**摘要：** 之前去一家公司面试的时候，面试官问了我一个问题，说:"如何才能深拷贝一个对象"。当时我心里有些窃喜，这么简单的问题还用想吗？于是脱口而出:"平时常用的有两种办法，第一种用JSON.parse(JSON.stringify(obj))，第二种可以使用for...in加递归完成"。

前言

之前去一家公司面试的时候，面试官问了我一个问题，说:"如何才能深拷贝一个对象"。当时我心里有些窃喜，这么简单的问题还用想吗？于是脱口而出:"平时常用的有两种办法，第一种用JSON.parse(JSON.stringify(obj))，第二种可以使用for...in加递归完成"。面试官听了以后点了点头觉得挺满意的。 当时我也并没有太过在乎这个问题，直到前段时间又想起这个问题，发现上面说的两种方法都是有Bug的。

提出问题

那么上面所说的Bug是什么呢？

* 特殊对象拷贝

首先让我们试想有这么一个对象，在不考虑普通类型的情况下，它有如下成员:

const obj = {

arr: [111, 222],

obj: {key: '对象'},

a: () => {console.log('函数')},

date: new Date(),

reg: /正则/ig

}

然后我们用上面两种方式分别拷贝一次

JSON法

JSON.parse(JSON.stringify(obj))

输出结果:



可以从中看出，obj中的普通对象和数组都能拷贝，然而date对象成了字符串，函数直接就不见了，正则成了一个空对象。

再来看看for...in加递归的方法

递归

function isObj(obj) {

    return (typeof obj === 'object' || typeof obj === 'function') && obj !== null

}

function deepCopy(obj) {

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

    for(let key in obj) {

        tempObj[key] = isObj(obj[key]) ? deepCopy(obj[key]) : obj[key]

    }

    return tempObj

}

结果:



结论

通过上面的测试可知，这两个方法都无法拷贝函数，date，reg类型的对象;

* 环

什么是环?

环就是对象循环引用，导致自己成为一个闭环，例如下面这个对象:

var a = {}

a.a = a



使用上面两个方法拷贝一下会直接报错



解决方案

* 环

可以使用一个WeakMap结构存储已经被拷贝的对象，每一次进行拷贝的时候就先向WeakMap查询该对象是否已经被拷贝，如果已经被拷贝则取出该对象并返回，将deepCopy函数改造成如下

function deepCopy(obj, hash = new WeakMap()) {

    if(hash.has(obj)) return hash.get(obj)

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

    hash.set(obj, cloneObj)

    for (let key in obj) {

        cloneObj[key] = isObj(obj[key]) ? deepCopy(obj[key], hash) : obj[key];

    }

    return cloneObj

}

拷贝环结果:



* 特殊对象的拷贝

这个问题的解决比较麻烦，因为需要特别对待的对象种类实在太多，于是我参考了MDN上的[结构化拷贝](https://link.juejin.im/?target=https%3A%2F%2Fdeveloper.mozilla.org%2Fzh-CN%2Fdocs%2FWeb%2FGuide%2FAPI%2FDOM%2FThe_structured_clone_algorithm%23%25E7%259B%25B8%25E5%2585%25B3%25E9%2593%25BE%25E6%258E%25A5" \t "_blank)，然后结合解决环的方案：

// 只解决date，reg类型，其他的可以自己添加

function deepCopy(obj, hash = new WeakMap()) {

    let cloneObj

    let Constructor = obj.constructor

    switch(Constructor){

        case RegExp:

            cloneObj = new Constructor(obj)

            break

        case Date:

            cloneObj = new Constructor(obj.getTime())

            break

        default:

            if(hash.has(obj)) return hash.get(obj)

            cloneObj = new Constructor()

            hash.set(obj, cloneObj)

    }

    for (let key in obj) {

        cloneObj[key] = isObj(obj[key]) ? deepCopy(obj[key], hash) : obj[key];

    }

    return cloneObj

}

拷贝结果:



完整版可以查看[lodash深拷贝](https://link.juejin.im/?target=https%3A%2F%2Fgithub.com%2Flodash%2Flodash%2Fblob%2Fmaster%2F.internal%2FStack.js%3F1528957393114)

* 函数的拷贝

但是MDN上的结构化拷贝依旧没有解决函数的拷贝



目前为止，我只想到使用eval的方法对函数进行拷贝，但是这种方法只能对箭头函数生效，如果是fun(){}这种形式的则会出错

拷贝函数增加函数类型



出错类型



后记

JavaScript的深拷贝还不止上面所说的这些坑，还存在的问题有如何拷贝原型链上的属性？如何拷贝不可枚举属性? 如何拷贝Error对象等等的坑，在这里就不一一赘述了。

不过在日常过程中还是建议使用JSON方法，这个方法已经覆盖了绝大部分的业务需求，所以不需要把简单的事情复杂化，不过面试中如果遇到面试官钻牛角尖对这个问题的解答绝对可以秀他一脸了。

<https://yq.aliyun.com/articles/610052?utm_content=m_1000006154>