Observable & Rxjs

Fravezzi Mattia
m.fravezzi@almaviva.it

AlmavivA

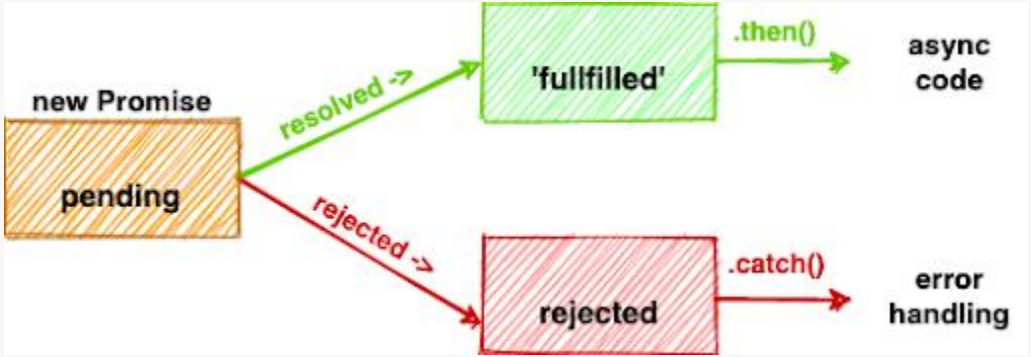
Callback

```
getUsers(function (err, users) {
    if (!!err) {
        ...
    } else {
        ...
    }
})
```

Callback Hell

```
function hell(win) {
 // for listener purpose
 return function() {
   loadLink(win, REMOTE_SRC+'/assets/css/style.css', function() {
     loadLink(win, REMOTE_SRC+'/lib/async.js', function() {
       loadLink(win, REMOTE_SRC+'/lib/easyXDM.js', function() {
         loadLink(win, REMOTE_SRC+'/lib/json2.js', function() {
           loadLink(win, REMOTE_SRC+'/lib/underscode.min.js', function() {
             loadLink(win, REMOTE_SRC+'/lib/backbone.min.js', function() {
               loadLink(win, REMOTE_SRC+'/dev/base_dev.js', function() {
                 loadLink(win, REMOTE_SRC+'/assets/js/deps.js', function() {
                   loadLink(win, REMOTE_SRC+'/src/' + win.loader_path + '/loader.js', function() {
                     async.eachSeries(SCRIPTS, function(src, callback) {
                       loadScript(win, BASE_URL+src, callback);
                     });
                   });
                 });
               });
             });
           });
         });
       });
     });
   });
 };
}
```

Promises


new Promise
pending
resolved ->
'fullfilled'
.then()
async
code
rejected ->
rejected
.catch()
error
handling

Promises - Hell

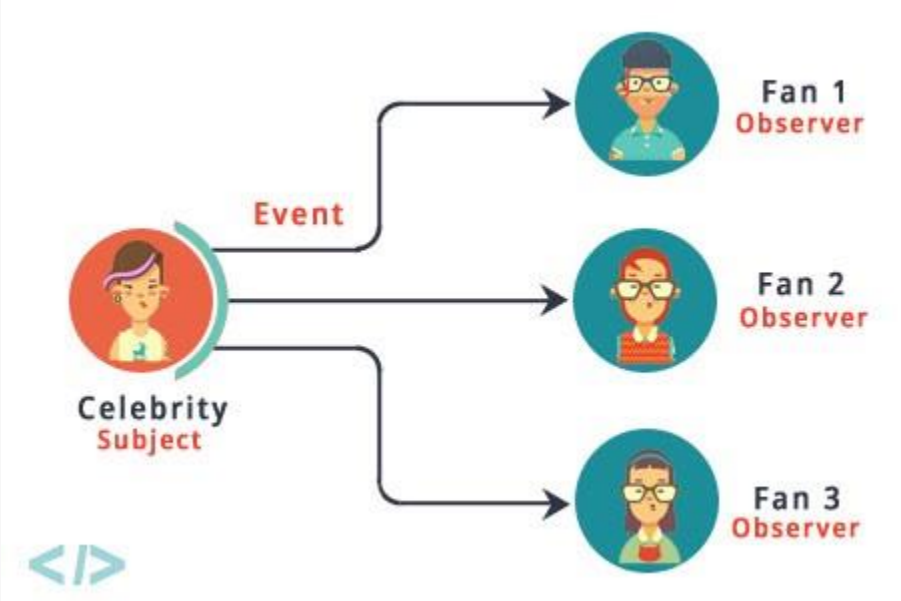
```
getUserRole() {
        return connectToDatabase()
            .then(db => {
                return getUsers(db)
                    .then(users => {
                        return getUsersRoles(db, users)
                            .then(usersRoles => {
                                return usersRoles;
                            })
                    })
            })
    }
```


CALLBACKS
PROMISES
ASYNC/AWAIT
imgflip.com

Promises - How to solve?

```
getUserRole() {
      const db = await connectToDatabase();
      const users = await getUsers(db);
      return await getUsersRoles(db, users);
      }
```

Observer pattern

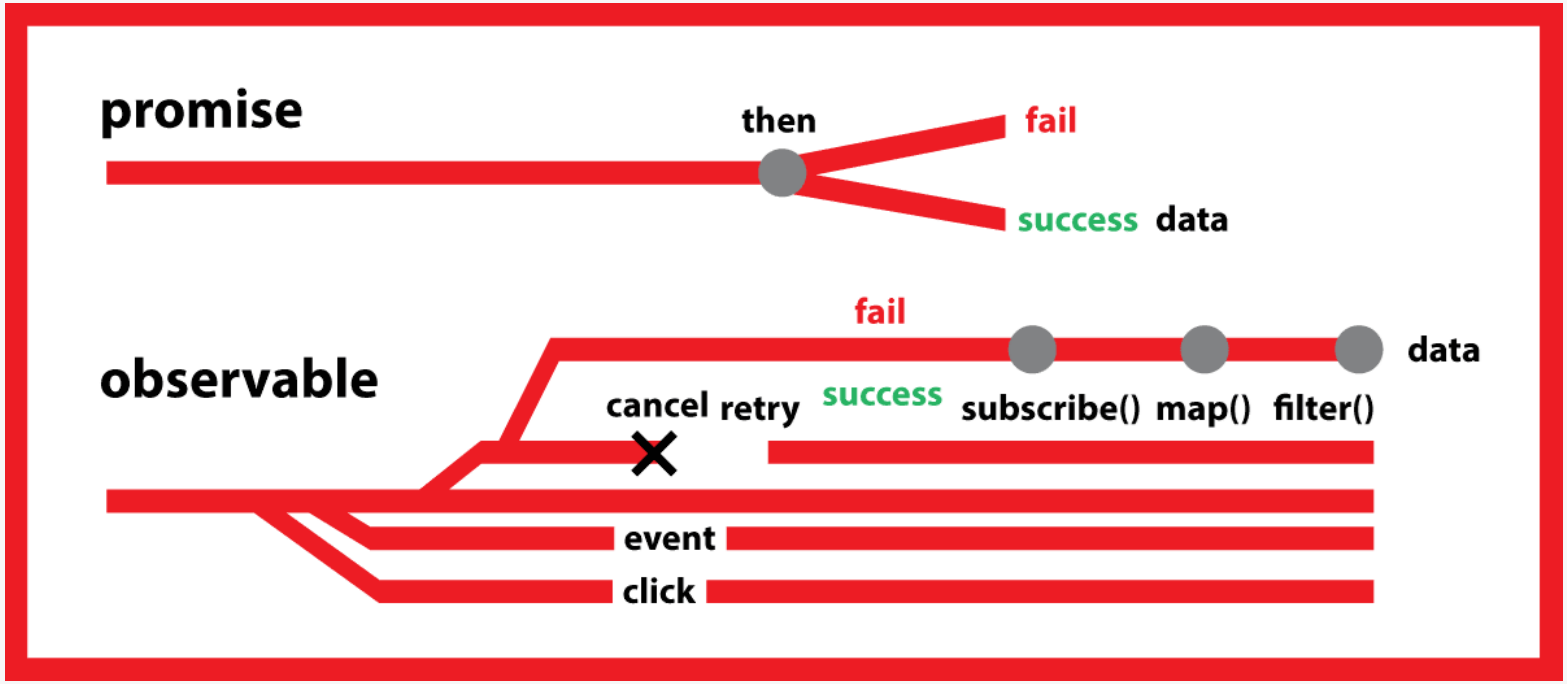


Promise Vs Observable

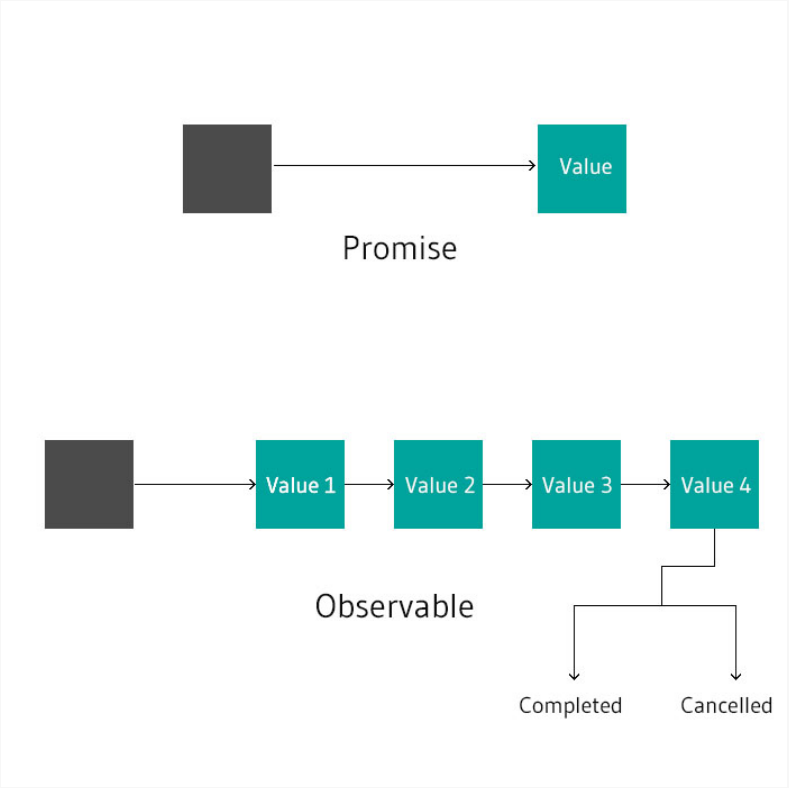
- emette un singolo evento alla volta.
- non permette la cancellazione.
- non fornisce nessun operatore sui dati.
- Non è Lazy. Il servizio viene eseguito immediatamente dopo la creazione.

- emette più valori in un periodo di tempo.
- può essere facilmente cancellato.
- fornisce operatori come map,filter,reduce etc..
- Lazy. Non emette valori finchè qualcuno non si sottoscrive.

Promise Vs Observable



Promise Vs Observable

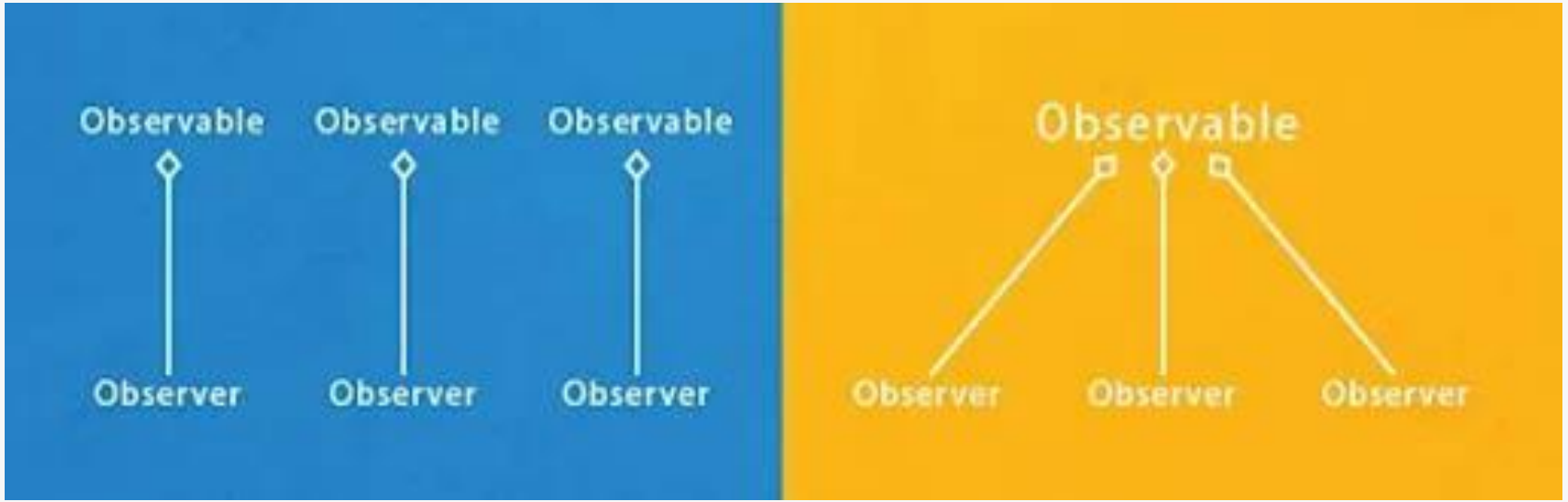


Cold Vs Hot

- completamente lazy
- unicast
- emette solo se ci sono ricevitori sottoscritti
- ogni sottoscrittore riceve una copia dello stream, quindi gli eventi sono sono indipendenti.
- api request, read file etc

- multicast
- emette che ci siano sottoscritti che non.
- tutti i ricevitori si sottoscrivono allo stesso stream e ricevono lo stesso evento.
- mouse / keyboard / system events

Cold Vs Hot


Observable
Observable
Observable
Observer
Observer
Observer
Observable
Observer
Observer
Observer

RxJs

RxJs - Subject

Observables	Subject	Replay Subject	Behavioral Subject
Cold	Hot	Hot	Hot
Creates copy of data	Shares data	Shares data	Shares data
Uni-directionall	bi-directional	bi-directional	bi-directional
-	-	Replay the message stream	Replay the message stream
-	-	-	You can set initial value

RxJs - Operators

Creation	`from`, `fromEvent`, `of`
Combination	`combineLatest`, `concat`, `merge`, `startWith` , `withLatestFrom`, `zip`
Filtering	`debounceTime`, `distinctUntilChanged`, `filter`, `take`, `takeUntil`
Transformation	`bufferTime`, `concatMap`, `map`, `mergeMap`, `scan`, `switchMap`
Utility	`tap`
Multicasting	`share`

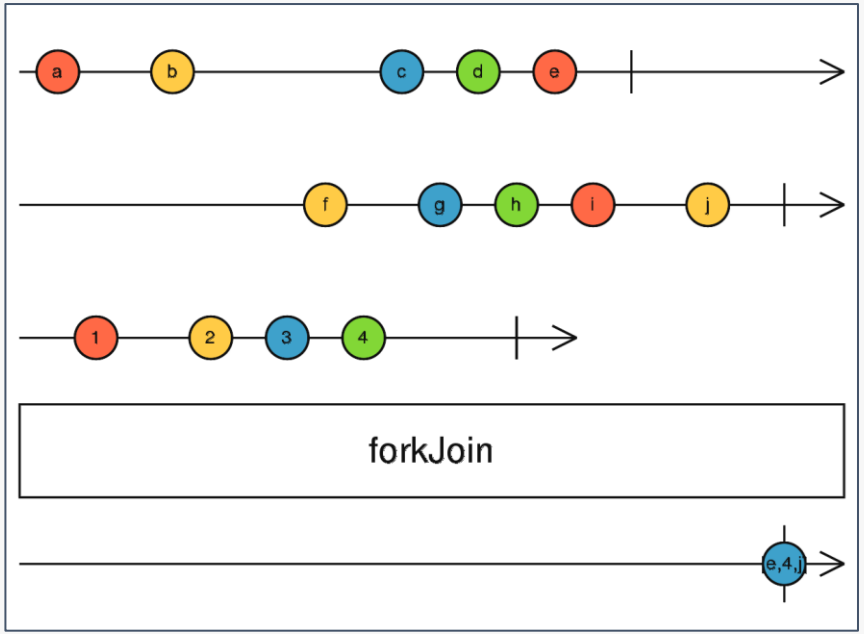
AlmavivA

RxJs - Creations

```
const source = fromEvent(document, 'click');
source.subscribe(res => alert("cliccato!"))
```

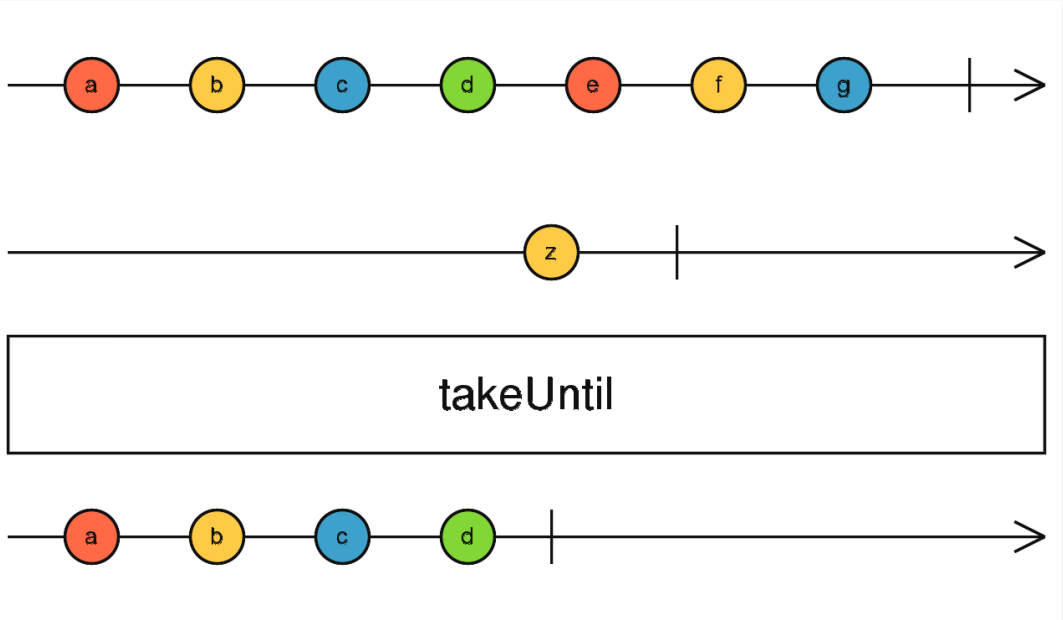
RxJs - Combination

```
forkJoin([
            this.customerService.getDetail(1),
      this.customerService.getPreference(1)
])
.subscribe(([userDetail, userPreference]) => {
            this.customerDetail = userDetail;
      this.customerDetail.preference = userPreference;
})
```



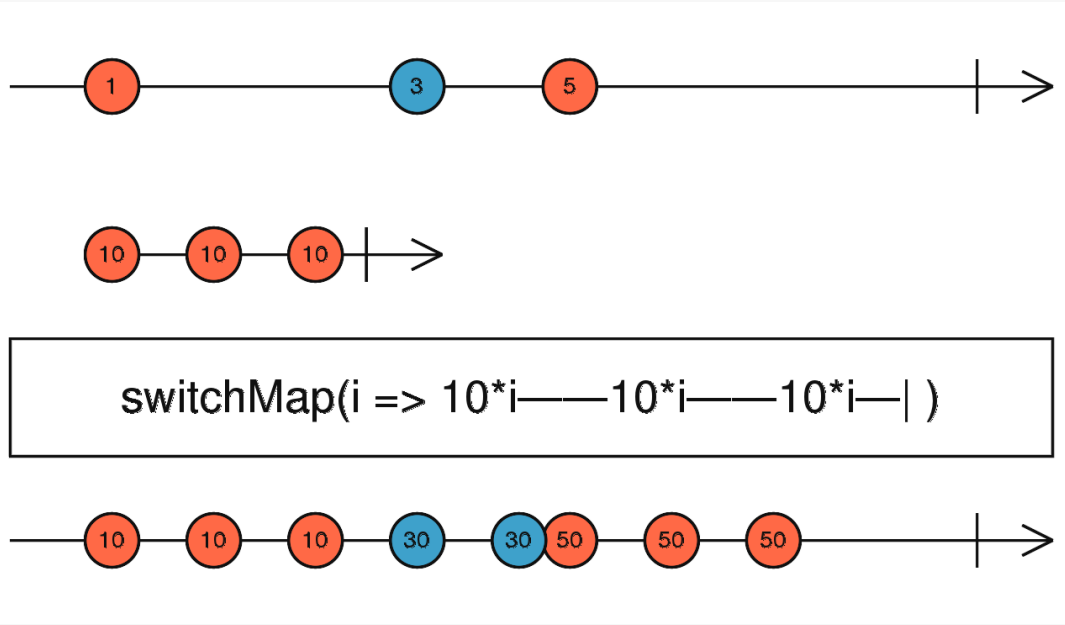
RxJs - Filtering

```
funzione(): void {
  const source = interval(1000);
  const timer$ = timer(5000);
  const example = source.pipe(takeUntil(timer$));
  example.subscribe(val => console.log(val));
}
```

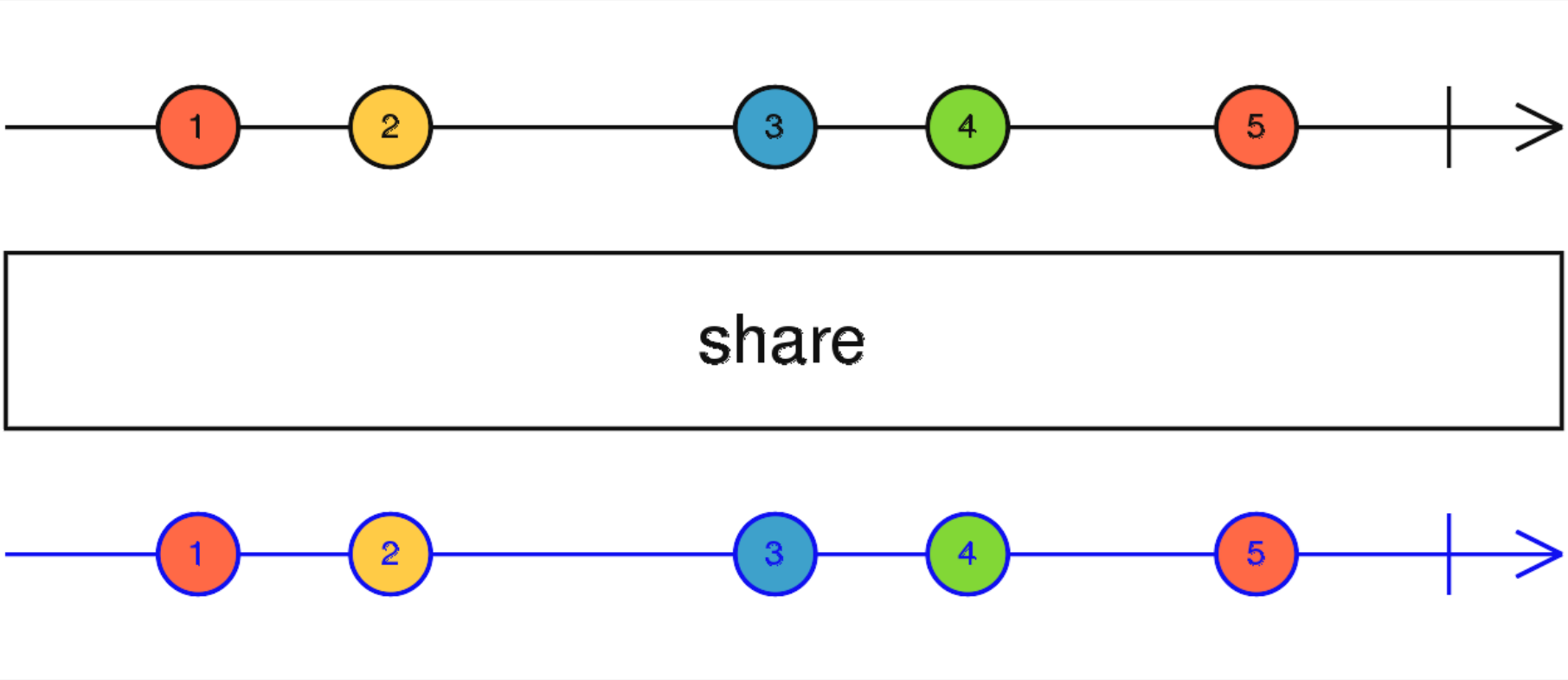


RxJs - Transformation

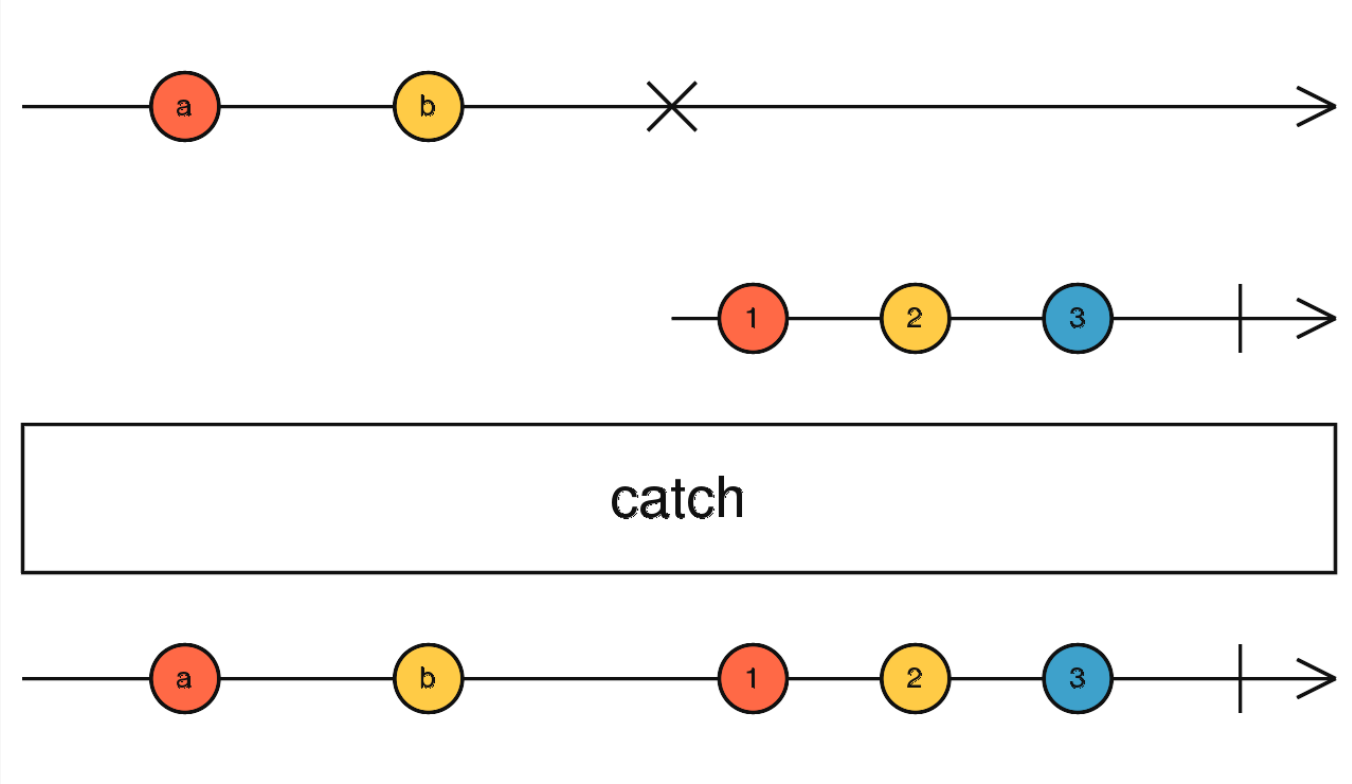
```
this.filterService.getFilter()
.pipe(switchMap(filter => this.getUsers(filter)))
.subscribe(res => console.log(res))
```



RxJs - Multicasting



RxJs - Error Handling



DEMO

AlmavivA
digitale assoluto

almaviva.it