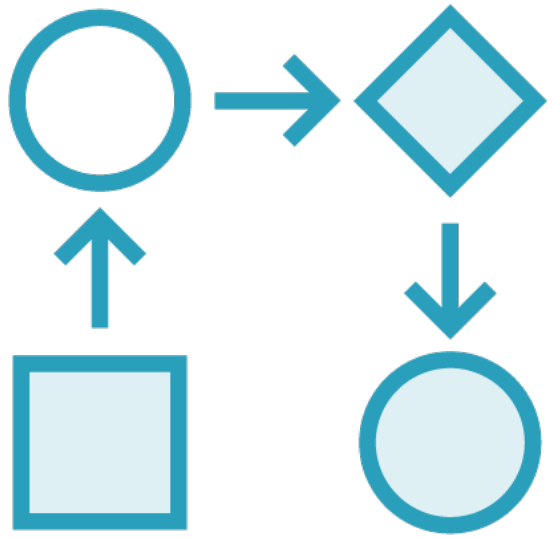
Filtering and Controlling the Sequences

Edin Kapić

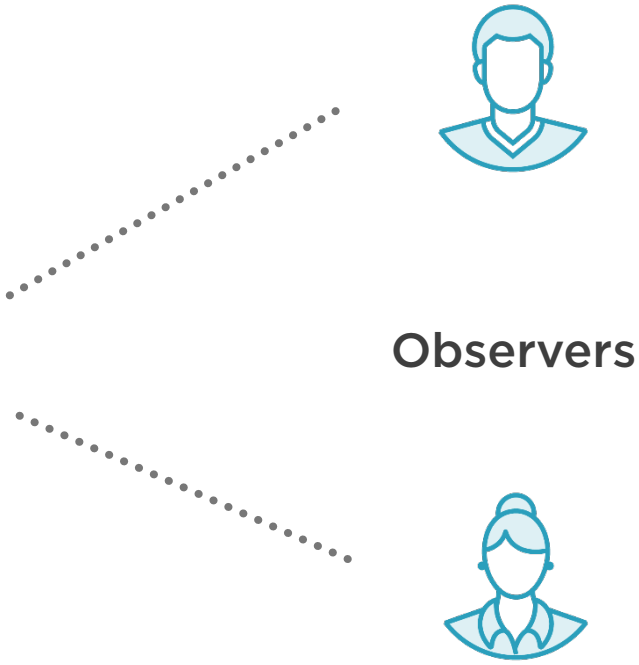
@ekapic www.edinkapic.com

What Have You Seen so Far

Observable



LINQ Operators



Observers

Types of Operators in Rx

Filtering

Combining

Aggregating

Utility

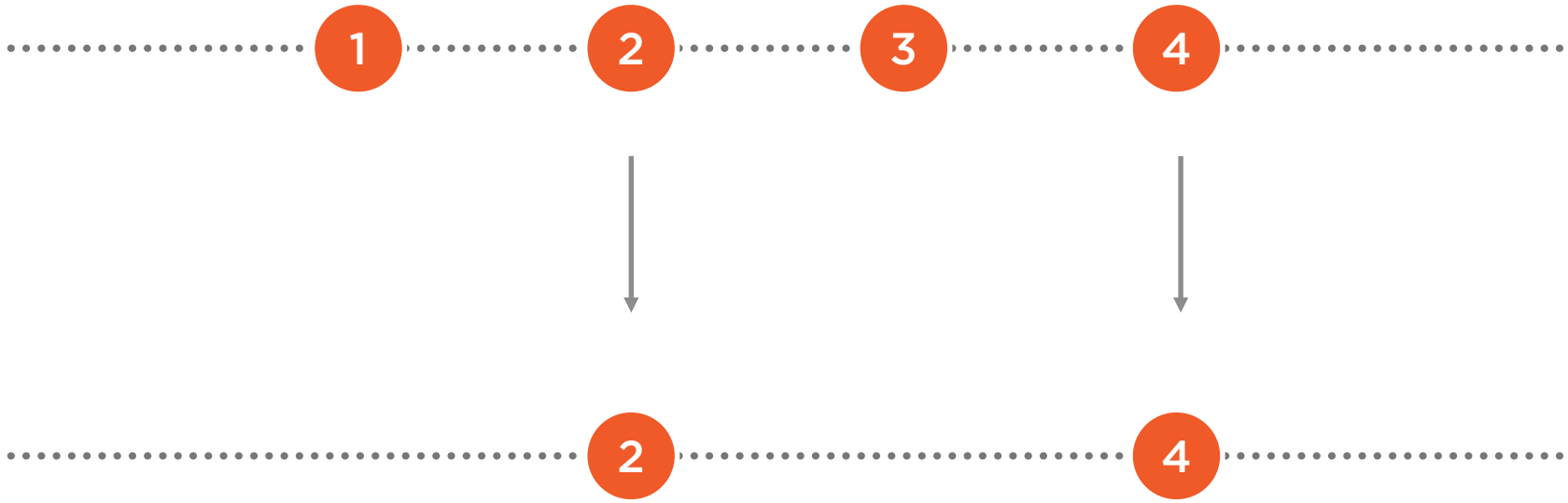
```
var sequence = Observable.Create(...);
var anotherSequence = sequence.Where(x => x.Name == "Rx");
anotherSequence.Subscribe(...);
```

Filtering Operators

Used to remove undesired elements in the sequence

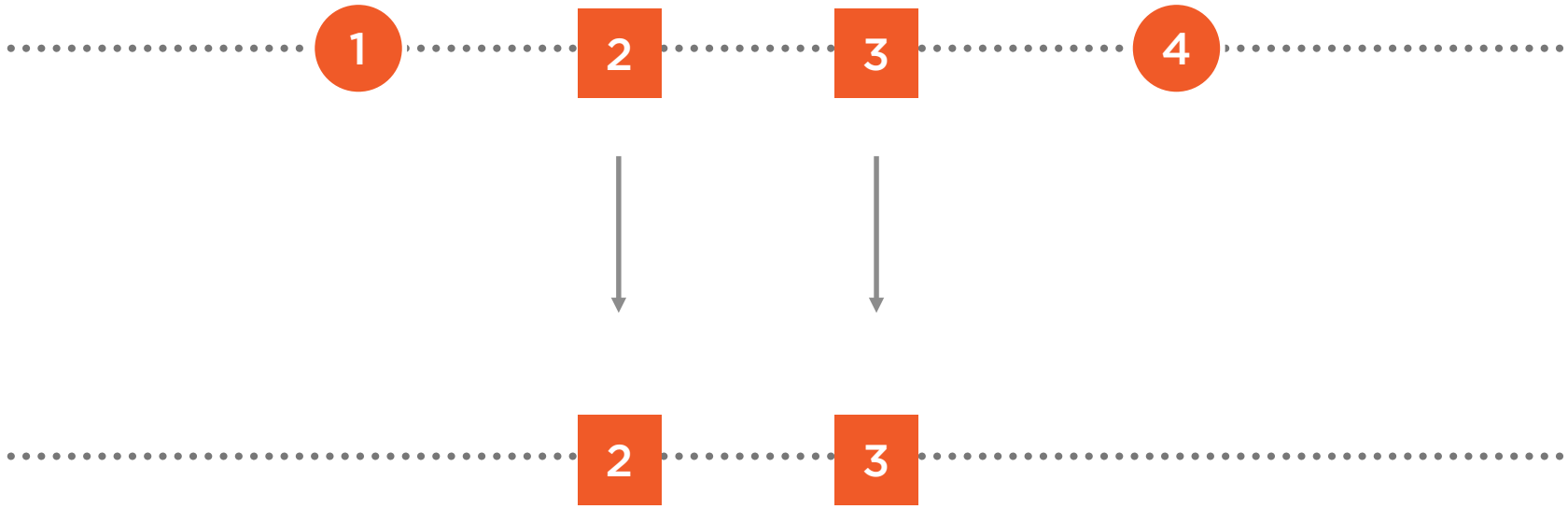
Filter Elements by Condition

Where




```
.Where( x => x % 2 == 0);
```

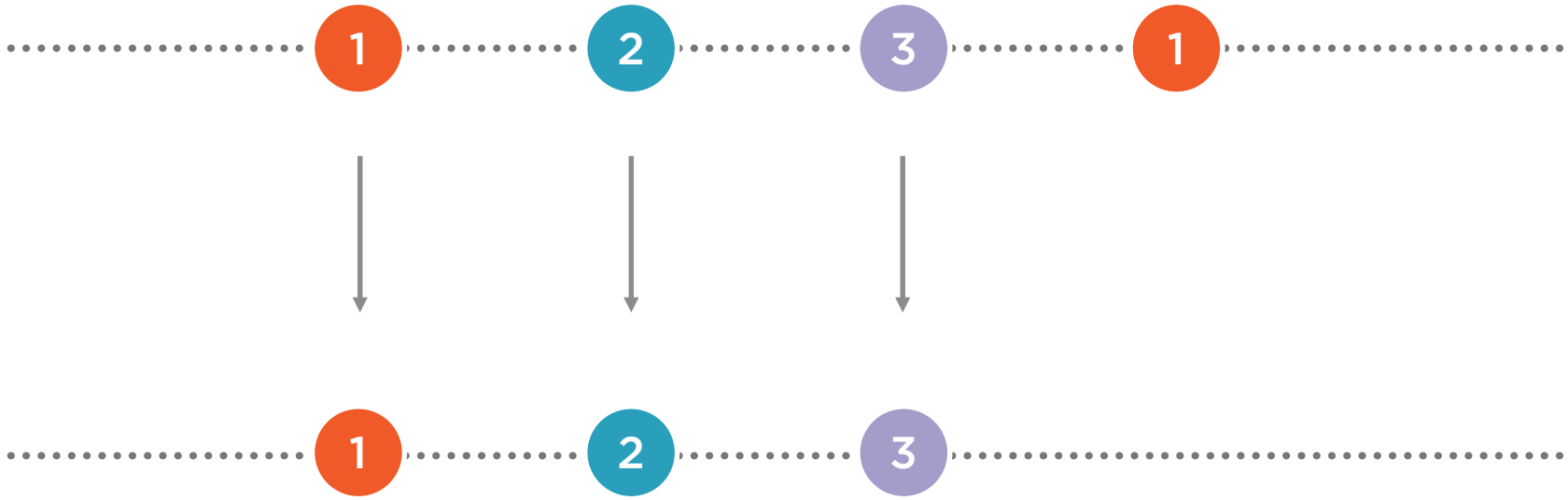
OfType<T>




```
.OfType<Square>();
```

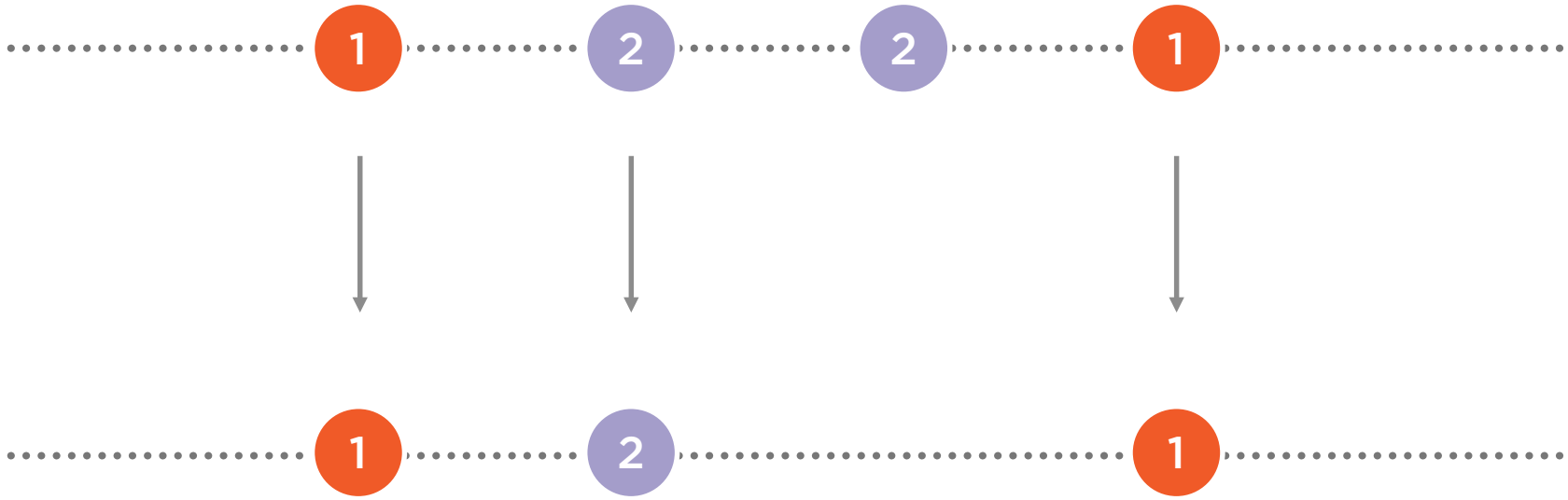
Filter Duplicate Elements

Distinct




```
.Distinct();
```

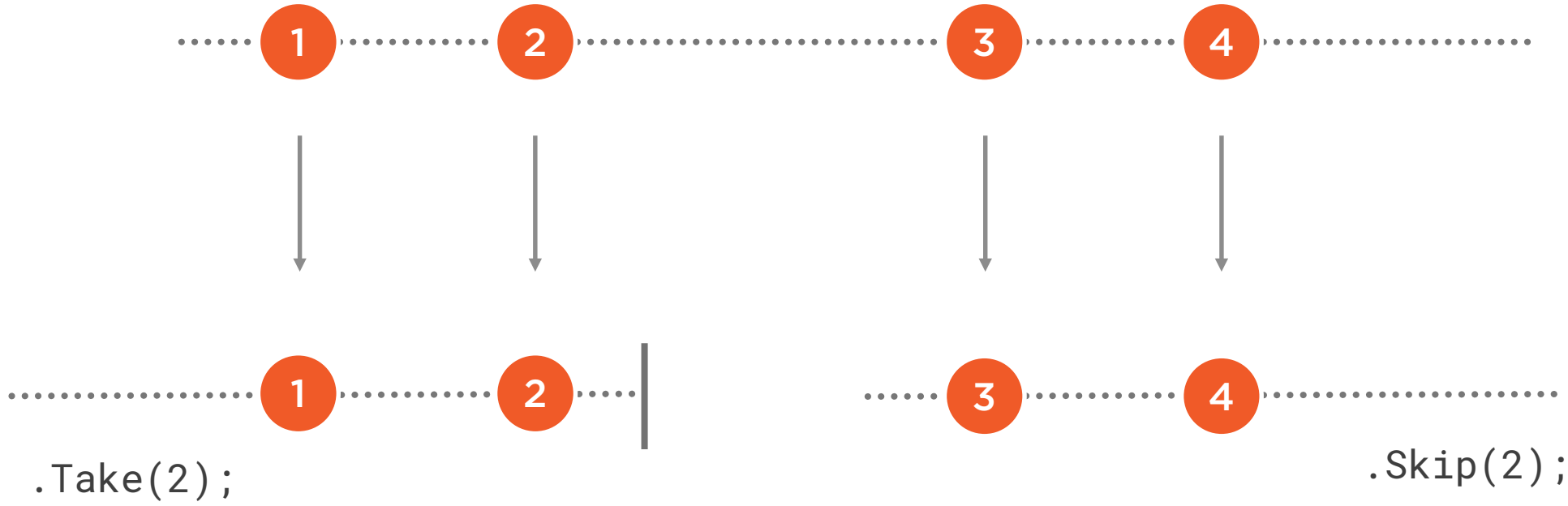
DistinctUntilChanged



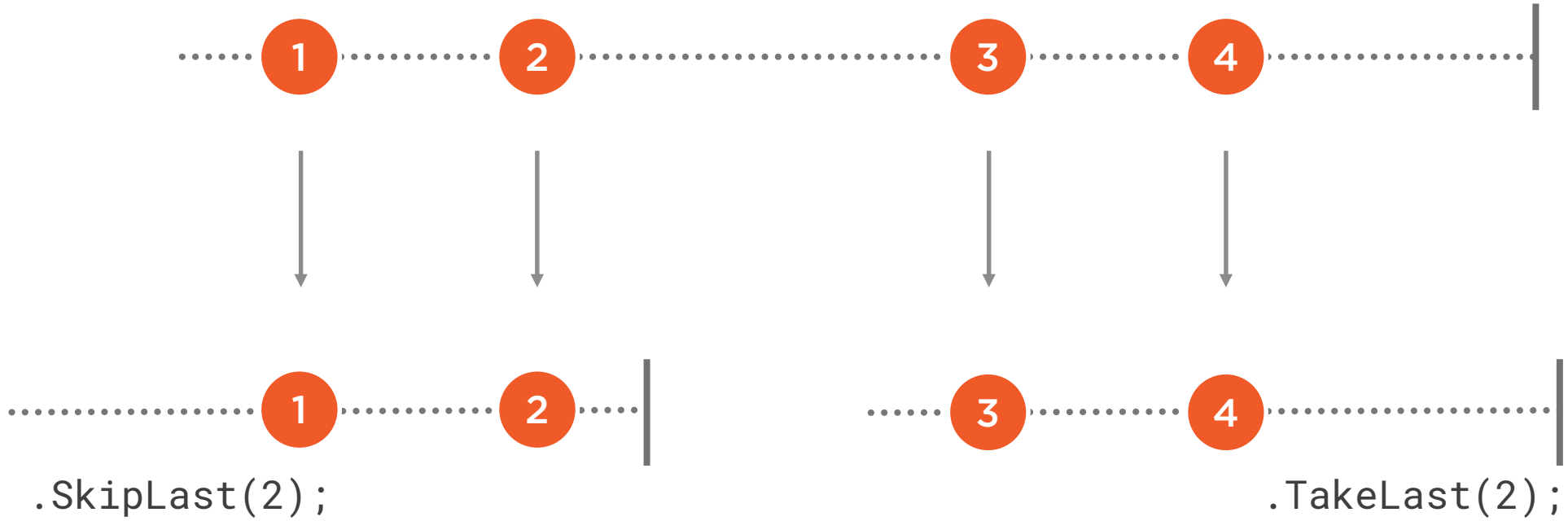

```
.DistinctUntilChanged();
```

Filter Head or Tail Elements

Skip / Take

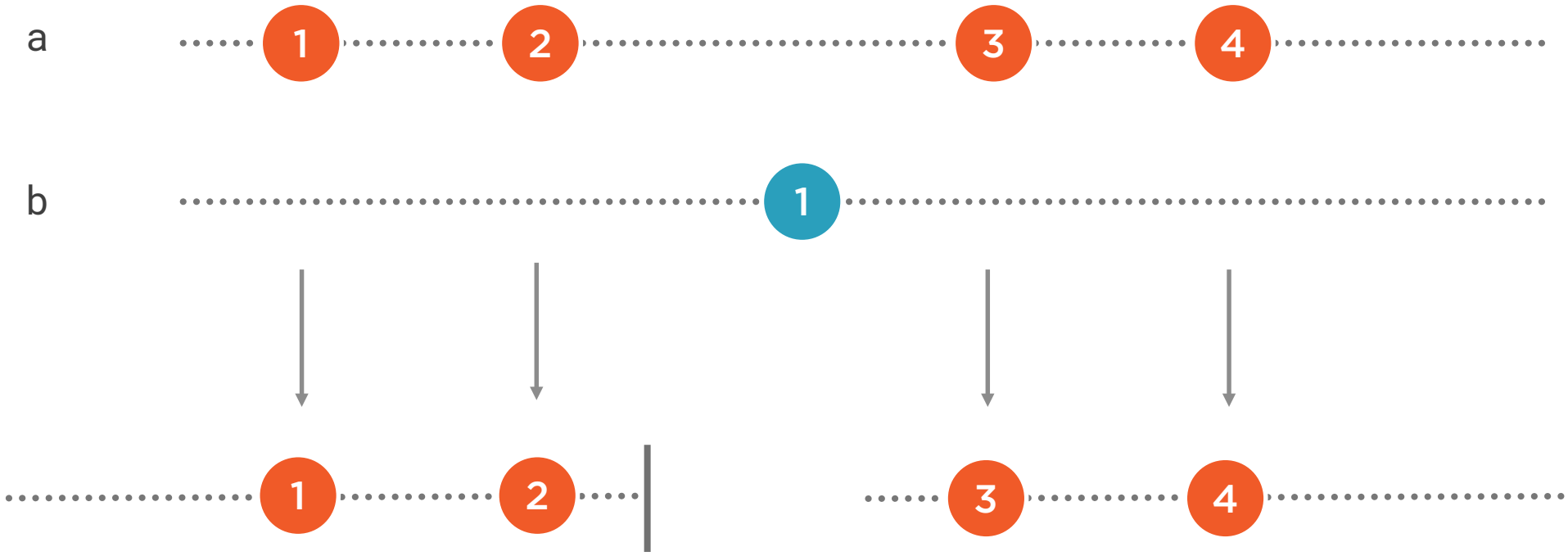


SkipLast / TakeLast



Sequence “Valves”

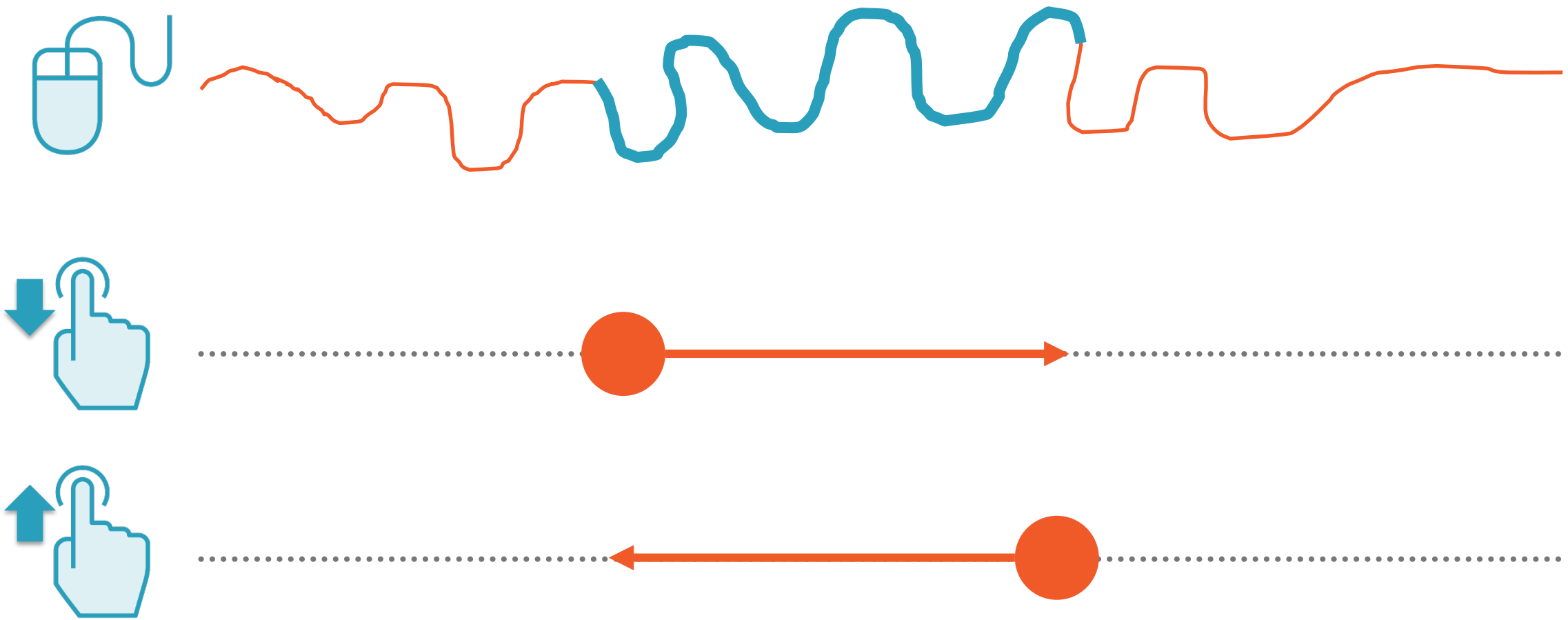
SkipUntil / TakeUntil




```
a.TakeUntil(b);
```

```
a.SkipUntil(b);
```

Converting Mouse Moves and Clicks into Drags



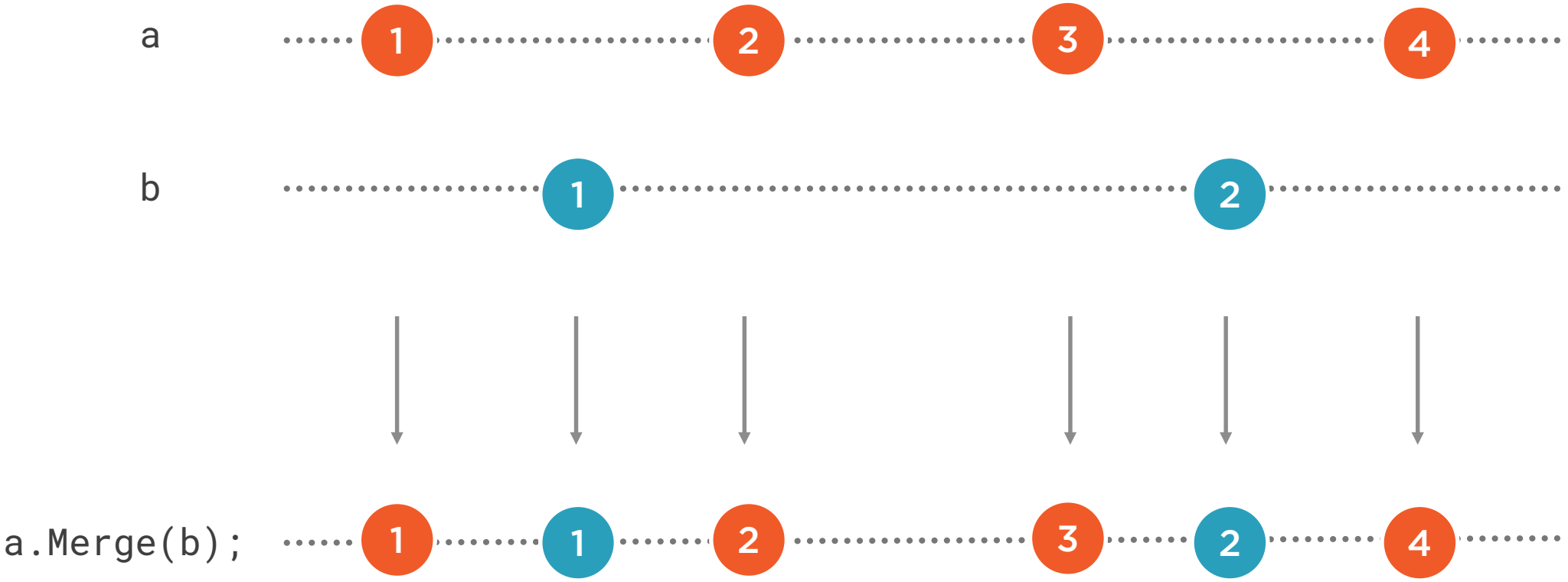
```
// mouseMoves (OnMouseMove)
// mouseUps   (OnMouseUp)
// mouseDowns (OnMouseDown)


var mouseDrags = mouseMoves

       .SkipUntil(mouseDowns)

       .TakeUntil(mouseUps);
```

- Pipe the mouse moves only
- When the mouse is down
- Until the mouse is up again

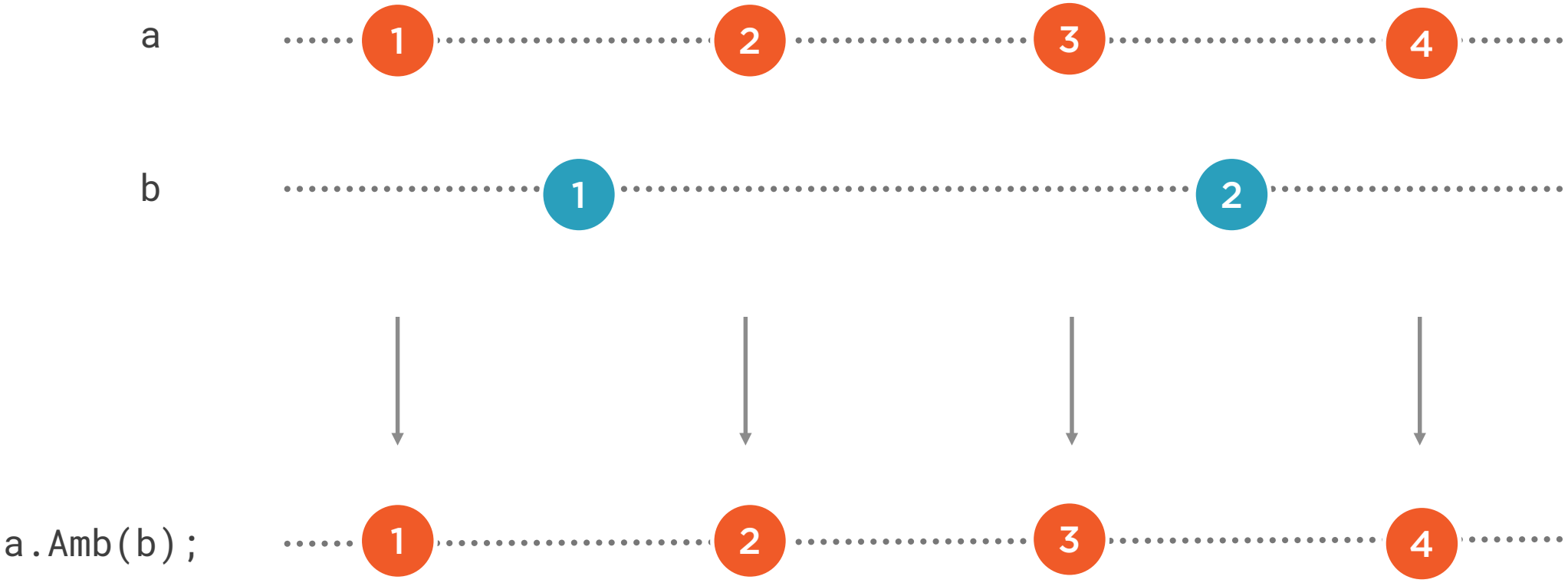
Combining Sequences

Combining Sequences of Same Type

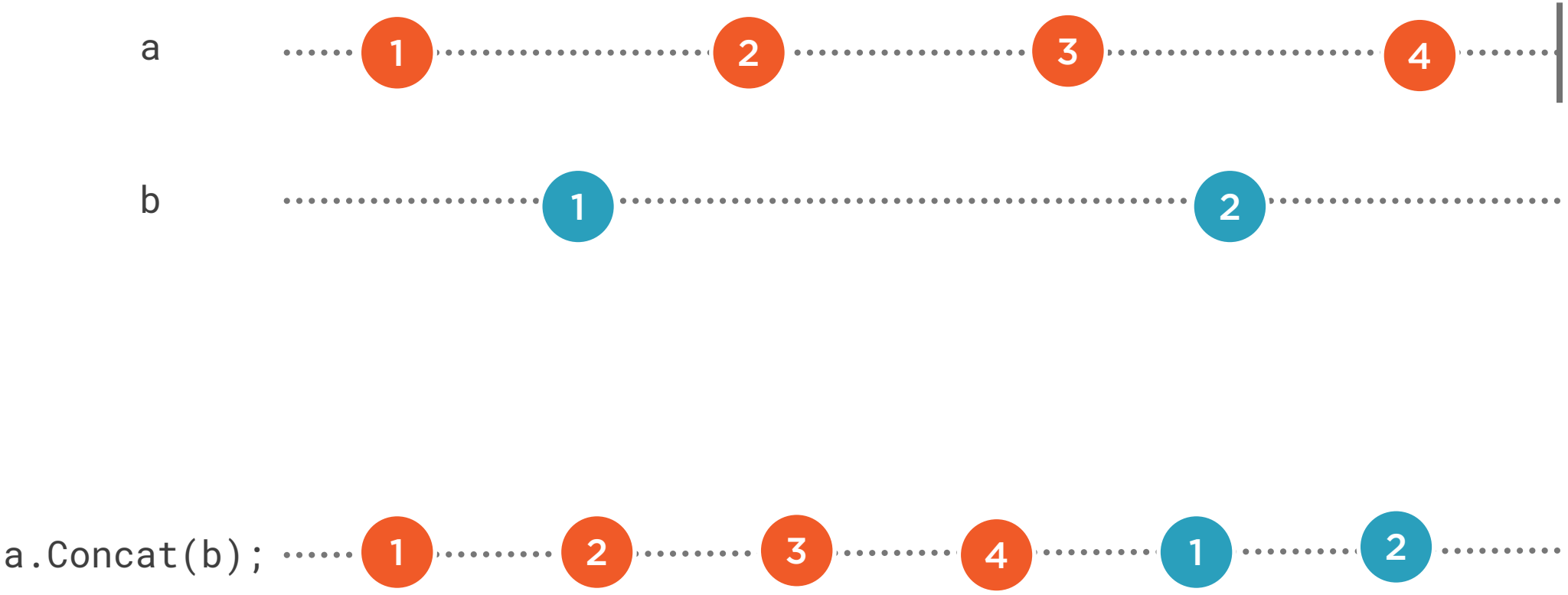
Merge



Amb

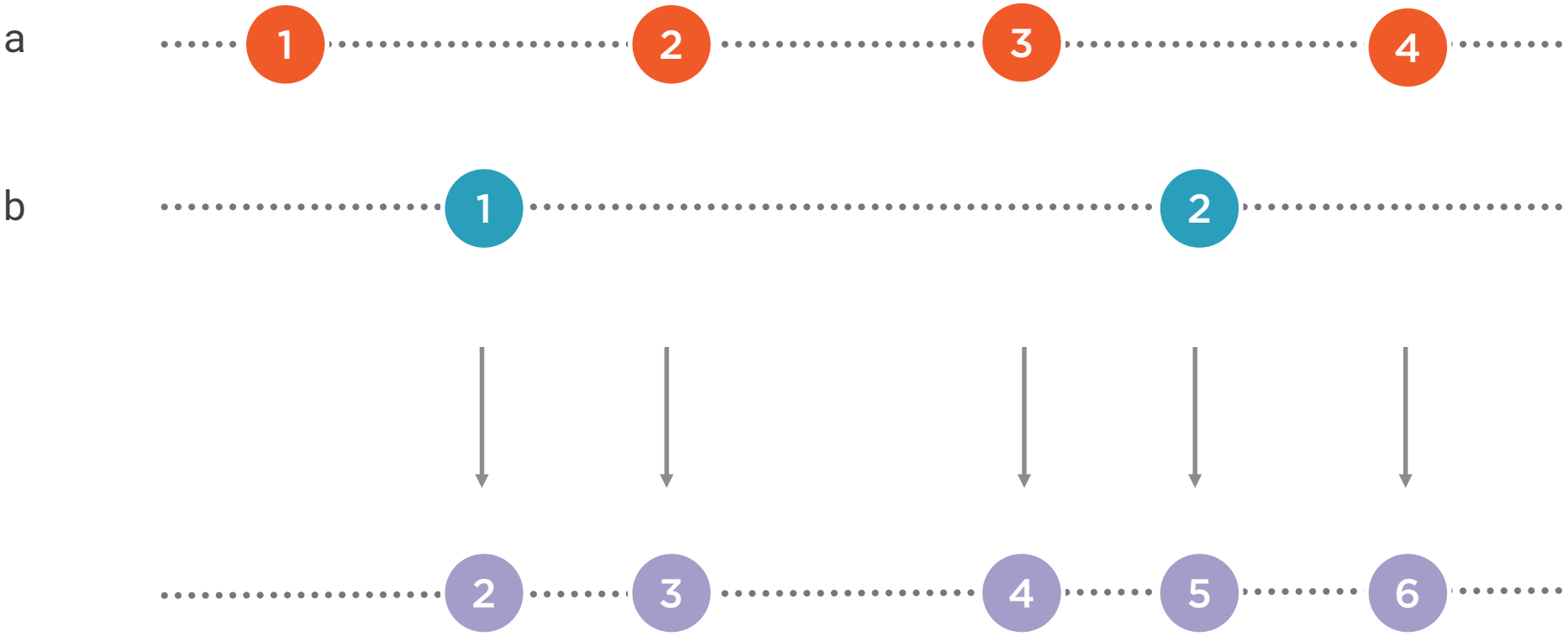


Concat



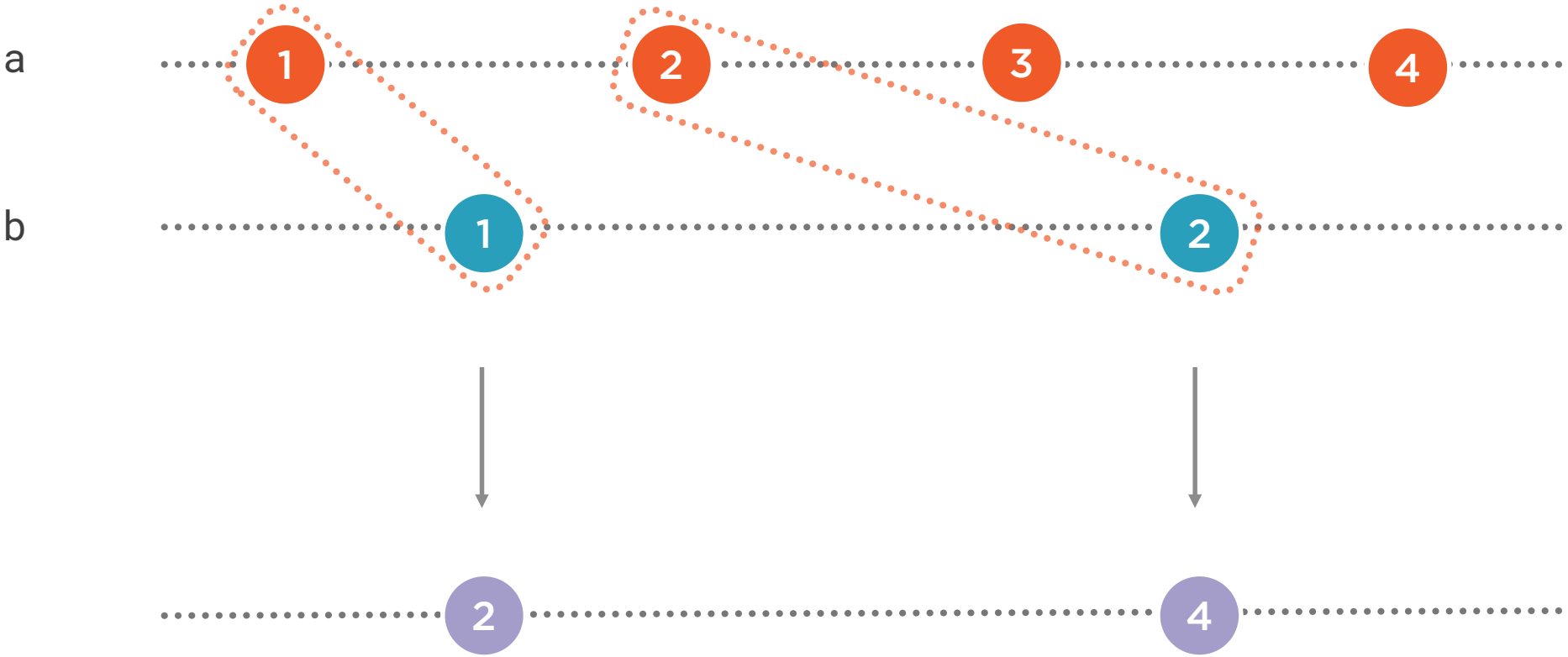
Pairing Sequences

CombineLatest




```
a.CombineLatest(b, (x, y) => x + y );
```

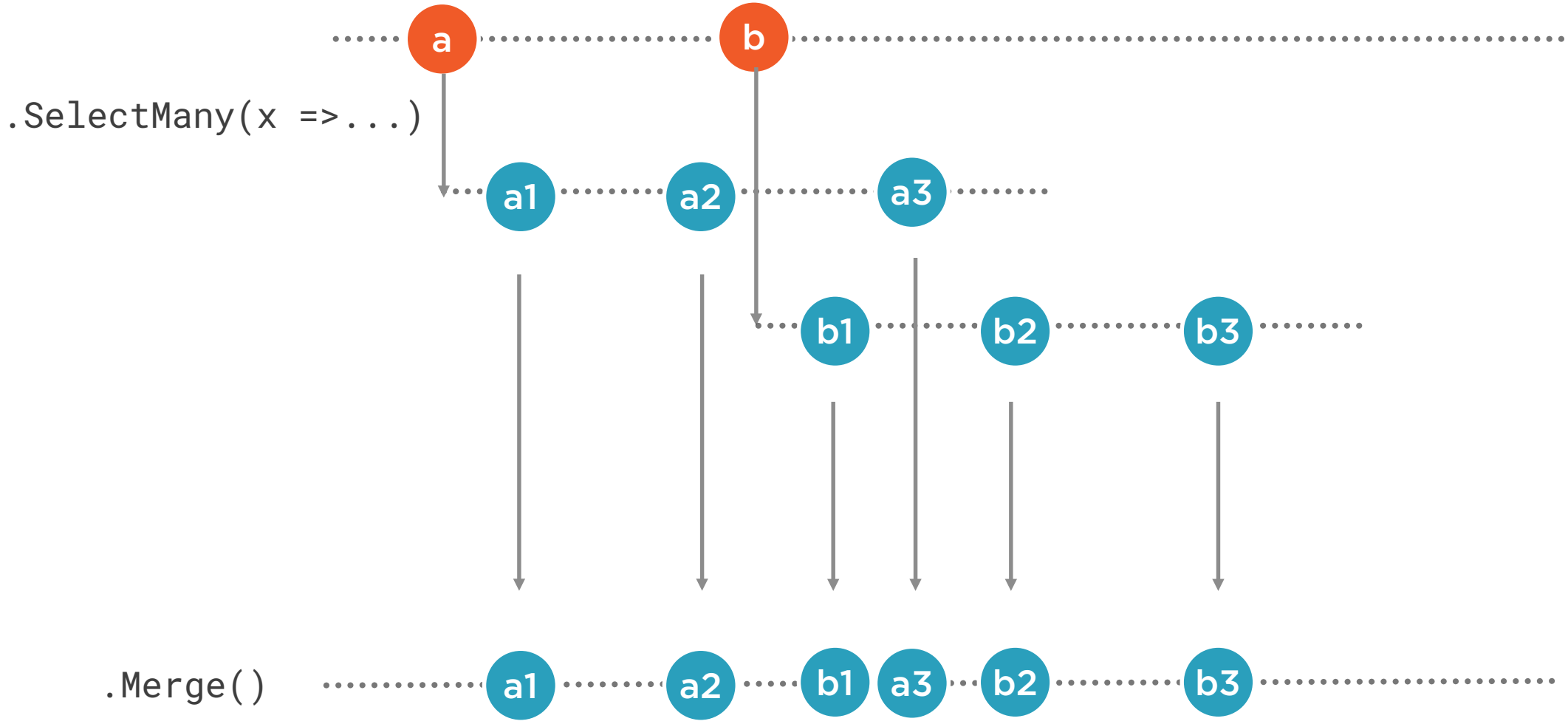
Zip



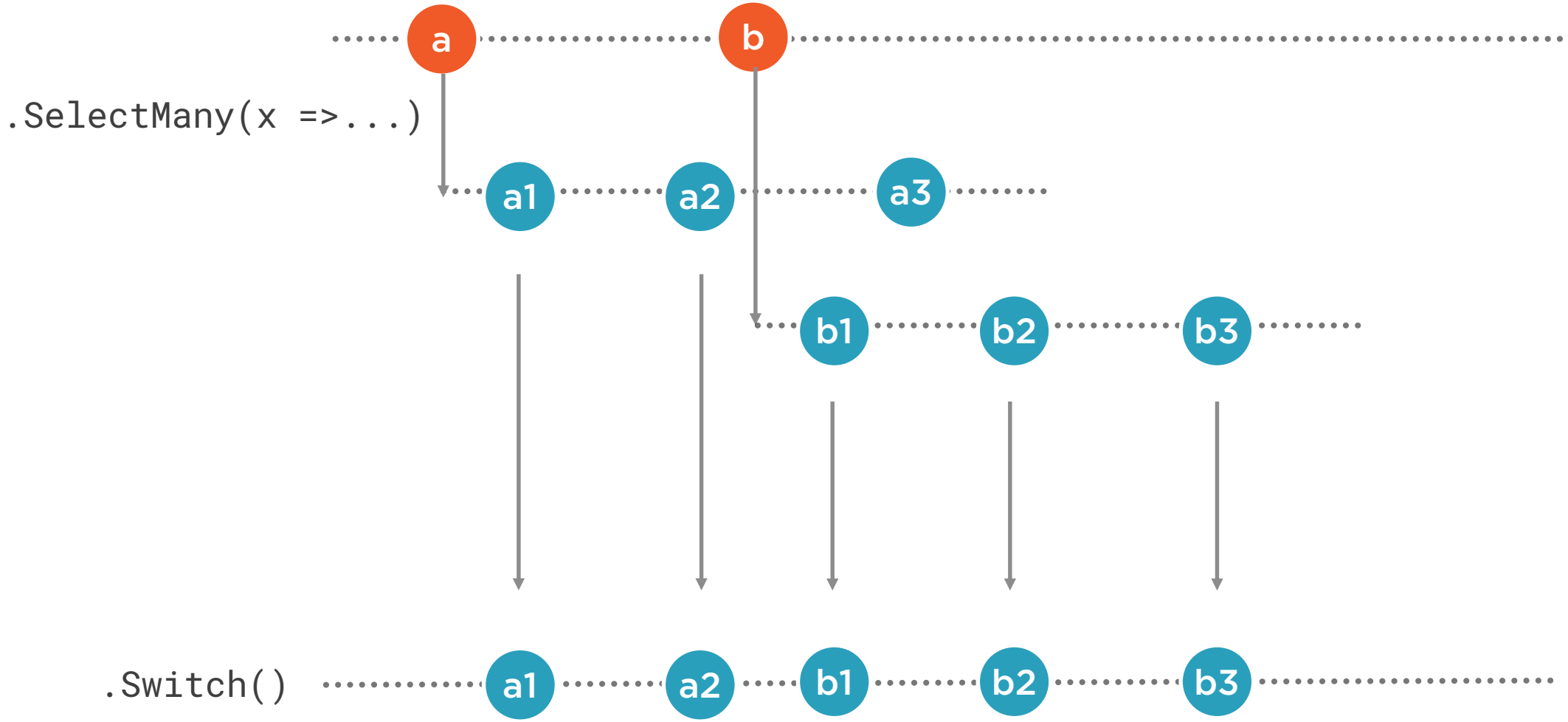

```
a.Zip(b, (x, y) => x + y );
```

Taming Sequences of Sequences

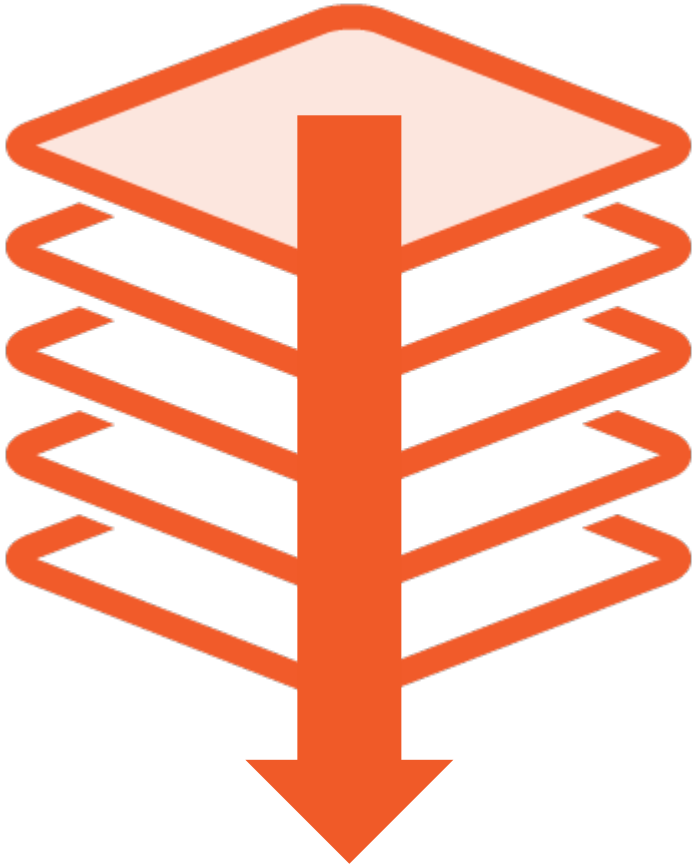
Merging Sequences of Sequences



Switch



Aggregating Operators



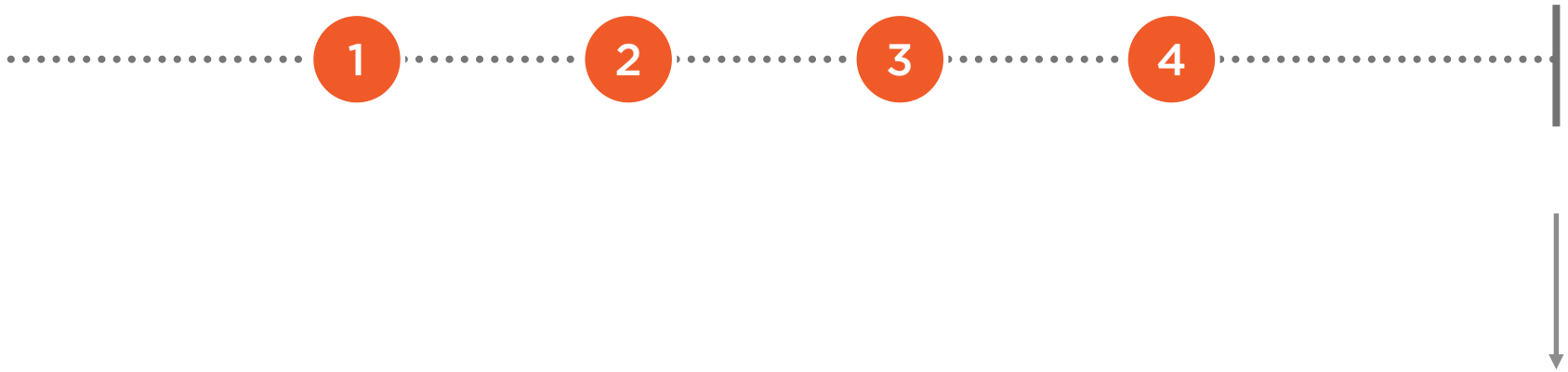
Aggregation is a way to consolidate a sequence into a single result

- Reduces a sequence into a scalar

Some aggregates can be blocked until the sequence completes

- However, the most of them are made async

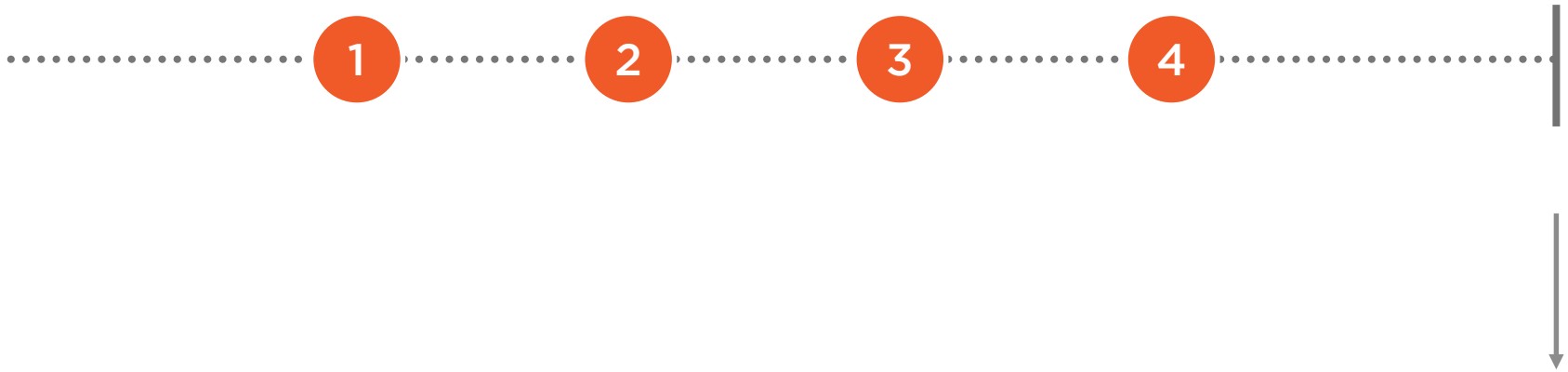
Count / Sum



`.Count()` **4**

`.Sum()` **10**

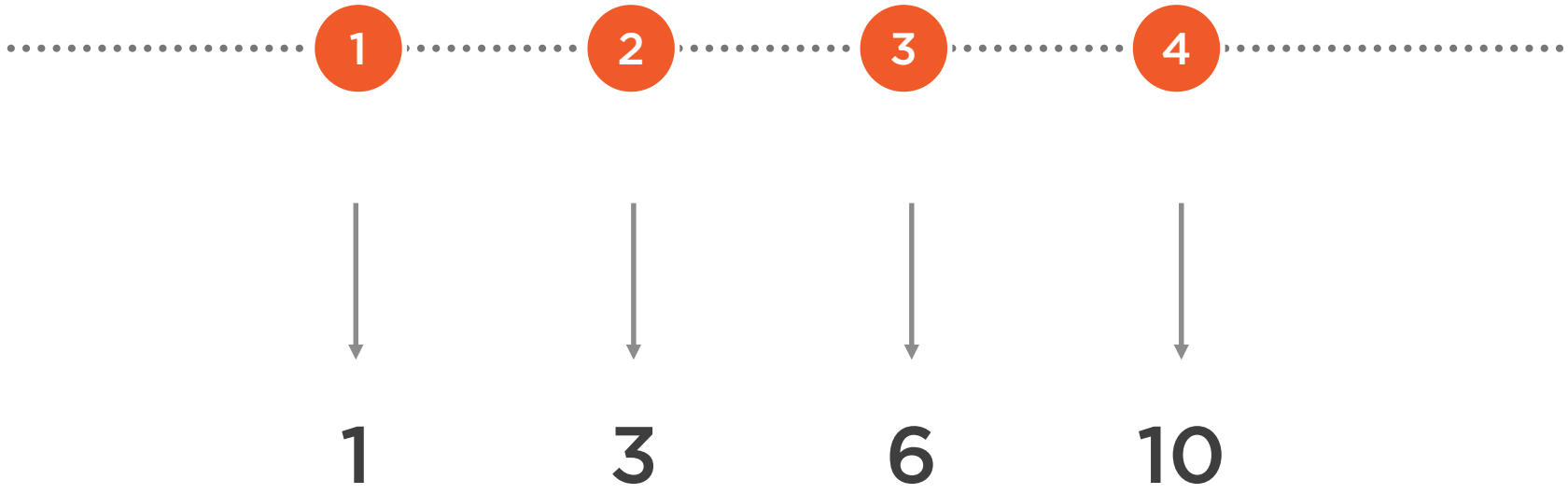
Aggregate




```
.Aggregate((acc, currValue) => acc + currValue)
```

10

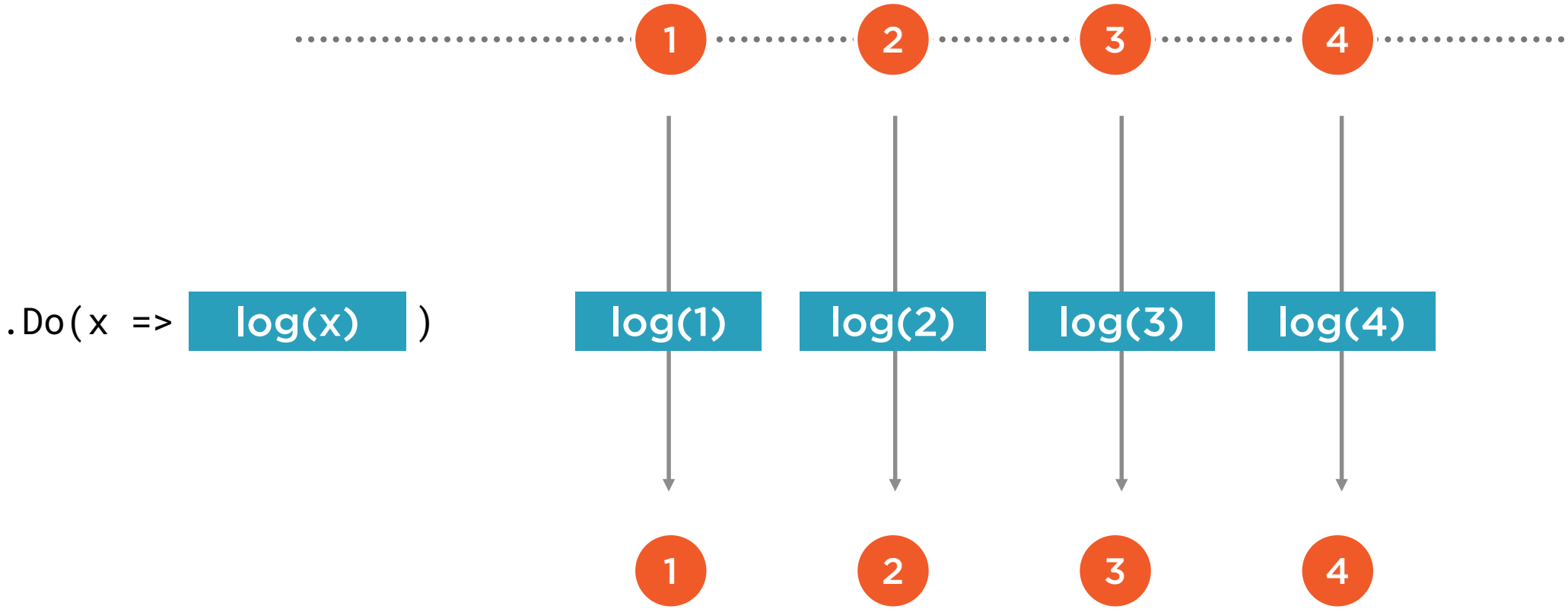
Scan



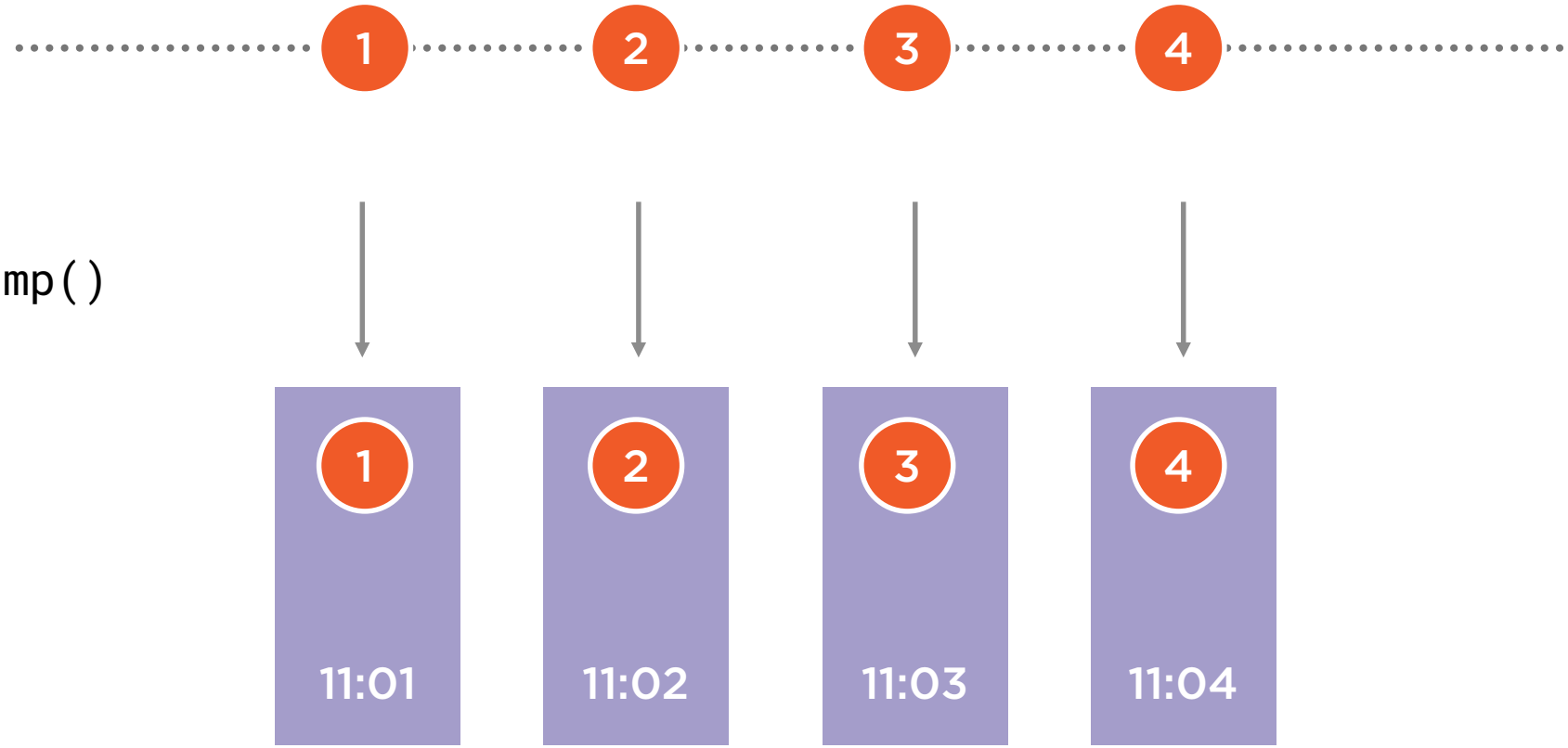

```
.Scan(0,(acc, currValue) => acc + currValue)
```

Utility Operators

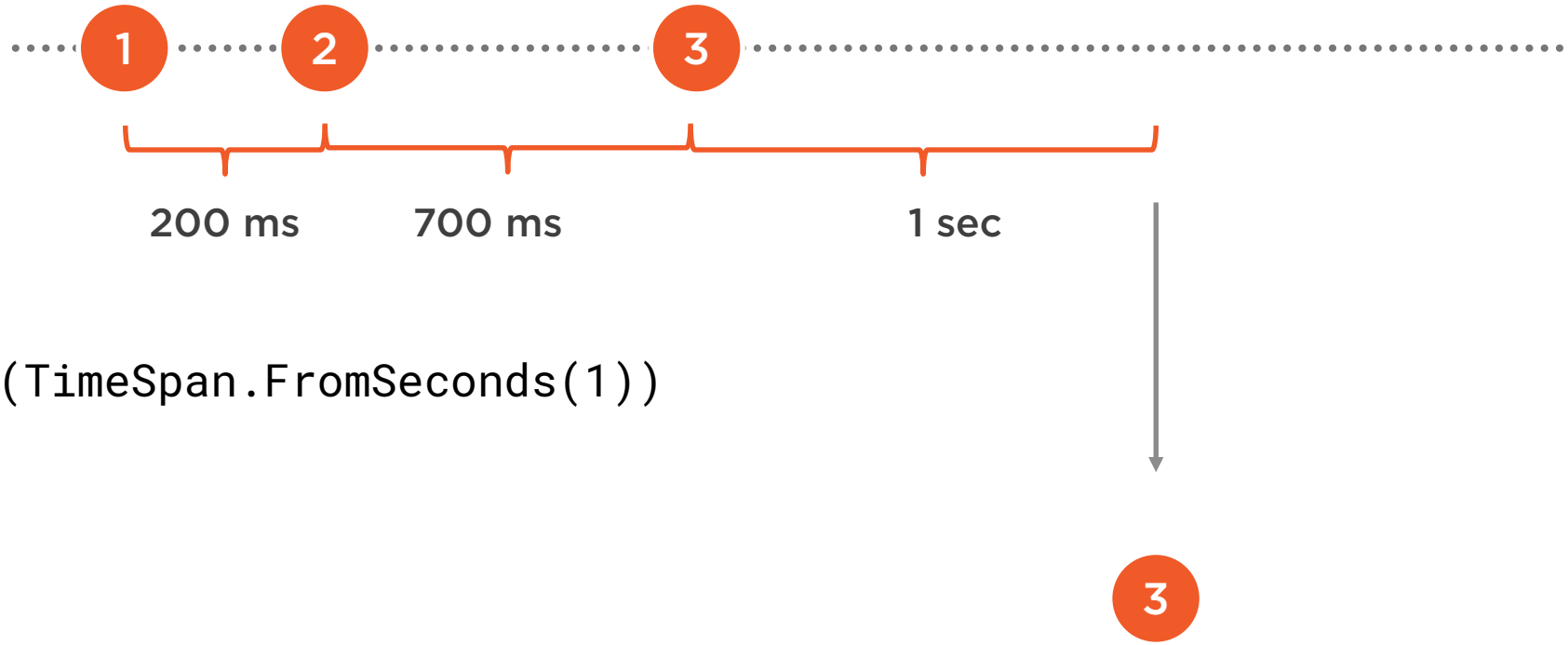
Do



Timestamp

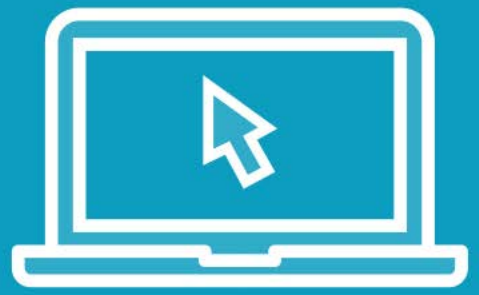


Throttle




```
.Throttle(TimeSpan.FromSeconds(1))
```

Demo



Enhancing “Reactive Tickets”

- Adding search filtering with Rx

Summary

LINQ operators are the “magic” of Rx

- Filtering
- Aggregating
- Combining
- Utility

It takes some practice to get them right