深度监听 data 变化

```
// 触发视图更新
function updateView() {
    console.log("视图更新");
}

// 重新定义数组原型，不会影响原生的数组原型方法、属性
const oldArrayProperty = Array.property;
// 创建新对象，原型指向 oldArrayProperty，再扩展新的方法，不会影响原型
const arrProto = Object.create(oldArrayProperty);
['push', 'pop', 'unshift', 'shift', 'join', 'splice'].forEach(methodName => {
    arrProto[methodName] = function () {
        updateView();
        oldArrayProperty[methodName].call(this, ...arguments);
        // Array.prototype.push.call(this, ...arguments);
    }
});

// 重新定义属性，监听起来
function defineReactive(target, key, value) {
    // 深度监听
    observer(value);

    Object.defineProperty(target, key, {
        get() {
            return value;
        },
        set(newValue) {
            // 深度监听
            observer(newValue);

            if (newValue !== value) {
                value = newValue;

                // 触发视图更新
                updateView();
            }
        }
    })
}

// 监听对象属性
function observer(target) {
    if (typeof target !== "object" || target === null) {
        // 不是对象或数组
        return target;
    }
    if (Array.isArray(target)) {
        target.__proto__ = arrProto;
```

```
    }

    // 重新定义各个属性（for in 也可以遍历数组）
    for (let key in target) {
        defineReactive(target, key, target[key]);
    }

}

// 准备数据
const data = {
    name: "张三",
    age: 20,
    info: {
        address: "成都",
        phone: "18227752001"
    },
    favoriteColor: ['red', 'black'];
};

// 监听数据
observer(data);

// 测试
data.name = "李四";
data.age = 21;
data.info.address = "重庆";
data.favoriteColor[0] = 'green';
```

缺点：

- **深度监听，需要递归到底，计算量很大。**
- **无法监听新增属性/删除属性（Vue.set, Vue.delete）。**
- **无法原生监听数组，需要特殊处理。**