

# JavaScript编码实践

@董金虎

# 使用CDN加速静态资源下载

如果有时CDN上资源下载不下来，需要下载自己服务器上的资源

# 测试JS执行效率

使用jsperf。或者自定义：

```
14 function PerfTest(implement, params, times) {
15     this.implement = implement;
16     this.params = params;
17     this.times = times || 10000;
18     this.average = 0;
19 }
20
21 PerfTest.prototype = {
22     run: function(){
23         var beginAt, endAt, sumTime = 0;
24         for(var i = 0, times = this.times; i < times; i++) {
25             beginAt = +new Date();
26             this.implement(this.params); //调用测试函数
27             endAt = +new Date();
28             sumTime += endAt - beginAt;
29         }
30         this.average = sumTime / this.times;
31         return console.log("Average executated " + this.times + " times and time is : " + this.average );
32     }
33 }
34
35 var list = [1,2,3,4,5,6,7,8,9,10];
36 var testFunc = function(list){
37     var arr = [];
38     for(var i = 0; i < list.length; i++) {
39         arr.push(list[i]);
40     }
41 }
42
43 var test = new PerfTest(testFunc, list);
44 //var test = new PerfTest(testFunc, list, 100000); //指定执行次数
45 test.run();
```

# JS加载位置

需要在DOM渲染之前就需要执行的操作  
(一般是初始化),在header中加载,剩下的放在body底部加载。

# 压缩代码

YUI Compressor  
UglifyJS  
CSS Compressor  
CSS Minifier



# 将配置数据，翻译，常量从代码中分离

```
var SN = SN || {};  
SN.Config = {  
  //Common message  
  messages: {  
    greeting: 'Hello World!'  
  },  
  
  //Error message  
  errors: {  
    mobile: {  
      blank: '电话号码不能为空',  
      invalid: '电话格式不正确'  
    }  
  }  
}  
  
$('#mobile').tooltip(SN.Config.errors.mobile.blank);
```

# 测试你的代码

Jasmine (BDD)

Qunit

Mocha

Sinon

# 组织代码

core: 核心封装和接口, Ajax, Common, Util...

config: 配置文件, 语言, 消息, 常量等等

models: 数据存储, localStorage, ...

controllers: 处理用户界面交互

views: 数据展示



# 模块模式

```
/**
 * The Mobile Common Object
 * @class Common
 * @static
 */
'use strict';
var SN = SN || {};
SN.Common = (function(){

    var _getUrlParams = function(ulr){
        //...
    }

    //...

    return {
        /**
         * Get the URL params
         * @params {string} url request url
         * @returns {Array} url params array
         * @method _getUrlParams
         */
        getUrlParams: function(url){_getUrlParams(url);}
    }

})();

//调用
SN.Common.getUrlParams('xxx');
```

# 语句加上分号

```
function test() {  
  return {  
    name: 'foobar'  
  };  
}
```

# 判断使用===

```
'1' === 1 // false  
'' === 0 // false  
false === '0' // false  
' \t\r\n ' === 0 // false
```

# 少用wrapper对象

```
var num = new Number(8);  
var str = new String('foobar');  
console.log(typeof num); //object  
console.log(typeof str); //object  
var str2 = 'foobar';  
str2.replace('foobar', 'hello, world') // str2先转化为内建包装对象
```

## 反问自己的属性

定义类时，应该把公用的属性，或者方法放在原型链上，高效利用内存。

```
for (var prop in obj) {  
  if (d.hasOwnProperty(prop)) {  
    console.log( prop + ": " + obj[prop] );  
  }  
}
```



# 少用with, eval

```
var obj = { name: "foobar"},
name = "helloworld";

with (obj) {
  console.log(name); // 如果变量和对象属性同名, 使用变量 => helloworld
  occupation = "IT"; // 对象无法带着这个属性出去, 但是它自己可以出去
}

console.log(obj.occupation); // undefined;
console.log(occupation); // IT

eval('console.log("Hello, World")'); //可以执行javascript文本
```

# parseInt带上进制

```
parseInt("10"); // 10  
parseInt("010"); // 8, 发现0开头, 认为要转成8进制  
parseInt(val, 10);
```

# 不要遵守所有规则

with, eval也有它的用处,有些好的实践方式在特定的环境下，可能会出问题，所以可以变通。

# 高效迭代访问

```
var arr = [1,2,3,4,5,6,7,8,9,10],  
    len = arr.length,  
    index = len;  
  
while(index--) { // 0为false, 不要比较, 这样效率会更好一些  
    console.log(arr[index]);  
}
```

# 用户自定义方法名称以动词开头

```
function getName(person) {  
    //...  
}
```



注意在闭包内部引入闭包外部的变量，当闭包执行完成后，对象无法被GC回收

```
var a = function() {  
  var numArr = [1,2,3];  
  return function() {  
    return numArr;  
  };  
}();
```

如果在一个div下面有多个按钮，或者a需要侦听事件，请  
直接在这个div上面监听事件。  
善用事件代理，可以提高效率

# 使用三目运算符

```
var authorized = false,  
    active;  
  
if(authorized) {  
    active = true;  
} else {  
    active = false;  
}  
  
//=>  
  
var active = authorized ? true : false;
```

# 短路运算符

```
var class = {
  _addMember: function(member){
    this.members = this.members ? this.members : [];
    this.members.push(member);
  }
};

//=>

var class = {
  _addMember: function(member){
    this.members = this.members || [];
    this.members.push(member);
  }
};
```

# switch语句

```
function findKindVal(kind){  
    if(kind == 'a') {  
        return 0;  
    } else if(kind == 'b') {  
        return 1;  
    } else if(kind == 'c') {  
        return 2;  
    }...  
}
```

//=>

```
function findKindVal(kind) {  
    switch(kind) {  
        case 'a':  
            return 0;  
        case 'b':  
            return 1;  
        case 'c':  
            return 2;  
        case 'd':  
        case 'e':  
            return 3;  
        default:  
            return '-1';  
    }  
}
```



# 使用createDocumentFragment添加迭代DOM

```
var list = document.getElementById('book_list');
for(var i = 0; i < 10; i++) {
    var element = document.createElement('li');
    element.appendChild(document.createTextNode('book_' + i));
    list.appendChild(element);
}

//=>

var list = document.getElementById('book_list'),
    fragment = document.createDocumentFragment(),
    element;

for(var i = 0; i < 10; i++) {
    element = document.createElement('li');
    element.appendChild(document.createTextNode('book_' + i));
    fragment.appendChild(element);
}

list.appendChild(fragment);
```

## 减少var定义变量

每使用一个var定义变量都会给JavaScript解析器添加一条查找路径，可以使用,将多个变量定义使用一个var定义。

```
var a = 1,  
    b = 2,  
    foo = 'bar';
```

## 在循环外定义变量

JavaScript作用域是函数级别的，不是代码块，所以在循环块外定义变量，避免在循环内一次又一次定义变量。

```
var element;  
for(var i = 0; i < 10; i++) {  
    element = document.createElement('li');  
}
```

# 使用join方法链接字符

```
var strs = ['a', 'b', 'c'],  
    len = strs.length,  
    result = '';  
  
for(var i = 0; i < len; i++) {  
    result += strs[i];  
}  
  
//=>  
  
var result = strs.join('');
```

# jQuery, Zepto级连调用

```
$obj.removeClass("xxxx").addClass("xxx");  
$("#xxxx").unbind("click").bind('click',function() {  
    //...  
});
```



# URL中变量要转义

```
location.href = "./error.html?errorMsg=" + encodeURIComponent(someVariable);
```

# jQuery, Zepto 合并或者更新对象

```
obj["a"] = a;  
obj["b"] = b;  
  
//=>  
  
obj = $.extend(obj, {  
  a: xxxx  
  //...  
});
```

# 尽量使用字面量对象声明的对象

```
var arr = new Array();
```

```
//=>
```

```
var arr = [];
```

# 自动化构建工具

Grunt

Gulp

Browserify

Webpack

# JavaScript转译语言

CoffeeScript  
TypeScript



在线编辑

jsfidde  
codepen

# 检查代码质量

jshint  
plato

# 代码Review

It is about code, not about coder.

保证:

编码规范

代码质量

简单易懂

可维护性

性能

类似马斯洛需求 :)

Thank You !