vue源码

vue3 MVVM - toRef && toRefs

toRefs 接受一个reactive对象，内部原理是让这个对象内部的每一个属性都返回一个类似ref的对象，就是把reactive对象里面的所有属性外面包裹一层，目的就是通过 obj.property.value 的形式替代 obj.property的形式来进行对reactive对象属性的操作，这么做就是为了保存reactive对象的响应性。

如果传入的对象不是reactive，则没有处理的需要直接返回即可。那么下面我们用代码实现，

```
function toRefs(obj) {
    // isReactive 方法检测是否obj为reactive 对象
    if (!isReactive(obj)) return obj
    let newObj = {}
    for (const key in obj) {
        newObj[key] =  process(obj, key)
    }

    return newObj
}

function process(obj, key) {
    return {
        __v_isRef: true,
        get value() {
            return obj[key]
        },
        set value(newValue) {
            obj[key] = newValue
        }
    }
}
```

那么考虑一下为什么经过上面的代码之后，解构reactive对象也可以是响应式的呢。其实这个需要知道最主要的重点就是什么时候响应式会响应。当我们把所有属性通过process方法之后，使用 .value 访问或者赋予新值，内部会让reactive的getter 和 setter 来处理。

如果不做这一层的封装，那么当解构一个响应式对象，只是直接创建一个新的对象，你在这个解构出来的新对象上面的操作比如赋值，都不会让响应式起效果，因为没有通过响应式对象来进行操作。

上面的代码没有做边界处理，但是足够可以展示toRefs的底层实现。

与toRefs相同, toRef可以让reactive对象的某一个属性进行响应式保存。

```
function toRef(obj, key) {
    return {
        __v_isRef: true,
```

```
        get value() {
            return obj[key]
        },
        set value(newValue) {
            obj[key] = new Value
        }
    }
}
```