**内存泄露，闭包惹的祸？**

来源：[jayceLin](http://home.cnblogs.com/u/jaycelin/) 日期：2011-11-24 13:12:19 编辑：admin 阅读 **107** 次

看完what is closure我们大概对闭包会有一定了解了吧！但是为什么js要有闭包呢，为什么总是有那么多的攻城师们吵吵嚷嚷的说没有闭包过不下去了，作为能让javascript成为一门有个性的语言的内涵之一，它究竟是怎样呢？闭包是内嵌型语言（可嵌套作用域，如js,ruby,python）的专有，在网上看到人家的总结：

* 函数是一阶值（First-class value），即函数可以作为另一个函数的返回值或参数，还可以作为一个变量的值。
* 函数可以嵌套定义，即在一个函数内部可以定义另一个函数。

进入正题，闭包真的拥有很多很有的作用，但他也有弊端！

**闭包的内存泄露问题**

大家都知道闭包没有释放外部变量，确实会造出一点泄露，但你会说“别死抠了！”是的，真正的问题不是在这里，而是隐藏在我们显见的编程中！在这里不得不插一下DOM与JS这两个冤家。

**DOM与JS**

因为在浏览器解析中DOM与js分开为两个解析器的（例如在IE分别为mshtml.dll跟jscript.dll），为什么呢，可能是为了让其它脚本语言可以对DOM进行引用吧，像VB script！但这样就给了js中DOM操作一个非常大的性能问题，很明显吧！DOM提供的API被引用时，借口的连接就会造成消耗，时间、内存都是问题！就像你去食堂打饭，饭和菜是分开打的，那么我得多花时间跟ATP才能吃上饭！ 而造成内存最多的是循环引用（问题值出现在IE与firefox），何为循环引用呢？ 当程序中出现了两个对象，并且它们之间保持着相互引用的状态，就称为循环引用！ IE 和 Firefox 均使用引用计数来为 DOM 对象处理内存。在引用计数系统，每个所引用的对象都会保留一个计数，以获悉有多少对象正在引用它。如果计数为零，该对象就会被销毁，其占用的内存也会返回给堆。所以当DOM遇上循环引用时，你的js程序就存在性能问题了！而闭包就是造成这个问题的一个不经意的杀手！也就是说循环引用经常会出现在闭包中，从而让闭包背上了这个恶名，它好无辜！举个简单的例子： 给一个列表绑定click事件，输出他的顺序！

<ul>

<li>第一个</li>

<li>第二个</li>

<li>第三个</li>

<li>第四个</li>

<li>第五个</li>

</ul>

(function (){

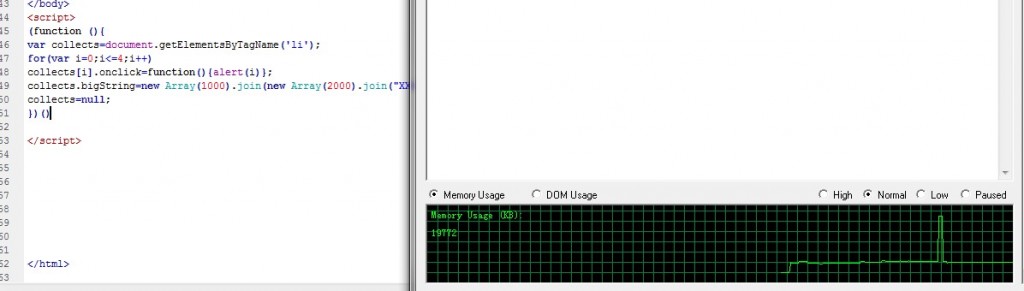
var collects=document.getElementsByTagName('li');

for(var i=0;i<=4;i++)

collects[i].onclick=function(){alert(i)};

collects.bigString=new Array(1000).join(new Array(2000).join("XXXXX"));

})()

为了使内存对比明显，我特意加上collects.testString=new Array(10000).join(new Array(1000));我通过了ie内存泄漏监测工具对比解决了泄漏问题前后的内存状况： [](http://files.blogcn.com/wp01/M00/01/B7/wKgKCk5gifgAAAAAAAD2IgPFCOc540.jpg)[](http://files.blogcn.com/wp04/M00/01/DC/wKgKDE5gihQAAAAAAAEPV14YsgI073.jpg)左边是DW中的代码！在删掉collects=null前后的内存确实变化超明显！ 解决的方法很简单，既然是对象关联，那么你只要将其中一个释放了就可以了！ 可能细心的同学还会注意到上面还隐藏着常见闭包引起的修改外部变量问题，本来想输出各个li的顺序，结果都是输出5；这是因为i是属于内部函数的外部变量，而当调用onclick时会在闭包的作用域链中记录下i的值！解决的方法很简单，让i成为内部函数的局部变量就可以了！

(function (){

var collects=document.getElementsByTagName('li');

for(var i=0;i<=4;i++)

(function(k)

{

collects[k].onclick=function(){alert(k)};

})(i)

})()