5.Coroutine 的操作

UniRx 对 Unity 的 Coroutine 也提供支持，可以将一个 Coroutine 转化为事件源（Observable）。

```
public class RxCoroutineTest : MonoBehaviour
{
    IEnumerator CoroutineA()
    {
        yield return new WaitForSeconds(1.0f);
        Debug.Log("A");
    }

    void Start()
    {
        Observable.FromCoroutine(CoroutineA())
                .Subscribe(_ =>
                {
                    // do something
                }).AddTo(this);
    }
}
```

一秒之后，输出结果为：

```
A
```

非常简单。

当然也支持将 Observable 转化为一个 Coroutine 中的 yield 对象。

比如:

```
public class Rx2YieldTest : MonoBehaviour
{
    IEnumerator Delay1Second()
    {
        yield return
Observable.Timer(TimeSpan.FromSeconds(1.0f)).ToYieldInstruction();
        Debug.Log("B");
    }

    void Start()
    {
        StartCoroutine(Delay1Second());
```

```
    }
}
```

一秒之后，输出结果为:

```
B
```

FromCoroutine 和 ToYieldInstruction 实现了 Observable 与 Coroutine 之间的互相转化。

而在之前说过，Observable 是一条事件流。UniRx 的操作符，比如 Merge 可以处理多个流，可以将流进行合并。

除了合并也支持别的操作，比如 顺序 (依赖) 执行 Coroutine，并行执行 Coroutine 等等。

在之后，通过学习新的操作符，可以让 Coroutine 更加强大。

今天的内容就这些。