Reactive Extensions for JavaScript

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

ANGULARarchitects.io

Contents

Overview

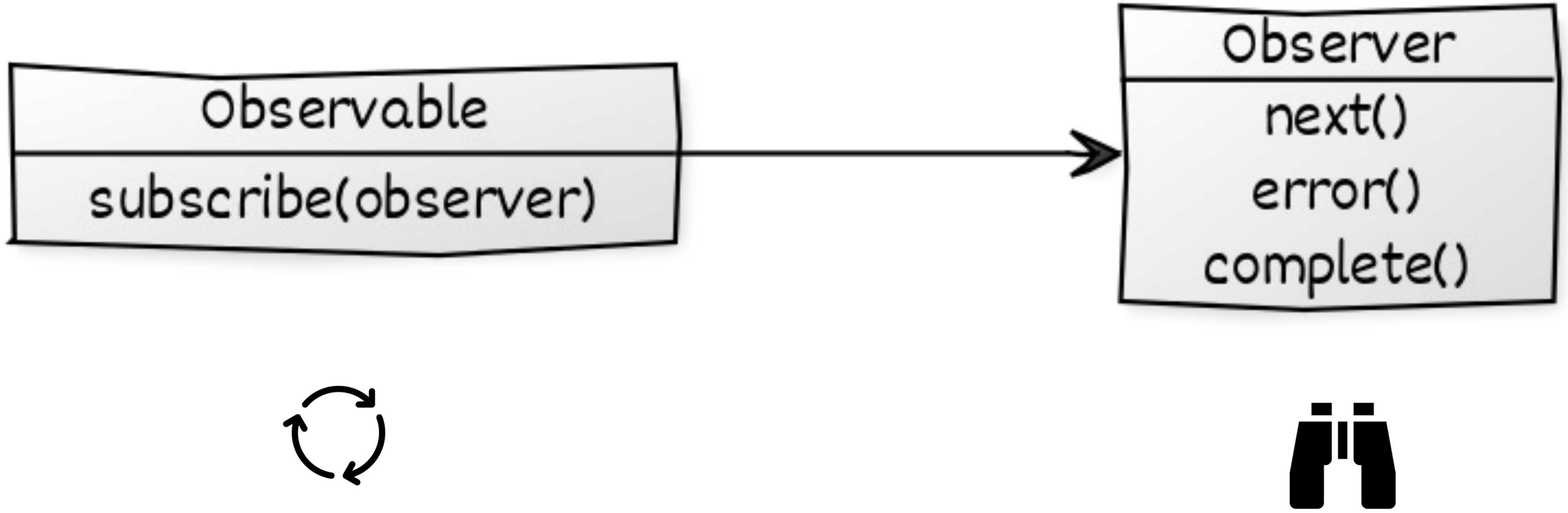
ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

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

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Observer

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

Observer

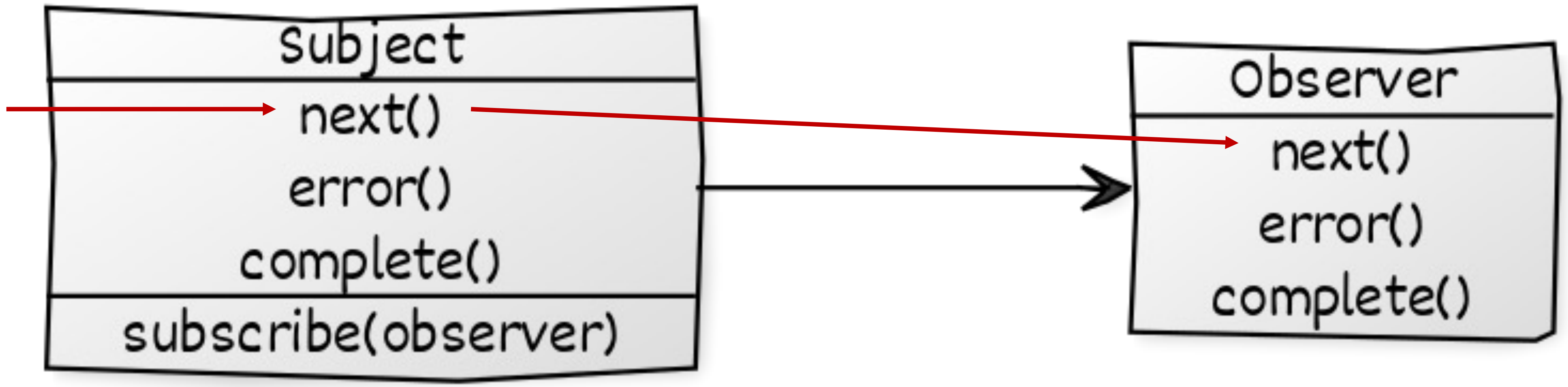
ANGULAR ARCHITECTS INSIDE KNOWLEDGE

Example with Pipeable Operators

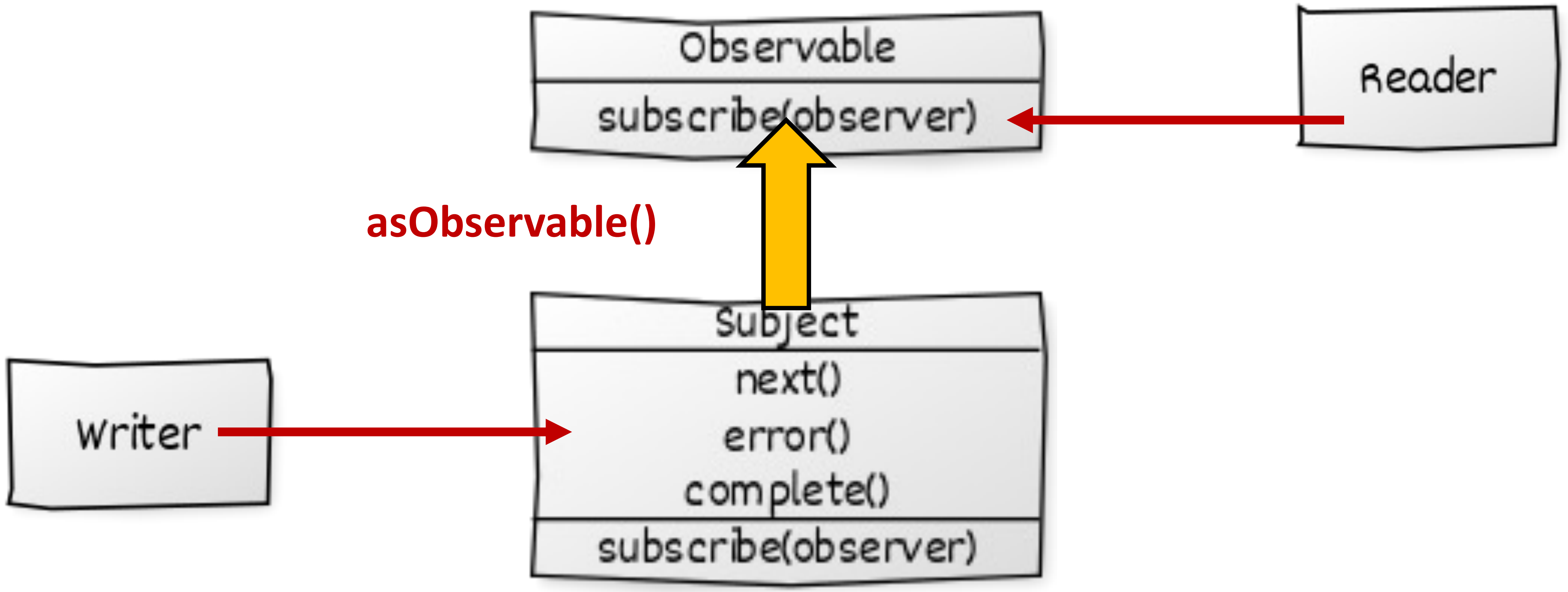
```
import { map } from 'rxjs/operators';

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

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

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'); };
});
```

$\left.\right\}$ **Sync & Async, Event-driven**

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

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

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Creation Operators (Factories)

[https://www.learnrxjs.io]

- fromEvent
- of
- throwError
- interval
- timer

Cold vs. Hot Observables

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Cold vs. Hot Observables

Cold	Hot
• Point to point • Lazy: Only starts with subscription	• Multicast • Eager: Sender starts without subscriptions

Default

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

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Operators

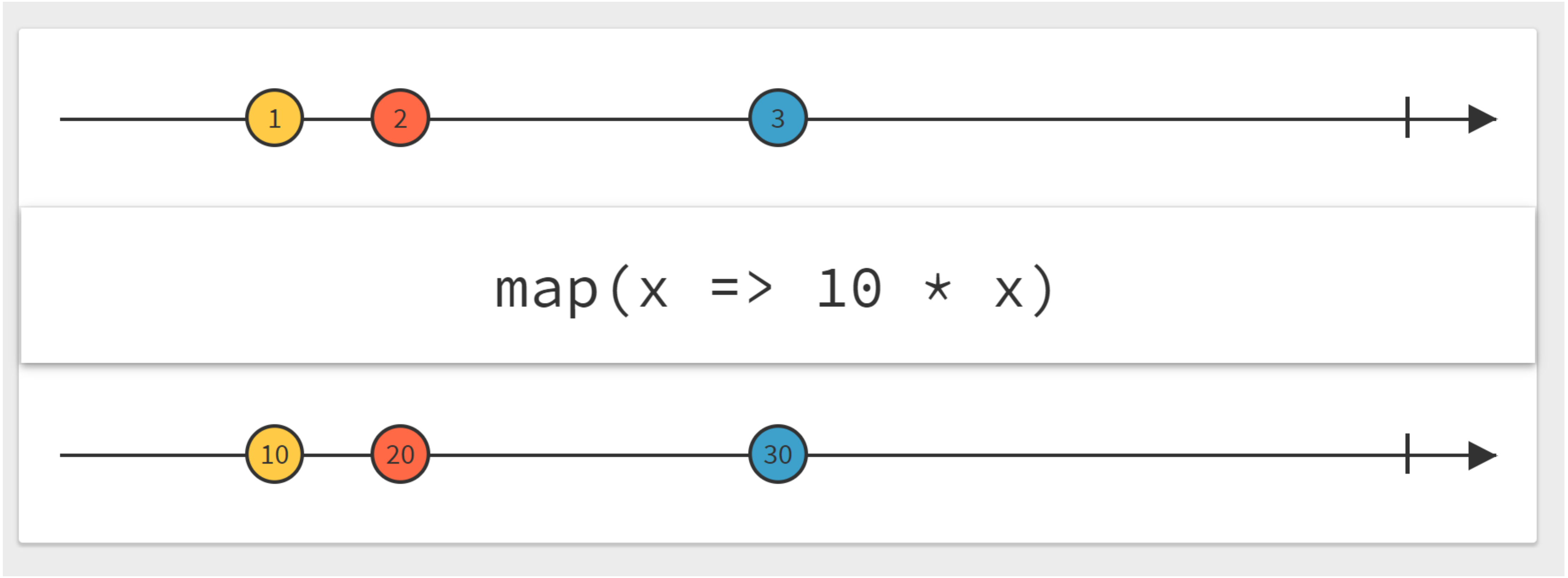
ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

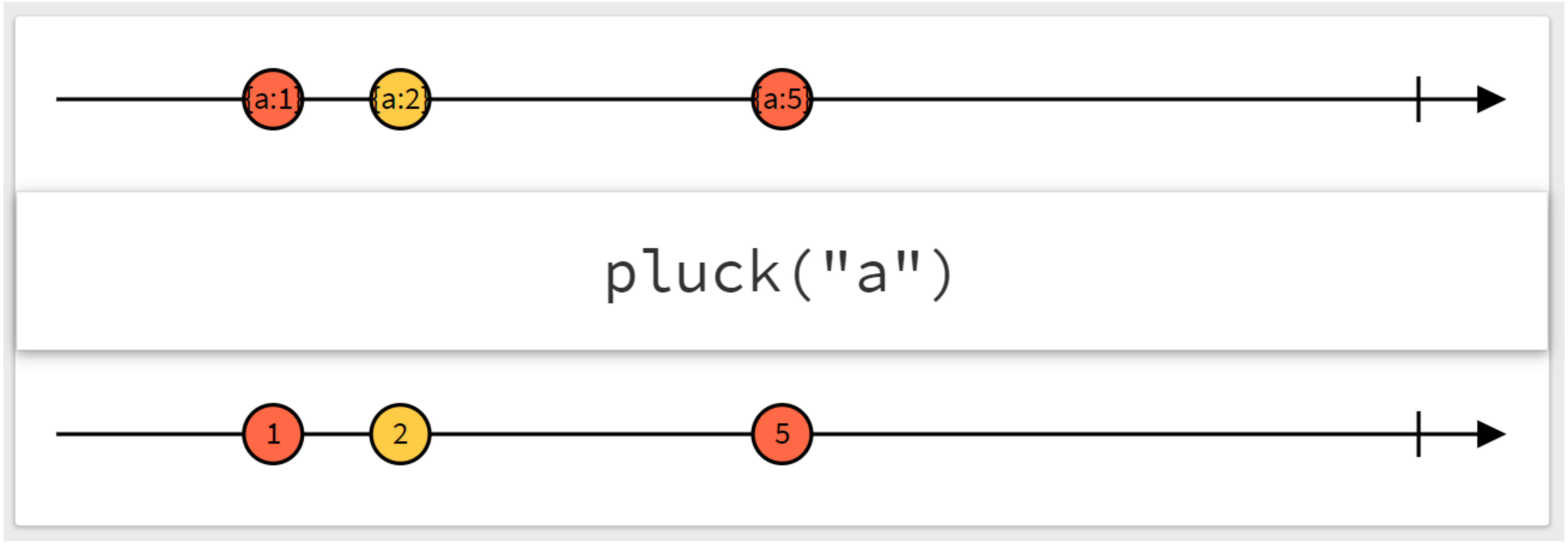
Transformation Operators

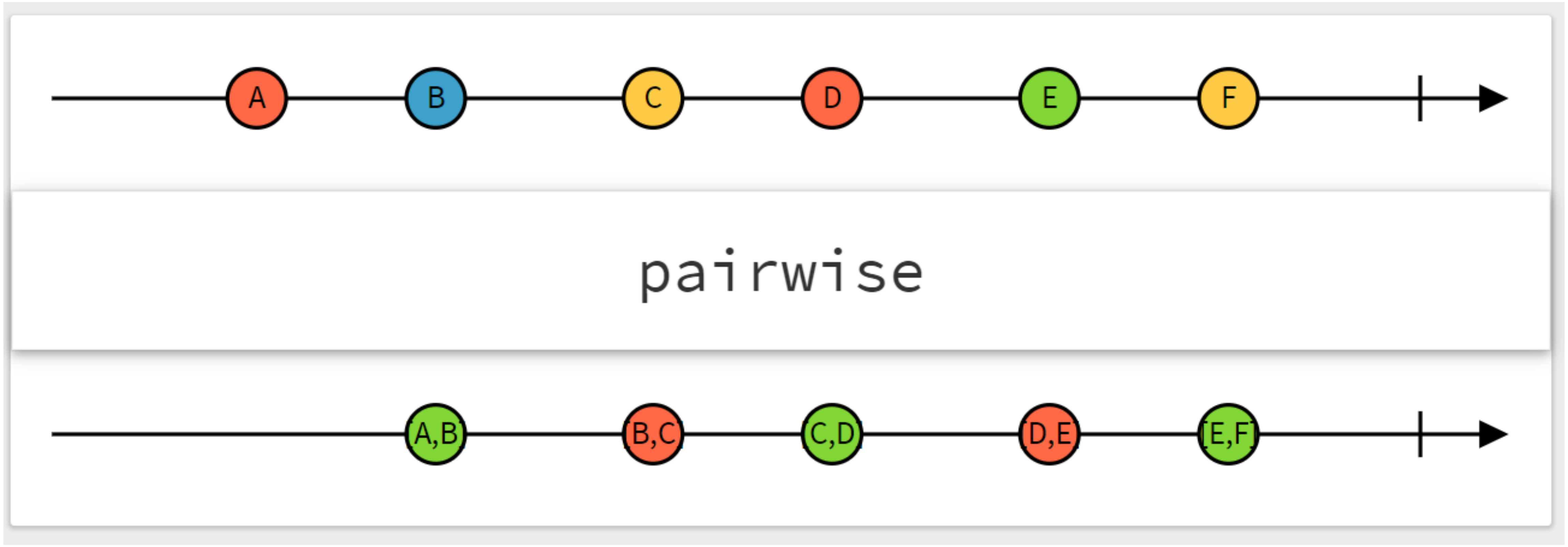
ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Operators

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

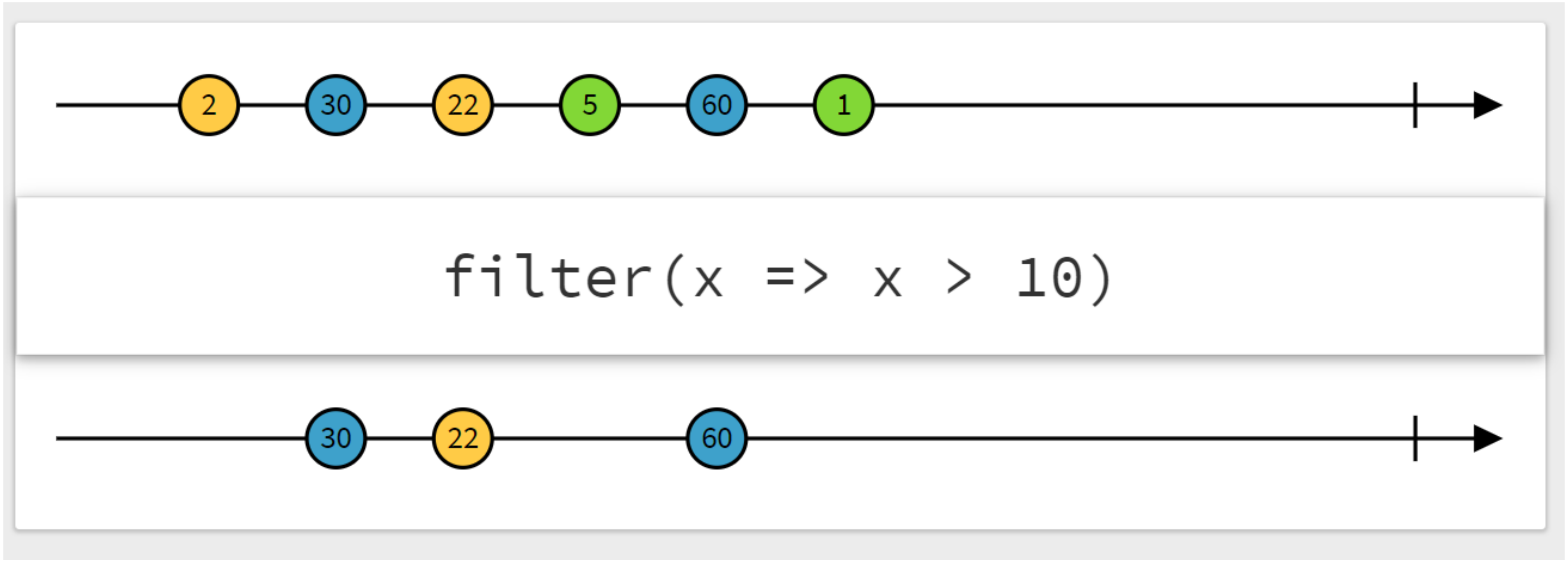



{a:1}
{a:2}
{a:5}
pluck("a")
1
2
5

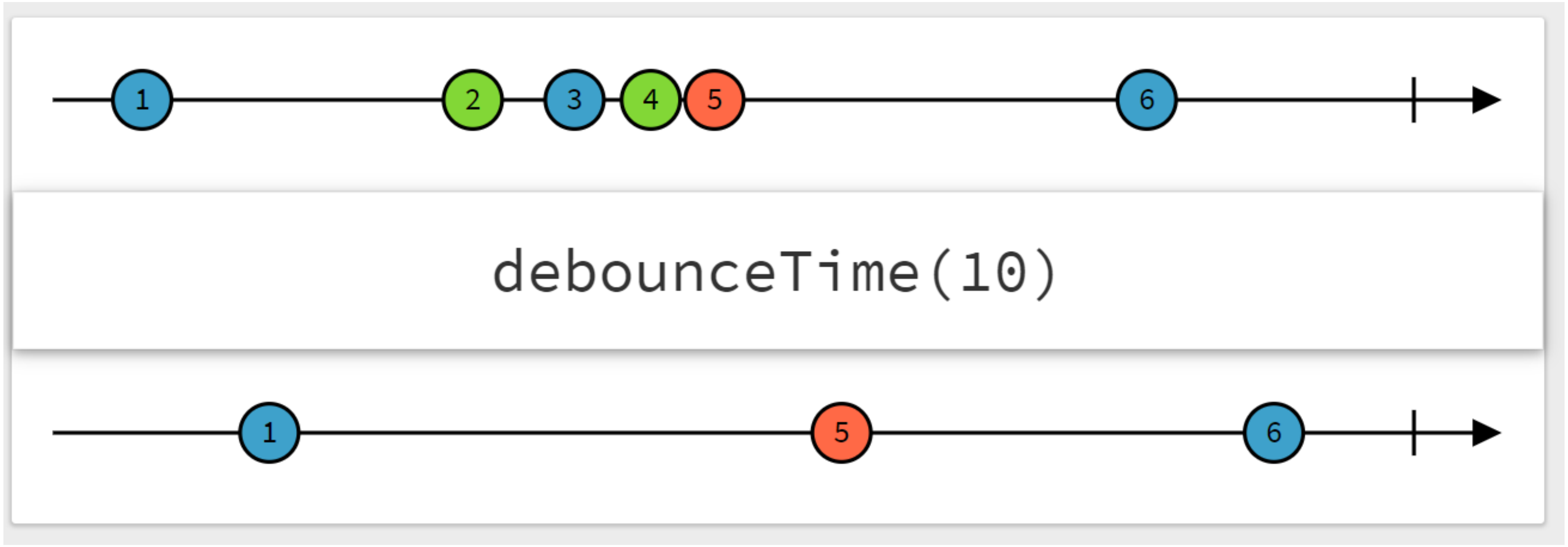


ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

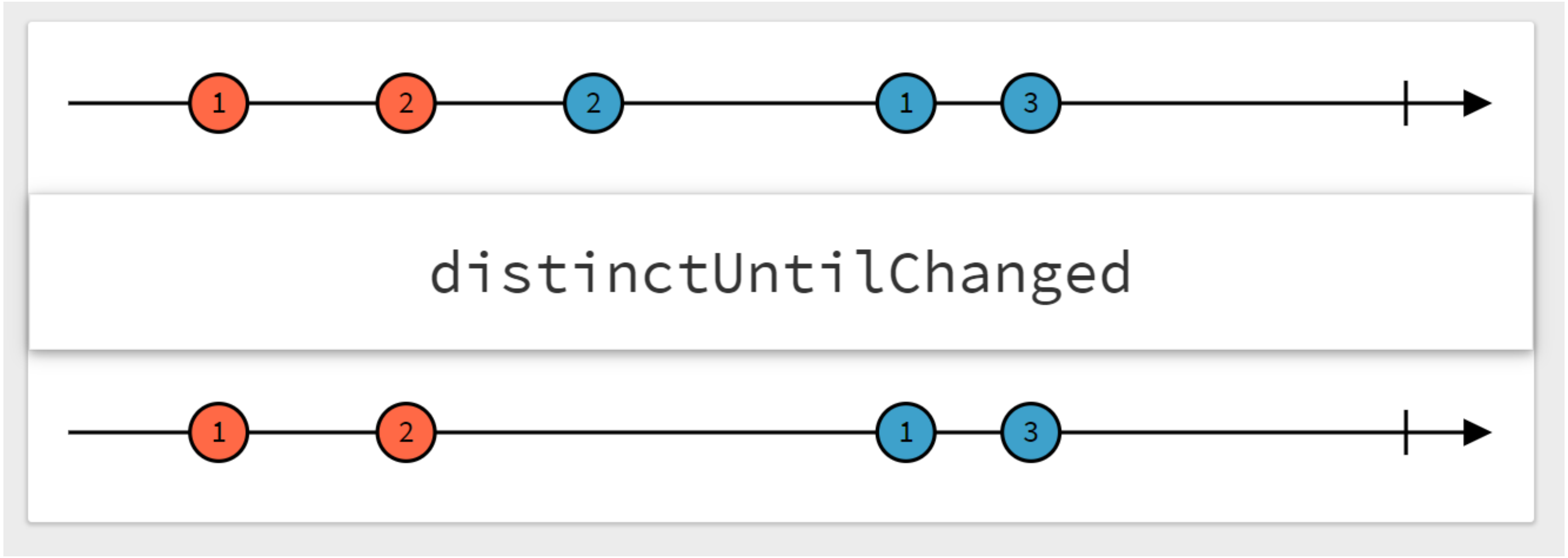
Filtering Operators



ANGULAR ARCHITECTS
INSIDE KNOWLEDGE



ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

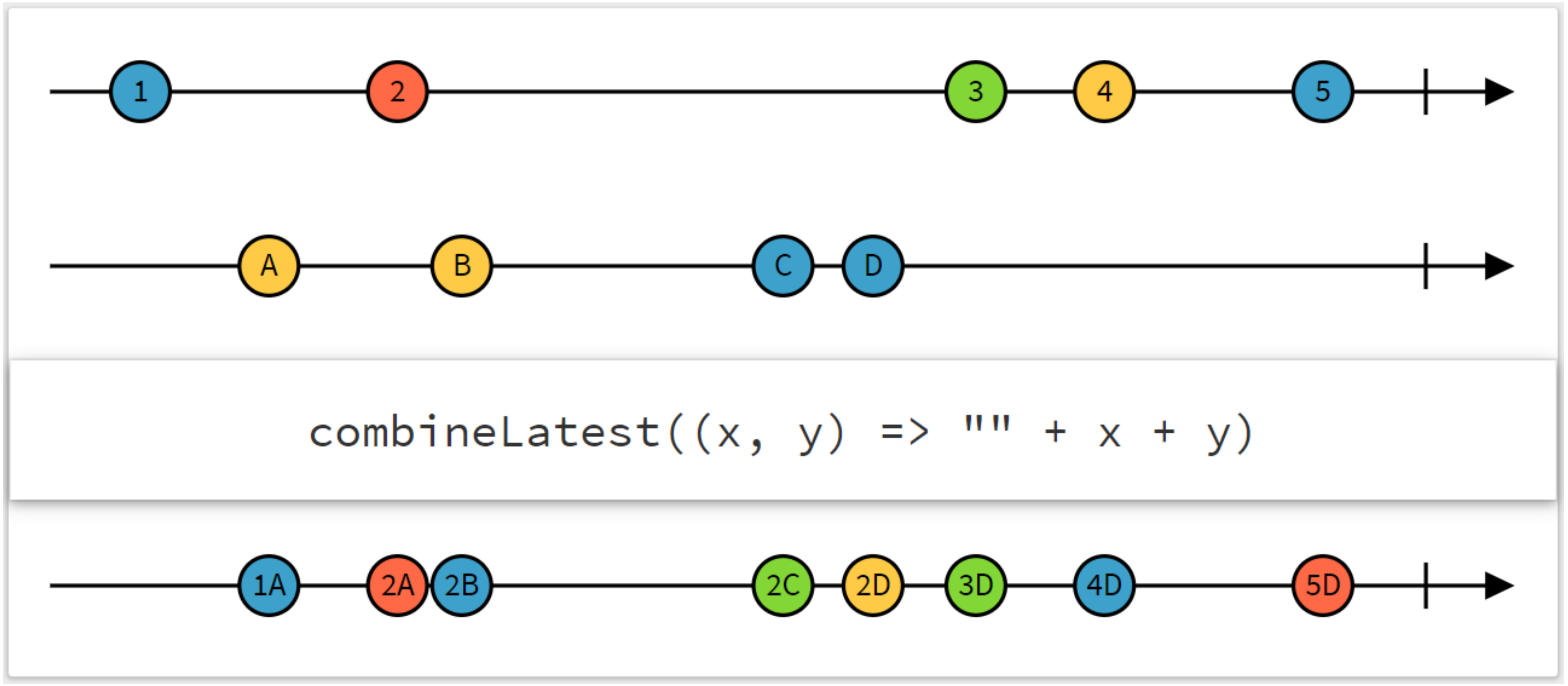

1
2
2
1
3
distinctUntilChanged
1
2
1
3

DEMO: Lookahead

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

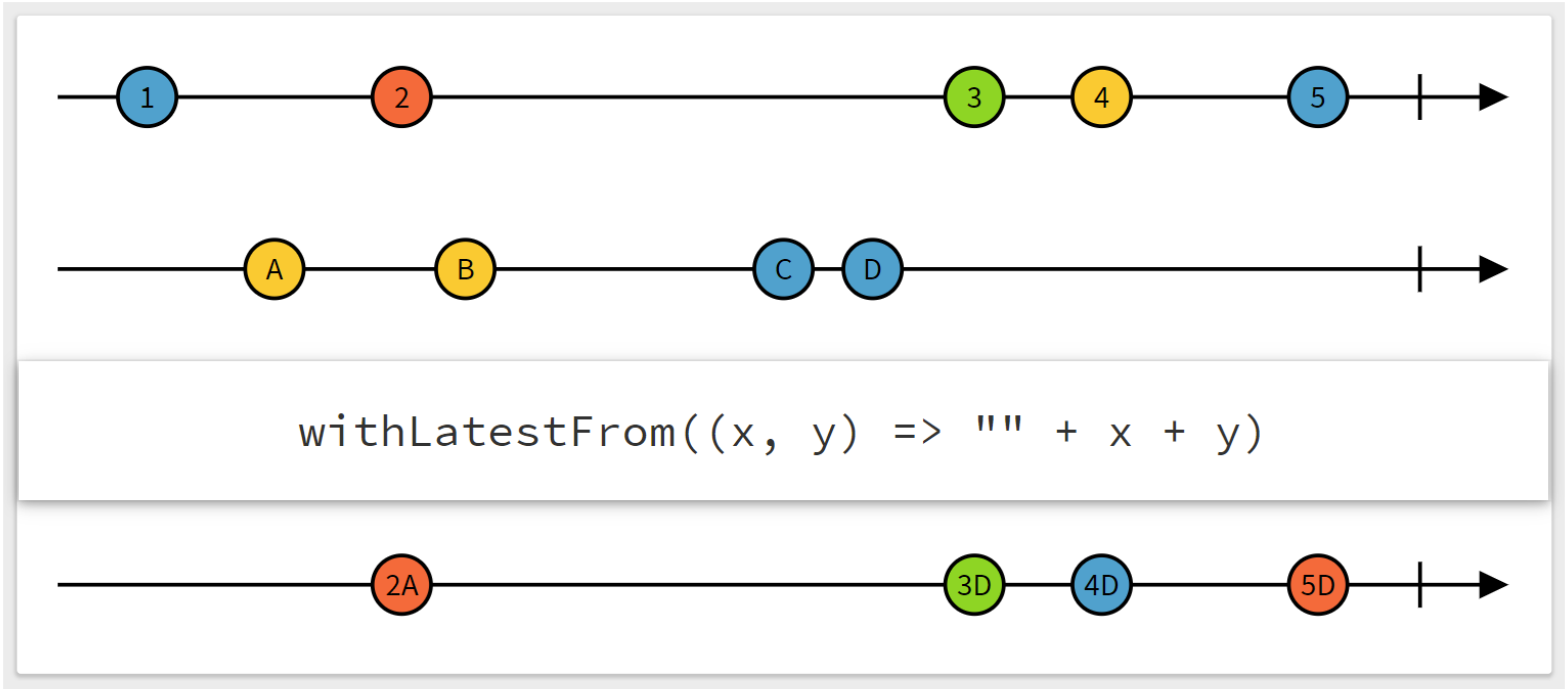
LAB

Combination Operators



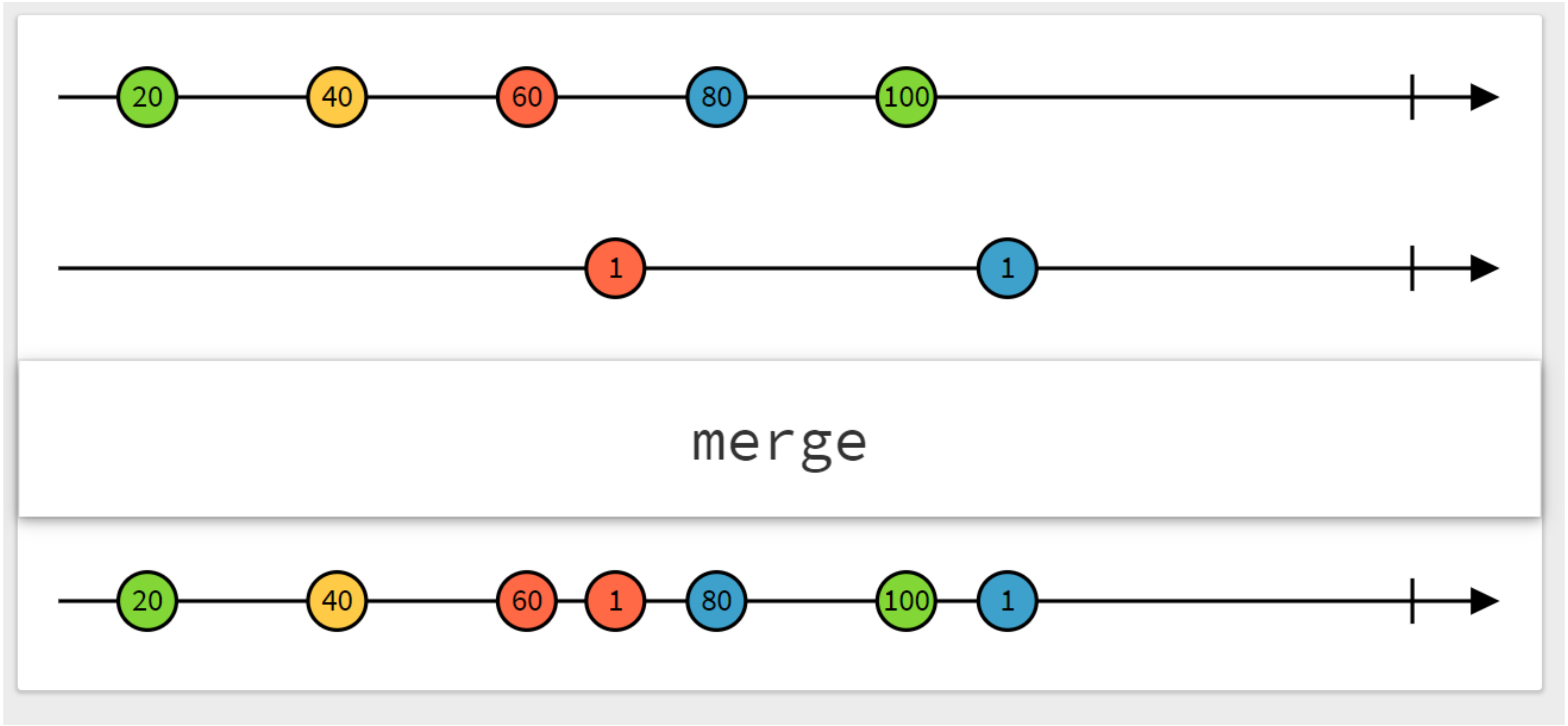
```
combineLatest((x, y) => "" + x + y)
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

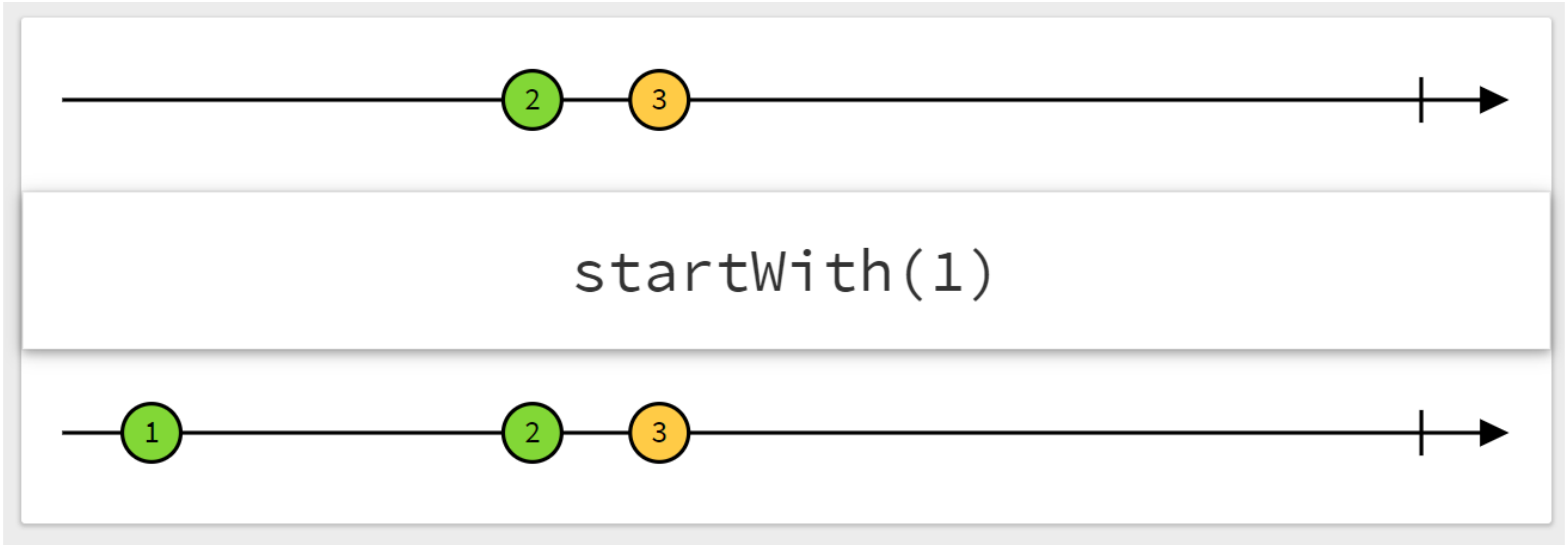


```
withLatestFrom((x, y) => "" + x + y)
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE



ANGULAR ARCHITECTS
INSIDE KNOWLEDGE



ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

DEMO

Labs

Higher Order Observables

Operators for Higher Order Observables

- switchMap
- mergeMap
- concatMap
- exhaustMap

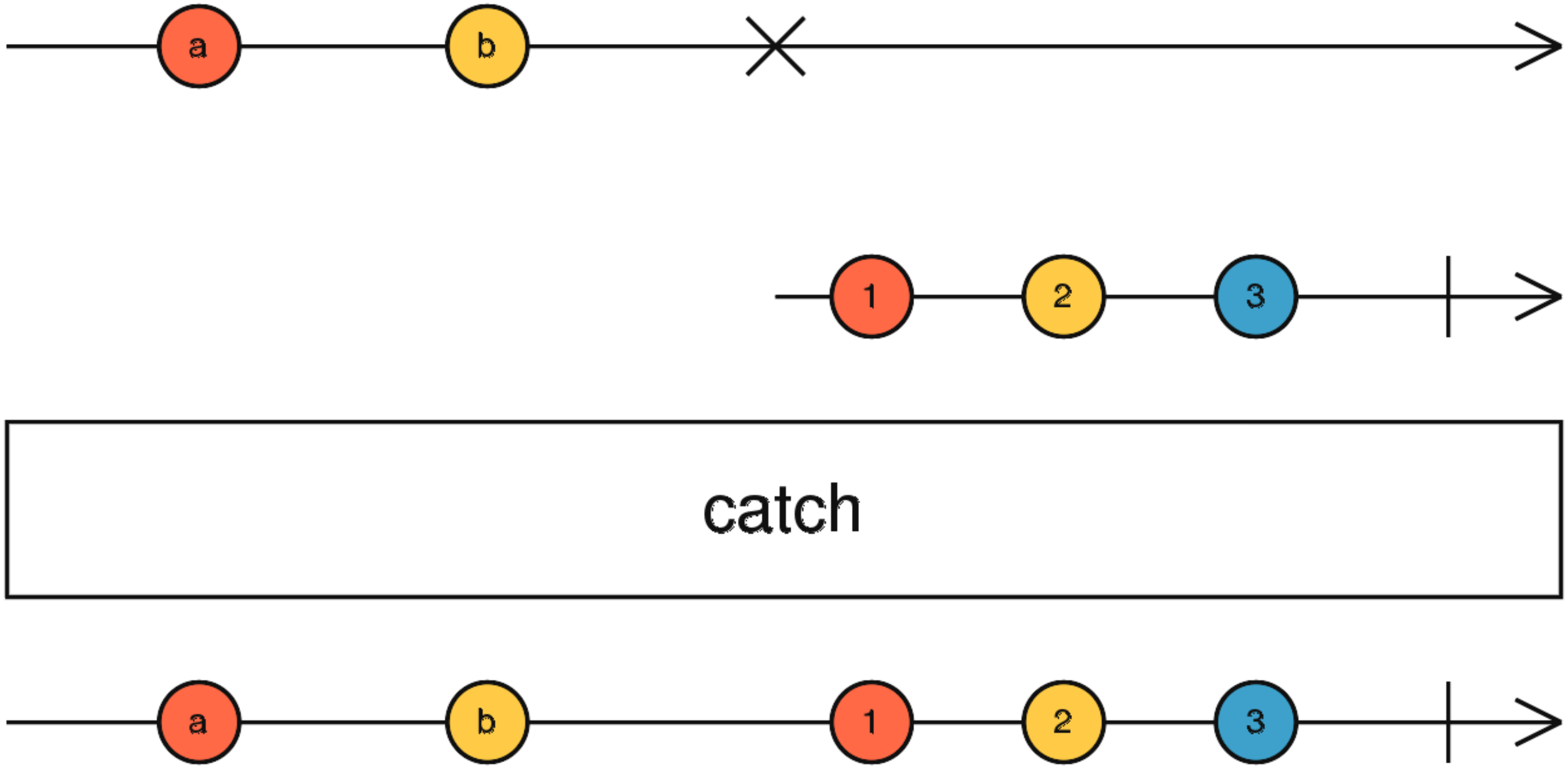
Error Handling

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Operators for Error Handling

- catchError
- retry
- retryWhen

- throwError


a
b
1
2
3
catch
a
b
1
2
3

Subjects

Subjects

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

DEMO: Pub/Sub with Subjects

Closing Observables

ANGULAR ARCHITECTS INSIDE KNOWLEDGE

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

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE