**文件组织**

* 所有的javascript文件都要放在项目公共的'''script'''文件夹下
* 使用的第三方库文件放置在'''script/lib'''文件夹下
* 可以复用的自定义模块放置在'''script/commons'''夹下,复用模块如果涉及多个子文件，需要单独建立模块文件夹
* 单独页面模块使用的javascript文件放置在'''script/{module\_name}'''文件夹下
* 项目模拟的json数据放置在'''script/json'''文件夹下，按照页面单独建立子文件夹
* javascript应用MVC框架时，使用的模板文件放置在'''script/templates'''文件夹下，按照页面单独建立子文件夹

**格式化**

* 通用原则可以参考[PhpGuideline PHP编码规范]
* 始终使用var定义变量

var global = '';

function method() {

var local = '';

}

* 始终使用分号结束一行声明语句
* 对于数组和对象不要使用多余的","（兼容IE）

//wrong

var arr = [1,2,];

var person = {

name: 'name',

age: 20,

};

//wrong

var arr = [1,2];

var person = {

name: 'name',

age: 20

};

* 定义顶级命名空间如inBike,在顶级命名空间下自定义私有命名空间，根据模块分级
* 所有的模块代码放在匿名自调用函数中，通过给window对象下的自定义命名空间赋值暴露出来

if (!window.inBike) {

window.inBike = {};

}

window.inBike.rideway = rideway;

* 绑定事件代码需要放置在dom ready函数中执行

$(function() {

//bind function is in the init function

window.inBike.rideway.init();

})

* 将自定义模块方法放置在对象中，方法名紧挨":"，":"与function之间空一格，function()与后面的"{"之间空一格

var module = function() {

method: function() {

}

};

* 使用单引号包裹字符串

var str = 'some text';

* 所用的变量使用之前需要定义，定义之后立即初始化

var obj = null;

var num = 0;

var arr = [];

var isEmpty = true;

* 使用浏览器console工具之前先要判断是否支持

if (console) {

console.log('this is my log');

}

**命名**

* 通用原则可以参考[PhpGuideline PHP编码规范]
* 使用驼峰法命名变量和方法名，首字母使用小写,对于类名首字母大写

var numberList = [1,2];

var util = {

removeNode: function(){

....

}

};

function Person(name, age) {

this.name = name;

this.age = age;

}

* 使用$name命名jquery对象，原生dom元素使用dom开头,对象中私有变量以下划线开头

var $image = $('#cover');

var domImage = document.getElementById('cover');

var obj = {

\_privateVar: null,

method: function() {

}

};

**注释**

* 通用原则可以参考[PhpGuideline PHP编码规范]
* 多使用单行注释表明逻辑块的意义

// Handle the case where IE and Opera return items

// by name instead of ID

if ( elem.id !== match[2] ) {

return rootjQuery.find( selector );

}

// Otherwise, we inject the element directly into the jQuery object

this.length = 1;

this[0] = elem;

* 指明类的构造方法

/\*\*

\* @constructor

\*/

some.long.namespace.MyClass = function() {

};

* 标注枚举常量的类型和意义

/\*\* @enum {string} \*/

some.long.namespace.Fruit = {

APPLE: 'a',

BANANA: 'b'

};

* 使用注释标识方法或者变量的可见性，方便静态检查

/\*\* @protected \*/

AA\_PublicClass.staticProtectedProp = 31;

/\*\* @private \*/

AA\_PublicClass.prototype.privateMethod\_ = function() {};

**最佳实践**

* 避免使用eval
* 对于对象避免使用with，对于数组避免使用for-in
* 谨慎使用闭包，避免循环引用
* 警惕this所处的上下文

var $button = $('#my-button');

$button.click(function(){

var self = this;

var util = {

getVal: function() {

return self.val();

}

}

});

* 尽量使用短码，比如三目运算符，逻辑开关，自增运算等

var name = ('undefined' == typeof(name)) ? '' : name;

(age < 0) && (age = 0);

count++;

* 不要在块级作用域中使用function

// 错误

if (x) {

function foo() {}

}

// 正确

if (x) {

var foo = function() {}

}

* 在父节点上绑定事件监听，根据事件源分别响应
* 对于复杂的页面模块使用依赖管理库如SeaJS, RequireJS, MVC框架Backbone, Knockout, Stapes组织你的代码