



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



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

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“



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Observer

Option with more than one
parameter is now deprecated!

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

← **Observer**



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Observer

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



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Observer

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



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Deprecated!

Example

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



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

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); }
  });
```

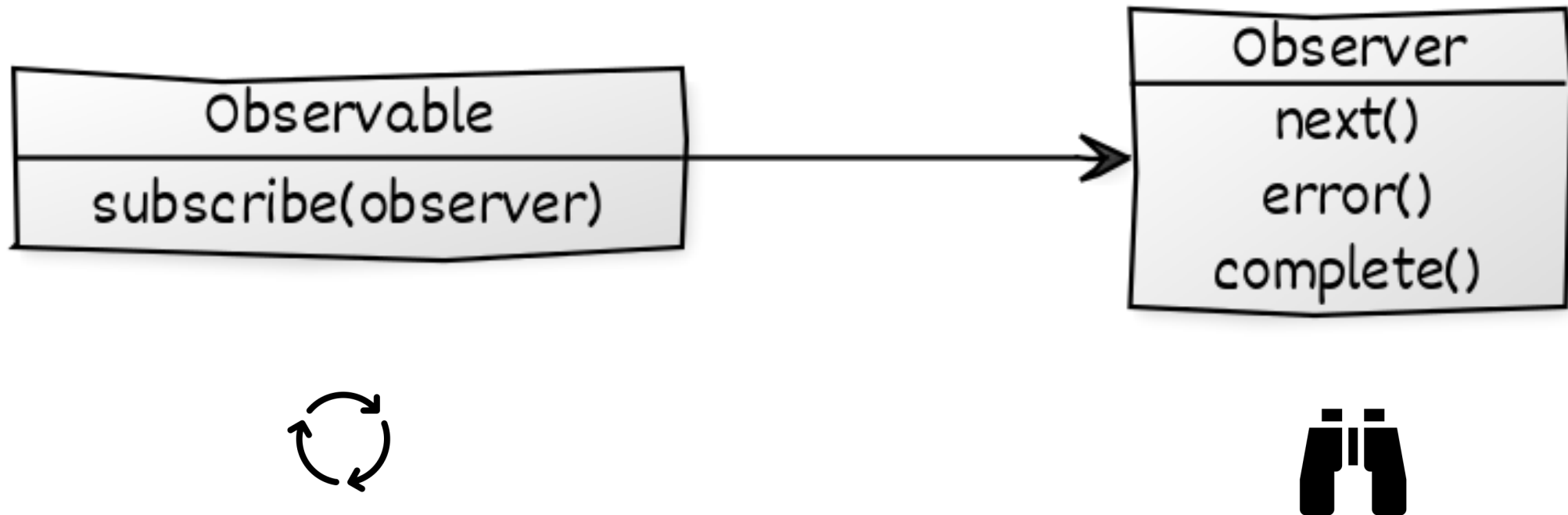


ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

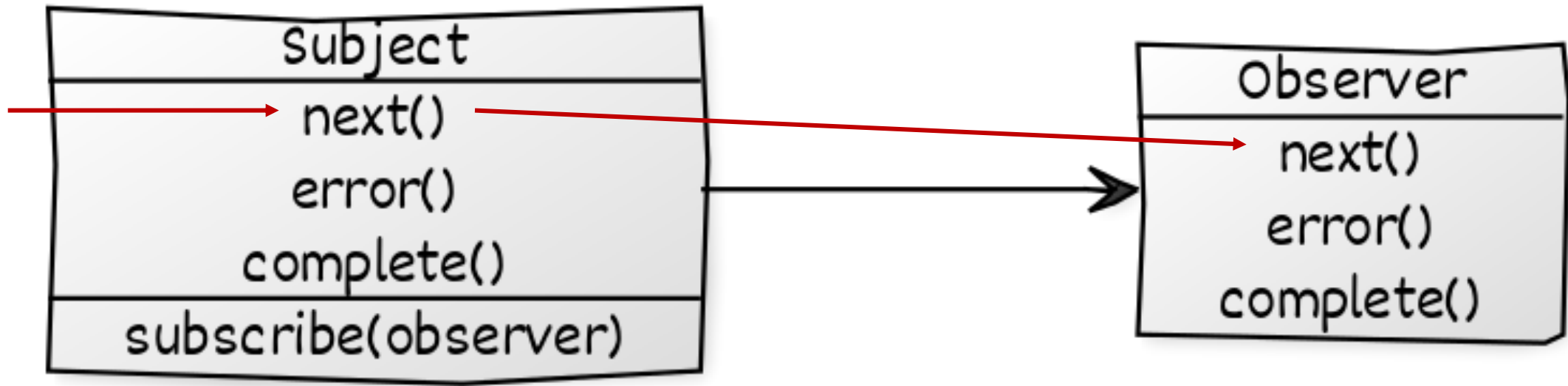


SOFTWARE
ARCHITECT

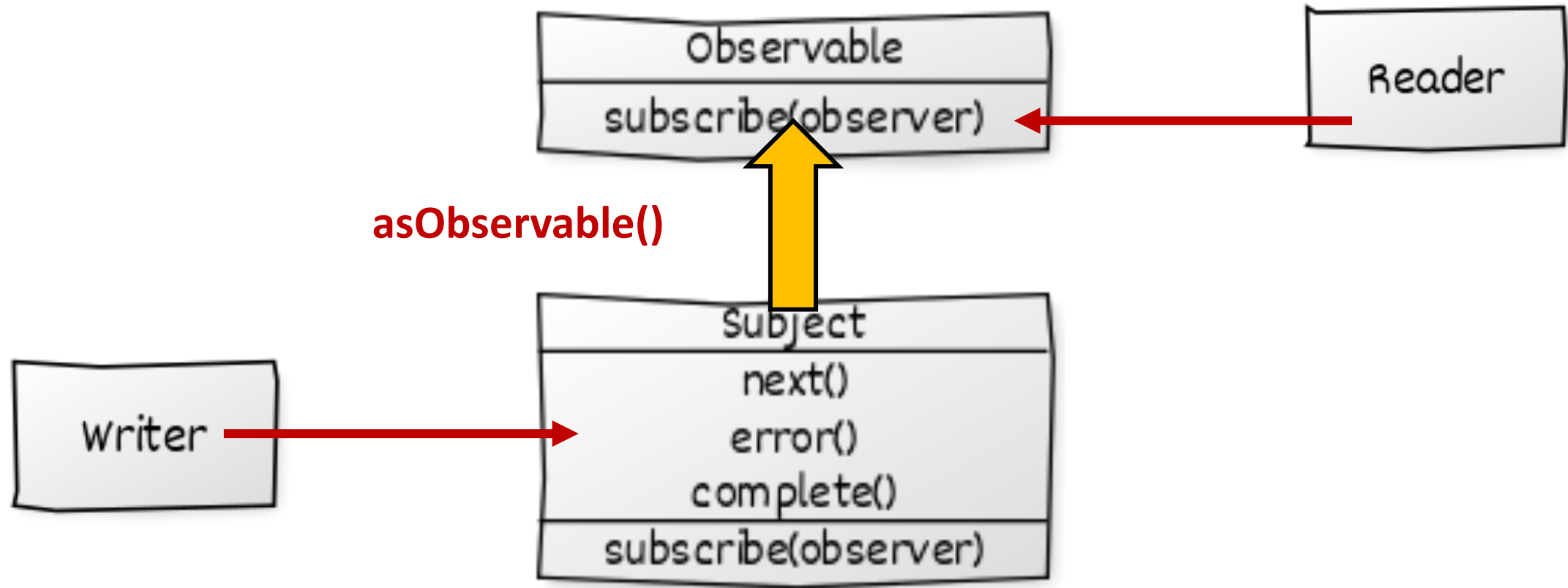
Observable und Observer



Subjects: Special Observables



Convert Subject into Observable



asObservable

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

```
[...]  
this.observable.subscribe(...)
```

```
[...]  
this.subject.next(...)
```



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Why Observables?

Asynchronous
operations

Interactive
(reactive)
behavior



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Creating Observables



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

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();
```



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Creation Operators (Factories)

[<https://www.learnrxjs.io>]

fromEvent

of

throwError

interval

timer



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Cold vs. Hot Observables

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(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



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Transformation Operators



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Operators

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



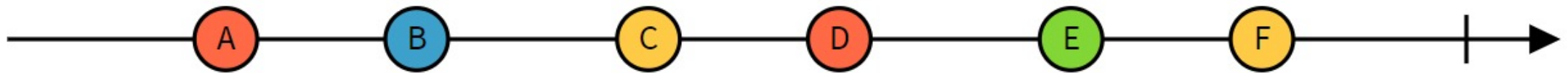
`map(x => 10 * x)`





`pluck("a")`





pairwise



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

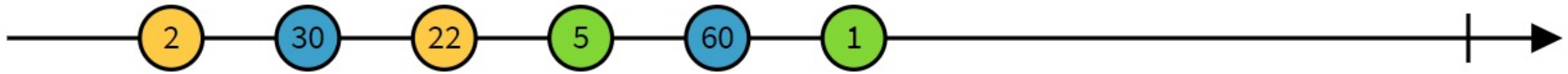
Filtering Operators



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT



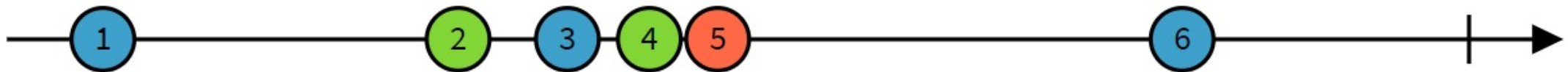
```
filter(x => x > 10)
```



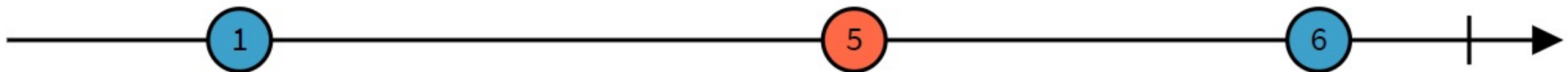
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT



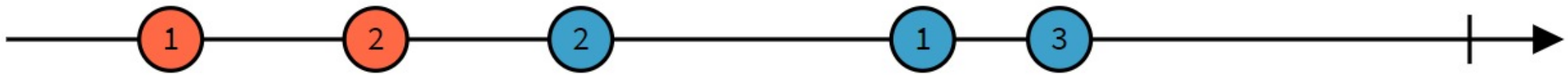
`debounceTime(10)`



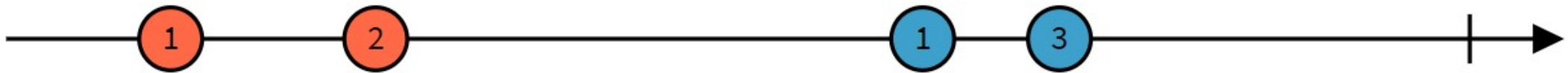
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT



`distinctUntilChanged`



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Lab/Demo

Simple Lookahead



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

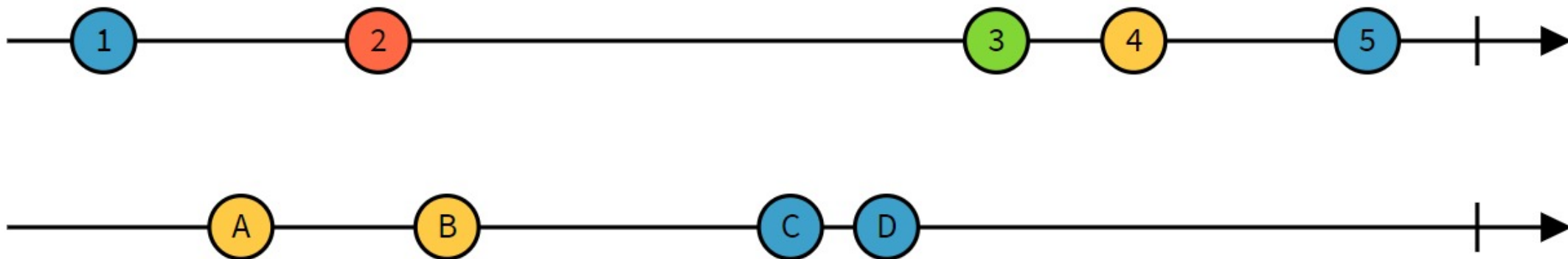
Combination Operators



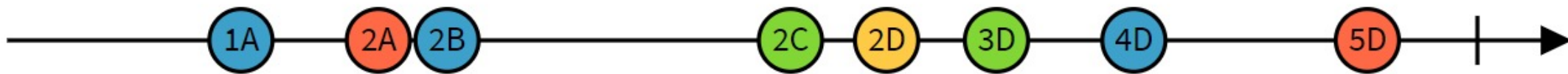
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT



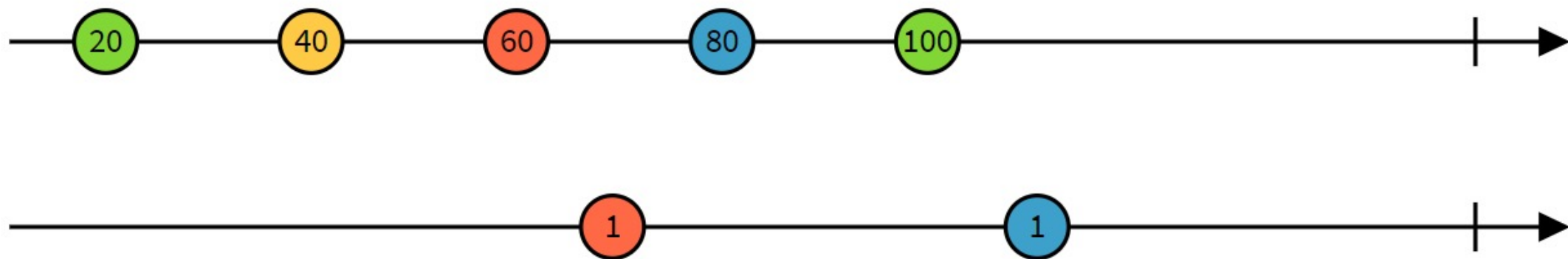
```
combineLatest((x, y) => "" + x + y)
```



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT



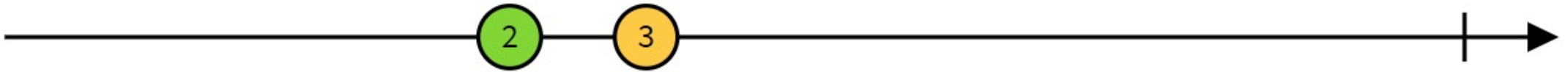
merge



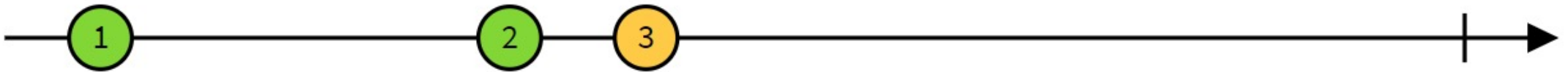
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT



`startWith(1)`



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Lab/Demo

Combine Streams



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Higher Order Observables



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



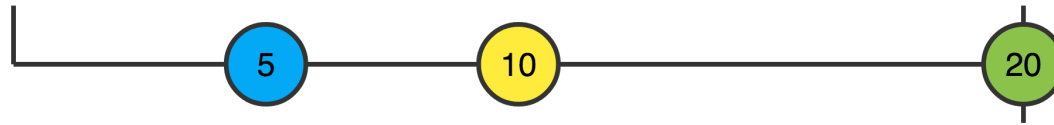
SOFTWARE
ARCHITECT

Operators for Higher Order Observables

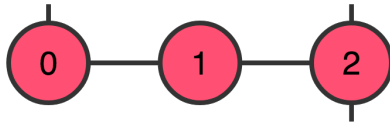
- mergeMap
 - merges outer (source) and inner observables
- exhaustMap
 - outer is ignored until inner is finished
- switchMap
 - inner will be completed after next outer
- concatMap
 - outer will be sent after inner is finished



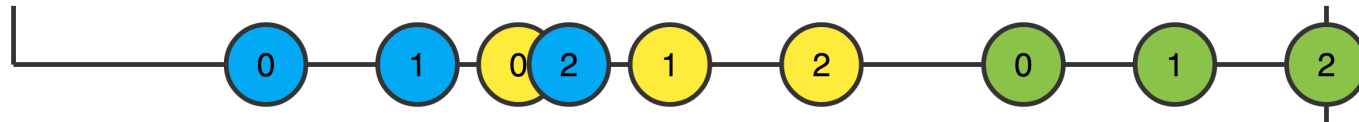
[source\$] A stream that emits at [5ms, 10ms, 20ms]



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



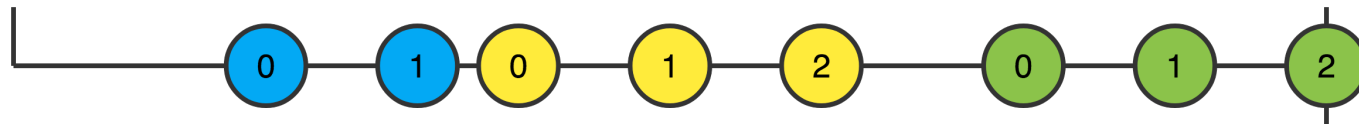
mergeMap



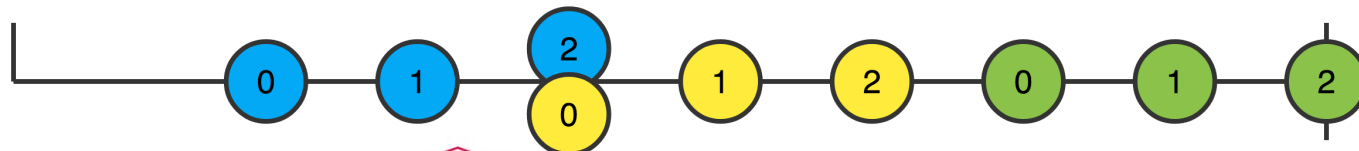
exhaustMap



switchMap



concatMap



Error Handling



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Operators for Error Handling

- catchError
- retry
- retryWhen

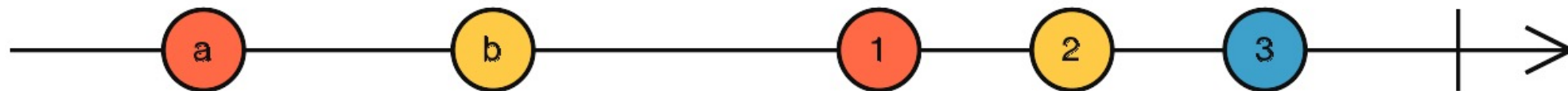
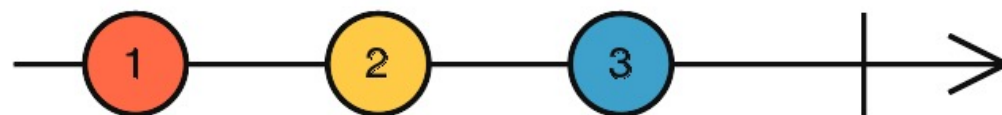
- throwError



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT



Lab/Demo

Error Handling



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Subjects

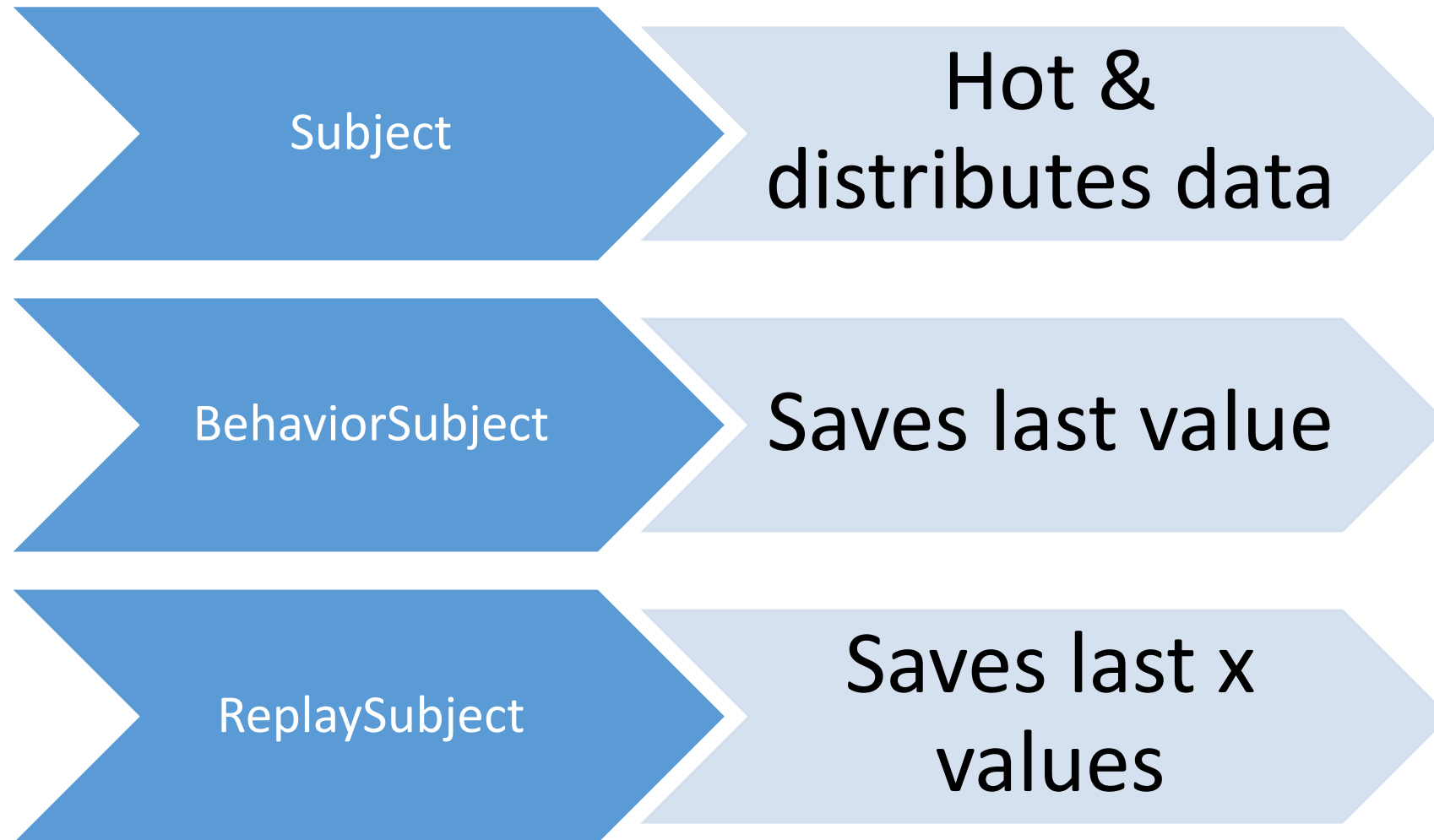


ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Subjects



Closing Observables



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

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



SOFTWARE
ARCHITECT

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>



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT