1. 什么是闭包

闭包是一个函数，能够读取其他函数内部变量的函数。任何函数都可以被看成闭包。并不一定要 return。

一个闭包就是当一个函数返回时，一个没有释放的资源栈区。

JS作用域外部无法访问内部函数变量，闭包相当于一座桥梁，把函数内部和外部连接起

常见的闭包形式：

Var bar = function foo(x){

var num=1;

return function(y){

return num+x+y;

}

}(3); //bar是一个闭包

Bar(4); //8

DOM中：

function assignHandler(){

var element =document.getElementByID("someElement");

var id=element.id;

element.onclick = function(){

alert(id);

}

element=null;

}

2.闭包的作用

2.1匿名自执行函数：函数只需执行一次，其内部变量不需要维护

2.2结果缓存

2.3实现封装、类和继承

3.闭包常见问题及解决

3.1匿名函数执行环境为全局

var name="window";

var object={

name="object";

getName:function(){

return function(){

return this.name //the window

}

}

}

var name="window";

var object={

name="object";

getName:function(){

that=this;

return function(){

return that.name

}

}

}

3.2 访问外部变量为最终值

function createArray(){

var arr = new Array();

for(var i=0;i<10;i++){

result[i]=function(){

return i;

};

return result;

}

var funcs = createArray();

for(var i=0;i<10;i++)

funcs[i] //每一个都是10

}

因为每个函数引用的都是同一个变量对象i，函数返回后都是i，要使得其符合我们的预期，应该是使用匿名函数：

function createArray(){

var arr = new Array();

for(var i=0;i<10;i++){

result[i]=function(num){

return function(){

return num;

}

}(i);

return result;

}

var funcs = createArray();

for(var i=0;i<10;i++)

funcs[i] //每一个都是10

}