妙味课堂——JS面试题

js 妙味课堂 前端基础 学习笔记

2014-11-21

这些面试题目前看起来还是有一定难度的，对我。

##选择题

题1：

```
(function(){
        return typeof arguments;
})();
```

- “object”
- “array”
- “arguments”
- “undefined”

解析：

typeof一共返回六种类型：字符串、number、布尔值、object对象、undefined、function函数，所以2、3项就剔除了

arguments是一个实参的集合，它是一个对象。

题2：

```
var f = function g(){ return 23; };
typeof g();
```

- “number”
- “undefined”
- “function”
- Error

解析：

如果第一行中的g不写的话，那么就是一个函数赋给了一个变量，这叫作“函数表达式”。这里这个函数是有名字的，因此叫作“有名的函数表达式”。多了这个名字，这种写法就是不规范的。这个f是能够找到的，但是这个g()在有些浏览器下是找不到的。因此选第四项Error。

题3：

```
(function(x){
        delete x;
        return x;
})(1);
```

- 1
- null
- undefined
- Error

解析：

delete不能删除变量，也不能删除参数。它只能删除一个对象下面的属性。因此选第1项。

题4：

```
var y = 1, x = y = typeof x;
x;
```

- 1
- “number”

- undefined
- “undefined”

解析：

是从右向左执行的，那么在执行typeof x的时候，x还没有值的。所以typeof x返回的是”undefined”（typeof返回的都是字符串类型的）。这个字符串”undefined”又赋值给了y。这时候y就不是1了。然后再赋给x，x就是字符串”undefined”。

题5：

```
(function f(f){
        return typeof f();
})(function(){ return 1; });
```

- “number”
- “undefined”
- “function”
- Error

解析：

传入的参数f是匿名函数function(){return 1;}，然后在自执行的函数里面f执行了（f()），所以这个f()就相当于1，那么typeof 1就是字符串的”number”

题6：

```
var foo = {
        bar: function(){ return this.baz; },
        baz: 1
};
(function(){
        return typeof arguments[0]();
})(foo.bar)
```

- “undefined”
- “object”
- “number”
- “function”

解析：

foo.bar可以视为函数function(){ return this.baz; }的函数名，它通过arguments[0]传到自执行函数里面。也就是说foo.bar就是arguments[0]，这个函数去执行，看起来这里的this是指向foo，其实不是。这个this要看从哪儿调用，它前面是谁。现在这个foo.bar看作一个整体，视为一个函数名，那么这个名为foo.bar的函数是在window下调用的，window下面没有baz，因此返回的应该是"undefined"。

题7：

```
var foo = {
        bar: function(){return this.baz;},
        baz: 1
}
typeof(f = foo.bar)();
```

- "undefined"
- "object"
- "number"
- "function"

解析:

这个题目跟上个题目差不多。也是将foo.bar作为一个整体赋给了f，那么f调用的时候，也还是在window下调用的。this指向是window，window下面没有baz，因此应选"undefined"。

题8：

```
var f = (function f(){return "1";},function g(){return 2;})();
typeof f;
```

- "string"
- "number"
- "function"
- "undefined"

解析:

分组选择符：小括号。

```
var a = (1, 2, 3);
alert(a); //弹出3 变量取得是最后一位
```

上面要走分组选择符后面的这个函数，后面函数return的是2，所以选择"number"

题9：

```
var x = 1;
if(function f(){}){
        x += typeof f;
}
x;
```

- 1
- "1function"
- "1undefined"
- NaN

解析：

函数声明是不能写到运算符的运算过程当中，例如if的括号或者for循环的括号中。函数写到这里面的话，函数名字就找不到了。函数声明必须是全局的或局部的，不能写到运算当中。所以typeof后面的f是找不到的，因此typeof f返回的是"undefined"。另外一个问题，这个函数写到括号当中，是返回真的。应该选第三个。

题10：

```
var x = [typeof x, typeof y][1];
typeof typeof x;
```

- "number"
- "string"
- "undefined"
- "object"

解析：

不管x是什么，typeof x返回的必然是字符串。因此typeof 字符串返回的是"string"

题11：

```
(function(foo){
        return typeof foo.bar;
```

```
})({foo: {bar: 1}});
```

- “undefined”
- “object”
- “number”
- “Error”

解析:

形参foo只的是{foo: {bar: 1}}这个整体。在typeof foo.bar;中的这个foo下面只有一个foo属性，并没有bar这个属性，因此应该选”undefined”

题12:

```
(function f(){
        function f(){return 1;}
        return f();
        function f(){return 2;}
})()
```

- 1
- 2
- Error(e.g. “Too much recursion”)
- undefined

解析:

函数声明是会预解析的，因此在return f();这一句还没有执行的时候，上面和下面的这两句就已经执行完毕了。后面把前面的覆盖了，因此在return f();这一句执行的时候，这个f已经是下面的那个函数了。因此应该选择“2”。

题13:

```
function f(){return f;}
new f() instanceof f;
```

- true
- false

解析:

instanceof方法是看前面的对象是否是后面的构造函数构造出来的。在构造函数中如果return了函数或者对象的话，那么这个函数或者对象就会把这个构造函数生成的对象覆盖掉。就相当于new f()执行完之后，就不是原本构造出来的对象了，而是f这个函数。f instanceof f返回的是false。

题14：

```
with (function(x, undefined){}) length;
```

- 1
- 2
- undefined
- Error

解析：

函数的长度就是函数的形参数量。选择第二项：2。

```
function test(num1, num2, num3){

}
alert(test.length); //弹出3
```

##运行题 作用域

1）外层的变量，内层可以找到（全局）；内层的变量，外层找不到（局部）

```
//内部的可以调到外部的
var a = 10;

function aaa(){
        alert(a);
}
aaa(); //10
```

```
//外部的调不到内部的
function aaa(){
        var a = 10;
}
```

```
aaa();
alert(a); //报错
```

```
var a = 10;

function aaa(){
        alert(a);
}
function bbb(){
        var a = 20;
        aaa();
}

bbb(); //弹出的是10
```

2) 当var不加的时候，会自动生成全局的变量（不建议这样写，最好把所有要定义的变量加上var）

```
function aaa(){
        a = 10;
}
aaa();
alert(a); //10
```

```
function aaa(){
        var a = b = 10;
}
aaa();
alert(a); //a找不到
alert(b); //弹出10
```

3）变量的查找是就近原则去寻找var定义的变量。当就近没有找到的话，就会查找外层。

```
var a = 10;

function aaa(){
        var a = 20;
        alert(a);
}
aaa(); //弹出20，因为在内层就找到a了
```

```
var a = 10;

function aaa(){
        a = 20;
        alert(a);
}
aaa(); //弹出20。在内层没有找到var定义的a，会去找到外层的a，找到外层a之后，在内层，变量a的值先又变成
```

```
var a = 10;

function aaa(){
        alert(a);
        a = 20;
}
aaa(); //弹出10
```

```
var a = 10;

function aaa(){
        alert(a);
        var a = 20;
}
aaa(); //undefined 在内层找到了var定义的a，因此就不会再找外层的var a = 10了。但是在内层找到了a，在al
```

```
//上面的代码其实相当于这段代码：

//预解析原理
var a = 10;

function aaa(){
        var a;
        alert(a);
        a = 20;
}
aaa(); //undefined
```

```
var a = 10;

function aaa(){
        bbb();
        alert(a);
        function bbb(){
                var a = 20;
        }
}
}
aaa(); //10 函数bbb里面定义的局部变量a，在alert的时候是找不到的，alert的时候找的a是外层的var a = 10
```

4）当参数跟局部变量重名的时候，优先级是等同的。

```
var a = 10;

function aaa(a){
        alert(a);
        var a = 20;
}
aaa(a); //10
```

```
var a = 5;
var b = a;
b += 3;
alert(a); //5 基本类型的赋值，只存在一个值的关系，不存在引用的关系。
```

```
var a = [1, 2, 3];
var b = a;
b.push(4);
alert(a); //[1, 2, 3, 4]; //对象之间是一种引用关系
```

```
var a = [1, 2, 3];
var b = a;
b = [1, 2, 3, 4];
alert(a); //[1, 2, 3] b是重新赋值的，与a之间的联系就断开了。a、b之间就不存在什么关系了
```

```
var a = 10;

function aaa(a){
        a += 3;
}
aaa(a);
alert(a); //10 当我们传参进来之后，就相当于是重新赋值了，a是个基本类型，它+=3是不会影响到外部的这个变
```

```
var a = [1, 2, 3];

function aaa(a){
        a.push(4);
}
aaa(a);
alert(a); //[1, 2, 3, 4] 函数aaa里面的参数a虽然与外面的a没有什么关系，但是因为这两者之间是一个引用关
```

```
var a = [1, 2, 3];

function aaa(a){
        a = [1, 2, 3, 4];
}
aaa(a);
alert(a); // [1, 2, 3] 里面的参数a是重新赋值了，而不是与外面的a存在引用关系。里面的a与外面的a没有任何
```

##字符串操作的试题

题1：写一个字符串转成驼峰的方法

例如：border-bottom-color => borderBottomColor

方法一：字符串操作

```
var str = 'border-bottom-color';

function test(str){
        var arr = str.split('-'); //[border, bottom, color]
        for(var i=1; i < arr.length; i++){
                arr[i] = arr[i].charAt(0).toUpperCase() + arr[i].substring(1); //[border, Bottom, Co
```

```
        }
        return arr.join('');
}
alert(test(str)); //borderBottomColor
```

方法二：正则操作

```
var str = 'border-bottom-color';

function test(){
        var re = /-(\w)/g;
        return str.replace(re, function($0, $1){
                return $1.toUpperCase();
        });
}

alert(test(str));
```

题2：查找字符串中出现最多的字符和个数

例如：sdjksfssscfssdd => 字符最多的是s，出现了7次

方法一：字符串操作

```
var str = sdjksfssscfssdd;

function test(str){
        var obj = {};
        var num = 0;
        var value = '';

        for(var i=0; i<str.length; i++){
                if( !obj[ str[i] ] ){
                        obj[ str[i] ] = [];
                }
                obj[ str[i] ].push( str[i] );
        }

        for (var attr in obj){
                if(num < obj[attr].length){
                        num = obj[attr].length;
```

```
                        value = obj[attr][0];
                }
        }

        return '最多的字符是：' + value + '，出现了：' + num;
}

alert(test(str));
```

方法二：正则操作

```
var str = sdjksfssscfssdd;

function test(str){
        var arr = str.split('');
        arr.sort();
        str = arr.join('');

        var re = /(\w)\1+/g;
        var num = 0;
        var value = 0;

        str.replace(re, function($0, $1){
                if(num < $0.length){
                        num = $0.length;
                        value = $1;
                }
        })

        return '最多的字符是：' + value + '，出现了：' + num;
}

alert(test(str));
```

题3：如何给字符串加千分符

例如：3562123761 => 3,562,123,761

方法一：字符串操作

```
var str = '3562123761';
```

```
function test(str){
        var iNum = str.length%3;
        var prev = '';
        var arr = [];
        var iNow = 0;
        var tmp = '';

        if(iNum != 0){
                prev = str.substring(0, iNum);
                arr.push(prev);
        }

        str = str.substring(iNum);

        for(var i=0; i<str.length; i++){
                iNow++;
                tmp += str[i];
                if(iNow==3 && tmp){
                        arr.push(tmp);
                        tmp = '';
                        iNow = 0;
                }
        }

        return arr.join(',');
}

alert(test(str));
```

方法二：正则操作

```
//正则语法：
//(?=) : 前向声明
//(?!) : 反前向声明

var str = 'abacad';
var re = /a(?=b)/g; //用前向声明，只有a后面是b的时候，这个a才被匹配到，而b不是被匹配的内容

str = str.replace(re, '*');
alert(str); //'*bacad'
```

```
var str = 'abacad';
var re = /a(?!b)/g; //用反前向声明，只有a后面不是b的时候，这个a才能被匹配到

str = str.replace(re, '*');
alert(str); //'ab*c*d'
```

```
var str = '3562123761';

function test(str){
        var re = /(?=(?!\b)(\d{3})+$)/g; //这里从后往前，匹配到三位三位的位置，而且匹配上的这个位置前

        return str.replace(re, ',');
}

alert(test(str));
```

练习：返回一个只包含数字类型的一个数组？

例如：js123ldka78sdassdfd653 -> [123, 78, 653]

##限制条件补全代码

题目一：a b 两个变量 不用第三个变量来切换两个变量值

```
/* 适用性不广
var a = 5;
var b = 6;

a = a + b;
b = a - b;
a = a - b;

alert(a); //6
alert(b); //5
*/

var a = 'hello';
var b = 'hi';

a = [a, b];
b = a[0];
```

```
a = a[1];

alert(a); //'hi'
alert(b); //'hello'
```

题目二：有一个数n=5，不用for循环，怎么返回[1, 2, 3, 4, 5]这样一个数组

方法一

```
var n = 5;

function show(n){
        //用什么可以替代循环呢？递归、定时器。但是定时器用不了，因为它是异步的。考虑用递归。

        var arr = [];

        return (function(){
                arr.unshift(n);
                n--;
                if(n!=0){
                        arguments.callee();
                }
                return arr;
        })();
}

alert(show(n)); //[1, 2, 3, 4, 5]
```

方法二

```
var n = 5;

function show(n){

        var arr = [];

        arr.length = n + 1; //数组arr变成 [, , , , ,]
        var str = arr.join('a'); //str -> 'aaaaa'
        var arr2 = [];
        str.replace(/a/g, function(){
                arr2.unshift(n--);
        })
```

```
        return arr2;
}

alert(show(n)); //[1, 2, 3, 4, 5]
```

题目三：一个数n，当n小于100就返回n，否则就返回100

不限定条件

```
var n = 50;

function show(){
        if(n<100){
                return n;
        } else {
                return 100;
        }
}

alert(show(n)); //50
```

限定条件：不允许用**if else**

```
var n = 50;

function show(){
        return n < 100 ? n : 100;
}

alert(show(n)); //50
```

限定条件：不允许用**if else**，也不允许用三目

```
var n = 50;

function show(){
        switch(n<100){
                case true:
                        return n;
                break;
```

```
                case false:
                        return 100;
                break
        }
}

alert(show(n)); //50
```

*限定条件：不允许用if else，三目和switch

```
var n = 50;

function show(){
        return Math.min(n, 100);
}

alert(show(n)); //50
```

*限定条件：不允许用if else、三目、switch和Math.min()

```
var n = 50;

function show(){
        var arr = [n, 100];
        arr.sort(function(n1, n2){
                return n1 - n2;
        })
        return arr[0];
}

alert(show(n)); //50
```

*限定条件：不允许用if else、三目、switch、Math.min()、数组

```
var n = 50;

function show(){
        var m = '' + n; //n=150的话，m.length是3；n=50的话，m.length是2

        for(var i=2; i<m.length && n>0; i++){
```

```
                return 100;
        }
        return n;
}

alert(show(n)); //50
```

限定条件：不允许用**if else**、三目、**switch**、**Math.min()**、数组、循环（包括**for**、**while**、**do while**）

```
var n = 50;

function show(){
        var json = {name: 'hello'};
        var m = n < 100 || json;
        for(var attr in m){
                return 100;
        }
        return n;
}

alert(show(n)); //50
```

限定条件：不允许用**if else**、三目、**switch**、**Math.min()**、数组、循环（包括**for**、**while**、**do while**）、**{} for in**

```
var n = 50;

function show(){
        var m = n >= 100 && 100; //n为150，前面为真，m就为100；n为50，m就是false
        return m = m || n; //如果n为150，m就是100，最终返回的就是100；如果n为50，m得false，返回时m就
}

alert(show(n)); //50
```

##算法题

题1：斐波那契数列：1、1、2、3、5、8、13、21

方法一

```
//递归的写法
function aaa(n){

        if(n <= 2) {
                return 1;
        }

        return aaa(n-1) + aaa(n-2);

}

alert(aaa(8)); //21 返回斐波那契数列第8项的值
```

方法二

```
//通过迭代的方式
//迭代就是开循环，三个数迭代一次；8个数迭代6次

function aaa(n){

        var num1 = 1;
        var num2 = 1;
        var num3 = 0;

        for(var i=0; i <n-2; i++){
                num3 = num1 + num2;
                num1 = num2;
                num2 = num3;
        }

        return num3;

}

alert(aaa(8));
```

题2：数组排序

冒泡排序

```
//冒泡排序是两个两个进行比较
function aaa(arr){

	for(var i=0; i<arr.length; i++){
		for(var j=0; j<arr.length-i; j++){
			toCon(j, j+1);
		}
	}

	function toCon(prev, next){
		var tmp = '';
		if(arr[prev] > arr[next]){
			tmp = arr[prev];
			arr[prev] = arr[next];
			arr[next] = tmp;
		}
	}

	return arr;

}

alert(aaa([4, 5, 1, 7, 2])); //[1, 2, 4, 5, 7]
```

简单选择排序

```
简单选择排序：找到最小值扔到数组一开始，从剩余的找到最小值，排到刚才那个的前面
function aaa(arr){

	if(arr.length == 1){
		return arr;
	}

	var iMin = arr[0];
	var iIndex = 0;

	for(var i=0; i<arr.length; i++){
		if( arr[i] < iMin ){
			iMin = arr[i];
			iIndex = i;
		}
	}
```

```
        var prev = arr.splice(iIndex, 1); //把1剪切出来成了一个数组[1]，arr现在就变成了[4,5,7,2] 后面

        return prev.concat(aaa(arr));

}

alert(aaa([4, 5, 1, 7, 2]));
```

题3：数组去重

方法一

```
//找到一个不重复的，就扔到新数组里面，剩余的里面再找不重复的，再扔到新数组里面，以此类推，最后返回新
function aaa(){

        var result = [ arr[0] ];
        for(var i=1; i<arr.length; i++){
                if( toCon( arr[i] ) ){ //如果没有重复的
                        result.push( arr[i] );
                }
        }

        function toCon(val){
                for(var i=0; i<result.length; i++){
                        if( result[i] == val){
                                return false;
                        }
                }
                return true;
        }

        return result;
}
alert(aaa([5, 2, 7, 5, 1, 7, 5, 4])); //[5, 2, 7, 1, 4]
```

方法二

```
//利用json的key值的唯一性
function aaa(){
```

```
        var result = [];
        var obj = {};

        for(var i=0; i<arr.length; i++){
                if(!obj[arr[i]]){
                        result.push(arr[i]);
                        obj[arr[i]] = 1;
                }
        }

        return result;
}
alert(aaa([5, 2, 7, 5, 1, 7, 5, 4]));
```

妙味课堂——正则表达式

妙味课堂——问题解答视频