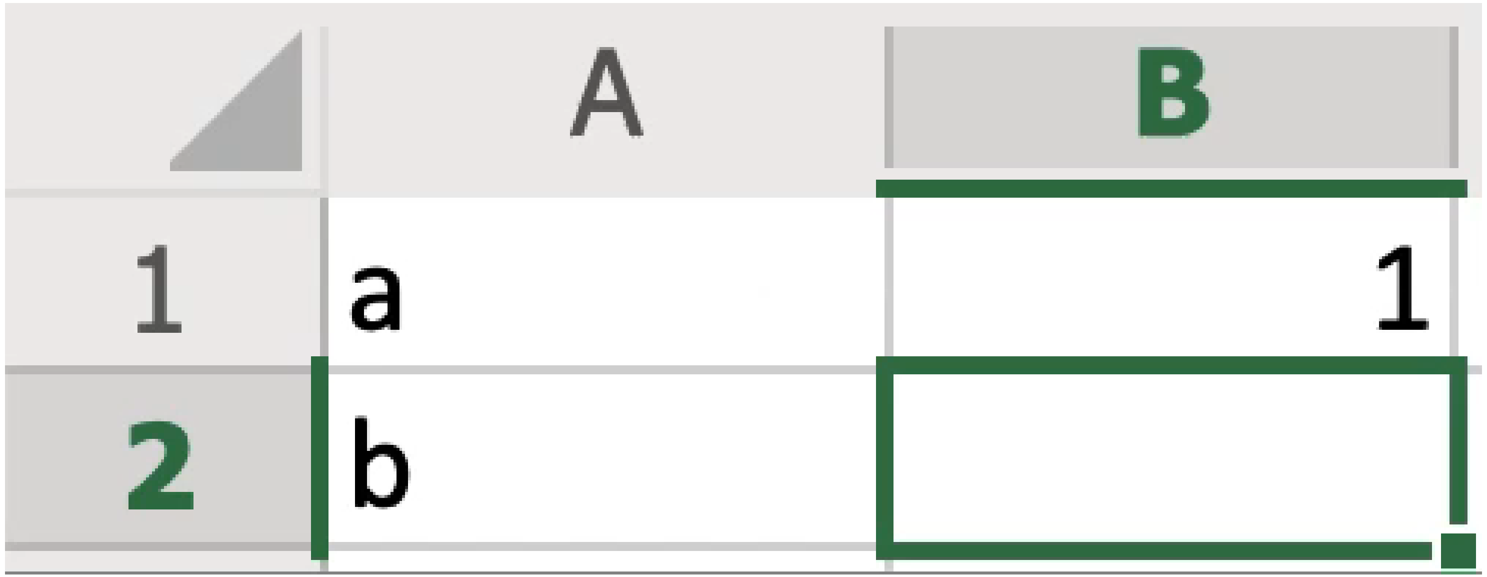
Реактивное программирование и его применение в frontend разработке (WPF)

Рогожин Владимир

What is reactive programming?



	A	B
1	a	1
2	b	

Disclaimer

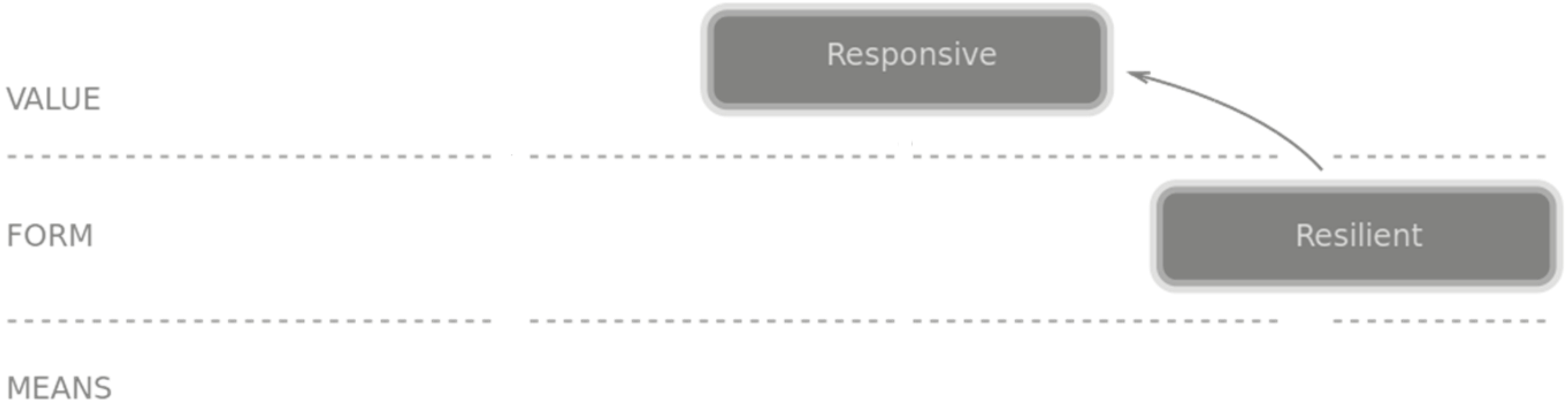
From reactive programming
To reactive systems

Reactive manifesto

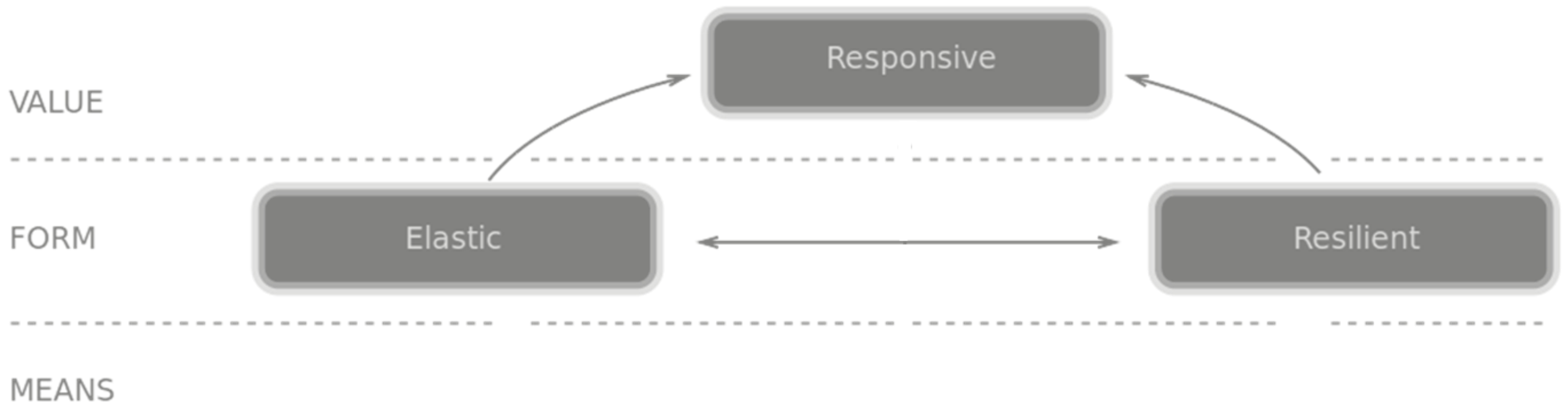
Reactive manifesto

Responsive
VALUE
FORM
MEANS

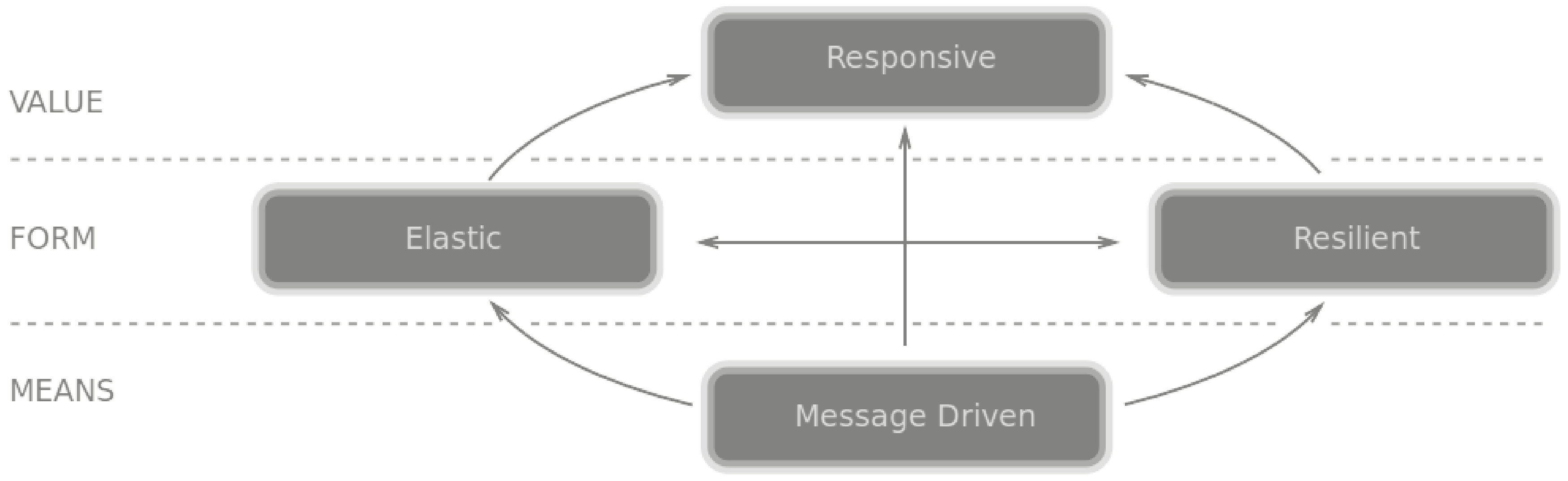
Reactive manifesto

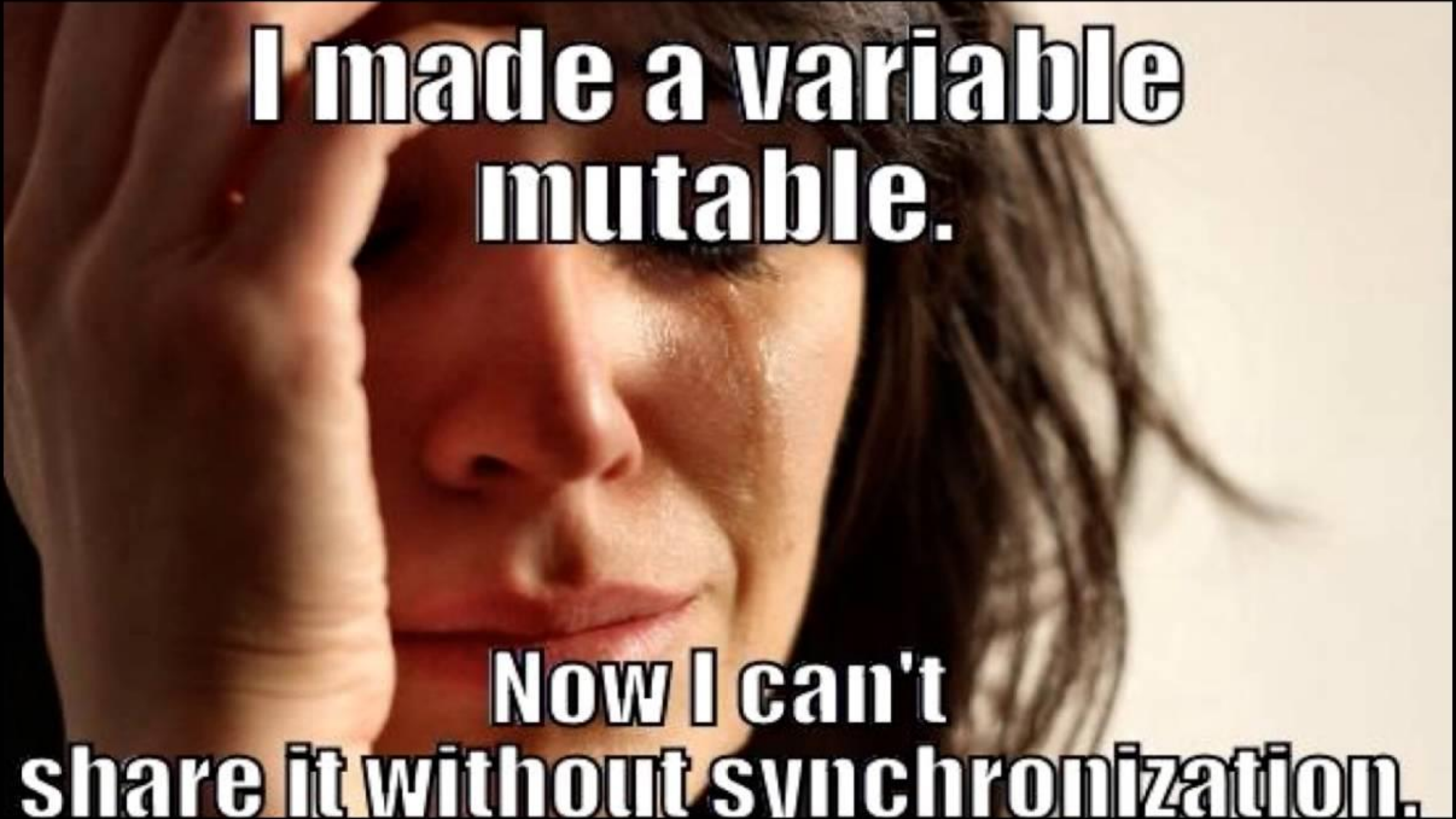


Reactive manifesto



Reactive manifesto




I made a variable mutable.
Now I can't share it without synchronization.

Tools and libraries

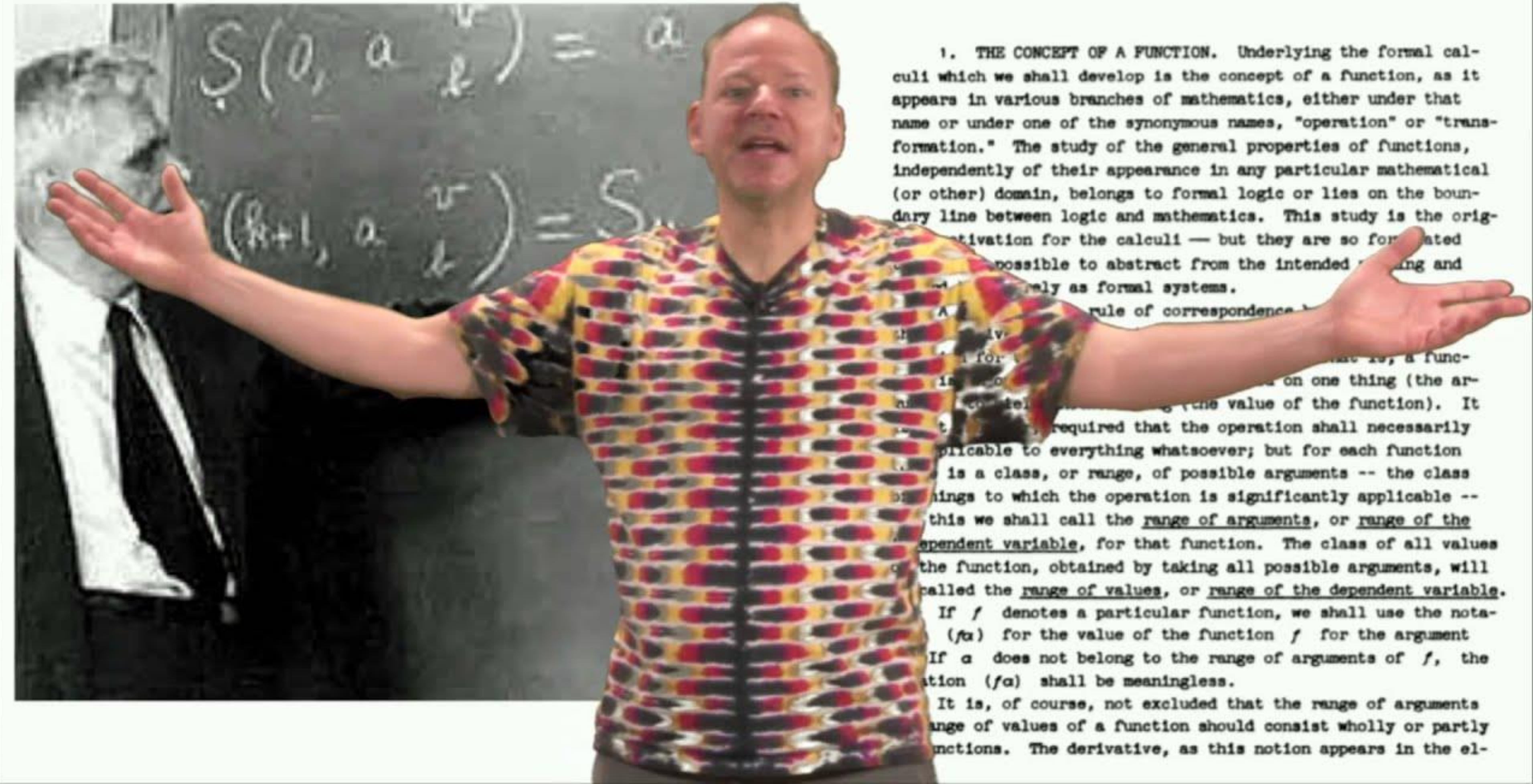
Libraries

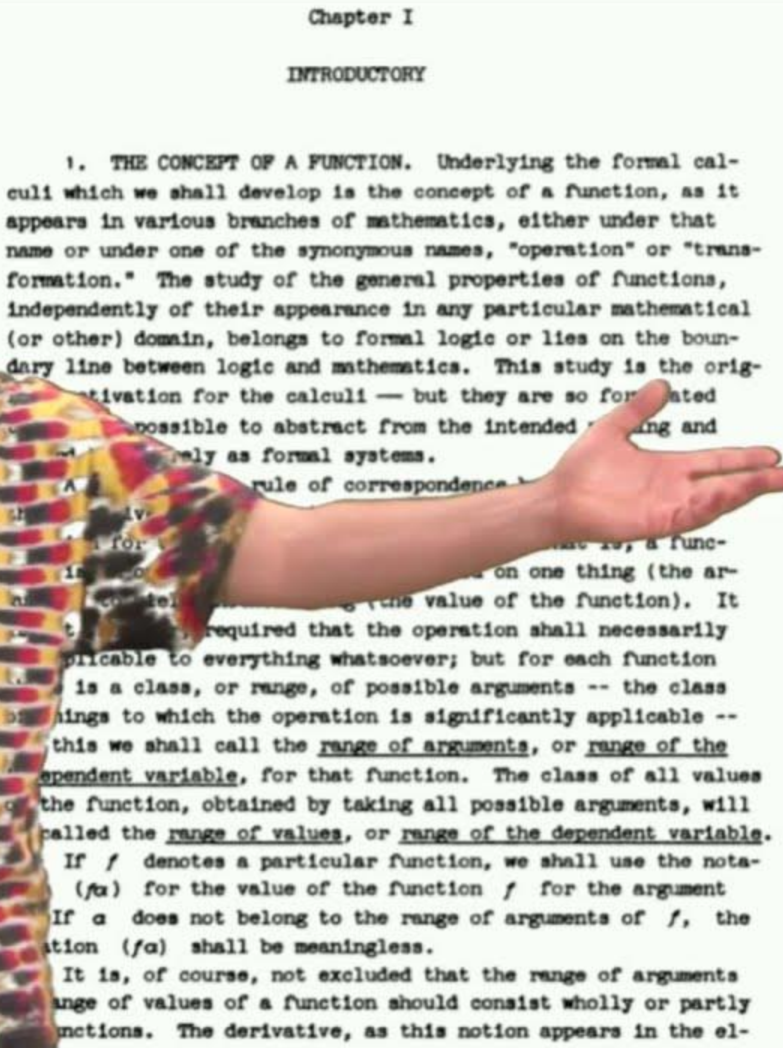
- Akka.Net (+ Akka.Streams)
- Microsoft Orleans
- protoactor-dotnet
- Reactive Streams
- Foundatio
- Obvs
- MassTransit
- Reactive extensions

Reactive extensions

ReactiveX

An API for asynchronous programming
with observable streams


$S(0, a \overset{v}{\underset{k}{}}) = a$
$S(k+1, a \overset{v}{\underset{k}{}}) = S$



Chapter I

INTRODUCTORY

1. THE CONCEPT OF A FUNCTION. Underlying the formal calculi which we shall develop is the concept of a function, as it appears in various branches of mathematics, either under that name or under one of the synonymous names, "operation" or "transformation." The study of the general properties of functions, independently of their appearance in any particular mathematical (or other) domain, belongs to formal logic or lies on the boundary line between logic and mathematics. This study is the orig- [illegible]tivation for the calculi — but they are so for[illegible]ated [illegible] possible to abstract from the intended [illegible]ing and [illegible]ely as formal systems.

A [illegible] rule of correspondence [illegible] [illegible] [illegible] for [illegible] that is, a func- [illegible] is [illegible] on one thing (the ar- [illegible] the value of the function). It [illegible] required that the operation shall necessarily [illegible]plicable to everything whatsoever; but for each function [illegible] is a class, or range, of possible arguments -- the class [illegible] things to which the operation is significantly applicable -- this we shall call the *range of arguments*, or *range of the* [illegible]*ependent variable*, for that function. The class of all values [illegible] the function, obtained by taking all possible arguments, will [illegible]called the *range of values*, or *range of the dependent variable*.

If f denotes a particular function, we shall use the nota- [illegible] $(f\alpha)$ for the value of the function f for the argument [illegible] If α does not belong to the range of arguments of f, the [illegible]ation $(f\alpha)$ shall be meaningless.

It is, of course, not excluded that the range of arguments [illegible]ange of values of a function should consist wholly or partly [illegible]nctions. The derivative, as this notion appears in the el-

Lua

C++

Scala

JS

Groovy

Java™

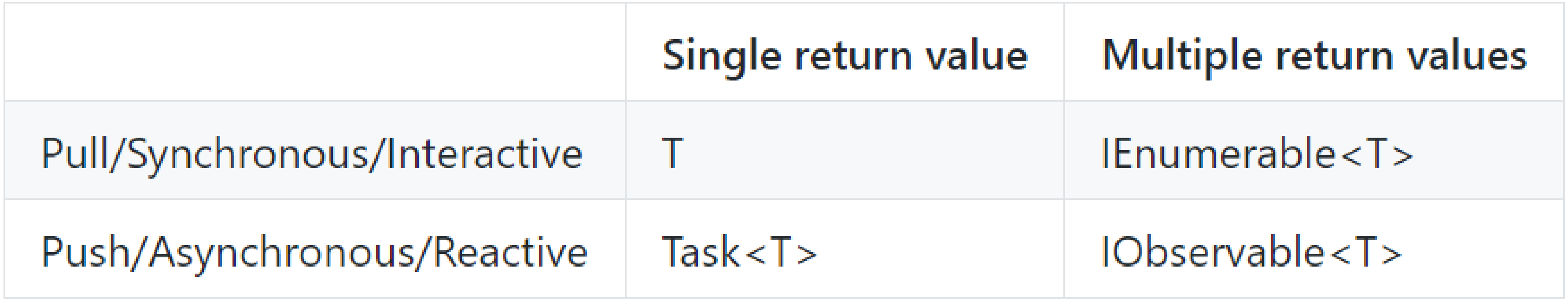
C#

Dart

elixir

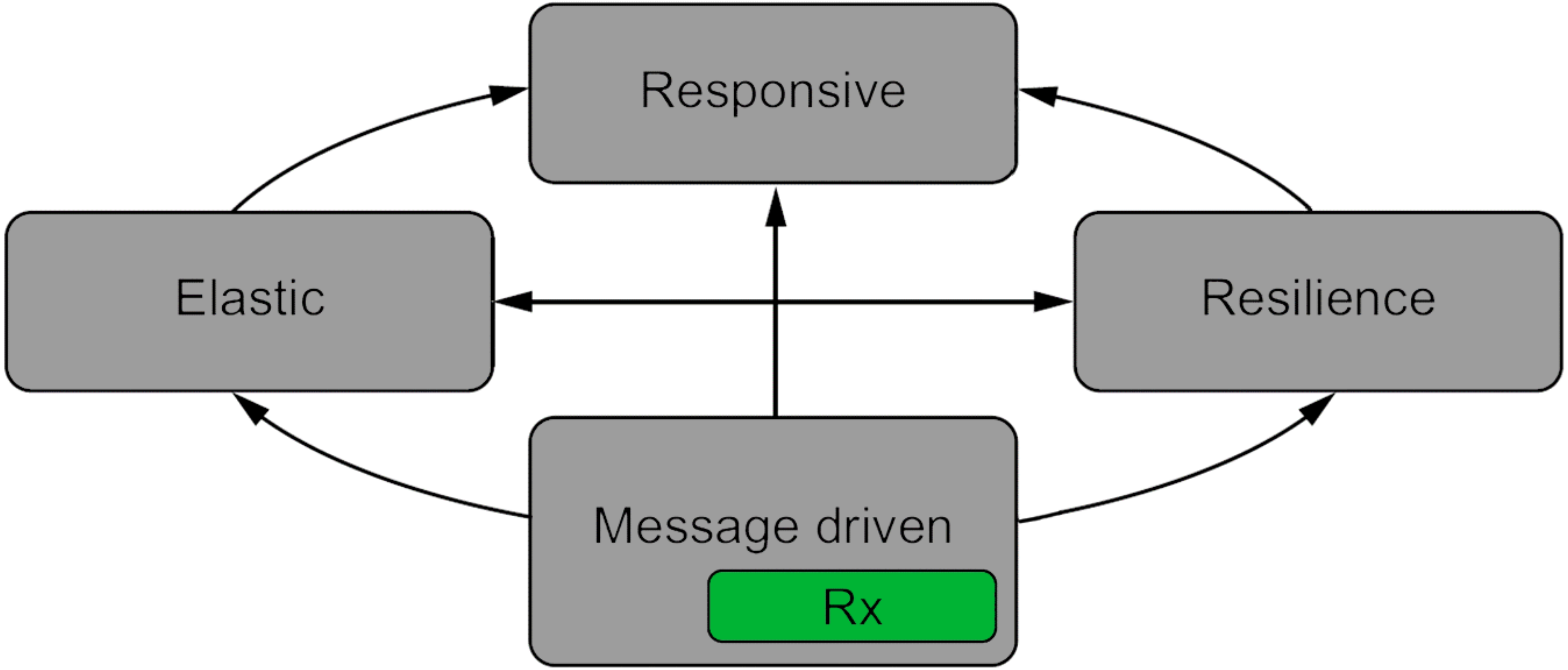
php

RX in C#



	Single return value	Multiple return values
Pull/Synchronous/Interactive	T	IEnumerable<T>
Push/Asynchronous/Reactive	Task<T>	IObservable<T>

RX and Reactive manifesto



Rx = Observables + LINQ + Schedulers

Observables

- Observable - коллекция значений, упорядоченная во времени

- Observer - можно представить как коллекцию callback’ов, которая умеет реагировать на изменения в потоке

- Subject - observable и observer в одном лице.

Interface IObservable<out T>

IDisposable Subscribe(IObserver<T>)

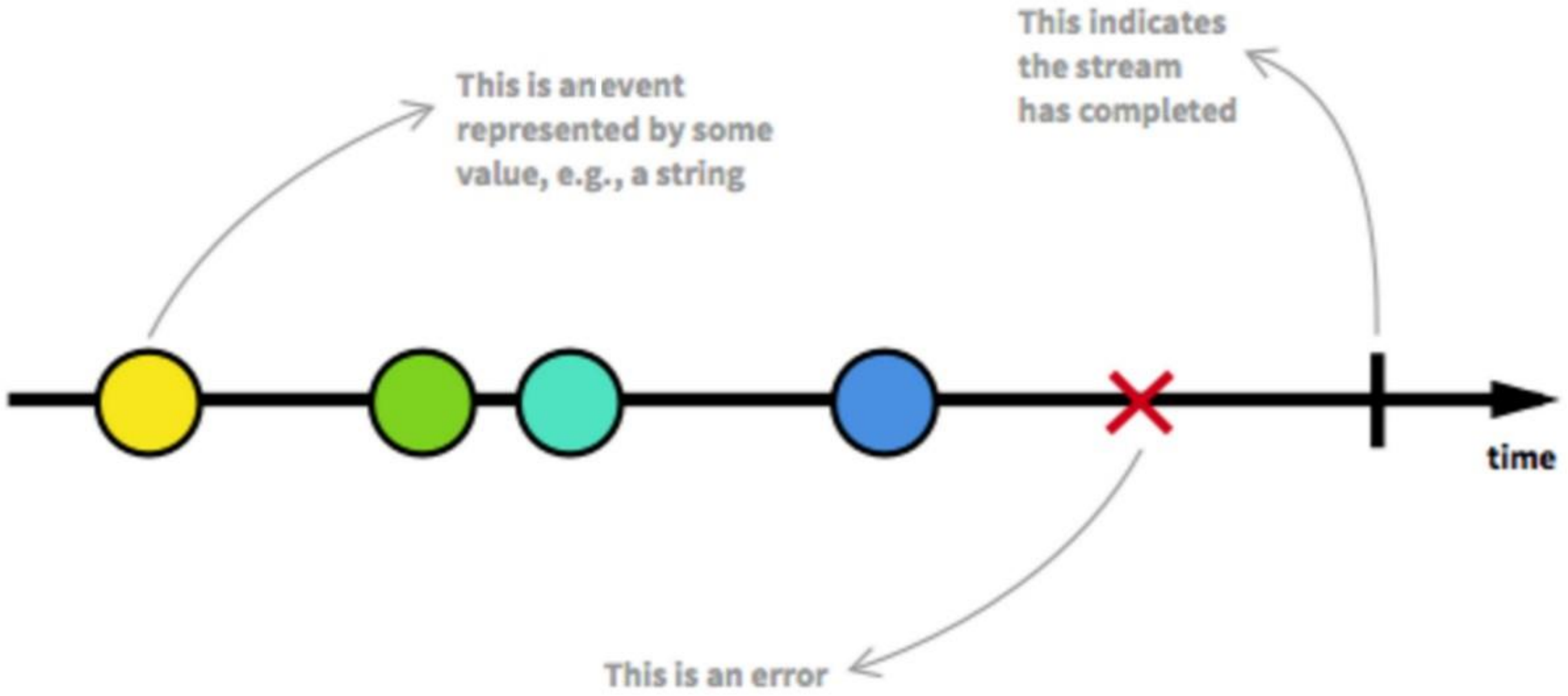
Interface IObserver<T>

OnNext(T)

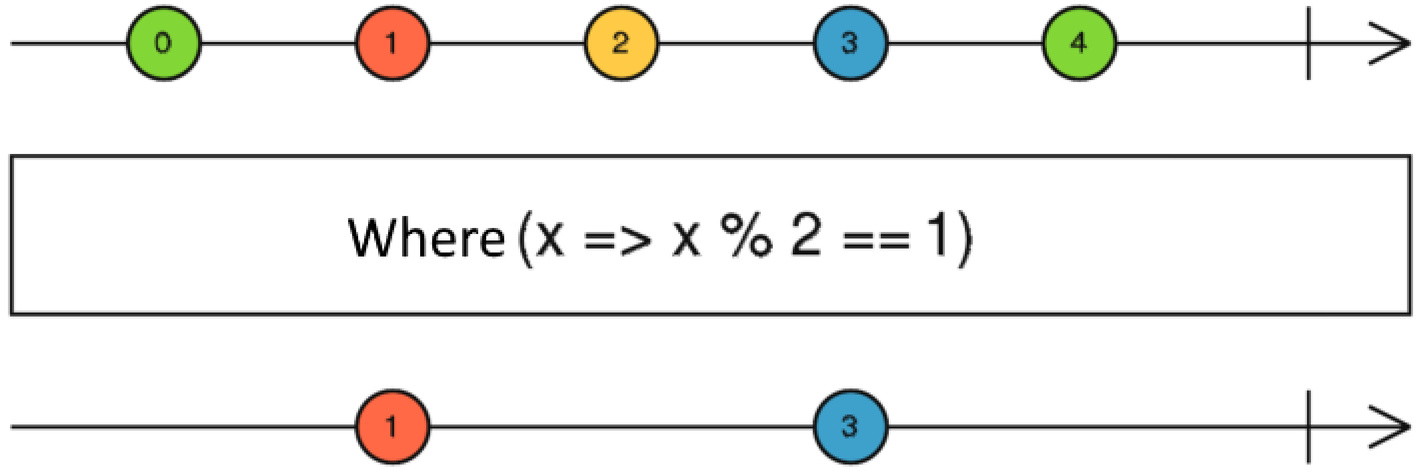
OnError(Exception)

OnCompleted()

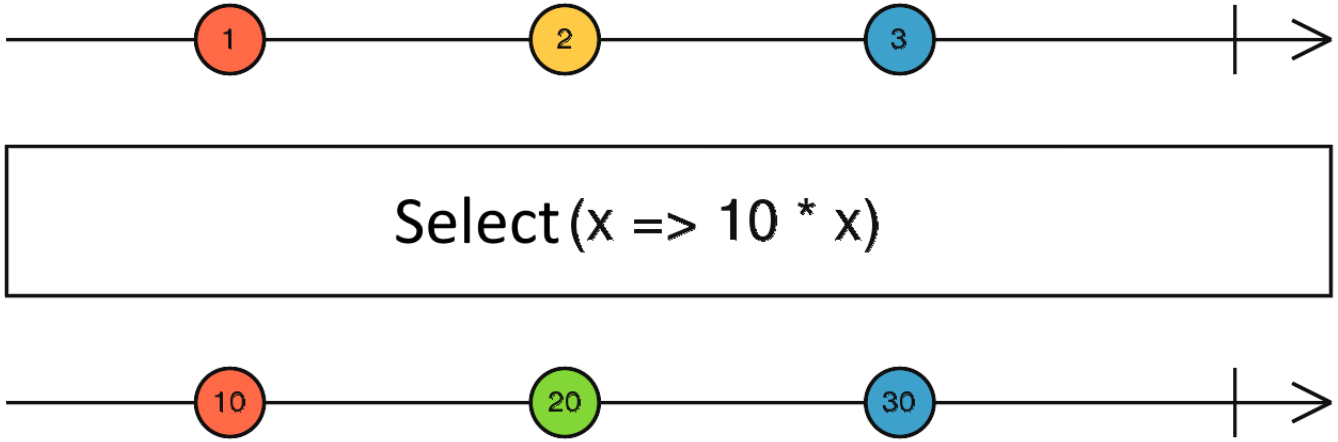
Marble diagrams



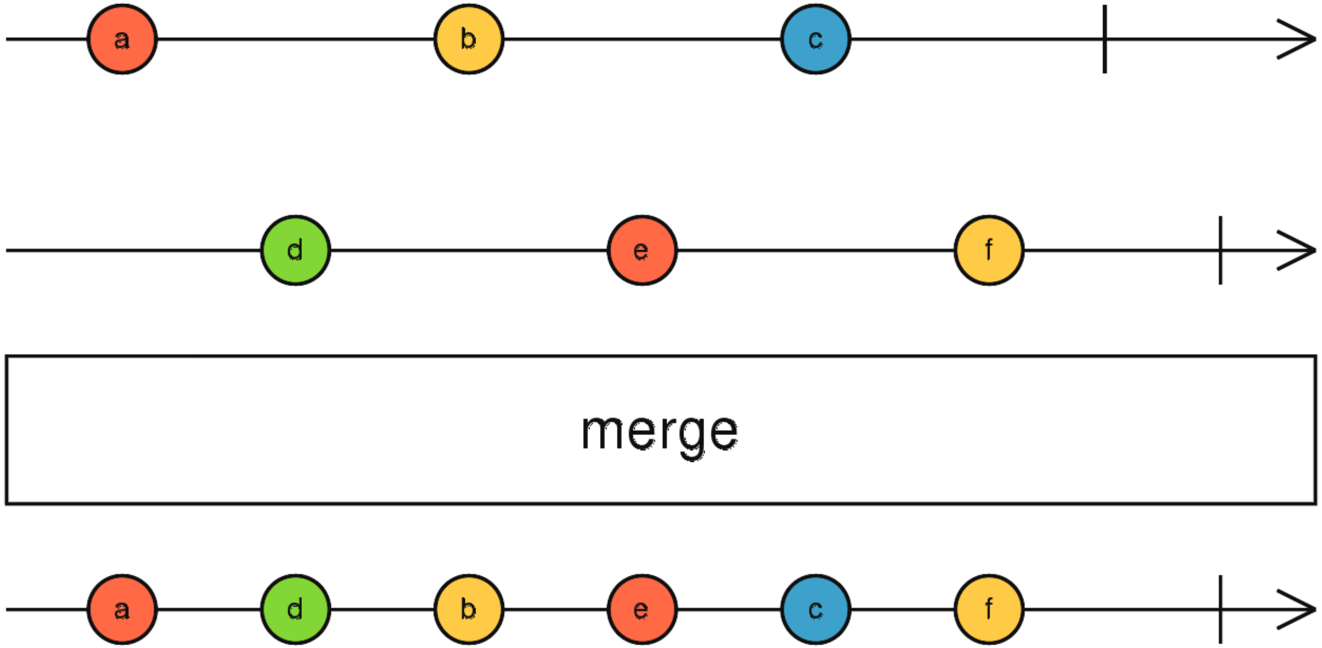
LINQ



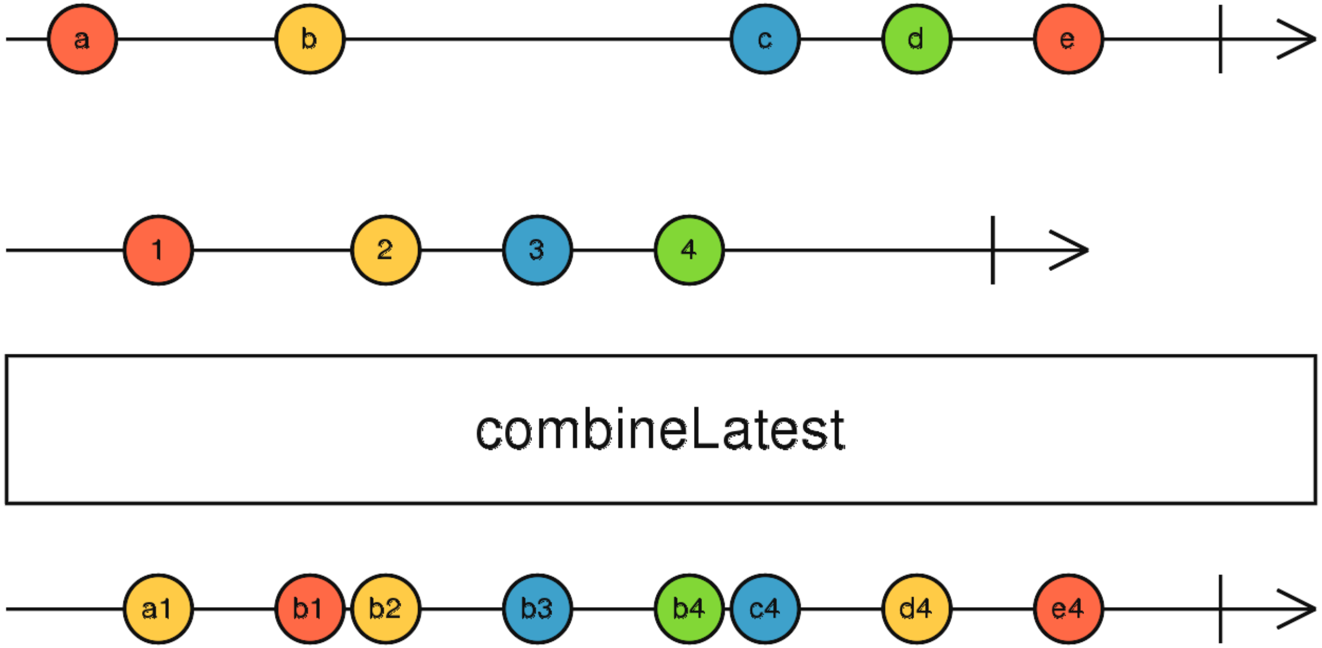
LINQ



LINQ



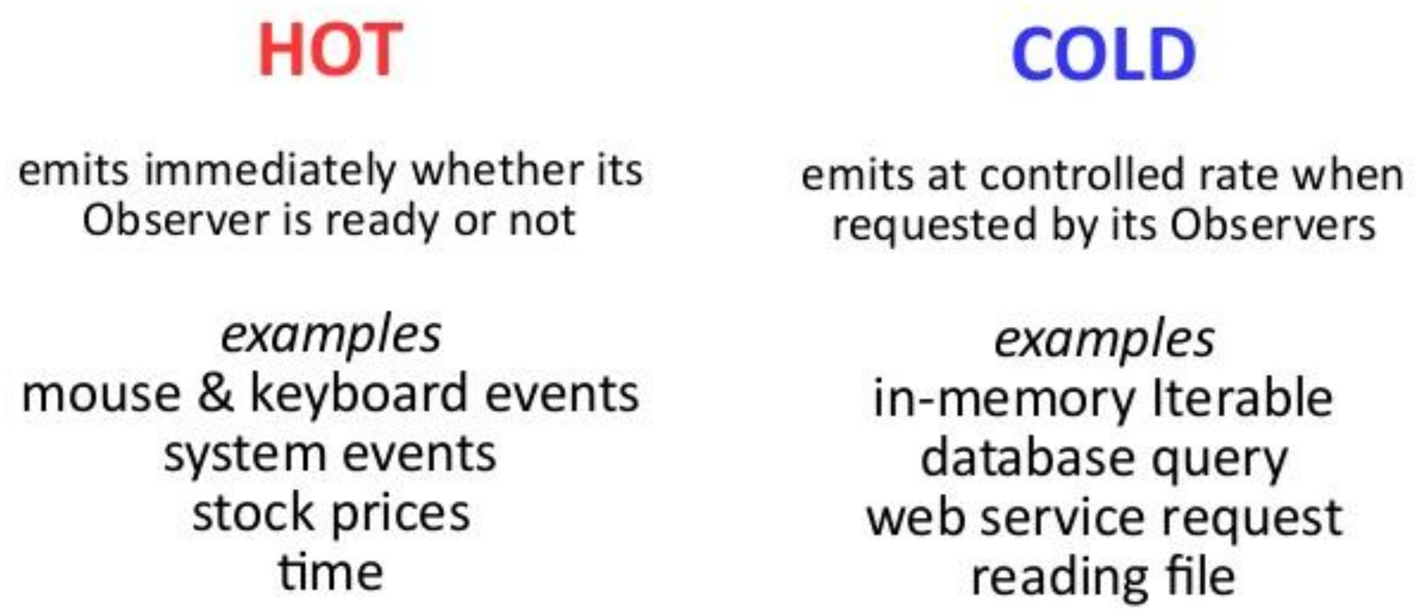
LINQ



Scheduler

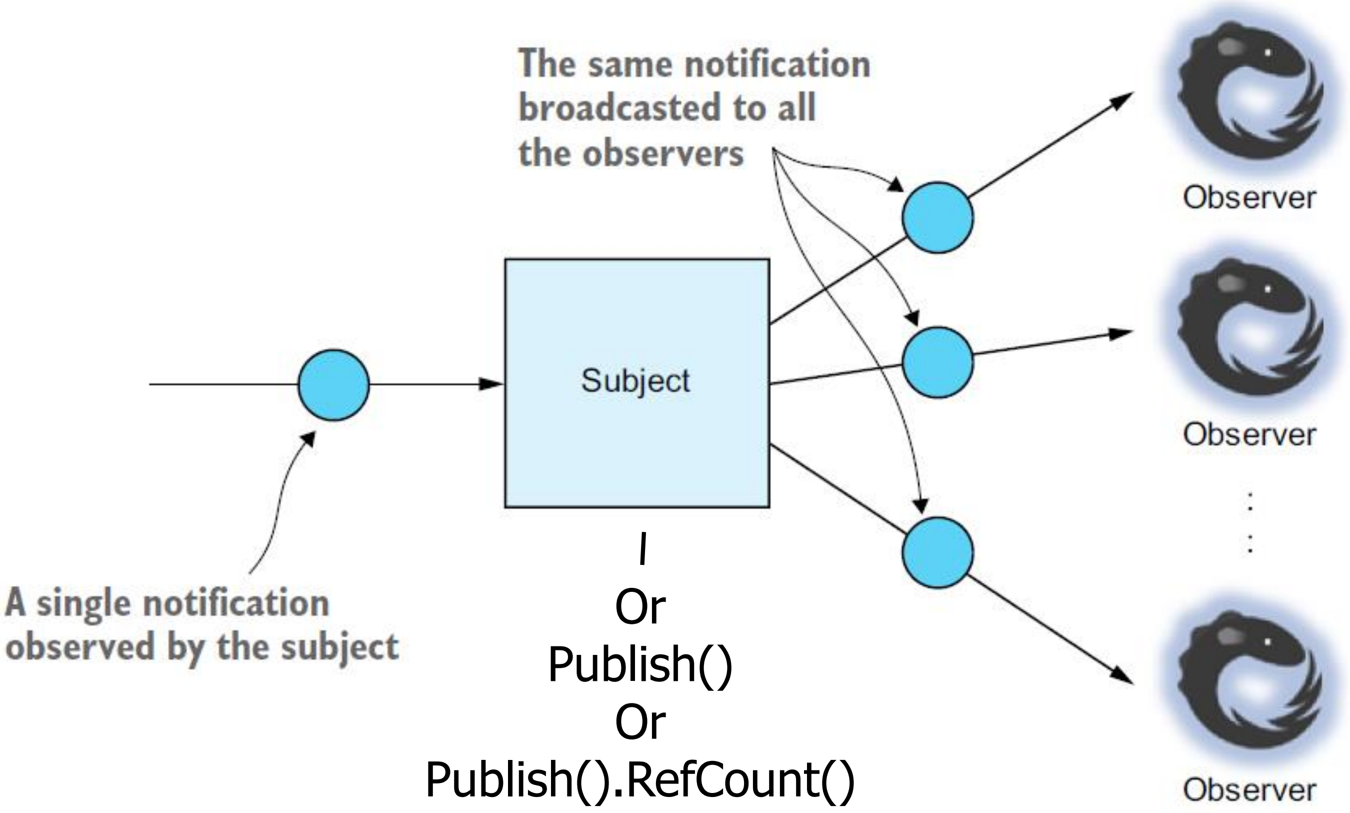
- Callbacks will run on same thread by default
- Schedulers' exist:
  - On task-pool
  - On thread-pool
  - On new thread (each time)
  - On specific thread
  - Custom

Hot and cold observables



HOT	COLD
emits immediately whether its Observer is ready or not	emits at controlled rate when requested by its Observers
examples	*examples*
mouse & keyboard events	in-memory Iterable
system events	database query
stock prices	web service request
time	reading file

Multicasting

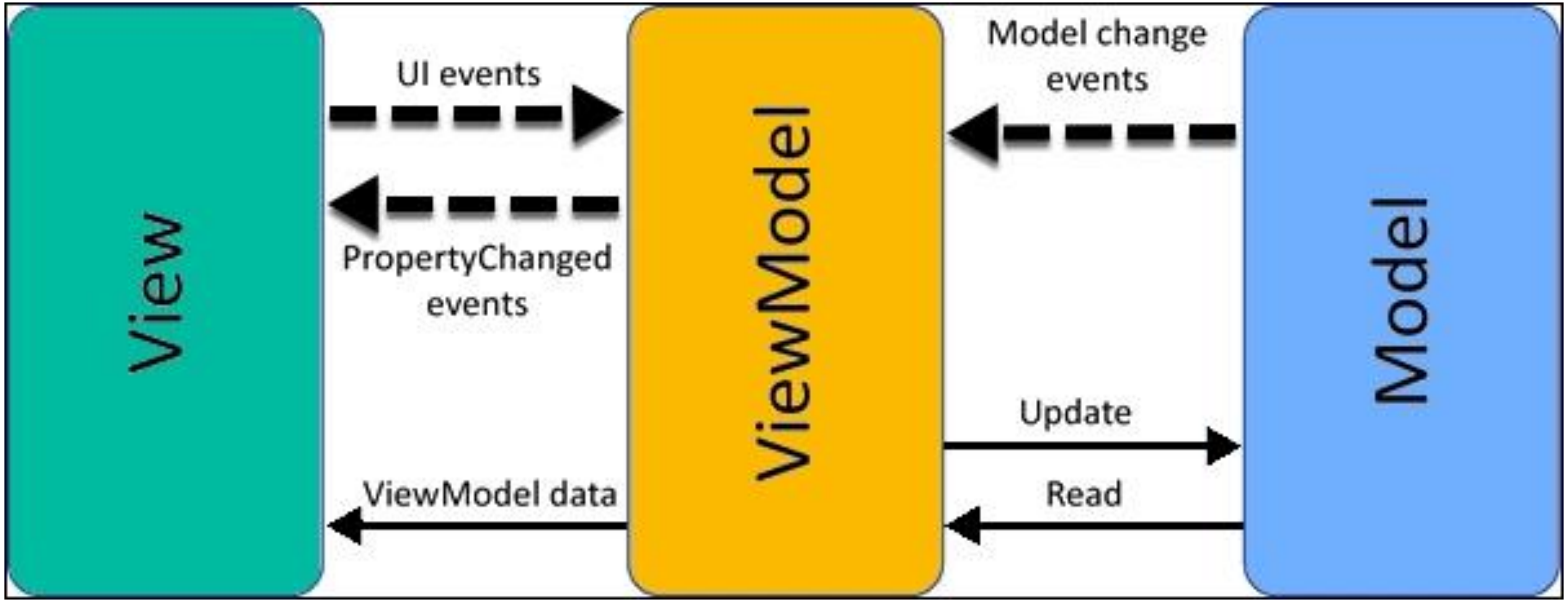

The same notification broadcasted to all the observers
Subject
Observer
Observer
Observer
A single notification observed by the subject

I
Or
Publish()
Or
Publish().RefCount()

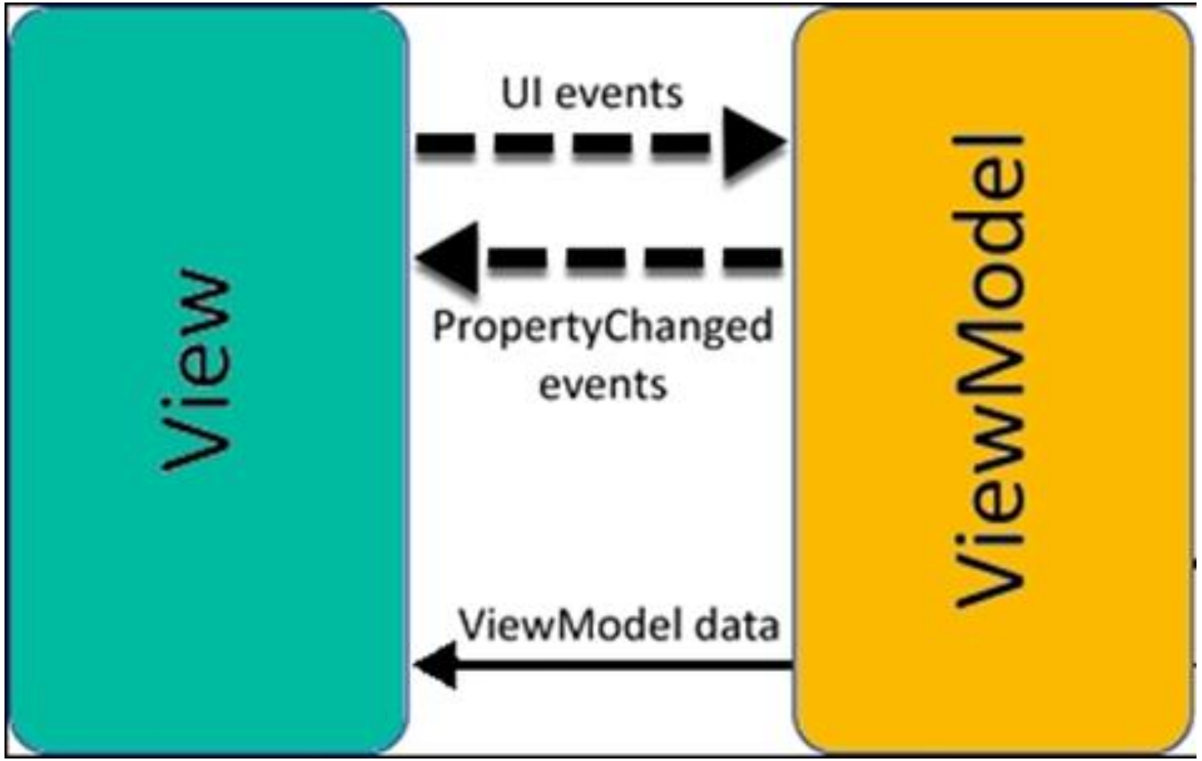
Demo

Reactive programming in frontend (WPF)

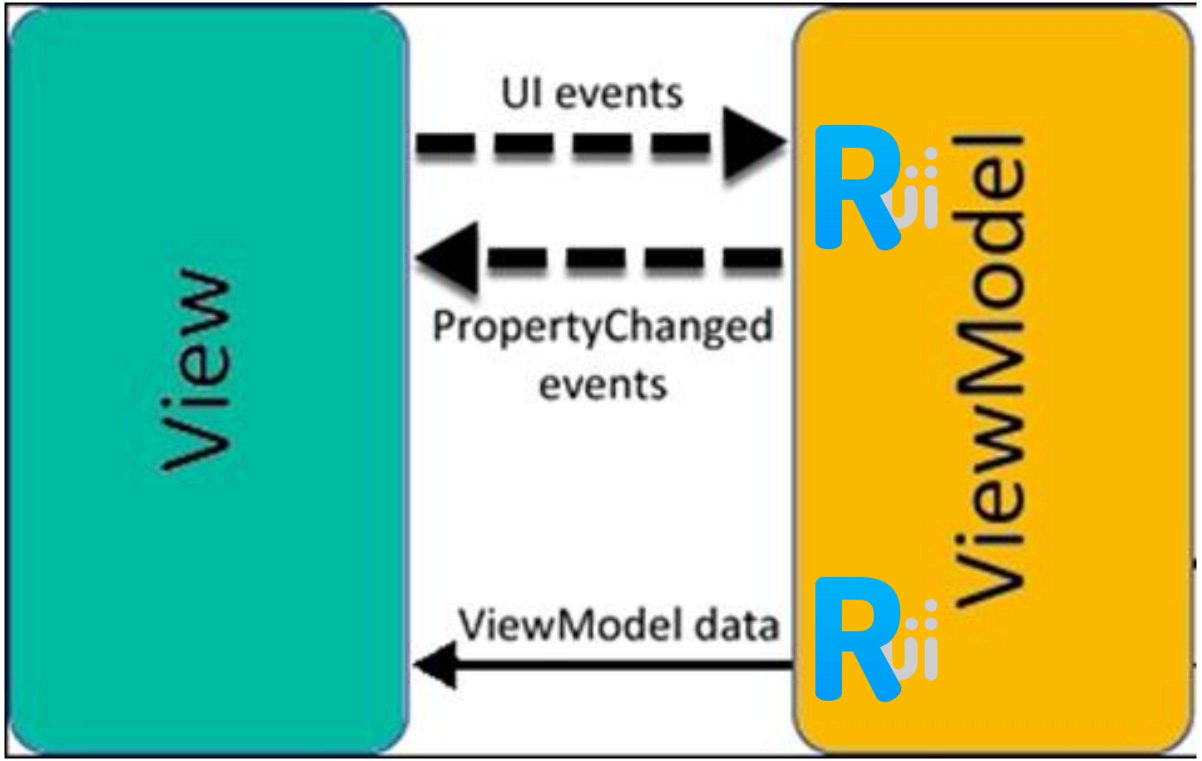
MVVM


View
UI events
PropertyChanged events
ViewModel data
ViewModel
Model change events
Update
Read
Model

MVVM

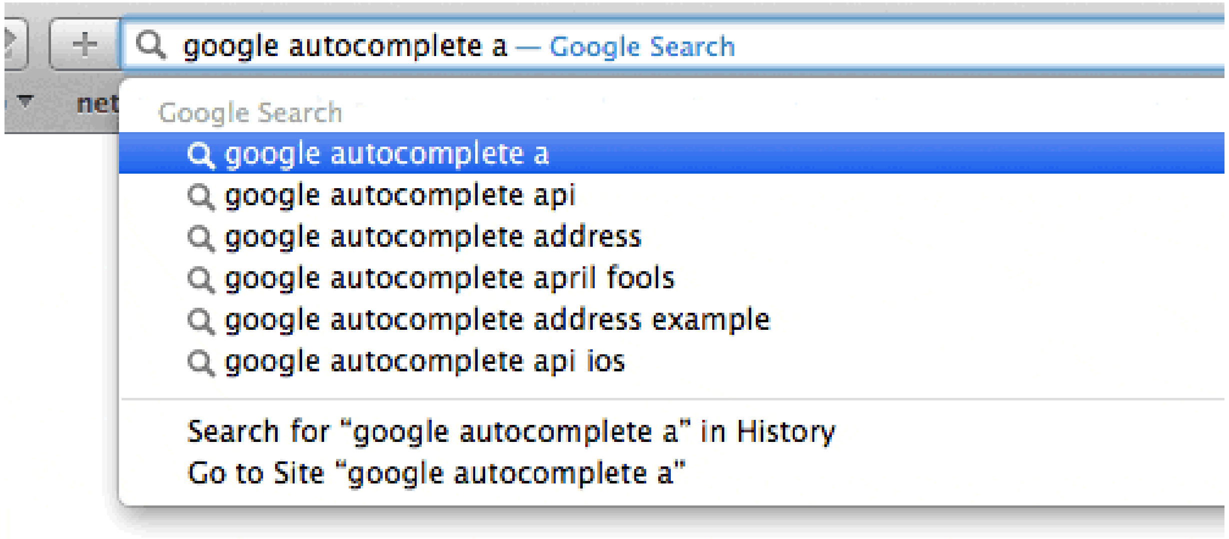

View
UI events
PropertyChanged events
ViewModel data
ViewModel

MVVM and ReactiveUI



ReactiveUI

Search example



google autocomplete a — Google Search

Google Search

google autocomplete a

google autocomplete api

google autocomplete address

google autocomplete april fools

google autocomplete address example

google autocomplete api ios

Search for "google autocomplete a" in History

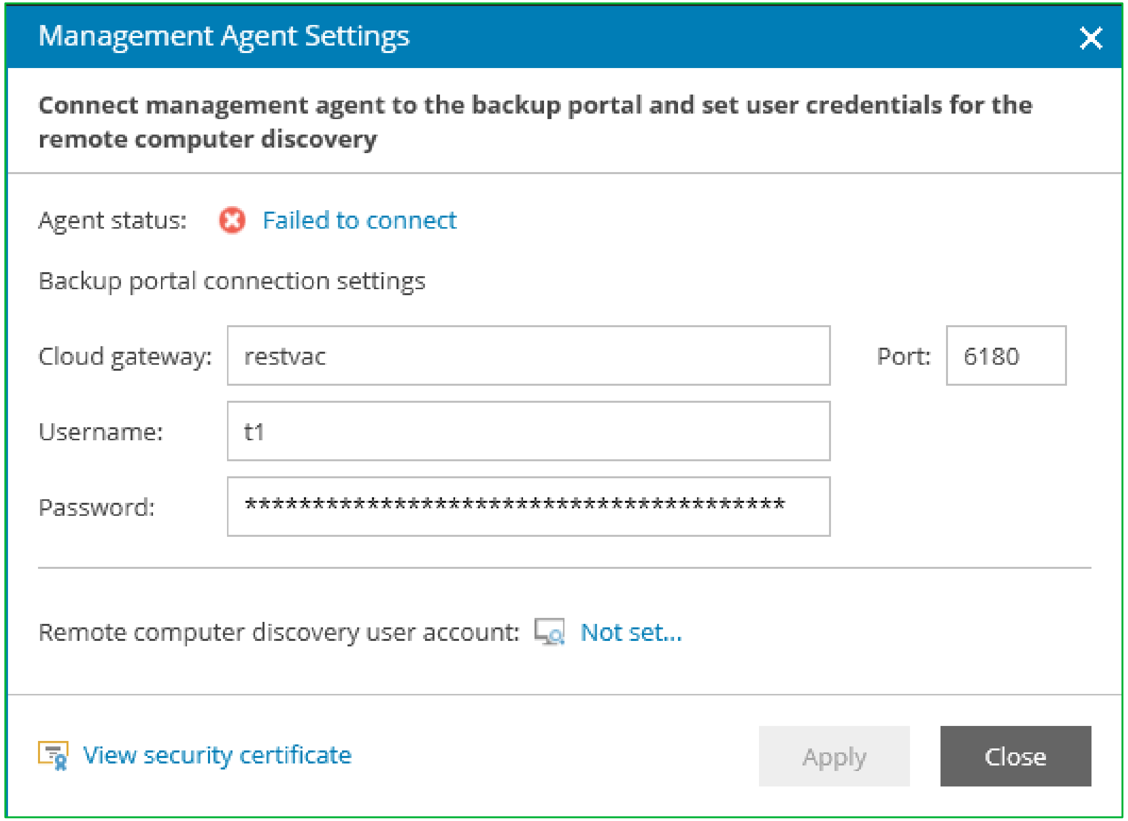
Go to Site "google autocomplete a"

Search example

```
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);
```

VSPC AgentUI

AgentUI



Management Agent Settings

Connect management agent to the backup portal and set user credentials for the remote computer discovery

Agent status: Failed to connect

Backup portal connection settings

Cloud gateway: restvac

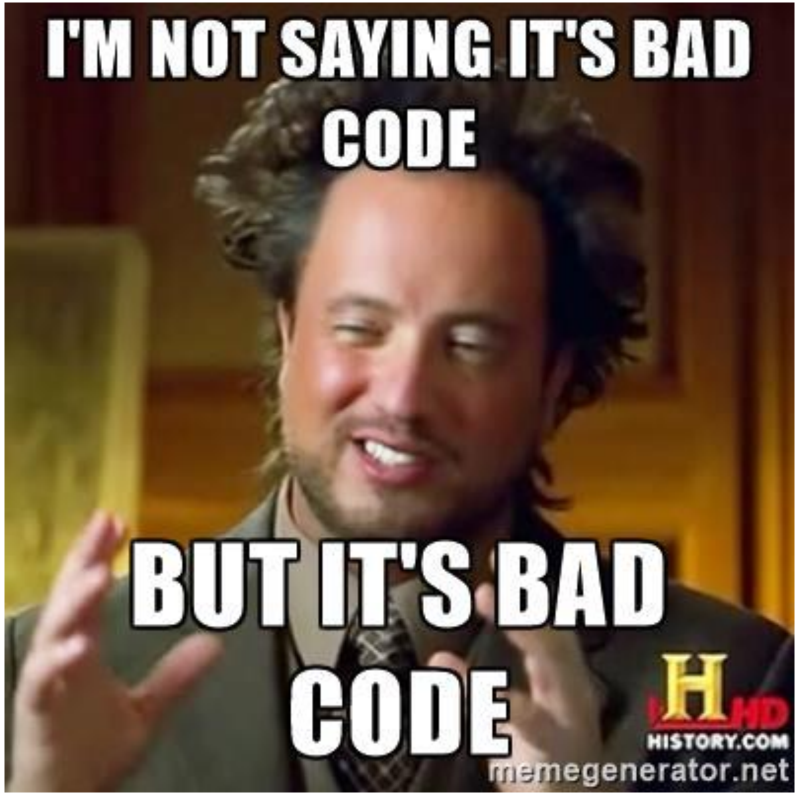
Port: 6180

Username: t1

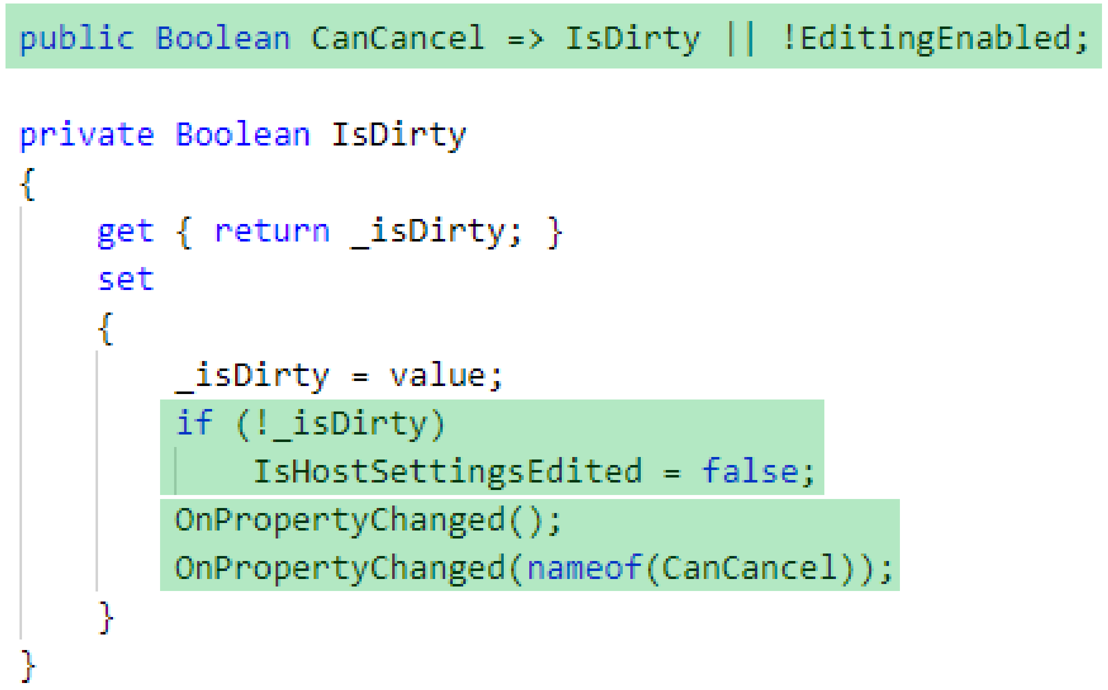
Password: **

Remote computer discovery user account: Not set...

AgentUI

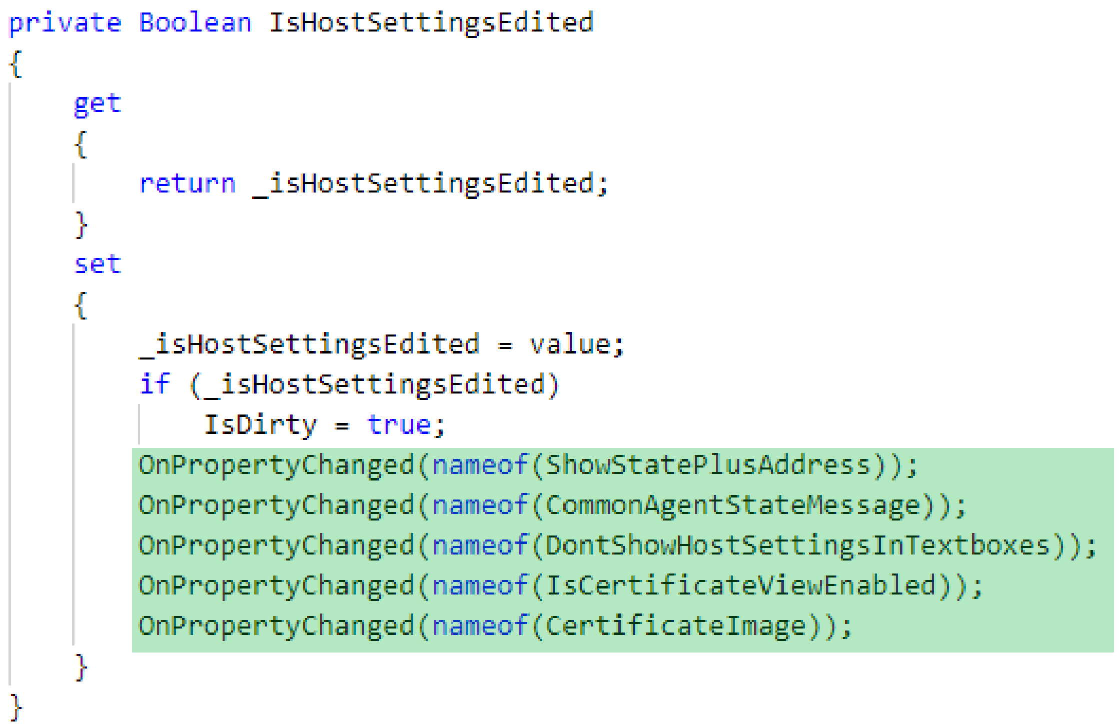

I'M NOT SAYING IT'S BAD CODE
BUT IT'S BAD CODE
HISTORY.COM
memegenerator.net

Before



```
public Boolean CanCancel => IsDirty || !EditingEnabled;

private Boolean IsDirty
{
    get { return _isDirty; }
    set
    {
        _isDirty = value;
        if (!_isDirty)
            IsHostSettingsEdited = false;
        OnPropertyChanged();
        OnPropertyChanged(nameof(CanCancel));
    }
}
```

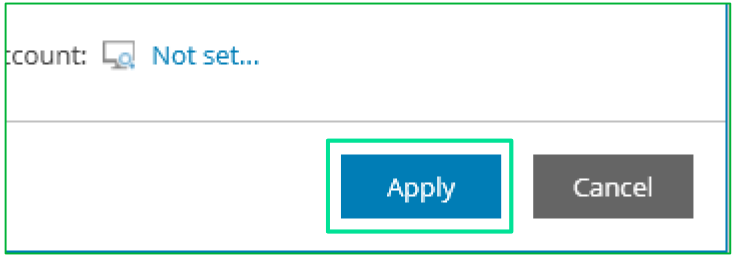
Before



```
private Boolean IsHostSettingsEdited
{
    get
    {
        return _isHostSettingsEdited;
    }
    set
    {
        _isHostSettingsEdited = value;
        if (_isHostSettingsEdited)
            IsDirty = true;
        OnPropertyChanged(nameof(ShowStatePlusAddress));
        OnPropertyChanged(nameof(CommonAgentStateMessage));
        OnPropertyChanged(nameof(DontShowHostSettingsInTextboxes));
        OnPropertyChanged(nameof(IsCertificateViewEnabled));
        OnPropertyChanged(nameof(CertificateImage));
    }
}
```

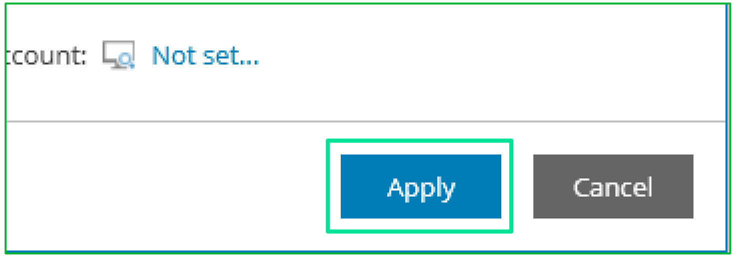
After

```
var canSaveChanges = isDirtyObservable
    .CombineLatest(agentHasCertificateIssue,
        (isDirty, isCertIssue) => isCertIssue || (isDirty && !HasErrors));
```


count: Not set...

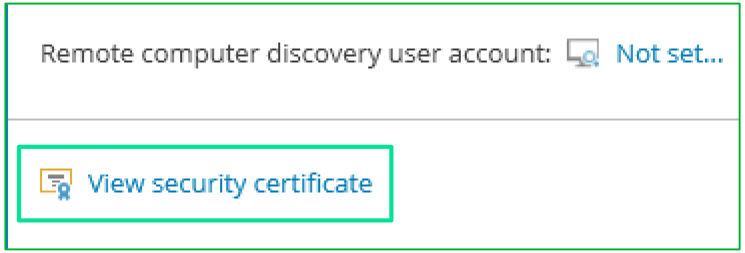
After

```
SaveCommand = ReactiveCommand.CreateFromObservable(
    () => SomeLogic(),
    canSaveChanges);
```


count: Not set...

After

```
var portAndHostAreNotEmptyObservable = hostnameObservable
    .CombineLatest(portObservable,
        (hostname, port) => !string.IsNullOrWhiteSpace(hostname) && !string.IsNullOrWhiteSpace(port));


var saveNotInProgressObservable = SaveCommand.IsExecuting
    .Invert()
    .Skip(1); // skip the first "isn't executing" state


_isCertificateViewEnabled = portAndHostAreNotEmptyObservable
    .Merge(saveNotInProgressObservable)
    .ToProperty(this, x => x.IsCertificateViewEnabled);
```

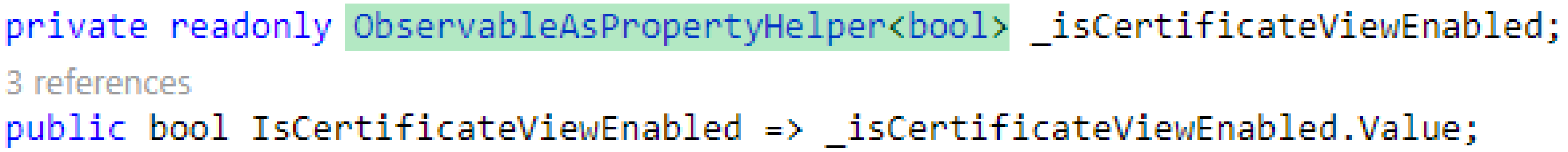

Remote computer discovery user account: Not set...

View security certificate

A bit more about ReactiveUI

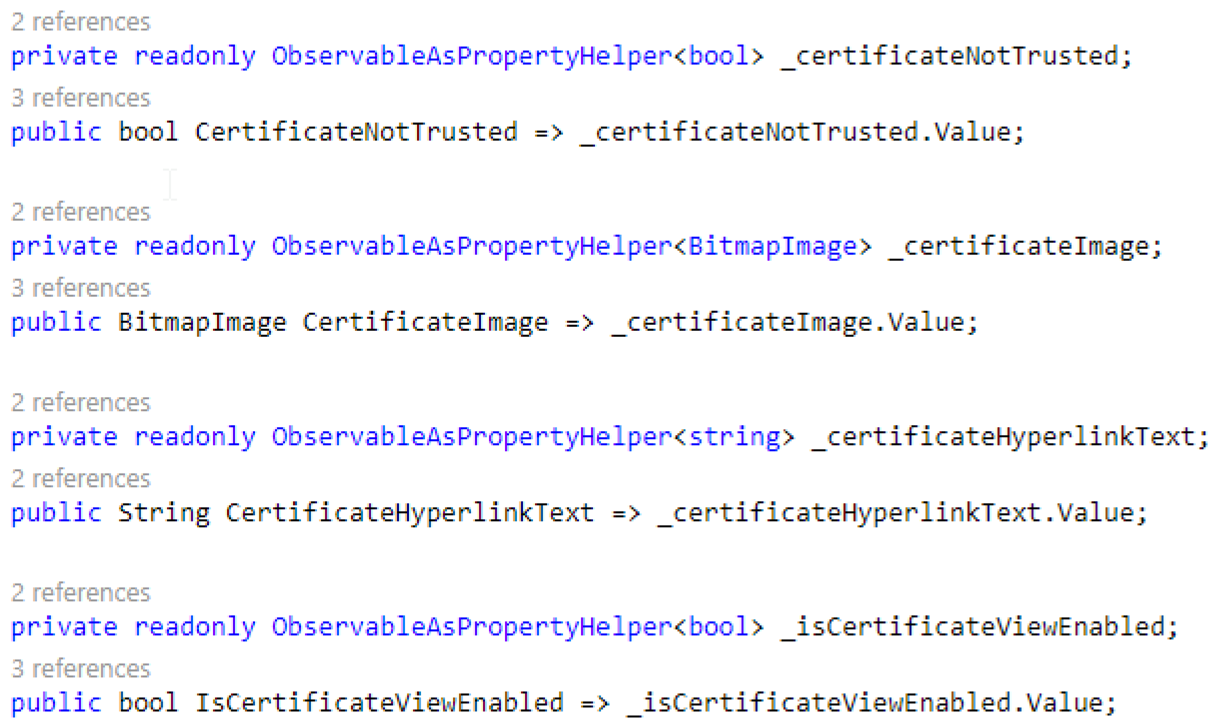
- Data Binding
- Data Persistence
- Routing
- View Location
- User Input Validation
- TestScheduler
- Message Bus

ReactiveUI disadvantages



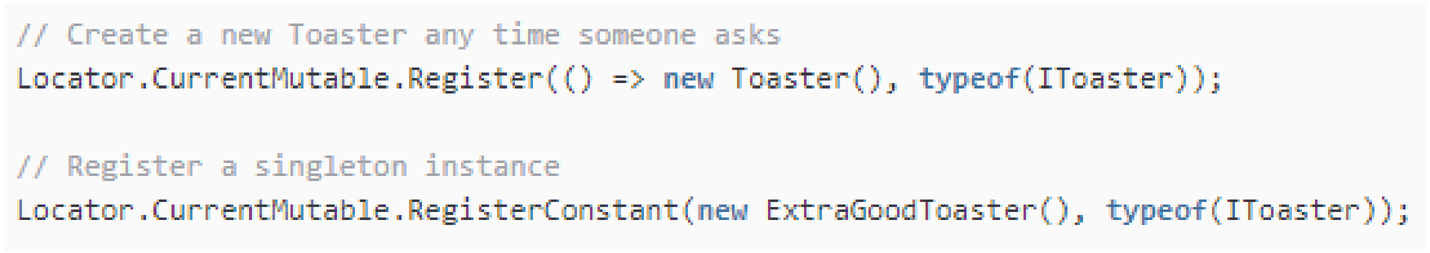
```
private readonly ObservableAsPropertyHelper<bool> _isCertificateViewEnabled;
3 references
public bool IsCertificateViewEnabled => _isCertificateViewEnabled.Value;
```

ReactiveUI disadvantages

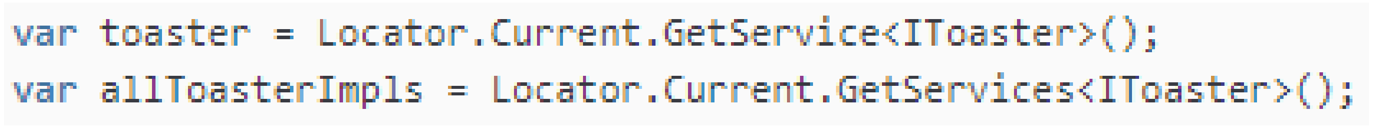


```
2 references
private readonly ObservableAsPropertyHelper<bool> _certificateNotTrusted;
3 references
public bool CertificateNotTrusted => _certificateNotTrusted.Value;

2 references
private readonly ObservableAsPropertyHelper<BitmapImage> _certificateImage;
3 references
public BitmapImage CertificateImage => _certificateImage.Value;

2 references
private readonly ObservableAsPropertyHelper<string> _certificateHyperlinkText;
2 references
public String CertificateHyperlinkText => _certificateHyperlinkText.Value;

2 references
private readonly ObservableAsPropertyHelper<bool> _isCertificateViewEnabled;
3 references
public bool IsCertificateViewEnabled => _isCertificateViewEnabled.Value;
```

ReactiveUI disadvantages

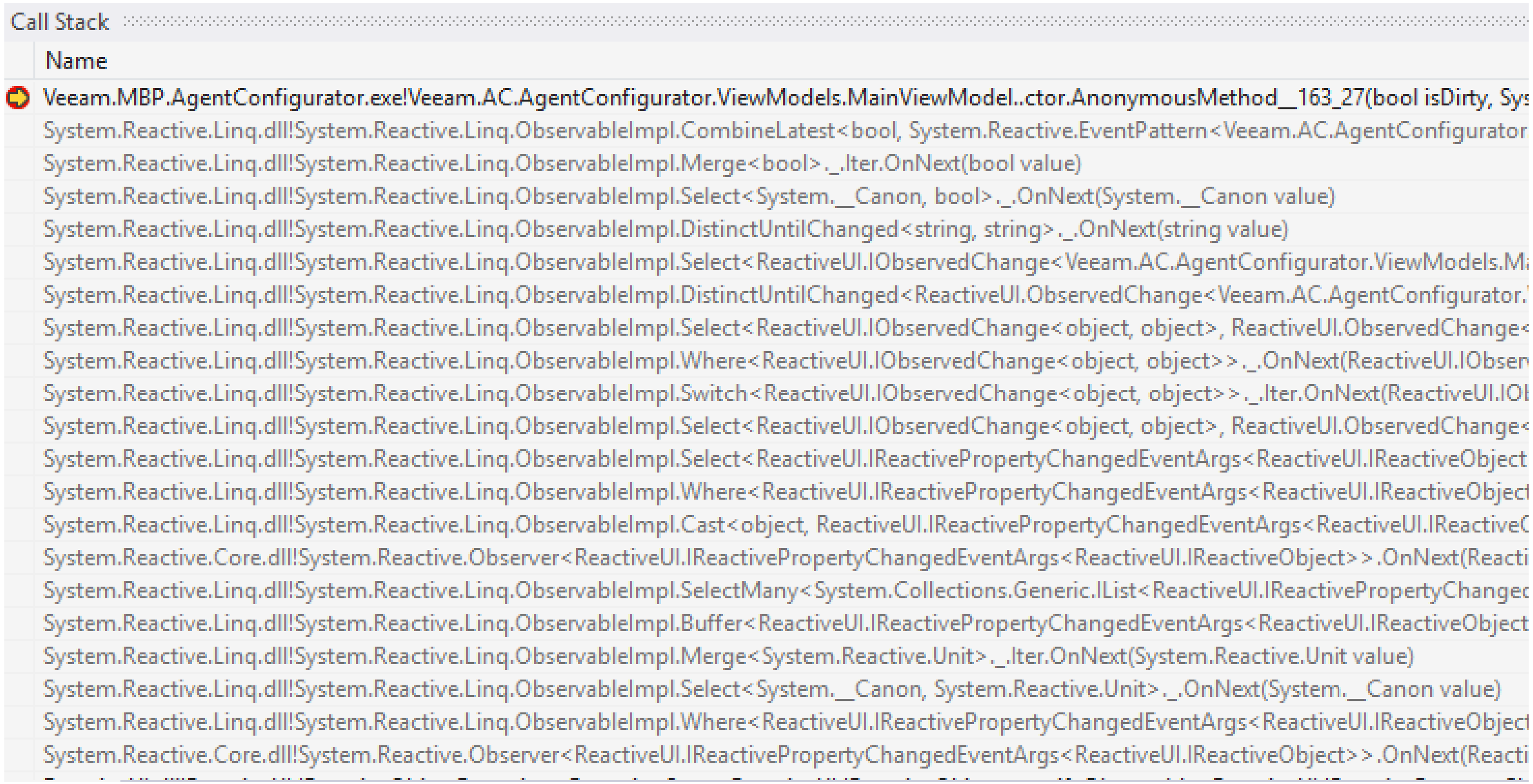


```
// Create a new Toaster any time someone asks
Locator.CurrentMutable.Register(() => new Toaster(), typeof(IToaster));

// Register a singleton instance
Locator.CurrentMutable.RegisterConstant(new ExtraGoodToaster(), typeof(IToaster));
```



```
var toaster = Locator.Current.GetService<IToaster>();
var allToasterImpls = Locator.Current.GetServices<IToaster>();
```

RX disadvantages

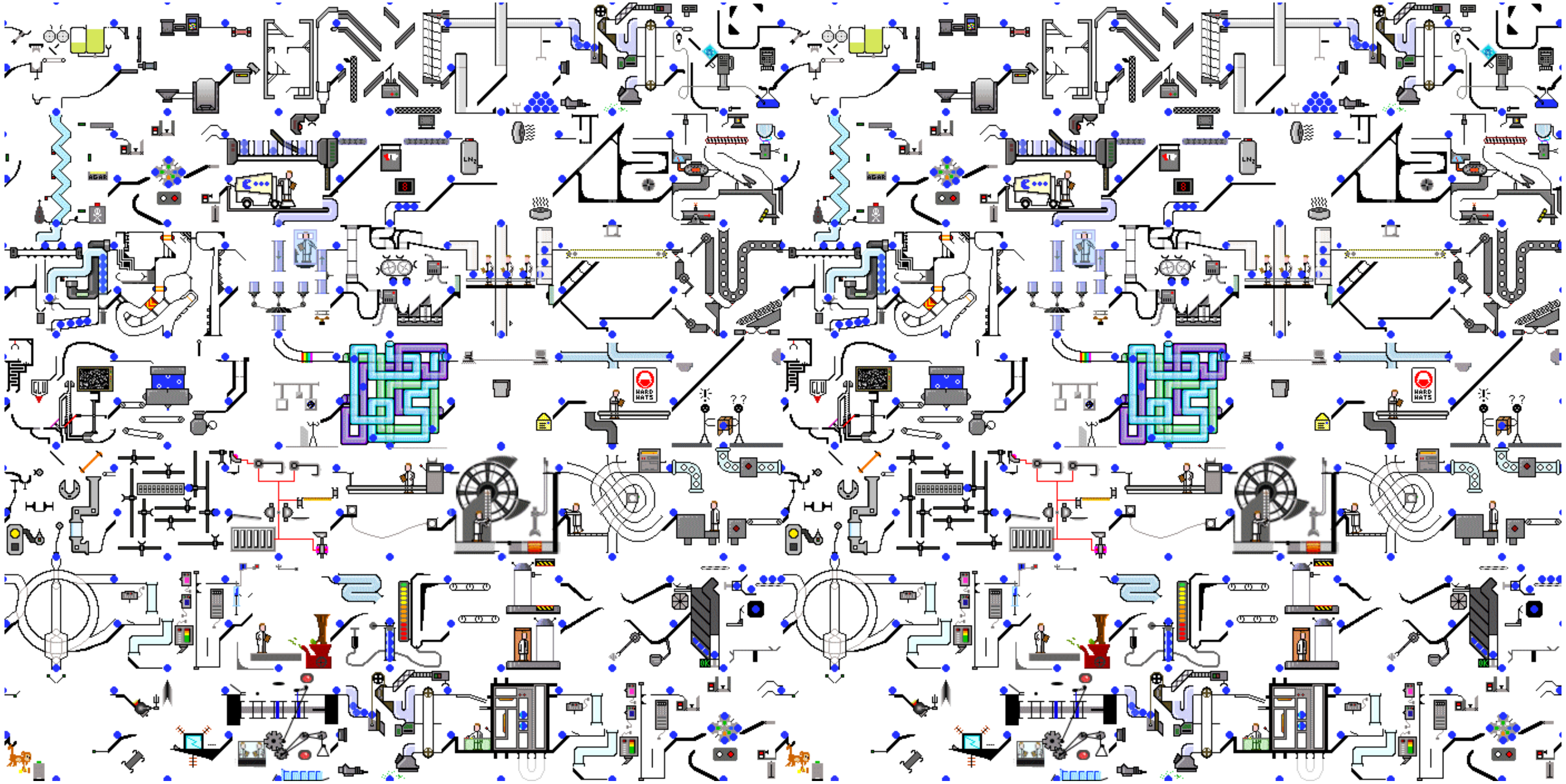


Call Stack

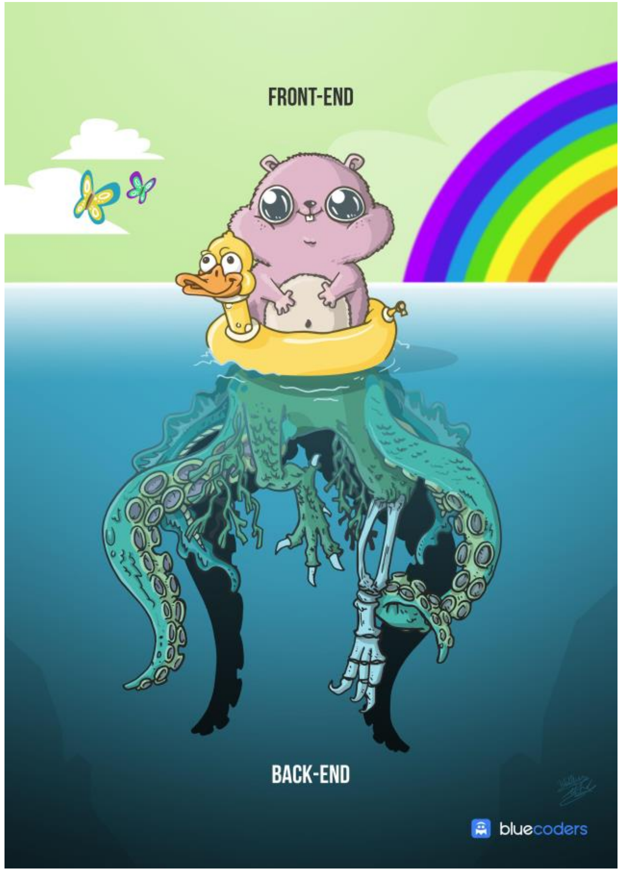
Name

```
Veeam.MBP.AgentConfigurator.exe!Veeam.AC.AgentConfigurator.ViewModels.MainViewModel..ctor.AnonymousMethod__163_27(bool isDirty, Sys
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.CombineLatest<bool, System.Reactive.EventPattern<Veeam.AC.AgentConfigurator
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.Merge<bool>._.Iter.OnNext(bool value)
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.Select<System.__Canon, bool>._.OnNext(System.__Canon value)
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.DistinctUntilChanged<string, string>._.OnNext(string value)
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.Select<ReactiveUI.IObservedChange<Veeam.AC.AgentConfigurator.ViewModels.M
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.DistinctUntilChanged<ReactiveUI.ObservedChange<Veeam.AC.AgentConfigurator.
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.Select<ReactiveUI.IObservedChange<object, object>, ReactiveUI.ObservedChange<
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.Where<ReactiveUI.IObservedChange<object, object>>._.OnNext(ReactiveUI.IObser
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.Switch<ReactiveUI.IObservedChange<object, object>>._.Iter.OnNext(ReactiveUI.IOb
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.Select<ReactiveUI.IObservedChange<object, object>, ReactiveUI.ObservedChange<
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.Select<ReactiveUI.IReactivePropertyChangedEventArgs<ReactiveUI.IReactiveObject
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.Where<ReactiveUI.IReactivePropertyChangedEventArgs<ReactiveUI.IReactiveObject
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.Cast<object, ReactiveUI.IReactivePropertyChangedEventArgs<ReactiveUI.IReactive
System.Reactive.Core.dll!System.Reactive.Observer<ReactiveUI.IReactivePropertyChangedEventArgs<ReactiveUI.IReactiveObject>>.OnNext(Reacti
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.SelectMany<System.Collections.Generic.IList<ReactiveUI.IReactivePropertyChange
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.Buffer<ReactiveUI.IReactivePropertyChangedEventArgs<ReactiveUI.IReactiveObject
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.Merge<System.Reactive.Unit>._.Iter.OnNext(System.Reactive.Unit value)
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.Select<System.__Canon, System.Reactive.Unit>._.OnNext(System.__Canon value)
System.Reactive.Linq.dll!System.Reactive.Linq.ObservableImpl.Where<ReactiveUI.IReactivePropertyChangedEventArgs<ReactiveUI.IReactiveObject
System.Reactive.Core.dll!System.Reactive.Observer<ReactiveUI.IReactivePropertyChangedEventArgs<ReactiveUI.IReactiveObject>>.OnNext(Reacti
```

RX disadvantages



RX disadvantages

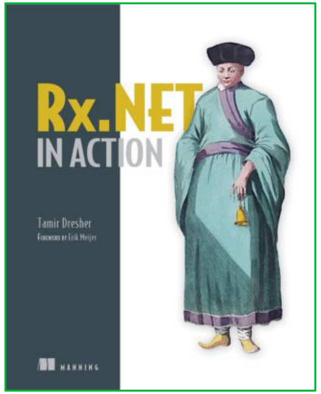


Conclusion

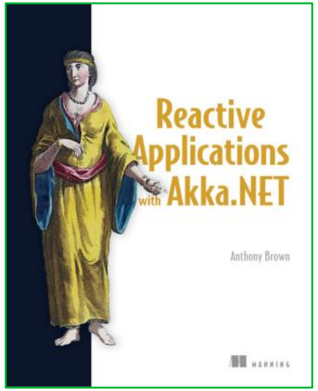
- Используйте реактивную парадигму при взаимодействии с “физическим” миром
- RX полезен, но не панацея
- Если вы разрабатываете XAML приложения, посмотрите на ReactiveUI
- При разработке распределённых систем, реактивный подход вне конкуренции

Links

- Introduction to Rx (Web book)
- ReactiveUI
- Доклад с .Next: Tamir Dresher — Reactive Extensions (Rx) 101
- Book Rx.NET in Action
- Book Reactive Design Patterns
- Book Reactive Applications with Akka.NET



Реактивные шаблоны проектирования
Роланд Кун
при участии Брайана Ханафи и Джейми Аллена
MANNING
ПИТЕР



Link

Link