



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



Motivation



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

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



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(flightDateStr => new Date(flightDateStr)))
  .subscribe({
    next: (bookings) => { ... },
    error: (err) => { console.error(err); },
    complete: () => { console.log('complete'); }
  });
```



Transformation Operators



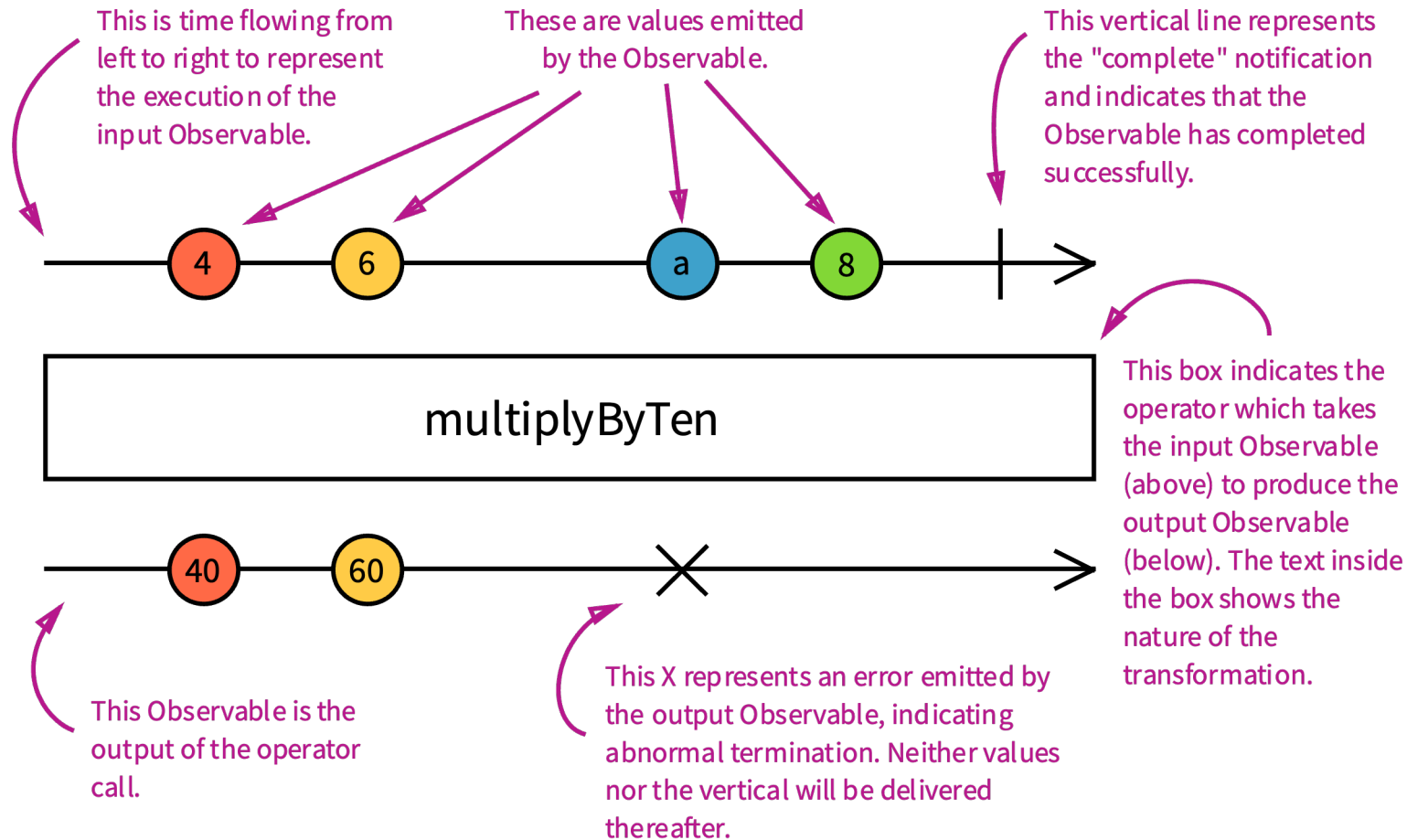
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Operators

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



Operators



`map(x => 10 * x)`



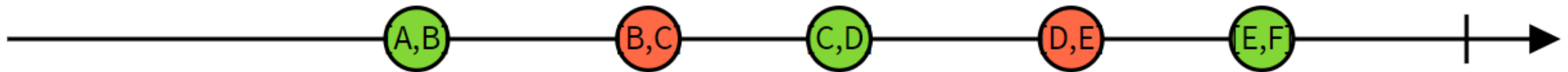


`pluck("a")`





pairwise



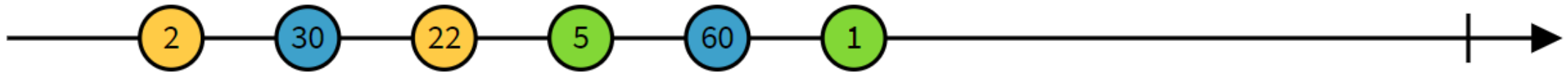
Filtering Operators



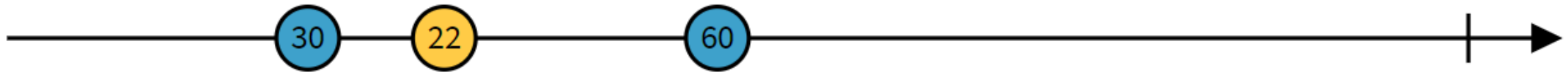
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

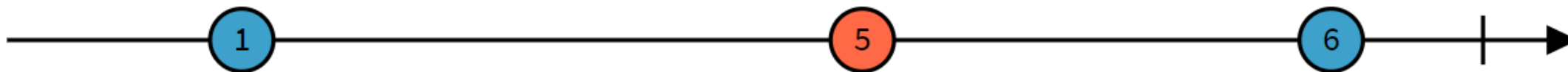


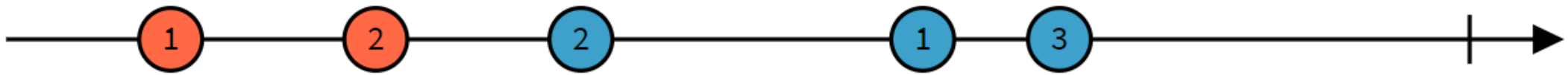
```
filter(x => x > 10)
```



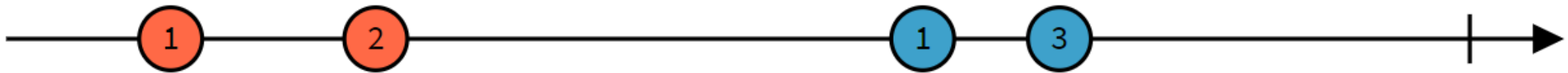


`debounceTime(10)`





`distinctUntilChanged`



Demo

Simple Lookahead



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

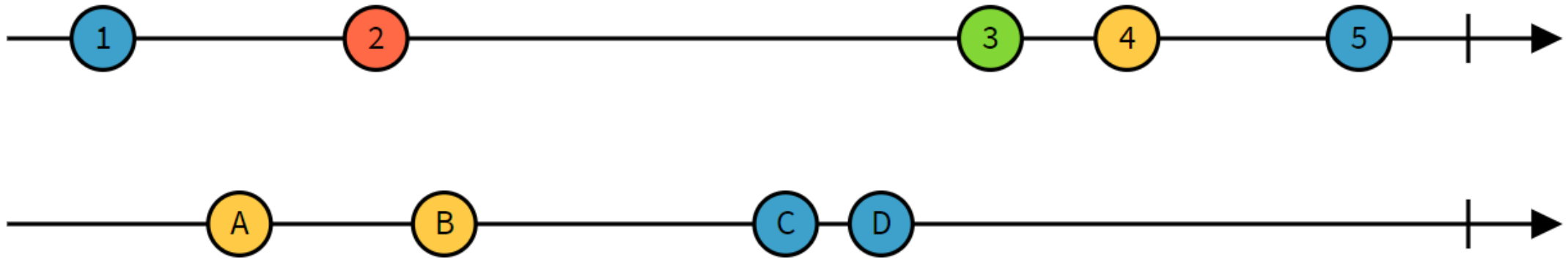
Combination Operators



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

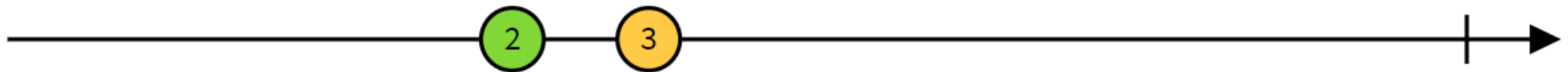


SOFTWARE
ARCHITECT

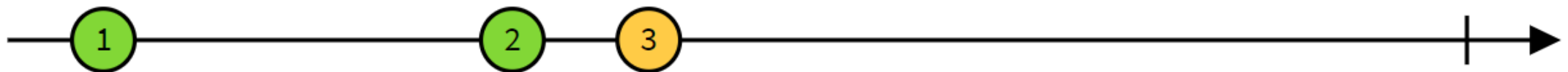


```
combineLatest((x, y) => "" + x + y)
```





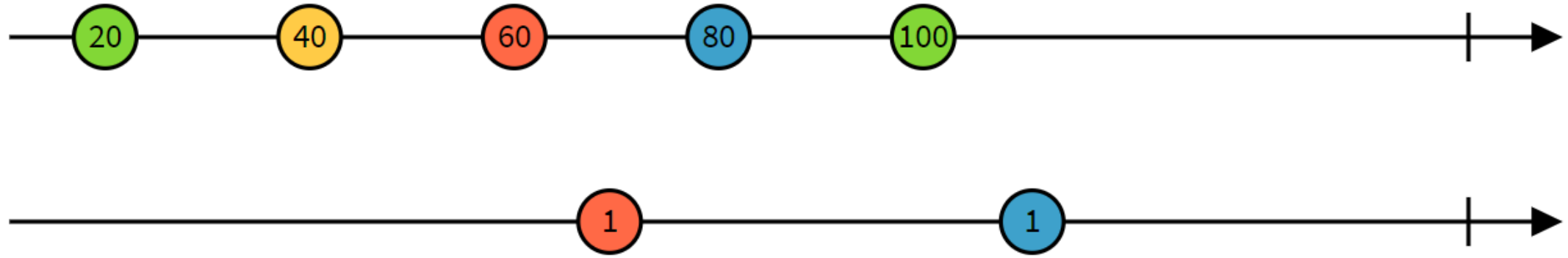
`startWith(1)`



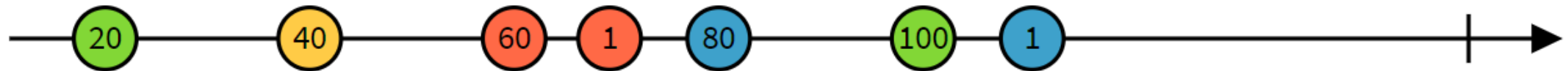
Digression: combineLatest vs forkJoin

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).





merge



Demo

Combine Streams



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Error Handling



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

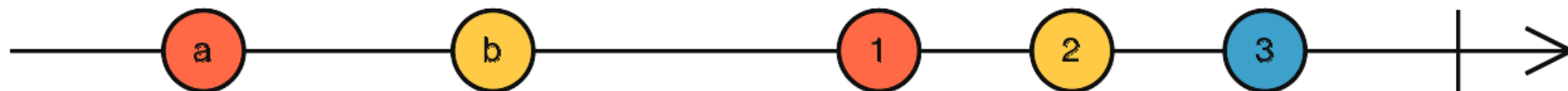
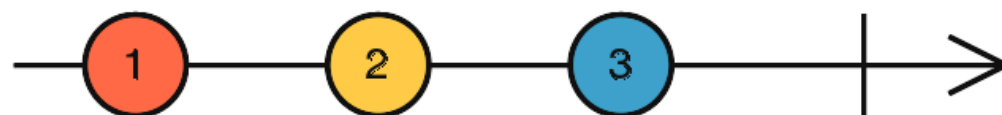


SOFTWARE
ARCHITECT

Operators for Error Handling

- catchError
- retry
- retryWhen
- throwError
- finalize





Demo

Error Handling



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

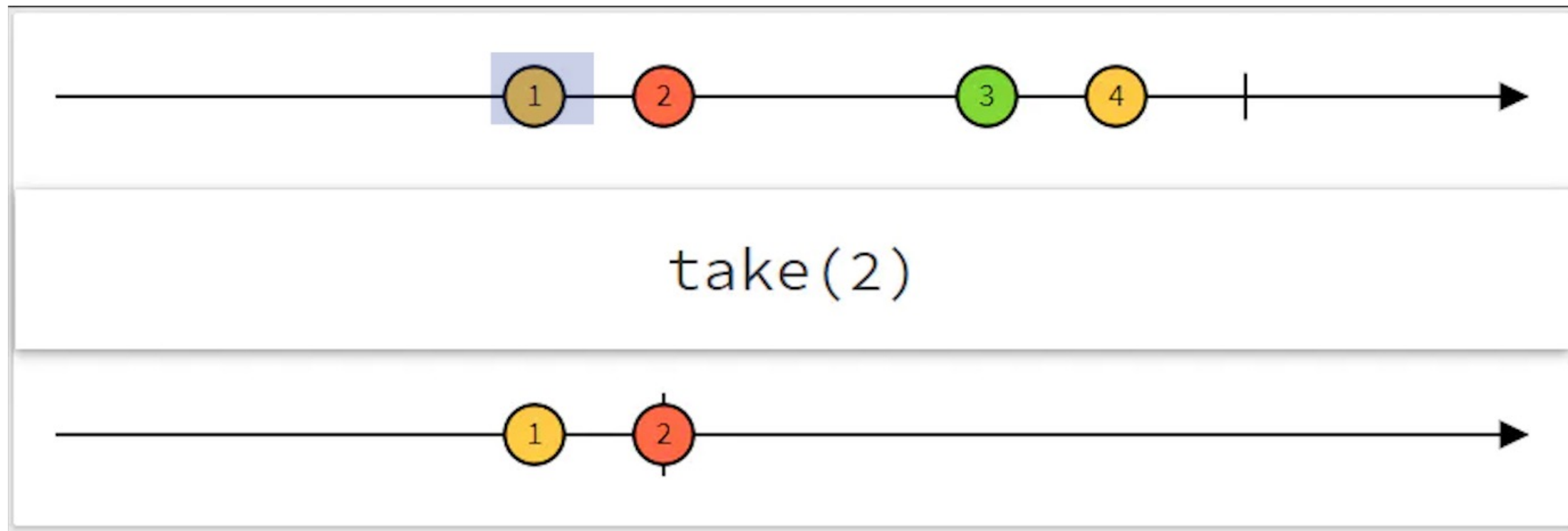
Taking Operators



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT



Taking operators

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



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

Lab Time



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

One more thing 😊



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Higher Order Observables



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



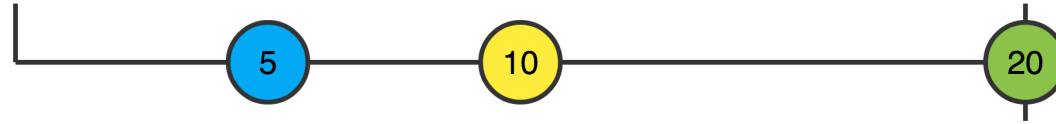
SOFTWARE
ARCHITECT

Operators for Higher Order Observables

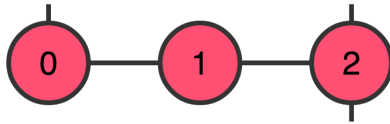
- mergeMap
 - merges outer (source) and inner observables
- exhaustMap
 - outer is ignored until inner is finished
- switchMap
 - inner will be completed after next outer
- concatMap
 - outer will be sent after inner is finished



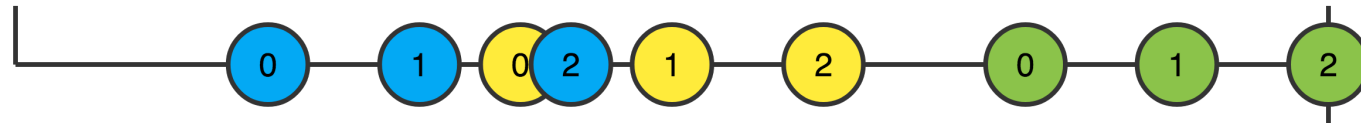
[source\$] A stream that emits at [5ms, 10ms, 20ms]



[target\$] will be mapped to a timer that emits at [N+0ms, N+3ms, N+6ms]



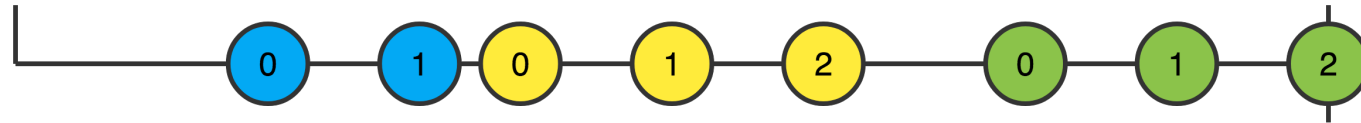
mergeMap



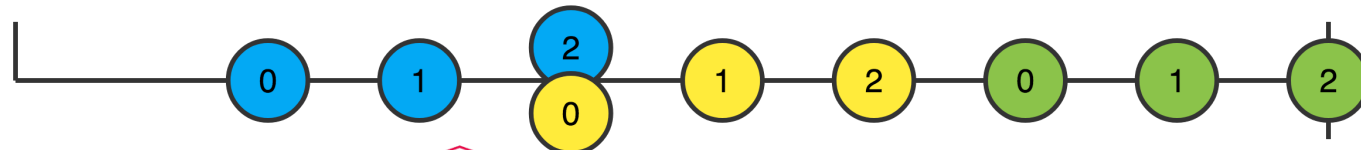
exhaustMap



switchMap



concatMap



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.firebaseio.com/operator-decision-tree> (ODT)
 - <https://www.learnrxjs.io/>
 - <https://angular.io/guide/rx-library>

