首先，我们将JavaScript中的数据类型分为两大类：原始类型和对象类型。（1）原始类型包括：数值、字符串、布尔值、null、undefined（我们需要克隆的主要是前面三个）（2）对象类型包括：对象（Object），函数（Function）、数组（Array）。在克隆过程中对这两类数据类型的处理方式是不一样的，下面先分别讲解两种数据类型的克隆方式。

**一、原始类型克隆**

**1、数值的克隆**

var x=1;

var y=x;

y=2;

console.log(x); //1

console.log(y); //2

**2、字符串的克隆**

var x="abc";

var y=x;

y="def";

console.log(x); //abc

console.log(y); //def

* 5

**3、布尔值的克隆**

var x = true;

var y = x;

y=false;

console.log(x); //true

console.log(y); //false

由于原始类型存储的是对象的实际数据，因此我们可以通过简单的复制方式即可得到正确的结果，而且不影响之前的对象。

### ****二、对象类型克隆****

**1、数组的克隆**

var x = [1,2,3];

var y = x;

console.log(y); //[1,2,3]

y.push(4);

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

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

对象类型存储的是对象的引用地址，而把对象的实际内容单独存放，因为对象类型通常比较庞大，这是数据开销和内存开销优化的手段。因此我们不能像原始数据类型一样简单的赋值，而应该遍历数据中的每一个元素，将其中的每一个原始数据类型复制过去，做法如下：

var x = [1,2,3];

var y = [];

for (var i = 0; i < x.length; i++) {

y[i]=x[i];

}

console.log(y); //[1,2,3]

y.push(4);

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

console.log(x); //[1,2,3]

**2、对象的克隆**

参照数组的克隆，我们采用同样的思想进行对象的克隆：

var x = {a:1,b:2};

var y = {};

for(var i in x){

y[i] = x[i];

}

console.log(y); //Object {a: 1, b: 2}

y.c = 3;

console.log(y); //Object {a: 1, b: 2, c: 3}

console.log(x); //Object {a: 1, b: 2}

**3、函数的克隆**

var x = function(){console.log(1);};

var y = x;

y = function(){console.log(2);};

x(); //1

y(); //2

由于函数对象克隆之后的对象会单独复制一次并存储实际数据，因此并不会影响克隆之前的对象。所以采用简单的复制“=”即可完成克隆。

### ****三、通用的对象克隆****

通过上面的分析，我们知道对于原始类型以及对象类型中的函数是可以直接通过“=”复制来实现克隆，而对于对象和数组，则需要遍历每一个元素，如果元素为对象或数组，则需继续遍历，直到为原始类型或函数，则直接通过“=”复制。

function deepClone(obj){

var result;

var oClass=isClass(obj);

if(oClass==="Object"){

result={};

}else if(oClass==="Array"){

result=[];

}else{

return obj;

}

for(var key in obj){

var copy=obj[key];

if(isClass(copy)=="Object"){

result[key]=arguments.callee(copy);//递归调用

}else if(isClass(copy)=="Array"){

result[key]=arguments.callee(copy);

}else{

result[key]=obj[key];

}

}

return result;

}

//判断对象的数据类型

function isClass(o){

if(o===null) return "Null";

if(o===undefined) return "Undefined";

return Object.prototype.toString.call(o).slice(8,-1);

}

我们设置一个数组对象测试一下：

var o=[1,2,[3,4],{a:1}];

var o2 = deepClone(o);

o2[2].push(5);

o2[3].b=2;

