# [Javascript--闭包引起的IE内存泄露](http://www.cnblogs.com/lufy/archive/2012/05/18/2507536.html)

function fors(){

obj\_a = obj\_b;

obj\_b.attr = obj\_a;

}

function fors(){

obj\_b = {};

obj\_b.attr = obj\_b;

}

上面是两个个很显示的循环引用，IE中产生了内存泄露，由于IE的内存回收机制，导至会长期占用内存而不能释放。

但闭包的内存泄露，有些隐蔽。因为闭包的循环引用,是间接的。

[复制代码](javascript:void(0);)

function iememery(){

var js\_obj = document.createElement("div");

js\_obj.oncontextmenu = function(){ return false;}

js\_obj = null;//加上这句，断开引用

}  
  
<body onload="iememery()">

[复制代码](javascript:void(0);)

从表面上看，没有任何循环引用。但上面是一个闭包，根据闭包的特性，内部函数有权访问外部函数的变量对象。所以当iememery()执行之后:

js\_obj是一个DOM元素的引用，DOM元素它长期在网页当中，不会消失，而这个DOM元素的一属性oncontextmenu，又是内部的函数引用（闭包），而这个匿名函数又和js\_obj之间有隐藏的关联（作用域链）

所以形成了一个，循环引用.即：

js\_obj.oncontextmenu 间接引用到 js\_obj   
  
也就是说，这个对象的一个属性，又间接的引用了自己。

只要有循环引用,就会在IE下产生内存泄露。  
  
打开你的windows任务管理器，在IE中不停刷新含有这个代码的html页面，看看Iexploer进程的内存占用情况，一直上升，且不会自动回收（降低）；

解决办法：

[复制代码](javascript:void(0);)

function iememery(){

var js\_obj = document.createElement("div");

js\_obj.oncontextmenu = function(){ return false;};

　　　　　　js\_obj.oncontextmenu = null;//加上这句，断开引用 }

[复制代码](javascript:void(0);)

当IE中发生js对象与dom对象直接的循环引用，并且之后没有引用指向他们，  
如果是IE 6, 内存泄漏，直到关闭IE进程为止http://www.blogjava.net/CuteSoft_Client/CuteEditor/images/emcry.gif  
如果是IE 7，内存泄漏， 直到离开当前页面为止http://www.blogjava.net/CuteSoft_Client/CuteEditor/images/emdgust.gif  
如果是IE 8， GC回收器回收他们的内存，无论当前是不是compatibility模式。http://www.blogjava.net/CuteSoft_Client/CuteEditor/images/emteeth.gif  
之前的IE js引擎里的GC回收器只能处理js对象，不能处理DOM对象。  
  
由于IE9之前的版本对JScript对象和COM对象使用不同的垃圾收集例程，因此闭包在IE的这些版本中会导致如上问题。这个匿名函数作为element元素事件处理程序，形成闭包的状态就会保存对父层函数内活动对象的引用，只要匿名函数存在，element的引用数至少也是1，因此它所占用的内存就永远不会被回收。注意，闭包会引用包含函数的整个活动对象，即使闭包不直接引用element，包含函数的活动对象中也仍然会保存一个引用。因此，有必要把使用完的element变量设置为null，解除对DOM对象的引用，确保正常回收其占用的内存。