逻辑思维题

考虑下列代码

(function() {  
   var a = b = 5;  
})();  
  
console.log(b);

什么会被打印在控制台上？   
  
**回答**  
  
上面的代码会打印 5。  
  
这个问题的诀窍是，这里有两个变量声明，但 a 使用关键字var声明的。代表它是一个函数的局部变量。与此相反，b 变成了全局变量。  
  
这个问题的另一个诀窍是，它没有使用严格模式 ('use strict';) 。如果启用了严格模式，代码就会引发ReferenceError的错误：B没有定义（b is not defined）。请记住，严格模式，则需要明确指定，才能实现全局变量声明。比如，你应该写：

(function() {  
   'use strict';  
   var a = window.b = 5;  
})();  
  
console.log(b);

### 2、**问题2：创建“原生”（native）方法**

给字符串对象定义一个repeatify功能。当传入一个整数n时，它会返回重复n次字符串的结果。例如：

console.log('hello'.repeatify(3));

应打印 hellohellohello。   
  
  
**回答**  
  
一个可能的实现如下所示：

String.prototype.repeatify = String.prototype.repeatify || function(times) {  
   var str = '';  
   for (var i = 0; i < times; i++) {  
      str += this;  
   }  
   return str;  
};

现在的问题测试开发者有关JavaScript继承和prototype的知识点。这也验证了开发者是否知道该如何扩展内置对象（尽管这不应该做的）。   
  
这里的另一个要点是，你要知道如何不覆盖可能已经定义的功能。通过测试一下该功能定义之前并不存在：

String.prototype.repeatify = String.prototype.repeatify || function(times) {/\* code here \*/};

当你被要求做好JavaScript函数兼容时这种技术特别有用。

### **问题3：声明提升（Hoisting）**

执行这段代码，输出什么结果。

function test() {  
   console.log(a);  
   console.log(foo());  
   var a = 1;  
   function foo() {  
      return 2;  
   }  
}  
  
test();

**回答**  
  
这段代码的结果是 undefined 和 2。   
  
原因是，变量和函数的声明都被提前了（移到了函数的顶部），但变量不分配任何值。因此，在打印变量的时候，它在函数中存在（它被声明了），但它仍然是 undefined 。表示换句话说，上面的代码等同于以下内容：

function test() {  
 var a;  
 function foo() {  
 return 2;  
 }  
   
 console.log(a);  
 console.log(foo());  
   
 a = 1;  
}  
   
test();

### **问题4：this在JavaScript中如何工作的**

下面的代码会输出什么结果？给出你的答案。

var fullname = 'John Doe';  
var obj = {  
 fullname: 'Colin Ihrig',  
 prop: {  
 fullname: 'Aurelio De Rosa',  
 getFullname: function() {  
 return this.fullname;  
 }  
 }  
};  
  
console.log(obj.prop.getFullname());  
   
var test = obj.prop.getFullname;  
   
console.log(test());

**回答**  
  
答案是Aurelio De Rosa和John Doe。原因是，在一个函数中，this的行为，取决于JavaScript函数的调用方式和定义方式，而不仅仅是看它如何被定义的。   
  
在第一个 console.log()调用中，getFullname() 被调用作为obj.prop对象的函数。所以，上下文指的是后者，函数返回该对象的fullname。与此相反，当getFullname()被分配到test变量时，上下文指的是全局对象（window）。这是因为test是被隐式设置为全局对象的属性。出于这个原因，该函数返回window的fullname，即定义在第一行的那个值。

### **问题5：call() 和 apply()**

现在让你解决前一个问题，使最后的console.log() 打印 Aurelio De Rosa。   
  
**回答**  
  
该问题可以通过强制使用 call() 或者 apply() 改变函数上下文。在下面我将使用call()，但在这种情况下，apply()会输出相同的结果：

console.log(test.call(obj.prop));

### **结论：** 在这篇文章中，我们已经讨论了用来测试JavaScript开发者的五个经典问题。面试的概念和涵盖的主题通常是非常相似的。如果你不知道的一些问题的答案，不必担心：学习和经验可以慢慢积累。 如果你有其他一些有趣的问题，不要犹豫，与我们分享。它会帮助很多开发者。