REACTIVE PROGRAMMING

TAKING THE RED PILL

Pieter Nijs

UNITE POWERED BY ORDINA

NCORE

Welcome

Pieter Nijs

Senior .NET Consultant @ Ordina Belgium
Competence Lead Mobile

Microsoft Extended Experts Team member

Realm MVP

E pieternijs@live.be
T @nijspieter
B blog.pieeatingninjas.be

Welcome

Pieter Nijs

Senior .NET Consultant @ Ordina Belgium
Competence Lead Mobile

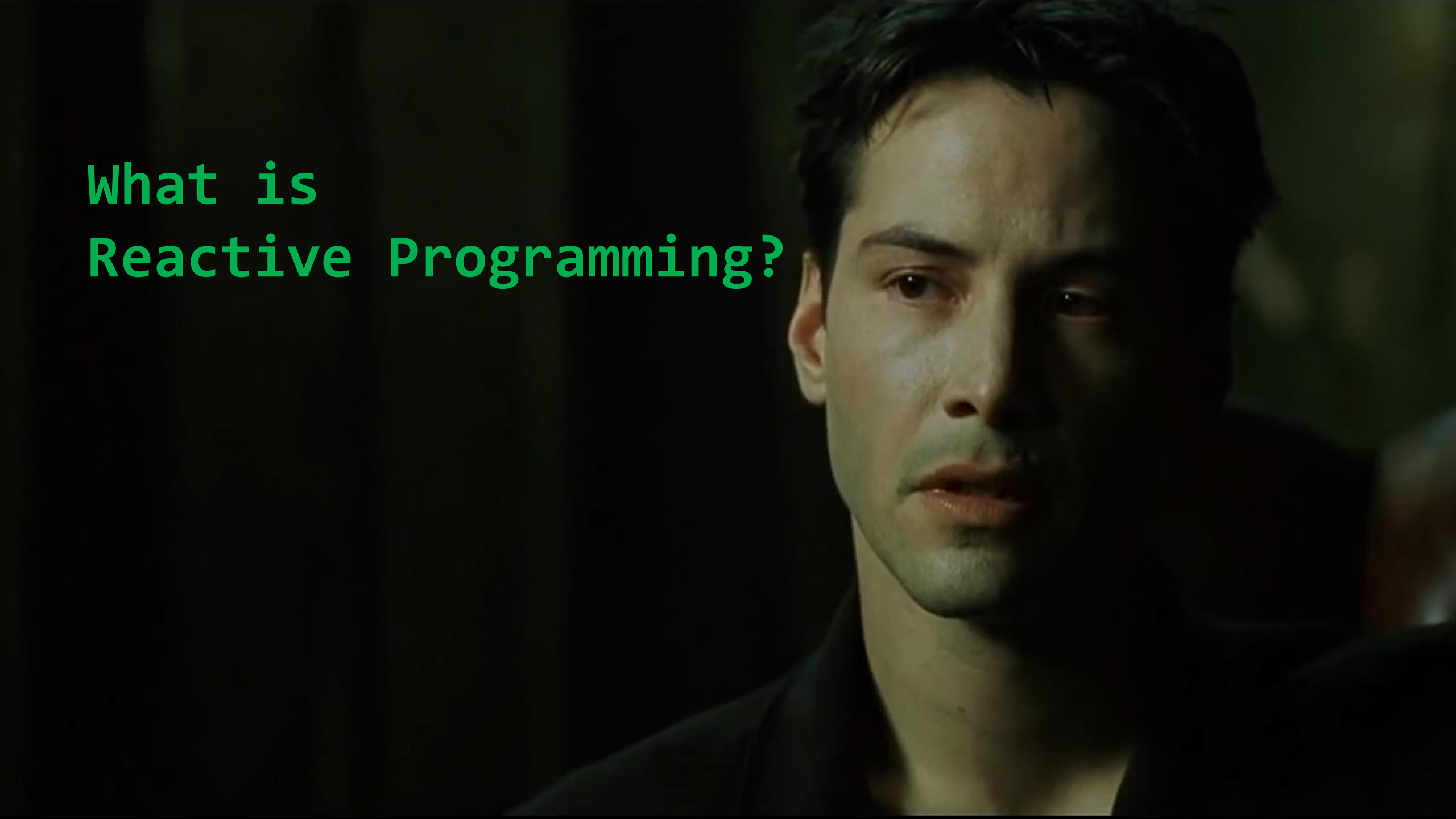
Microsoft Extended Experts Team member

Realm MVP

E pieternijs@live.be
T @nijspieter
B blog.pieeatingninjas.be


Don’t worry about the
difference of opinion

What is Reactive Programming?



Reactive programming

Nothing new

Helps with today’s challenges

Reactive programming

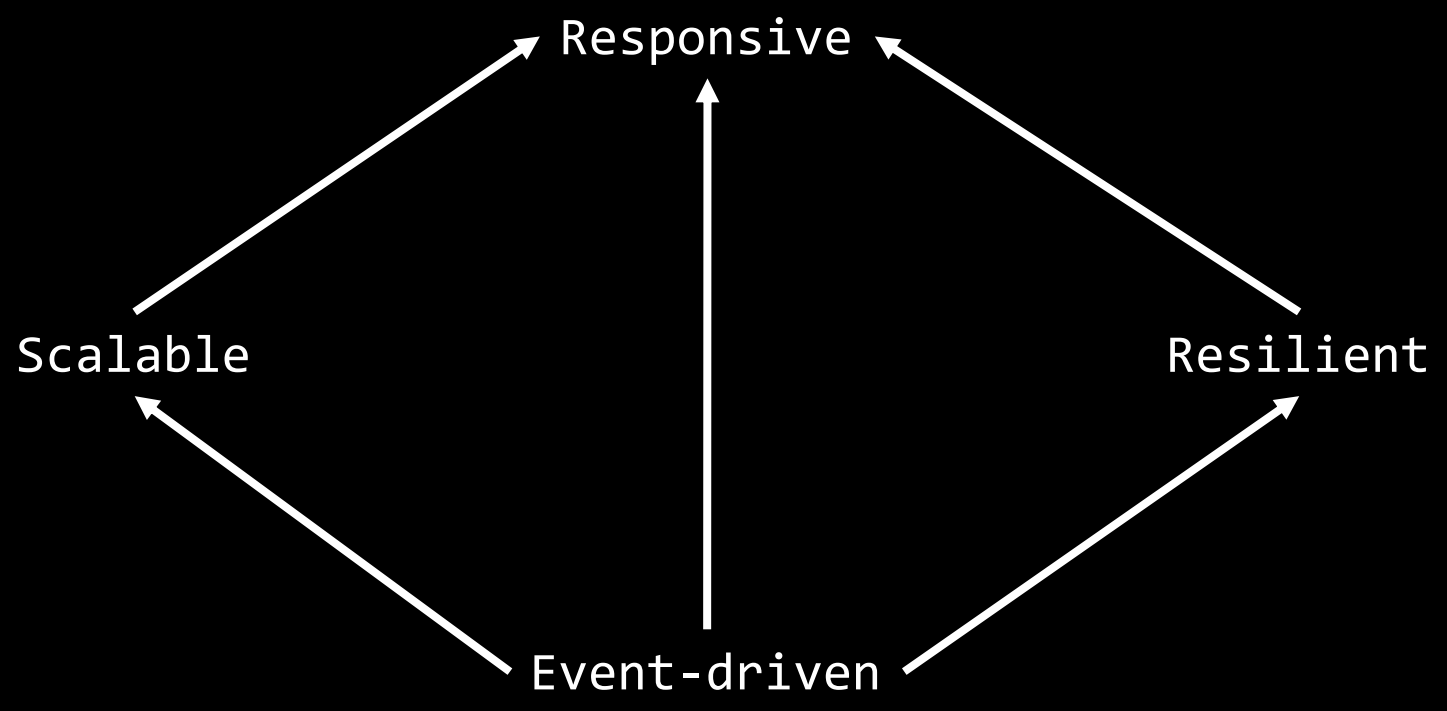
World around us has changed

LOB / internal business applications → Consumer-facing apps

CRUD → Real-time data consumption

PCs / terminals → Smart phones, tablets, TVs, wearables, coffee machines, mirrors,…

Reactive programming



Reactive programming

Responsive

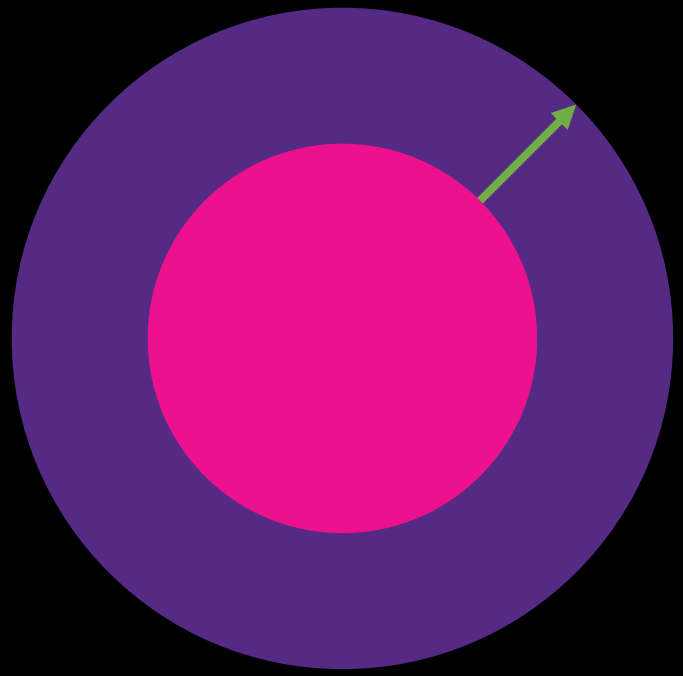
…and lazy

Interactive programming

Program asks the enviroment for something

Programs **PULLS** data from the environment

```
var input = Console.ReadLine();
…
foreach(var item in items) { }
```



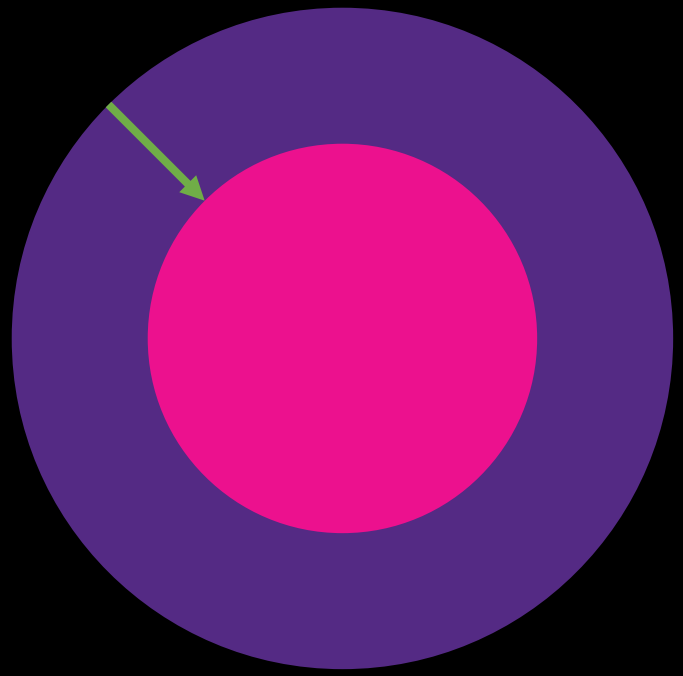
Reactive programming

Environment sends data to the program
Program may need to react to that

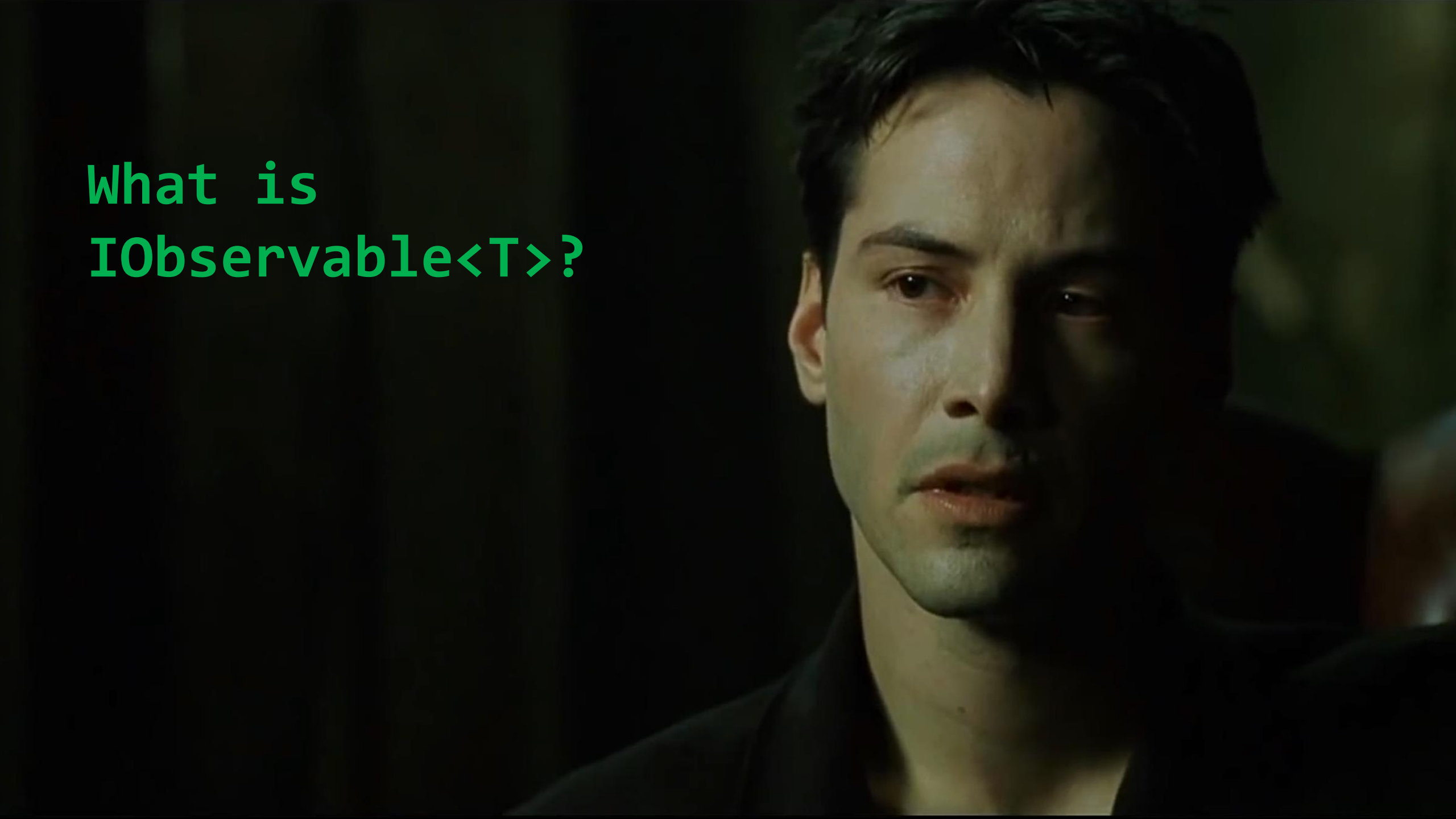
Environment **PUSHES** data to the program

```
okButton.Click += OnButtonClicked;

DoStuffAsync(callbackWhenItsDone);
```



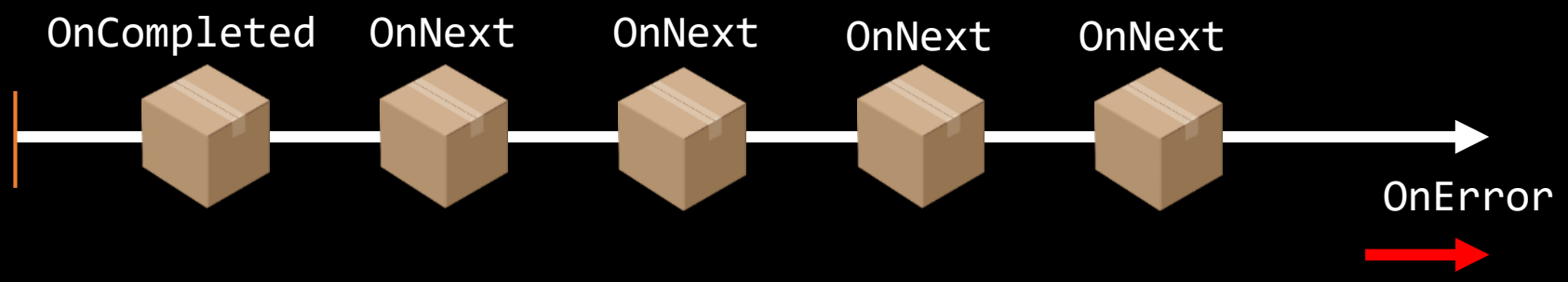
What is IObservable<T>?



“Events are just lists... backwards.”

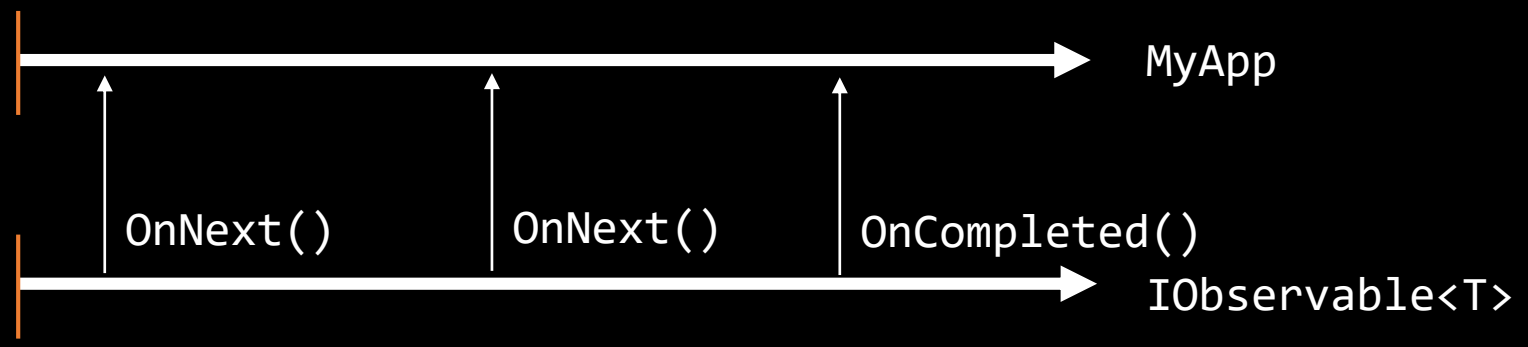
- Paul Betts

IObservable<T>



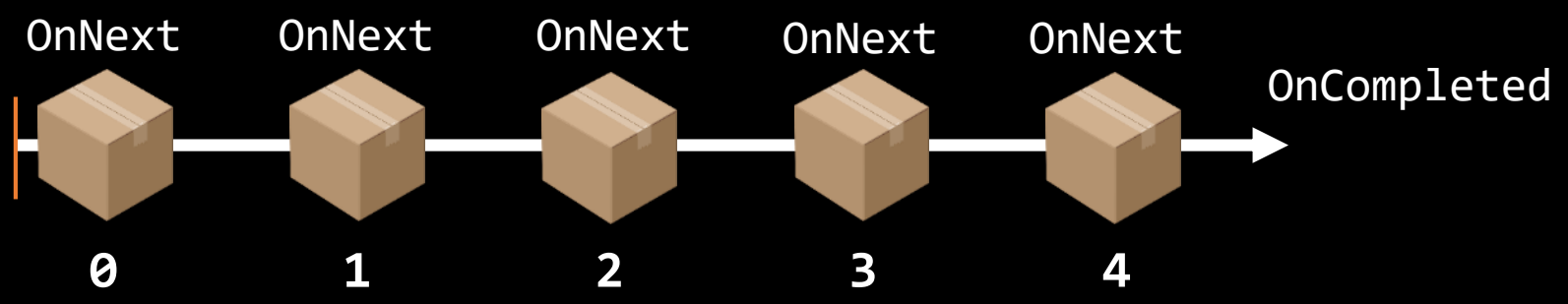
IEnumerable<T>

IObservable<T>



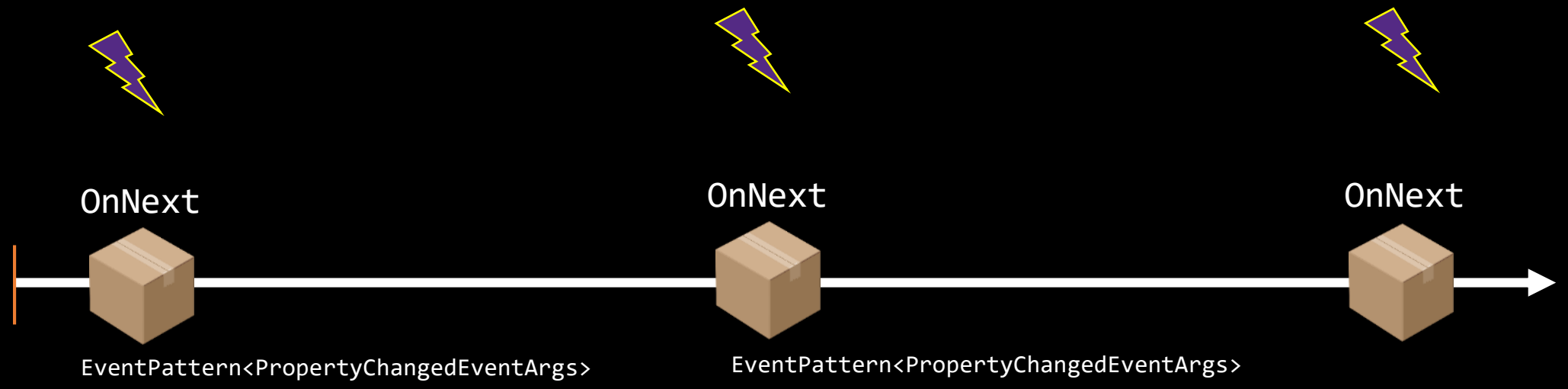
Example Observables

```
Observable.FromRange(0, 5);
```



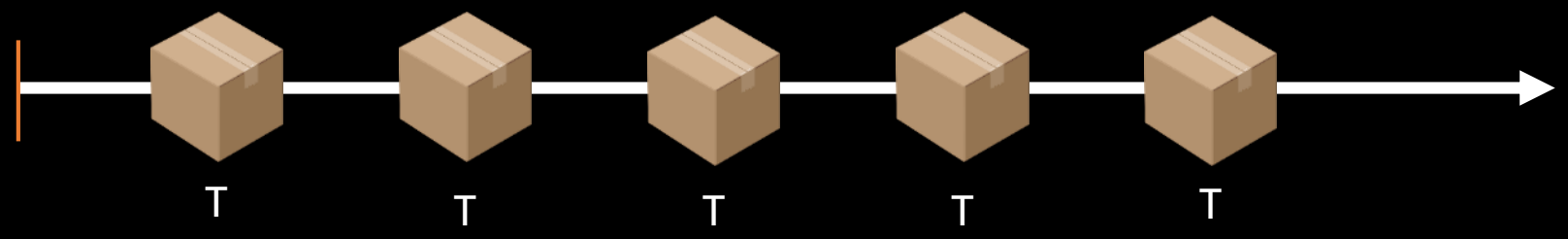
Example Observables

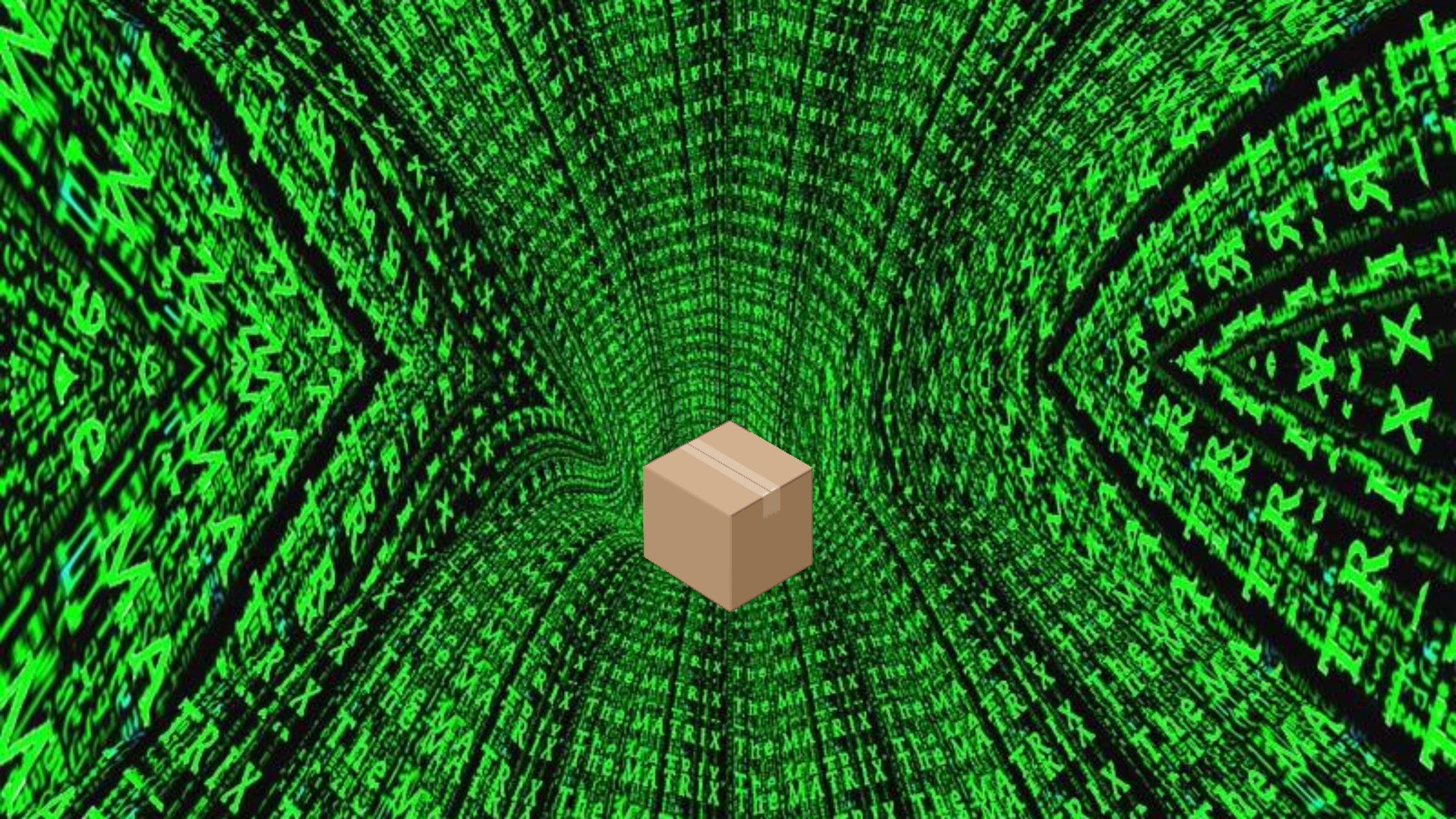
```
Observable.FromEventPattern<PropertyChangedEventHandler,
    PropertyChangedEventArgs>(
                x => PropertyChanged += x,
                x => PropertyChanged -= x);
```

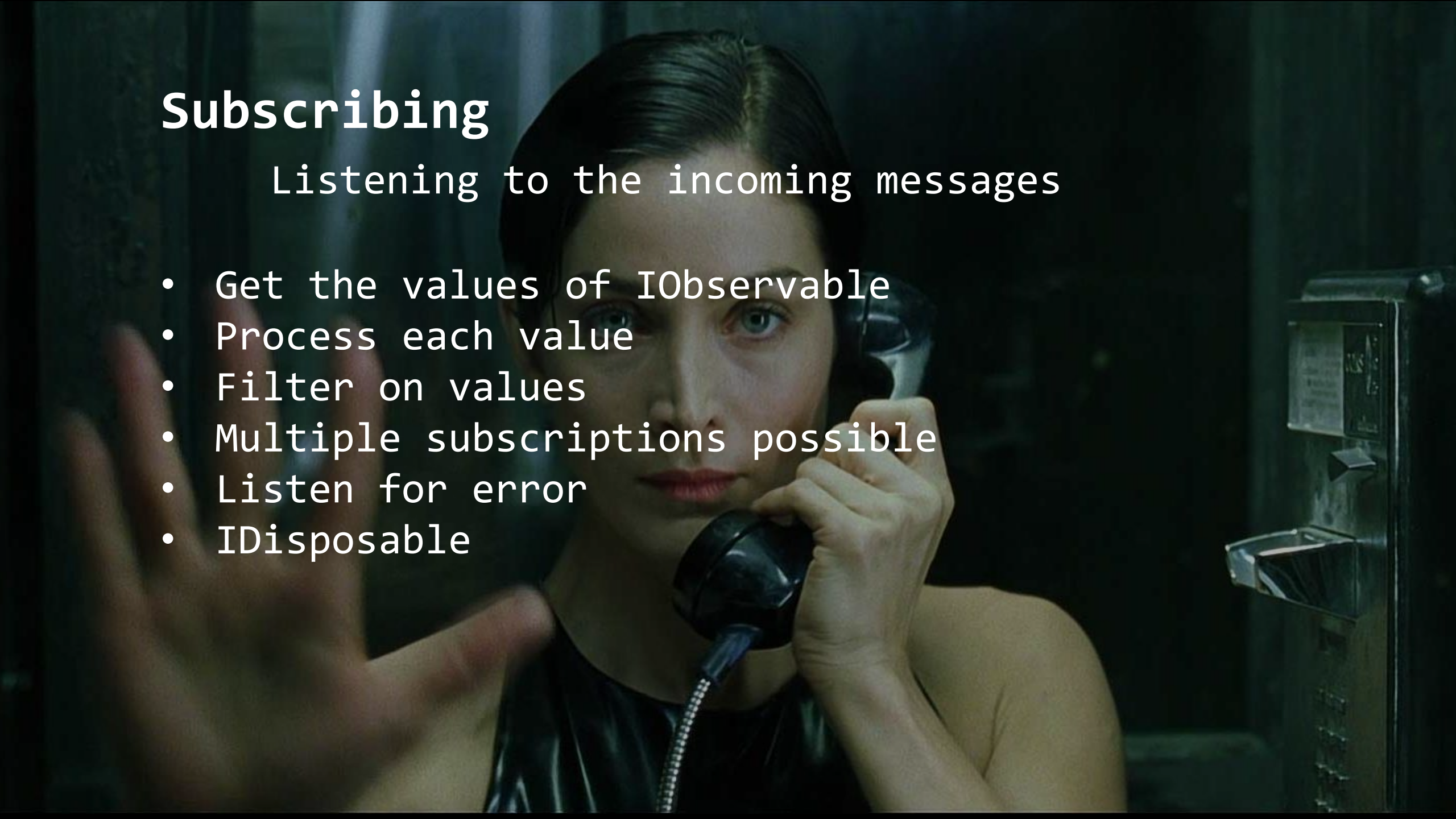


What is an Observer?

IObservable<T>




The MATRIX



Subscribing

Listening to the incoming messages

- Get the values of IObservable
- Process each value
- Filter on values
- Multiple subscriptions possible
- Listen for error
- IDisposable

Subscribing

```
var rangeObservable = Observable.Range(0, 5);

var rangeObserver = rangeObservable.Subscribe(i =>
{
    Debug.WriteLine(i);
});

rangeObserver.Dispose();
```

Subscribing

```
var propChangedObservable =
      Observable.FromEventPattern
            <PropertyChangedEventHandler,
            PropertyChangedEventArgs>(
                  x => PropertyChanged += x,
                  x => PropertyChanged -= x);

var propChangedObserver =
      propChangedObservable.Subscribe(propchange =>
{
      Debug.WriteLine(
         $"Prop'{propchange.EventArgs.PropertyName}' changed.");
});
```

What does it look like in code?


Convinced?


“Honestly,
I don’t know…”

Who loves
functional programming?


LINQ

Why do we love LINQ so much?

```
var names = new string[] { "Pieter", "Stefanie", "Bruce" };

var query = names.Where(filterOnName).OrderBy(n => n).Select(n => n.ToUpper());

foreach (var item in query)
{
    Console.WriteLine(item);
}
```

Abstraction

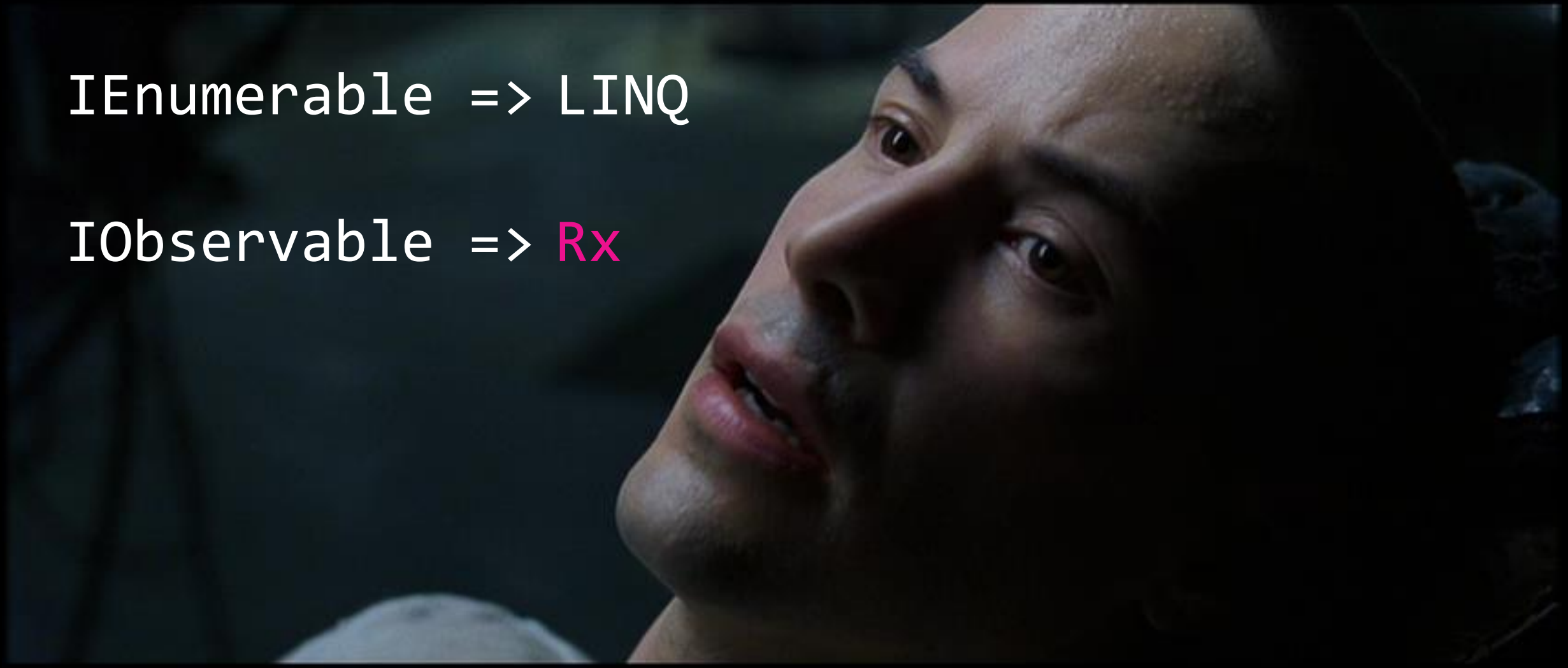
Immutability

Lazy evaluation

Function pipeline

IEnumerable => LINQ

IObservable => Rx



Reactive Extensions (Rx)

Created by Microsoft in 2009

RxJS (2010)
RxJava (2012)
RxCpp (2012)
RxRuby (2012)
RxScala (2013)
RxPHP (2013)
RxSwift (2015)

Reactive Extensions (Rx)

Select

Sample

Take

Retry

Throttle

Buffer

Delay

Where

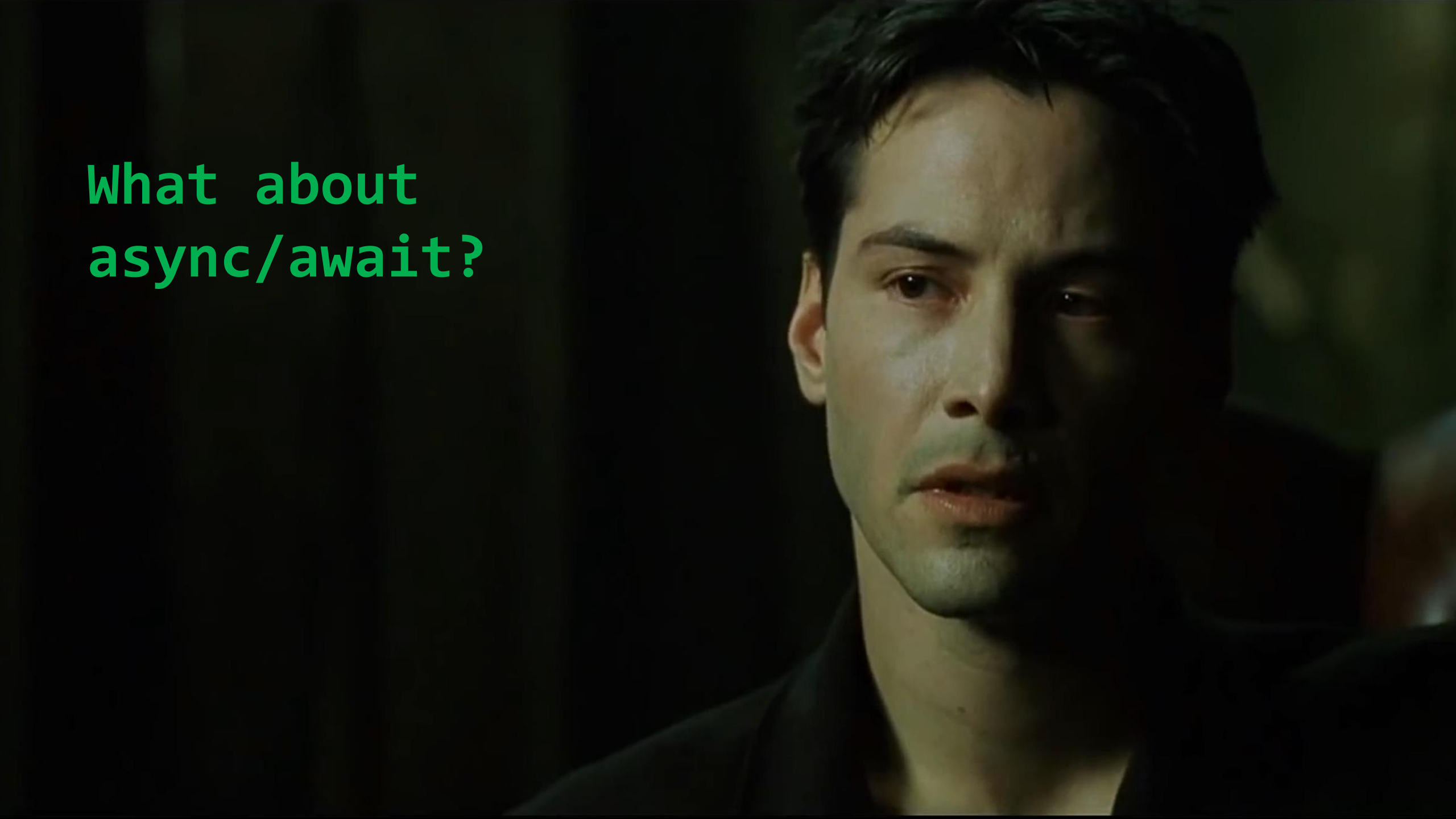
Aggregate

CombineLatest

DistinctUntilChanged

What does it look like in code?

What about
async/await?




What if I told you
async/await is
compatible with Rx

Async/await

```
Observable.FromAsync(() => DoRequestAsync(url))

            .Timeout(TimeSpan.FromMilliseconds(200))
            .Retry(3)
            .OnErrorResumeNext(
                 Observable.FromAsync(
                      () => DoRequestAsync2(url)))
```

- Async/await ++

Async/await

You can `await` an Observable!

```
var result = await observable;

string result = await observable.FirstAsync();

string result = await observable.FirstOrDefaultAsync();

string result = await observable.LastAsync();
```

What does it look like in code?

But…
I like MVVM…


What if I told you
MVVM and Rx can work
perfectly together?

DIY
INotifyPropertyChanged

INotifyPropertyChanged

```
var propChangedObservable =
      Observable.FromEventPattern
            <PropertyChangedEventHandler,
            PropertyChangedEventArgs>(
                  x => PropertyChanged += x,
                  x => PropertyChanged -= x);

var propChangedObserver =
      propChangedObservable.Subscribe(propchange =>
{
      ...
});
```

ReactiveUI

```
public class MyViewModel : ReactiveObject
```

- Reactive Bindings
- ReactiveCommands
- PropertyChanged observation with WhenAnyValue()

What does it look like in code?

Conclusion

Reactive programming

- Event-driven architecture
  - Resilient
  - Scalable
  - Responsive

- Events
  - User interaction
  - Data flowing-in through streams

- Not the silver bullet
  - Use the right tool for the job!

Thank you

Pieter Nijs

Senior .NET Consultant @ Ordina Belgium
Competence Lead Mobile

Microsoft Extended Experts Team member

Realm MVP

E pieternijs@live.be
T @nijspieter
B blog.pieeatingninjas.be

Thank you

Pieter Nijs

Senior .NET Consultant @ Ordina Belgium
Competence Lead Mobile

Microsoft Extended Experts Team member

Realm MVP

E pieternijs@live.be
T @nijspieter
B blog.pieeatingninjas.be