4.操作符 Where

上堂课，老师介绍了 UniRx 的基本语法格式。

其中有一句话：UniRx 的侧重点，不是发布者和订阅者这两个概念如何使用，而是事件从发布者到订阅者之间的过程如何处理。

如何处理呢?

今天就先处理一个给大家看看。

在之前，我们有介绍 Update API。

其中有一段代码如下:

```
Observable.EveryUpdate()
        .Subscribe(_ =>
        {
            if (Input.GetMouseButtonUp(0))
            {
                // do something
            }
        }).AddTo(this);
```

这段代码实现了鼠标按钮的点击事件处理。

但是这段代码，还不够简洁。如何更加简洁?

使用 Where 操作符。

代码如下:

```
Observable.EveryUpdate()
```

```
        .Where(_ => Input.GetMouseButtonUp(0))
        .Subscribe(_ =>
        {
            // do something
        }).AddTo(this);
```

Where 意思是在哪儿。这里我们理解成做一个条件语句。也就是 if 语句。

类似:

```
if (Input.GetMouseButtonUp(0))
```

这段代码和之前的一样。

Where 是一个过滤的操作，过滤掉不满足条件的事件。

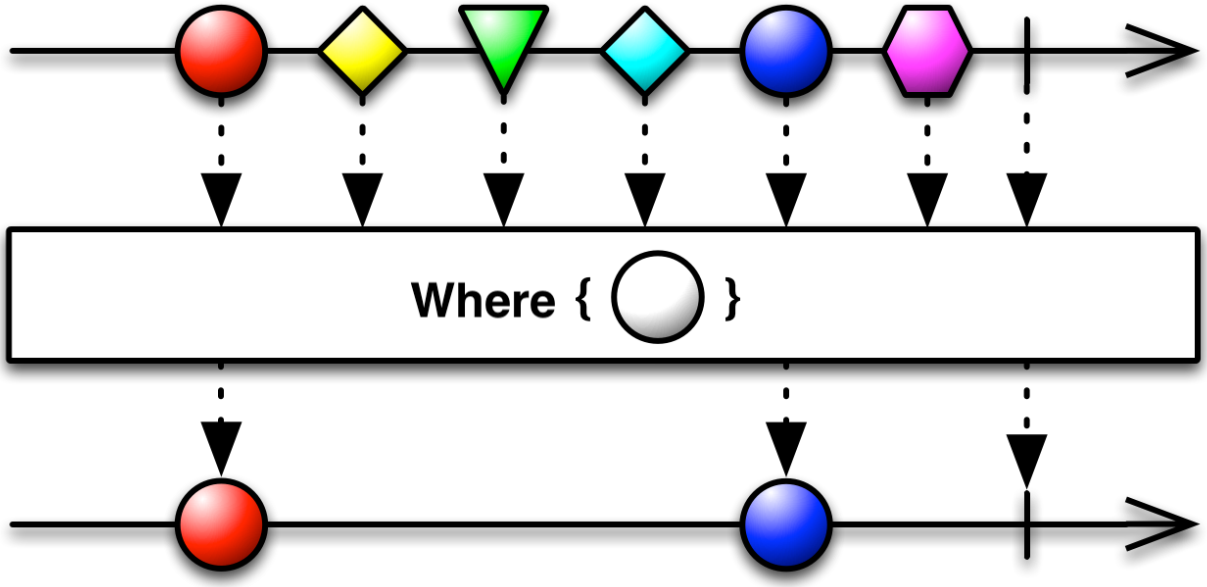
老师呢，给大家一个比较容易理解的解释。

1. EveryUpdate 是事件的发布者。他会每帧会发送一个事件过来。
2. Subscribe 是事件的接收者，接收的是 EveryUpdate 发送的事件。
3. Where 则是在事件的发布者和接收者之间的一个过滤操作。会过滤掉不满足条件的事件。

所以，Subscribe 处理的事件，都是满足 Input.GetMouseButtonUp(0) 条件的事件。

看一下这两幅图，就可以理解了。


```
Where(x => x > 10)
```



是圆形的，才可以通过。

事件的本身可以是参数，但是 EveryUpdate 没有参数，所以在 Where 这行代码中不需要接收参数，所以使用 _ 来表示，不用参数。当然 Subscribe 也是用了一个 _ 来接收参数。

在两幅图中，第一幅图，发送的事件的类型是整数类型。第二个不清楚，应该是自定义 class 吧。

总之，Where 操作符就介绍到这里。相信大家理解了 UniRx 的这种编程模型。