[**JavaScript的垃圾回收机制**](http://www.cnblogs.com/jenry/archive/2011/02/13/1953265.html)

JavaScript语言是一门优秀的脚本语言。其中包含脚本语言的灵活性外还拥有许多高级语言的特性。例如充许构建和实例化一个对象，垃圾回收机制(GC:Garbage Collecation)。通常我们使用new创建对象，GC负责回收对象占用内存区域。因此了解GC，可以加深对javascript垃圾回收机制的理解。

GC在回收内存时，首先会判断该对象是否被其它对象引用。在确定没有其它对象引用便释放该对象内存区域。因此如何确定对象不再被引用是GC的关键所在。

<script type="text/javascript">  
function aa()  
{  
 this.rr = "弹窗";   
}  
  
function bb()  
{  
 this.rr = "弹窗";  
}  
  
var b1;  
  
function cc()  
{  
 var a1 = new aa();  
 b1 = new bb();  
 return b1;  
}  
  
cc();  
alert(b1.rr)  
</script>

如上代码中，执行完cc()后a1被回收了，此后我们可以通过b1.rr弹出文字窗口。在一些基础书籍中解释为:a1为局部变量，b1是全局变量。局部变量执行完后会被GC回收.但不全是这样，如下代码：

<script type="text/javascript">  
function aa()  
{  
 this.rr = "弹窗";   
}  
  
function bb()  
{  
 this.rr = "弹窗";  
}  
  
function cc()  
{  
 var a1 = new aa();  
 var b1 = new bb();  
 return b1;  
}  
  
var b1 = cc();  
alert(b1.rr);  
</script>

此时cc函数中的 a1，b1都是局部变量，但仍然会弹出文字窗口。说明b1并没有被GC回收。因此javascript中局部变量不是所有时候都被GC回收的。

GC回收机制还需要近一步了解。在此时引入几个概念：双向链表，作用域链，活动对象(为了方便理解，简化了原文的概念[http：//softbbs。pconline。com。cn/9497825。html]) ， 其中双向链表描述复杂对象的上下层级关系。 作用域链与活动对象分别是双向链表中的某个节点。以函数cc为例变量层级关系为：

window<=>cc<=>a1<=>rr  
<=>b1<=>rr

在执行cc()方法时，内存中变量的引用关系如上图，文字解释如下：

1. cc的活动对象包括a1和b1，其作用域链是window
2. cc的活动对象包括a1和b1，其作用域链是window
3. a1的活动对象包括rr，其作用域链是cc
4. b1的活动对象包括rr，其作用域链是cc

执行cc()时，cc的执行环境会创建一个活动对象和一个作用域链。其局部变量a1，b1都会挂在cc的活动对象中。当cc()执行完毕后，执行环境会尝试回收活动对象占用的内存。但因局部变量b1 通过return b1，为其增加了一条作用域链：window<=>b1<=>rr，所以GC停止对b1回收。因此如果想将一个局部变量/函数提升为全局的，为其增加一条作用域链就OK了。

同时控制好对象的作用域链也变得重要了。因作用域链会意外导致GC无法回收目标对象。例如：

<SCRIPT LANGUAGE="JavaScript">  
<!--  
//猫  
function cat(name)  
{  
 var zhuren ;  
 this.name = name;  
   
 //设置主人  
 this.addZhuRen = function(zr){  
 zhuren = zr;  
 }  
  
 this.getZhuRen = function(){  
 return zhuren;  
 }  
}  
   
//主人  
function zhuren(name){  
 this.name = name;  
}  
  
//创建主人:  
var zr = new zhuren("zhangsan");  
//创建猫  
var cat1 = new cat("asan");  
//设置该猫的主人  
cat1.addZhuRen(zr);  
//释放主人  
zr = null ;  
//此处还存在对主人对象的引用  
alert(cat1.getZhuRen().name)  
//-->  
</SCRIPT>