2.IObserver 与 IEnumerator

首先给出 IObserver 与 IEnumerator 的定义。

IObserver.cs

```
public interface IObserver<T>
{
    void OnCompleted();

    void OnError(Exception error);

    void OnNext(T value);
}
```

IEnumerator

```
public interface IEnumerator
{
    bool MoveNext();

    object Current { get; }

    void Reset();  // 忽略。
}
```

这两个接口看起来非常相似，其作用也是非常相像的。

我们逐步分析下。

IEnumerator.Current 与 IObservaber.OnNext

IEnumerator.Current 是用于获取当前数据的，直接从远端拉，然后返回一个数据。

而 IObservaber.OnNext 则是远端推一个数据(事件)过来的时候进行的相应的处理(捕获)。

IEnumerator.MoveNext 与 IObserver. OnCompleted 、 OnError.

当 MoveNext 为 False 的时候，则 foreach 完成迭代。

而当 OnCompleted 或 OnError 被调用时，也是类似，是完成了事件的捕获。

非常相像，那么 LINQ 与 Rx 的区别是什么呢?

唯一的区别就是，Rx 是非阻塞的，迭代是阻塞的。

Rx 是用了一种观察者模式的方法，本身就是在时间上异步的，而且常常处于一种”监听”数据的状态，所以是非阻塞的。

而 LINQ 本质上就是迭代器与静态扩展关键字，它是在时间上同步的，遍历时，当有一些数据出现了一些问题，则非常有可能拉取不到数据，从而导致程序卡死、崩溃等异常。

总结如下:

UniRx 的数据(事件)是从 IObservable 推送过来，然后在 IObserver 进行捕获。

而 LINQ 则是，在 IEnumerable 扔过来一个吸管，然后在遍历时(IEnumerator)进行拉取数据。

今天的内容就这些。