MOVIETIME 电影售票系统

代码风格说明

系统分析与设计 MovieLight 小组

JavaScript 代码风格说明

本篇主要介绍 JS 的命名规范、注释规范以及框架开发的一些问题。

目录

1. 命名规范

驼峰式命名法介绍：

驼峰式命名法由小(大)写字母开始，后续每个单词首字母都大写。

按照第一个字母是否大写，分为：

① **Pascal Case 大驼峰式命名法**：首字母大写。eg：**S**tudent**I**nfo、**U**ser**I**nfo、**P**roduct**I**nfo

② **Camel Case 小驼峰式命名法**：首字母小写。eg：student**I**nfo、user**I**nfo、product**I**nfo

1.1 变量

命名方法：小驼峰式命名法。

命名规范：前缀应当是名词。(函数的名字前缀为动词，以此区分变量和函数)

命名建议：尽量在变量名字中体现所属类型，如:length、count 等表示数字类型；而包含 name、title 表示为字符串类型。

示例：

```
1  // 好的命名方式
2  var maxCount = 10;
3  var tableTitle = 'LoginTable';
4
5  // 不好的命名方式
6  var setCount = 10;
7  var getTitle = 'LoginTable';
```

1.2 函数

命名方法：小驼峰式命名法。

命名规范：前缀应当为动词。

命名建议：可使用常见动词约定

动词	含义	返回值
can	判断是否可执行某个动作(权限)	函数返回一个布尔值。true：可执行；false：不可执行
has	判断是否含有某个值	函数返回一个布尔值。true：含有此值；false：不含有此值
is	判断是否为某个值	函数返回一个布尔值。true：为某个值；false：不为某个值
get	获取某个值	函数返回一个非布尔值
set	设置某个值	无返回值、返回是否设置成功或者返回链式对象
load	加载某些数据	无返回值或者返回是否加载完成的结果

示例：

```
1  // 是否可阅读
2  function canRead() {
3      return true;
4  }
5
6  // 获取名称
7  function getName() {
8      return this.name;
9  }
```

1.3 常量

命名方法：名称全部大写。

命名规范：使用大写字母和下划线来组合命名，下划线用以分割单词。

命名建议：无。

示例：

```
1  var MAX_COUNT = 10;
2  var URL = 'http://www.baidu.com';
```

1.4 构造函数

介绍：在 JS 中，构造函数也属于函数的一种，只不过采用 new 运算符创建对象。

命名方法：大驼峰式命名法，首字母大写。

命名规范：前缀为名称。

命名建议：无。

示例：

```
1  function Student(name) {
2      this.name = name;
3  }
4
5  var st = new Student('tom');
```

1.5 类的成员

类的成员包含：

① 公共属性和方法：跟变量和函数的命名一样。

② 私有属性和方法：前缀为_(下划线)，后面跟公共属性和方法一样的命名方式。

示例：

```
1  function Student(name) {
2    var _name = name; //  私有成员
3
4    //  公共方法
5    this.getName = function () {
6      return _name;
```

```
7     }
8
9     // 公共方式
10    this.setName = function (value) {
11        _name = value;
12    }
13 }
14 var st = new Student('tom');
15 st.setName('jerry');
16 console.log(st.getName()); // => jerry：输出_name 私有变量的值
```

2. 注释规范

JS 支持两种不同类型的注释：单行注释和多行注释。

2.1 单行注释

说明：单行注释以两个斜线开始，以行尾结束。

语法：// 这是单行注释

使用方式：

① 单独一行：//(双斜线)与注释文字之间保留一个空格。

② 在代码后面添加注释：//(双斜线)与代码之间保留一个空格，并且//(双斜线)与注释文字之间保留一个空格。

③ 注释代码：//(双斜线)与代码之间保留一个空格。

示例：

```
1 // 调用了一个函数；1)单独在一行
2 setTitle();
3
4 var maxCount = 10; // 设置最大量；2)在代码后面注释
5
6 // setName(); // 3)注释代码
```

2.2 多行注释

说明：以/*开头，*/结尾

语法：/* 注释说明 */

使用方法：

① 若开始(/*)和结束(*/)都在一行，推荐采用单行注释。

② 若至少三行注释时，第一行为/*，最后行为*/，其他行以*开始，并且注释文字与*保留一个空格。

示例：

```
1  /*
2  * 代码执行到这里后会调用 setTitle()函数
3  * setTitle()：设置 title 的值
4  */
5  setTitle();
```

2.3 函数(方法)注释

说明：函数(方法)注释也是多行注释的一种，但是包含了特殊的注释要求，参照 javadoc(百度百科)。

语法：

/**

* 函数说明

* @关键字

*/

常用注释关键字：(只列出一部分，并不是全部)

注释名	语法	含义	示例
@param	@param 参数名 {参数类型} 描述信息	描述参数的信息	@param name {String} 传入名称
@return	@return {返回类型} 描述信息	描述返回值的信息	@return {Boolean} true:可执行;false:不可执行

@author	@author 作者信息 [附属信息：如邮箱、日期]	描述此函数作者的信息	@author 张三 2015/07/21
@version	@version XX.XX.XX	描述此函数的版本号	@version 1.0.3
@example	@example 示例代码	演示函数的使用	@example setTitle('测试')

示例：

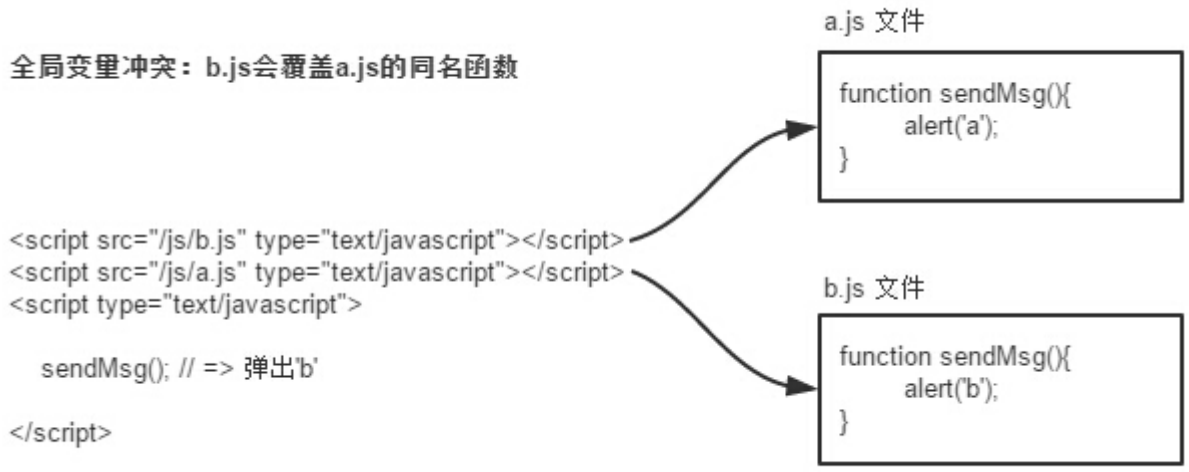
```
/**
* 合并 Grid 的行
* @param grid {Ext.Grid.Panel} 需要合并的 Grid
* @param cols {Array} 需要合并列的 Index(序号)数组；从 0 开始计数，序号也包含。
* @param isAllSome {Boolean} ：是否 2 个 tr 的 cols 必须完成一样才能进行合并。true：完成一样；false(默认)：不完全一样
* @return void
* @author polk6 2015/07/21
* @example
* ┌──────┬──────┐                              ┌──────┬──────┐
* │ 年龄 │ 姓名 │                              │ 年龄 │ 姓名 │
* ├──────┼──────┤      mergeCells(grid,[0])    ├──────┼──────┤
* │ 18   │ 张三 │              =>              │      │ 张三 │
* ├──────┼──────┤                              │ 18   ├──────┤
* │ 18   │ 王五 │                              │      │ 王五 │
* └──────┴──────┘                              └──────┴──────┘
*/
function mergeCells(grid, cols, isAllSome) {
  // Do Something
}
```

3. 框架开发

3.1 全局变量冲突

在团队开发或者引入第三方 JS 文件时，有时会造成全局对象的名称冲突，比如 a.js 有个全局函数 sendMsg()，b.js 也又有个全局函数 sendMsg()，引入 a.js 和 b.js 文件时，会造成 sendMsg()函数冲突。

示例：



3.2 单全局变量

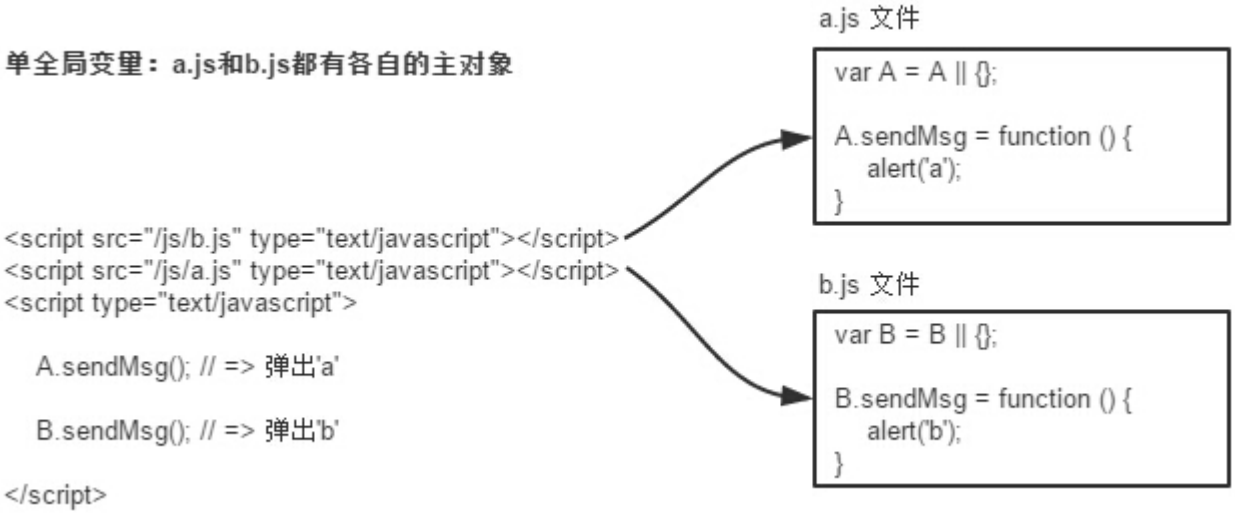
所创建的全局对象名称是独一无二的，并将所有的功能代码添加到这个全局对象上。调用自己所写的代码时，以这个全局对象为入口点。

如：

* JQuery 的全局对象：$和 JQuery

* ExtJS 的全局对象： Ext

示例：



3.3 命名空间

在项目规模日益壮大时，可采用命名空间方式对 JS 代码进行规范：即将代码按照功能进行分组，以组的形式附加到单全局对象上。

以 Ext 的 chart 模块为例：