基本数据类型总是通过值复制的方式来赋值/传递

引用数据类型总是通过引用复制来完成赋值/传递

注意引用指向的是值而非变量，所以一个引用无法更改另一个引用的指向，

但是可以更改不同变量共同指向的值！

var a=2;

var b=a;

b++;

console.log(a);//2

console.log(b);//3

var arr1 = [1,2,3];

var arr2=arr1;

console.log(arr1);// [1,2,3]

console.log(arr2);//[1,2,3]

arr2.push(4);

console.log(arr1);//[1,2,3,4]

console.log(arr2);//[1,2,3,4]

函数里的数组被重新赋值了，传进去的数出来只能是arr，下面那个是同名不同地址的函数

function foo(arr){

arr.push(4);

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

arr=["a","b","c"];

console.log(arr);//["a","b","c"]//重新赋值了 那他就是他自己的数组，和传进来的数组无关

}

var arr=[1,2,3];

foo(arr);

数组的地址没有换，换了内容，地址不变，出来的也是这个经过处理的

function foo(arr){

arr.push(4);

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

//清空数组，只修改里面的值 不是修改地址重新创， 所以输出还是这个地址

arr.length=0;

arr.push("a");

arr.push("b");

arr.push("c");

console.log(arr);//[a,b,c]

}

var arr=[1,2,3];

foo(arr);

console.log(arr);//[a,b,c]

在外复制了一个数组，然后传进去的是复制的这个数组

function foo(arr){2

arr.push(4);

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

//清空数组

arr.length=0;

arr.push("a");

arr.push("b");

arr.push("c");

console.log(arr);//[a,b,c]

}

var arr=[1,2,3];

foo(arr.slice());//复制一个新的数组和arr一模一样，但是不是原来的数组，相当于

//var arr[1,2,3],arr1=arr.slice(),所以穿的是arr1

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