='jack'; // 未声明变量直接使用

错误

name='jack';

var name=’rose’; // 声明语句在使用语句之后

正确

var name; // 此处不能省略

name='jack';

1. 建议变量声明时做初始化，可用null值做初始化。
2. 变量的定义要准确，不能产生歧义。
3. 把有联系的变量尽量定义在一起。

var day,week,month;

var objButtonSetDay, objButtonSetWeek, objButtonSetMonth;

1. 变量应该放在和他有关系的代码块中。

function setName(){

~~var i = 0;~~ // 此处的声明应该放置到do some thing 下面

... do some thing ...

var i = 0; // 放在此处使代码更紧凑，也更容易理解

while(i<10){

...

}

}

1. 使变量具有最小的声明周期。

不建议使用的方式

var tmp,arrBox=[...][...];

for(var i=0;i<10;i++){

tmp=arrBox[i];

alert(tmp[0]);

}

建议使用的方式

var arrBox=[...][...];

for(var i=0;i<10;i++) {

var tmp=arrBox[i]; // tmp变量只在循环体内使用，则在循环体内声明

alert(tmp[0]);

}

1. 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。

// 不好的方式

if (a==b&&b==c&&c==d){ // 循环体太长或太复杂，不要使用此方式

alert(’ok’);

}

// 建议的方式

var isOk=(a==b && b==c && c==d);

if(isOk){

alert('ok');

}

* 1. 应避免在条件表达式中加入执行语句。

if (a=b){ // 不要使用

if (a==b){ // 正确

# 布局

## 4.1缩进

一律使用Tab进行代码缩进。

整合在HTML/JSP中的代码段首缩进，应该和<script>标签保持相同。

## 4.2每行长度

不必刻意对每行的代码长度进行控制，自然伸展代码。但在有运算符号的情况下，最好是在运算符号前换行。下一行缩进时运算符与上一行的“=”或运算符同列。

var str='sample text'

+'sample text line 2';

## 4.3块

1. 普通代码段应该看起来如下：

while(!isDone){

doSomething();

isDone=moreToDo();

}

1. if语句如下：

if(someCondition){

statements;

}else if(someOtherCondition){

statements;

}else{

statements;

}

1. for语句如下：

for(initialization;condition;update){

statements;

}

1. while语句如下：

while(!isDone){

doSomething();

isDone=moreToDo();

}

1. do...while语句如下：

do{

statements;

}while(condition);

1. switch语句如下：

switch(condition){

case ABC:

statements;

//fallthrough

case DEF:

statements;

break;

default:

statements;

//最后的case语句不需要使用break

}

1. try...catch...finally语句如下：

try{

statements;

}catch (e){

statements;

}finally{

statements;

}

1. with 语句，[**不要使用**](http://yuiblog.com/blog/2006/04/11/with-statement-considered-harmful/)**with语句**。
2. 只有一条语句的if-else, while或for块不能省略{}，也不要放在同一行上书写：

//错误

if (condition){ statement; }

while (condition){ statement; }

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

// 正确

if(confition){

statement;

}

while(condition){

statement;

}

for(intialization;condition;update){

statement;

}

## 4.4空白

1. 保留字后面应该都跟上一个空格：

var foo=function (){}

1. 分号前永远不要有空格

var city='nanjing';

1. 函数方法调用后不要加空格，例如：

doSomething(someParameter); // NOT doSomething (someParameter)

1. 如果if判断中条件较多，其中的“||”和“&&”左右的各加一个空格隔开，例如：

if(condition1 || condition2 || (condition3 && condition4)){...}

1. 一个相对独立的逻辑代码块前后可以用一个空行和前后区分开

Function doSomething(){

Logic block 1...

Logic block 2...

}

# 注释

## 5.1 行内注释

1. 全局变量要求有注释，仔细说明用途。
2. 从已解决的方案到未开发的功能，注释必须与代码保持同步。
3. 所有的注释请尽量使用中文。
4. 简短的注释可以在本行代码后空一格添加。

doSomething(); // 提高油价

1. 较长的注释放在代码段的上方并且缩进相同。

// 图标式按钮的鼠标移动效果

doSometing();

1. 连续注释超过一行的注释方法使用 /\* \*/进行段注释。

/\*

some old code...

some old code...

\*/

1. 对于用注释的方法保存的代码段，一定要在注释开头写明为什么注释掉该段代码。

/\* 因需求修改，不用再取用户数据，此处代码作废

some old code...

some old code...

\*/

## 5.2 方法注释

### 5.2.1注解原则

1. 脚本段中代码若自定义方法的代码行数超出5行（不包括正则表达式），则方法需要完整的注释。参考6.1.2节。
2. 脚本段中代码对于行数小于等于5行的方法，需在方法前按照代码段的方式进行注释，并在注释中进行参数、返回数据的说明。

// 判断给定字符是否全为a-z的字母，大小写均算

// val 输入的验证字符

// return 是否符合判断要求

var alpha = function(val) {

return /^[a-zA-Z]+$/.test(val);

};

1. 脚本文件中的方法一律要求完整的注释。参考6.1.2节。
2. 脚本文件头部需用 /\* \*/ 说明本文件的大致功能，及包含内容。
3. 如果方法因某种原因被修改，则需求增加修改历史注解。
4. 修改历史包括：修改人、修改时间、修改原因。
5. 修改历史可根据实际情况放置在方法注释或是代码段中。

// 判断给定字符是否全为a-z的字母，大小写均算

// xingming 2009-9-3 需求修改为同时支持判定小写字母 // 可以放置在此处

var alpha = function(val) {

// xingming 2009-9-3 需求修改为同时支持判定小写字母 // 也可以放置在此处

return /^[a-zA-Z]+$/.test(val);

};

### 5.2.2完整的注解要求

1. 以"/\*\*"、 "\*/"将注解封闭；
2. 注解内容按顺序包括：功能描述、参数说明、返回数据说明、作者、开发时间等信息；
3. 功能描述：内容独立一行或多行，详细描述方法的功能
4. 参数：每个参数的说明占一行，多个参数则用多行描述
5. 返回数据：返回数据描述
6. 作者：以邮箱地址的用户名为准，可以附加中文名字加以描述
7. 时间：首次编写时间,格式YYYY-MM-DD
8. 作者时间放在同一行，中间以空格分割。并保证和上面的注释保持一个空白行。

/\*\*

\* 格式化上传文件的文件名，保证不包含一些特殊字符，不超过特定长度

\* realname 传入文件名

\* return 格式化后的文件名，长度不超过20

\*

\* xingming 2009-9-3

\*/

var formatRealname = function(realname) {

if (realname.length>20) {

return realname.substr(0, 15) + '...' +

realname.substr(realname.length-4, 4);

} else {

return realname;

}

};

# 其他建议

## 6.1 对象及数组的使用

使用{}代替new Object()。使用[]代替new Array()。

当成员名是一组有序的数字时使用数组来保存数据。当成员名是无规律的字符串或其他时使用对象来保存数据。

var arrCity=new Array('南京','上海'); // 不要这样写

var arrCity=['南京','上海']; // 正确

// 不要这样写

var objPerson=new Object();

objPerson.name='jakson';objPerson.sex='male';

// 正确

Var objPerson={

name:'jakson',

sex:'male'

};

## 6.2 字符串限定符

没有特殊理由的情况下统一使用单引号作为字符串的限定符。例如：

var strIntro = 'I\’m a teacher. Say : ”hello~!”';

对于字符串中包含的单引号可以使用\进行转义。

## 6.3 恒等运算符和相等运算符

为了程序的可读性，建议少用相等运算符进行错值比较。

var a; // 如果要明确比较a是null/undefined/某类型值，不要使用==

alert(a==null); // true

alert(a===null); // false

alert(a===undefined); // true

## 6.4 减少DOM访问

* 缓存已经访问过的元素

var objButton=document.getElementById('button'); // 在使用前置入变量

for (var i=0;i<10;i++){

// 不要直接获取再操作

~~document.getElementById('button').innerHTML += i.toString();~~

// 这样正确

objButton.innerHTML+=i.toString();

}

* "离线"更新节点, 再将它们添加到树中

for (var i=0;i<items.length;i++) {

var item=document.createElement('li');

item.appendChild(document.createTextNode("Option "+i);

list.appendChild(item);

}

// 上面这样的操作是很低效的

// 正确

var fragment=document.createDocumentFragment();

for (var i=0;i<items.length;i++){

var item=document.createElement('li');

item.appendChild(document.createTextNode("Option "+i);

fragment.appendChild(item);

}

list.appendChild(fragment);

* 避免使用 JavaScript 输出页面布局--应该是 CSS 的事儿

element.style.backgroundColor='blue';

element.style.color='red';

element.style.fontSize='12em';

// 尽量不要使用javascript同时更新一个元素的style

// 应该定义class，然后更新元素的class

.newStyle { background-color: blue; color: red; font-size: 12em; }

element.className='newStyle';

## 6.5 始终使用分号

语句结束后请总是用分号结束它。

var someItem='some string';// 这里的分号不要省略，虽然对有些浏览器不用分号也可以

var doSomething=function (){

return 'something';

}; // 这里的分号不要省略，虽然对有些浏览器不用分号也可以

## 6.6 eval 、setInterval、setTimeout

我们知道eval给了我们接近javascript编译器的机会，我们可以执行一串给eval的字符串参数。实质上这样不仅降低了你脚本的性能，而且它也会给那段字符串代码过于高的权限而导致安全问题，所以避免使用eval。直接传递字符串给setInterval或是setTimeout是很低效代码也很难维护的，而且这也会遇到跟eval一样的问题，最好传递一个函数名称给它们。

// 不好的方式

setInterval("document.getElementById('container').innerHTML += 'number: '+i", 3000);

// 使用function更恰当

setInterval(someFunction,3000);

# 代码示例

## 7.1 脚本段

<script type="text/javascript">

// 当前页面的表单对象

**var** MMS\_FORM;

// 页面载入后的初始化操作

window.onload=**function**() {

// 表单对象引用

MMS\_FORM=document.forms[0];

}

// 让所有name为mmsId的checkbox元素都选中

**function** checkAllMmsid() {

**for** (**var** i=0;i<MMS\_FORM.elements.length;i++){

**if**(MMS\_FORM.elements[i].name=='mmsId') {

MMS\_FORM.elements[i].checked=all.checked;

}

}

}

// 取消单个彩信推荐的表单

// mmsId 需提交的彩信id

**function singleCancel(mmsId) {**

**MMS\_FORM.elements['method'].value='cancel';**

**MMS\_FORM.elements['ids'].value=mmsId;**

**MMS\_FORM.submit();**

**}**

// 选中的彩信推荐项目一次性提交取消

**function** batchCancel(){

**if** (check('1')){

MMS\_FORM.elements['method'].value='cancel';

MMS\_FORM.submit();

}

}

/\*\*

\* 检查用户选中彩信推荐checkbox的情况

\* sign 0 推荐操作 1 取消推荐操作

\* return 如果未选择或则不存在选项则返回false

\*

\* xingming 2009-09-01

\*/

**function** check(sign){

**var** strCheckedValue='';

**var** j=0;

**if**(**typeof** MMS\_FORM.elements['mmsId']!='undefined') {

**var** countMmsid=MMS\_FORM.elements['mmsId'].length;

// 是否存在目标选择元素

**if**(MMS\_FORM.mmsId) {

// 是否多个选择元素

// chengyao 2009-09-03 因为可能存在没有元素的情况，故增加判断逻辑

**if**(MMS\_FORM.elements['mmsId'].length){

**for**(**var** i=0;i<countMmsid;i++){

**var** objMmsid=MMS\_FORM.elements['mmsId'][i];

**if**(objMmsid.checked){

j++;

**if**(j>1){

strCheckedValue+=',';

}

strCheckedValue+=objMmsid.value;

}

}

// 单个元素的情况

} **else** **if**(MMS\_FORM.elements['mmsId'].checked){

strCheckedValue+=MMS\_FORM.elements['mmsId'].value;

}

}

**if**(strCheckedValue==''){

alert('请先选择信息');

**return** **false**;

}

MMS\_FORM.ids.value=strCheckedValue;

**return** **true**;

}**else**{

**if**(sign=='0'){

alert('没有可推荐的信息！');

}**else**{

alert('没有可取消推荐的信息！');

}

**return** **false**;

}

}

</script>

## 7.2 脚本文件

/\* 文件头部说明 \*/

/\*\*

\* 获取说明

\* element 元素

\* return 宽高对象 {top,left,rigth,bottom}

\*

\* xingming 2009-09-20

\*/

**function** getpos(element){

**if**(arguments.length!=1 || element==**null**){

**return** **null**;

}

**var** elm=element;

**var** offsetTop=elm.offsetTop;

**var** offsetLeft=elm.offsetLeft;

**var** offsetWidth=elm.offsetWidth;

**var** offsetHeight=elm.offsetHeight;

**while**(elm=elm.offsetParent){

// add this judge

**if**(elm.style.position=='absolute'

// || elm.style.position == 'relative'

|| (elm.style.overflow!='visible' && elm.style.overflow!='')){

**break**;

}

offsetTop+=elm.offsetTop;

offsetLeft+=elm.offsetLeft;

}

**return** {

top:offsetTop,

left:offsetLeft,

right:offsetWidth+offsetLeft,

bottom:offsetHeight+offsetTop

};

}

/\*\*

\* 得到给定字符的长度，中文为2个长度

\* str 给定字符

\* return 字符长度

\*

\* xingming 2009-09-20

\*/

**function** getStringByteLength(str){

**var** numLength = 0;

**for**(**var** i=0;i<str.length;i++){

**if**((str.charCodeAt(i)<0) || (str.charCodeAt(i)>255)){

numLength=numLength+2;

}**else** {

numLength=numLength+1;

}

}

**return** numLength;

}

/\*\*

\* 判断浏览器是否为ie

\* **@return** true为ie false非ie

\*

\* xingming 2009-09-20

\*/

**function** IsIE(){

**return document.all?true:false;**

}