# 2. UI 增强

作为初学者，在日常开发中接触得最多的就是 UGUI 了。而 UGUI 的开发大多需要遵循着一个 MVC 的模式。

但是 MVC 模式对很多人来说是一个非常模糊的架构模式。但是本质很简单，想办法把表现和数据分离。也就是 View 和 Model 分离。

用 Unity 实现的方式有非常多种。

而用 UniRx 的 Reactive Property 则是可以完全实现一种 MVC 的变种（MVP），并且是非常明确的。这样在开发的时候就不用再去纠结怎么实现了。

单单这一个概念，就让一个 UGUI 的开发简化了很多

除此之外，还支持了非常多的 UGUI 控件。

所有的 UGUI 控件支持列出如下：

```
[SerializeField] Button mButton;
[SerializeField] Toggle mToggle;
[SerializeField] Scrollbar mScrollbar;
[SerializeField] ScrollRect mScrollRect;
[SerializeField] Slider mSlider;
[SerializeField] InputField mInputField;

void Start()
{
    mButton.OnClickAsObservable().Subscribe(_ => Debug.Log("On Button Clicked"));

    mToggle.OnValueChangedAsObservable().Subscribe(on => Debug.Log("Toggle " +
on));

    mScrollbar.OnValueChangedAsObservable().Subscribe(scrollValue =>
Debug.Log("Scrolled " + scrollValue));

    mScrollRect.OnValueChangedAsObservable().Subscribe(scrollValue =>
Debug.Log("Scrolled " + scrollValue);

    mSlider.OnValueChangedAsObservable().Subscribe(sliderValue =>
Debug.Log("Slider Value " + sliderValue));

    mInputField.OnValueChangedAsObservable().Subscribe(inputText =>
Debug.Log("Input Text: " + inputText));

    mInputField.OnEndEditAsObservable().Subscribe(result => Debug.Log("Result :" +
result));
}
```

以上就是所有的 Observable 支持。

当然除了 Observable 增强，还支持了 Subscribe 的增强。

比如 SubscribeToText

```
Text resultText = GetComponent<Text>();

mInputField.OnValueChangedAsObservable().SubscribeToText(resultText);
```

这段代码实现的功能是，当 mInputField 的输入值改变，则会马上显示在 resultText 上。

也就是完成了，mInputField 与 resultText 的绑定。

除此之外还支持，SubscribeToInteractable。基本上这样就够用了。

本节课介绍的就是 UniRx 对 UI 的全部支持。

今天的内容就这些。