WPF

Microsoft MVP

Connor Park

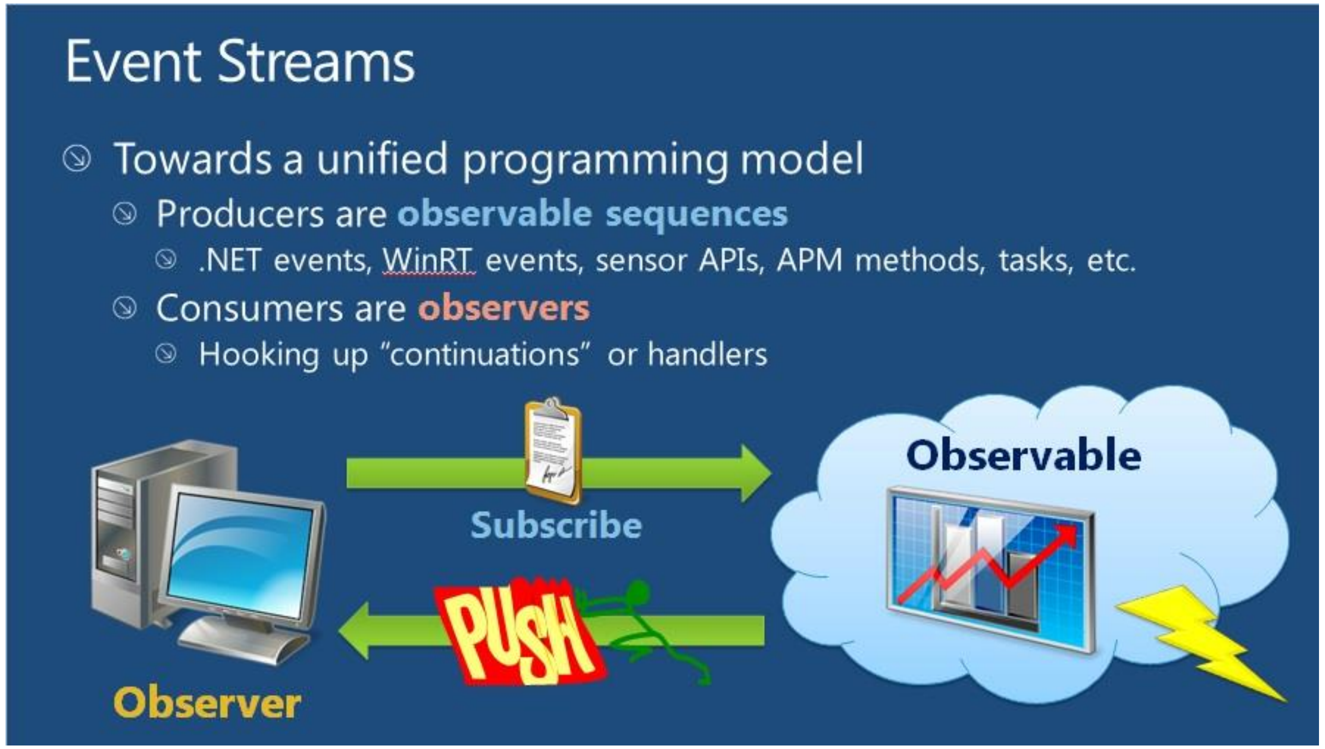
Day 5

Reactive Extension(RX)

http://reactivex.io/

Rx overview

- Curing Your Event Processing Blues with Reactive Extension (Rx)
- 이벤트를 감시하는 observable sequences를 생성
- Observer가 원하는 observable를 구독

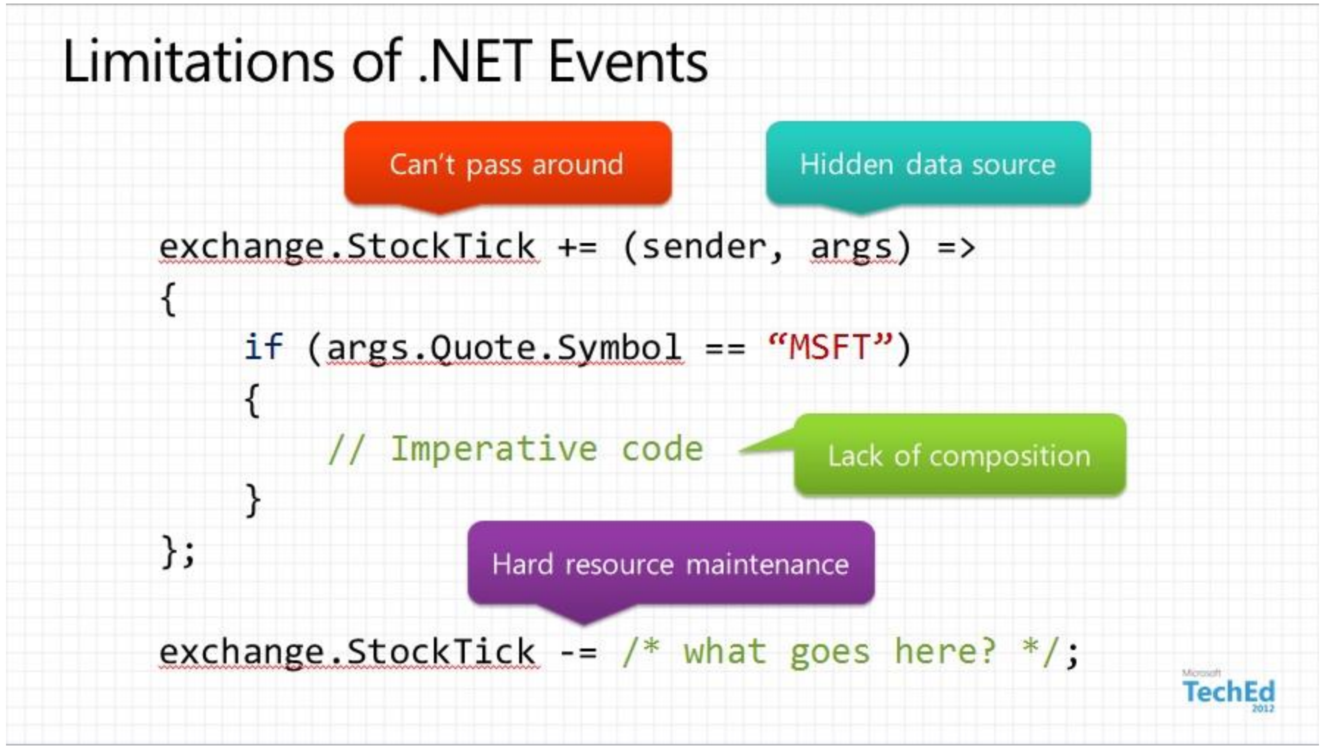


Event Streams

- Towards a unified programming model
  - Producers are **observable sequences**
    - .NET events, WinRT events, sensor APIs, APM methods, tasks, etc.
  - Consumers are **observers**
    - Hooking up "continuations" or handlers

.NET Event의 한계

- 이벤트를 object가 아니기 때문에…
- 이벤트 아규먼트를 이용하려면?
- 이벤트 핸들러 처리시 Lack 발생
- 이벤트 핸들러의 해지 불가

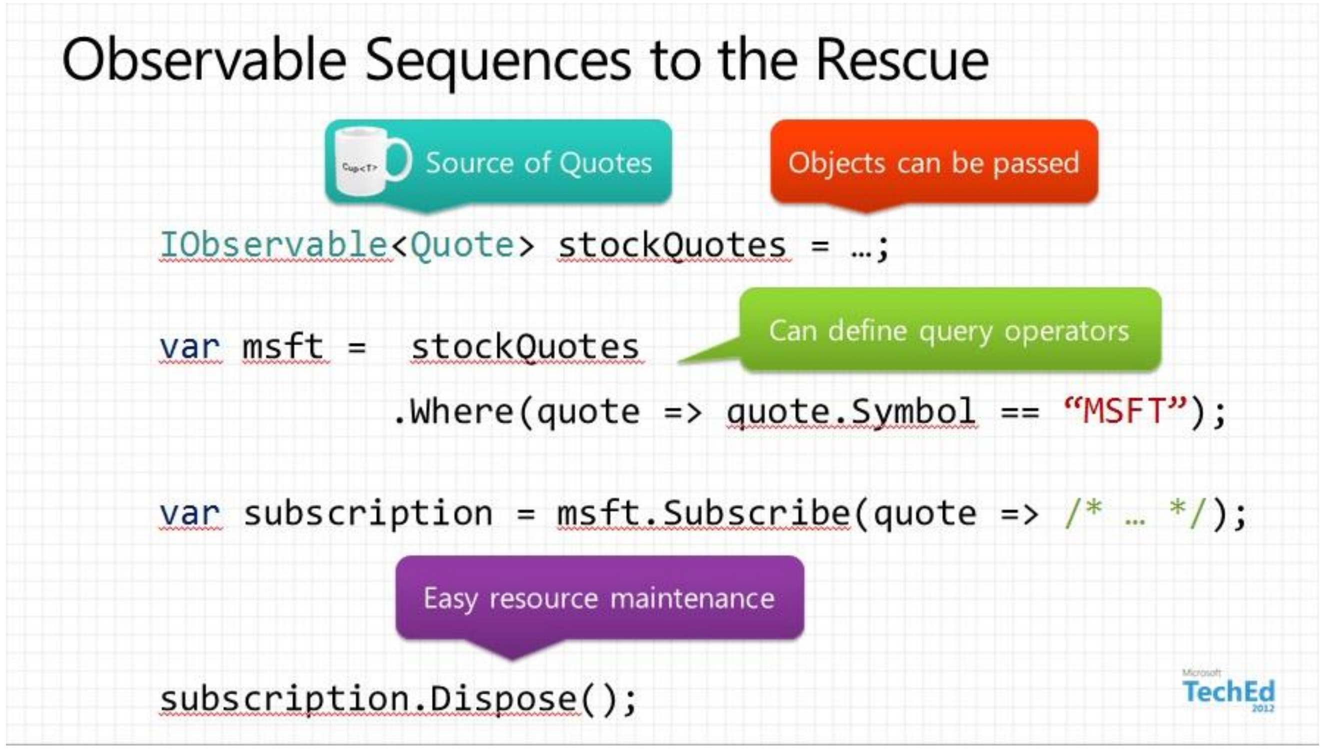


Limitations of .NET Events

```
exchange.StockTick += (sender, args) =>
{
    if (args.Quote.Symbol == "MSFT")
    {
        // Imperative code
    }
};

exchange.StockTick -= /* what goes here? */;
```

Rx가 해결?

- 이벤트 핸들러를 연결하지 않아도 …
- 이벤트를 IObservable<T> objec로 처리 …
    - 다른 곳으로 전송 가능
    - 저장 가능
    - LINQ, Lamda
- Dispose 가능



Observable Sequences to the Rescue

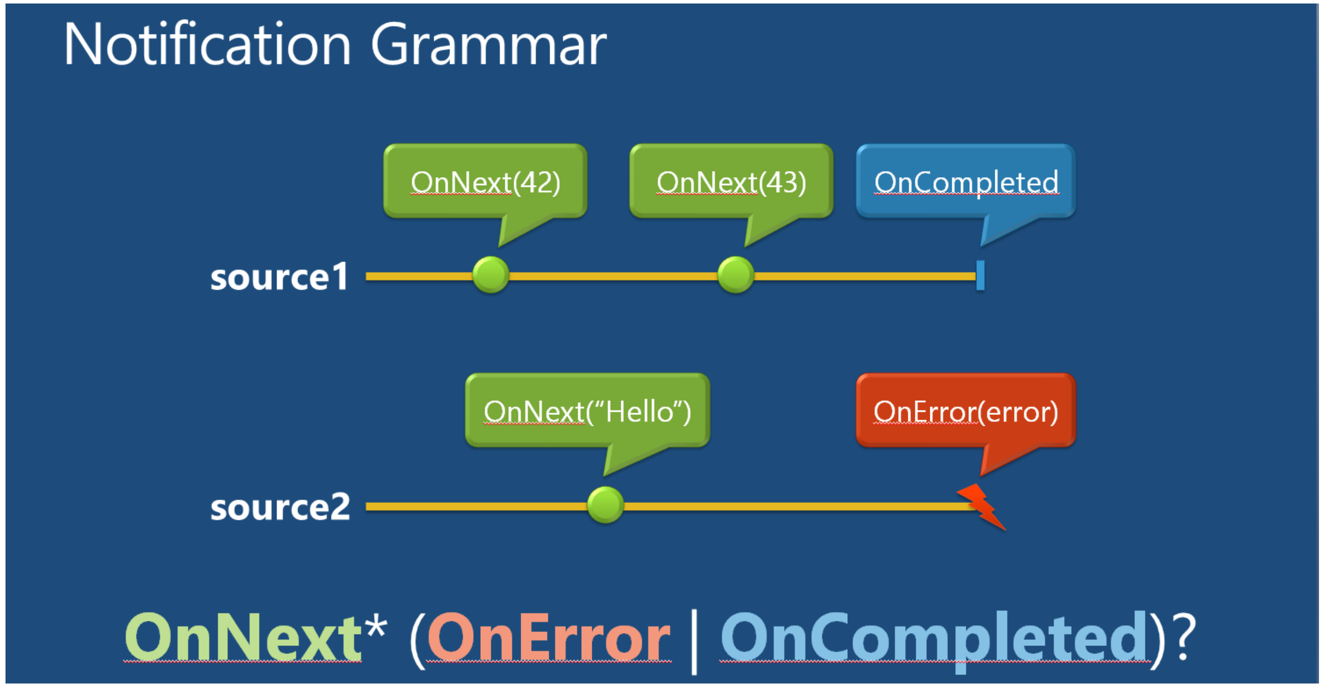
```
IObservable<Quote> stockQuotes = …;

var msft =  stockQuotes
              .Where(quote => quote.Symbol == "MSFT");

var subscription = msft.Subscribe(quote => /* … */);

subscription.Dispose();
```

.NET Events와 Observable 비교

.Net Events	Observables
코드 중심	데이터 중심
디자인(XAML)에 표현 가능	디자인(XAML)에 표현 불가
클래스 생성 불가	클래스 생성 가능
구성방법 변경 불가	다양한 구성방법 변경 가능
가벼움	약간 무거움
단단한 실행 모델	Expression Tree로 번역됨

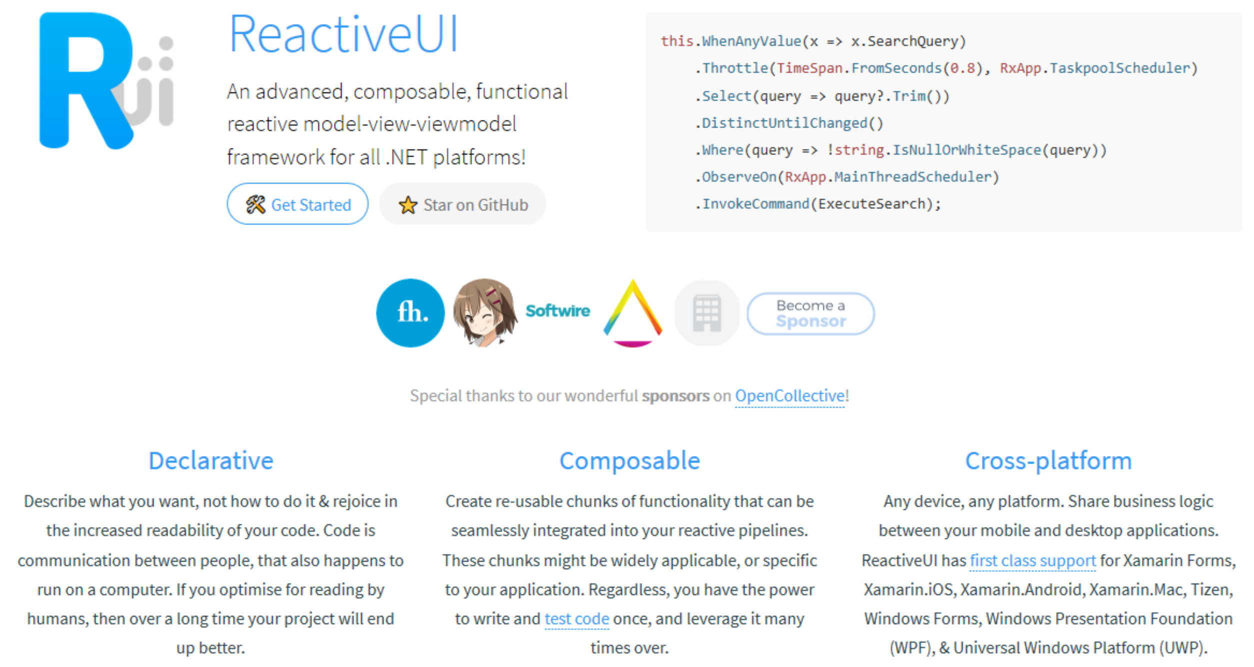
Observable

- IObservable
  - IObserver 프로퍼티
    - OnCompleted
    - OnError
    - OnNext
- IObservable는 한번에 하나의 data를 IObserver로 push
  - Observable sequences



ReactiveUI

- Install ReactiveUI.WPF NuGet package
  - ReactiveUI
  - System.Reactive
  - System.Reactive.Linq



ReactiveUI

An advanced, composable, functional reactive model-view-viewmodel framework for all .NET platforms!

```
this.WhenAnyValue(x => x.SearchQuery)
    .Throttle(TimeSpan.FromSeconds(0.8), RxApp.TaskpoolScheduler)
    .Select(query => query?.Trim())
    .DistinctUntilChanged()
    .Where(query => !string.IsNullOrWhiteSpace(query))
    .ObserveOn(RxApp.MainThreadScheduler)
    .InvokeCommand(ExecuteSearch);
```

Special thanks to our wonderful **sponsors** on OpenCollective!

Declarative

Describe what you want, not how to do it & rejoice in the increased readability of your code. Code is communication between people, that also happens to run on a computer. If you optimise for reading by humans, then over a long time your project will end up better.

Composable

Create re-usable chunks of functionality that can be seamlessly integrated into your reactive pipelines. These chunks might be widely applicable, or specific to your application. Regardless, you have the power to write and test code once, and leverage it many times over.

Cross-platform

Any device, any platform. Share business logic between your mobile and desktop applications. ReactiveUI has first class support for Xamarin Forms, Xamarin.iOS, Xamarin.Android, Xamarin.Mac, Tizen, Windows Forms, Windows Presentation Foundation (WPF), & Universal Windows Platform (UWP).

Observable.Range

```
IObservable<int> source = Observable.Range(0, 10);

var subscription = source.Subscribe(
x => Debug.WriteLine($"OnNext:{x}"),
ex => Debug.WriteLine($"OnError:{ex.Message}"),
() => Debug.WriteLine("OnCompleted"));
```

Observable.Generate

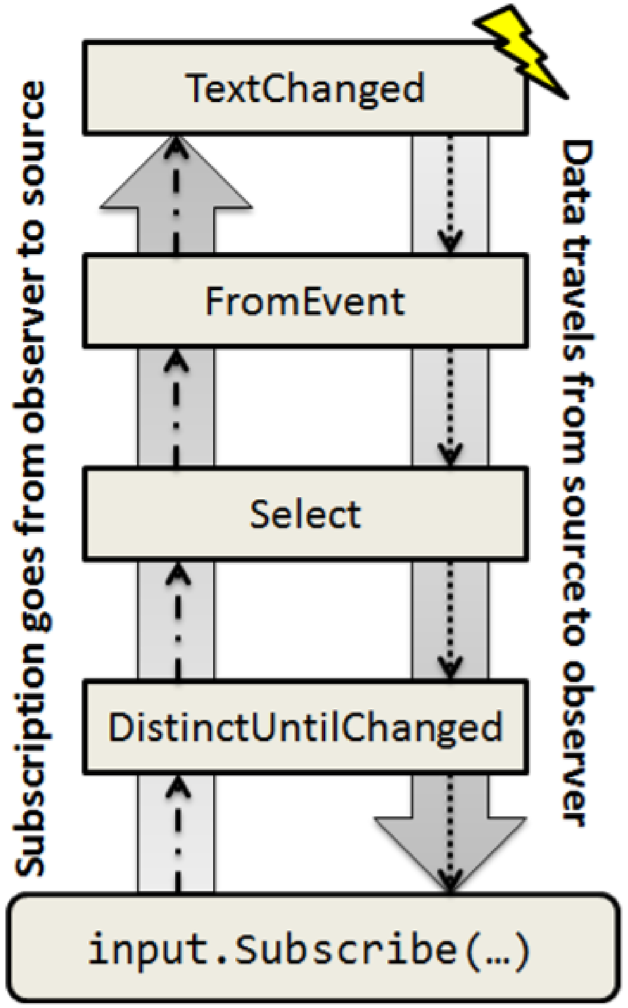
```
IObservable<int> source = Observable
.Generate(0, i => i < 5, i => i + 1, i => i * i, i =>
TimeSpan.FromSeconds(i));

var subscription = source.Subscribe(
x => Debug.WriteLine($"OnNext:{x}"),
ex => Debug.WriteLine($"OnError:{ex.Message}"),
() => Debug.WriteLine("OnCompleted"));
```

Observable.FromEventPattern

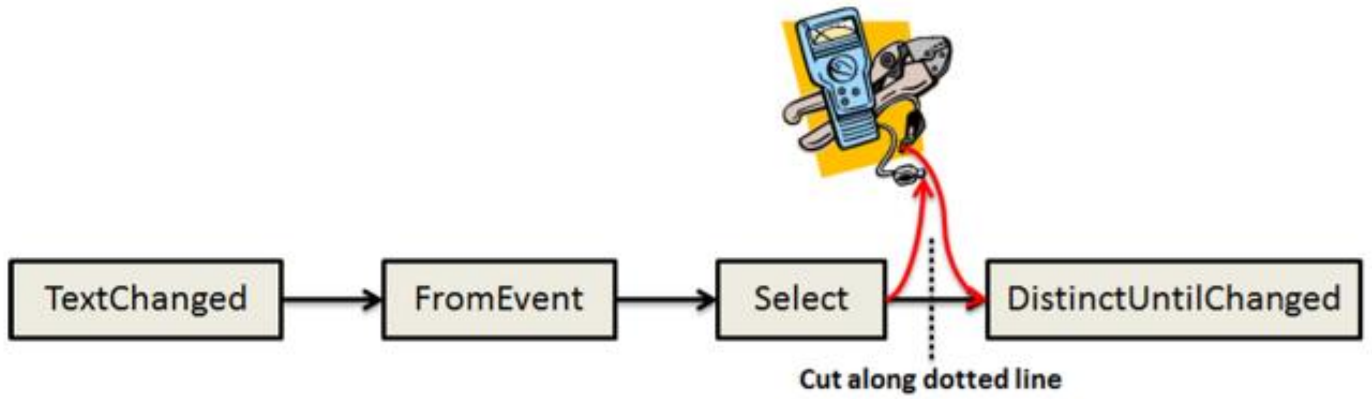
```
var move =
Observable.FromEventPattern<MouseEventArgs>(this,
"MouseMove");
var subscribe = move.Subscribe(
async pattern =>
{
  var position = pattern.EventArgs.GetPosition(this);
  index++;
  var number = _random.Next(0, 10);
  await Task.Delay(TimeSpan.FromSeconds(number));
  ListBox.Items.Add($"{index},
{position.X}:{position.Y}");
});
```

Observable.FromEventPattern

- DistinctUntilChanged()
  - 텍스트가 변경될 때까지 대기
- Throttle(TimeSpan.FromSeconds(1)
  - 지정된 시간 동안 입력이 없으면 다음으로 넘어 갑니다.
- ObserveOnDispatcher()
  - 옵저버 디스패처
- SubscribeOnDispatcher()
  - 서브스크래버 디스패처
- ObserveOn
  - SynchronizationContext.Current
  - 옵저버가 관찰 스케줄러를 지정

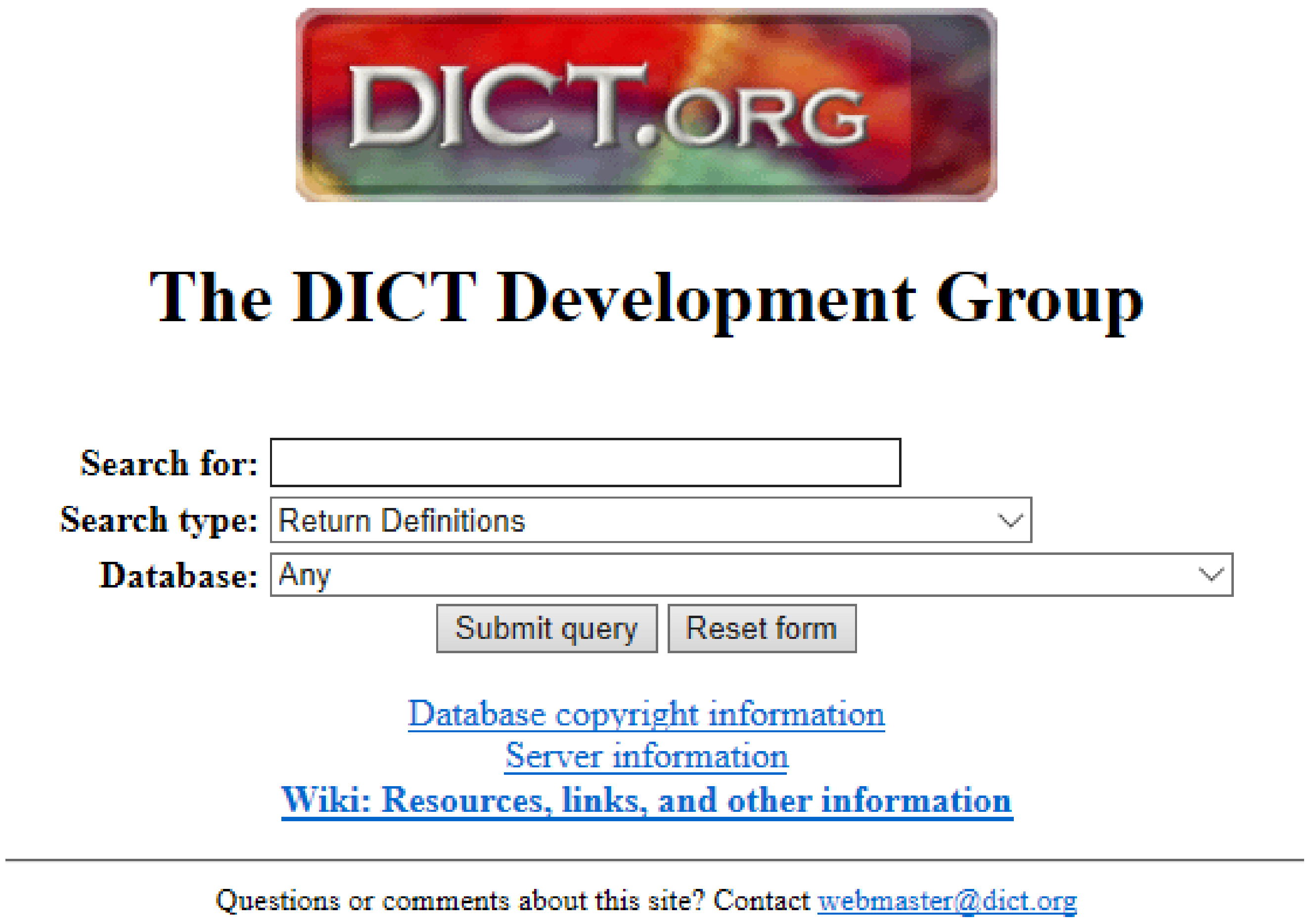

Subscription goes from observer to source
TextChanged
FromEvent
Select
DistinctUntilChanged
input.Subscribe(…)
Data travels from source to observer

```
var input = Observable.FromEventPattern<EventArgs>(InputTextBox,
"TextChanged");
var subscript = input
.Select(p => ((TextBox)p.Sender).Text)
.DistinctUntilChanged()
.Subscribe(
text =>
{
  ListBox.Items.Add(text);
});
```



DICT.org

- The DICT development Group
- http://www.dict.org
- http://services.aonaware.com/DictService/DictService.asmx
- DictServiceSoapClient
  - DictServiceSoap



The DICT Development Group

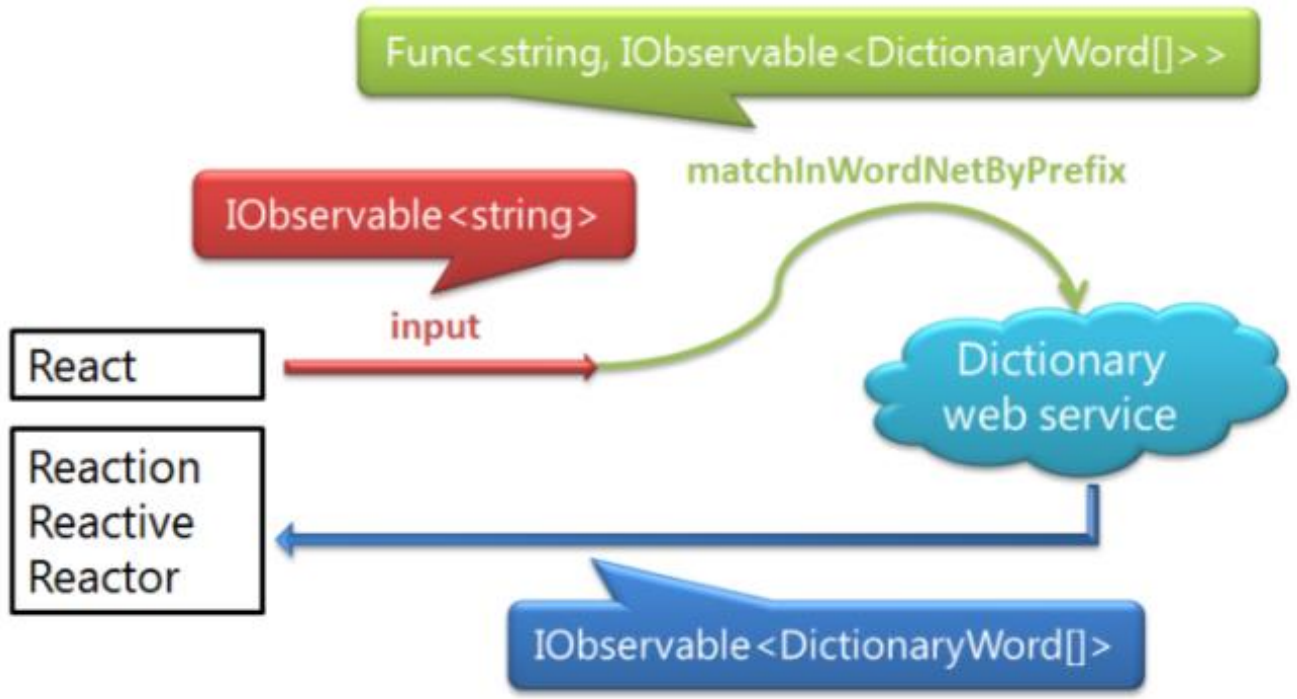
Search for:	
Search type:	Return Definitions
Database:	Any

Database copyright information
Server information
Wiki: Resources, links, and other information

Questions or comments about this site? Contact webmaster@dict.org

DictService1

- 서비스 전체 흐름도



DictService2

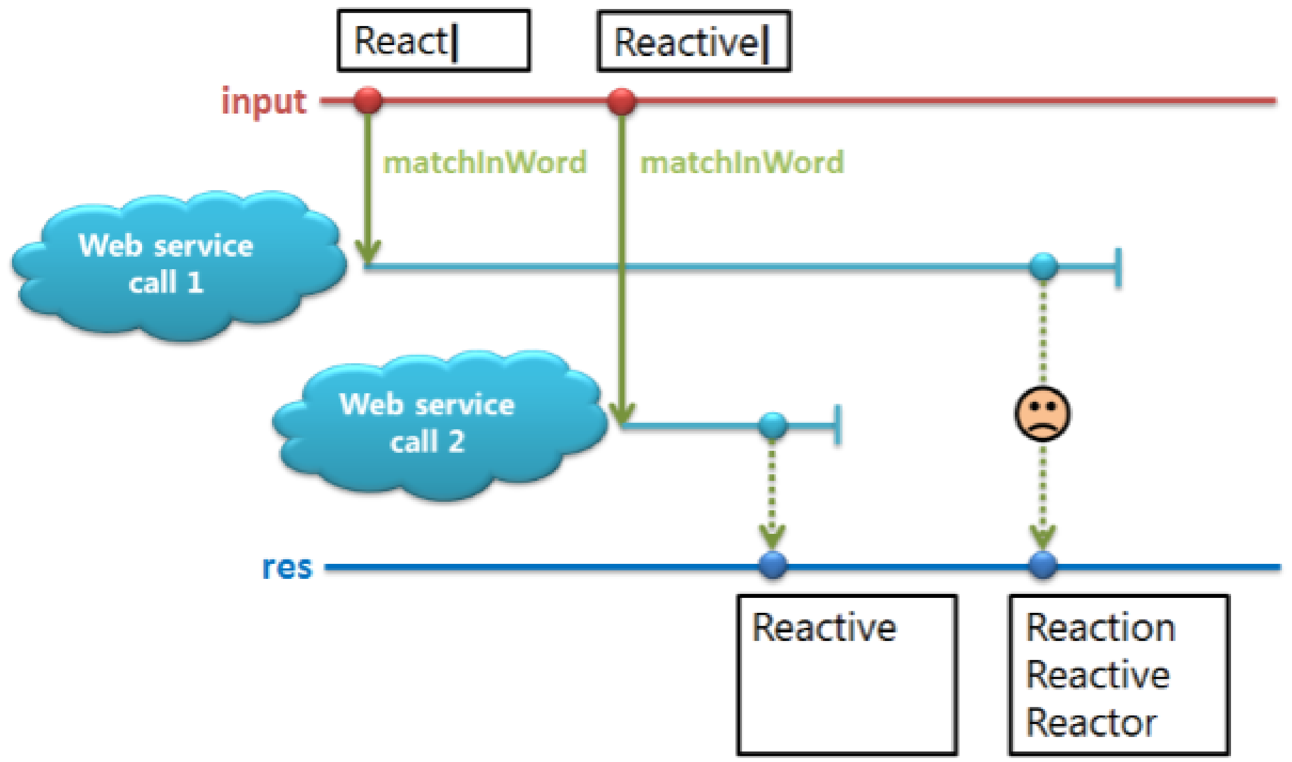
- 입력
- 서비스 호출
- 결과 anonymous type

```
var input =
Observable.FromEventPattern<EventArgs>(InputTextBox,
"TextChanged")
.Select(p => ((TextBox)p.Sender).Text)
.DistinctUntilChanged()
.Throttle(TimeSpan.FromSeconds(1));
```

```
var results = from text in input
from words in service.MatchInDictAsync("wn", text,
"prefix")
select new { KeyWord = text, Results = words };
var subscript = results
.ObserveOnDispatcher()
.Subscribe(
result =>
{
  ListBox.Items.Add($"KeyWord : {result.KeyWord}");
  foreach (var r in result.Results)
  {
    ListBox.Items.Add(r.Word);
  }
});
```

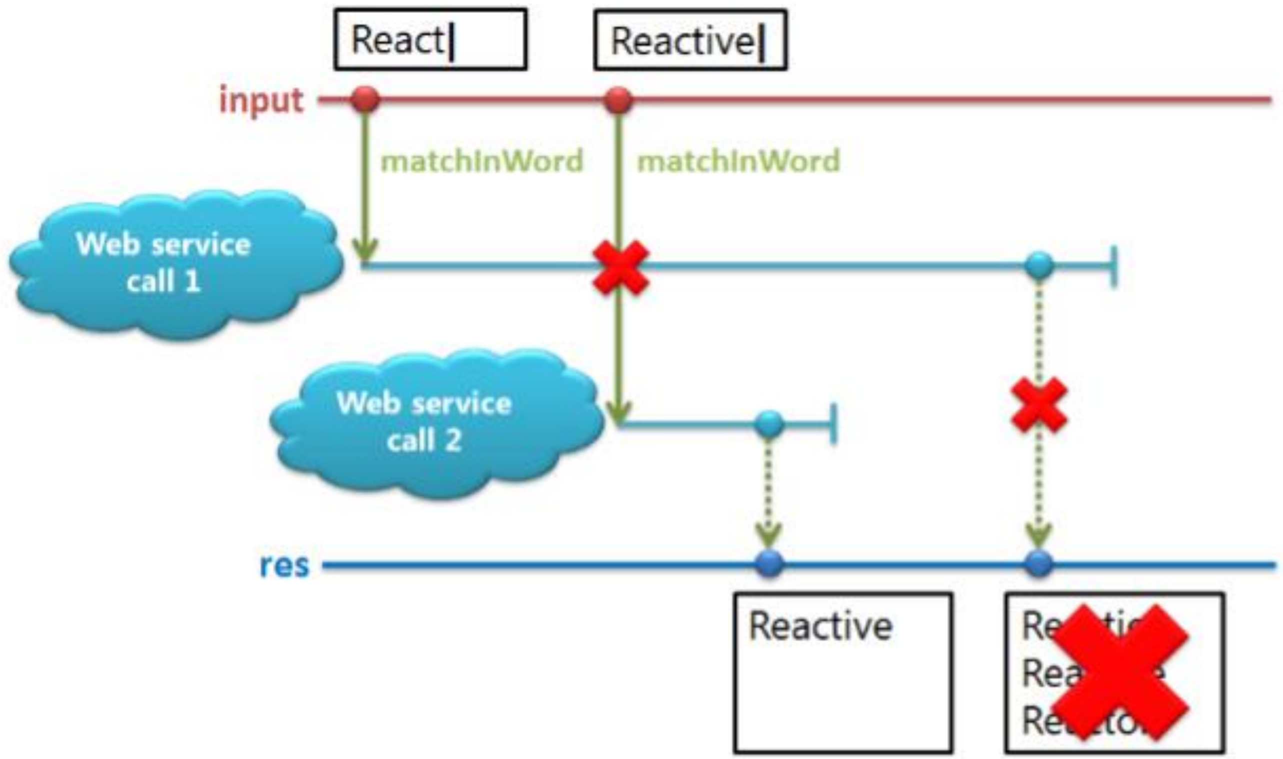
DictService3

- 문제점
  - 먼저 호출된 결과가 나중에 반환



DictService4

- 문제 해결
  - 이미 결과과 도착했으면 나중에 도착한 결과는 무시한다.



DictService5

- System.Reactive.Linq
- ToObservable()
  - Task<T> -> IObservable<T>
- Finally()
  - 시퀀스 종료시 실행
- TakeUntil()
  - 시퀀스와 맞지 안으면 버림

```
var results = from text in input
from words in service
.MatchInDictAsync("wn", text, "prefix")
.ToObservable()
.Finally(() =>
Debug.WriteLine($"Disposed request for {text}"))
.TakeUntil(input)
select new { KeyWord = text, Results = words };
var subscript = results
.ObserveOnDispatcher()
.Subscribe(
result =>
{
  ListBox.Items.Add($"KeyWord : {result.KeyWord}");
  foreach (var r in result.Results)
  {
    ListBox.Items.Add(r.Word);
  }
});
```

Day 5 정리