**非构造函数的继承--把对象拷贝了**

[**开发部知识管理空间**](http://wiki.cregcloud.com/document/index?document_id=2) **/**[**WEB组**](http://wiki.cregcloud.com/document/index?document_id=6) **/**[**前端**](http://wiki.cregcloud.com/document/index?document_id=11)

 被 [xiaozongjie（肖宗杰）](http://wiki.cregcloud.com/system/main/index" \t "_parent) 创建于 2019/04/03 13:44:58 ， 被 [xiaozongjie（肖宗杰）](http://wiki.cregcloud.com/system/main/index" \t "_parent)最后更新于 2019/04/03 15:02:50 (查看修改历史)

[编辑](http://wiki.cregcloud.com/page/edit?document_id=58) 收藏[分享](http://wiki.cregcloud.com/page/display?document_id=58" \t "_blank) 导出

“js的继承机制”一节中我们知道继承一个对象用“构造函数+new”的模式就可以了，那么有没有其他的方法了呢，毕竟原型链查找也是需要消耗一些性能的，我们有时并不需要原型链上的所有属性和方法。  
简单说，如果把父构造函数对象的原型上所有属性和方法，拷贝进子构造函数对象的原型对象上，不也能够实现简单的继承吗？  
代码如下

1. function extend2(Child, Parent) {
2. let p = Parent.prototype;
3. let c = Child.prototype;
4. for (let i in p) {
5. c[i] = p[i];
6. }
7. c.uber = p;
8. } // 小提示，for+in 方式的循环将遍历对象本身的所有可枚举属性，以及对象从其构造函数原型中继承的属性（更接近原型链中对象的属性覆盖原型属性）。

这个函数的作用，就是将Parent对象的prototype对象中的属性，一一拷贝给Child对象的prototype对象。  
使用方法：

1. function Animal(){}
2. Animal.prototype.species = "动物";
3. function Cat(){}
4. extend2(Cat, Animal);
5. let cat1 = new Cat("大毛","黄色");
6. alert(cat1.species); // 动物

不过这只是父构造函数原型对象上的属性和方法，话句话说，我们只拷贝了原型链上的一层，实际中可能够用也可能不够用。那么通过拷贝来实现继承能不能完全替代原型链绑定呢。这个方法还是可以的。而且由于是拷贝得到的结果，所以跟原来的原型链已经是两个了，原型链互相之间的改变不会影响到对方。  
由于有些属性的值可以是数组和对象，这就造成浅拷贝的结果是属性只取到了对应值的引用而不是具体的值，所以我们最终的目的是进行深拷贝。  
下面分别给出代码：  
1.浅拷贝

1. function extendCopy(p) {
2. let c = {};
3. for (let i in p) {
4. c[i] = p[i];
5. }
6. c.uber = p;
7. return c;
8. }

使用的时候，这样写：

1. let Chinese = {
2. nation:'中国'
3. };
4. let Doctor ={
5. career:'医生'
6. }
7. let Doctor = extendCopy(Chinese);
8. Doctor.career = '医生';
9. alert(Doctor.nation); // 中国

当父对象的属性等于数组或另一个对象，那么，子对象获得的只是一个内存地址，而不是真正拷贝，因此存在父对象被篡改的可能。现在给Chinese添加一个”出生地”属性，它的值是一个数组：  
Chinese.birthPlaces = [‘北京’,’上海’,’香港’];  
通过extendCopy()函数，让Doctor继承了Chinese。  
let Doctor = extendCopy(Chinese);  
然后，我们为Doctor的”出生地”添加一个城市：  
Doctor.birthPlaces.push(‘厦门’);  
我们测试：  
alert(Doctor.birthPlaces); //北京, 上海, 香港, 厦门  
alert(Chinese.birthPlaces); //北京, 上海, 香港, 厦门  
所以，extendCopy()只是拷贝基本类型的数据，对引用类型的数据还是用的同一内存  
2.深拷贝  
所谓”深拷贝”，就是能够实现真正意义上的数组和对象的拷贝。它的实现并不难，只要递归调用”浅拷贝”就行了。

1. function deepCopy(p, c) {
2. let c = c || {};
3. for (let i in p) {
4. if (typeof p[i] === 'object') {
5. c[i] = (p[i].constructor === Array) ? [] : {};
6. deepCopy(p[i], c[i]);
7. } else {
8. c[i] = p[i];
9. }
10. }
11. return c;
12. }

使用方式：

1. let Doctor = deepCopy(Chinese);

我们测试：给父对象加一个属性，值为数组。然后，在子对象上修改这个属性：

1. Chinese.birthPlaces = ['北京','上海','香港'];
2. Doctor.birthPlaces.push('厦门');

这时，父对象就不会受到影响了。

1. alert(Doctor.birthPlaces); //北京, 上海, 香港, 厦门
2. alert(Chinese.birthPlaces); //北京, 上海, 香港

**引用出处：**

<http://www.ruanyifeng.com/blog/2010/05/object-oriented_javascript_inheritance.html>  
<http://www.ruanyifeng.com/blog/2010/05/object-oriented_javascript_inheritance_continued.html>

最后修改时间：2019-04-03 15:02:50