8.ReactiveProperty

UniRx 中有个一个非常强大的概念，叫做 ReactiveProperty。响应式属性。

强大在哪呢？它可以替代一切变量，给变量创造了很多功能。

假如我们想监听一个值是否发生了改变。

用通常的方法实现可能如下：

```
public int Age
{
    get
    {
        ...
    }
    set
    {
        if (mAge != value)
        {
            mAge = value;
            // send event
            OnAgeChanged();
            // call delegate
        }
    }
}

public void OnAgeChanged()
{

}
```

这样在类的内部，写一次 OnAgeChanged 是没问题的。但是我想在这个类的外部监听这个值的改变，那就要声明一个委托来搞定了。委托的维护成本比较低，是可以接受的，直到笔者发现了 UniRx 的 ReactiveProperty。就再也不想用委托来做这种工作了。

UniRx 实现：

```
public ReactiveProperty<int> Age = new ReactiveProperty<int>();

void Start()
{
    Age.Subscribe(age =>
    {
        // do age
    });

    Age.Value = 5;
}
```

当任何时候，Age 的值被设置，就会通知所有 Subscribe 的回调函数。

而 Age 可以被 Subscribe 多次的。

并且同样支持 First、Where 等操作符。

这样可以实现一个叫做 MVP 的架构模式。

也就是 在 Ctrl 中，进行 Model 和 View 的绑定。

Model 的所有属性都是用 ReactiveProperty，然后在 Ctrl 中进行订阅。

通过 View 更改 Model 的属性值。

形成一个 View->Ctrl->Model->Ctrl->View 这么一个事件响应环。

这里只是一个简介，具体的在下一堂课进行介绍。