ReactiveX: The observer pattern done right

STEFAN GLIENKE

About me

- Experience in Turbo Pascal and Delphi since 1997
- Education as software developer
- Participation in several open source projects

- Embarcadero MVP since 2014 and „MVP of the year“ 2015

- Specialized in following areas
  - Development of logic and data layers
  - Software design and architecture
  - „Clean code“

- Lead developer of Spring4D

Agenda

What is ReactiveX?

- Observable
- Pull vs Push
- Operators
- Schedulers

First look at Spring.Reactive

What is ReactiveX?

Reactive Extensions

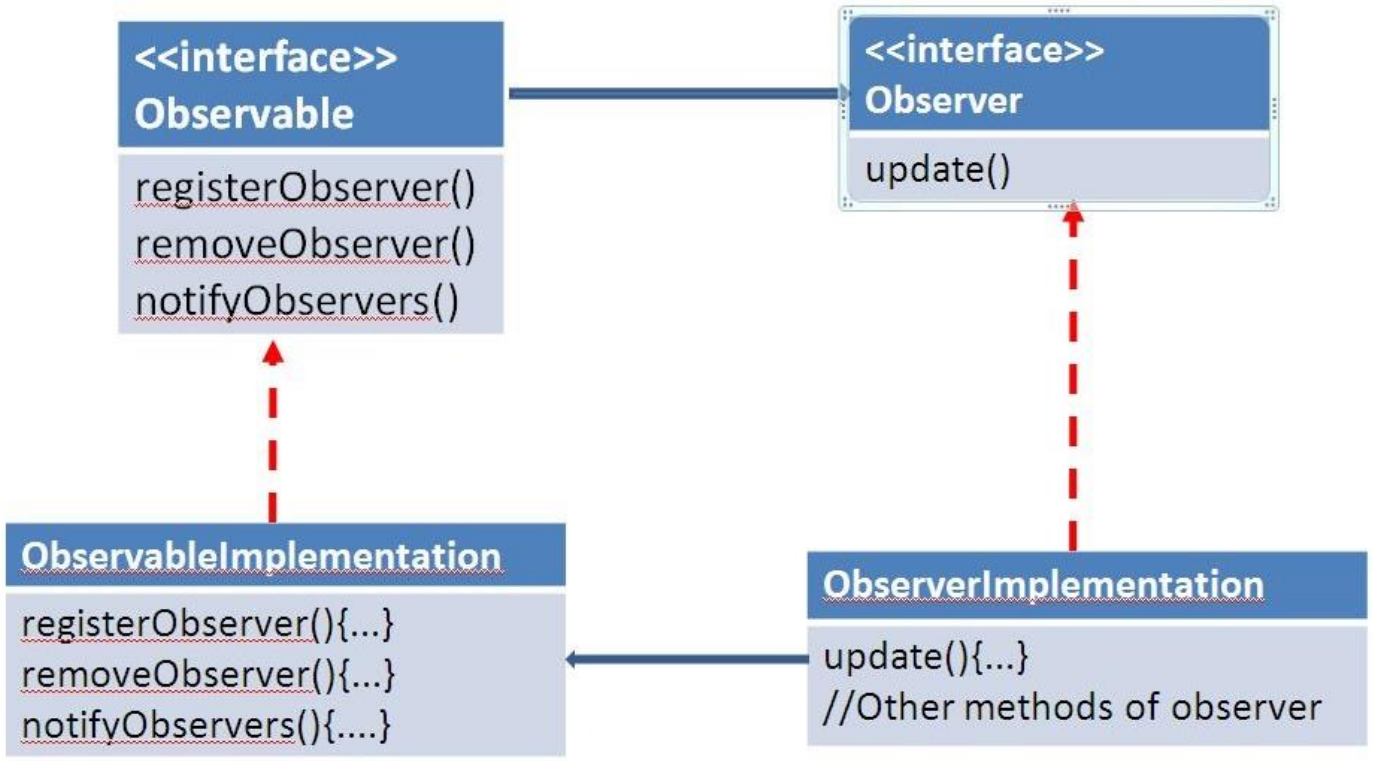
Reactor Pattern

a combination of the best ideas from

- the Observer pattern,
- the Iterator pattern,
- and functional programming

Polyglot implementation

What's the observer pattern again?



Iterator + Observable

	Single items	Multiple items
synchronous	T getData()	IEnumerable<T> GetData()
asynchronous	IFuture<T> getData()	IObservable<T> GetData()

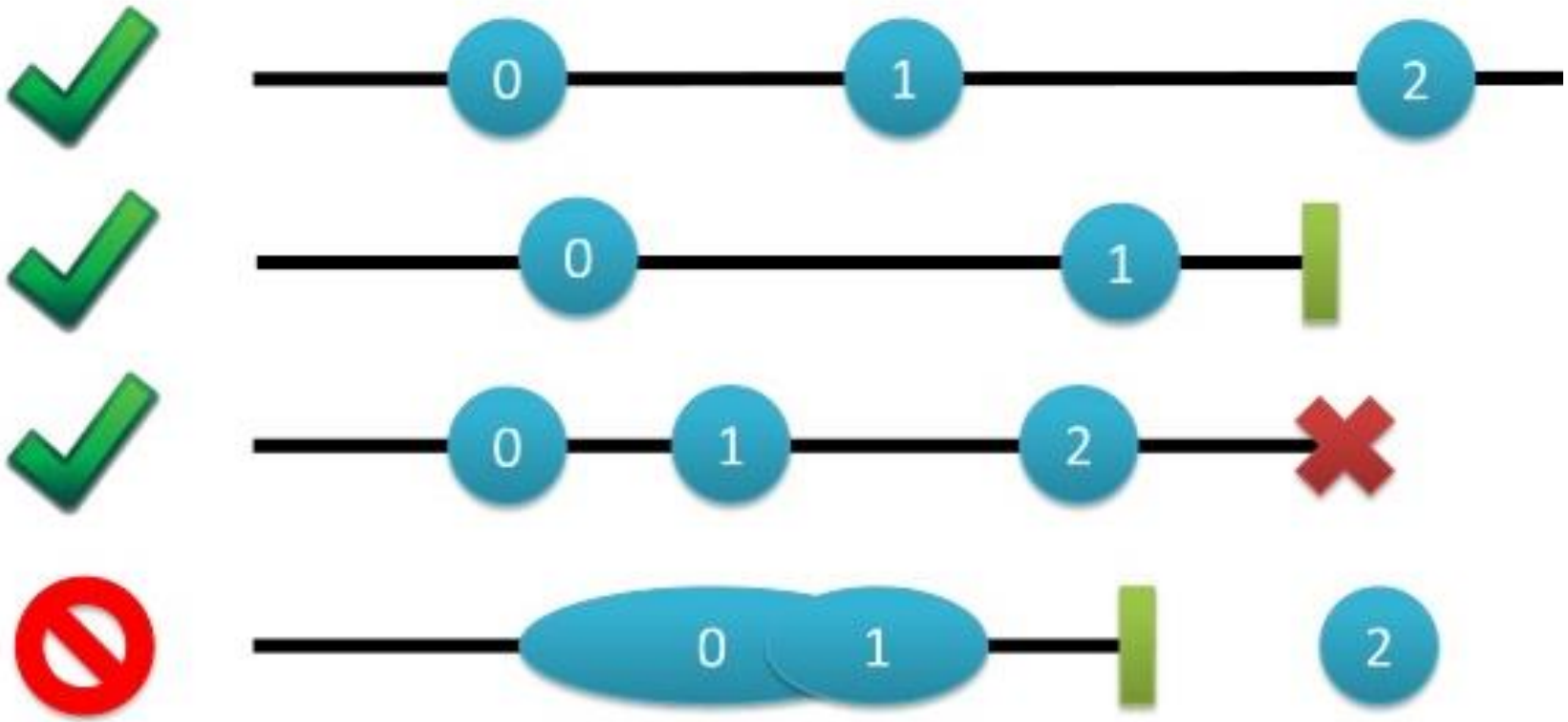
Iterator <-> Observable

event	Enumerable (pull)	Observable (push)
Retrieve data	MoveNext + GetCurrent	onNext(T)
Discover error	Raises exception	onError(Exception)
Complete	MoveNext returned False	onCompleted()

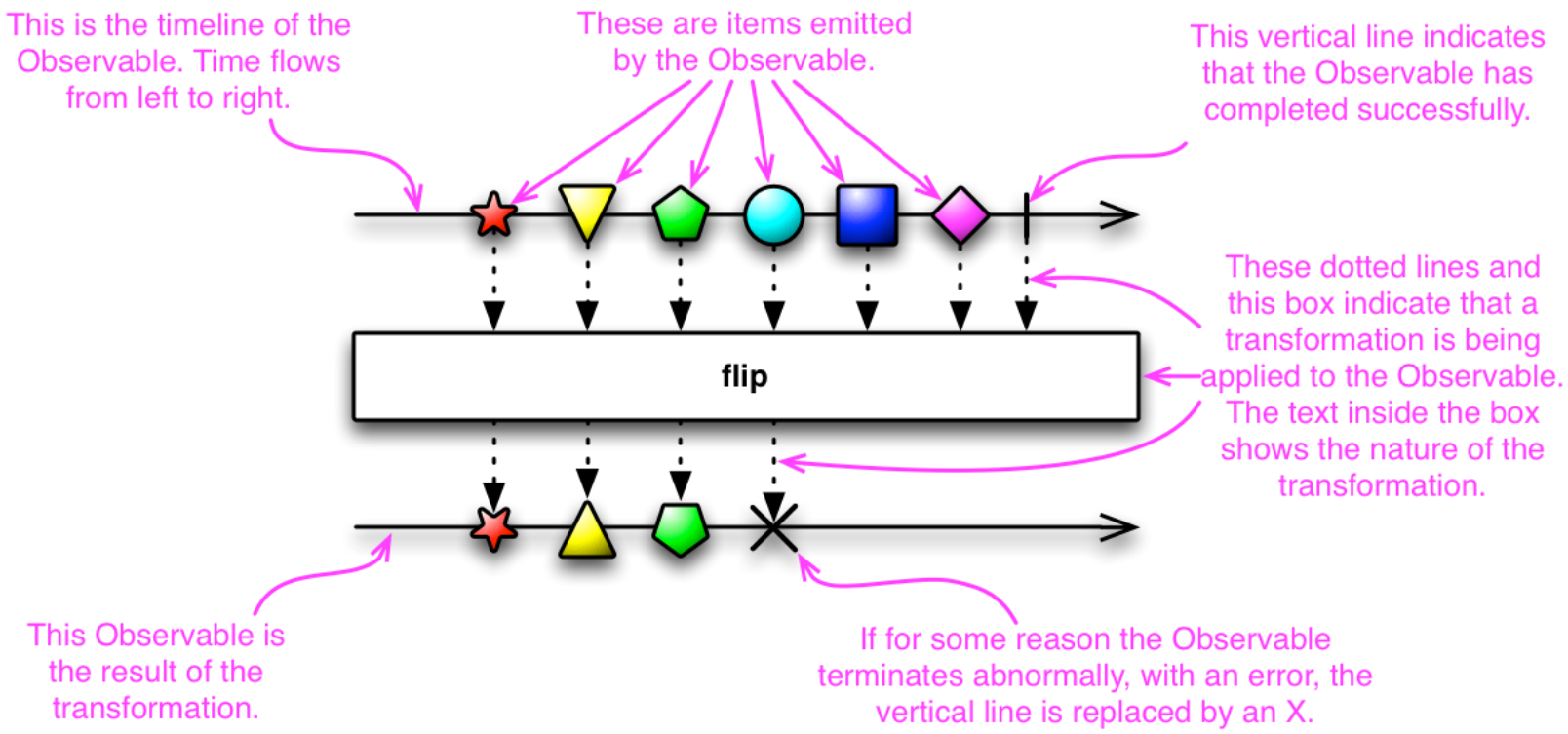
Iterator <-> Observable

Enumerable	Observable
`for x in getDataFromNetwork do` `  Process(x);`	`getDataFromNetwork` `  .Subscribe(Process);`

The contracts



Observable


This is the timeline of the Observable. Time flows from left to right.
These are items emitted by the Observable.
This vertical line indicates that the Observable has completed successfully.
These dotted lines and this box indicate that a transformation is being applied to the Observable. The text inside the box shows the nature of the transformation.
flip
This Observable is the result of the transformation.
If for some reason the Observable terminates abnormally, with an error, the vertical line is replaced by an X.

Why use Observable?

Composable

Flexible

Less opinionated

Operators

Creating

- Create, Defer, Empty/Never/Throw, From, Interval, Just, Range, Repeat, Start, Timer

Transforming:

- Buffer, FlatMap, GroupBy, Map, Scan, Window

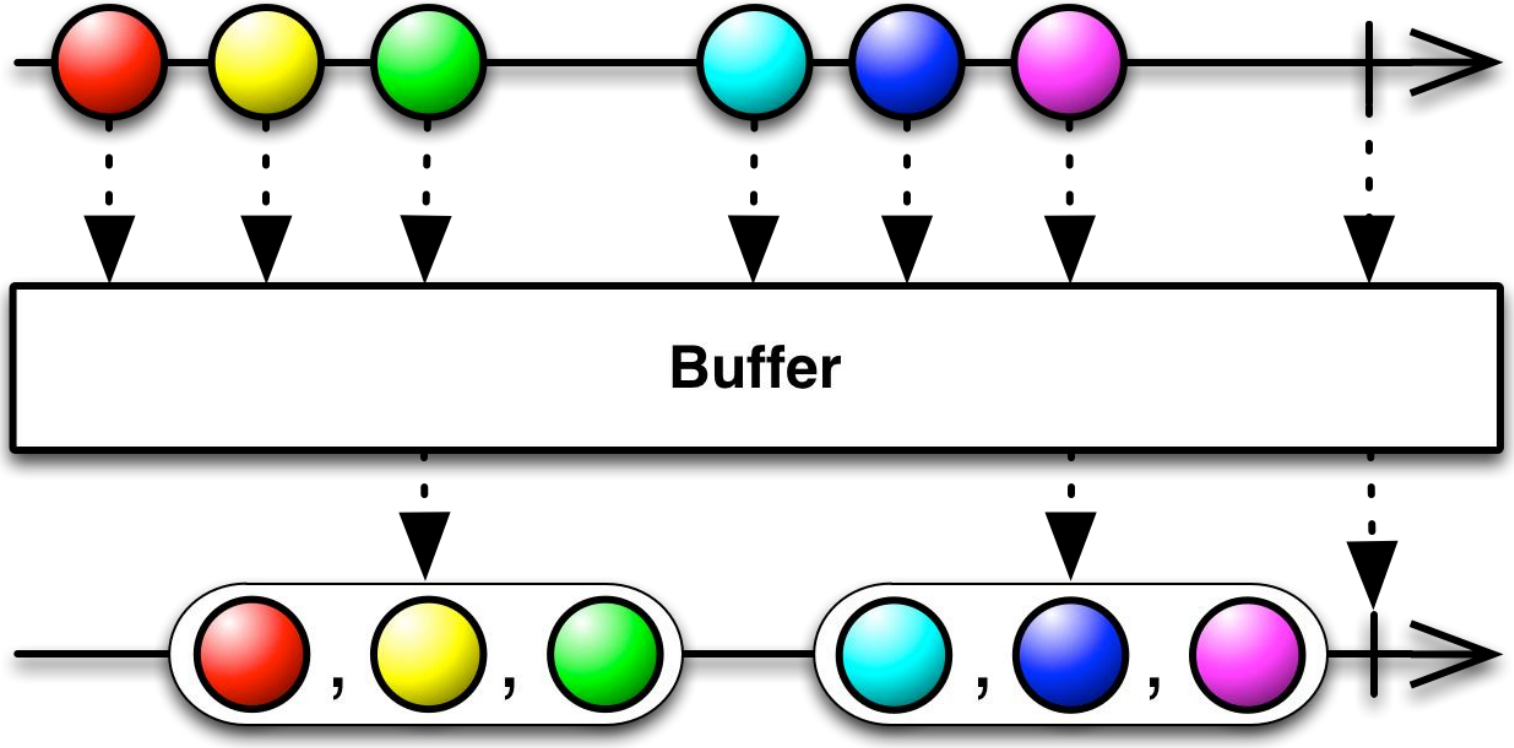
Filtering:

- Debounce, Distinct, ElementAt, Filter, First, IgnoreElements, Last, Sample, Skip, SkipLast, Take, TakeLast

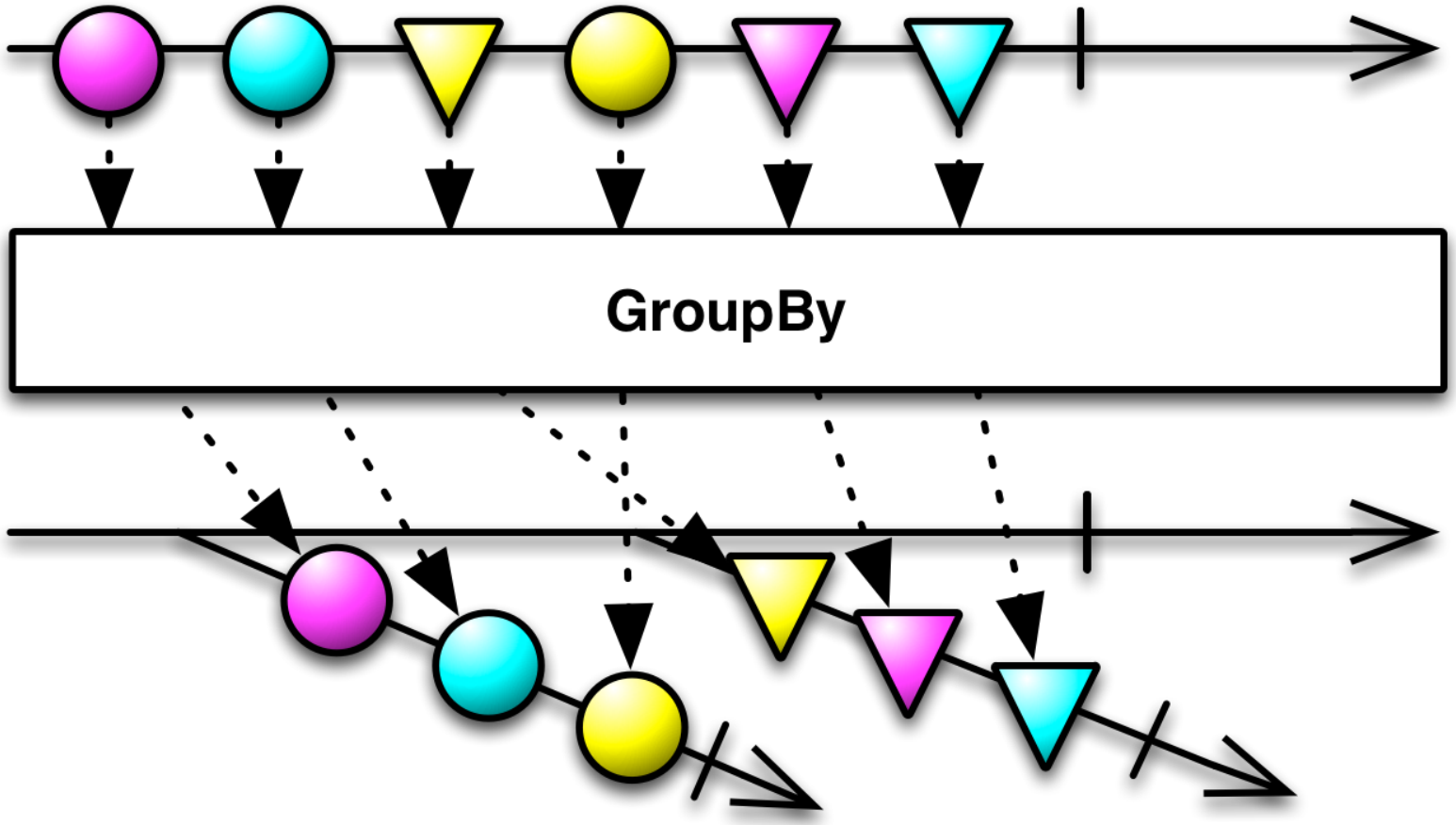
Combining:

- And/Then/When, CombineLatest, Join, Merge, StartWith, Switch, Zip

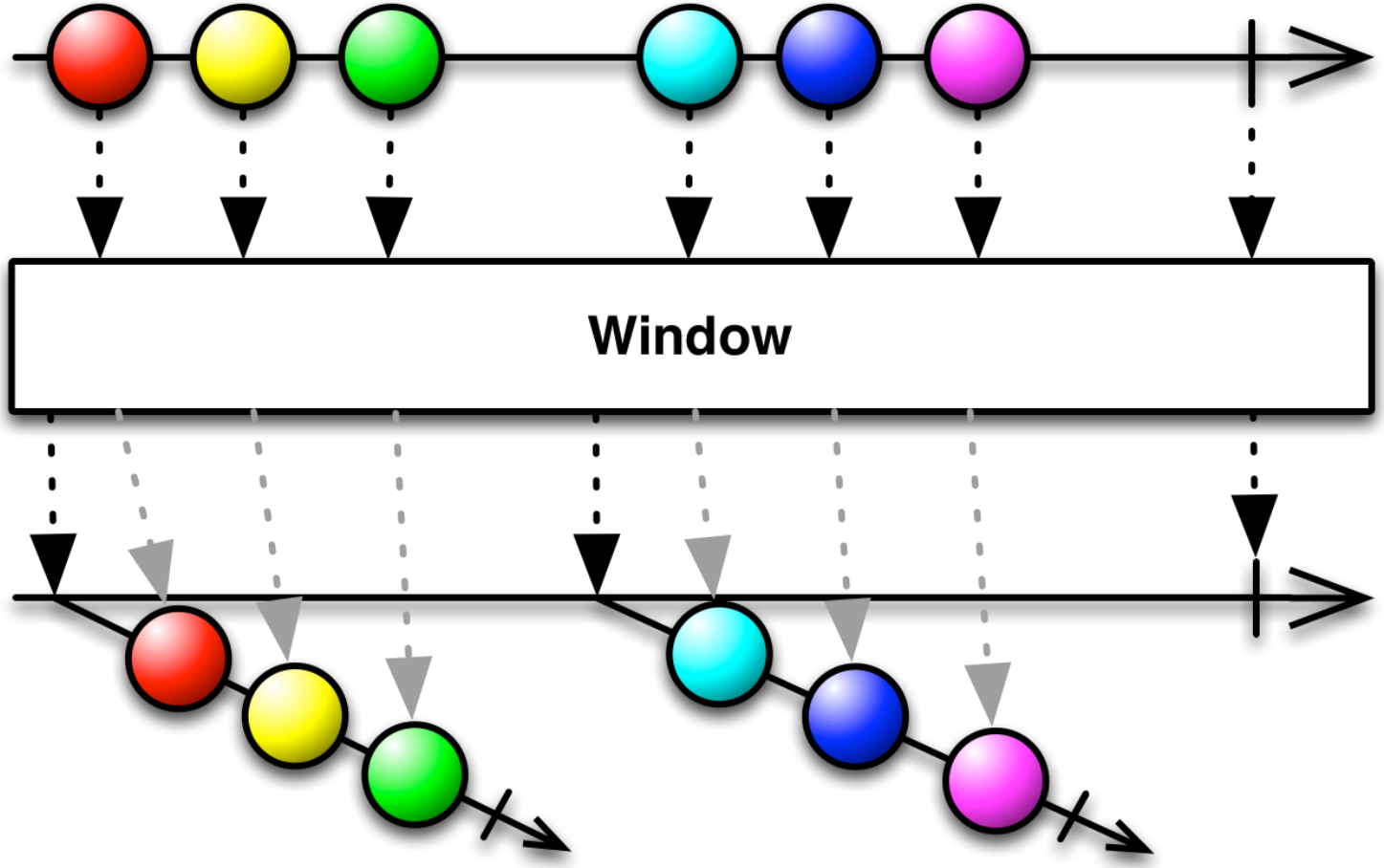
Buffer



GroupBy



Window



Schedulers

RX is a free-threaded model

Schedulers are controlling where work is being done

Control over where subscribers are doing their work

IObservable<T> and IObserver<T>

```
IObservable<T> = interface
  function Subscribe(const observer: IObserver<T>): IDisposable;
end;
```

```
IObserver<T> = interface
  procedure OnNext(const value: T);
  procedure OnError(const error: Exception);
  procedure OnCompleted;
end;
```

Let‘s look at some code!

Delphi implementation difficulties

All operators need to be implemented on the IObservable<T> interface or via static methods

- for better support we need interface helpers (aka extension methods)
- Please vote!
  - https://quality.embarcadero.com/browse/RSP-10336 (generic type helpers)
  - https://quality.embarcadero.com/browse/RSP-16763 (interface helpers)

Lifetime management of objects being processed through observables need to be considered

- ARC vs no ARC

ARC on interfaces is working slightly different than in GC languages

- Easy to cause circular references especially when using nested observables and anonymous methods

Future plans for Spring.Reactive

Support for all canonical operators of Reactive including extensive unit tests

Implementation of various schedulers for different situations

Pushing the Delphi language evolution ;)

I want to know more about ReactiveX!

ReactiveX page:
- http://reactivex.io

Introduction to Rx (mostly the C# implementation):
- http://www.introtorx.com

Video series done by Microsoft developers:
- https://channel9.msdn.com/Series/Rx-Workshop/Rx-Workshop-Introduction

Spring4D:
- http://spring4d.org

Email:
sglienke@dsharp.org

Thank you very much for your attention!

Questions?