没有捷径, 唯有积累。

# 【JS】深拷贝与浅拷贝的区别,实现深拷贝的几种方法

如何区分深拷贝与浅拷贝,简单点来说,就是假设B复制了A,当修改A时,看B是否会发生变化,如果B也跟着变了,说明这是浅拷贝,拿人手短,如果B没变,那就是深拷贝,自食其力。

此篇文章中也会简单阐述到<mark>栈堆,基本数据类型与引用数据类型</mark>,因为这些概念能更好的让你理解深拷贝与浅拷贝。

我们来举个浅拷贝例子:

```
let a=[0,1,2,3,4],
    b=a;
console.log(a===b);
a[0]=1;
console.log(a,b);
```

```
true

▼(5) [1, 1, 2, 3, 4] ① ▼(5) [1, 1, 2, 3, 4] ②

0: 1
1: 1
1: 1
2: 2
2: 2
3: 3
4: 4
4: 4
4: ength: 5

▶ _proto_: Array(0)

▶ _proto_: Array(0)
```

嗯?明明b复制了a,为啥修改数组a,数组b也跟着变了,这里我不禁陷入了沉思。



# 于是我陷入了沉思

那么这里,就得引入基本数据类型与引用数据类型的概念了。

面试常问,基本数据类型有哪些,number,string,boolean,null,undefined,symbol以及未来ES10新增的BigInt(任意精度整数)七类。

引用数据类型(Object类)有常规名值对的无序对象{a:1},数组[1,2,3],以及函数等。

而这两类数据存储分别是这样的:

a.基本类型--名值存储在栈内存中,例如let a=1;

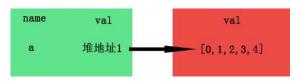
栈内存	
name	val
a	1

当你b=a复制时,栈内存会新开辟一个内存,例如这样:

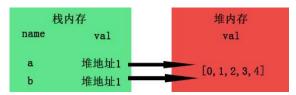
栈内存	
name	val
a	1
b	1

所以当你此时修改a=2,对b并不会造成影响,因为此时的b已自食其力,翅膀硬了,不受a的影响了。当然,let a=1,b=a;虽然b不受a影响,但这也算不上深拷贝,因为深拷贝本身只针对较为复杂的object类型数据。

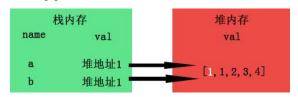
**b.**引用数据类型--名存在栈内存中,值存在于堆内存中,但是栈内存会提供一个引用的地址指向堆内存中的值,我们以上面浅拷贝的例子画个图:



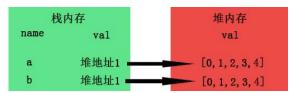
当b=a进行拷贝时,其实复制的是a的引用地址,而并非堆里面的值。



而当我们**a[0]=1**时进行数组修改时,由于a与b指向的是同一个地址,所以自然b也受了影响,这就是所谓的浅拷贝了。



那,要是在堆内存中也开辟一个新的内存专门为b存放值,就像基本类型那样,岂不就达到深拷贝的效果了



1.我们怎么去实现深拷贝呢,这里可以递归递归去复制所有层级属性。

这么我们封装一个深拷贝的函数(PS: 只是一个基本实现的展示,并非最佳实践)

```
function deepClone(obj){
   let objClone = Array.isArray(obj)?[]:{};
   if(obj && typeof obj==="object"){
       for(key in obj){
          if(obj.hasOwnProperty(key)){
              //判断oib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;
let a=[1,2,3,4],
   b=deepClone(a);
a[0]=2;
console.log(a,b);
```

可以看到

跟之前想象的一样,现在b脱离了a的控制,不再受a影响了。

这里再次强调,深拷贝,是拷贝对象各个层级的属性,可以看个例子。JQ里有一个extend方法也可以拷贝对象,我们来看看

```
let a=[1,2,3,4],
  b=a.slice();
a[0]=2;
console.log(a,b);
```

```
11:07:47.929 v (4) [2, 2, 3, 4]  v (4) [1, 2, 3, 4]  v (2)  v (3)  v (4) [1, 2, 3, 4]  v (5)  v (6)  v
```

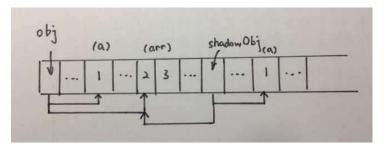
那是不是说slice方法也是深拷贝了,毕竟b也没受a的影响,上面说了,深拷贝是会拷贝所有层级的属性,还是这个例子,我们把a改改

```
let a=[0,1,[2,3],4],
```

```
b=a.slice();
a[0]=1;
a[2][0]=1;
console.log(a,b);
```

拷贝的不彻底啊,b对象的一级属性确实不受影响了,但是二级属性还是没能拷贝成功,仍然脱离不了a的控制,说明slice根本不是真正的 深拷贝。

这里引用知乎问答里面的一张图



第一层的属性确实深拷贝,拥有了独立的内存,但更深的属性却仍然公用了地址,所以才会造成上面的问题。

同理, concat方法与slice也存在这样的情况, 他们都不是真正的深拷贝, 这里需要注意。

### 2.除了递归,我们还可以借用JSON对象的parse和stringify

```
function deepClone(obj){
  let _obj = JSON.stringify(obj),
    objClone = JSON.parse(_obj);
  return objClone
}
let a=[0,1,[2,3],4],
  b=deepClone(a);
a[0]=1;
console.log(a,b);
```

可以看到,这下b是完全不受a的影响了。

附带说下,JSON.stringify与JSON.parse除了实现深拷贝,还能结合localStorage实现对象数组存储。有兴趣可以阅读博客这篇文章。

localStorage存储数组,对象,localStorage,sessionStorage存储数组对象

# **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。

```
let a=[0,1,[2,3],4],
  b=$.extend(true,[],a);
a[0]=1;
a[2][0]=1;
console.log(a,b);
```

可以看到,效果与上面方法一样,只是需要依赖JQ库。

说了这么多,了解深拷贝也不仅仅是为了应付面试题,在实际开发中也是非常有用的。例如后台返回了一堆数据,你需要对这堆数据做操作,但多人开发情况下,你是没办法明确这堆数据是否有其它功能也需要使用,直接修改可能会造成隐性问题,深拷贝能帮你更安全安心的去操作数据,根据实际情况来使用深拷贝,大概就是这个意思。

本文算是个人对于深浅拷贝的学习笔记整理,这里借用了以下资料的思想。

## 【 js 基础 】 深浅拷贝

# js面试题:实现对象深度克隆(deepClone)的三种方案 javascript中的深拷贝和浅拷贝? PS: 下面这些话与深拷贝浅拷贝无关,可以不看 本觉得知识在于分享,不想添加转载标明出去,图片加水印,直到我绑定博客园的邮箱收到了这两条评论: #Re:【JS】深拷贝与浅拷贝的区别,实现深拷贝的几种方法 作者侵权! 请作者附注原文链接 https://www.cnblogs.com/mikeCao/p/8710837.html URL: https://www.cnblogs.com/echolun/p/7889848.html#4109442 #Re:【JS】深拷贝与浅拷贝的区别,实现深拷贝的几种方法 举报,侵权! 评论者: Mike、 URL: https://www.cnblogs.com/echolun/p/7889848.html#4109447 我好奇的点进了他发的'原文链接', 然后发现: //因为b浅拷贝a, ab指向同一个内存地址(堆内存中存的值) //b会随着a的变化而变化 //[2, 2, 3, 4, 5] //[2, 2, 3, 4, 5] h 栈内存 堆内存 name val val 堆地址1 a [1, 1, 2, 3, 4]堆地址1 b 深拷贝实现: function deepClone(obj) 图直接用的我的不说,连我抽烟的表情包都拿过去了.... 全文阐述也只是在我的文章基础上换了下顺序,这就算是一篇"原创"了,我都不敢说自己是原创... 我在他博文下客气的回复也被他删除了。 知识在于分享, 转载标明出处, 不得不说, 我屈服了。 标签: js | 好文要顶 | 关注我 | 收蔵该文 | 💣 🏗 听风是风 关注 - 8 粉丝 - 40 +加关注 « 上一篇: 【JS】for in循环对象, hasOwnProperty()的作用 » 下一篇: Javascript权威指南阅读笔记--第3章类型、值和变量(1) posted @ 2017-11-24 16:01 听风是风 阅读(66106) 评论(27) 编辑 收藏 评论列表



```
谢谢你的提醒,原文已做修改!
                                                                       支持(0) 反对(0)
#3楼 2018-05-15 10:38 Shmily-HJT
                                                                            回复 引用
原来以为深复制是多么复杂,哎,怪我太年轻==。不过第二种暴力解法确实惊艳到了我,前几天阿里的第二道编程笔试题,也是dee
pClone的实现, 哎, 当时没写出来, 惭愧
                                                                       支持(0) 反对(0)
#4楼[楼主] 2018-05-15 11:23 听风是风
                                                                           回复 引用
@ Shmily-HJT
这篇博客也是面试的时候整理的....本来就是一个学习的过程啊,知道深拷贝这个东西,以后不管面试还是会用到,脑袋里立马会想到
概念就挺好了。加油加油。
                                                                       支持(0) 反对(0)
#5楼 2018-05-15 17:16 Shmily-HJT
                                                                           回复 引用
@ 听风是风
上午借鉴你的思路,顺便实现了对象的合并去重,哈哈哈
https://www.zhihu.com/question/275792654
                                                                       支持(0) 反对(0)
#6楼 2018-06-06 18:30 逗伴不是瓣
                                                                           回复 引用
关于函数的深浅拷贝有提示吗?
                                                                       支持(0) 反对(0)
#7楼[楼主] 2018-06-06 18:55 听风是风
                                                                            回复 引用
@ 逗伴不是瓣
最简单的,拷贝完成后修改之前的对象,做个对比就知道了啊。
                                                                       支持(0) 反对(0)
#8楼 2018-06-26 18:44 Yukiee
                                                                           回复 引用
写的好详细,博主可以转载吗?
                                                                       支持(0) 反对(0)
#9楼[楼主] 2018-06-26 18:58 听风是风
                                                                           回复 引用
@ Yukiee
过奖了,可以转载的。
                                                                       支持(0) 反对(0)
#10楼 2018-09-12 10:55 Bulewater
                                                                            回复 引用
引用数据类型--名存在栈内存中,值存在于堆内存中,
请问, 当引用数据变量析构了(出栈)时,它所指向的值(堆内存)会释放吗?
例如
void test ()
QList<QString> list_str;
list str.append("1111");
list_str.append("2222");
list str生命周期到了, "1111"和"2222"会释放吗?
                                                                       支持(0) 反对(0)
#11楼[楼主] 2018-09-12 11:15 听风是风
                                                                           回复 引用
说说我的理解,这个要看堆内存提供的地址还有没有被栈中的某个变量引用,如果引用它的变量都被释放了,堆内存中的值处于闲置状
态,是会被垃圾回收机制回收释放的,避免内存的浪费。
                                                                       支持(0) 反对(0)
#12楼[楼主] 2018-09-12 11:29 听风是风
                                                                            回复 引用
@ Bulewater
顺便提下栈堆分配的原则概念, 便于理解。

    栈 (操作系统): 由操作系统自动分配释放 , 存放函数的参数值 , 局部变量的值等。其操作方式
类似于数据结构中的线;

 2、堆(操作系统): 一般由程序员分配释放, 若程序员不释放,程序结束时可能由OS回收,分配
方式倒是类似于链表,
                                                                       支持(0) 反对(0)
```

**⋓** 状示的双十姓

看完之后,还不太理解,但是一行一行敲下来,然后每行都打印一下,才明白了,写的还是很好的,但是我还有个问题,就是JSON.st ringify(obj) 和 parse 拷贝的时候是怎么样的一个原理,这点还不太明白,博主有空给就给讲一下呗,或者直接把结果发我邮箱,we b\_xiaoke@163.com, 谢谢博主

支持(0) 反对(0)

#14楼[楼主] 2018-09-18 19:33 听风是风

回复 引用

#### @ web 可可

首先得明白JSON.stringify()与JSON.parse()的作用,针对你的疑问,我们可以这样理解,前者能将一个对象转为json字符串(基本类型),后者能将json字符串还原成一个对象(引用类型)。

基本类型拷贝是直接在栈内存新开空间,直接复制一份名-值,两者互不影响。

而引用数据类型,比如对象,变量名在栈内存,值在堆内存,拷贝只是拷贝了堆内存提供的指向值的地址,而JSON.stringify()巧就巧在能将一个对象转换成字符申,也就是基本类型,那这里的原理就是先利用JSON.stringify()将对象转变成基本数据类型,然后使用了基本类型的拷贝方式,再利用JSON.parse()将这个字符串还原成一个对象,达到了深拷贝的目的。

JSON.stringify()对对象的转换,避开了只拷贝地址指向,无法直接拷贝值本身的问题,不知道这样说你能否理解。

JSON.stringify()与JSON.parse()的作用还是挺强大的,如果有兴趣,可以看看博主这篇关于这两个方法的小归纳。

https://www.cnblogs.com/echolun/p/9631836.html

希望有帮助到你。

支持(0) 反对(0)

#15楼 2018-10-20 15:25 王凯的影迷朋友

回复 引用

mark

支持(0) 反对(0)

#16楼 2018-11-06 17:25 shoushou70

回复 引用

基本类型使用递归的深拷贝方法只能返回一个空对象

支持(0) 反对(0)

#17楼[楼主] 2018-11-06 17:29 听风是风

回复 引用

#### @ shoushou70

因为深拷贝本身是针对于复杂类型的数据。

基本数据类型直接复制,在内存中会直接新开内存,不存在改其中一个影响另一个变量的问题,所以基本数据类型不会用到深拷贝的方法。

严谨点递归里确实是需要做一个数据判断的,这里只是作为一个实现展示,谢谢指正。

支持(0) 反对(0)

# #18楼 2018-11-19 10:22 呵呵python

回复 引用

写的不错

支持(0) 反对(0)

#19楼[楼主] 2018-11-19 11:59 听风是风

回复 引用

### @ IIIIpvthon

谢谢夸张,能帮助更多人才是最好的。

支持(0) 反对(0)

## #20楼 2019-01-04 11:06 我爱搬砖2018

回复 引用

博主,文章写的不错,但以下地方应该有错误:

let a=[1,2,3,4] b=a.slice(); a[0]=2;

console.log(a,b);

你看一下哈!

支持(0) 反对(0)

## #21楼[楼主] 2019-01-04 11:12 听风是风

回复 引用

## @ 我爱搬砖2018

看来是跟着实践了一遍,确实错了,哈哈,已修正,非常感谢指出!



支持(0) 反对(0)

#22楼 2019-01-18 09:21 不安静的小游客

回复 引用

哇! 佩服佩服 感谢楼主分享!!!!!!

+++(0) ==+(0)

	支持(0) 反对(0)
#23楼[楼主] 2019-01-18 09:24 听风是风	回复 引用
◎ 不安静的小游客 过奖了! 只是整理一些我个人的理解,互相学习的过程。这么早就在学习了,佩服佩服。	
	支持(0) 反对(0)
#24楼 2019-01-23 13:56 lyt_angularjs	回复 引用
楼主,递归有溢出的风险.而且需要考虑到不同类型的克隆是不一样的.感兴趣可以看看我的这篇文章.互相挤 https://blog.csdn.net/lyt_angularjs/article/details/86599820	<b>采讨</b> .
	支持(0) 反对(0)
#25楼[楼主] 2019-01-23 14:06 听风是风	回复 引月
@ lyt_angularjs 谢谢你的建议,其实这篇文章更多是偏向于深拷贝概念的讲解,什么是深拷贝,对于实现文中我也说了只 方案,针对于正则等特殊对象的深拷贝确实满不足,很谢谢你的实现分享。	是基本实现,确实不是最佳的
	支持(0) 反对(0)
#26楼 2019-03-22 11:37 神隐小菜	回复 引月
谢谢博主,原来只知道个模糊,现在栈内存和堆内存一说就全都清楚了	
	支持(0) 反对(0)
#27楼[楼主] 2019-03-22 11:52 听风是风	回复 引月
#27楼[楼主] 2019-03-22 11:52 听风是风 @ 神隐小菜 拷贝确实牵扯到了内存的问题,画图也方便理解,对你有帮助那当然是最好的,共同学习!	回复 引

刷新评论 刷新页面 返回顶部



- 【推荐】超50万C++/C#源码: 大型实时仿真组态图形源码
- 【推荐】百度云"猪"你开年行大运,红包疯狂拿,低至1折
- 【推荐】专业便捷的企业级代码托管服务 Gitee 码云
- 【活动】2019第四届全球人工技术大会解码"智能+时代"



# 相关博文:

- ·浅拷贝与深拷贝的实现
- · js深拷贝和浅拷贝
  · js深拷贝与浅拷贝
- · 深拷贝与浅拷贝的区别
- ·js深拷贝与浅拷贝

## 最新新闻:

- · Naspers要和腾讯"分家" 但摆脱不了"腾讯依赖症"
- · 团货网暴雷背后的谎言与真相 · 特斯拉2亿美元收购电池公司Maxwell难产 2月还未完成 · 科学家首次在实验中让原子伴着光子"跳舞"
- · 盛大游戏宣布更名为盛趣游戏 将转型科技文化公司
- » 更多新闻...

Copyright ©2019 听风是风