Senparc 盛派®

密级：公开

浅拷贝与深拷贝

2020.08.12 · 苏州盛派网络科技有限公司

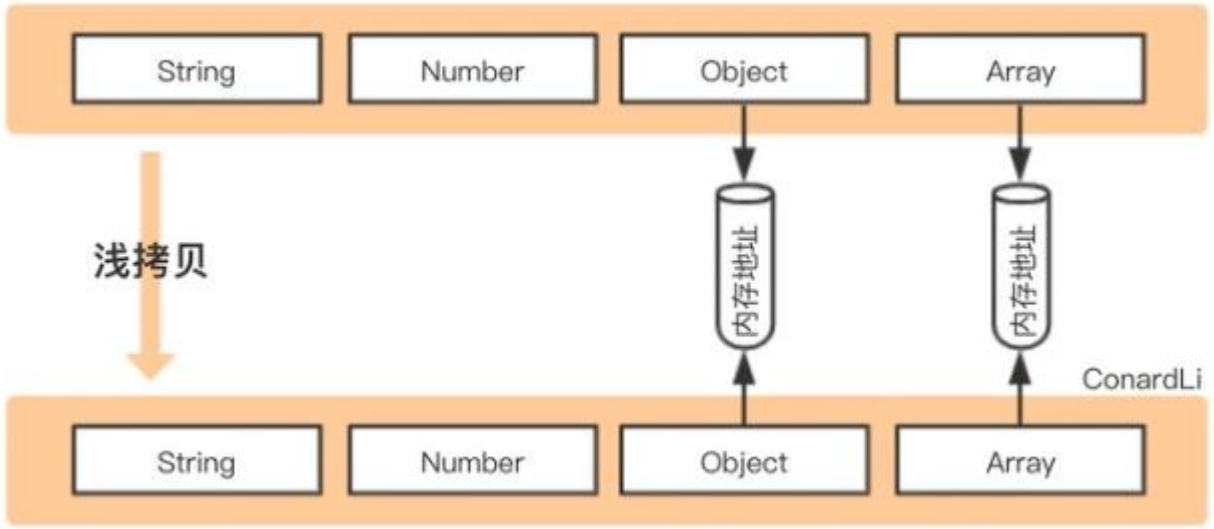
主持人/分享人：Jam

1. 什么是深/浅拷贝，他们跟赋值有何区别？

2. 深/浅拷贝的实现方式有几种？

浅拷贝

浅拷贝是创建一个新对象，这个对象有着原始对象属性值的一份精确拷贝。如果属性是基本类型，拷贝的就是基本类型的值，如果属性是引用类型，拷贝的就是内存地址 ，所以**如果其中一个对象改变了这个地址，就会影响到另一个对象**。

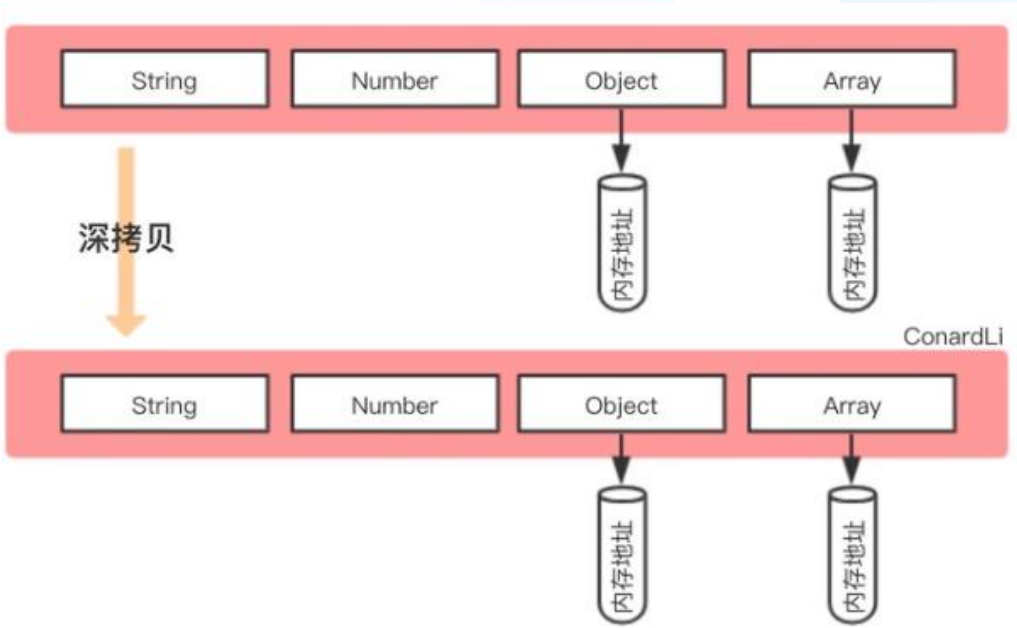


深拷贝

Senparc 盛派®

深拷贝是将一个对象从内存中完整的拷贝一份出来 从堆内存中开辟一个新的区域存放新对象

且**修改新对象不会影响原对象**。



总而言之，浅拷贝只复制指向某个对象的指针，而不复制对象本身，**新旧对象还是共享同一块内存**。但深拷贝会另外创造一个一模一样的对象，**新对象跟原对象不共享内存**，修改新对象不会改到原对象。

赋值和深/浅拷贝的区别

Senparc 盛派®

- 当我们把一个对象赋值给一个新的变量时，**赋的其实是该对象的在栈中的地址，而不是堆中的数据**。也就是两个对象指向的是同一个存储空间，无论哪个对象发生改变，其实都是改变的存储空间的内容，因此，两个对象是联动的。

- 浅拷贝：重新在堆中创建内存，拷贝前后对象的基本数据类型互不影响，但拷贝前后对象的引用类型因共享同一块内存，会相互影响。

- 深拷贝：从堆内存中开辟一个新的区域存放新对象，对对象中的子对象进行递归拷贝,拷贝前后的两个对象互不影响。

浅拷贝的实现方式

Senparc 盛派®

1. Object.assign()

   函数库　　的　　方法

   展开运算符

Object.assign()

Senparc 盛派®

方法可以把任意多个的源对象自身的可枚举属性拷贝给目标对象，然后返回目标对象。

javascript 复制代码

```javascript
let obj1 = { person: {name: "kobe", age: 41},sports:'basketball' };
let obj2 = Object.assign({}, obj1);
obj2.person.name = "wade";
obj2.sports = 'football'
console.log(obj1); // { person: { name: 'wade', age: 41 }, sports: 'basketball' }
```

函数库lodash的_.clone方法

Senparc 盛派®

该函数库也有提供_.clone用来做 Shallow Copy,后面我们会再介绍利用这个库实现深拷贝。

javascript 复制代码

```javascript
var _ = require('lodash');
var obj1 = {
    a: 1,
    b: { f: { g: 1 } },
    c: [1, 2, 3]
};
var obj2 = _.clone(obj1);
console.log(obj1.b.f === obj2.b.f);// true
```

展开运算符...

Senparc 盛派®

展开运算符是一个 es6 / es2015特性，它提供了一种非常方便的方式来执行浅拷贝，这与Object.assign ()的功能相同。

javascript 复制代码

```javascript
let obj1 = { name: 'Kobe', address:{x:100,y:100}}
let obj2= {... obj1}
obj1.address.x = 200;
obj1.name = 'wade'
console.log('obj2',obj2) // obj2 { name: 'Kobe', address: { x: 200, y: 100 } }
```

Array.prototype.concat()

Senparc 盛派®

javascript 复制代码

```javascript
let arr = [1, 3, {
    username: 'kobe'
    }];
let arr2 = arr.concat();
arr2[2].username = 'wade';
console.log(arr); //[ 1, 3, { username: 'wade' } ]
```

Array.prototype.slice()

Senparc 盛派®

```javascript
let arr = [1, 3, {
    username: ' kobe'
    }];
let arr3 = arr.slice();
arr3[2].username = 'wade'
console.log(arr); // [ 1, 3, { username: 'wade' } ]
```

深拷贝的实现方式

Senparc 盛派®

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

   手写递归方法

   函数库　　的　　方法

4. 其他

JSON.parse方法

Senparc 盛派®

Object.assign() 方法可以把任意多个的源对象自身的可枚举属性拷贝给目标对象，然后返回目标对象。

```
1 const newObj = JSON.parse(JSON.stringify(oldObj));
```

缺点：

- 他无法实现对函数 、RegExp等特殊对象的克隆
- 会抛弃对象的constructor,所有的构造函数会指向Object
- 对象有循环引用,会报错

递归实现

Senparc 盛派®

递归方法实现深度克隆原理：遍历对象、数组直到里边都是基本数据类型，然后再去复制，就是深度拷贝。

javascript 复制代码

```javascript
function deepClone(obj, hash = new WeakMap()) {
  if (obj === null) return obj; // 如果是null或者undefined我就不进行拷贝操作
  if (obj instanceof Date) return new Date(obj);
  if (obj instanceof RegExp) return new RegExp(obj);
  // 可能是对象或者普通的值  如果是函数的话是不需要深拷贝
  if (typeof obj !== "object") return obj;
  // 是对象的话就要进行深拷贝
  if (hash.get(obj)) return hash.get(obj);
  let cloneObj = new obj.constructor();
  // 找到的是所属类原型上的constructor,而原型上的 constructor指向的是当前类本身
  hash.set(obj, cloneObj);
  for (let key in obj) {
    if (obj.hasOwnProperty(key)) {
      // 实现一个递归拷贝
      cloneObj[key] = deepClone(obj[key], hash);
    }
  }
  return cloneObj;
}
let obj = { name: 1, address: { x: 100 } };
obj.o = obj; // 对象存在循环引用的情况
let d = deepClone(obj);
obj.address.x = 200;
console.log(d);
```

函数库lodash的_.cloneDeep方法

Senparc 盛派®

该函数库也有提供_.cloneDeep用来做 Deep Copy。

javascript 复制代码

```javascript
var _ = require('lodash');
var obj1 = {
    a: 1,
    b: { f: { g: 1 } },
    c: [1, 2, 3]
};
var obj2 = _.cloneDeep(obj1);
console.log(obj1.b.f === obj2.b.f);// false
```

• 总结

Senparc 盛派®

谢谢!

Jam
E-mail: jdeng@senparc.com