ManfredSteyer

Reactive Extensions for JavaScript

Rainer Hahnekamp

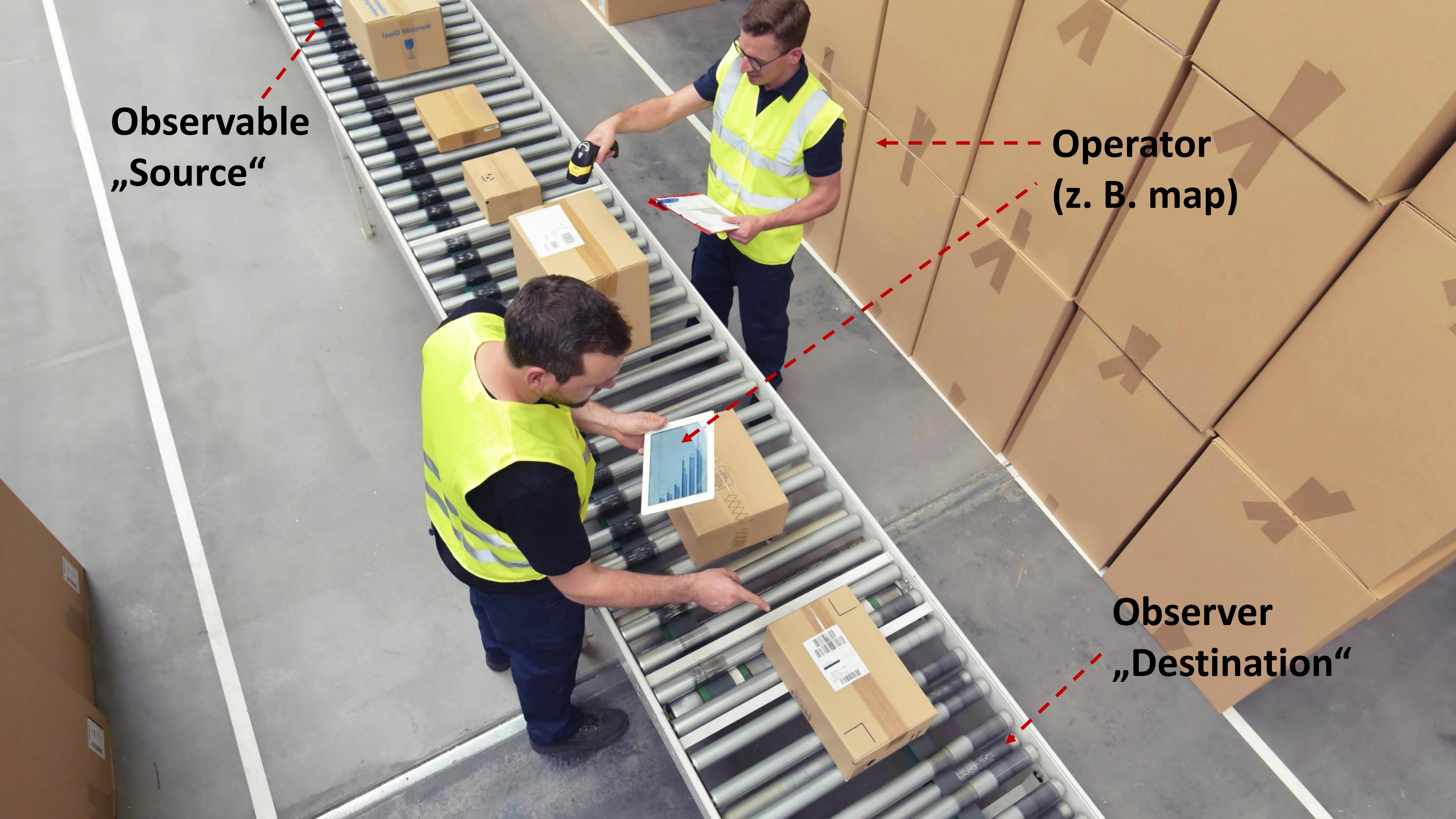
SOFTWAREarchitekt.at

Contents

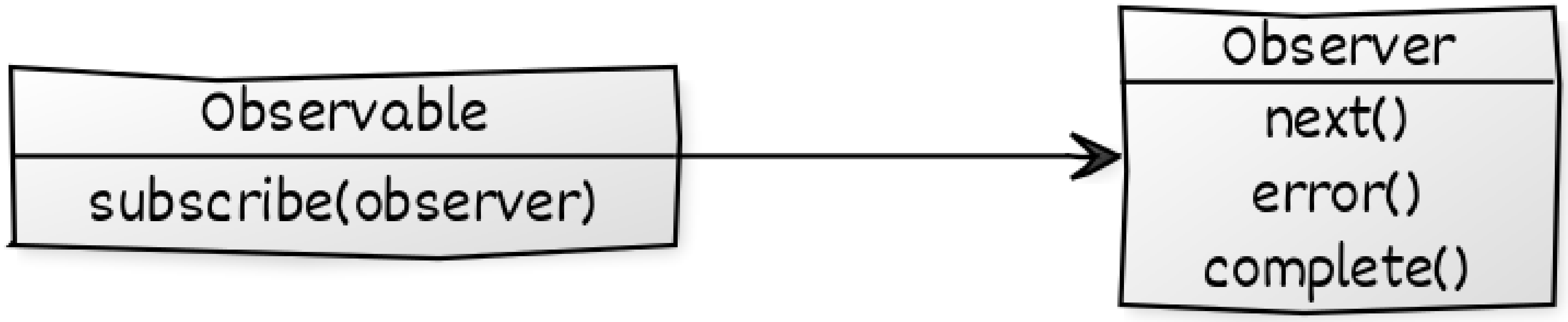
Overview

What are observables?

- Represents (asynchronous) data that is published over time


Observable
„Source“
Operator
(z. B. map)
Observer
„Destination“

Observable und Observer



Observer

```
myObservable.subscribe(
    (result) => { … }
);
```

Observer

Observer

```
myObservable.subscribe({
    next: (result) => { … },
    error: (error) => { … },
    complete: () => { … }
});
```

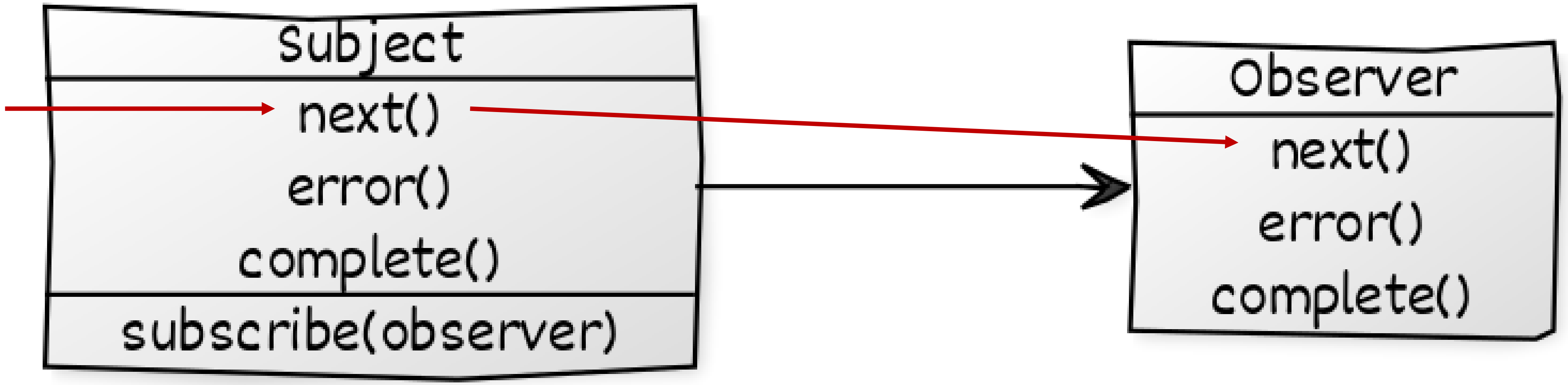
Observer

Example with Pipeable Operators

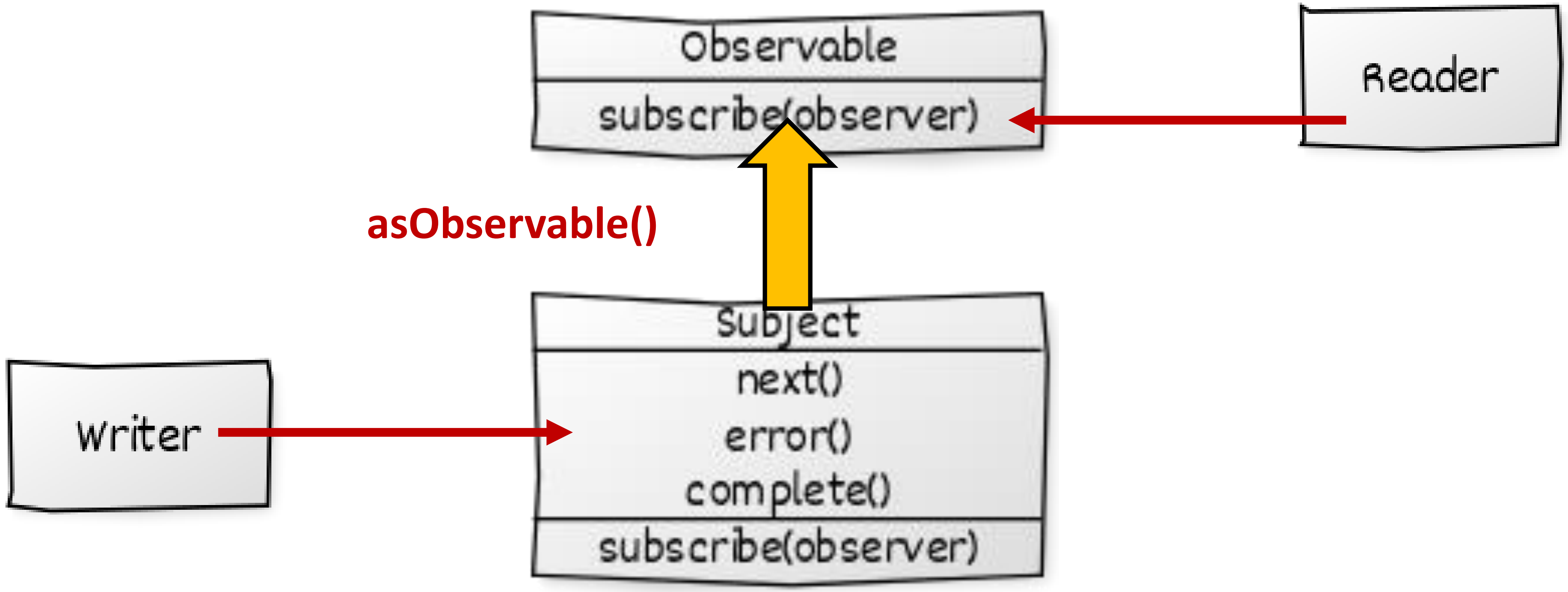
```
import { map } from 'rxjs/operators';

this
    .http
    .get("http://www.angular.at/api/...")
    .pipe(map(flightDateStr => new Date(flightDateStr)))
    .subscribe({
        next: (bookings) => { … },
        error: (err) => { console.error(err); }
    });
```

Subjects: Special Observables



Convert Subject into Observable



asObservable

```
private subject = new Subject<Flight>();
readonly observable = subject.asObservable();

[…]
this.observable.subscribe(…)

[…]
this.subject.next(…)
```

Why Observables?

Asynchronous operations

Interactive (reactive) behavior

Creating Observables

Creating an Observable

```
let observable = new Observable((observer) => {

    observer.next(4711);
    observer.next(815);

    // observer.error("err!");

    observer.complete();

    return () => { console.debug('Bye bye'); };
});
```

Sync & Async, Event-driven

```
let subscription = observable.subscribe(observer);
```

```
subscription.unsubscribe();
```

Creation Operators (Factories)

[https://www.learnrxjs.io]

fromEvent

of

throwError

interval

timer

Cold vs. Hot Observables

Cold vs. Hot Observables

Cold

- Point to point
- Lazy: Only starts with subscription

Default

Hot

- Multicast
- Eager: Sender starts without subscriptions

Create Hot Observable

```
let o = this.find(from, to).pipe(share());

o.subscribe(…);

o.subscribe(…);
```

Sender starts with first subscription

Sender stops after all receiver have been unsubscribed

Create Hot Observable

```
let o = this.find(from, to)
            .pipe(shareReplay({ bufferSize:1, refCount: true }));

o.subscribe(…);

[…]

o.subscribe(…);
```

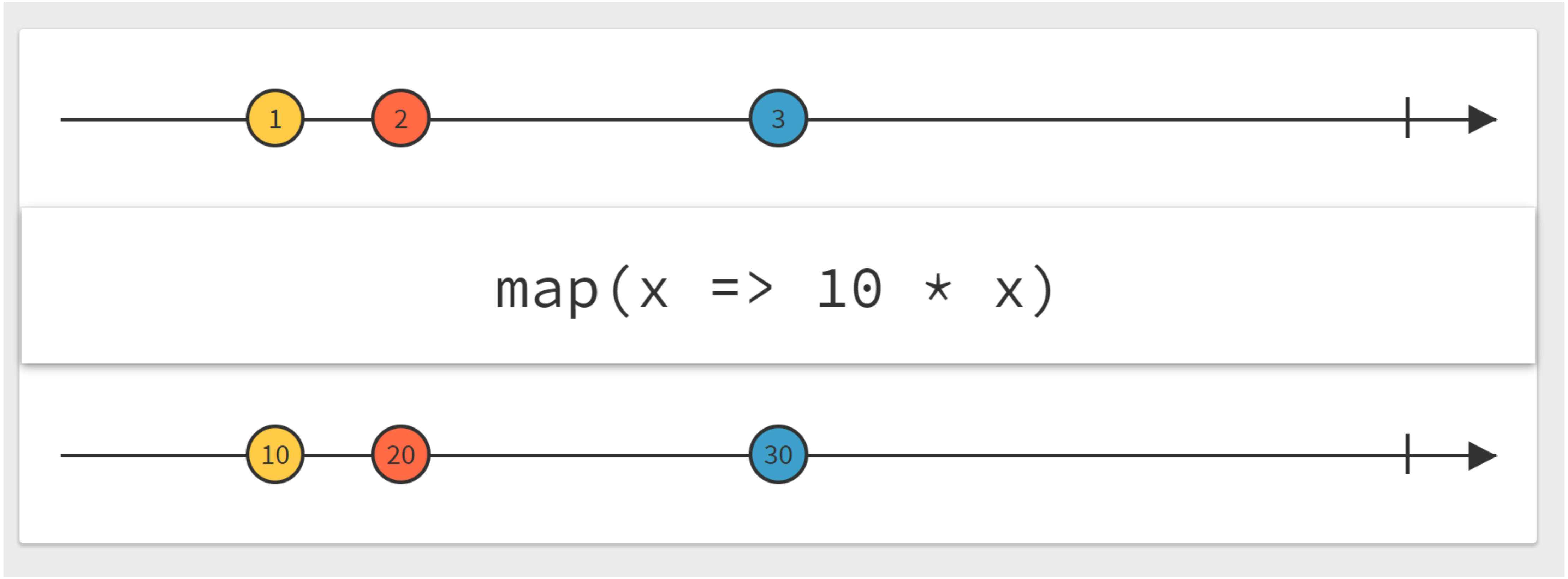
DEMO

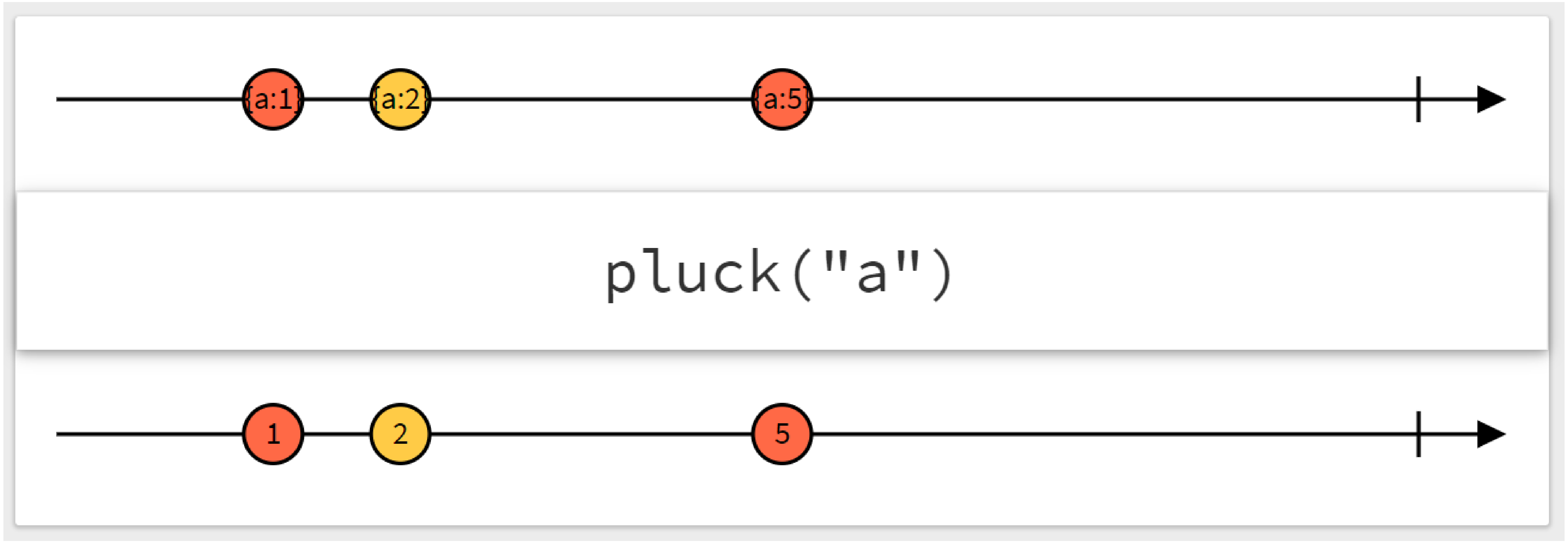
Operators

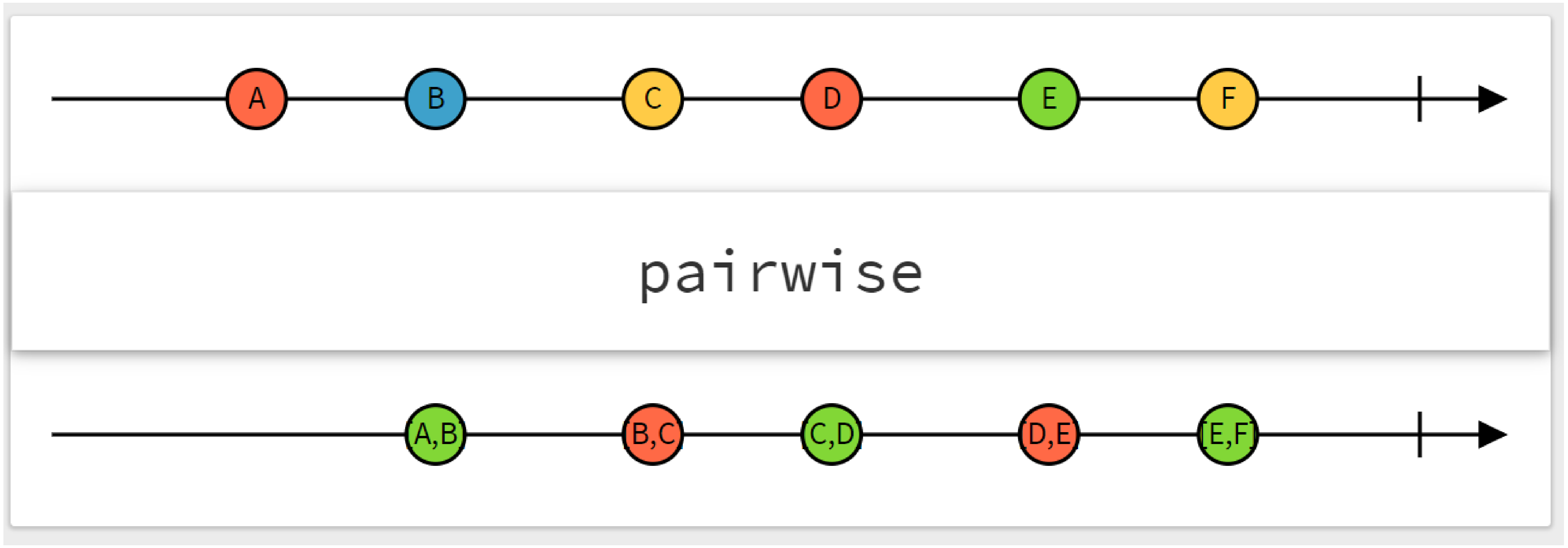
Transformation Operators

Operators

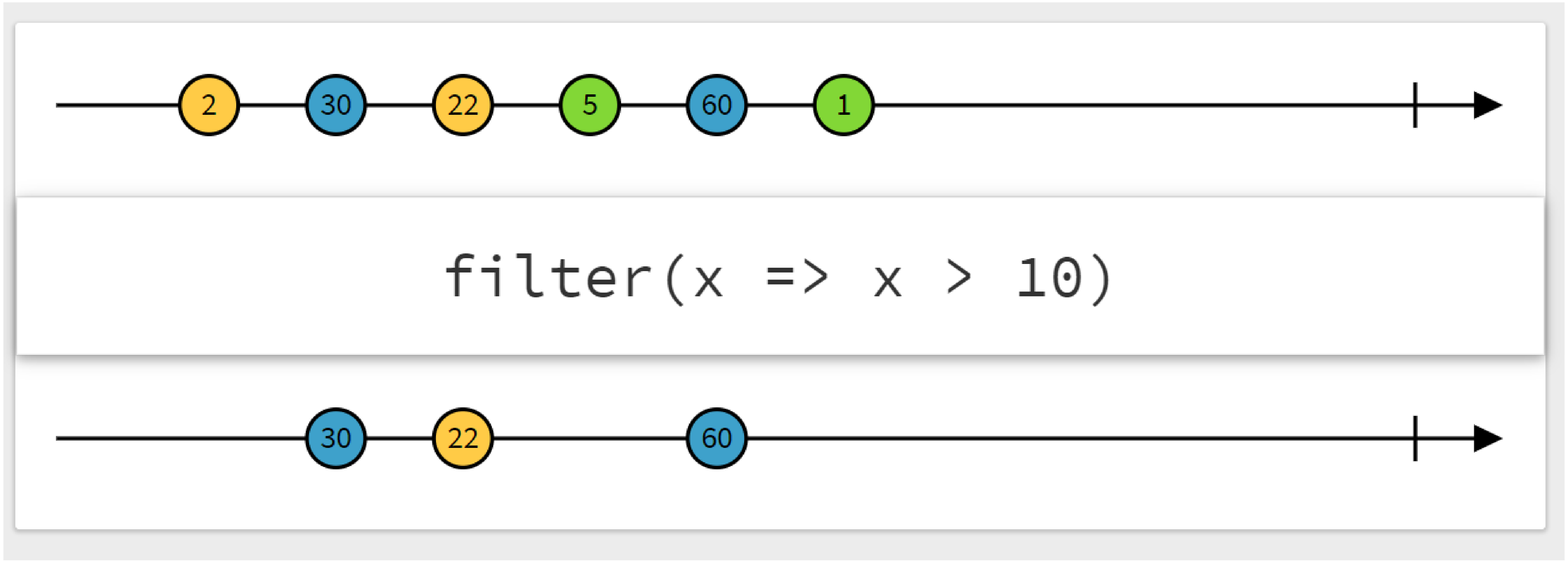
[http://rxmarbles.com/#map]

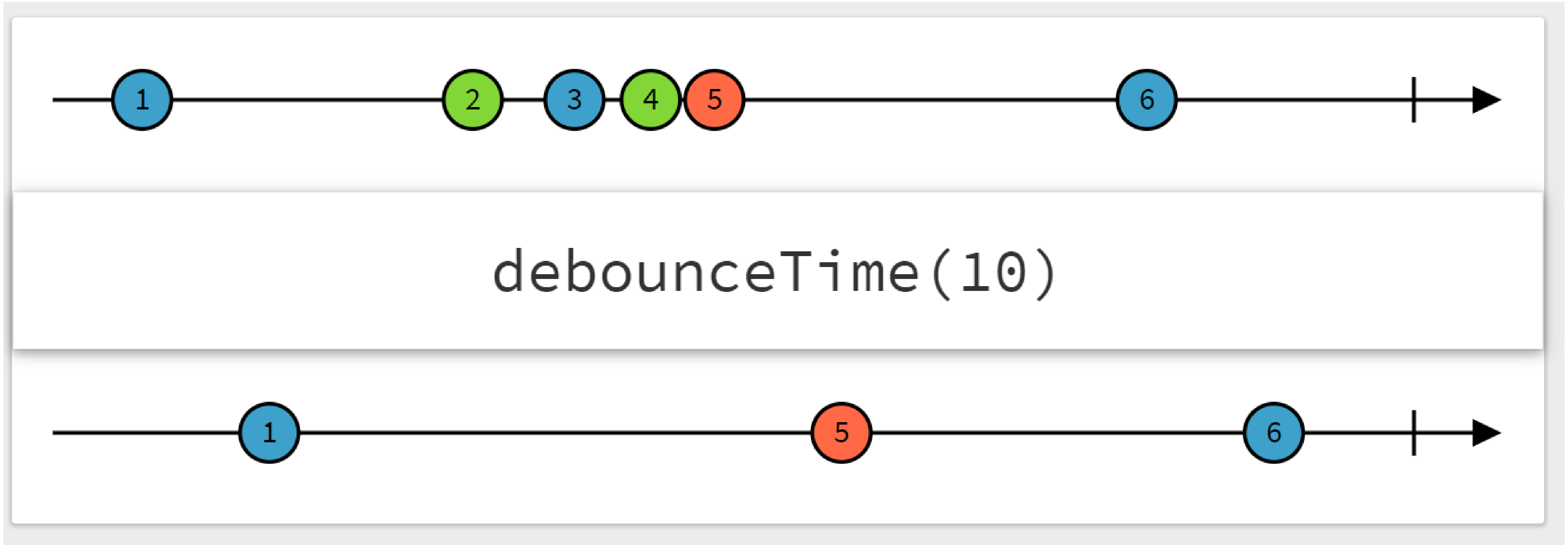


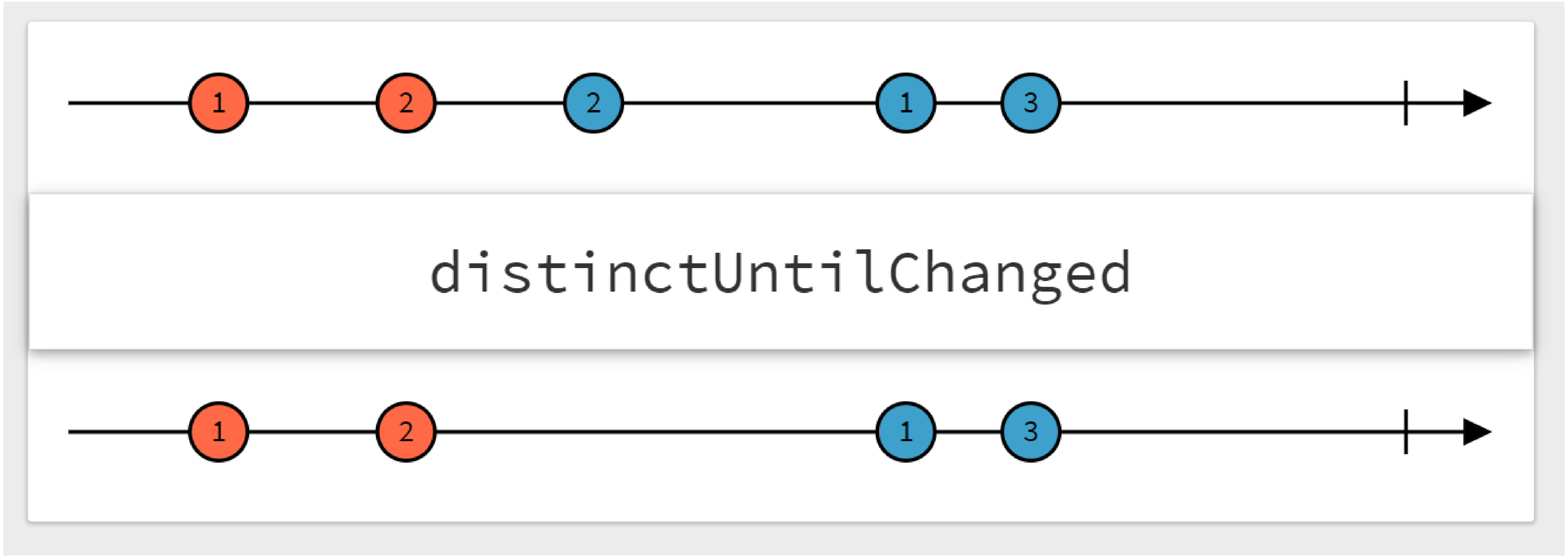

{a:1}
{a:2}
{a:5}
pluck("a")
1
2
5


A
B
C
D
E
F
pairwise
[A,B]
[B,C]
[C,D]
[D,E]
[E,F]

Filtering Operators


2
30
22
5
60
1
filter(x => x > 10)
30
22
60

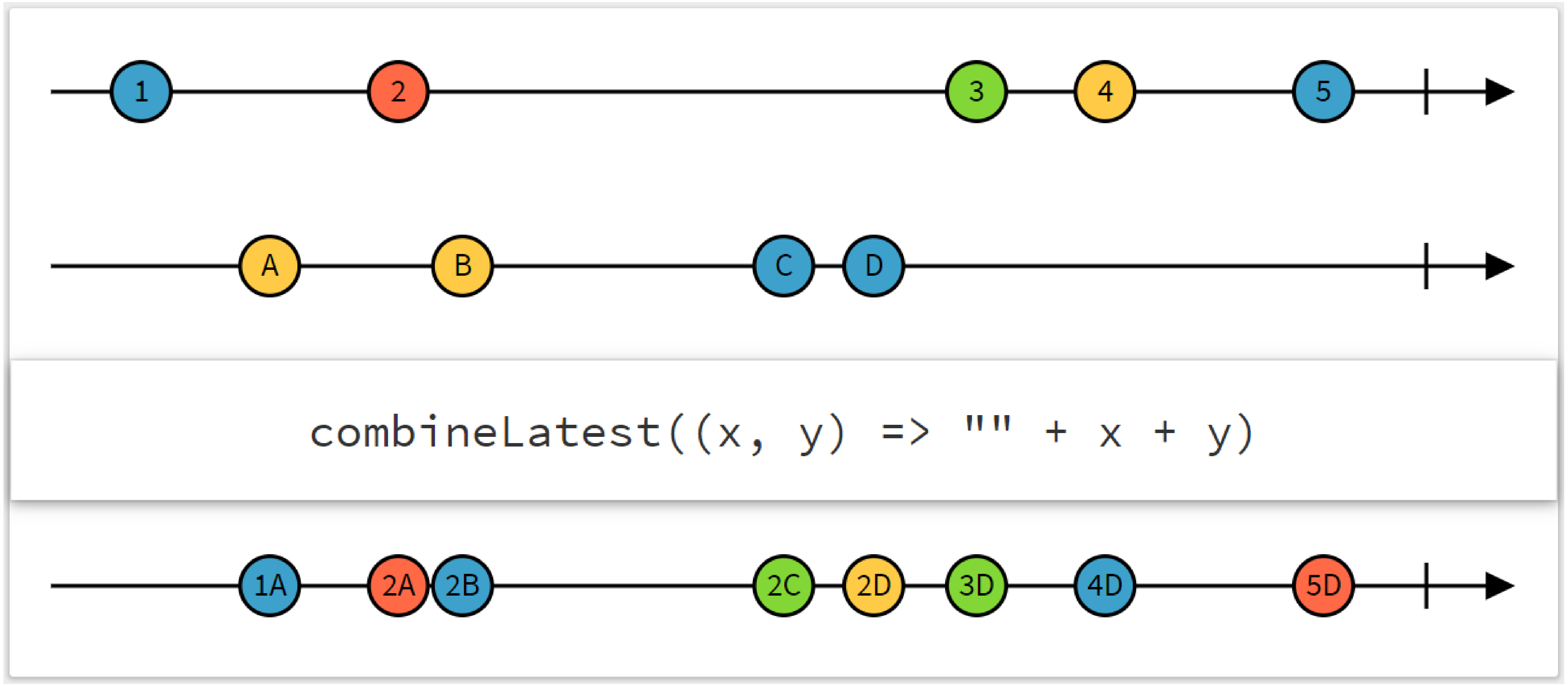

1
2
3
4
5
6
debounceTime(10)
1
5
6

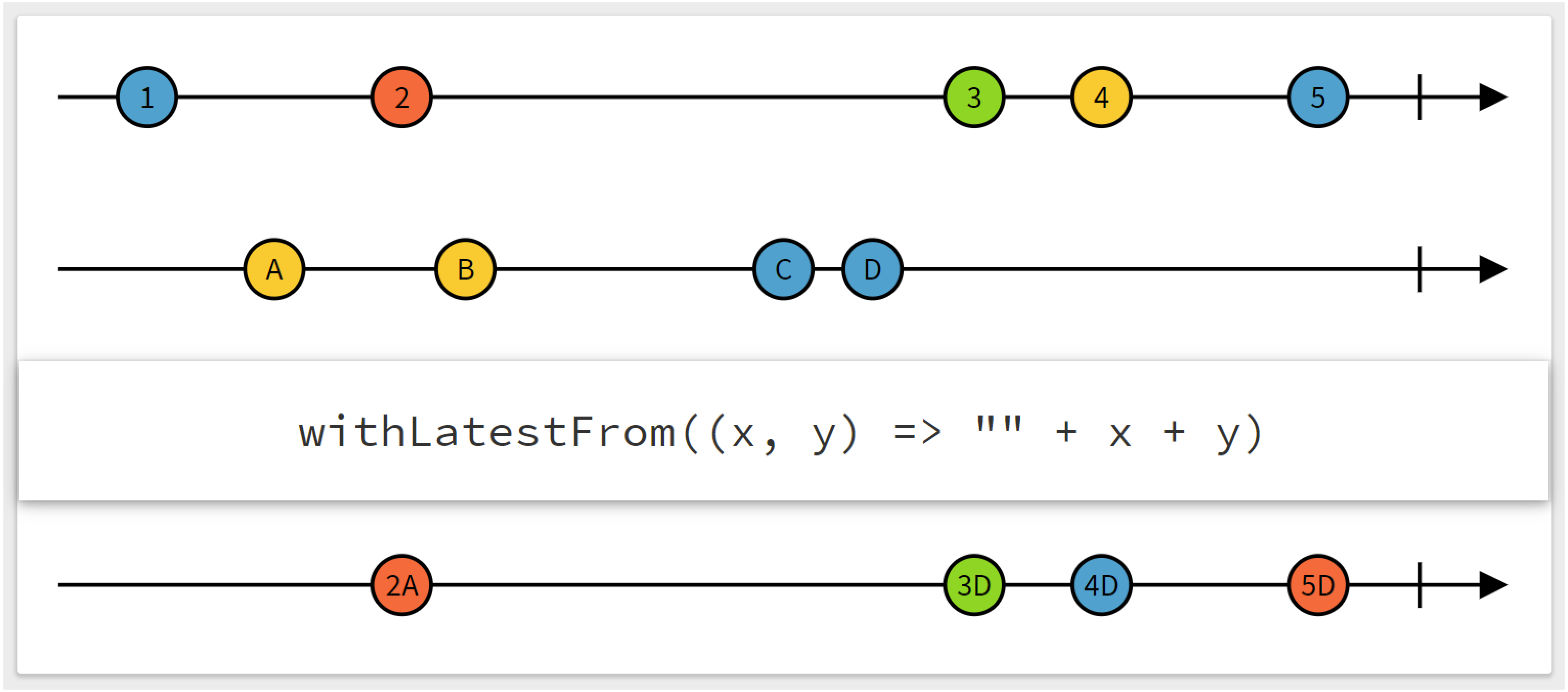

1
2
2
1
3
distinctUntilChanged
1
2
1
3

DEMO: Lookahead

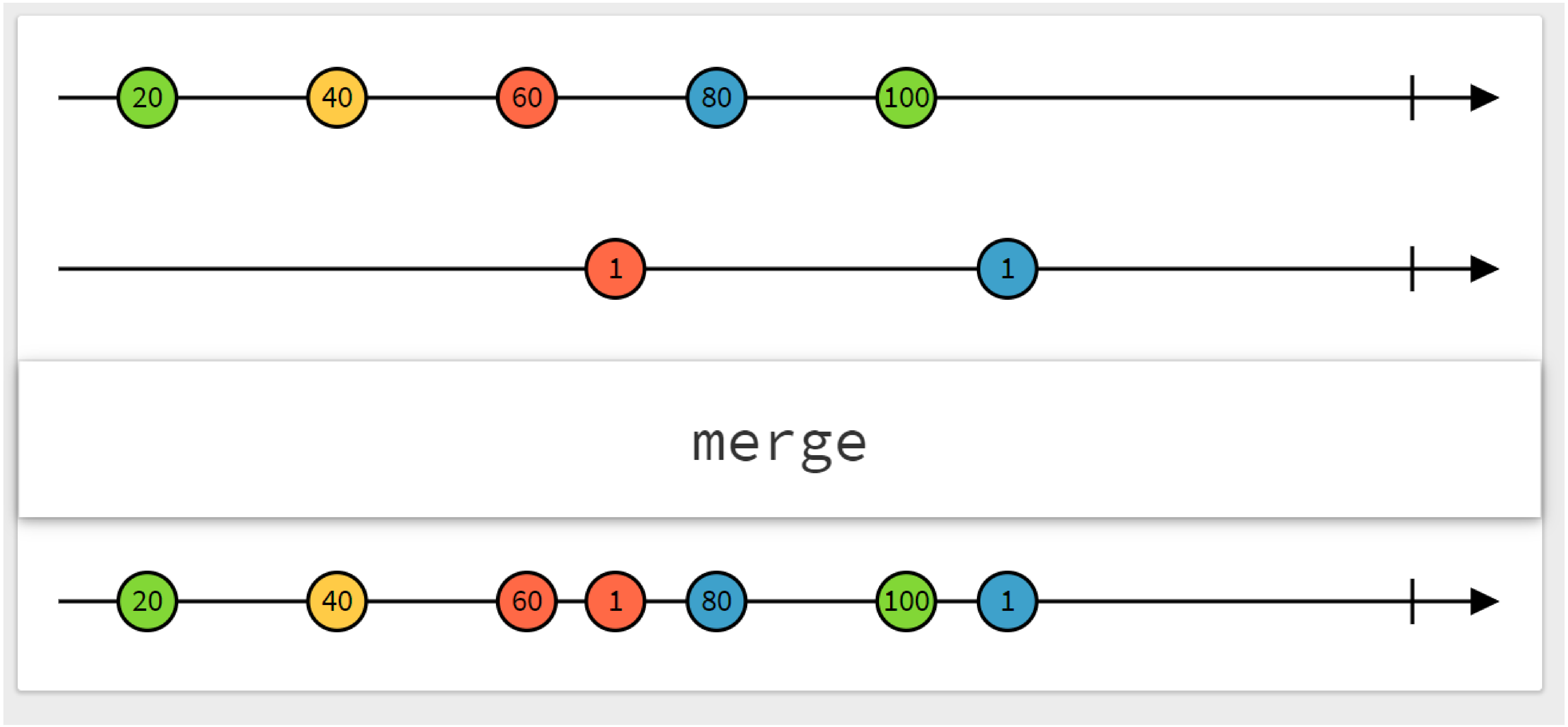
LAB

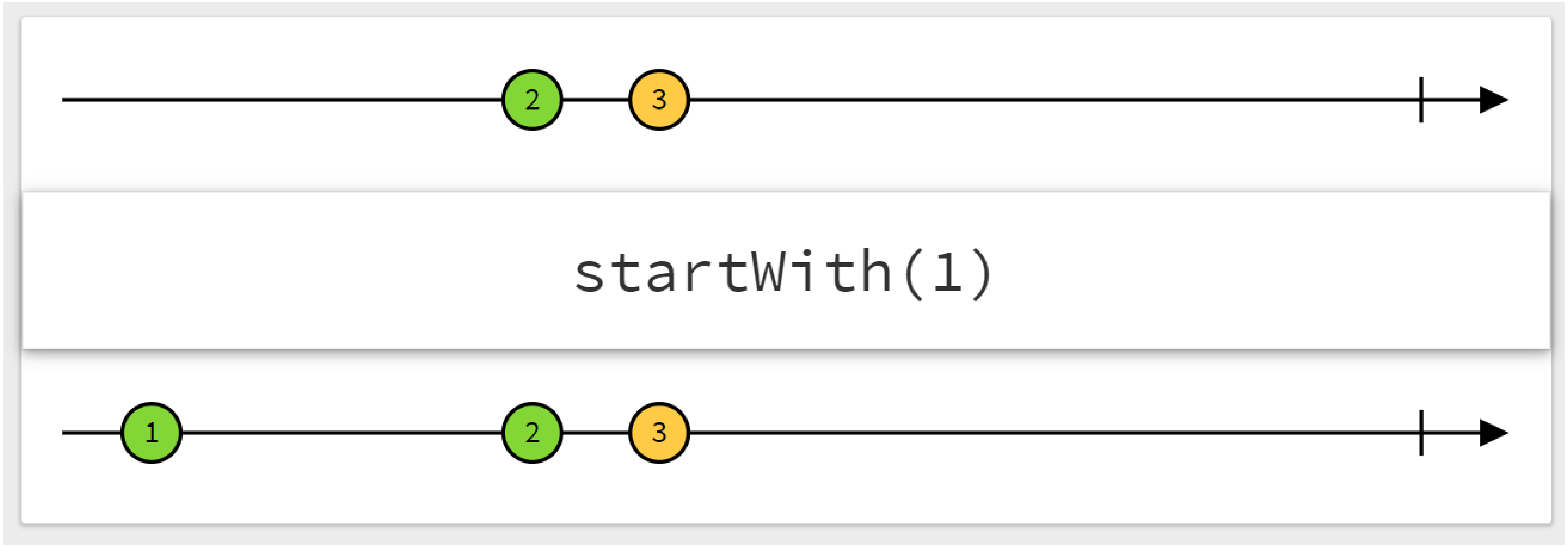
Combination Operators


1
2
3
4
5
A
B
C
D
combineLatest((x, y) => "" + x + y)
1A
2A
2B
2C
2D
3D
4D
5D



```
withLatestFrom((x, y) => "" + x + y)
```


20
40
60
80
100
1
1
merge
20
40
60
1
80
100
1


2
3
startWith(1)
1
2
3

DEMO

Labs

Higher Order Observables

Operators for Higher Order Observables

- switchMap
- mergeMap
- concatMap
- exhaustMap

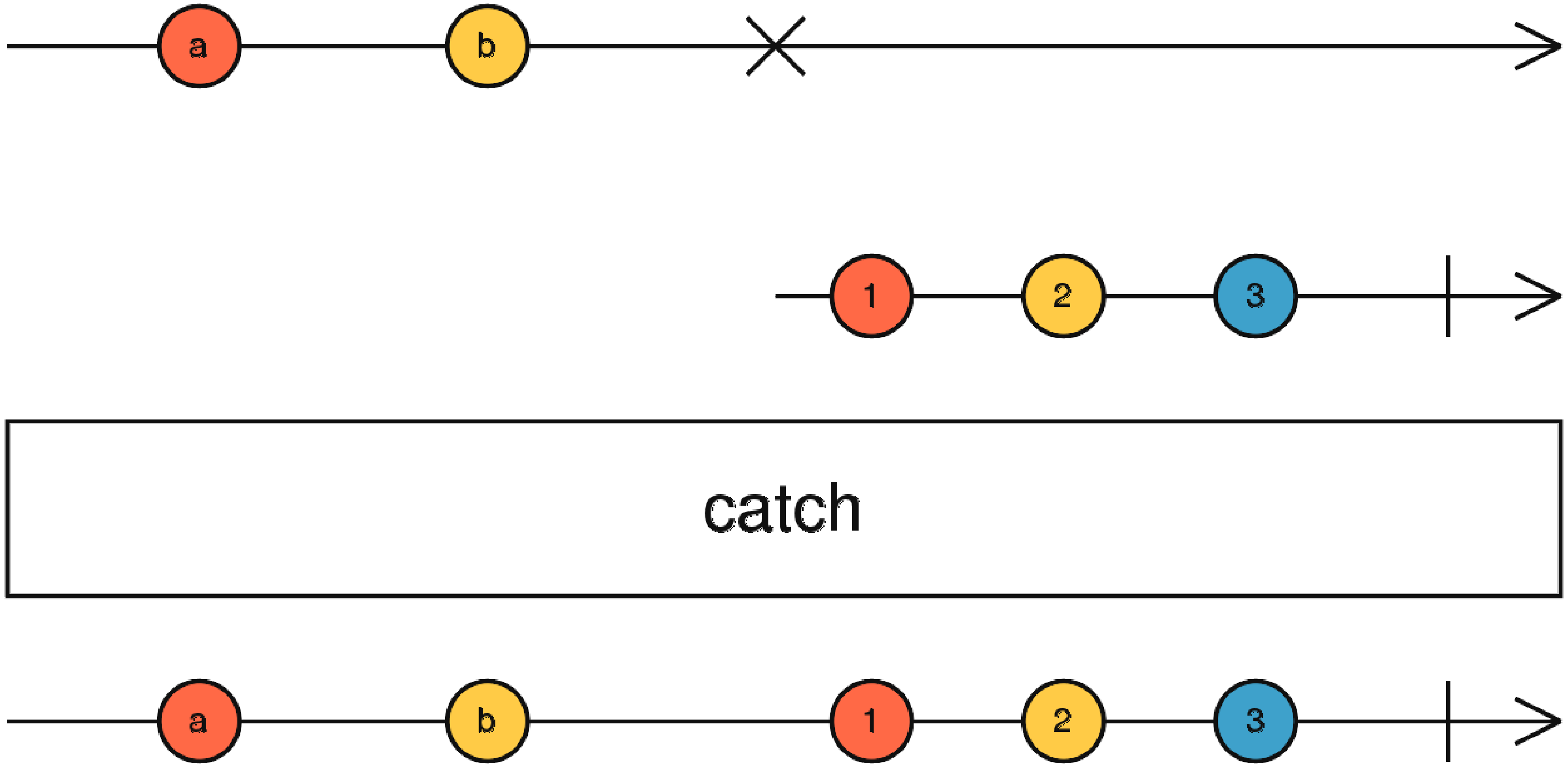
DEMO

Error Handling

Operators for Error Handling

- catchError
- retry
- retryWhen

- throwError


a
b
1
2
3
catch
a
b
1
2
3

DEMO

LAB

Subjects

Subjects

Subject	Hot & distributes data
BehaviorSubject	Saves last value
ReplaySubject	Saves last x values

DEMO: Pub/Sub with Subjects

Closing Observables

Closing Observables

- Explicitly

  let subscription = observable$.subscribe(…);
  subscription.**unsubscribe()**;

- Implicitly
  - observable$.pipe(**take(2)**).subscribe(…);
  - observable$.pipe(**first()**).subscribe(…);
  - observable$.pipe(**takeUntil(otherSubject)**).subscribe(…);
- Implicitly with async-Pipe in Angular

  {{ observable$ | **async** }}

- Automatic by Angular
  - Everything, Angular opens is also closed by it

DEMO