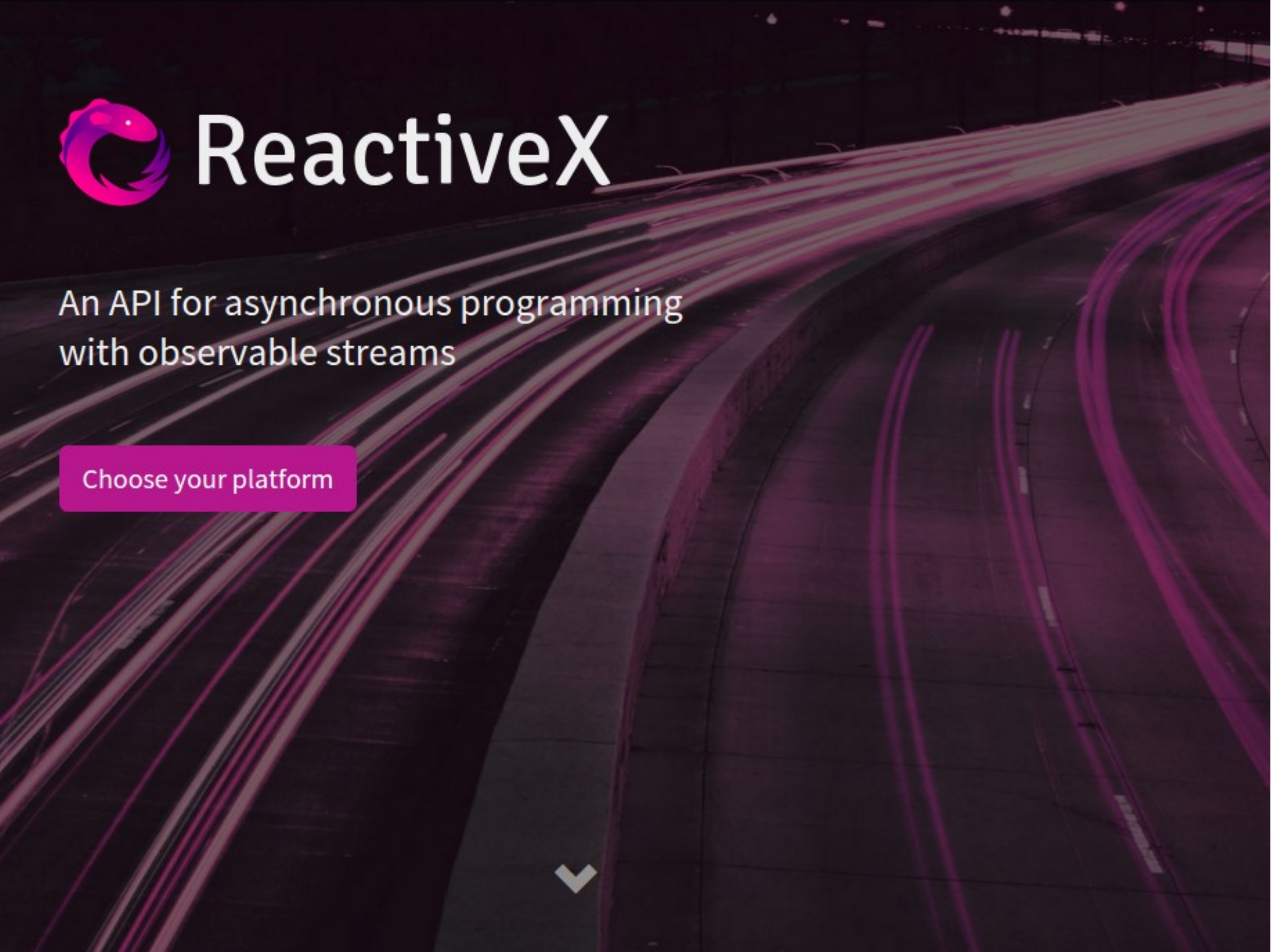
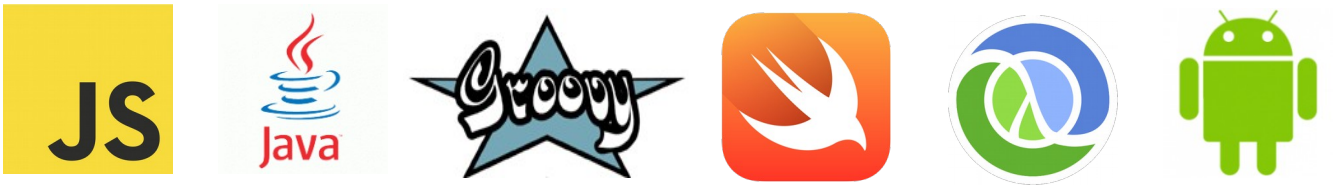
RXJava

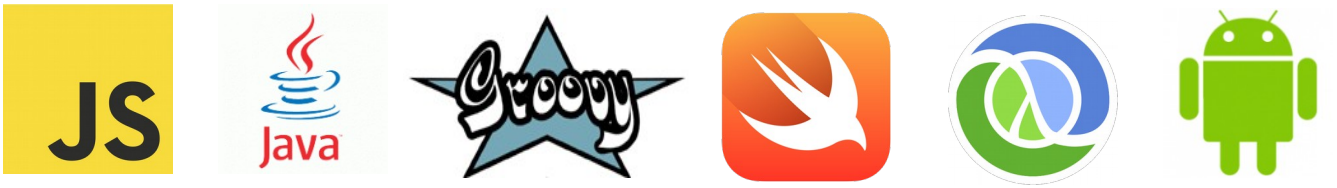
Daniel Hinojosa
@dhinojosa


ReactiveX
An API for asynchronous programming
with observable streams
Choose your platform

Reactive Extensions for Async Programming



RX
for Async Programming



RX Java

Definition

A library for composing asynchronous and event-based programs by using observable sequences.

Unbiased

The source for any `Observable` can from anywhere.

formerly of....

Microsoft

“Mouse is a database”


PUSH

Observer/Iterator

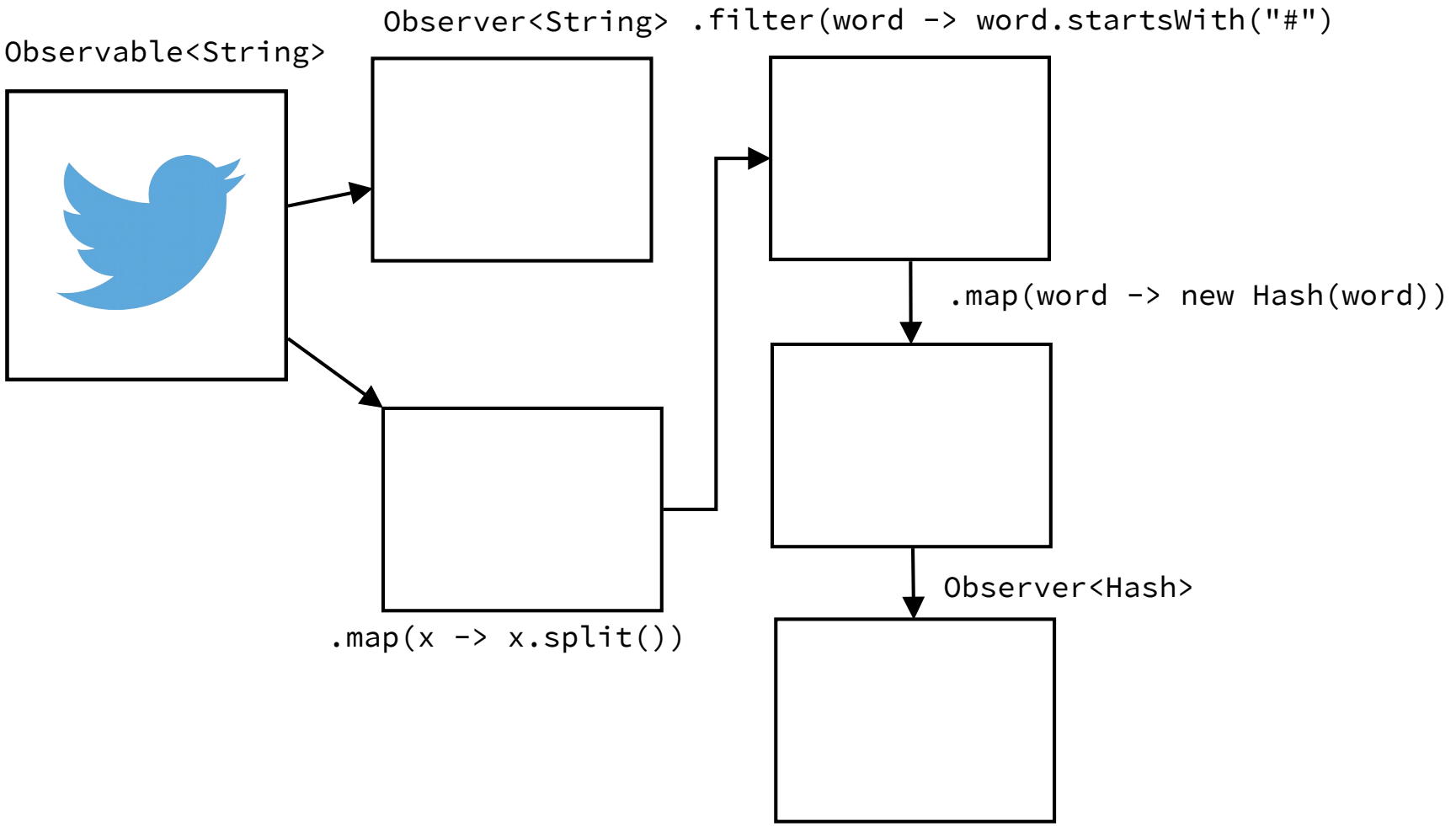
What pains does RXJava alleviate?

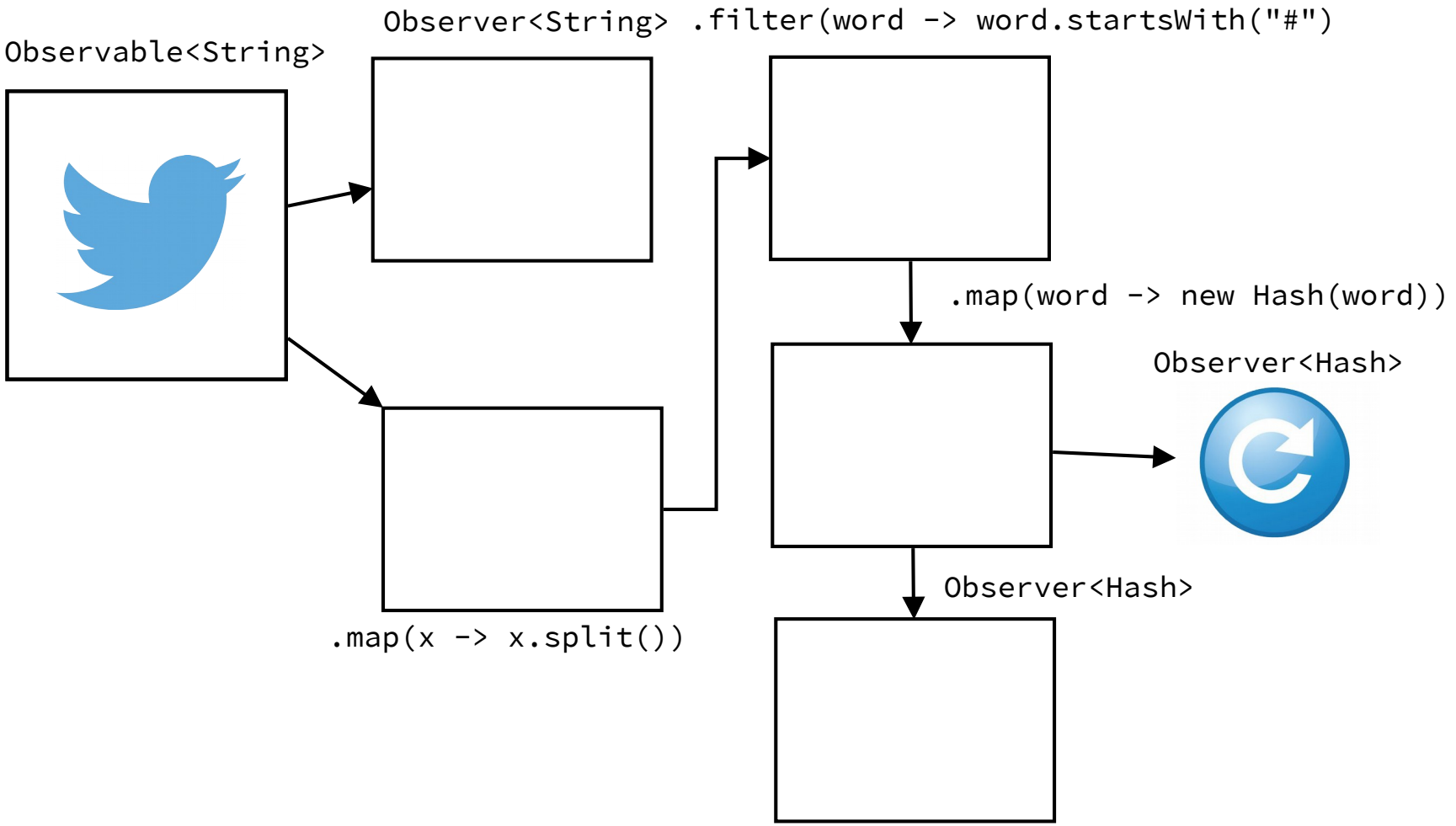
Callback Hell

Retaining a Source

Backpressure Remedies

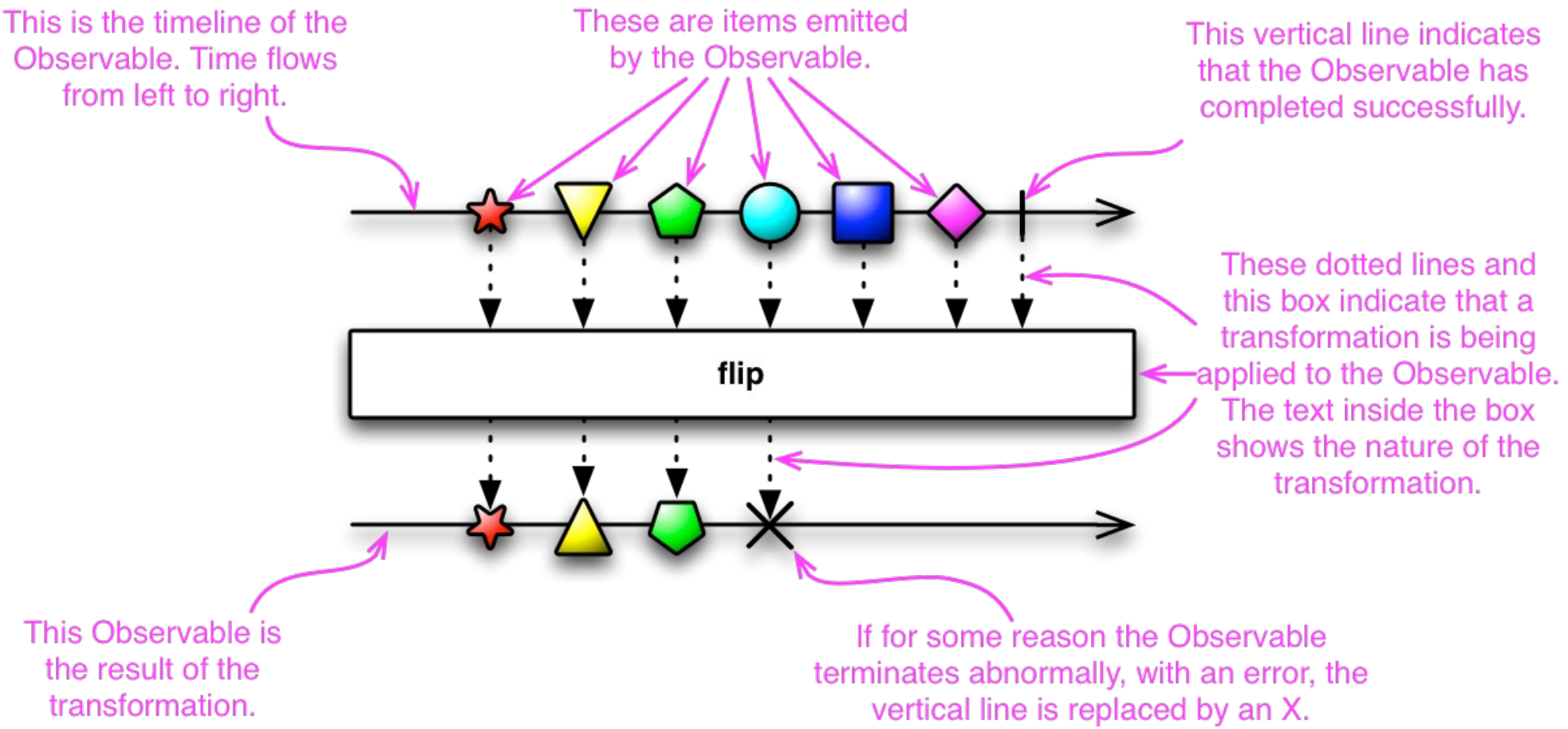
	One	Many
Sync	`T`	`Iterator<T>`
Async	`Future<T>`	`Observable<T>`


Observable<String>
Observer<String>
.filter(word -> word.startsWith("#")
.map(word -> new Hash(word))
.map(x -> x.split())
Observer<Hash>


Observable<String>
Observer<String>
.filter(word -> word.startsWith("#")
.map(word -> new Hash(word))
Observer<Hash>
.map(x -> x.split())
Observer<Hash>

Thinking of `Observable`s as Pipes

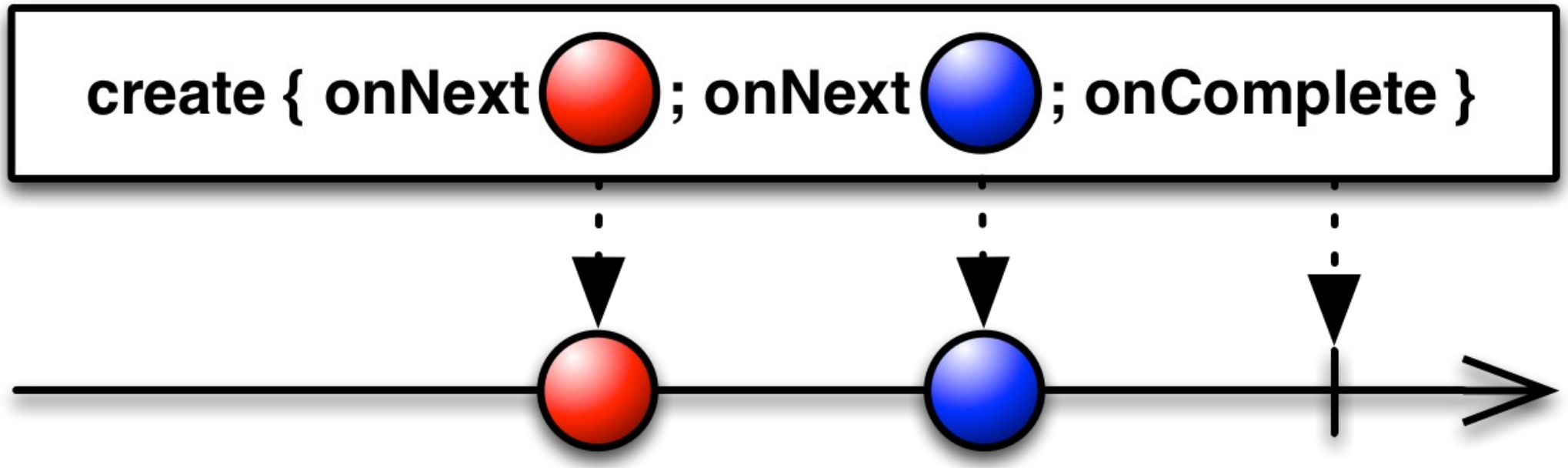
Marble Diagrams

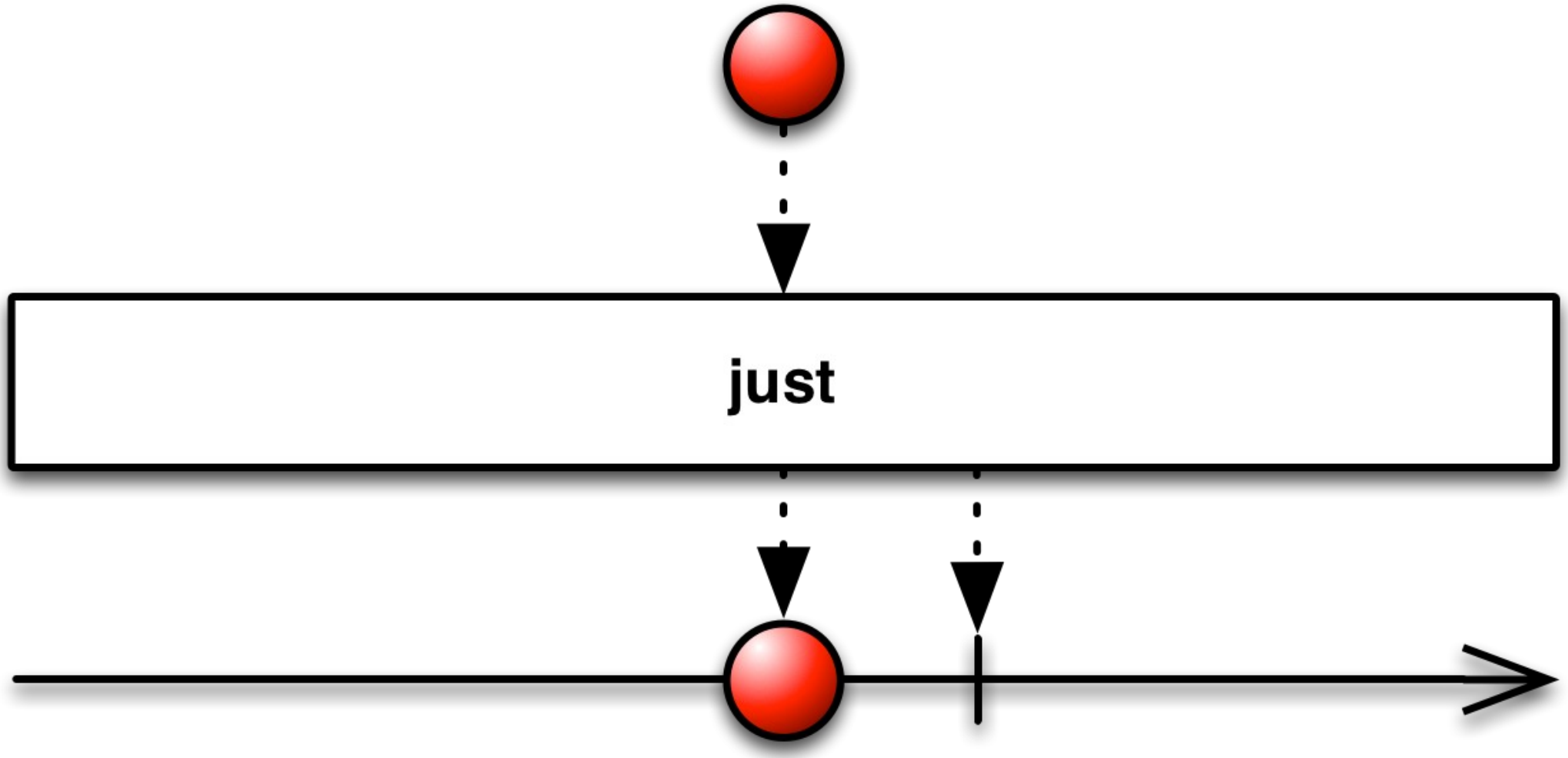


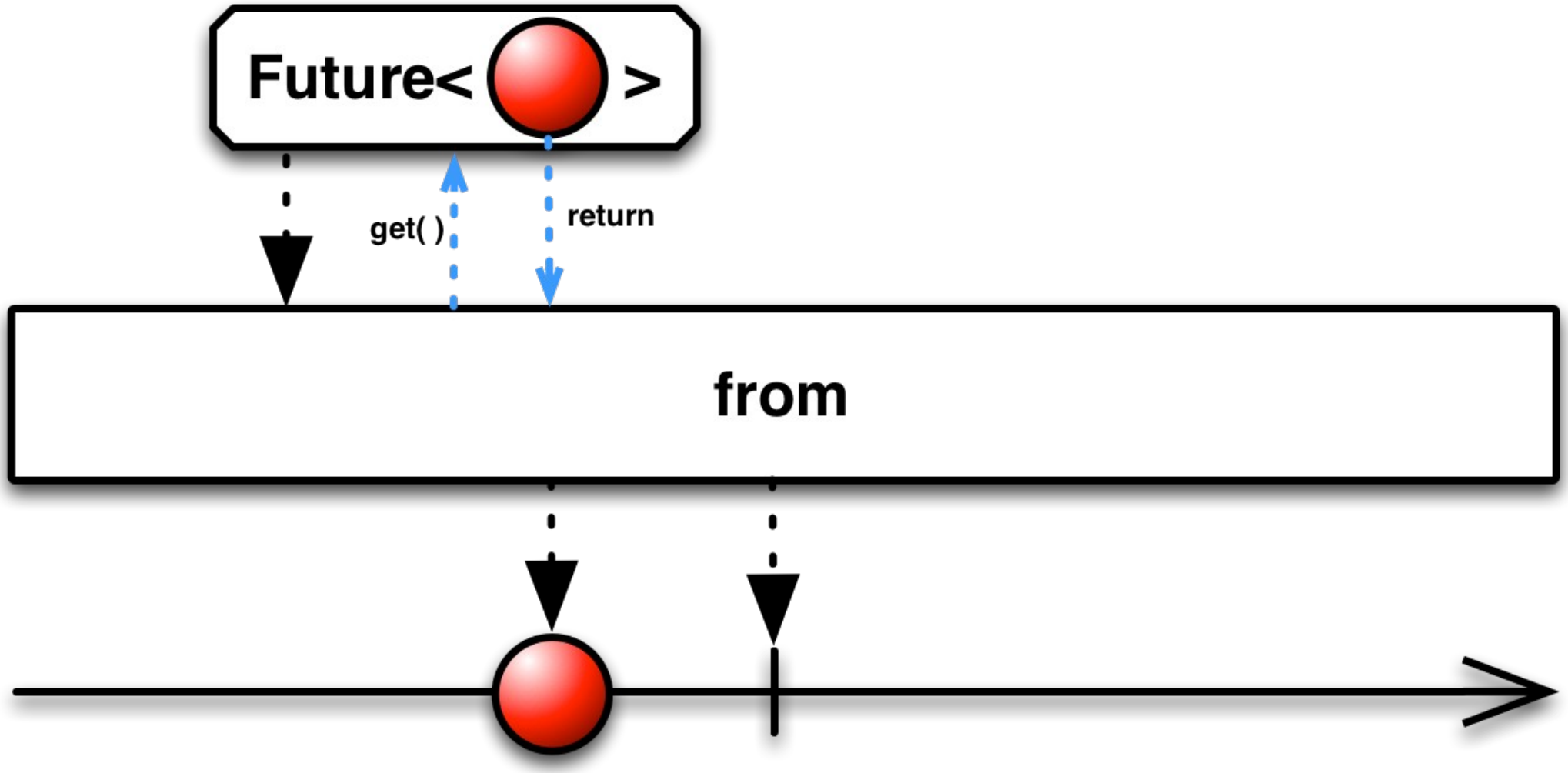
Find all your favorite operators at:

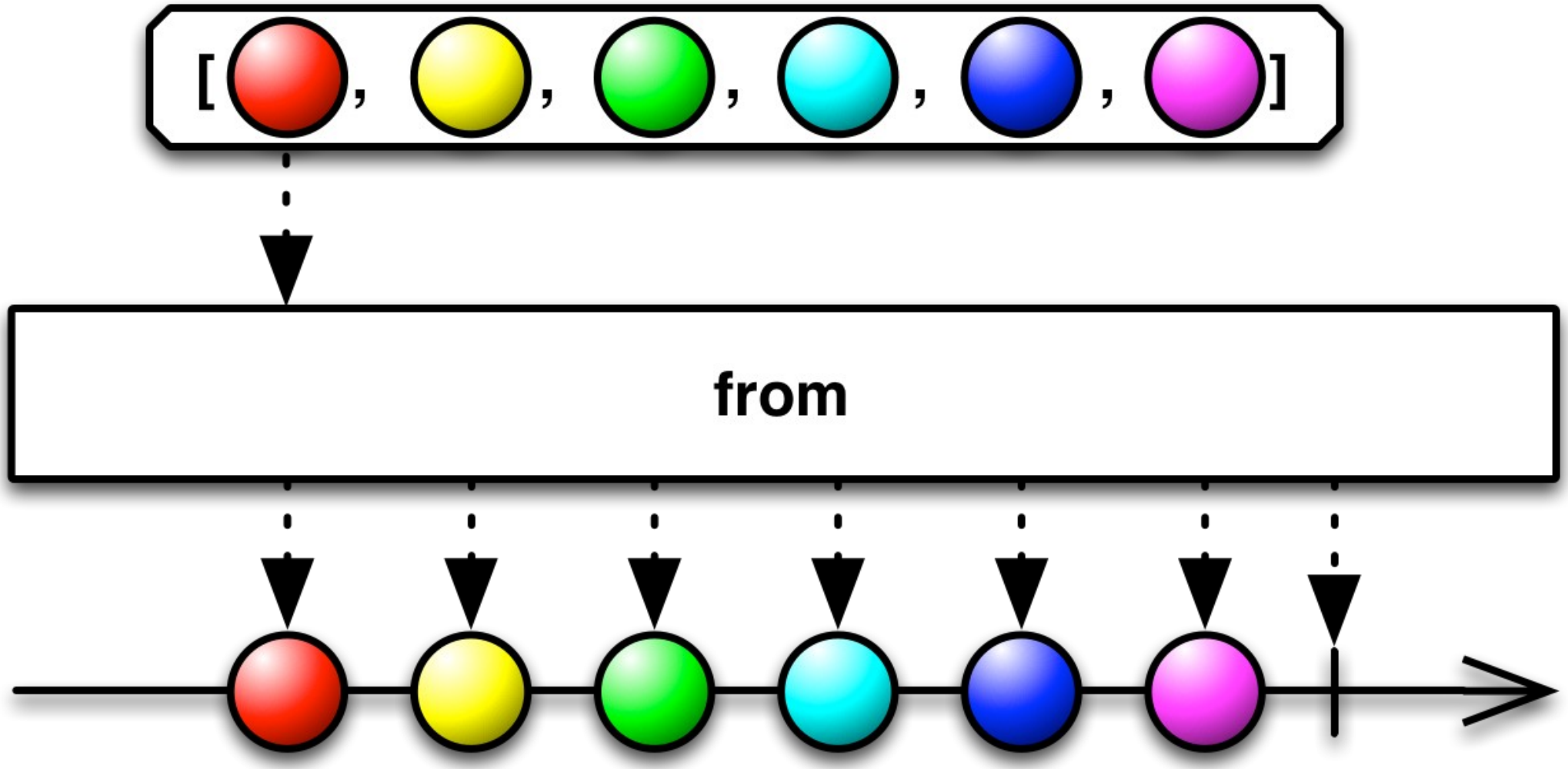
http://reactivex.io/documentation/operators

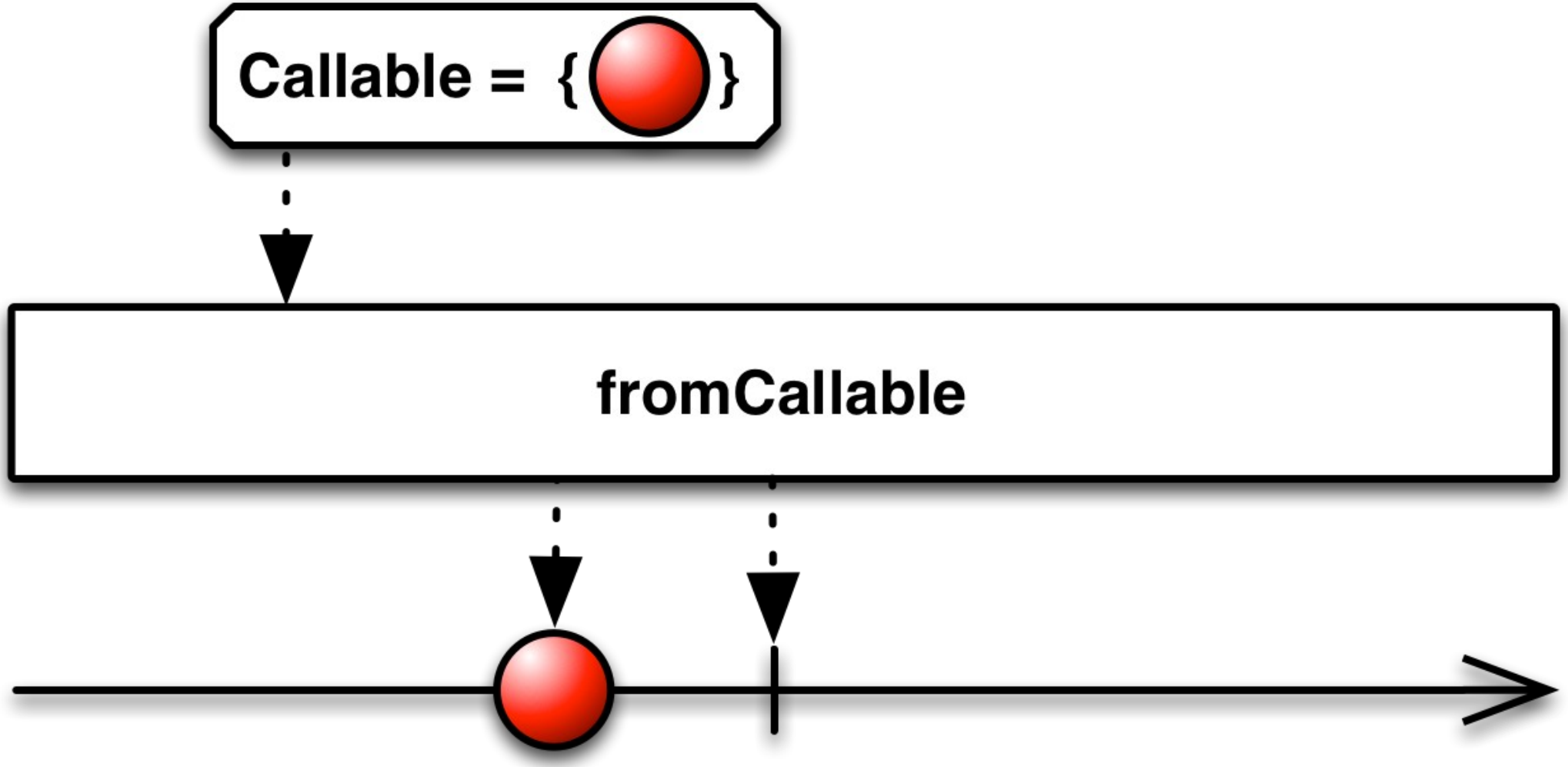
Creation of Observables


create { onNext ; onNext ; onComplete }


just


Future< >
get( )
return
from


[, , , , ,]
from


Callable = {}
fromCallable

Creation of Observables

`Observables` can be created from an `Iterables`, `Futures`, or `Functions`.

Operators

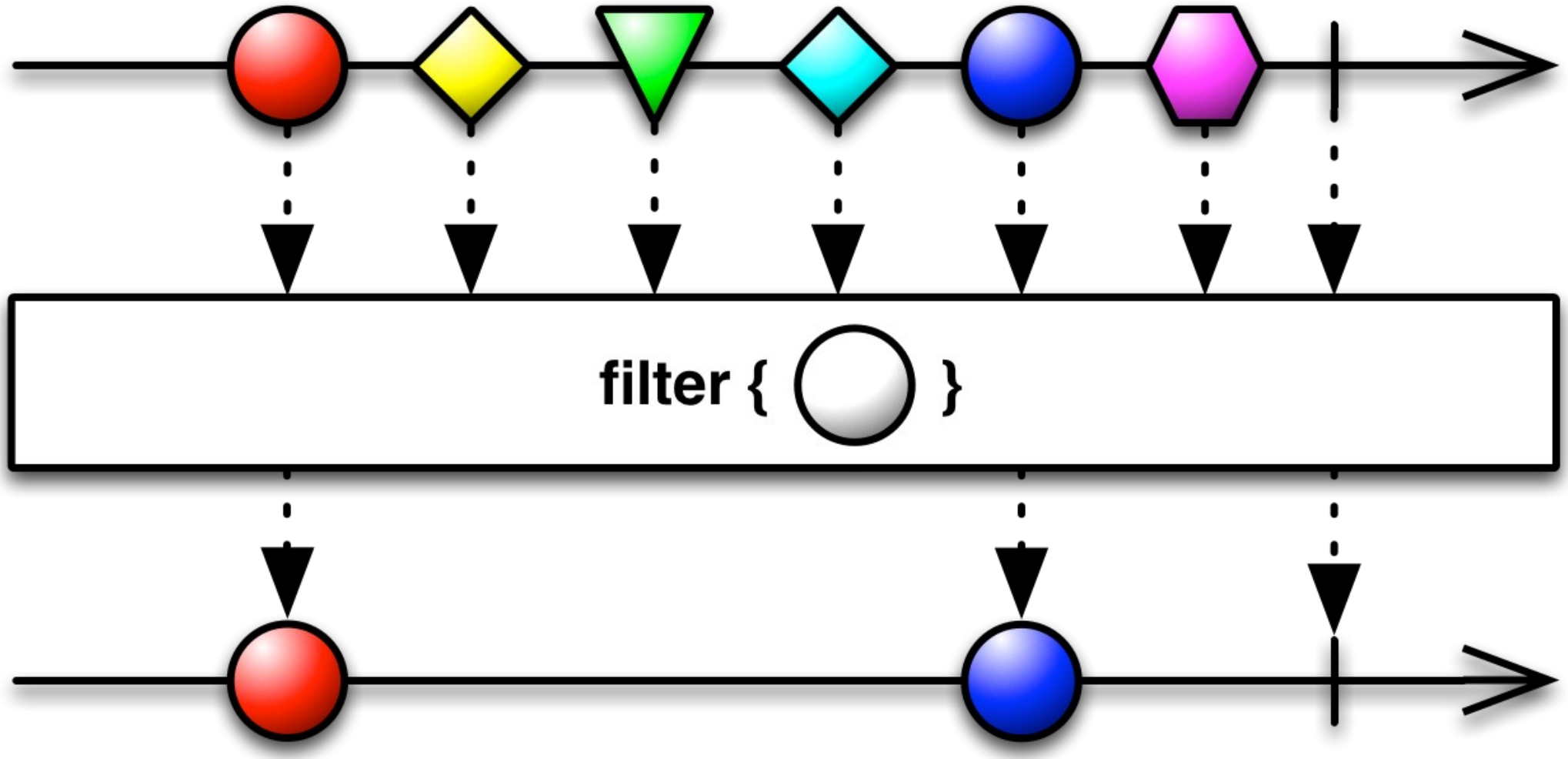
Demo: Creation of Observables

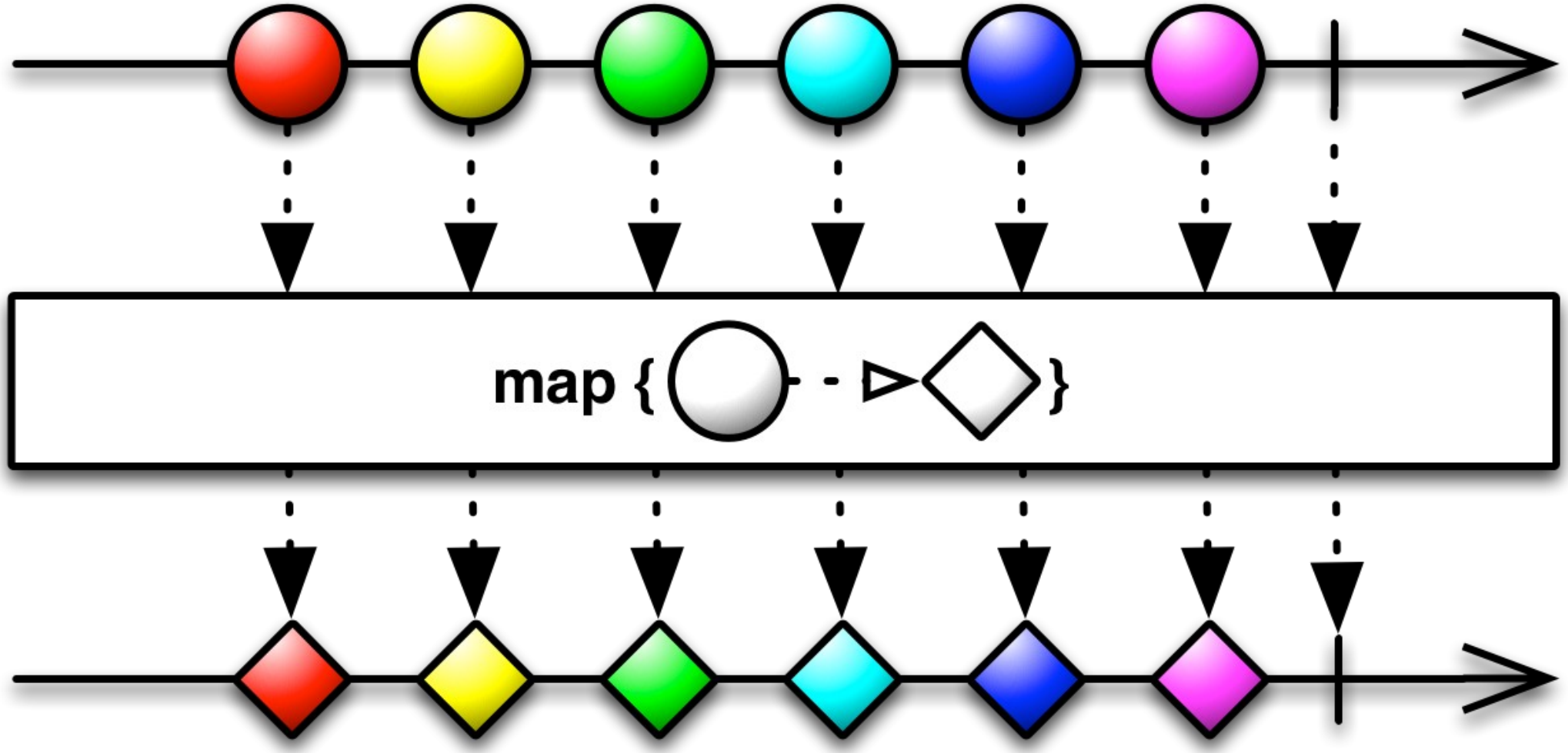
Operators

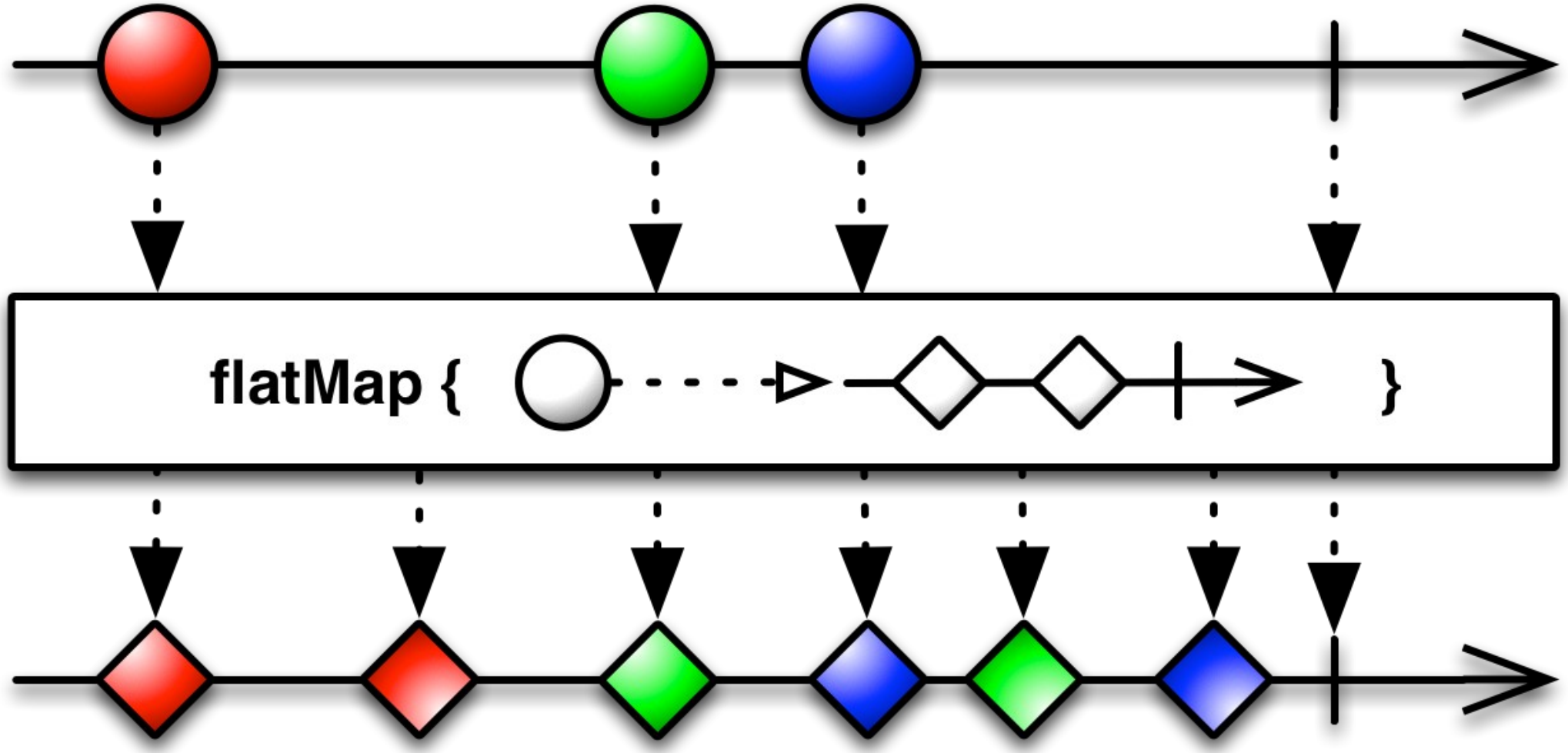
Find all your favorite operators at:

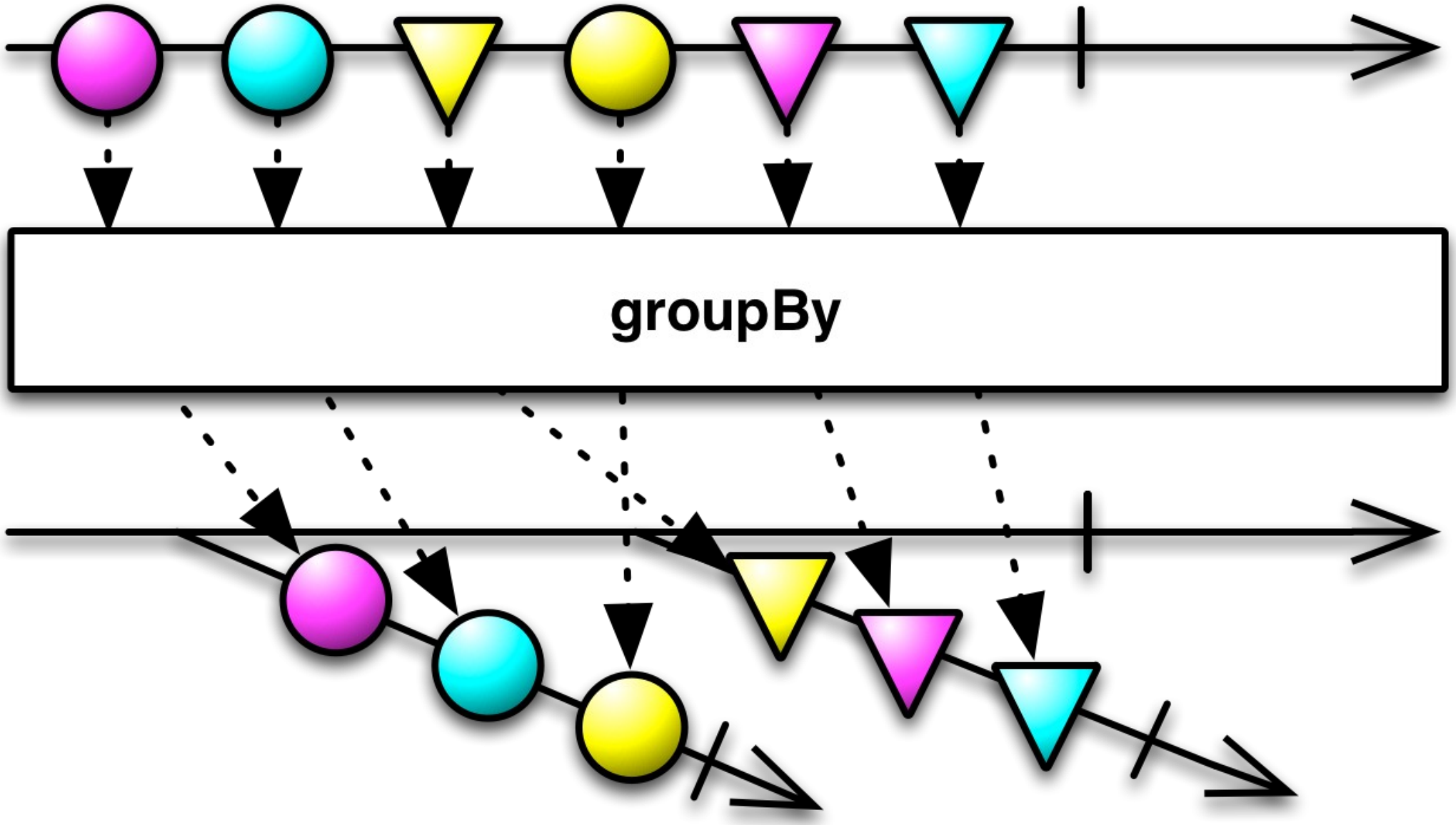
http://reactivex.io/documentation/operators

Standard Functional Operators


filter { }

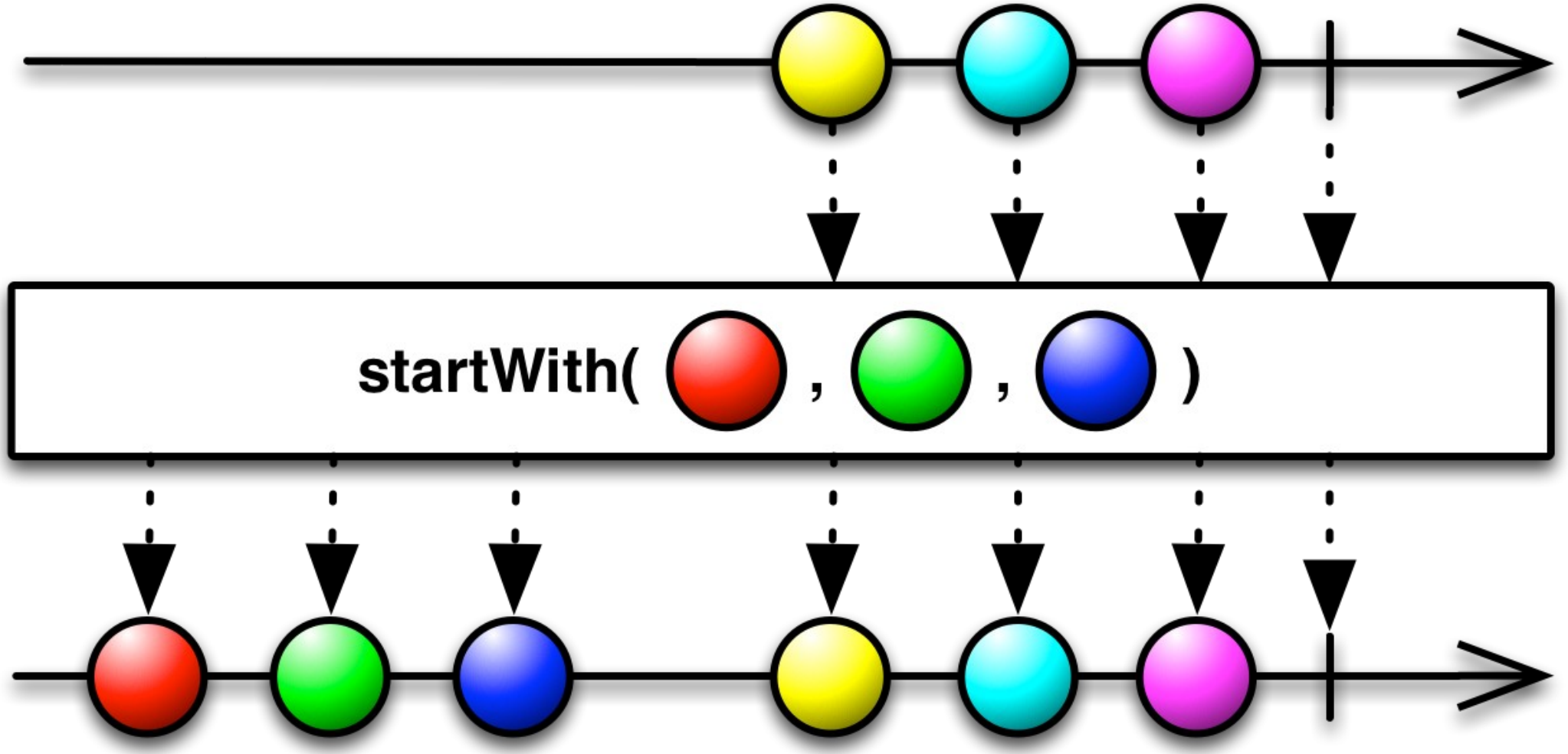

map {

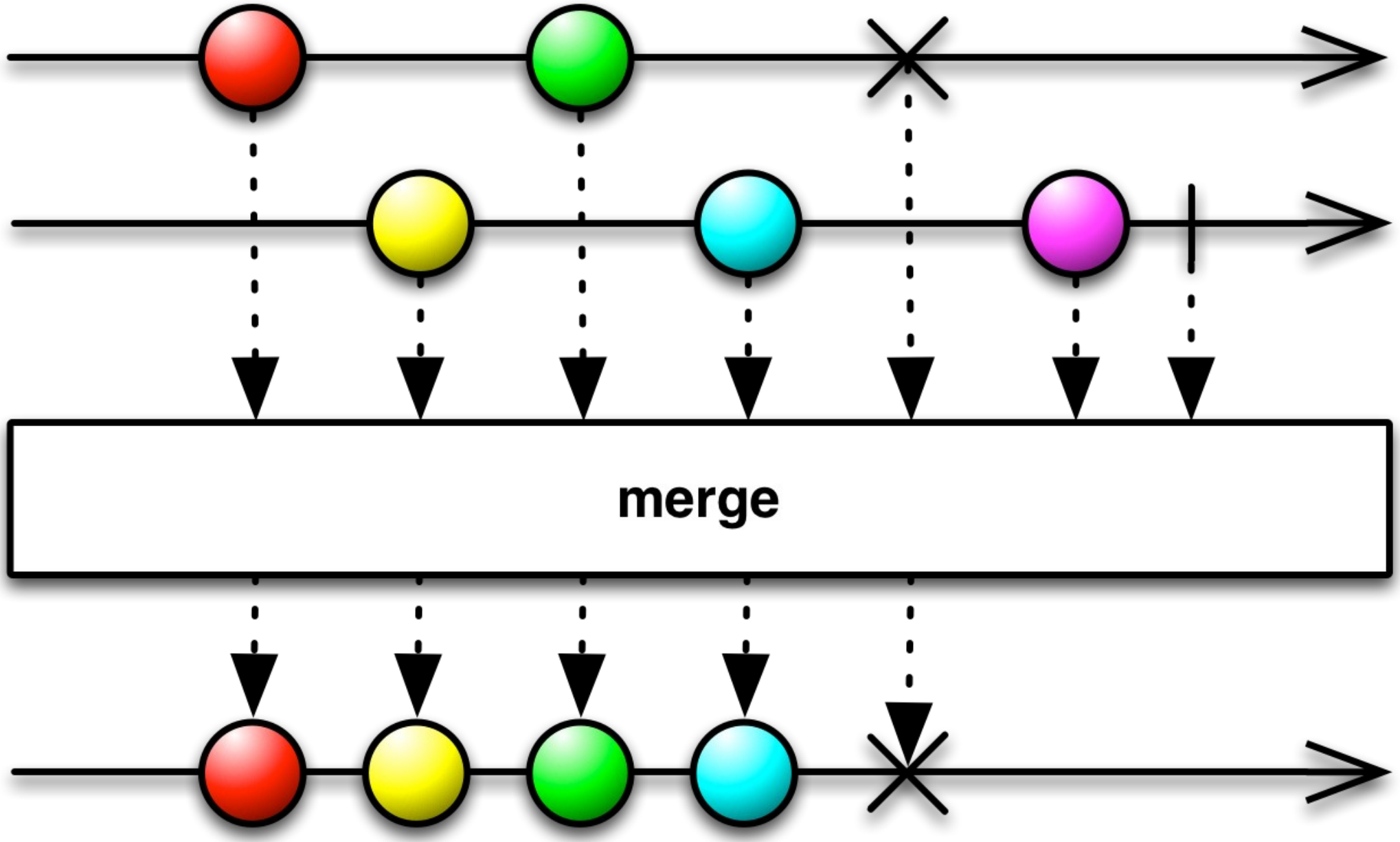

flatMap {
}

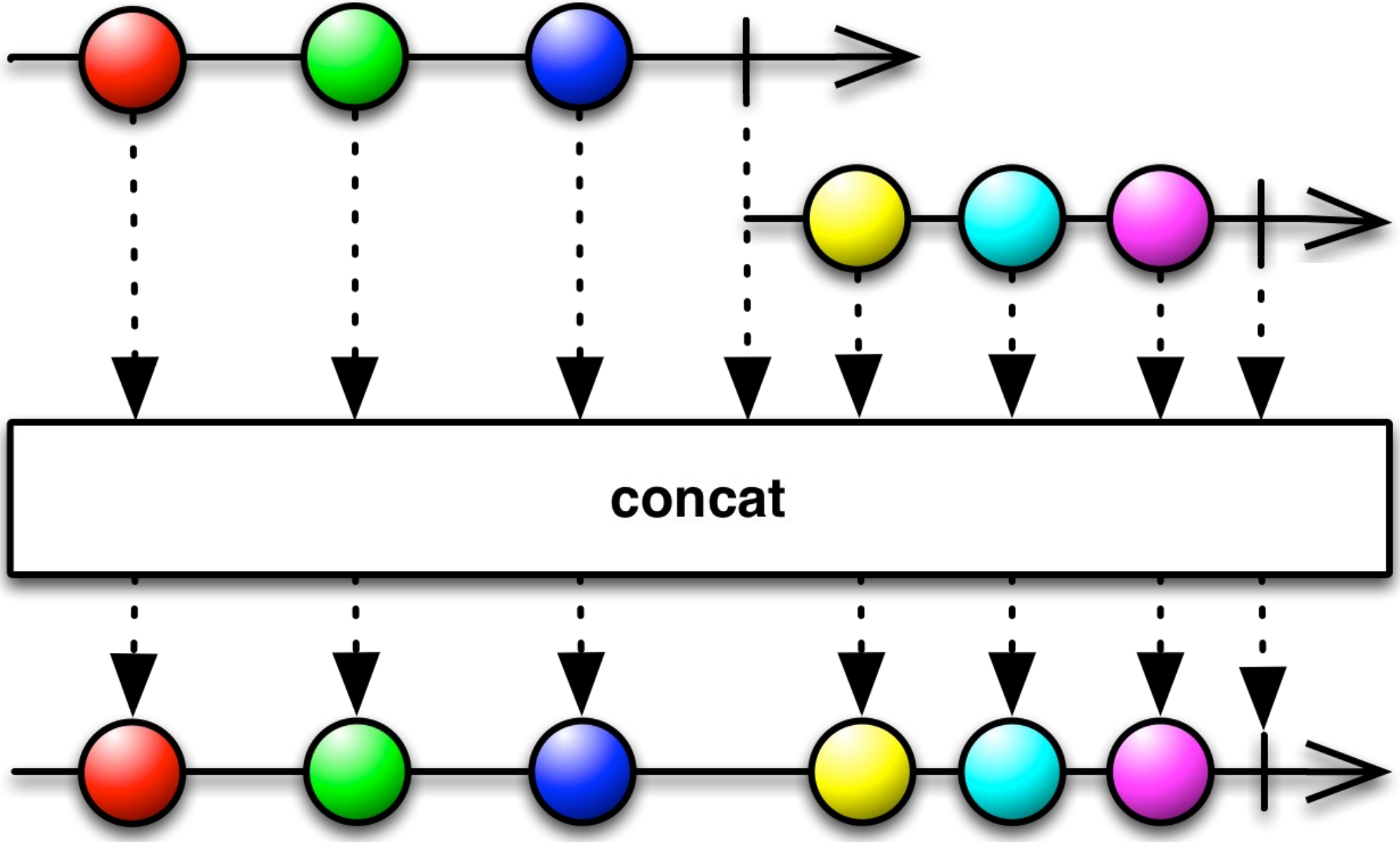

groupBy

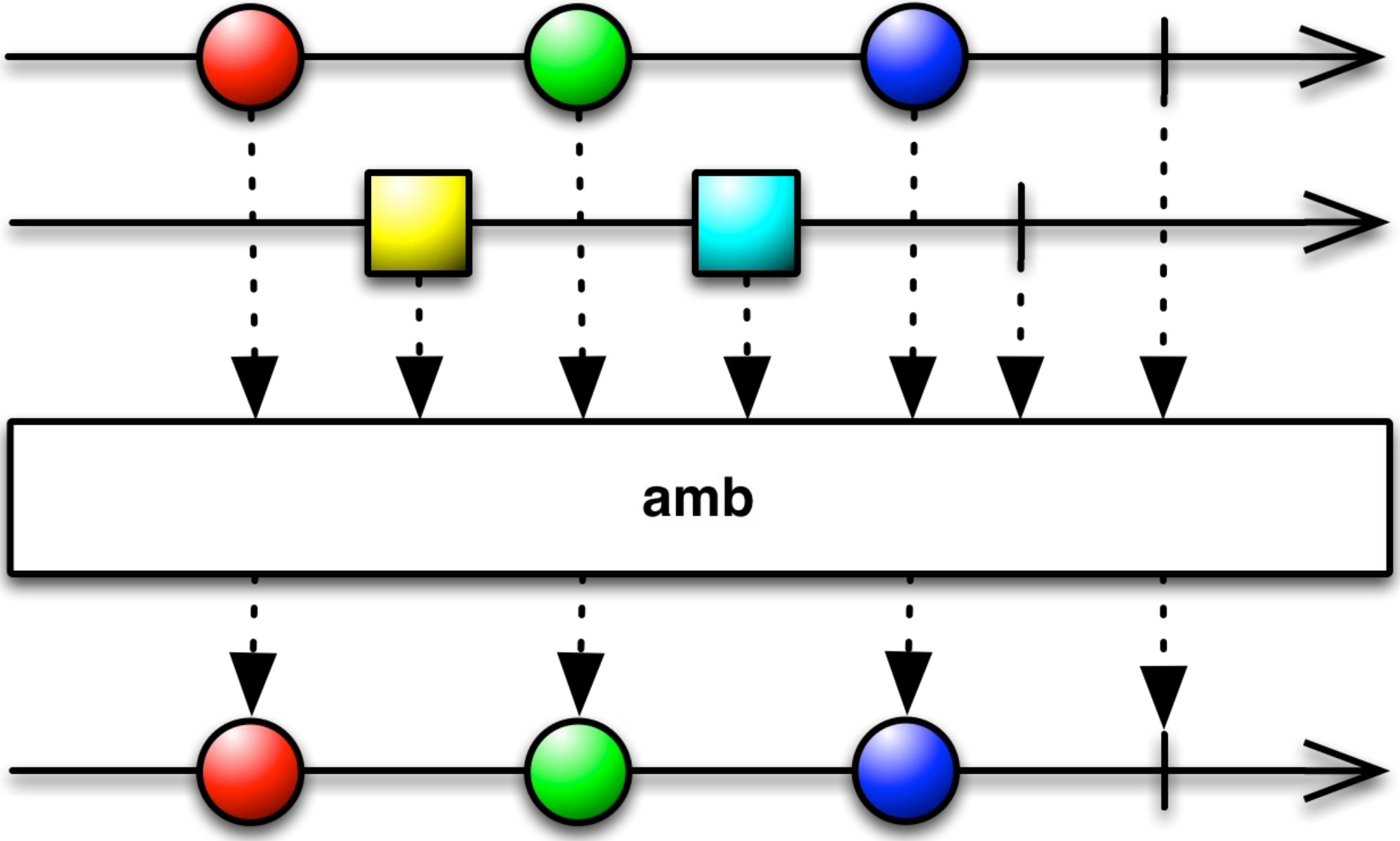
Demo: Standard Functional Operators

Combining Observable Operators


startWith(, ,)

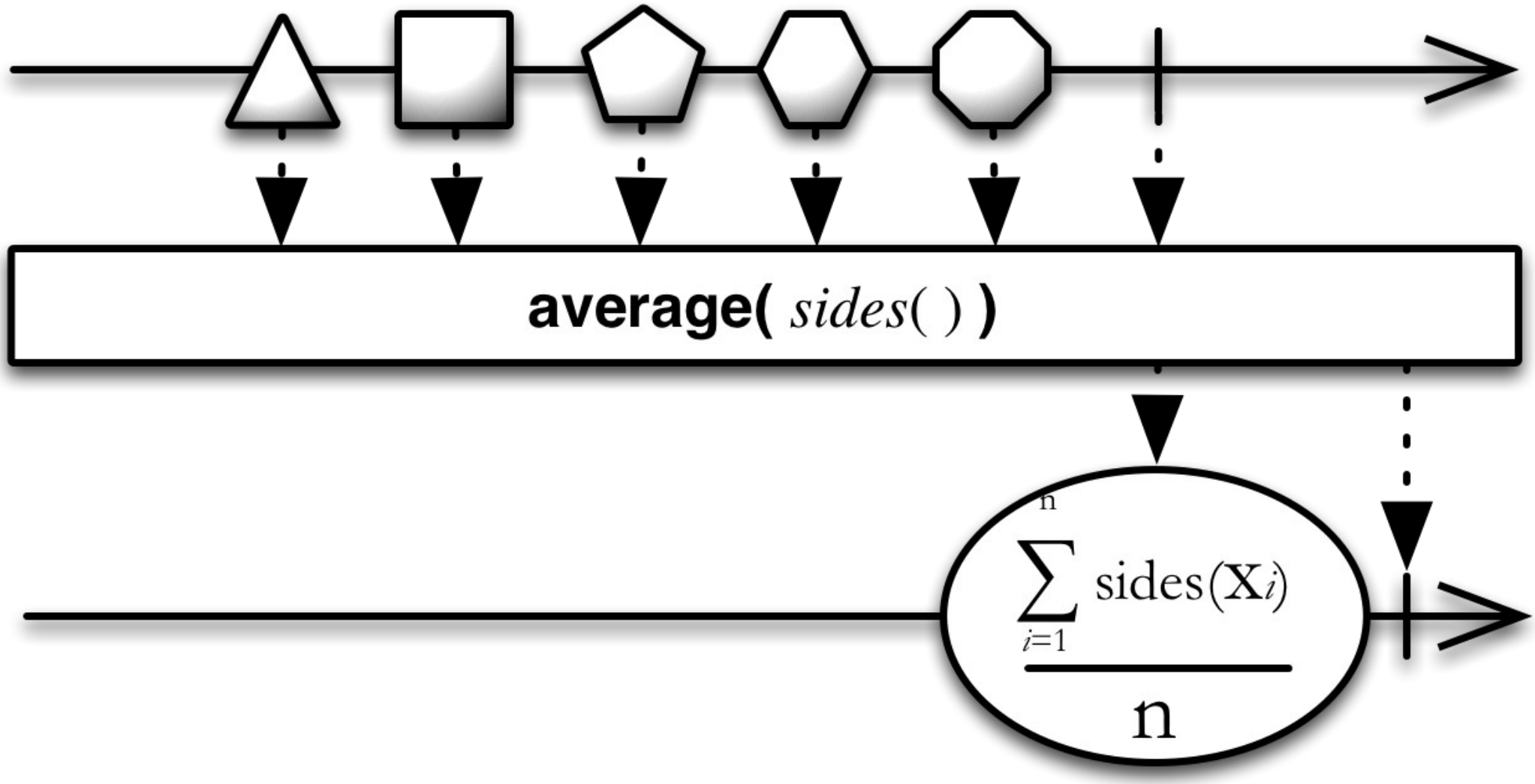

merge

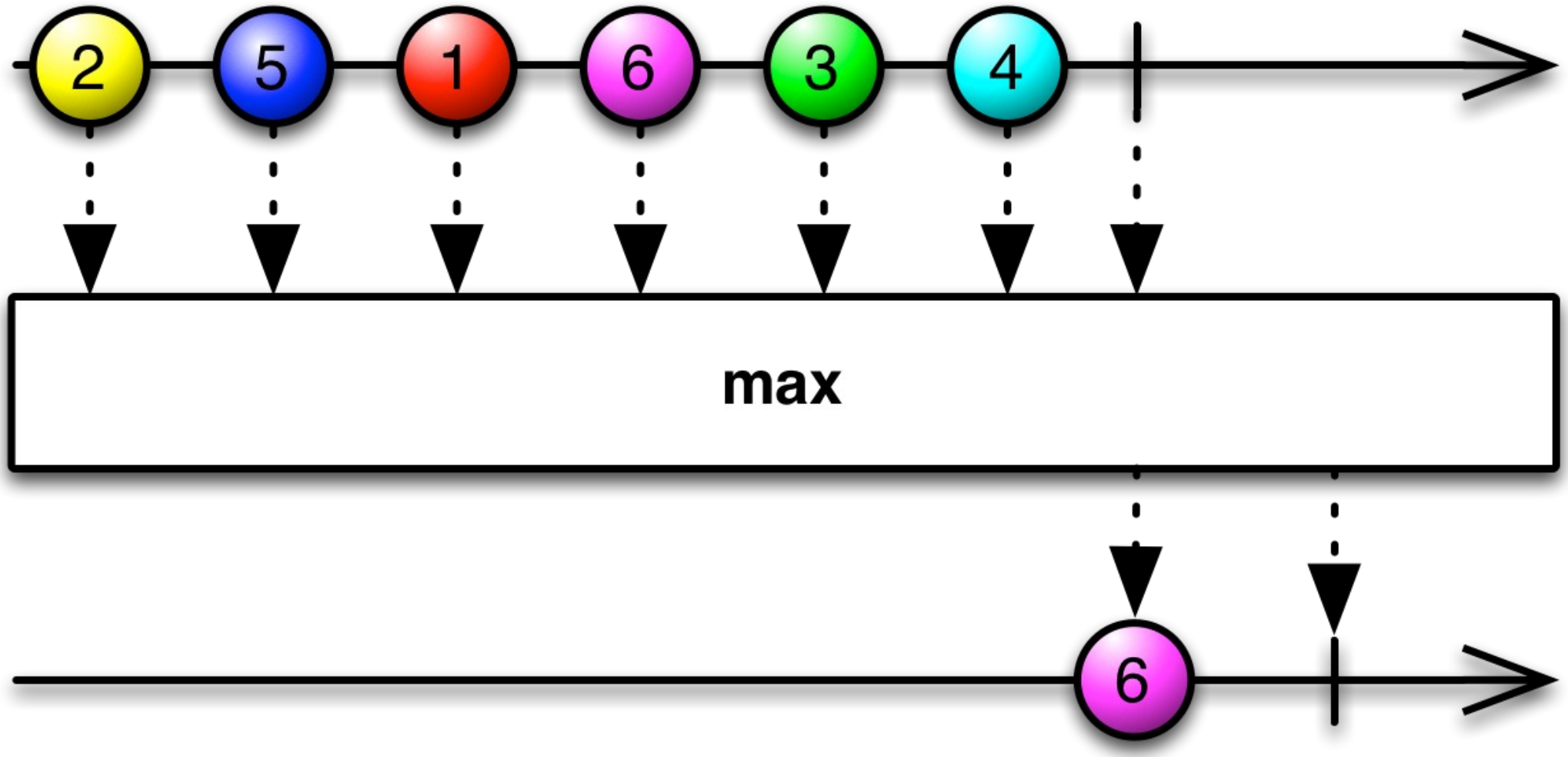

concat

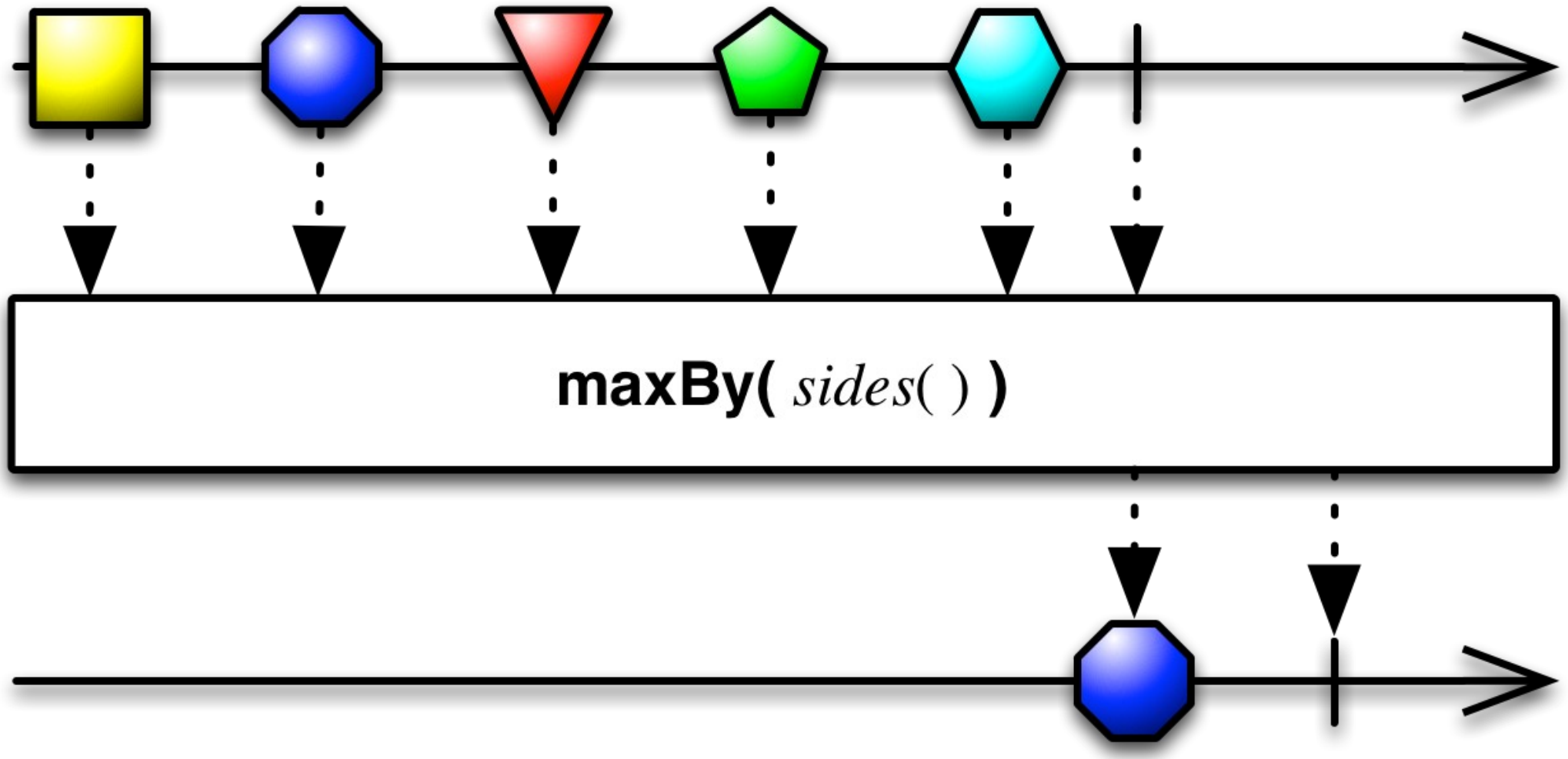

amb

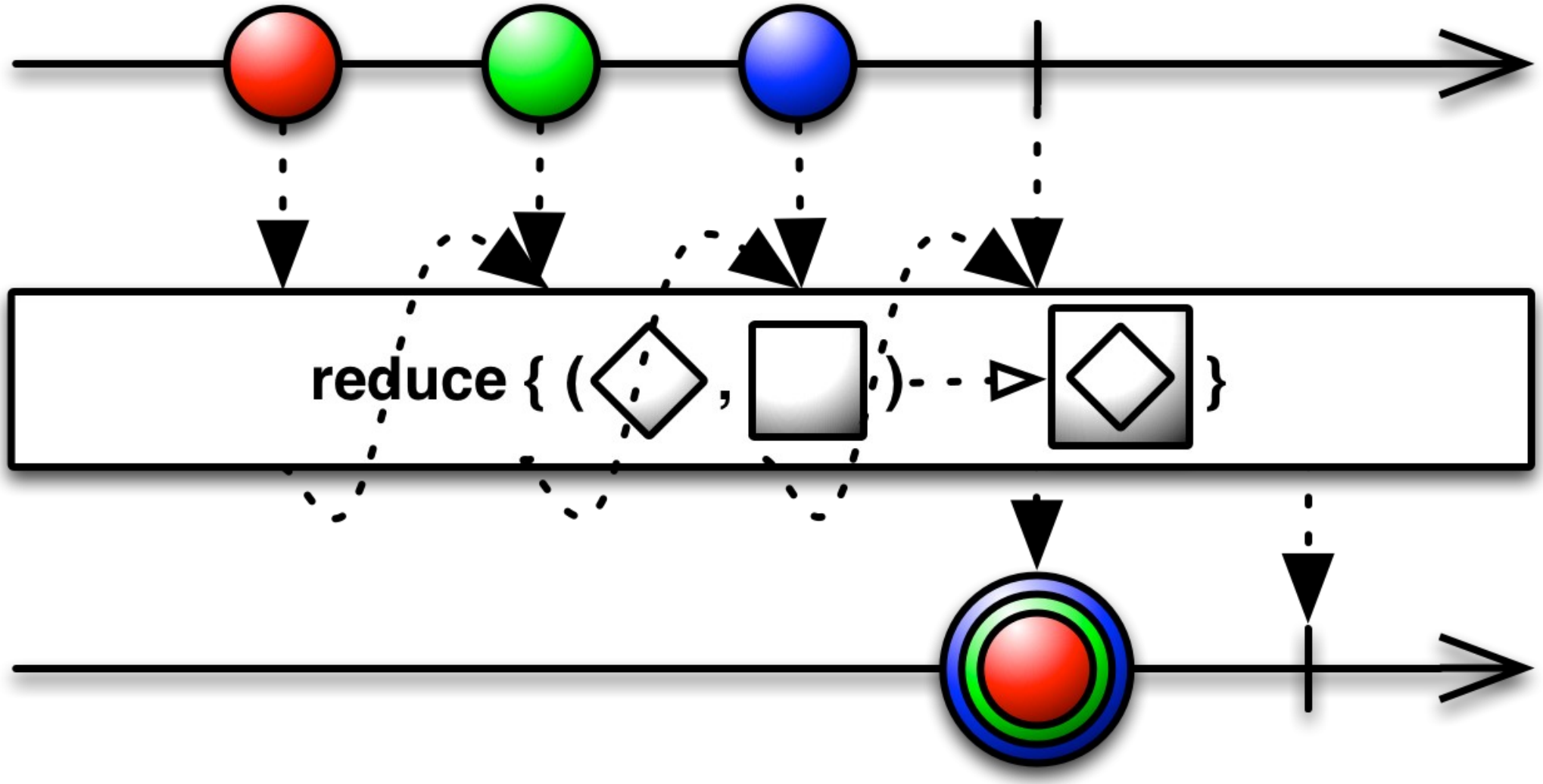
Demo: Combining Observable Operators

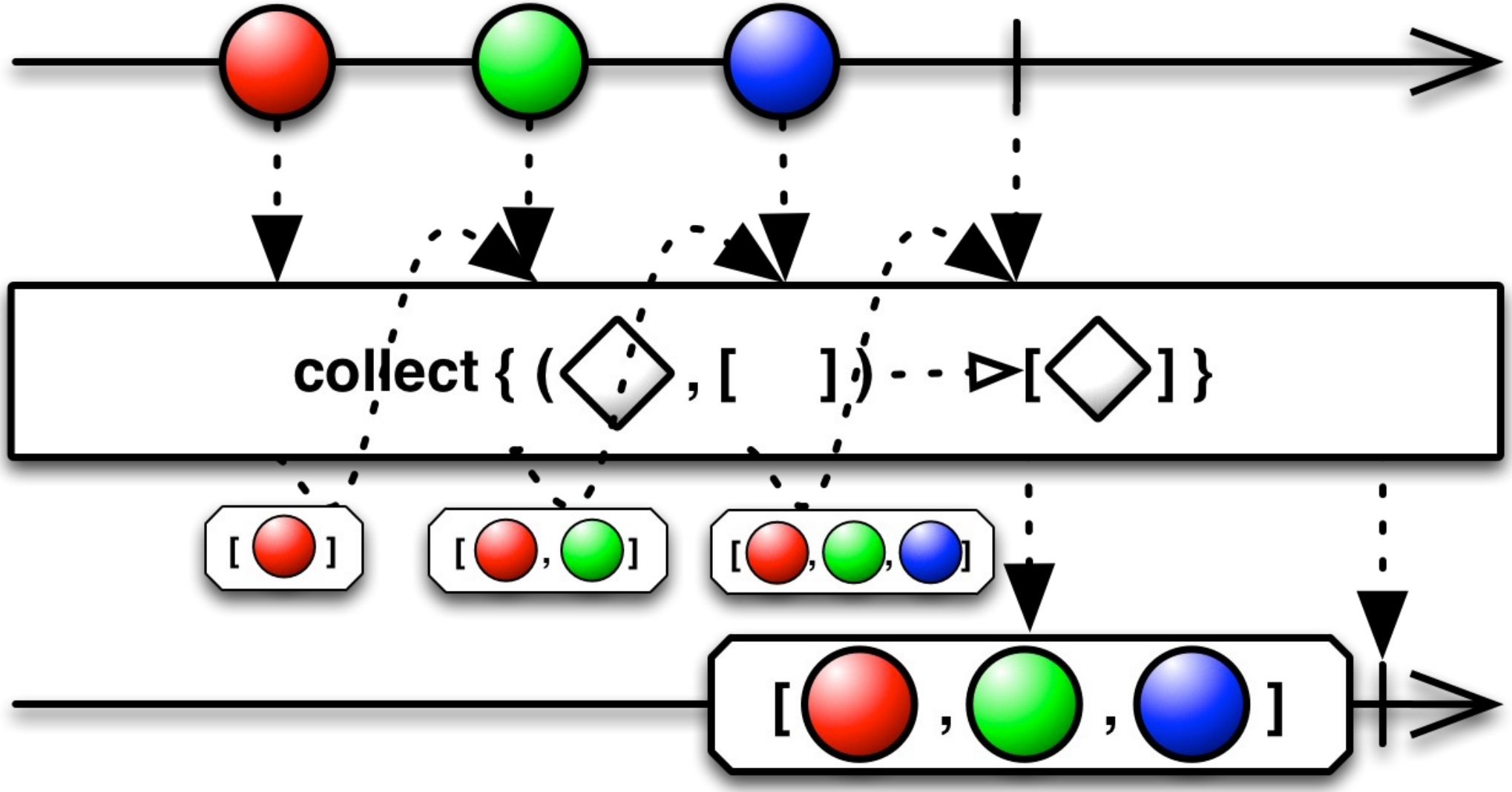
Mathematics, Aggregate, & Reductions

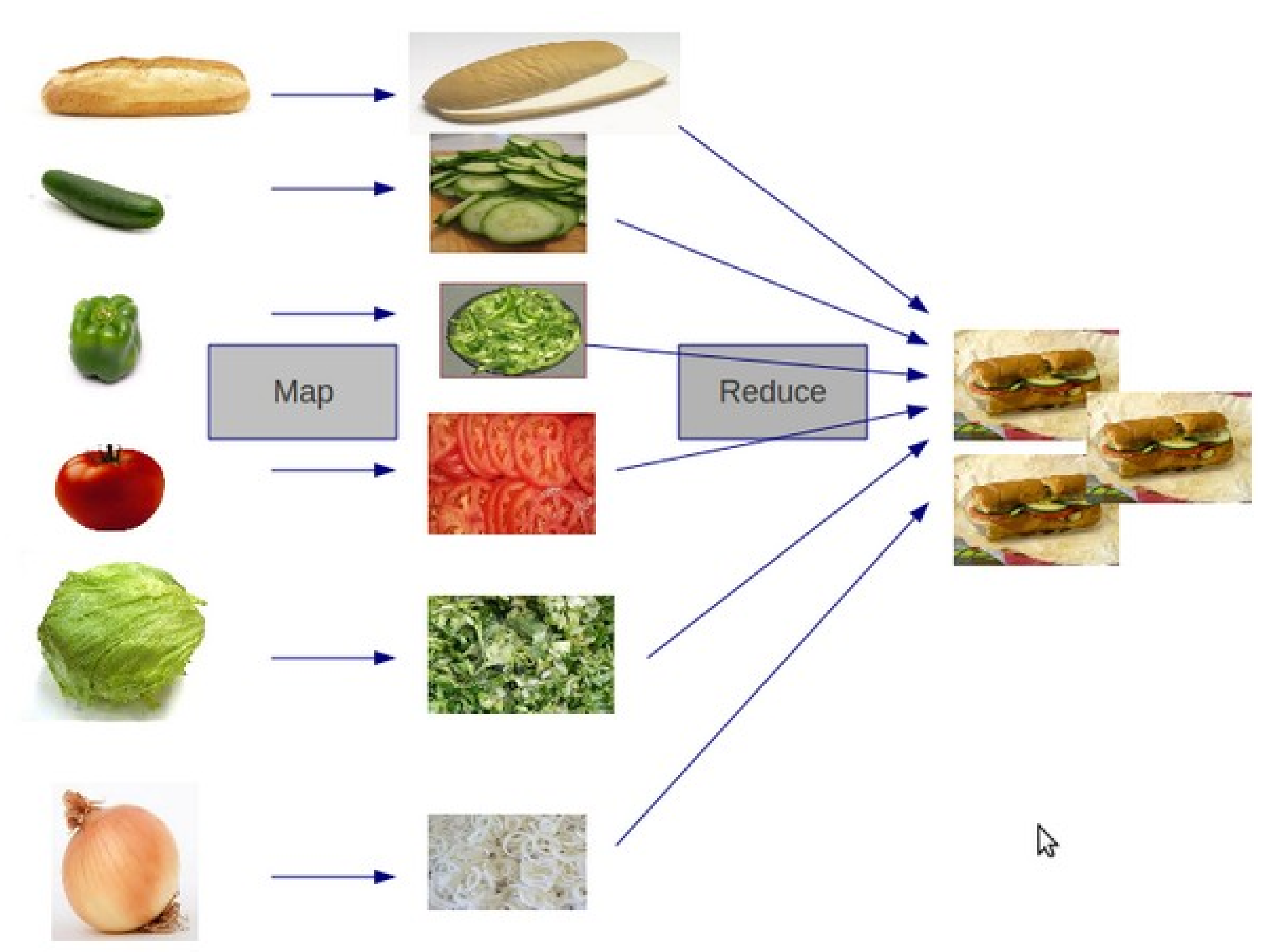

average(sides( ))
n
∑ sides(Xi)
i=1
n


2
5
1
6
3
4
max
6


maxBy(sides( ))


reduce { (,)- - }


collect { (, [ ]) - - ▷[ ] }
[ ]
[,]
[, ,]
[, ,]


Map
Reduce

Demo: Mathematics, Aggregate, & Reductions

Demo: Converting Observables (toMultimap, toMap, ...)


UNDER BACKPRESSURE

Demo: Backpressure

Schedulers

RX Single Threads

All threads in RXJava are single threaded by default

RX Scheduler

Schedulers assign a different a different Thread for different purposes

RX subscribeOn()

`subscribeOn(...)` assigns a `Scheduler` on the subscription side of invocation

RX observeOn()

`observeOn(...)` assigns a `Scheduler` on the subscription side of invocation

RX immediate()

Creates and returns a Scheduler that executes work immediately on the current thread.

RX trampoline()

Creates and returns a Scheduler that queues work on the current thread to be executed after the current work completes.

RX newThread()

Creates and returns a Scheduler that creates a new Thread for each unit of work.

RX computation()

Creates and returns a `Scheduler` intended for computational work. This can be used for event-loops, processing callbacks and other computational work.

RX computation()

Do not perform IO-bound work on this scheduler. Use `Schedulers.io()` instead.

RX computation()

Backed by a bounded thread-pool with size equal to the number of available processors.

RX computation()

Avoid big performance hits by thread creation overhead and context switching overhead

http://stackoverflow.com/questions/31276164/rxjava-schedulers-use-cases

RX io()

Creates and returns a Scheduler intended for IO-bound work. This can be used for asynchronously performing blocking IO. Do not perform computational work on this scheduler. Use Schedulers.computation() instead.

RX io()

The implementation is backed by an Executor thread-pool that will grow as needed.

RX io()

This can be used for asynchronously performing blocking IO.

RX io()

Do not perform computational work on this scheduler. Use `Schedulers.computation()` instead.

RX Android Schedulers

Android based Schedulers.
`mainThread(...)` to perform tasks on the Android main thread.

https://github.com/ReactiveX/RxAndroid

RX JavaFX Schedulers

`JavaFxScheduler.getInstance()`
used to update UI components on the JavaFX Event Dispatch Thread

https://github.com/ReactiveX/RxJavaFX

Demo: Schedulers

Hot & Cold Observables

PSY- Gangnam Style (Official Music Video)

DanceGangnamStyle

Subscribe 46,824

44,983,176

Add to Share More

88,792 23,430

Southwest Florida Eagle Cam

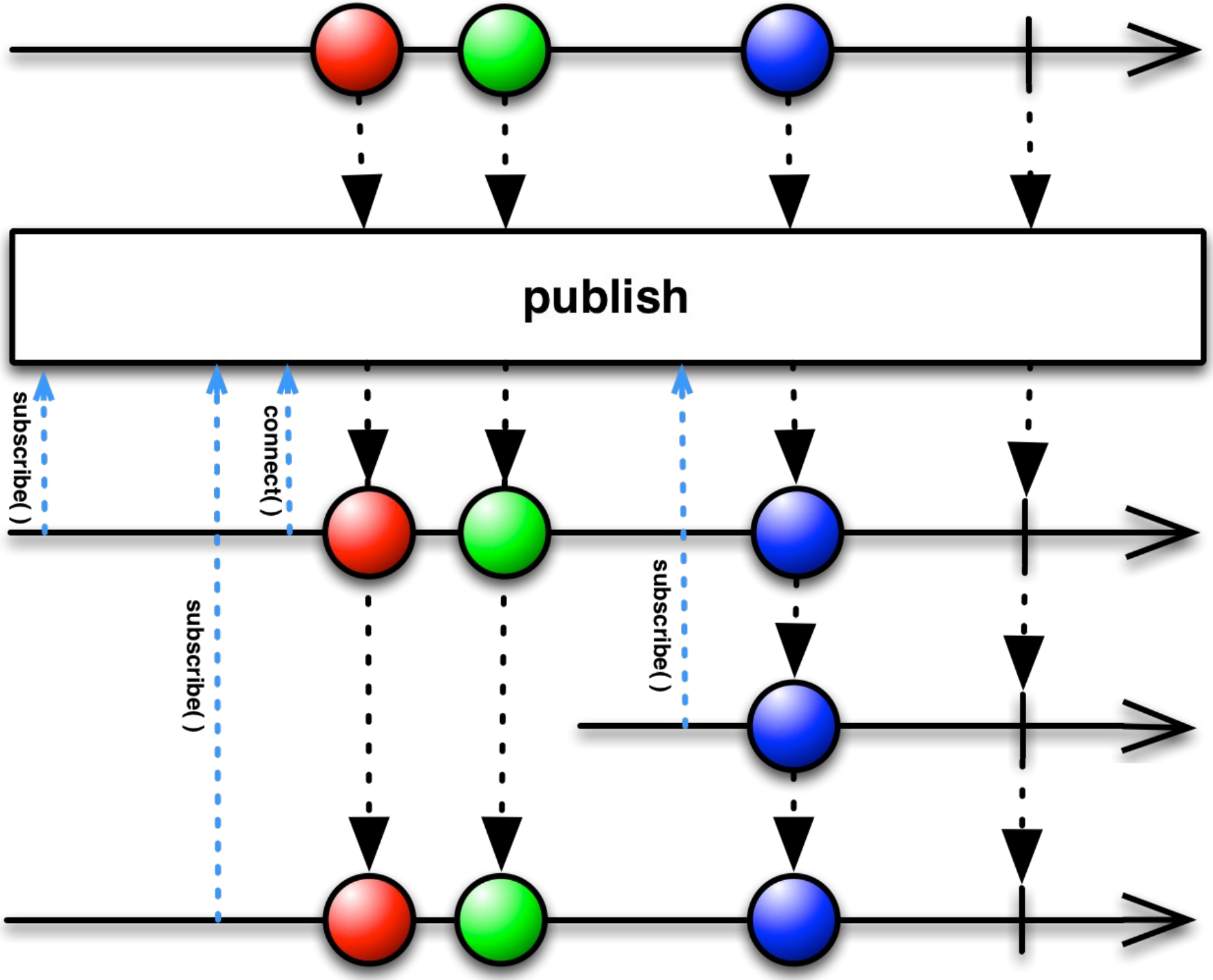
Southwest Florida Eagle Cam

Subscribe 6,394

2592 watching now

Add to Share More

4,599 392


publish
subscribe( )
connect( )
subscribe( )
subscribe( )

RX ConnectableObservables

`publish()` returns a `ConnectableObservable`

RX autoConnect

`autoConnect()`
Returns an Observable that automatically connects to this ConnectableObservable when the first Subscriber subscribes.

RX connect

`connect()`
Instructs the `ConnectableObservable` to begin emitting the items from its underlying `Observable` to its `Subscribers`.

RX refCount()

`refCount()`
Returns an `Observable` that stays connected to this `ConnectableObservable` as long as there is at least one subscription to this `ConnectableObservable`.

Demo: Hot/Cold Observables

Thank You