



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Reactive Extensions for JavaScript

ANGULARarchitects.io

Contents

- Overview to Observables
- Generating Observables
- Hot vs. Cold Observables
- Piping operators (lookahead)
- Subjects (Pub / Sub)
- Closing Observables



Overview



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

What are observables?

- Represents (asynchronous) data that is published over time



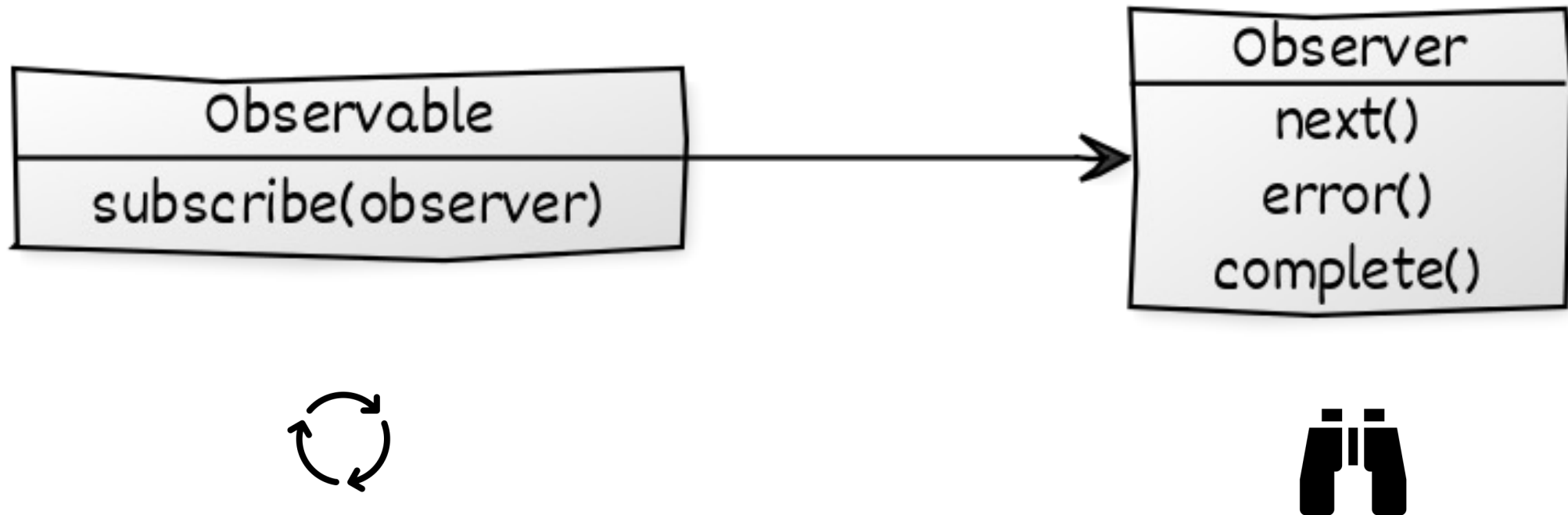
**Observable
„Source“**

**Operator
(z. B. map)**

**Observer
„Destination“**

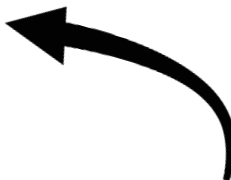


Observable und Observer



Observer

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

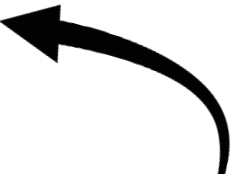


Observer



Observer

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



Observer



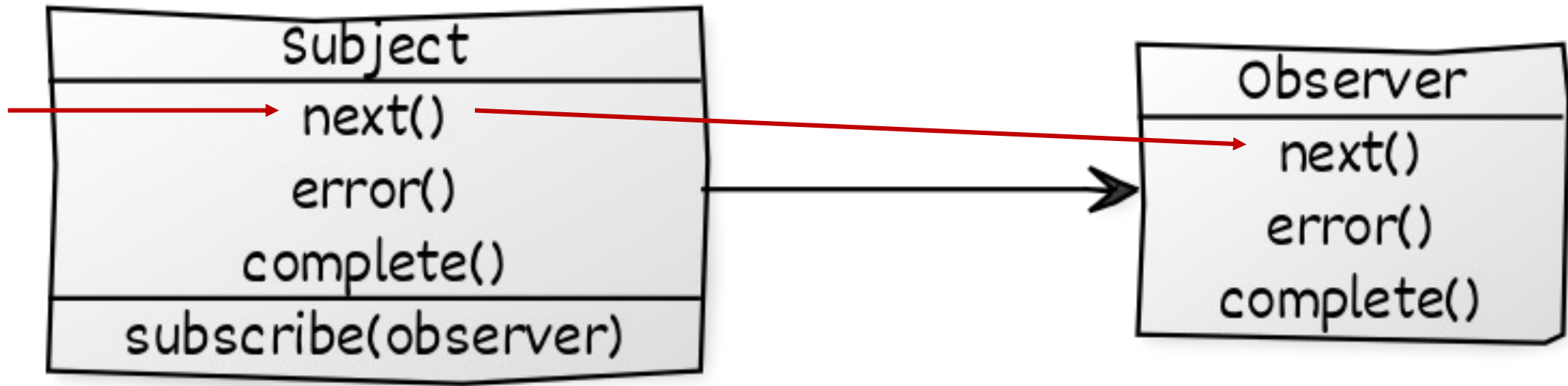
Example with Pipeable Operators

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

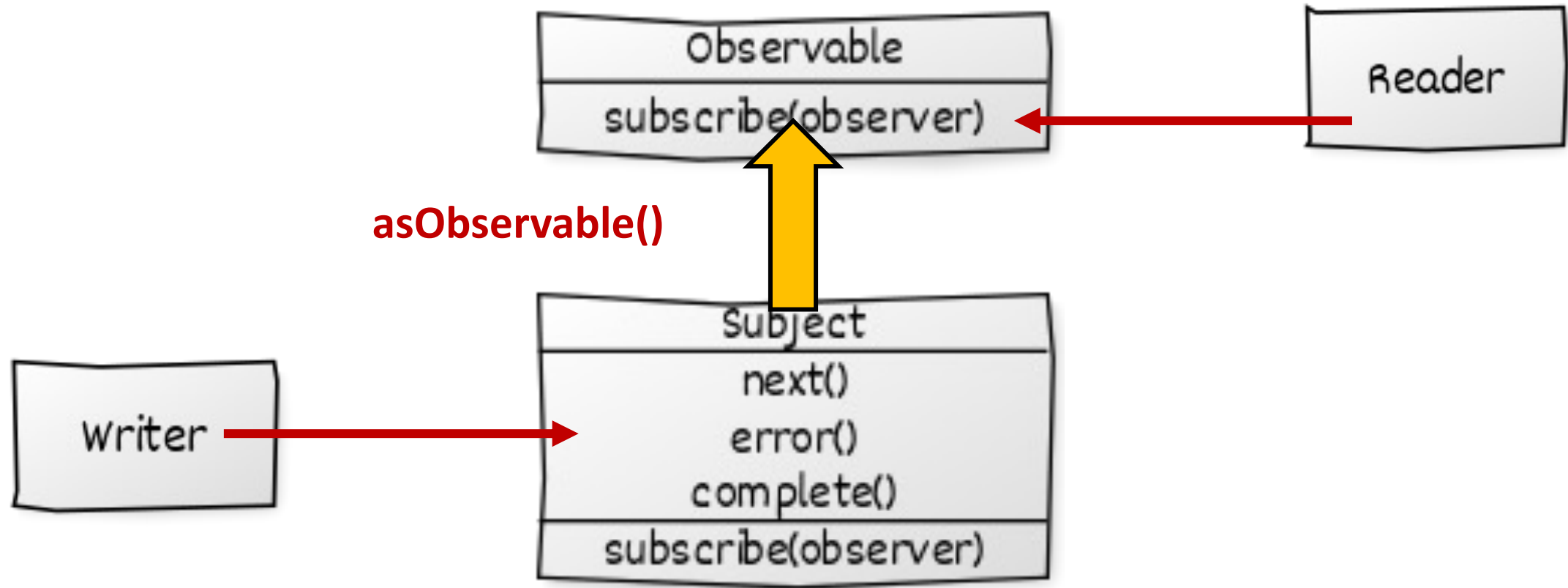
this
  .http
  .get("http://www.angular.at/api/...")
  .pipe(map(flightDateStr => new Date(flightDateStr)))
  .subscribe({
    next: (bookings) => { ... },
    error: (err) => { console.error(err); }
  });
```



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



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Creating an Observable

```
let observable = new Observable((observer) => {  
  observer.next(4711);  
  observer.next(815);  
  // observer.error("err!");  
  observer.complete();  
  return () => { console.debug('Bye bye'); };  
});
```

Sync & Async, Event-driven

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

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



Creation Operators (Factories)

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

fromEvent

of

throwError

interval

timer



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Cold vs. Hot Observables



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Cold vs. Hot Observables

Cold

- Point to point
- Lazy: Only starts with subscription

Hot

- Multicast
- Eager: Sender starts without subscriptions

Default



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({ 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>]



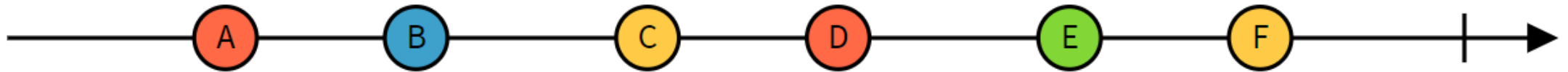
`map(x => 10 * x)`





`pluck("a")`





pairwise

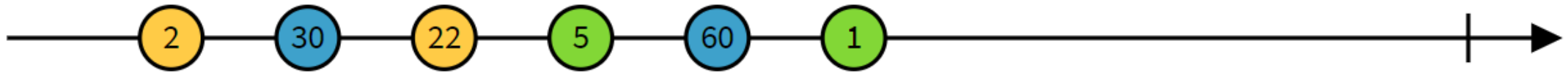


ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

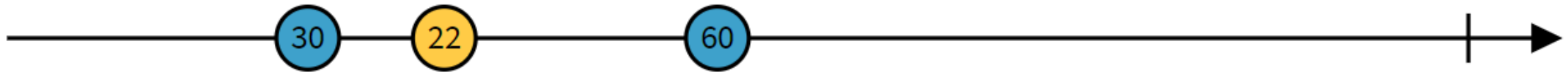
Filtering Operators

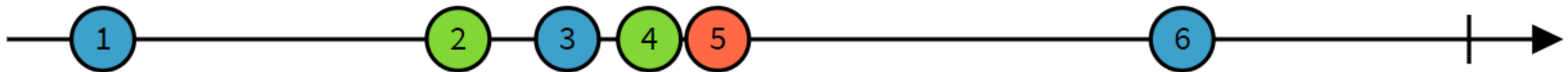


ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

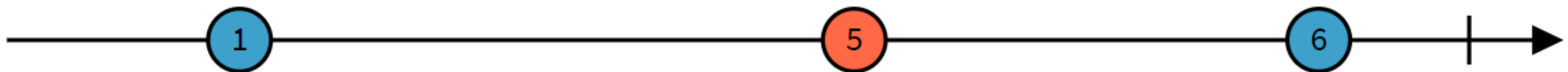


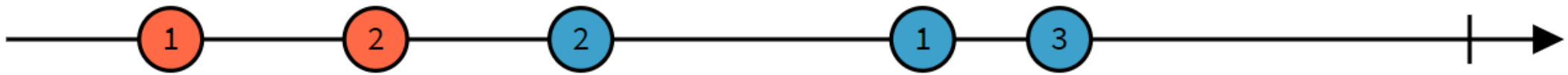
```
filter(x => x > 10)
```



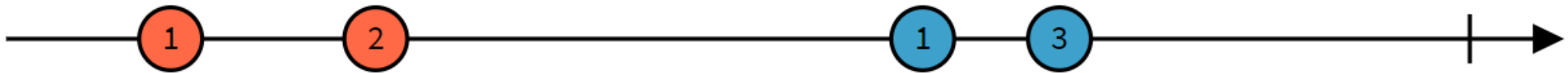


`debounceTime(10)`





`distinctUntilChanged`



DEMO: Lookahead



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

LAB

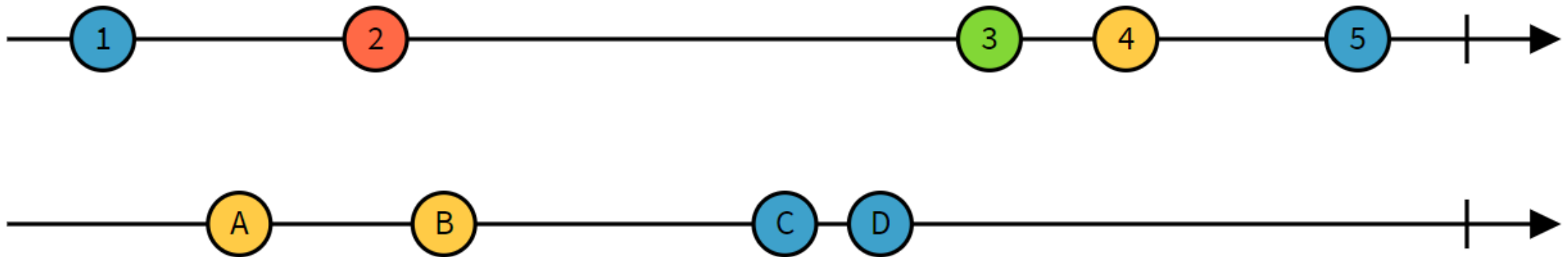


ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Combination Operators

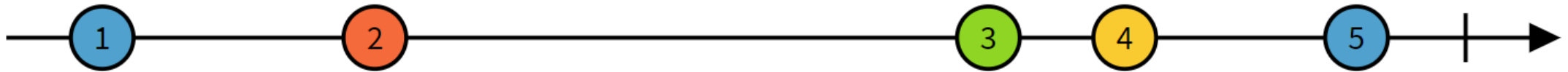


ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

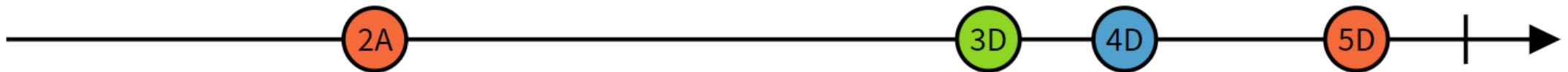


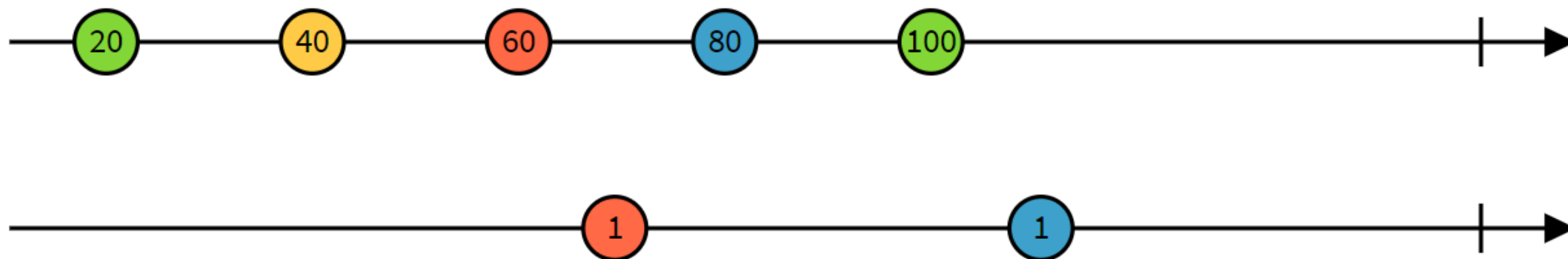
```
combineLatest((x, y) => "" + x + y)
```



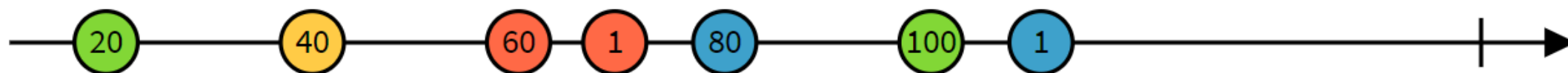


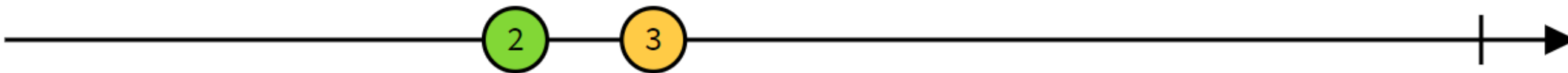
```
withLatestFrom((x, y) => "" + x + y)
```



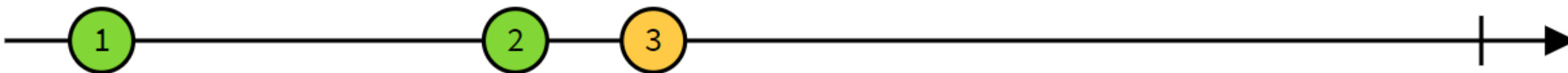


merge





`startWith(1)`



DEMO



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Labs



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Higher Order Observables



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Operators for Higher Order Observables

- switchMap
- mergeMap
- concatMap
- exhaustMap



Error Handling



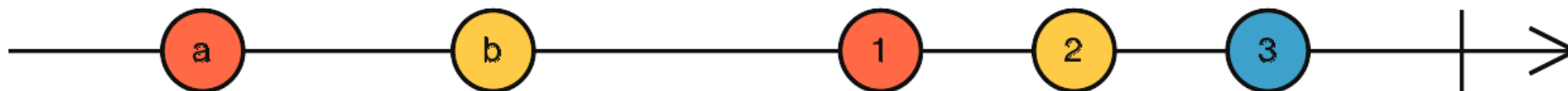
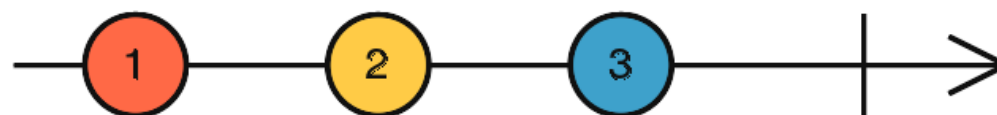
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Operators for Error Handling

- catchError
- retry
- retryWhen

- throwError



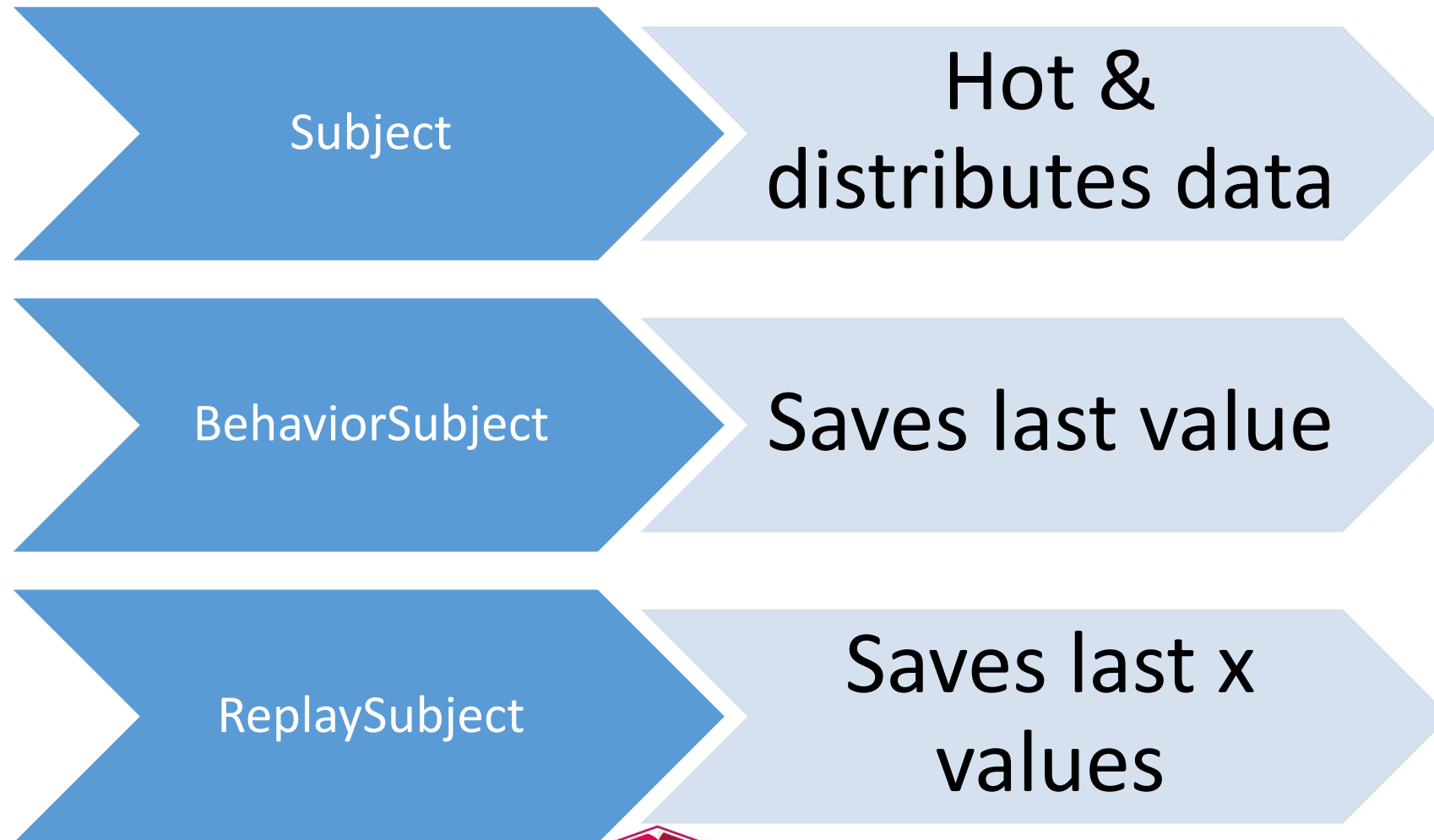


Subjects



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Subjects



DEMO: Pub/Sub with Subjects



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

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

