# Reactive Extensions for JavaScript

Alex Thalhammer

# Contents

- Overview to Observables
- Generating Observables
- Hot vs. Cold Observables
- Piping operators (lookahead)
- Combination Operators
- Error Handling
- Subjects
- Closing Observables

# Overview

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

# What are observables?

- Represents (asynchronous) data that is published over time

Observable
„Source“

Operator
(z. B. map)

Observer
„Destination“

# Observer

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

**Observer**

Option with more than one parameter is now deprecated!

# Observer

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

# Observer

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

Deprecated!

# Example

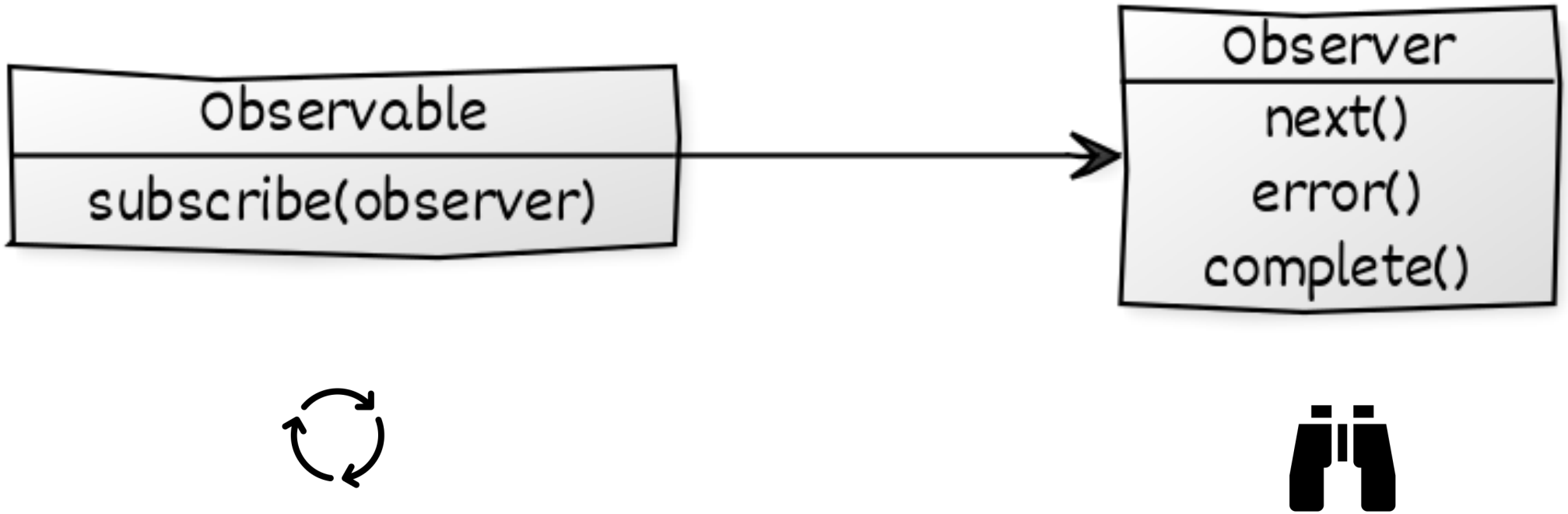
```
this
    .http
    .get("http://www.angular.at/api/...")
    .map(flightDateStr => new Date(flightDateStr))
    .subscribe({
        next: (date) => { … },
        error: (err) => { console.error(err); }
    });
```

# Example with Pipeable Operators

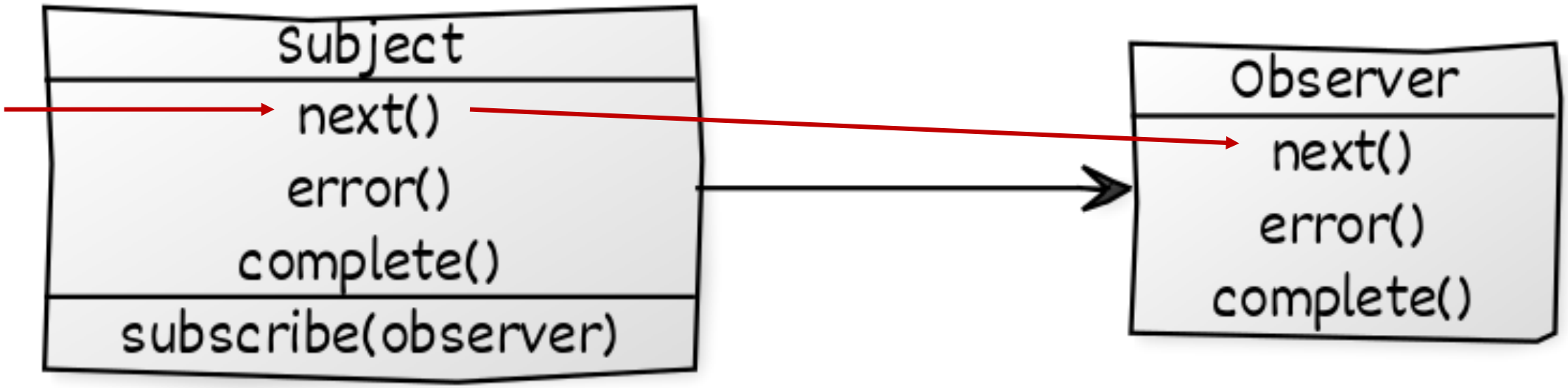
```
import { map } from 'rxjs/operators';

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

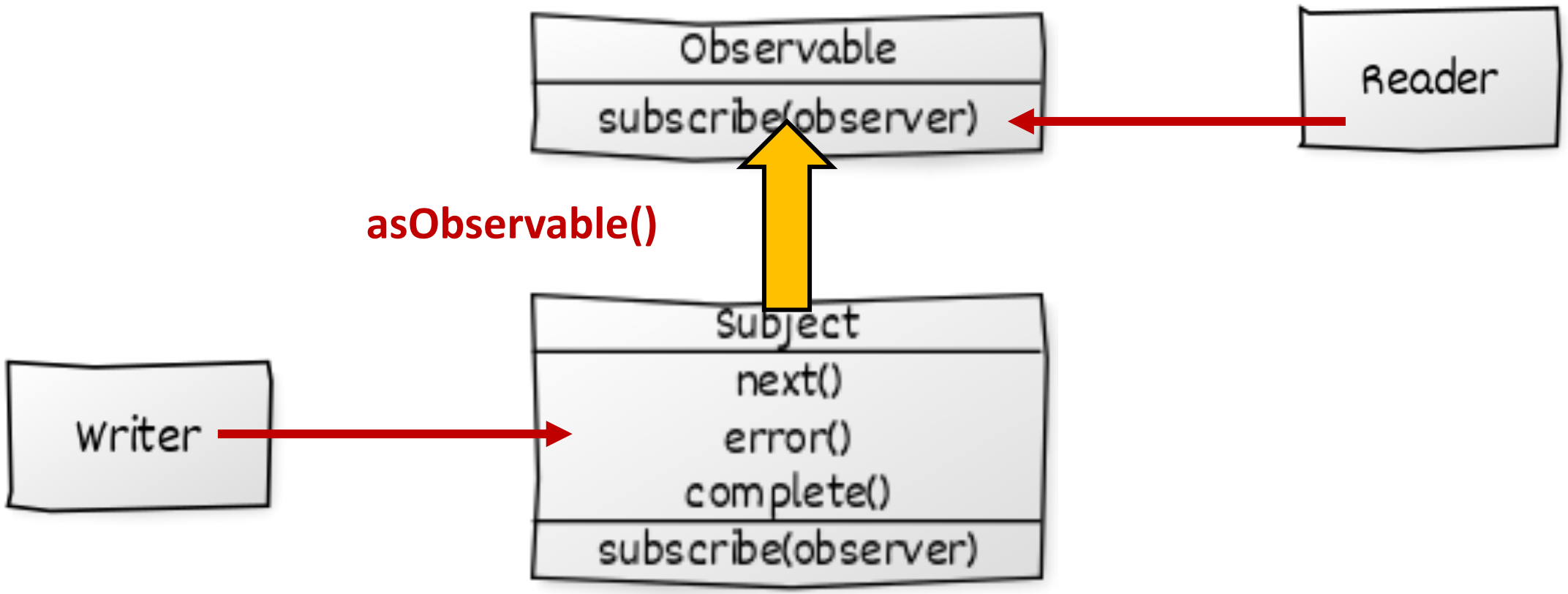
# Observable und Observer



# 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((sender) => {

    sender.next(4711);
    sender.next(815);

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

    sender.complete();

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

Sync/Async, Event-driven

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

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

# Creation Operators (Factories)

[https://www.learnrxjs.io]

fromEvent

of

throwError

interval

timer

# Cold vs. Hot Observables

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

# Cold vs. Hot Observables

## Cold

- Default
- Point to point
- One Sender per consumer
- Lazy: Only starts at subscription

## Hot

- Multicast
- Eager: Sender starts without subscriptions

# Create Hot Observable

```
let o = this.find(from, to)
            .pipe(publish()) as ConnectableObservable<Flight[]>;

o.subscribe(…);

o.connect();

o.subscribe(…);
```

# 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(1));

o.subscribe(…);

o.subscribe(…);
```

# DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

# Operators

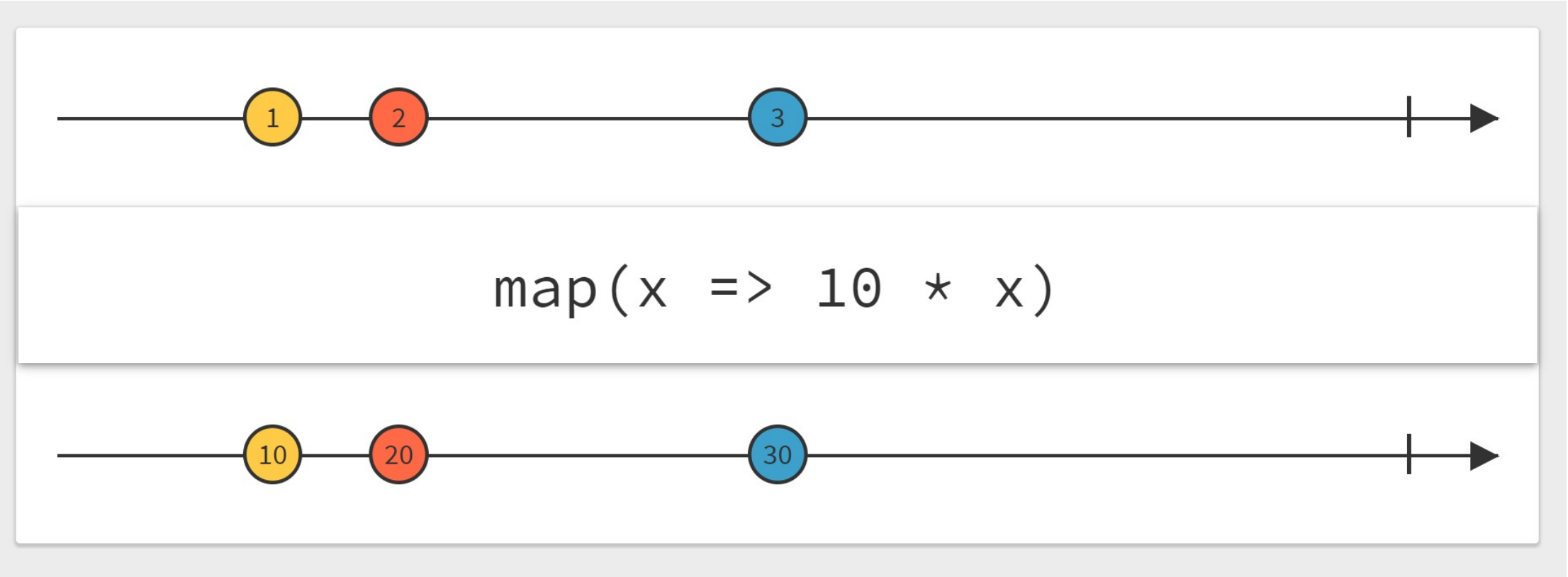
# Transformation Operators

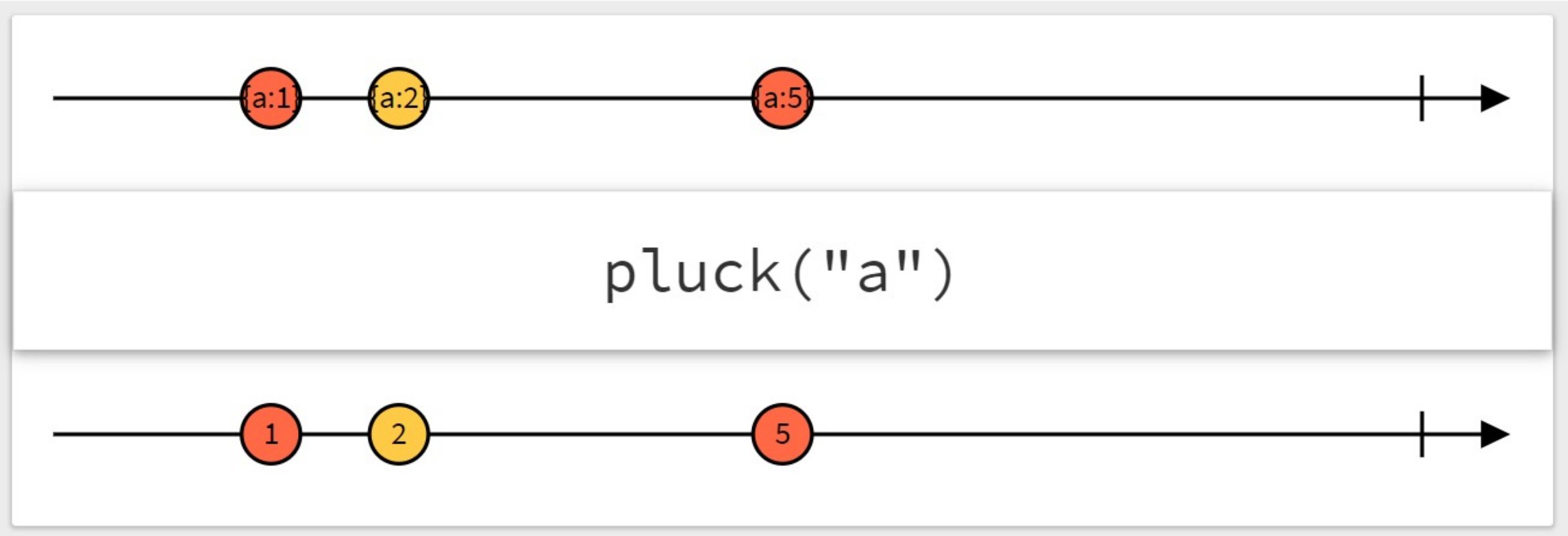
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

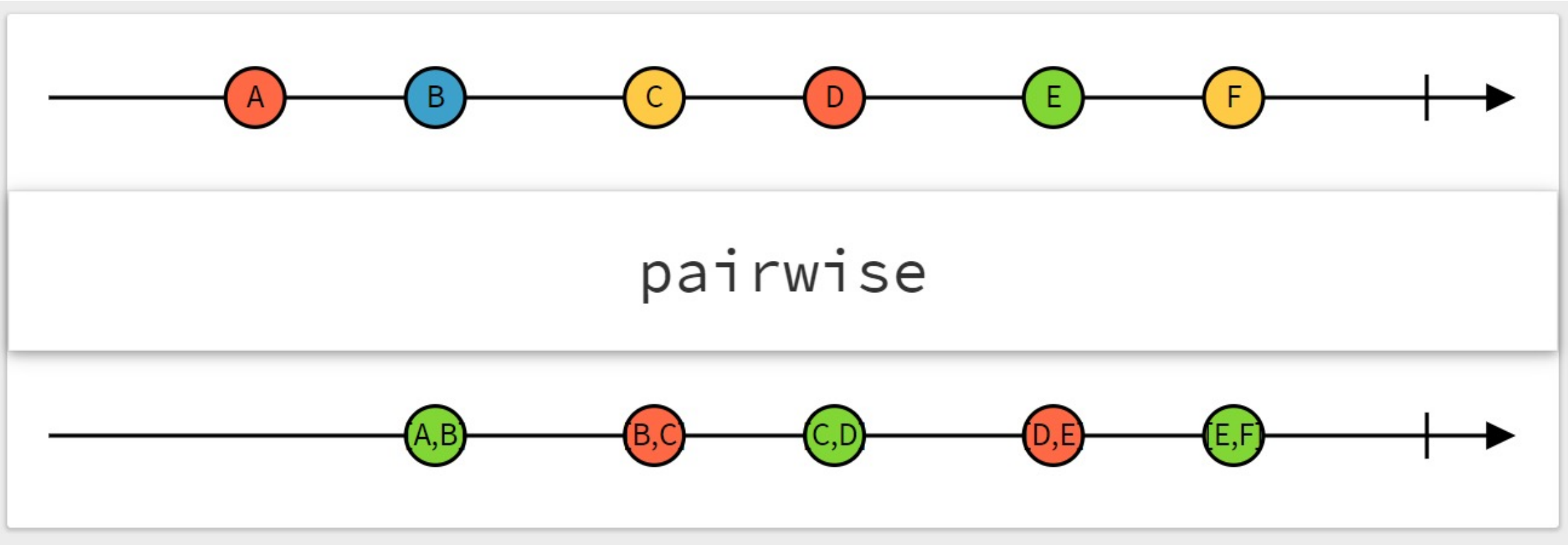
SOFTWARE
ARCHITECT

# Operators

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



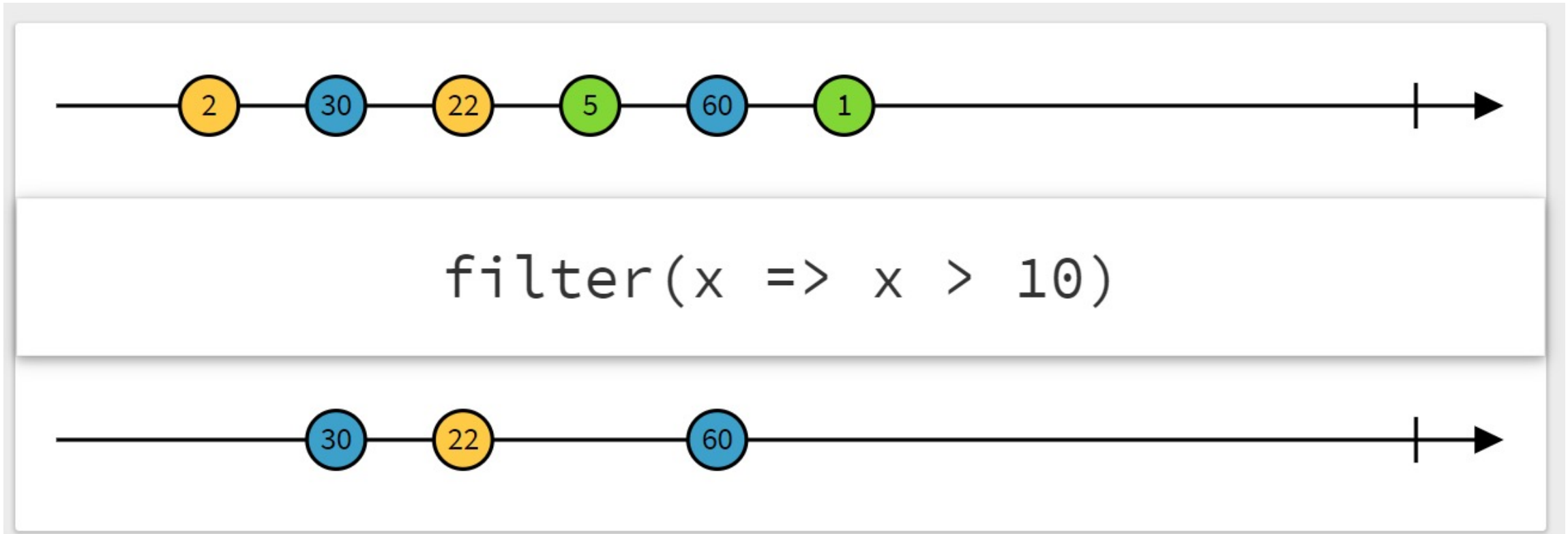

{a:1}
{a:2}
{a:5}
pluck("a")
1
2
5

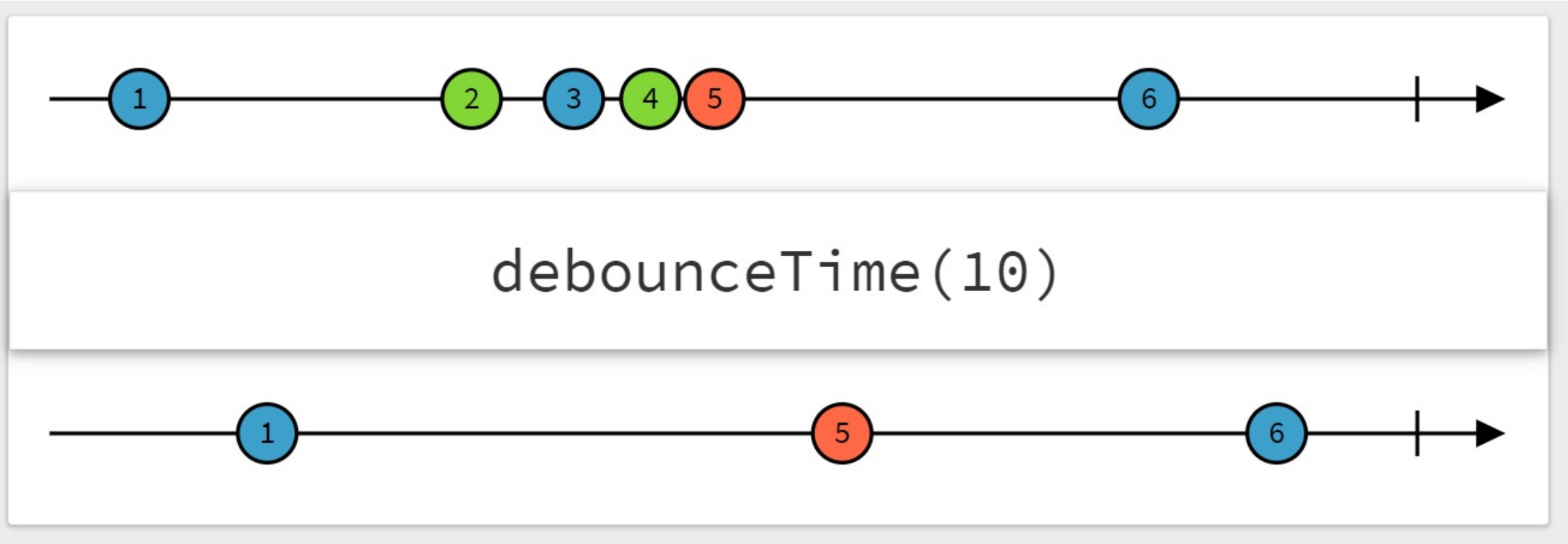

A
B
C
D
E
F
pairwise
A,B
B,C
C,D
D,E
E,F

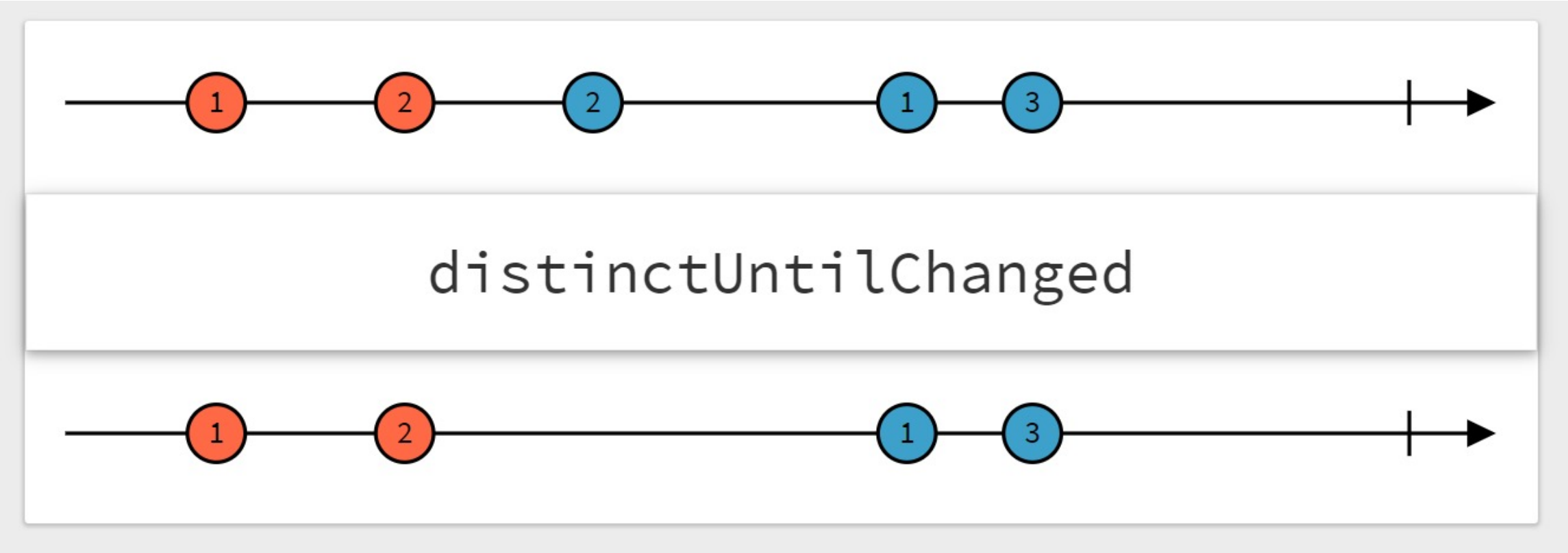
# Filtering Operators

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT


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

# Lab/Demo

Simple Lookahead

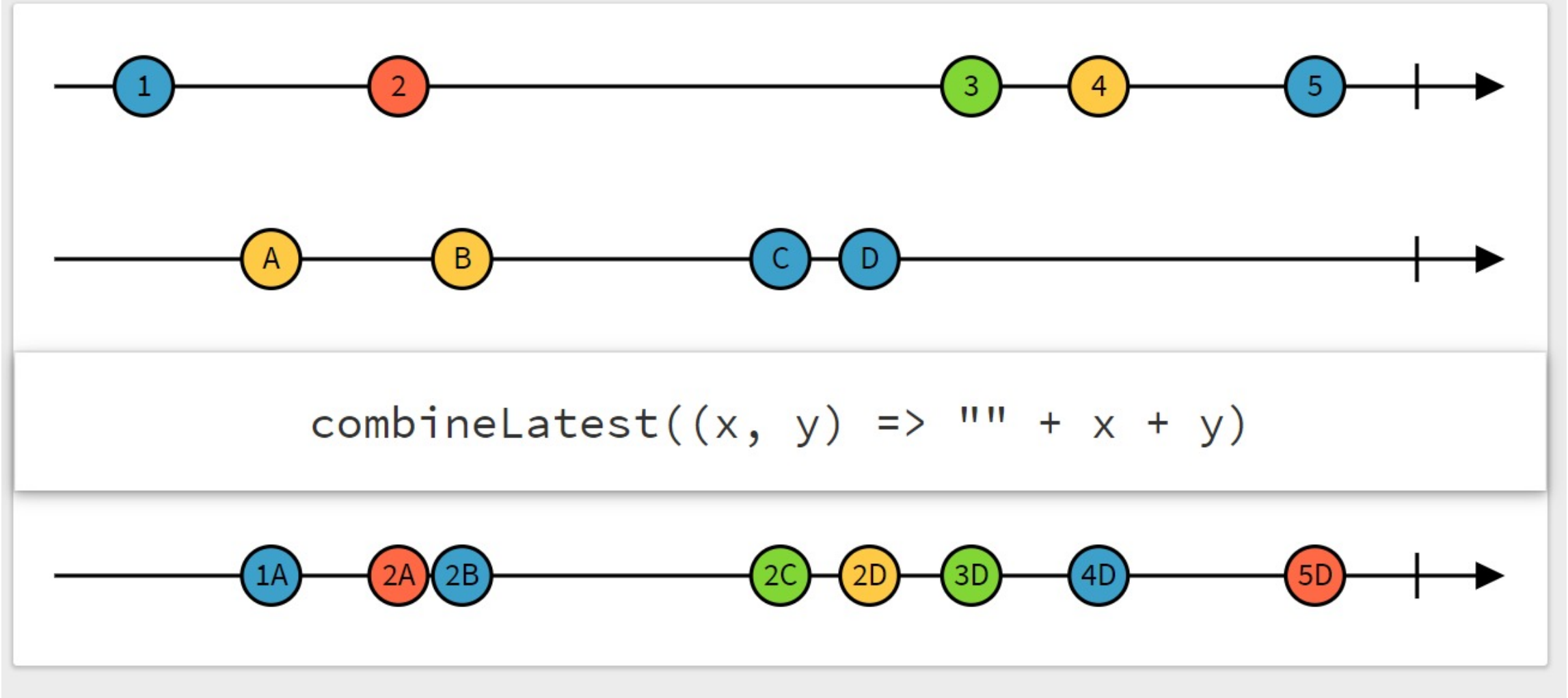
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

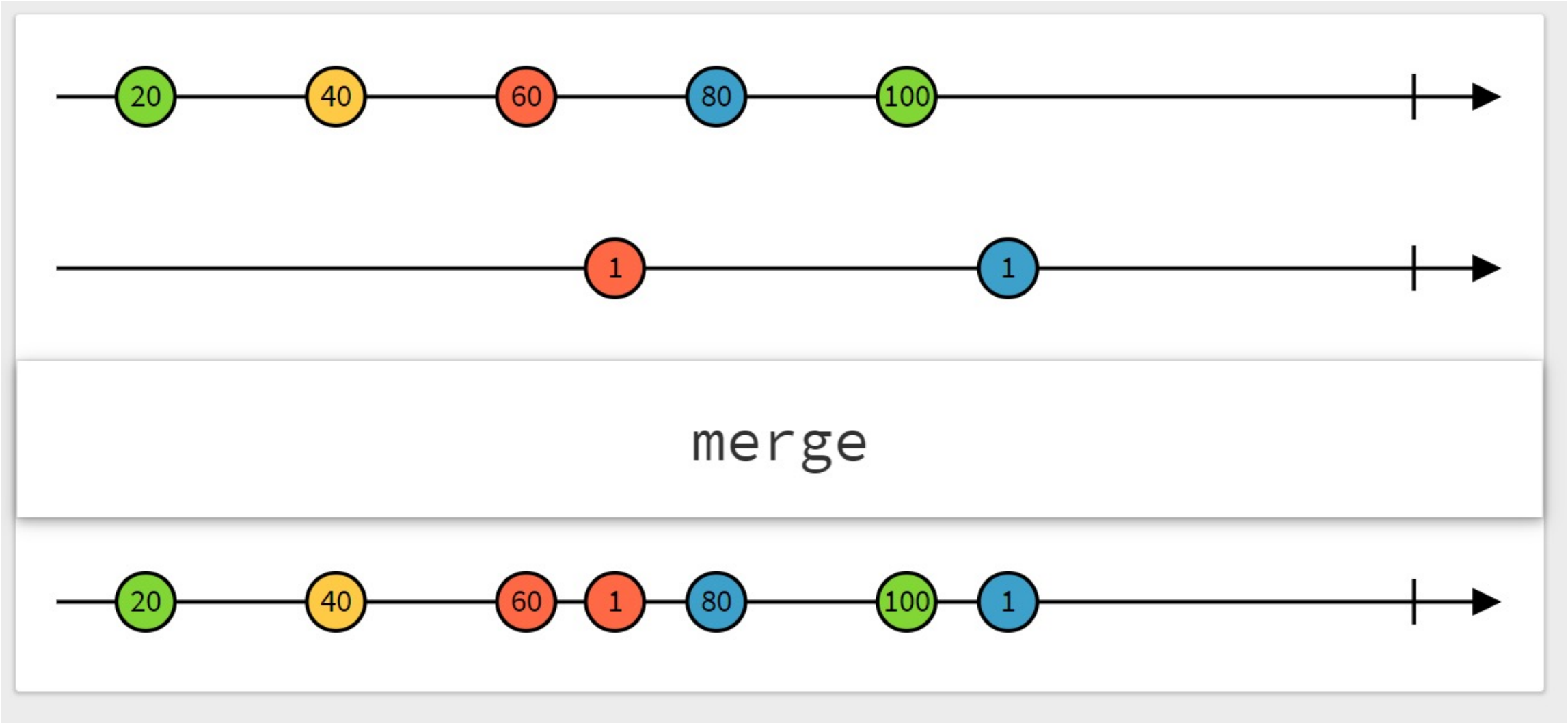
SOFTWARE
ARCHITECT

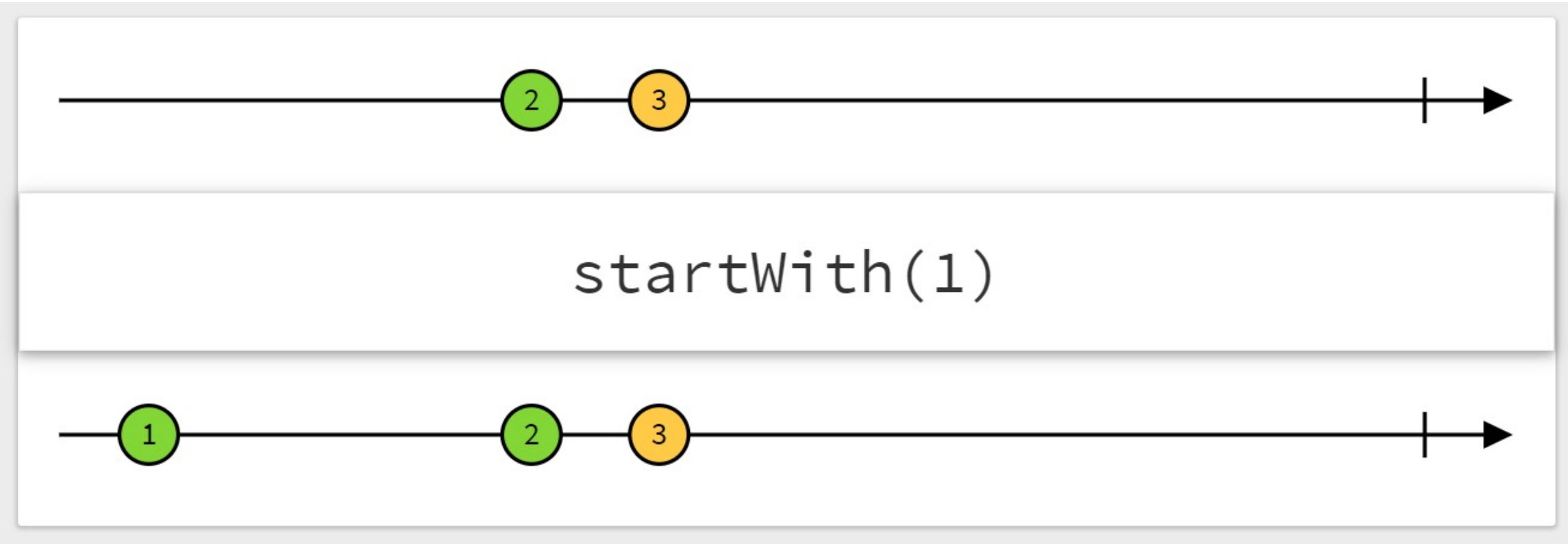
# Combination Operators

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT


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


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


2
3
startWith(1)
1
2
3

# Lab/Demo

Combine Streams

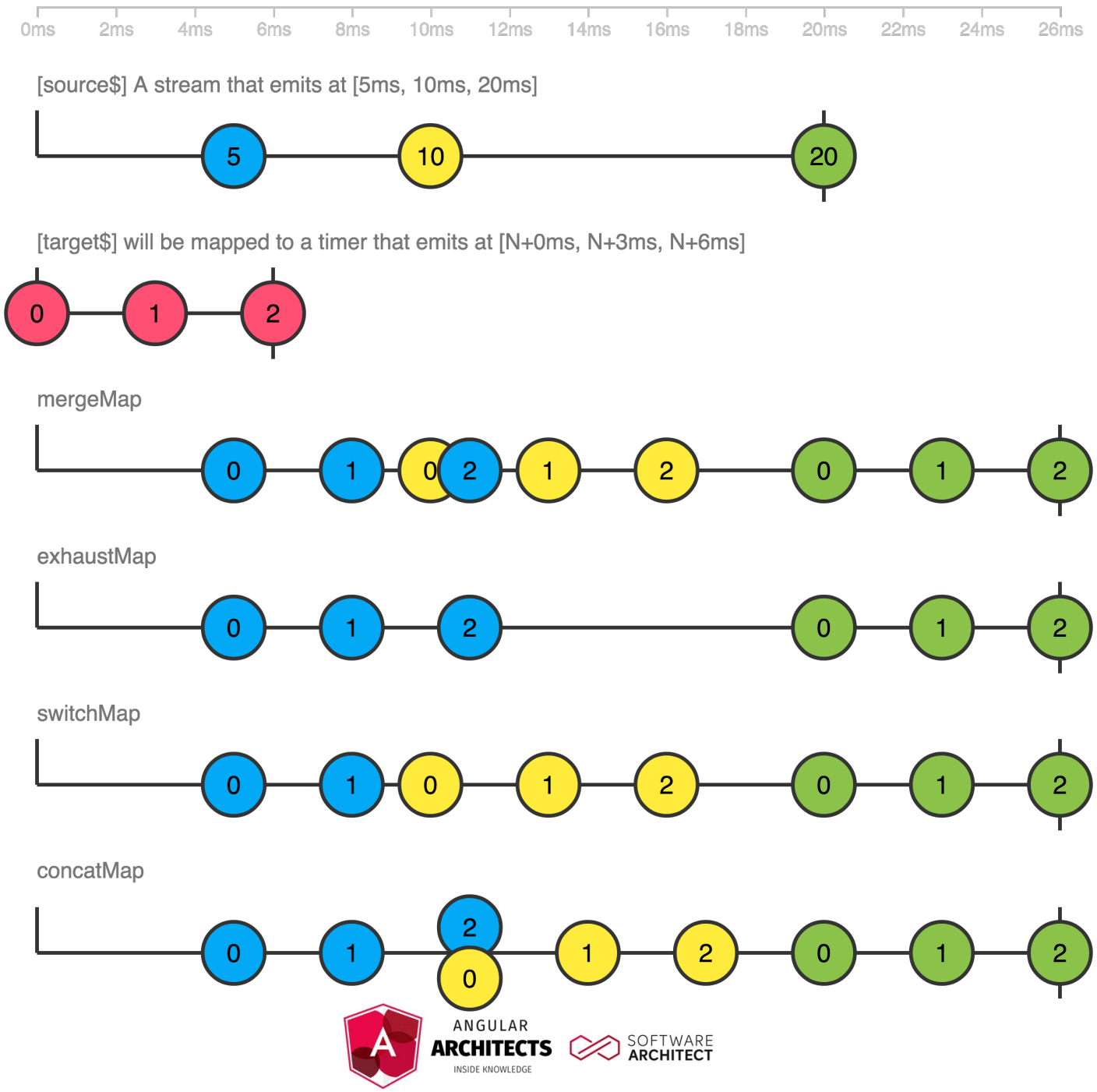
ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE ARCHITECT

# Higher Order Observables

# Operators for Higher Order Observables

- mergeMap
  - merges outer (source) and inner observables
- exhaustMap
  - outer is ignored until inner is finished
- switchMap
  - inner will be completed after next outer
- concatMap
  - outer will be sent after inner is finished


0ms 2ms 4ms 6ms 8ms 10ms 12ms 14ms 16ms 18ms 20ms 22ms 24ms 26ms
[source$] A stream that emits at [5ms, 10ms, 20ms]
5
10
20
[target$] will be mapped to a timer that emits at [N+0ms, N+3ms, N+6ms]
0
1
2
mergeMap
0
1
0
2
1
2
0
1
2
exhaustMap
0
1
2
0
1
2
switchMap
0
1
0
1
2
0
1
2
concatMap
0
1
2
0
1
2
0
1
2
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE
SOFTWARE
ARCHITECT

# Error Handling

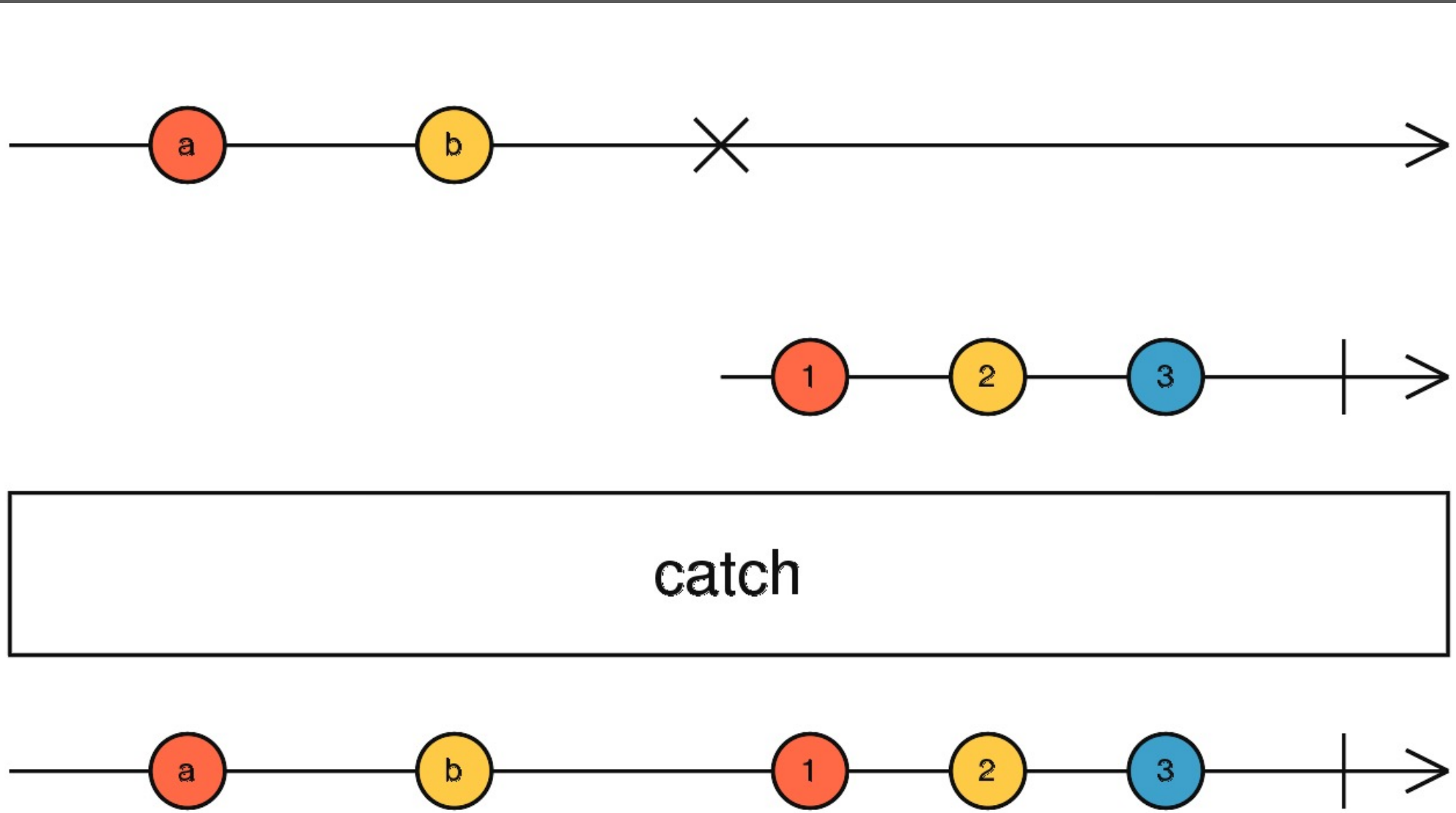
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

# Operators for Error Handling

- catchError
- retry
- retryWhen

- throwError


a
b
1
2
3
catch
a
b
1
2
3

# Lab/Demo

Error Handling

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

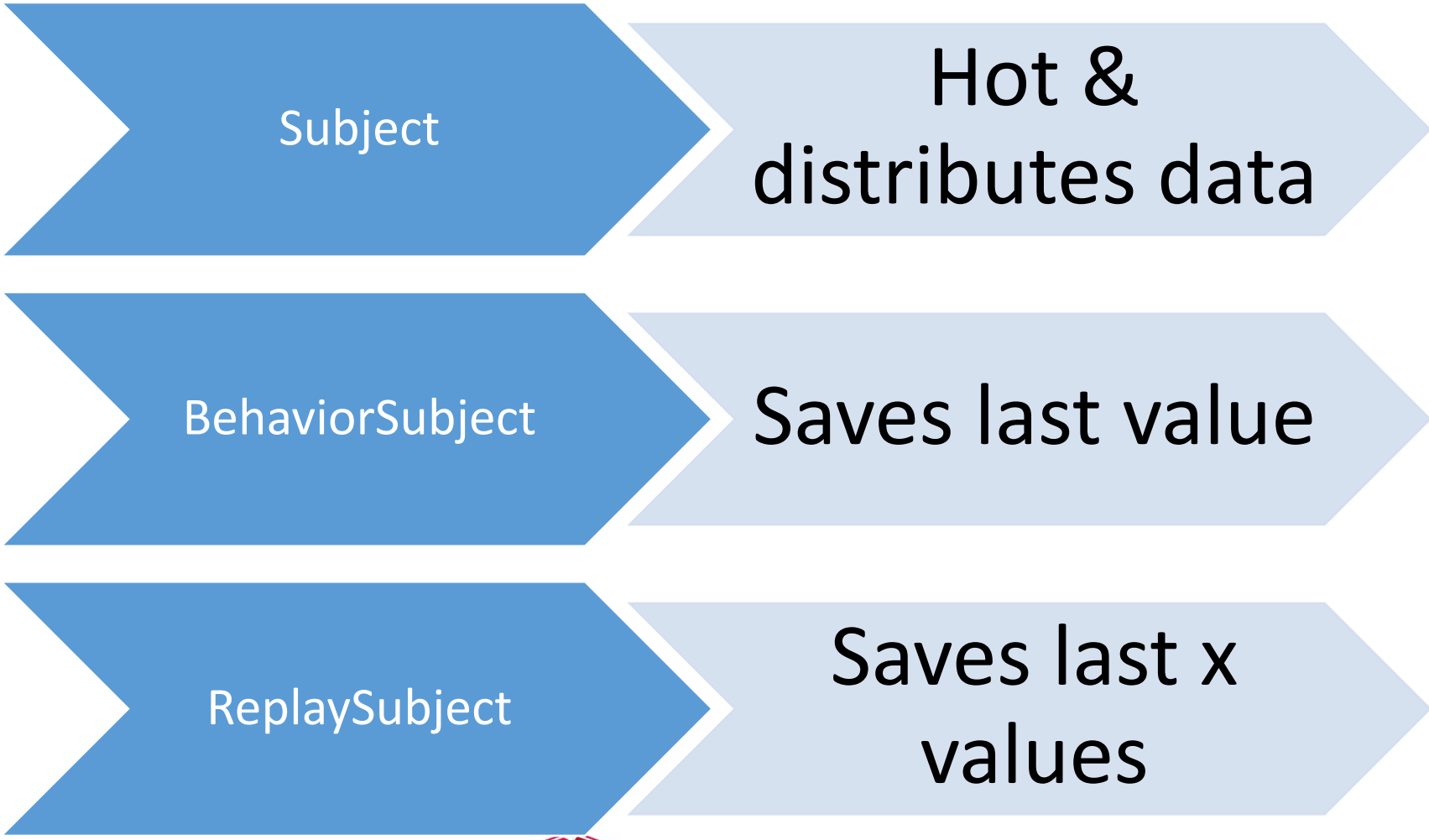
SOFTWARE ARCHITECT

# Subjects

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

# 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: TakeUntil in ngDestroy

# Like this topic?

- Marble Diagrams
  - http://rxmarbles.com
- Other Links
  - https://rxjs.dev/guide/overview
    - https://reactive.how/rxjs/
  - https://www.learnrxjs.io/
  - https://angular.io/guide/rx-library
- Official documentation
  - http://reactivex.io/rxjs/class/es6/Observable.js~Observable.html