**深入剖析javaScript中的深拷贝和浅拷贝**

在面试时经常会碰到面试官问:什么是深拷贝和浅拷贝，请举例说明？如何区分深拷贝与浅拷贝，简单来说，假设B复制了A，当修改A时，看B是否会发生变化，如果B也跟着变了，说明这是浅拷贝，如果B没变，那就是深拷贝；我们先看两个简单的案例：

//案例1  
var a1 = 1, a2= a1;

console.log(a1) //1

console.log(a2) //1

a2 = 2; //修改 a2

console.log(a1) //1

console.log(a2) //2  
//案例2  
var o1 = {x: 1, y: 2}, o2 = o1;

console.log(o1) //{x: 1, y: 2}

console.log(o2) //{x: 1, y: 2}

o2.x = 2; //修改o2.x

console.log(o1) //{x: 2, y: 2}

console.log(o2) //{x: 2, y: 2}

按照常规思维，o1应该和a1一样，不会因为另外一个值的改变而改变，而这里的o1 却随着o2的改变而改变了。同样是变量，为什么表现不一样呢？为了更好的理解js的深浅拷贝，我们先来理解一些js基本的概念 —— 目前JavaScript有五种基本数据类型（也就是简单数据类型），它们分别是：Undefined，Null，Boolean，Number和String。还含有一种复杂的数据类型（也叫引用类型），就是对象，引用类型有：Object、Array、Function(之所以说“目前”，因为之后也可能会有新的类型出来。)

**一、基本类型和引用类型**

ECMAScript变量可能包含两种不同数据类型的值：基本类型值和引用类型值。基本类型值指的是那些保存在栈内存中的简单数据段，即这种值完全保存在内存中的一个位置。而引用类型值是指那些保存堆内存中的对象，意思是变量中保存的实际上只是一个指针，这个指针指向内存中的另一个位置，该位置保存对象。

打个比方，基本类型和引用类型在赋值上的区别可以按“双胞胎”和“影子”来理解：基本类型赋值等于一位妈妈生的双胞胎，二者互不相关，各自有各自的特性；而引用类型赋值相当于自己跟影子，你在变化时影子会随着变化；

上面清晰明了的介绍了基本类型和引用类型的定义和区别。

再回到前面的案例，案例1中的值为基本类型，案例2中的值为引用类型。案例2中的赋值就是典型的浅拷贝，并且深拷贝与浅拷贝的概念只存在于引用类型。

**二、深拷贝与浅拷贝**

既然已经知道了深拷贝与浅拷贝的来由，那么该如何实现深拷贝？我们先分别看看Array和Object自有方法是否支持：

1、Array

对于数组我们可以使用slice() 和 concat() 方法来解决上面的问题

1. slice

var arr1 = ['a', 'b'], arr2 = arr1.slice();

console.log(arr1); // ["a", "b"]

console.log(arr2); // ["a", "b"]

arr2[0] = 'c'; //修改arr2

console.log(arr1); // ["a", "b"]

console.log(arr2); // ["c", "b"]

此时，arr2的修改并没有影响到arr1，看来深拷贝的实现并没有那么难嘛。

2) concat

var arr1 = ['a', 'b'], arr2 = arr1. concat ();

console.log(arr1); // ["a", "b"]

console.log(arr2); // ["a", "b"]

arr2[0] = 'c'; //修改arr2

console.log(arr1); // ["a", "b"]

console.log(arr2); // ["c", "b"]

我们把arr1改成二维数组再来看看：

var arr1 = ['a', 'b', ['c', 'd']], arr2 = arr1.concat();

arr2[2][1] = 100;

console.log(arr1); //['a', 'b', ['c', 100]]

console.log(arr2); //['a', 'b', ['c', 100]]

咦，arr2又改变了arr1，看来slice()/concat()只能实现一维数组的深拷贝

除了上面两种方法外，我们还可以借用JQ的extend方法。

$.extend( [deep ], target, object1 [, objectN ] )

deep表示是否深拷贝，为true为深拷贝，为false，则为浅拷贝

target Object类型 目标对象，其他对象的成员属性将被附加到该对象上。

object1  objectN可选。 Object类型 第一个以及第N个被合并的对象。

var a=[0,1,[2,3],4],

b=$.extend(true,[],a);

a[0]=1;

a[2][0]=1; //

console.log(a); //[1,1,[1,3],4]

console.log(b); //[0,1,[2,3],4]

不过这种方法需要依赖JQ库。

2、Object

1) 利用对象的深拷贝实现原理

定义一个新的对象，遍历源对象的属性 并 赋给新对象的属性

var obj = {

name:'sonia',

age: 18

}

var obj2 = new Object();

obj2.name = obj.name;

obj2.age = obj.age

obj.name = 'alice';

console.log(obj); //Object {name: "'alice'", age: 18}

console.log(obj2); //Object {name: "'sonia'", age: 18}

理解了以上的基本思想，我们就可以封装一个方法 deepCopy来实现对象的深拷贝，代码如下

var obj = {

name: 'sonia',

age: 18

}

var deepCopy = function (source) {

var result = {};

for(var key in source) {

if(typeof source[key] === 'object') {

result[key] = deepCopy(source[key])

} else {

result[key] = source[key]

}

}

return result;

}

var objCopy = deepCopy(obj)

obj.name = 'aaa';

console.log(obj);//Object {name: "aaa", age: 18}

console.log(objCopy);//Object {name: "sonia", age: 18}

1. JSON.parse(JSON.stringify(obj))

var obj = {

name: 'sonia',

age: 18

}

var obj2 = JSON.parse(JSON.stringify(obj));

obj.name = 'alice';

console.log(obj) // {name: "alice", age: 18}

console.log(obj2) // {name: "sonia", age: 18}

了解拷贝也不仅仅是为了应付面试，在实际开发中也是非常有用的。