ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Reactive Extensions for JS
Operators

Alex Thalhammer

Outline

- Motivation
- Example
- Transformation Operators
- Filtering Operators
- Combination Operators
- Error Handling
- Higher Order Observables
- Reference

ANGULAR ARCHITECTS INSIDE KNOWLEDGE
SOFTWARE ARCHITECT

Motivation

Observables vs Promises – Operators

Observables (Streams)	Promises (Single Event)
More features	Less powerful
Can emit zero, **one or multiple** values over time.	Emit a **single** value at a time.
Lazy: they're not executed until we subscribe using the subscribe() method.	**Eager**: execute immediately after creation.
Subscriptions are **cancellable** using the unsubscribe() method, which stops the listener from receiving further values.	Are **not cancellable**.
RxJS provides a **ton of functionality** to operate on observables like the map, forEach, filter, reduce, retry, and retryWhen operators.	Don't provide any operations.
Deliver errors to the subscribers.	Push errors to the child promises.
Used by Angular in HTTP Client & Route Params	Used by Angular in Router.navigate

ANGULAR ARCHITECTS INSIDE KNOWLEDGE SOFTWARE ARCHITECT

Example

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Example with Pipeable Operators

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

this.httpClient.get<Booking>("http://www.angular.at/api/...")
    .pipe(map((booking) => new Date(booking.date)))
    .subscribe({
        next: (date) => { … },
        error: (err) => { console.error(err); }
        complete: () => { console.log('complete'); }
    });
```


ANGULAR ARCHITECTS INSIDE KNOWLEDGE
SOFTWARE ARCHITECT

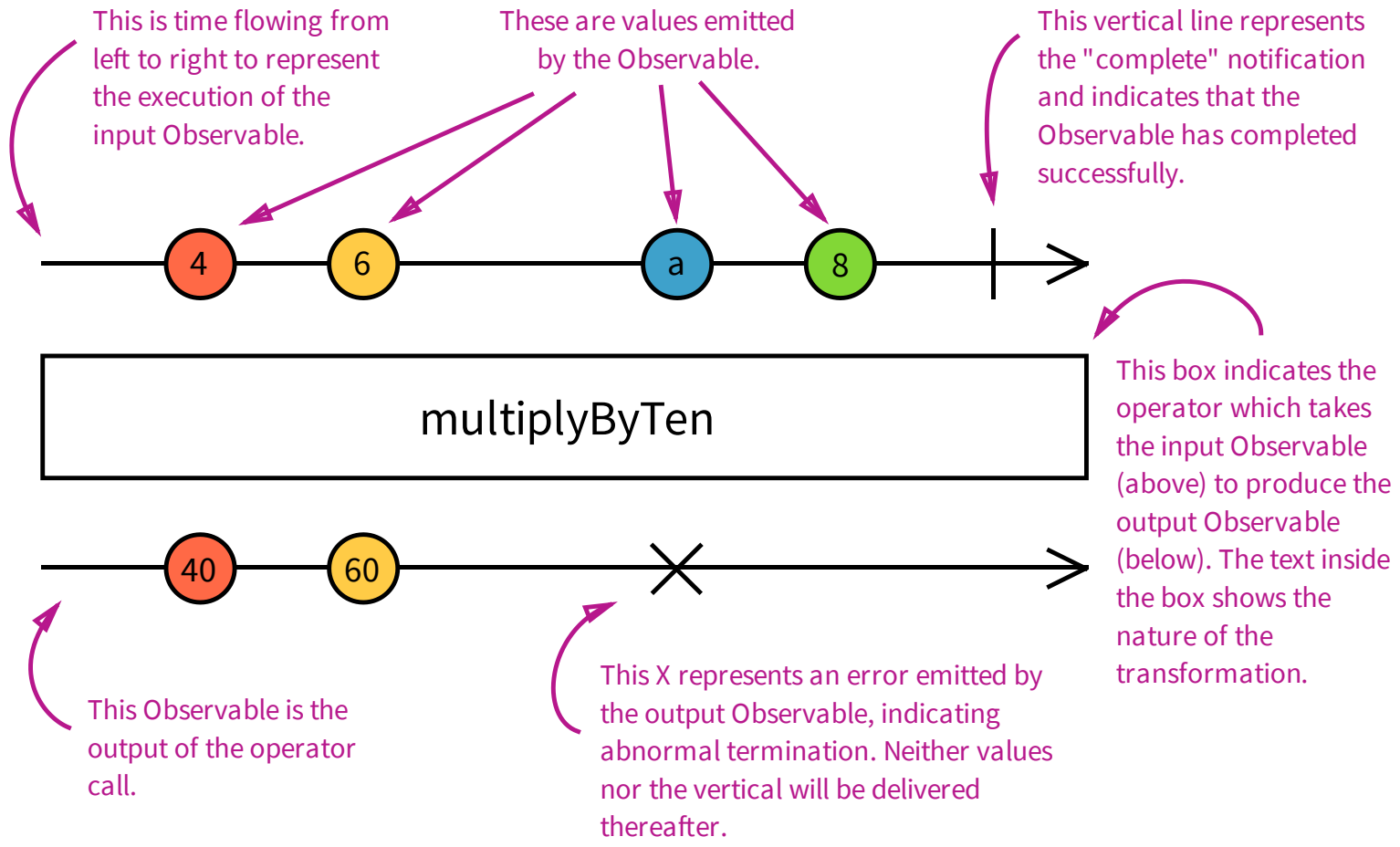
Transformation Operators

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

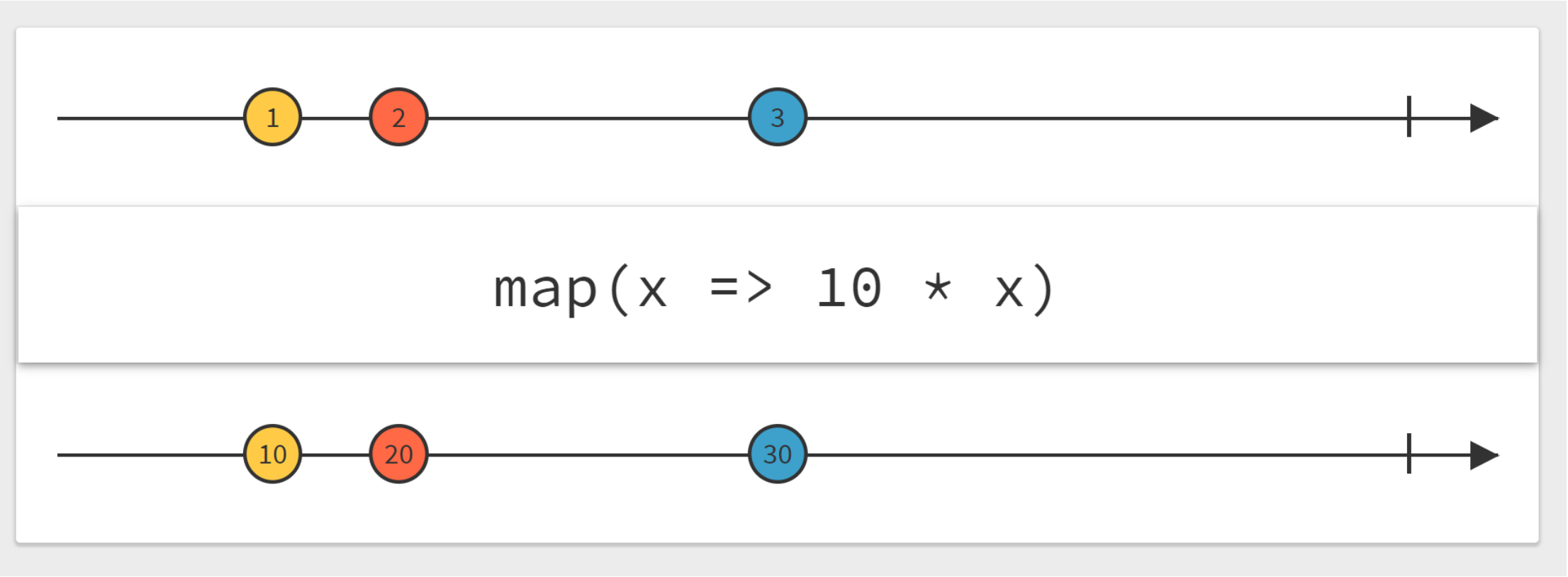
SOFTWARE
ARCHITECT

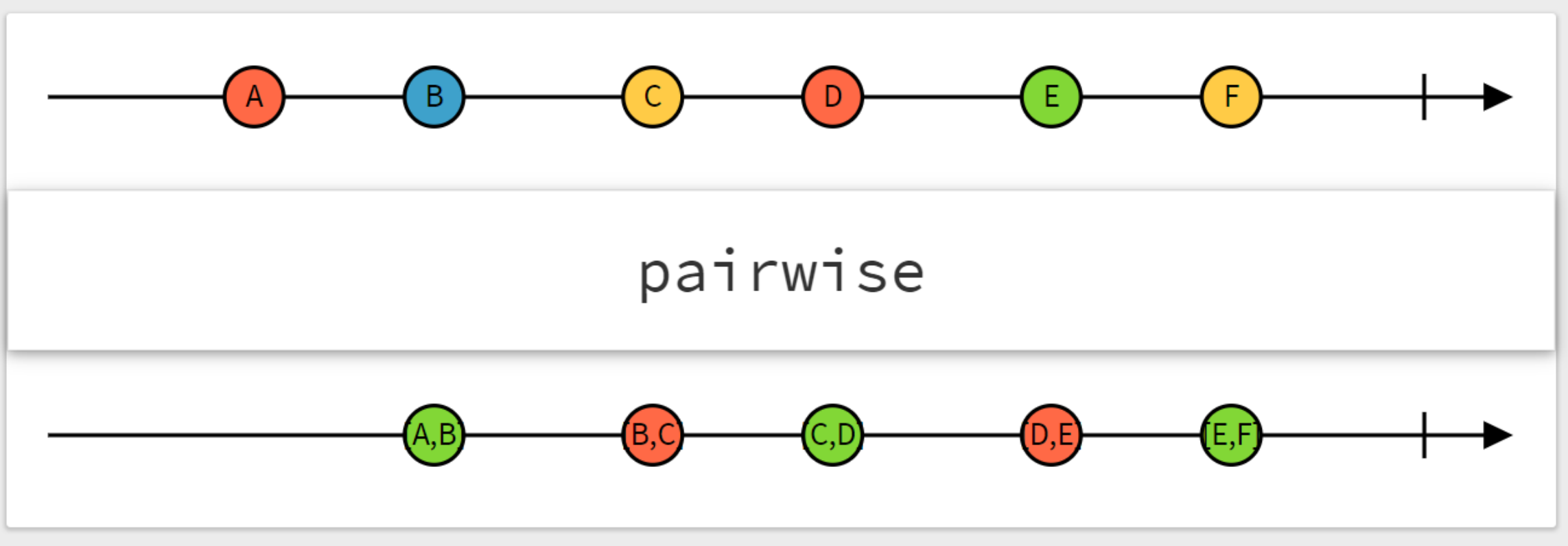
Operators

[https://rxjs.dev/guide/operators]



Operators



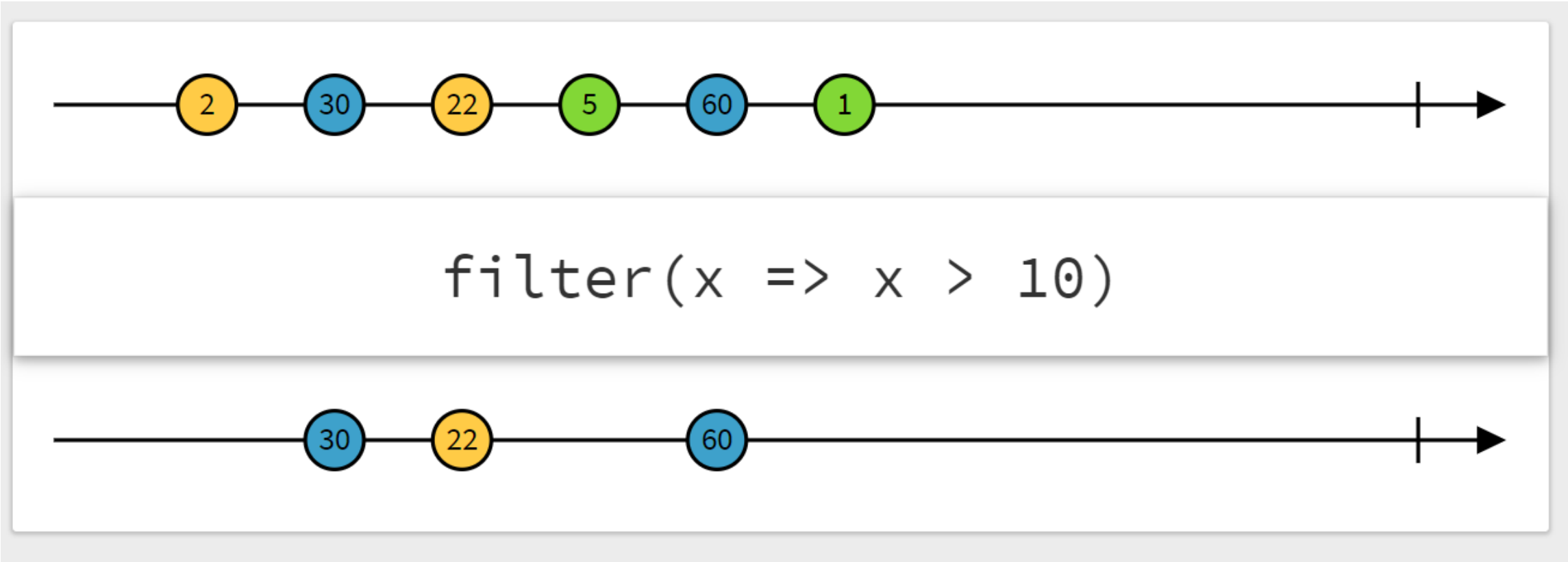


ANGULAR ARCHITECTS INSIDE KNOWLEDGE
SOFTWARE ARCHITECT

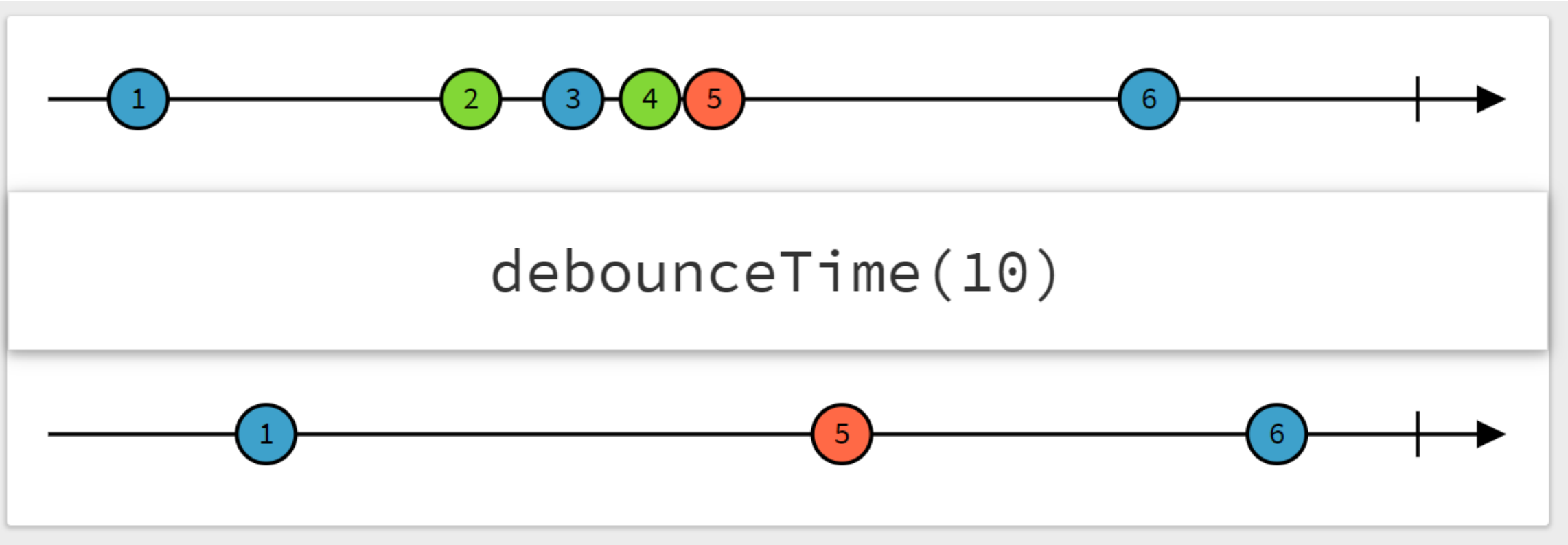
Filtering Operators

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

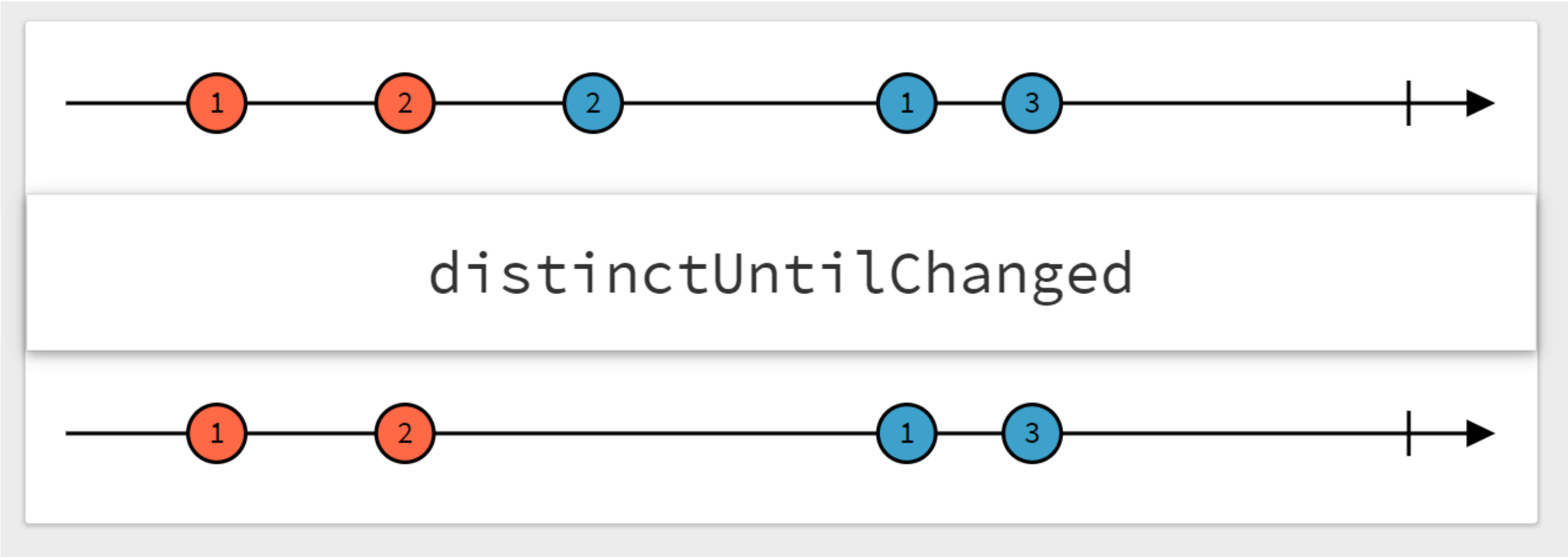
SOFTWARE
ARCHITECT



ANGULAR ARCHITECTS INSIDE KNOWLEDGE
SOFTWARE ARCHITECT



ANGULAR ARCHITECTS INSIDE KNOWLEDGE
SOFTWARE ARCHITECT



ANGULAR ARCHITECTS INSIDE KNOWLEDGE
SOFTWARE ARCHITECT

Demo

Simple Lookahead

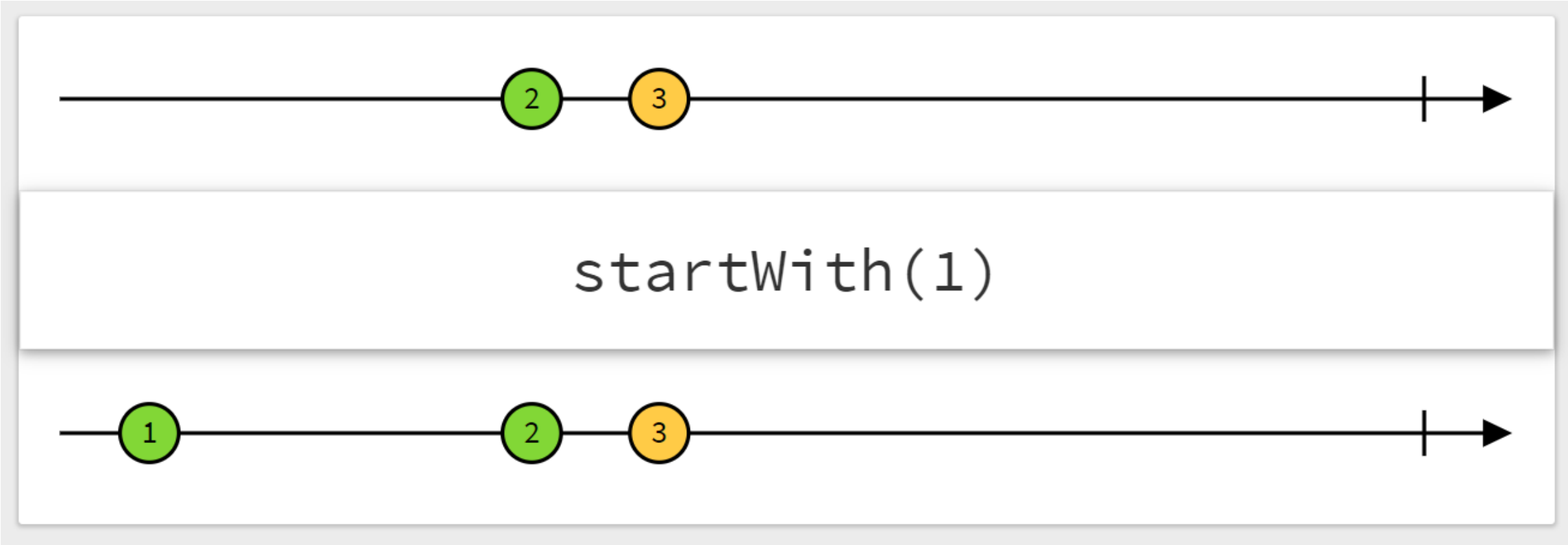
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

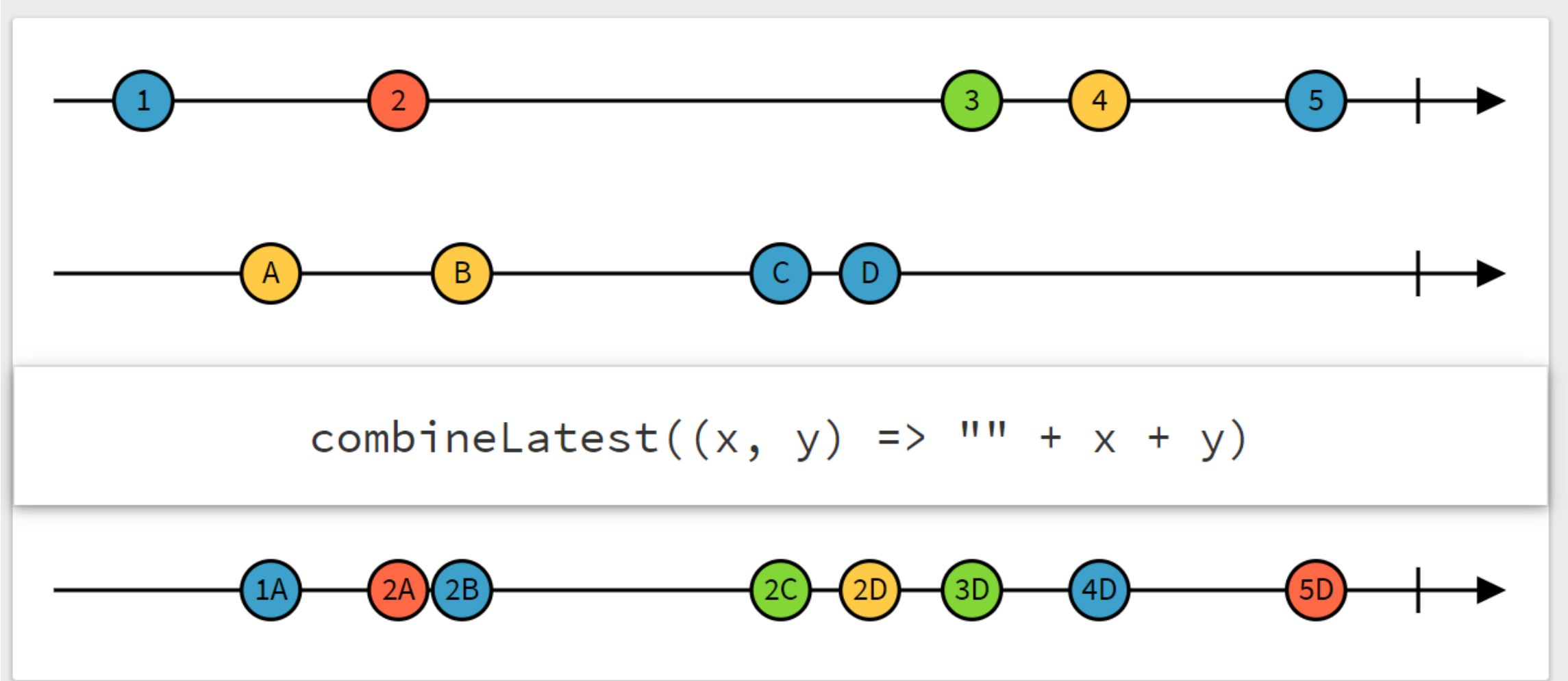
Combination Operators

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT



ANGULAR ARCHITECTS INSIDE KNOWLEDGE
SOFTWARE ARCHITECT

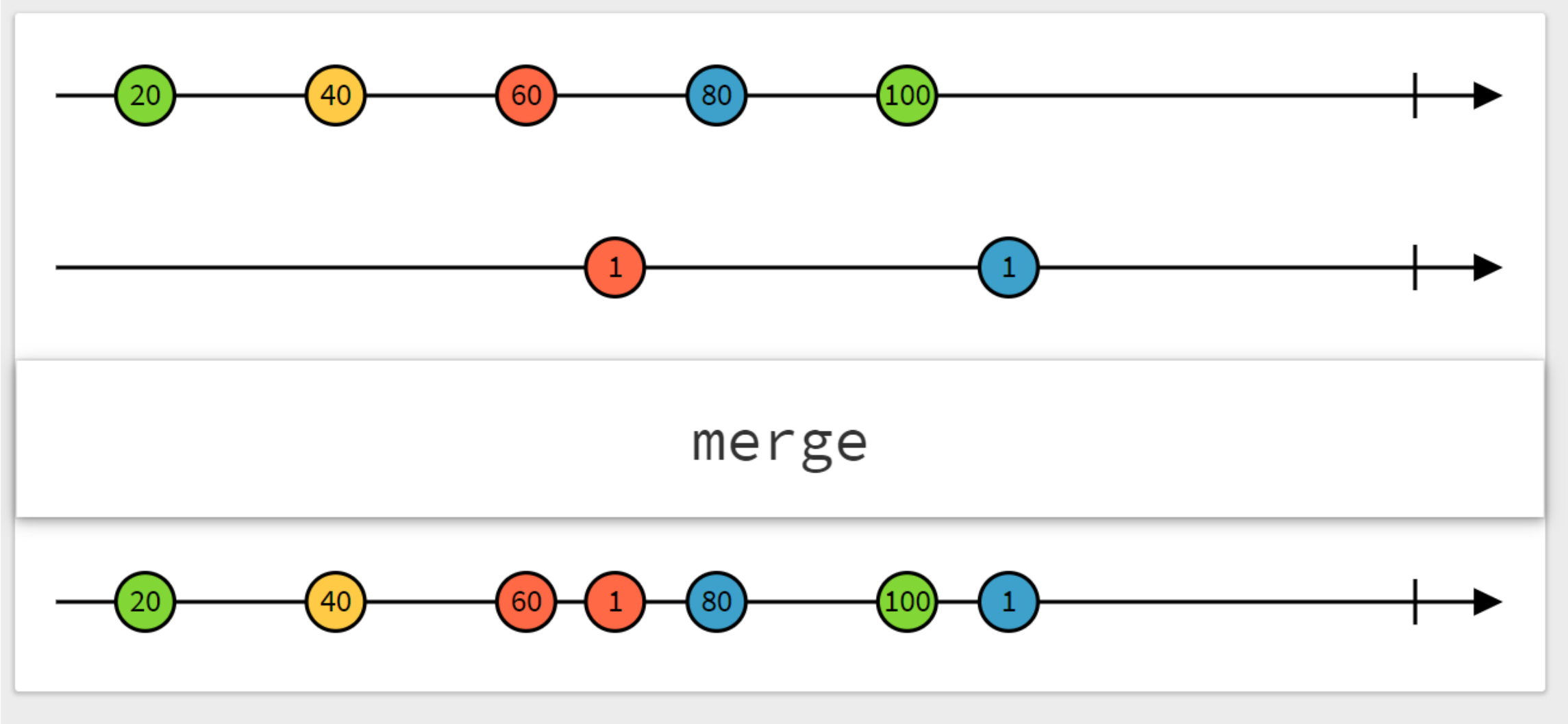


ANGULAR ARCHITECTS INSIDE KNOWLEDGE
SOFTWARE ARCHITECT

Digression: combineLatest vs forkJoin

combineLatest	forkJoin
Emits whenever all input $ have emitted at least once.	Emits when all input $ have been completed.
Multiple emits over time.	**Single** emit with last values.
Error will stop the emits.	Error will avoid any emit.
Unsubscribing necessary.	**Unsubscribing** recommended (as almost always).

ANGULAR ARCHITECTS INSIDE KNOWLEDGE
SOFTWARE ARCHITECT



ANGULAR ARCHITECTS INSIDE KNOWLEDGE
SOFTWARE ARCHITECT

Demo

Combine Streams

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE ARCHITECT

Error Handling

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Operators for Error Handling

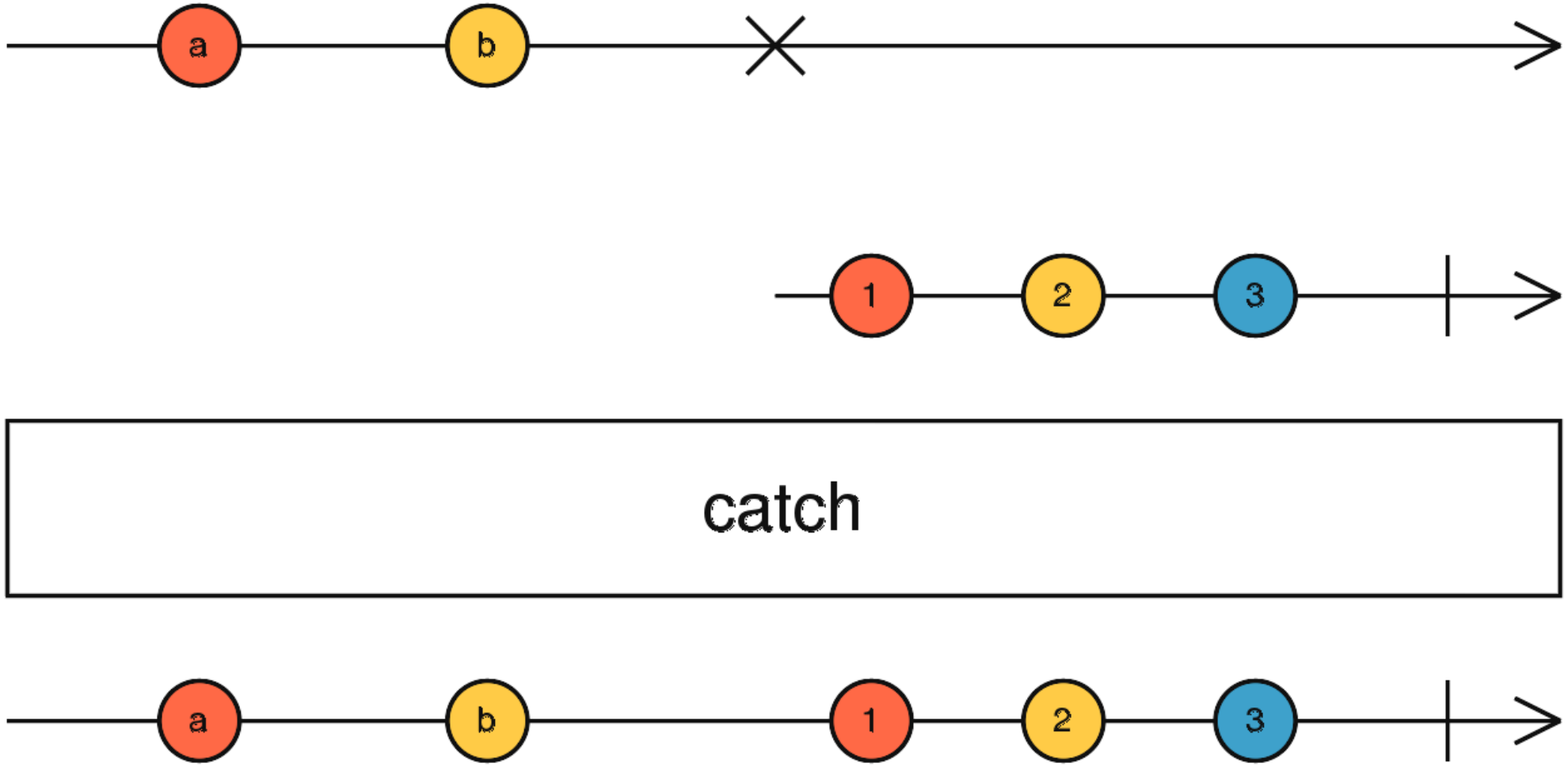
- ```
  catchError((err) => {
    console.log('Error caught:');
    console.log(err);
    return of([]);
  })
  ```

- ```
  retry(3)
  ```

- ```
  throwError(() => new Error('im an error'))
  ```

- ```
  finalize(() => {
    this.isLoading = false;
  })
  ```


ANGULAR ARCHITECTS INSIDE KNOWLEDGE
SOFTWARE ARCHITECT


a
b
1
2
3
catch
a
b
1
2
3

Demo

Error Handling

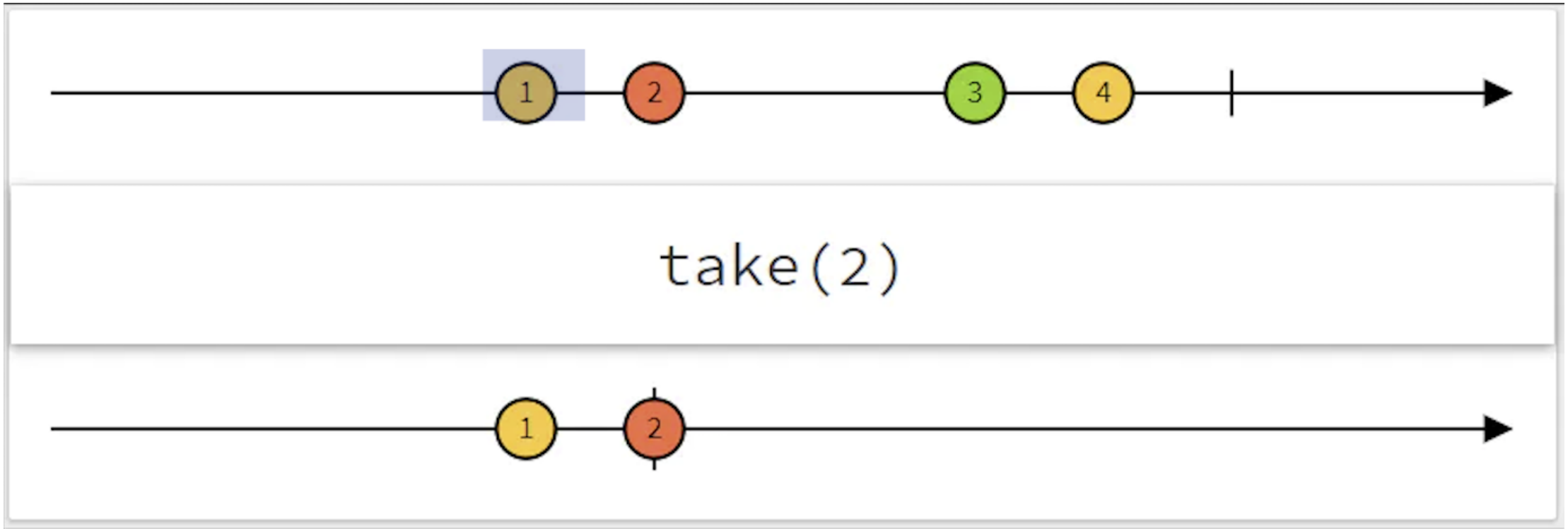
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Taking Operators

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT



ANGULAR ARCHITECTS INSIDE KNOWLEDGE
SOFTWARE ARCHITECT

Taking operators

- take(n)
- first(predicate?: boolean)
- takeUntil(observable) & takeWhile(boolean)

ANGULAR ARCHITECTS INSIDE KNOWLEDGE
SOFTWARE ARCHITECT

One more thing ☺

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

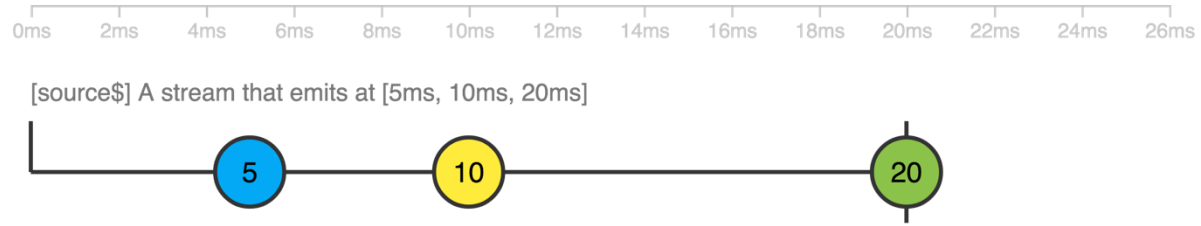
Higher Order Observables

Operators for Higher Order Observables

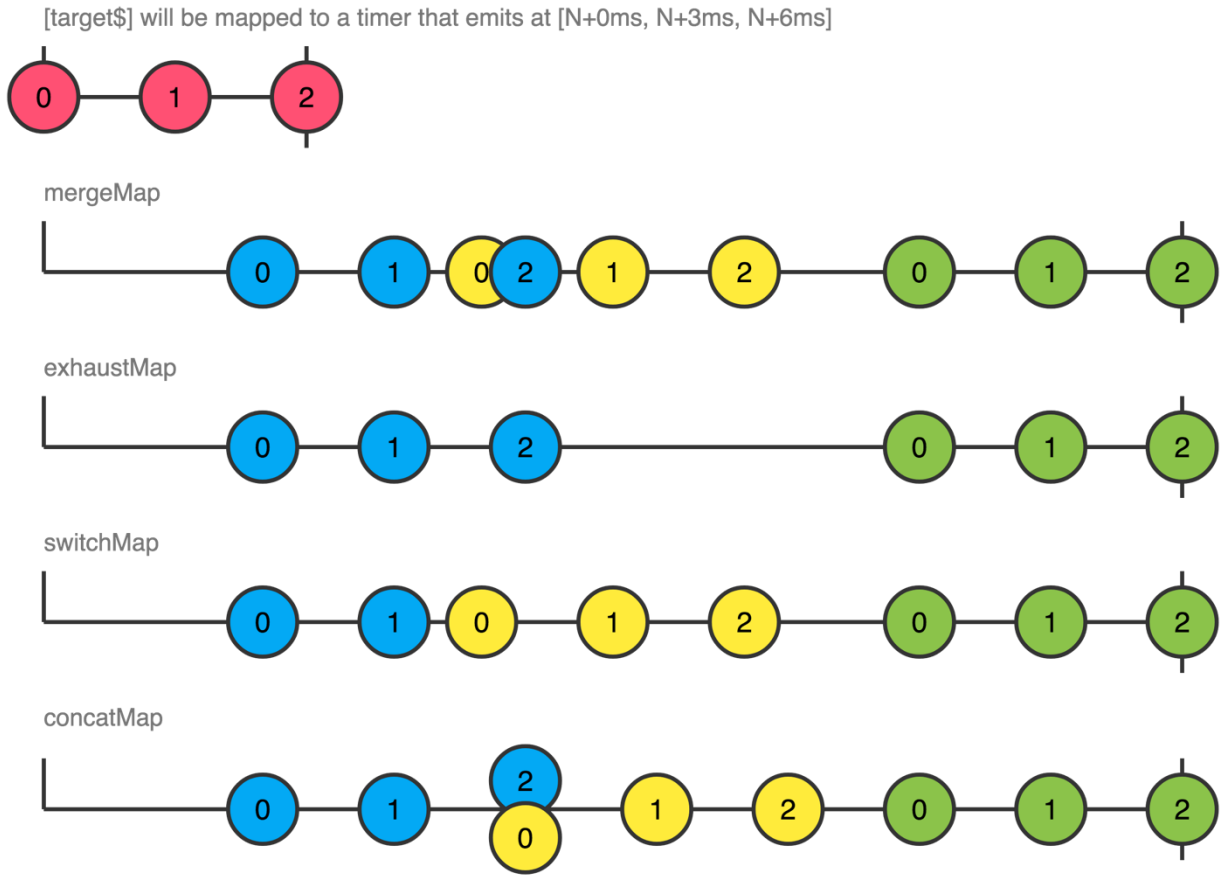
- mergeMap
  - merges outer (source) and inner observables
- exhaustMap
  - outer is ignored until inner is finished
- switchMap
  - inner will be completed after next outer
- concatMap
  - outer will be sent after inner is finished

ANGULAR ARCHITECTS INSIDE KNOWLEDGE SOFTWARE ARCHITECT

from, to



API



https://thinkrx.io/rxjs/mergeMap-vs-exhaustMap-vs-switchMap-vs-concatMap/

ANGULAR ARCHITECTS INSIDE KNOWLEDGE
SOFTWARE ARCHITECT

Lab Time

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Custom operators?

- Operators are functions with input and output of type observable
- We can create our own operator function
- To avoid code duplications (DRY as ever)
- There is an example at the end of the lab

ANGULAR ARCHITECTS INSIDE KNOWLEDGE SOFTWARE ARCHITECT

Recap

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Like RxJS?

- Marble Diagrams
  - http://rxmarbles.com

- Usefull Links
  - https://rxjs.dev/guide/overview
    - https://reactive.how/rxjs/ (Launchpad)
    - https://rxjs-dev.firebaseapp.com/operator-decision-tree (ODT)
  - https://www.learnrxjs.io/
  - https://angular.io/guide/rx-library

ANGULAR ARCHITECTS INSIDE KNOWLEDGE SOFTWARE ARCHITECT