

小小沧海

多写博客常总结，多写代码常看书

大部分人都会做错的经典JS闭包面试题

目录

- [由工作中演变而来的面试题](#)
- [JS中有几种函数](#)
- [创建函数的几种方式](#)
- [三个fun函数的关系是什么？](#)
- [函数作用域链的问题](#)
- [到底在调用哪个函数？](#)
- [后话](#)

由工作中演变而来的面试题

这是一个我工作当中的遇到的一个问题，似乎很有趣，就当做了一道题去面试，发现几乎没人能全部答对并说出原因，遂拿出来聊一聊吧。

先看题目代码：



```
function fun(n,o) {
  console.log(o)
  return {
    fun:function(m) {
      return fun(m,n);
    }
  };
}

var a = fun(0);  a.fun(1);  a.fun(2);  a.fun(3); //undefined,?,?,?
var b = fun(0).fun(1).fun(2).fun(3); //undefined,?,?,?
var c = fun(0).fun(1);  c.fun(2);  c.fun(3); //undefined,?,?,?
//问：三行a,b,c的输出分别是什么？
```



这是一道非常典型的JS闭包问题。其中嵌套了三层fun函数，搞清楚每层fun的函数是那个fun函数尤为重要。

可以先在纸上或其他地方写下你认为的结果，然后展开看看正确答案是什么？



都答对了么？如果都答对了恭喜你在js闭包问题当中几乎没什么可以难住你了；如果没有答对，继续往下分析。


JS中有几种函数

首先，在此之前需要了解的是，在JS中函数可以分为两种，**具名函数（命名函数）**和**匿名函数**。


区分这两种函数的方法非常简单，可以通过输出 `fn.name` 来判断，有name的就是具名函数，没有name的就是匿名函数

注意：在低版本IE上无法获取具名函数的name，会返回undefined，建议在火狐或是谷歌浏览器上测试

或是采用兼容IE的获取函数name方法来获取函数名称：



```
/**
 * 获取指定函数的函数名称（用于兼容IE）
 * @param {Function} fun 任意函数
 */
function getFunctionName(fun) {
    if (fun.name !== undefined)
        return fun.name;
    var ret = fun.toString();
    ret = ret.substr('function '.length);
    ret = ret.substr(0, ret.indexOf('('));
    return ret;
}
```



遂用上述函数测试是否为匿名函数：

```
> getFunctionName(fn1)
< "xxcanghai"
> getFunctionName(fn2)
< ""
```

@xxcanghai 博客园

可以得知变量fn1是具名函数，fn2是匿名函数

创建函数的几种方式

说完函数的类型，还需要了解JS中创建函数都有几种创建方法。

1、声明函数

最普通最标准的声明函数方法，包括函数名及函数体。

```
function fn1() {}
```

2、创建匿名函数表达式

创建一个变量，这个变量的内容为一个函数

```
var fn1=function () {}
```

注意采用这种方法创建的函数为**匿名函数**，即没有函数name

```
var fn1=function () {};  
getFunctionName(fn1).length;//0
```

3、创建具名函数表达式

创建一个变量，内容为一个带有名称的函数

```
var fn1=function xxcanghai() {};
```

注意：具名函数表达式的函数名只能在创建函数内部使用

即采用此种方法创建的函数在函数外层只能使用fn1不能使用xxcanghai的函数名。xxcanghai的命名只能在创建的函数内部使用

测试：



```
var fn1=function xxcanghai() {  
    console.log("in:fn1<",<typeof fn1,">xxcanghai:<",<typeof xxcanghai,">");
```

```
};  
console.log("out:fn1<",typeof fn1,">xxcanghai:<",typeof xxcanghai,">");  
fn1();  
//out:fn1< function >xxcanghai:< undefined >  
//in:fn1< function >xxcanghai:< function >
```



可以看到在函数外部（out）无法使用xxcanghai的函数名，为undefined。

注意：在对象内定义函数如var o={ fn : function () {...} }, 也属于函数表达式

4、Function构造函数

可以给 `Function` 构造函数传一个函数字符串，返回包含这个字符串命令的函数，此种方法创建的是**匿名函数**。

```
> Function("alert(1)")  
< function anonymous() {  
  alert(1)  
}  
  
> new Function("alert(1)")  
< function anonymous() {  
  alert(1)  
}  
  
>
```

@xxcanghai 博客园

5、自执行函数

```
(function(){alert(1);})();  
(function fn1(){alert(1);})();
```

自执行函数属于上述的“函数表达式”，规则相同

6、其他创建函数的方法

当然还有其他创建函数或执行函数的方法，这里不再多说，比如采用 `eval`，`setTimeout`，`setInterval` 等非常用方法，这里不做过多介绍，属于非标准方法，这里不做过多展开

三个fun函数的关系是什么？

说完函数类型与创建函数的方法后，就可以回归主题，看这道面试题。

这段代码中出现了三个fun函数，所以第一步先搞清楚，这三个fun函数的关系，哪个函数与哪个函数是相同的。



```
function fun(n,o) {  
  console.log(o)  
  return {  
    fun:function(m) {  
      //...  
    }  
  };  
}
```



先看第一个fun函数，属于标准具名函数声明，是**新创建**的函数，他的返回值是一个对象字面量表达式，属于一个新的object。

这个新的对象内部包含一个也叫fun的属性，通过上述介绍可得知，属于匿名函数表达式，即fun这个属性中存放的是一个**新创建**匿名函数表达式。

注意：所有**声明的匿名函数都是一个新函数**。

所以第一个fun函数与第二个fun函数不相同，均为新创建的函数。

函数作用域链的问题

再说第三个fun函数之前需要先说下，在函数表达式内部能不能访问存放当前函数的变量。

测试1，对象内部的函数表达式：

```
var o={  
  fn:function () {  
    console.log(fn);  
  }  
};  
o.fn(); //ERROR报错
```

```
> var o={  
  fn:function (){  
    console.log(fn);  
  }  
};  
o.fn();
```

✖ ▶ Uncaught ReferenceError: fn is not defined(...)

@xxcanghai 博客园

测试2，非对象内部的函数表达式：

```
var fn=function () {  
  console.log(fn);  
};  
fn();//function () {console.log(fn);};正确
```

```
> var fn=function (){  
  console.log(fn);  
};  
fn();  
  
function (){  
  console.log(fn);  
}
```

@xxcanghai 博客园

结论是：使用var或是非对象内部的函数表达式内，可以访问到存放当前函数的变量；在对象内部的不能访问到。

原因也非常简单，因为**函数作用域链**的问题，采用var的是在外部创建了一个fn变量，函数内部当然可以在内部寻找不到fn后向上层作用域查找fn，而在创建对象内部时，因为没有在函数作用域内创建fn，所以无法访问。

所以综上所述，可以得知，**最内层的return出去的fun函数不是第二层fun函数，是最外层的fun函数。**

所以，三个fun函数的关系也理清楚了，第一个等于第三个，他们都不等于第二个。

到底在调用哪个函数？

再看下原题，现在知道了程序中两个fun函数(第一个和第三个相同)，遂接下来的问题是搞清楚，运行时他执行的是哪个fun函数？



```
function fun(n,o) {
  console.log(o)
  return {
    fun:function(m) {
      return fun(m,n);
    }
  };
}
var a = fun(0);  a.fun(1);  a.fun(2);  a.fun(3); //undefined,?,?,?
var b = fun(0).fun(1).fun(2).fun(3); //undefined,?,?,?
var c = fun(0).fun(1);  c.fun(2);  c.fun(3); //undefined,?,?,?
//问: 三行a,b,c的输出分别是什么?
```



1、第一行a

```
var a = fun(0);  a.fun(1);  a.fun(2);  a.fun(3);
```

可以得知，第一个fun(0)是在调用**第一层fun函数**。第二个fun(1)是在调用前一个fun的返回值的fun函数，所以：

第后面几个fun(1),fun(2),fun(3),函数都是在调用**第二层fun函数**。

遂：

在第一次调用fun(0)时，o为undefined；

第二次调用fun(1)时m为1，此时fun闭包了外层函数的n，也就是第一次调用的n=0，即m=1，n=0，并在内部调用第一层fun函数fun(1,0);所以o为0；

第三次调用fun(2)时m为2，但依然是调用a.fun，所以还是闭包了第一次调用时的n，所以内部调用第一层的fun(2,0);所以o为0

第四次同理；

即：最终答案为undefined,0,0,0

2、第二行b

```
var b = fun(0).fun(1).fun(2).fun(3); //undefined,?,?,?
```

先从fun(0)开始看，肯定是调用的第一层fun函数；而他的返回值是一个对象，所以第二个fun(1)调用的是第二层fun函数，后面几个也是调用的第二层fun函数。

遂：

在第一次调用第一层fun(0)时，o为undefined；

第二次调用.fun(1)时m为1，此时fun闭包了外层函数的n，也就是第一次调用的n=0，即m=1，n=0，并在内部调用第一层fun函数fun(1,0);所以o为0；

第三次调用 `.fun(2)`时`m`为2，此时当前的`fun`函数不是第一次执行的返回对象，而是**第二次执行的返回对象**。而在第二次执行第一层`fun`函数时`(1,0)`所以`n=1,o=0`,返回时闭包了第二次的`n`，遂在第三次调用第三层`fun`函数时`m=2,n=1`，即调用第一层`fun`函数`fun(2,1)`，所以`o`为1；

第四次调用 `.fun(3)`时`m`为3，闭包了第三次调用的`n`，同理，最终调用第一层`fun`函数为`fun(3,2)`；所以`o`为2；

即最终答案： `undefined,0,1,2`

3、第三行c

```
var c = fun(0).fun(1);  c.fun(2);  c.fun(3); //undefined,?,?,?
```

根据前面两个例子，可以得知：

`fun(0)`为执行第一层`fun`函数，`.fun(1)`执行的是`fun(0)`返回的第二层`fun`函数，这里语句结束，遂`c`存放的是`fun(1)`的返回值，而不是`fun(0)`的返回值，所以`c`中闭包的也是`fun(1)`第二次执行的`n`的值。`c.fun(2)`执行的是`fun(1)`返回的第二层`fun`函数，`c.fun(3)`执行的**也是**`fun(1)`返回的第二层`fun`函数。

遂：

在第一次调用第一层`fun(0)`时，`o`为`undefined`；

第二次调用 `.fun(1)`时`m`为1，此时`fun`闭包了外层函数的`n`，也就是第一次调用的`n=0`，即`m=1, n=0`，并在内部调用第一层`fun`函数`fun(1,0)`；所以`o`为0；

第三次调用 `.fun(2)`时`m`为2，此时`fun`闭包的是第二次调用的`n=1`，即`m=2, n=1`，并在内部调用第一层`fun`函数`fun(2,1)`；所以`o`为1；

第四次`.fun(3)`时同理，但依然是调用的第二次的返回值，遂最终调用第一层`fun`函数`fun(3,1)`，所以`o`还为1

即最终答案： `undefined,0,1,1`

后话

这段代码原本是在做一个将异步回调改写为同步调用的组件时的代码，发现了这个坑，对JS的闭包有了更深入的了解。

关于什么是闭包，网上的文章数不胜数，但理解什么是闭包还是要在代码中自己去发现与领悟。

如果要我说什么是闭包，我认为，广义上的闭包就是指一个变量在他自身作用域外被使用了，就叫发生了闭包。

希望读者能通过本文对闭包现象有进一步的了解，如有其它见解或看法，欢迎指正或留言讨论。

(完)

如果您认为本文对得起您所阅读他所花的时间，欢迎点击右下角\ **推荐**。您的支持是我继续写作最大的动力，谢谢

作者：[小小沧海](#)
出处：<http://www.cnblogs.com/xxcanghai/>
本文地址：<http://www.cnblogs.com/xxcanghai/>
本文版权归作者和博客园共有，欢迎转载，但未经作者同意必须保留此段声明，且在文章页面明显位置给出原文连接，否则保留追究法律责任的权利。

分类： JavaScript

好文要顶

关注我

收藏该文

[小小沧海](#)
[关注 - 37](#)
[粉丝 - 914](#)
[+加关注](#)

135

1

« 上一篇：[Type Script在Visual Studio 2013中的问题汇总\(持续更新...\)](#)
» 下一篇：[千万不要在JS中使用连等赋值操作](#)

posted @ 2015-11-24 15:52 小小沧海 阅读(54888) 评论(113) 编辑 收藏

评论列表

#101楼 2016-10-25 15:22 [寻找进步的乐趣](#)

```
1  /*
2      a.fun(1)
3
4      一开始fun(n=0,o=undefined)!!!!!!理解初始化状态这点非常重要
5
6      a.fun(m=1)→执行后→fun(n=1,o=0)→n赋值为1,o赋值为0.
7      var a={
8          fun:执行console.log(0),返回:{
9              fun:function(1){
10                  return fun(m,1)由于没有执行所以n不会赋值为undefined,o也不会赋值为1
11              }
12          }
13      }
14  */
```

#102楼 2016-10-25 15:23 寻找进步的乐趣

```
1           a.fun(2)
2
3 一开始fun(n=0,o=undefined)!!!!!!理解初始化状态这点非常重要
4
5 a.fun(m=2)→执行后→fun(n=2,o=0)→n赋值为2,o赋值为0.
6 var a={
7     fun:执行console.log(0),返回:{
8         fun:function(2){
9             return fun(m,2) 由于没有执行所以n不会赋值为undefined,o也不会赋值为2
10        }
11    }
12 }
13 同理 a.fun(3)
```

支持(0) 反对(0)

#103楼 2016-10-25 15:23 寻找进步的乐趣

```
1  var b = fun(0).fun(1).fun(2).fun(3); //undefined,0,1,2
2
3  /*一开始fun(n=undefined,o=undefined)*/
4  /*
5     var b = fun(n=0,o=undefined); 执行后,相对应改变局部变量n和o.
6     执行console.log(undefined),返回:
7     var b={
8         fun:执行fun(n=1,o=0);console.log(0),返回:{
9             fun:执行fun(n=2,o=1);console.log(1),返回:{
10                fun:执行fun(n=3,o=2);console.log(2),返回:{
11                    fun:function(3){
12                        return fun(m,3) 由于没有执行所以n不会赋值为undefined,o也不会赋值为3
13                    }
14                }
15            }
16        }
17    }
18  */
```

支持(0) 反对(0)

#104楼 2016-11-24 21:28 舞蹈与网络

第一个是函数，第二个是返回对象的方法吧。

支持(0) 反对(0)

#105楼 2017-03-10 14:51 Kevin94

```
var fn1=function (){};
getFunctionName(fn1).length;//0
```

试了一下，这里 fn1 是有 name 的

#106楼 2017-03-23 11:35 生之须臾

@小小沧海

楼主这节创建函数的几种方式中的例子返回的length不是0呢？

```
> function getFunctionName(fun) {
  if (fun.name !== undefined)
    return fun.name;
  var ret = fun.toString();
  ret = ret.substr('function '.length);
  ret = ret.substr(0, ret.indexOf('('));
  return ret;
}
var fn1=function (){};
getFunctionName(fn1).length;//0
3
> function getFunctionName(fun) {
  if (fun.name !== undefined)
    return fun.name;
  var ret = fun.toString();
  ret = ret.substr('function '.length);
  ret = ret.substr(0, ret.indexOf('('));
  return ret;
}
var fn1=function xxcanghai(){};
getFunctionName(fn1).length;//0
9
> var fn1=function (){};
undefined
> fn1.name
"fn1"
```

#107楼[楼主] 2017-03-23 11:46 小小沧海

@ 生之须臾

恩，在最新版的chrome浏览器上已经可以支持获取匿名函数的名称了。
但是在目前为止（2017年03月23日）Safari firefox ie等其他浏览器上测试都是空。
不过能获取到名称不影响他依然是匿名函数的本质。或许以后只能通过 函数.toString在正则校验来判断了

#108楼 2017-03-23 11:52 生之须臾

@ 小小沧海

soga，谢谢回复哈，看到楼主github有个仓库，jGrid构建复杂表格 create a complex table，请问现在有这种表格：

检查项目		检查内容	操作
公示信息检查	企业即时公示信息的检查	检查企业即时公示信息是否真实、及时、全面。	<input checked="" type="radio"/> 是 <input type="radio"/> 否
	农民专业合作社年度报告公示信息的检	农民专业合作社年度报告公示信息是否真实、及时、全面。	<input checked="" type="radio"/> 是 <input type="radio"/> 否
	在中国境内从事生产经营活动的外国	检查外国（地区）企业年度报告公示信息是否真实、及时、全面。	<input checked="" type="radio"/> 是 <input type="radio"/> 否
登记事项检查	企业登记事项的核查	检查企业实际情况是否与登记事项一致。	<input checked="" type="radio"/> 是 <input type="radio"/> 否

看了下demo里面好像不能不支持这样table分大类小类呢，我找了好多表格控件都是一行一行这种的，想要一行里面还可以支持子类这种有木有呢？

#109楼[楼主] 2017-03-23 11:54 小小沧海

@ 生之须臾

我没太明白你说的大类小类 还有子类时什么意思？是说左侧那种跨越多行的单元格么？

#110楼 2017-03-23 12:00 生之须臾

@ 小小沧海
恩，是的，左侧那种跨越多行的单元格就是一个大类，想要通过循环返回的大类数组显示出表格，大类里面有个对象存放子类数据。
数据结构是树形的表格。这样说是不是好理解一点。。。

支持(0) 反对(0)

#111楼 2017-03-28 21:19 小二水

@ 小小沧

1

支持(0) 反对(0)

#112楼 2017-04-17 19:59 寒枫归尘

虽然都答对了，但是还是有点云里雾里

支持(0) 反对(0)

#113楼 2017-12-30 18:25 xianshenglu

都答对了，貌似逻辑也没问题，，，

支持(0) 反对(0)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

公告

姓名：沧海
联系我：xxcanghai@gmail.com
Github：[github.com/xxcanghai](#)
自我介绍：JS前端开发，网络工程专业，致力于推广TypeScript，正在学习NodeJs，VueJs。目前就职于美团 [工作机会](#)
昵称：小小沧海
园龄：3年3个月
粉丝：914
关注：37
[+加关注](#)

搜索

我的标签

- Vue(5)
- Vue.js(5)
- nw.js(4)

TypeScript(3)
笔记(2)
Express(2)
nodejs(2)
npm(1)
Number(1)
zepto bug(1)
更多

随笔分类(86)

BAT批处理(17)
C#Winform程序(1)
fixIE(5)
JavaScript(28)
Node.js(4)
nw.js(4)
TypeScript(6)
VirtualBox虚拟机
Vue.js(4)
WebView
电脑相关(8)
日常(3)
未解之谜(6)

随笔档案(64)

2018年1月 (1)
2017年12月 (1)
2017年7月 (3)
2017年6月 (1)
2017年5月 (1)
2017年2月 (2)
2016年12月 (1)
2016年11月 (1)
2016年10月 (1)
2016年9月 (1)
2016年8月 (2)
2016年5月 (1)
2016年4月 (1)
2016年3月 (2)
2016年2月 (4)
2016年1月 (1)
2015年12月 (3)
2015年11月 (6)
2015年9月 (3)
2015年7月 (4)
2015年6月 (23)
2015年4月 (1)

文章档案(5)

2016年6月 (1)

2016年2月 (1)
2016年1月 (1)
2015年12月 (2)

友情链接

李明月的博客
孟谦

积分与排名

积分 - 108927
排名 - 3194

最新评论

1. Re:[WinForm]平均切割图片AvgCutImage

@龙钦👍👍👍 赞...

--小小沧海

2. Re:[WinForm]平均切割图片AvgCutImage

你好！非常感谢你的开源和分享，本人之前使用ps切片工具来切图，但是后来发现像素高于30000px的图片无法导出，还好找到了你这款神器，基于原本的基础之上我又进行了一些功能扩展，添加了通过像素切图，同时.....

--龙钦

3. Re:如何在Vue2中实现组件props双向绑定

如果两个组件不在一个页面那该怎么办呢

--Daisl

4. Re:如何在Vue2中实现组件props双向绑定

@听见你说要永远时间有点久了。。我也有点记不得当初是啥问题了。。印象中就是内部组件要修改外部组件的数据，要修改的时候可以触发一个事件，外部组件收到这个事件以后自己去修改。...

--abcwt112

5. Re:如何在Vue2中实现组件props双向绑定

楼主，首先十分感谢您的技术分享。今天碰巧遇到了这个问题，想用这种双向绑定的情况，正好又符合你说的适用的三个条件。可是可是，试了你的两种方式，还是报错，不允许修改。请问有解么？求答案。。...

--听见你说要永远

阅读排行榜

1. 一道常被人轻视的前端JS面试题(85656)
2. 大部分人都会做错的经典JS闭包面试题(54888)
3. 如何在Vue2中实现组件props双向绑定(44342)
4. 2016十家公司前端面试小记(33465)
5. Gitlab的develop角色的人没有权限无法提交的问题解决方案(24637)

评论排行榜

1. 一道常被人轻视的前端JS面试题(118)
2. 2016十家公司前端面试小记(117)
3. 大部分人都会做错的经典JS闭包面试题(113)
4. Visual Studio 2015初体验——前端开发工作的问题(59)
5. 千万不要在JS中使用连等赋值操作(35)

推荐排行榜

1. 一道常被人轻视的前端JS面试题(214)

2. 2016十家公司前端面试小记(210)
3. 大部分人都会做错的经典JS闭包面试题(135)
4. 千万不要在JS中使用连等赋值操作(24)
5. 如何在Vue2中实现组件props双向绑定(23)