**浅拷贝**

浅拷贝的意思就是只复制引用（指针），而未复制真正的值。

浅拷贝和深拷贝都是对于JS中的引用类型而言的，浅拷贝就只是复制对象的引用（堆和栈的关系，简单类型Undefined，Null，Boolean，Number和String是存入堆，直接引用，object array 则是存入桟中，只用一个指针来引用值），如果拷贝后的对象发生变化，原对象也会发生变化。只有深拷贝才是真正地对对象的拷贝。

const originObj = {a:'a',b:'b',c:[1,2,3],d:{dd:'dd'}};

const cloneObj = originObj;

cloneObj.a = {aa:'aa'};

console.log(originObj); // {a:{aa:'aa'},b:'b',c:Array[3],d:{dd:'dd'}}

注：改变了原先的值

**深拷贝**

深拷贝就是对目标的完全拷贝，不像浅拷贝那样只是复制了一层引用，就连值也都复制了。

目前实现深拷贝的方法不多，主要是两种：

**利用 JSON 对象中的 parse 和stringify**

const originObj = {a:'a',b:'b',c:[1,2,3],d:{dd:'dd'}};

const cloneObj = JSON.parse(JSON.stringify(originObj));

console.log(cloneObj === originObj); // false

cloneObj.a = 'aa';

cloneObj.c = [1,1,1];

cloneObj.d.dd = 'doubled';

console.log(cloneObj); // {a:'aa',b:'b',c:[1,1,1],d:{dd:'doubled'}};

console.log(originObj); // {a:'a',b:'b',c:[1,2,3],d:{dd:'dd'}};

但是

const originObj = {

 name:'axuebin',

sayHello:function(){

console.log('Hello World');

  }

}

就是说如果对象中含有一个函数时（很常见），就不能用这个方法进行深拷贝undefined、function、symbol 会在转换过程中被忽略

**递归的方法（推荐）**

递归的思想就很简单了，就是对每一层的数据都实现一次 创建对象->对象赋值 的操作，

function deepClone(source){

  const targetObj = source.constructor === Array ? [] : {}; // 判断复制的目标是数组还是对象

 for(let keys in source){ // 遍历目标

 if(source.hasOwnProperty(keys)){

  if(source[keys] && typeofsource[keys] === 'object'){ // 如果值是对象，就递归一下

 targetObj[keys] = source[keys].constructor === Array ? [] : {};

  targetObj[keys] = deepClone(source[keys]);

 }else{ // 如果不是，就直接赋值

 targetObj[keys] = source[keys];

  } }  }

  return targetObj;

}

**首层浅拷贝**

我们知道了，会有一种情况，就是对目标对象的第一层进行深拷贝，然后后面的是浅拷贝，可以称作“首层浅拷贝”。

JavaScript 中，数组有两个方法 concat 和 slice 是可以实现对原数组的拷贝的，这两个方法都不会修改原数组，而是返回一个修改后的新数组。