vue源码

Vue3 MVVM - 1. Reactive

Reactive

reactive 底层使用的是proxy，此方法接受一个参数，参数的类型需要对象。返回一个proxy代理过的对象。开始之前我们先搞懂什么是proxy。代理的意思就是找人帮忙做事情，这里的帮忙做事情的含义就是当给对象属性赋值或者从对象属性取值的时候就不直接返回了，让代理返回。

为什么需要代理呢？不用代理之前也可以好好的给对象赋值或者从对象取值。使用代理是因为代理可以在默认的行为上面添加自己的行为。比如往一个对象属性上面赋值一个数字，该数字可以是来自用户输入的。但是项目想让这个值再赋值之前*10。比如用户输入的是10，但是实际想存储的是100。当然，我们可以简单的在这个参数上面*10然后再赋值。这个是方法之一。那有没有可以让我们不显示的*10，让对象赋值之前自动*10。这个就可以使用代理。

```
let a = {'abc': 1}
let p = new Proxy(a, {
    set(target,key,value,receiver) {
        Reflect.set(target, key, value*10, receiver)
        return true
    },
    get (target,key,receiver) {
        return Reflect.get(target,key,receiver)
    },
    deleteProperty(target, key) {
        // 删除一个属性，代码省略
        return true
    }
})
p.a = 10
console.log(p.a) // 100
```

这个就是代理的使用场景，让对象属性赋值或者取值之前做额外的自定义操作。

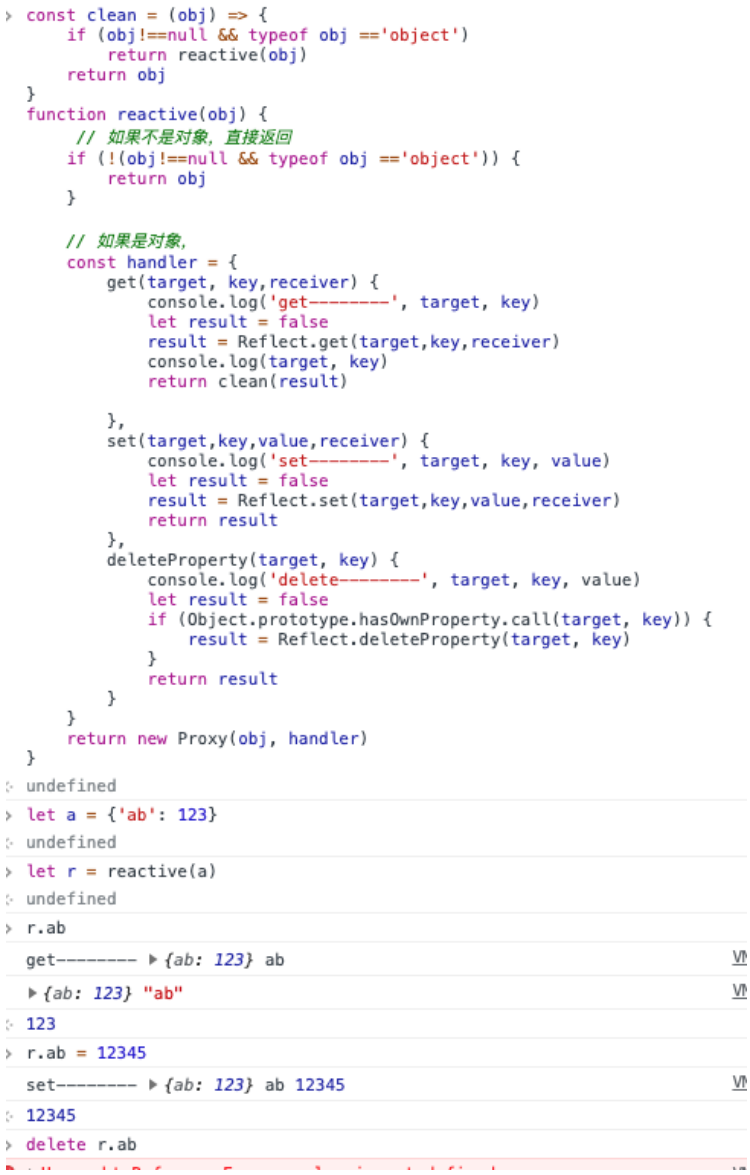
vue3 里面的reactive 内部其实很类似于上面的代码，我们现在来完成一个简单版本的，

```
const clean = (obj) => {
    if (obj!==null && typeof obj =='object')
        return reactive(obj)
    return obj
}
function reactive(obj) {
     // 如果不是对象，直接返回
    if (!(obj!==null && typeof obj =='object')) {
        return obj
    }

    // 如果是对象，
    const handler = {
        get(target, key,receiver) {
            console.log('get--------', target, key)
            let result = false
            result = Reflect.get(target,key,receiver)
            console.log(target, key)
            return clean(result)

        },
        set(target,key,value,receiver) {
            console.log('set--------', target, key, value)
            let result = false
            result = Reflect.set(target,key,value,receiver)
            return result
        },
        deleteProperty(target, key) {
            console.log('delete--------', target, key)
            let result = false
            if (Object.prototype.hasOwnProperty.call(target, key)) {
                result = Reflect.deleteProperty(target, key)
            }
            return result
        }
    }
    return new Proxy(obj, handler)
}
```

上面的代码设置非常简单，可能有个小问题会迷惑，就是set函数有返回值。一般没有返回值，这里的返回值表达是否设置成功。然后我们来做个测试，



可以在图片里面看到，现在当我们获取或者删除或者设置，都会使用代理来完成，代理可以在做真正的操作之前添加额外的自定义逻辑。其实这里的set 方法完全可以先判断即将设置的新的值是否和已经存在的老的值相等，如果相等则直接返回不再设置，增加性能，

```
...
set (target,key,value, receiver) {
    let result = true
    let previousVal = Reflect.get(target,key,receiver)
    if (previousVal===value) return result
    result = Reflect.set(target,key,value,receiver)
    return result
}
...
```

这里还需要说一下的地方，为了面试，可能会问为什么 vue 3 在 vue 2 响应式设计上面性能更好？

其实你看了上面的代码可以知道，vue 3 的reactive是【懒】响应式，它首先定义了响应式的行为该是什么样子，但是却没有在第一时间处理，只有当你访问相对应的属性才会做出处理。相比较vue2 直接把所有的属性不管是否有被使用都直接先处理的方法，肯定性能更好。

那么现在我们继续下去，vue3里面需要使用proxy 在添加属性和获取属性的时候自定义什么逻辑呢？回想一下，vue3里面当我们设置一个对象是reactive 响应式，然后更改了这个对象里面相对应的值，如果我们使用类似于vue3 提供的watchEffect(()=>{...}) ，而且函数内部有用到这个对象内的某一个或者多个属性，那么就会在我们每次更新这个属性的时候，watchEffect(()=>{}) 里面的函数就会被运行一次，从而看上去像是响应了我们的操作，即响应式。

那么，实现的逻辑和原理你能想到么？让我们在下一节演示，MVVM - 依赖收集。