简言：在有指针的情况下，java通过new产生的对象，js新创建的对象等都带有指针，（出去基本的数据类型：number，string，Boolean，null，undefined其他的都属于引用类型，带有指针）

浅拷贝就是增加了一个指向已经存在的内存的指针，当其中一个指针的数据改变的时候也会影响到其他指向该内存的指针变量的值，深拷贝是增加一个指针的时候新增一个新的内存，是这个指针指向这个新的内存，采用深拷贝的情况下，释放内存的时候就不会出现在浅拷贝时重复释放同一内存的错误。

1. java的深拷贝与浅拷贝

**浅拷贝**（Shallow Copy）：①对于数据类型是基本数据类型的成员变量，浅拷贝会直接进行值传递，也就是将该属性值复制一份给新的对象。因为是两份不同的数据，所以对其中一个对象的该成员变量值进行修改，不会影响另一个对象拷贝得到的数据。②对于数据类型是引用数据类型的成员变量，比如说成员变量是某个数组、某个类的对象等，那么浅拷贝会进行引用传递，也就是只是将该成员变量的引用值（内存地址）复制一份给新的对象。因为实际上两个对象的该成员变量都指向同一个实例。在这种情况下，在一个对象中修改该成员变量会影响到另一个对象的该成员变量值。

**一般的通过构造函数进行的赋值属于浅拷贝**

**深拷贝**，new两个对象出来后，通过遍历或者重写clone方法将一个对象的各个属性都复制给另外一个元素

1. js的深拷贝与浅拷贝

js的拷贝方式：一般的引用类型的=赋值都属于浅拷贝，或者使用es6中的object.assign也属于浅拷贝

如何实现深拷贝:当一个引用里只有一层没有更深层的引用时，我们可以使用jq的slice（）方法实现，

另外我们可以自定义一个深拷贝的递归方法来实现：

function deepClone(obj){

let objClone = Array.isArray(obj)?[]:{};

if(obj && typeof obj==="object"){

for(key in obj){

if(obj.hasOwnProperty(key)){

//判断ojb子元素是否为对象，如果是，递归复制

if(obj[key]&&typeof obj[key] ==="object"){

objClone[key] = deepClone(obj[key]);

}else{

//如果不是，简单复制

objClone[key] = obj[key];

}

}

}

}

return objClone;

}

除了递归我们也可以使用JSON对象的parse和stringify

function deepClone(obj){

let \_obj = JSON.stringify(obj),

objClone = JSON.parse(\_obj);

return objClone

} ;

**3.除了上面两种方法之外，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。