**二十三 、闭包**

**一、购物车**

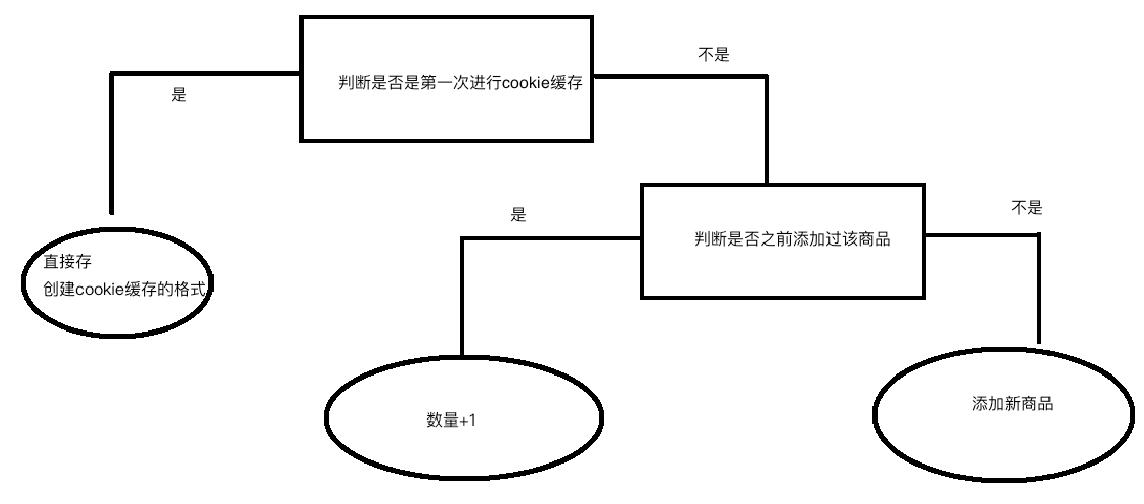
**1.首先获取data.json中的数据，将数据进行解析，添加到页面上**



**效果如图：**



**2.给购物车按钮添加事件(存储cookie)**



**$(".goods\_box").on("click", ".sc\_btn", function(){**

**// alert(this.id);**

**//是否是第一次添加cookie**

**var id = this.id;**

**var first = $.cookie("goods") == null ? true : false;**

**if(first){**

**//第一次添加 [{id:id,num:2}]**

**$.cookie("goods", '[{id:' + id + ',num:1}]', {**

**expires: 7**

**});**

**}else{**

**var str = $.cookie("goods");**

**var arr = eval(str);**

**var same = false; //代表是否有相同商品**

**//遍历所有的对象，判断是否id相同，num++**

**for(var i in arr){**

**if(arr[i].id == id){**

**arr[i].num = arr[i].num + 1;**

**var cookieStr = JSON.stringify(arr);**

**$.cookie("goods", cookieStr, {**

**expires: 7**

**});**

**same = true;**

**break;**

**}**

**}**

**//没有相同的商品**

**if(!same){**

**var obj = {id: id, num: 1};**

**arr.push(obj);**

**var cookieStr = JSON.stringify(arr);**

**$.cookie("goods", cookieStr, {**

**expires: 7**

**});**

**}**

**sc\_car();**

**}**

**alert($.cookie("goods"));**

**//页面控件非常多，非常容易叠加，很容易造成事件冒泡**

**return false;**

**})**

**【补充】ECMA5新增方法**

**JSON.parse()**

**JSON.stringify()**

**【注】低版本浏览器解析JSON数据的函数**

**eval() 进行解析 对于json数据的格式要求非常高，最外层必须是数组，里面的元素必须是对象**

**3.给侧边栏添加添加移入移出事件**



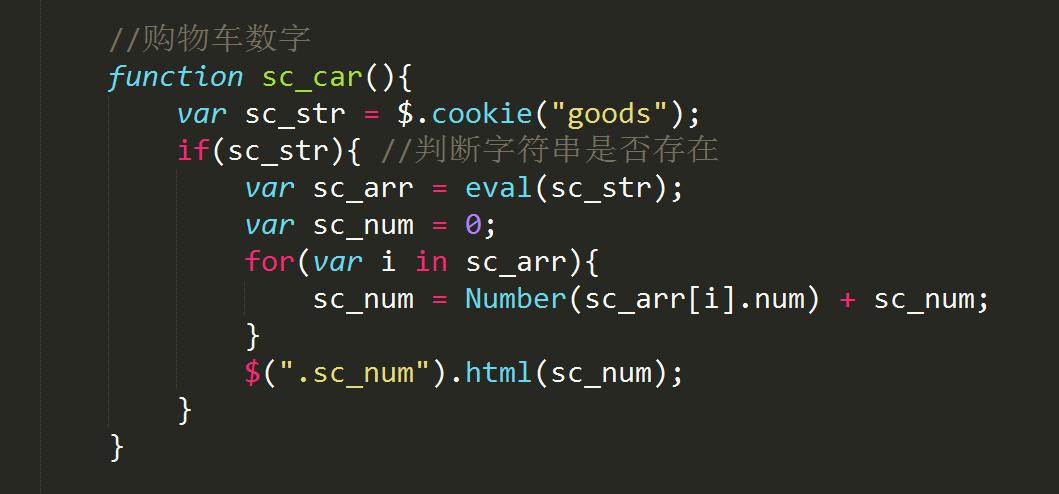
**【注】在这里添加移入移出事件使用mouseenter和mouseleave。**

**【注】mouseenter和mouseover的区别：**

与 mouseenter 事件不同，不论鼠标指针穿过被选元素或其子元素，都会触发 mouseover 事件。只有在鼠标指针穿过被选元素时，才会触发 mouseenter 事件。

**4.购物车数字的变化**

【注】在页面加载后以及“加入购物车”按钮点击事件中调用。



**效果如图：**

0248944352774452BCB5B164BEB47485.jpg

**5.对已经存储在cookie中的数据进行加载，并展示到侧边栏中。**

【注】在侧边栏移入事件中调用。



**效果如图：**



**================================================**

**二、认识闭包**

**1.闭包特点**

（1）函数嵌套函数

（2）内部函数可以引用外部函数的参数和变量

（3）参数和变量不会被垃圾回收机制所回收

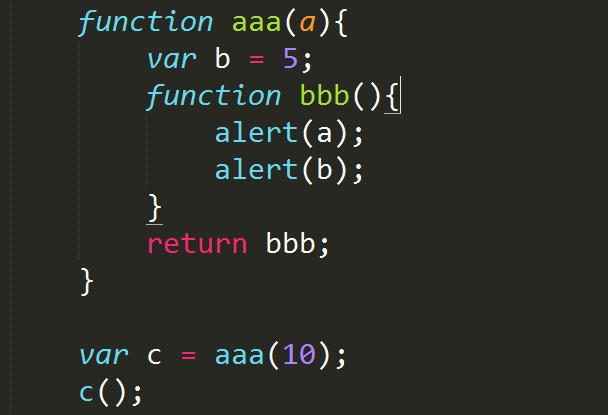
【例】

(1)外部函数aaa里面嵌套了一个内部函数bbb；

(2)内部函数bbb引用了外部函数的参数a和变量b；

(3)执行外部函数aaa的返回值为bbb这个内部函数，将外部函数赋值给一个全局变量aaa,此时aaa中的参数与局部变量会一直保存在内存中，而不会在函数调用之后清除。

**原因就在于aaa是bbb的父函数，而bbb被赋给了一个全局变量，这导致bbb始终在内存中，而bbb的存在依赖于aaa，因此aaa也始终在内存中，不会在调用结束后，被垃圾回收机制（garbage collection）回收。**



**2.闭包的好处**

（1）希望有一个变量长期驻扎在内存中

（2）避免全局变量的污染

（3）私有成员存在

用法：

（1）在循环中直接找到对应元素的索引

（2）模块化代码

=====================

当声明全局变量a，可实现a的累加

var a = 1;

function aaa(){

a++;

alert(a);

}

aaa(); //2

aaa(); //3

=============================

当声明函数内部变量a，由于函数调用后即销毁，a的值始终为2

function aaa(){

var a = 1;

a++;

alert(a);

}

aaa(); //2

aaa(); //2

=============================

【注】如何做到既让a进行累加，又让a不在外部能被访问到呢。有，闭包。

第一种写法：

function aaa(){

var a = 1;

return function(){

a++;

alert(a);

}

}

var b = aaa();

b(); //2

b(); //3

第二种写法：

var aaa = (function(){

var a = 1;

return function(){

a++;

alert(a);

}

})();

aaa(); //2

aaa(); //3

**3.函数表达式写法**

(function(){

alert(1);

})()

需要在函数function外加（）。

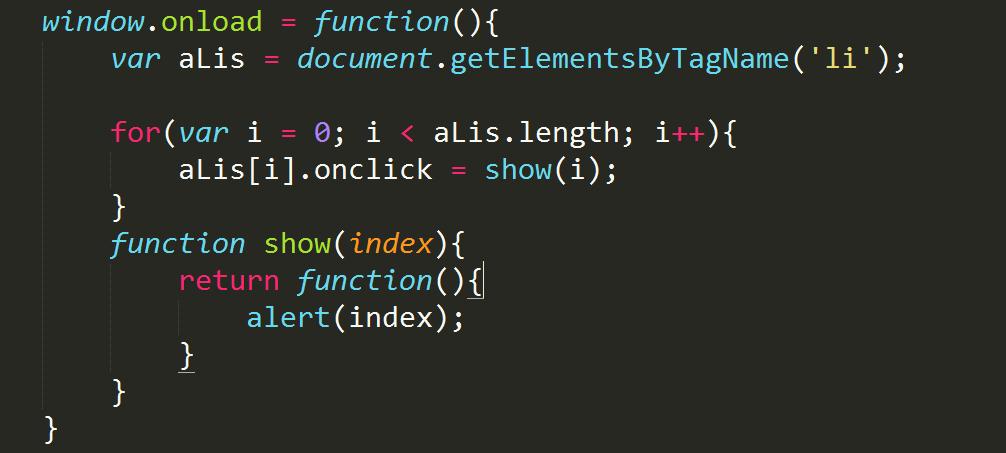
**4.应用**

（1）在循环中直接找到对应元素的索引

第一种写法：



第二种写法：



（2）模块化开发

【私有变量和私有方法】

//模块化开发

var aaa = (function(){

var a = 1; //私有变量

//【注】私有变量可以被函数的内部方法所访问。

function bbb(){ //私有方法

a++;

alert(a);

}

function ccc(){

a++;

alert(a);

}

return {

b: bbb,

c: ccc

}

})();

aaa.b(); //2

aaa.c(); //3

alert(bbb);//ReferenceError: bbb is not defined

alert(ccc);

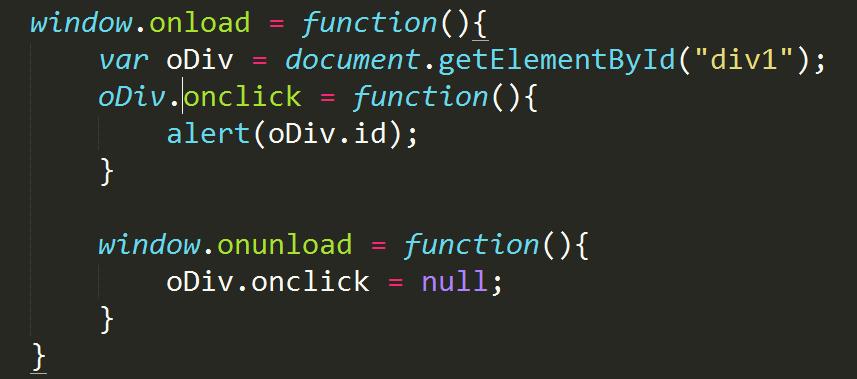
**5.闭包需要注意的行为**

**IE下引发内存泄漏**

由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题，在IE中可能导致内存泄露。解决方法是，在退出函数之前，将不使用的局部变量全部删除。

第一种写法：

当页面解构（关闭）时，将函数删除。



第二种写法：

将所求值单独赋值，销毁其他不使用的（oDiv = null;）

