闭包及js面试题

1. 当内部函数被保存到外部时，将会生成闭包。闭包会导致原有作用域链不释放，造成内存泄漏（系统空间过多被占用）

eg:function a(){  
 function b(){  
 var bbb=234;

console.log(aaa);  
 }  
 var aaa=’123’;  
 return b;

}  
var demo=a();  
demo(); //’123’,AO不释放

1. 内存泄漏：内存像泄漏出去，可用空间越来越少
2. 不互相污染变量
3. 做缓存（类似缓存）
4. 闭包的作用：
5. 实现公有变量
6. 做缓存（存储结构）
7. 可以实现封装，属性私有化
8. 模块化开发，防止污染全局变量
9. 多个函数与一个函数形成闭包时，它们公用一个AO，类似一个缓存仓库
10. 闭包是编程经常出现的问题，写着写着就容易忘，注意时刻提醒自己
11. 利用立即执行函数解决AO中变量无法实时打印问题
12. Alibaba挖坑考察闭包的面试题

<ul>

<li>a</li>

<li>a</li>

<li>a</li>

<li>a</li>

<li>a</li>

</ul>

给每个li绑定单击响应事件，返回索引

Solution:

function test(){  
 var listCollection=document.getElementsByTagName(‘li’);  
 for(var i=0;i<listCollection.length;i++)

{

(function(j){

listCollection[j].onclick=function(){  
 alert(“当前索引为：”+j+” ”);

}

}(i))

}  
 }

1. 百度面试题：

var x=1,y=z=0;

function add(n){return n+=1;}

y=add(x);

function add(n){return n+=3;}

z=add(x);

console.log(x,y,z) //1 4 4

Analyze：(1)GO: x:undefined

y:undifined

add:function(){return n+=1;}

(2)GO: x:undefined

y:undifined

add:function(){return n+=3;}

(3)执行语句时：z为暗示全局变量

GO: x:1

y:0

add:function(){return n+=3;}

z:0

(4)执行语句:y=add(x)

AO: n:1计算后将4返回并赋值给y

(5)AO被销毁，执行z=add(x)

GO: x:1

y:4

add:function(){return n+=3;}

z:4

1. 命名空间解决办法（过去解决办法）：

var org={department1:{jicheng:{name:'jicheng',age:123}},

department2:{zhangsan:{name:'zhangsan',age:34}},

};

var jicheng=org.department1.jicheng;

console.log(jicheng.name);

1. 闭包作用：变量私有化，不污染全局命名空间，实现模块化开发

Example： var init=(function(){

var name='abc';

function callName(){

console.log(name)

}

return function(){

callName(); //闭包，name私有化

} //name不污染全局命名空间

}())

init();

12、调用多个方法的小技巧（不返回this多次调用会报错无法读取undefined的drink）：

Var deng={

smoke:function(){

console.log("smoke");

return this;},

drink:function(){

console.log("drink");

return this;},

perm:function(){

console.log("perm");return this;}}

deng.smoke().drink().perm()；

13、枚举for in 循环

var obj={

name:'haha',

age:13,

sex:'male',

height:180,

\_\_proto\_\_:{

\_\_proto\_\_:Object.prototype

}

}

for(var i in obj){console.log(obj.i);} //4个undefined

Console.log(i) //打印所有属性名，obj[i]打印所有属性值

//系统内部会将obj.i ---->obj[‘i’]，然后找i属性，返回undefined

14、深度克隆：

var obj={

name:'abc',

age:123,

card:['visa','master'],

wife:{

name:'bcd',

son:{

name:'aaa'

}

}

}

function deepClone(origin,target){

var target=target||{},

toStr=Object.prototype.toString,

arrStr="[object Array]";

for(var prop in origin){

if(origin.hasOwnProperty(prop)){

if(origin[prop]!=="null"&&typeof(origin[prop])=='object'){

if(toStr.call(origin[prop])==arrStr){

target[prop]=[];}

else{target[prop]={};}

}

else{

target[prop]=origin[prop];

}

}

}

return target;

}

var obj1={};

deepClone(obj,obj1);

obj1;

15、深度克隆简单写法：

function deepClone(target){

var o = target instanceof Array ? [] : {};

for(var prop in target){

o[prop] = typeof target[prop]== 'object' ? deepClone(target[prop]) : target[prop];}

return o;

}

var c=deepClone(obj);

console.log(c)