RxJS Marble Testiranje

Angular Belgrade Meetup

Mladen Jakovljević

Ematic Solutions
Banja Luka

Twitter: @jakovljevicMla
GitHub: jakovljevic-mladen

Šta je RxJS?

Šta je Observable?

Kako se pravi Observable?

- `of(value1, value2, ...)`
- `from(value)`
- `fromEvent(document, 'click')`
- `interval()`
- `EMPTY i NEVER`
- `defer()`
- `range()`

Kako se pravi Observable?

```
new Observable( subscribe: subscriber => {
  const xhr = new XMLHttpRequest()
  xhr.onreadystatechange = function () {
    if (xhr.readyState === xhr.DONE) {
      subscriber.next( value: JSON.parse( text: this.response));
      subscriber.complete();
    }
  }
  xhr.open( method: 'GET',  url: 'https://www.github.com');
  xhr.send();
});
```

Angular i Observable-i

- Router
  - events: Observable<Event>
- ActivatedRoute
  - url: Observable<UrlSegment[]>
  - params: Observable<Params>
  - queryParams: Observable<Params>

Angular i Observable-i

- Forms API
  - valueChanges: Observable<any>
  - statusChanges: Observable<any>
- HTTPClient
- EventEmmiter

Subject-i

- Subject
- BehaviorSubject
- ReplaySubject
- AsyncSubject

Scheduler-i

- `queueScheduler`
- `asapScheduler`
- `asyncScheduler`
- `animationFrameScheduler`
- `TestScheduler`

TestScheduler

TestScheduler

```
const ts = new TestScheduler();
```

TestScheduler

```
describe('test', () => {
  let ts: TestScheduler;

  beforeEach(() => {
    ts = new TestScheduler();
  });
});
```

TestScheduler

```
const ts = new TestScheduler();
```

TestScheduler

```
const ts = new TestScheduler((actual, expected) => {

});
```

TestScheduler

```
const ts = new TestScheduler((actual, expected) => {
  expect(actual).toEqual(expected);
});
```

TestScheduler

```
const ts = new TestScheduler((actual, expected) => {
  expect(actual).toEqual(expected);
});

ts.run();
```

TestScheduler

```
ts.run();
```

TestScheduler

```
ts.run(helpers => {

});
```

TestScheduler

```
ts.run(helpers => {
  const { expectObservable } = helpers;
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {

});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  expectObservable(result$)
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  expectObservable(result$).toBe(expected);
});
```

TestScheduler

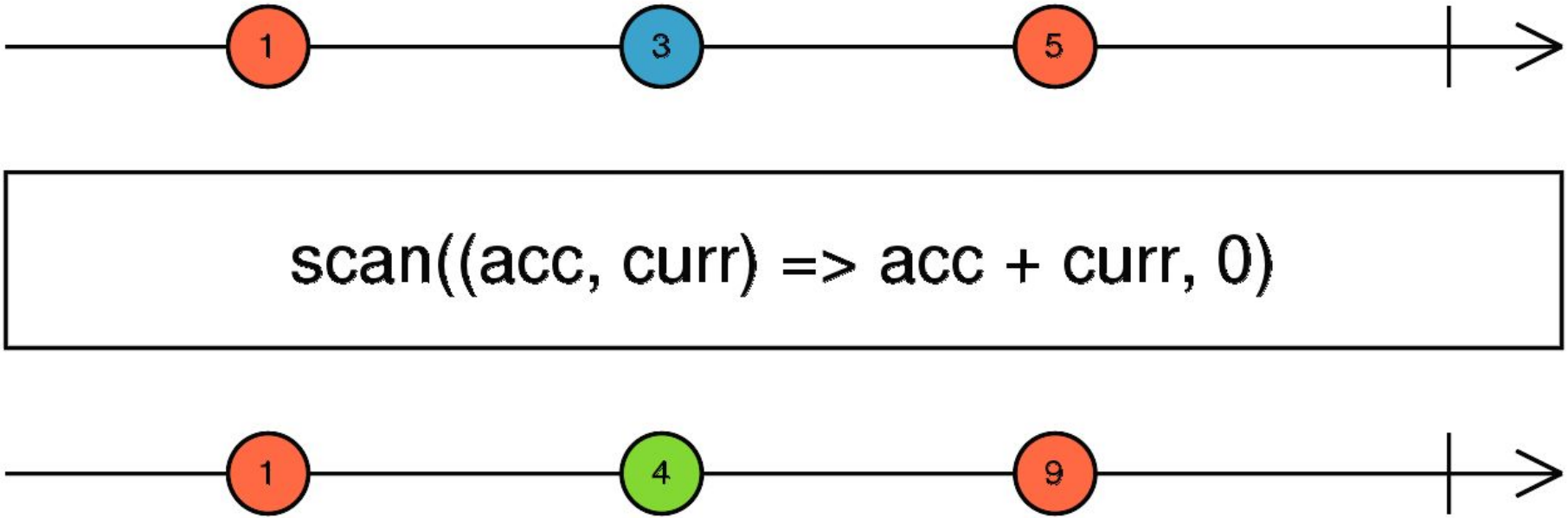
```
ts.run(({ expectObservable }) => {
  const result$ = ?
  const expected = ?
  expectObservable(result$).toBe(expected);
});
```

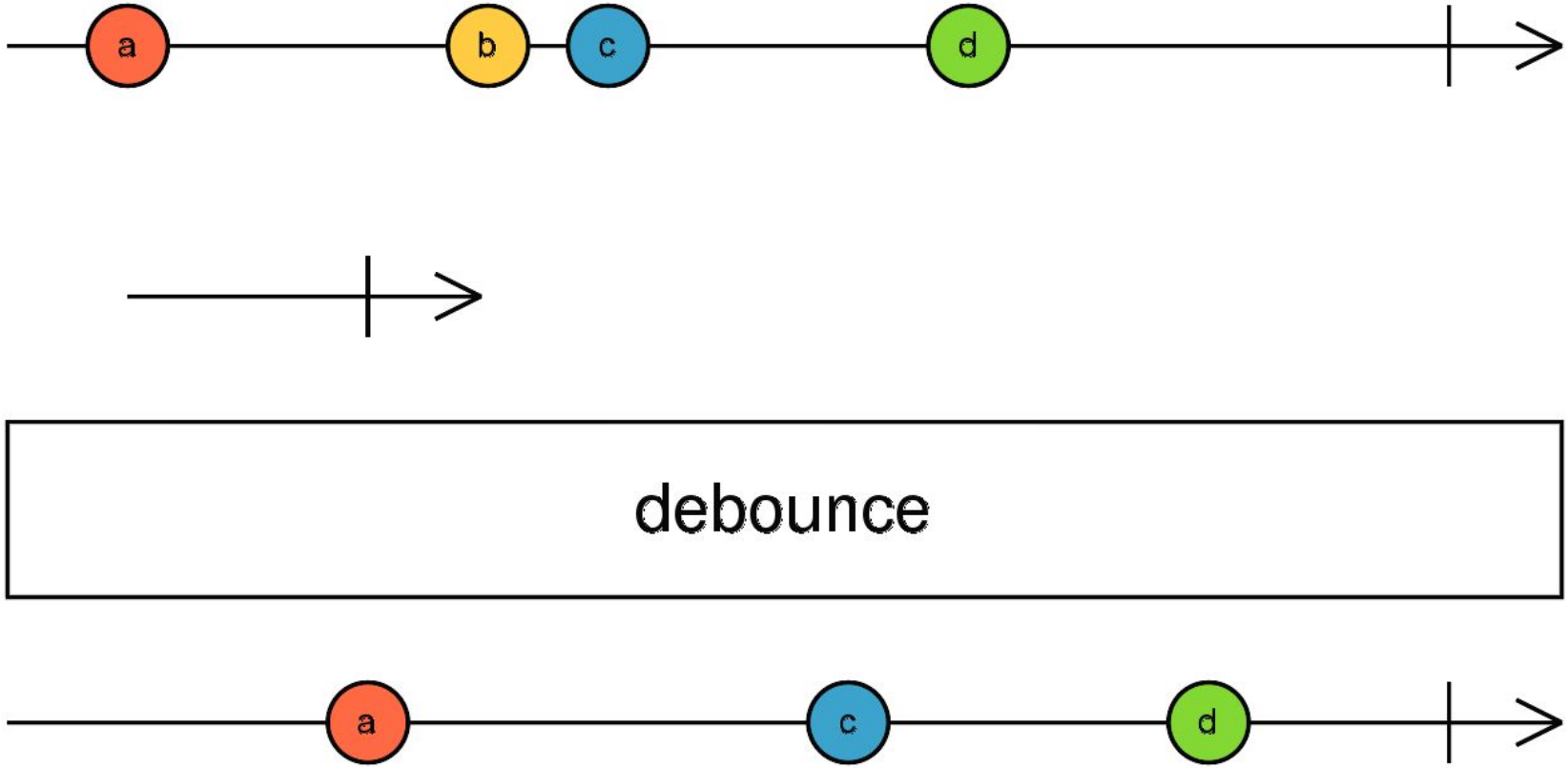
TestScheduler

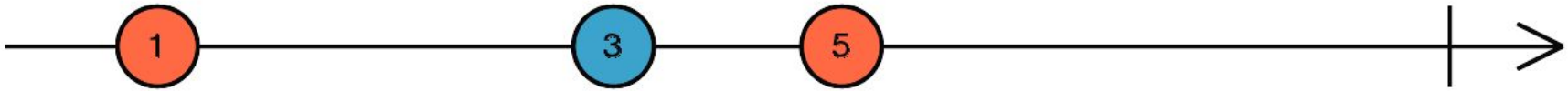
```
ts.run(({ expectObservable }) => {
  const result$ = ? // Observable
  const expected = ?
  expectObservable(result$).toBe(expected);
});
```

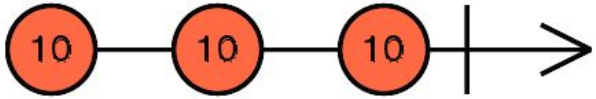
TestScheduler

```
ts.run(({ expectObservable }) => {
  const result$ = ? // Observable
  const expected = ? // string
  expectObservable(result$).toBe(expected);
});
```

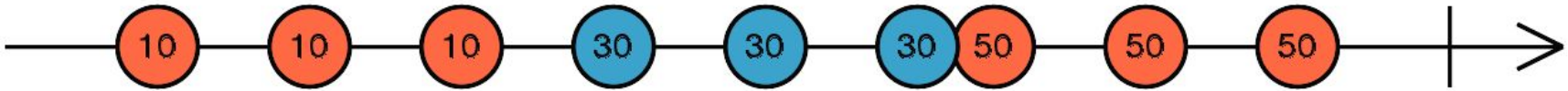

1
3
5
scan((acc, curr) => acc + curr, 0)
1
4
9


a
b
c
d
debounce
a
c
d





concatMap(i => 10*i——10*i——10*i—|)



TestScheduler

```
ts.run(({ expectObservable }) => {
  const result$ = ?
  expectObservable(result$).toBe(expected);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const result$ = timer(2_000, 5_000);
  expectObservable(result$).toBe(expected);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const result$ = timer(2_000, 5_000);
  const expected = '';
  expectObservable(result$).toBe(expected);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const result$ = timer(2, 5);
  const expected = '';
  expectObservable(result$).toBe(expected);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const result$ = timer(2, 5);
  const expected = '--';
  expectObservable(result$).toBe(expected);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const result$ = timer(2, 5);
  const expected = '--0';
  expectObservable(result$).toBe(expected);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const result$ = timer(2, 5);
  const expected = '--0----1';
  expectObservable(result$).toBe(expected);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const result$ = timer(2, 5);
  const expected = '--0----1----2';
  expectObservable(result$).toBe(expected);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const source$ = timer(2, 5);
  const expected = '--0----1----2';
  expectObservable(result$).toBe(expected);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const source$ = timer(2, 5);
  const result$ = source$.pipe(take(3));
  const expected = '--0----1----2';
  expectObservable(result$).toBe(expected);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const source$ = timer(2, 5);
  const result$ = source$.pipe(take(3));
  const expected = '--0----1----2';
  const values = { 0: 0, 1: 1, 2: 2 };
  expectObservable(result$).toBe(expected, values);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const source$ = timer(2, 5);
  const result$ = source$.pipe(take(3));
  const expected = '--a----b----c';
  const values = { a: 0, b: 1, c: 2 };
  expectObservable(result$).toBe(expected, values);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const source$ = timer(2, 5);
  const result$ = source$.pipe(take(3));
  const expected = '--a----b----(c|)';
  const values = { a: 0, b: 1, c: 2 };
  expectObservable(result$).toBe(expected, values);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const source$ = timer(2, 5);
  const result$ = source$.pipe(take(3));
  const expected = '--a----b----(c|)';
  const values = { a: 0, b: 1, c: 2 };
  expectObservable(result$).toBe(expected, values);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const values = { a: 0, b: 1, c: 2 };
  const source$ = timer(2, 5);
  const expected = '--a----b----(c|)';
  const result$ = source$.pipe(take(3));
  expectObservable(result$).toBe(expected, values);
});
```

TestScheduler

```
ts.run(({ expectObservable, cold }) => {
  const values = { a: 0, b: 1, c: 2 };
  const source$ = timer(2, 5);
  const expected = '--a----b----(c|)';
  const result$ = source$.pipe(take(3));
  expectObservable(result$).toBe(expected, values);
});
```

TestScheduler

```
ts.run(({ expectObservable, cold }) => {
  const values = { a: 0, b: 1, c: 2 };
  const source$ = cold('');
  const expected = '--a----b----(c|)';
  const result$ = source$.pipe(take(3));
  expectObservable(result$).toBe(expected, values);
});
```

TestScheduler

```
ts.run(({ expectObservable, cold }) => {
  const values = { a: 0, b: 1, c: 2 };
  const source$ = cold('--a----b----c');
  const expected = '--a----b----(c|)';
  const result$ = source$.pipe(take(3));
  expectObservable(result$).toBe(expected, values);
});
```

TestScheduler

```
ts.run(({ expectObservable, cold }) => {
  const values = { a: 0, b: 1, c: 2 };
  const source$ = cold('--a----b----c----d----e');
  const expected = '--a----b----(c|)';
  const result$ = source$.pipe(take(3));
  expectObservable(result$).toBe(expected, values);
});
```

TestScheduler

```
ts.run(({ expectObservable, cold }) => {
  const values = { a: 0, b: 1, c: 2 };
  const source$ = cold('--a----b----c----d----e');
  const expected = '    --a----b----(c|)       ';
  const result$ = source$.pipe(take(3));
  expectObservable(result$).toBe(expected, values);
});
```

TestScheduler

```
ts.run(({ expectObservable, cold }) => {
  const values = { a: 0, b: 1, c: 2 };
  const source$ = cold('--a----b----c----d----e', values);
  const expected = '    --a----b----(c|)       ';
  const result$ = source$.pipe(take(3));
  expectObservable(result$).toBe(expected, values);
});
```

TestScheduler

```
ts.run(({ expectObservable, cold }) => {
  const values = { a: 0, b: 1, c: 2 };
  const source$ = cold('--a----b----c----d----e', values);
  const expected = '    --a----b----(c|)       ';
  const result$ = source$.pipe(take(3));
  expectObservable(result$).toBe(expected, values);
});
```

TestScheduler

```
ts.run(({ expectObservable, cold }) => {
  const source$ = cold('--a----b----c----d----e');
  const expected = '    --a----b----(c|)       ';
  const result$ = source$.pipe(take(3));
  expectObservable(result$).toBe(expected);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const result$ = timer(2, 5);
  const expected = '--0----1----2';
  expectObservable(result$).toBe(expected);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const result$ = timer(2, 5);
  const expected = '--0----1----2';
  const unsub = '    ';
  expectObservable(result$, unsub).toBe(expected);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const result$ = timer(2, 5);
  const expected = '--0----1----2';
  const unsub = '   ---------------!';
  expectObservable(result$, unsub).toBe(expected);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const result$ = timer(2, 5);
  const expected = '--0----1----2';
  const unsub = '   ------------!';
  expectObservable(result$, unsub).toBe(expected);
});
```

TestScheduler

```
ts.run(({ expectObservable }) => {
  const result$ = timer(2, 5);
  const expected = '--0----1-----';
  const unsub = '   ------------!';
  expectObservable(result$, unsub).toBe(expected);
});
```

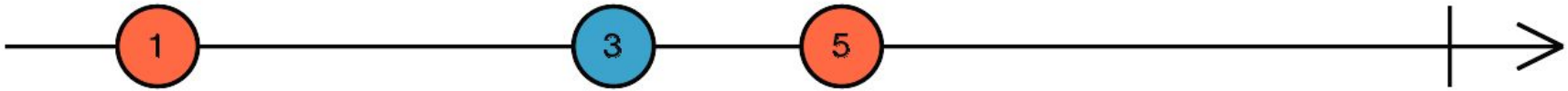
TestScheduler

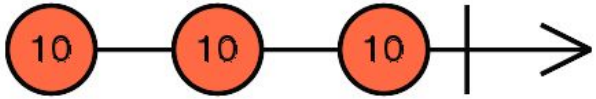
```
ts.run(({ expectObservable }) => {
  const result$ = timer(2, 5);
  const expected = '--0----1----2---';
  const unsub = '   ---------------!';
  expectObservable(result$, unsub).toBe(expected);
});
```

TestScheduler

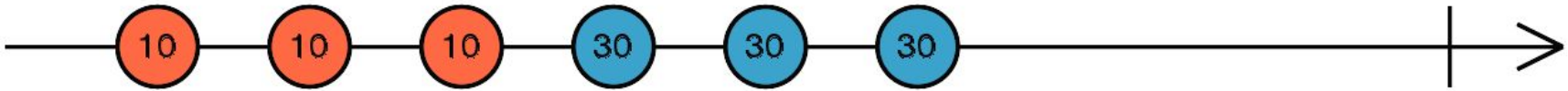
```
ts.run(({ expectObservable }) => {
  const result$ = timer(2, 5);
  const expected = '--0----1----2---';
  const unsub = '   ---------------!';
  expectObservable(result$, unsub).toBe(expected, [0, 1, 2]);
});
```

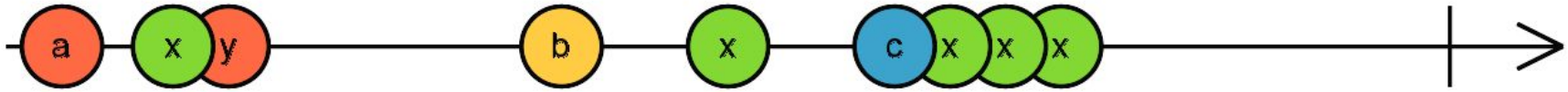
```
it('should map-and-flatten each item to an Observable', () => {
  testScheduler.run(({ cold, hot, expectObservable, expectSubscriptions }) => {
    const values = { x: 10, y: 30, z: 50 };
    const e1 = hot('   --1-----3--5-------|');
    const e1subs = '   ^------------------!';
    const e2 = cold('    x-x-x|           ', values);
    //                        x-x-x|
    //                           x-x-x|
    const expected = ' --x-x-x-y-y-y------|';

    const result = e1.pipe(exhaustMap((x) => e2.pipe(map((i) => i * +x))));

    expectObservable(result).toBe(expected, values);
    expectSubscriptions(e1.subscriptions).toBe(e1subs);
  });
});
```

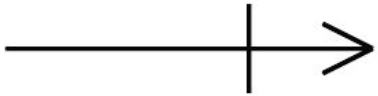




exhaustMap(i => 10*i——10*i——10*i—|)







throttle(fn, { leading: true, trailing: true })

```
it('should throttle with duration Observable using next to close the duration',
  testScheduler.run(({ cold, hot, expectObservable, expectSubscriptions }) => {
    const e1 = hot('  -a-xy-----b--x--cxxx-|');
    const e1subs = '  ^--------------------!';
    const e2 = cold('  ----i-j-k            ');
    //                         ----i-j-k
    //                              ----i-j-k
    const e2subs = [
      '               -^---!               ',
      '               ----------^---!      ',
      '               ----------------^---! ',
    ];
    const expected = '-a--------b-----c----|';

    const result = e1.pipe(throttle(() => e2));

    expectObservable(result).toBe(expected);
    expectSubscriptions(e1.subscriptions).toBe(e1subs);
    expectSubscriptions(e2.subscriptions).toBe(e2subs);
  });
});
```

```
it('should emit the last value in each time window', () => {
  testScheduler.run(({ cold, hot, expectObservable, expectSubscriptions }) => {
    const e1 = hot('  -a-xy-----b--x--cxxx-|');
    const e1subs = '  ^--------------------!';
    const e2 = cold('  ----i-j-k            ');
    //                        ----i-j-k
    //                             ----i-j-k
    const e2subs = [
      '               -^---!               ',
      '               ----------^---!      ',
      '               ----------------^---! ',
    ];
    const expected = '-----y--------x-----x|';

    const result = e1.pipe(audit(() => e2));

    expectObservable(result).toBe(expected);
    expectSubscriptions(e1.subscriptions).toBe(e1subs);
    expectSubscriptions(e2.subscriptions).toBe(e2subs);
  });
});
```

Pisanje operatora

- Korištenjem `pipe()` funkcije
- `<S, D>(source: Observable<S>) => Observable<D>;`
- Na `source` treba da se `subscribe`-ujemo
- Prilikom `subscribe`-a treba da slušamo svaki event
- Prilikom `unsubscribe`-a, potrebno je da se `unsubscribe`-ujemo i od `source` Observable-a

```
it('should allow unsubscribing explicitly and early', () => {
  testScheduler.run(({ hot, expectObservable, expectSubscriptions }) => {
    const e1 = hot('  --a--b--c--d--|');
    const e1subs = '  ^------!       ';
    const expected = '--------       ';
    const unsub = '   -------!       ';

    expectObservable(e1.pipe(last()), unsub).toBe(expected);
    expectSubscriptions(e1.subscriptions).toBe(e1subs);
  });
});
```

Hvala na pažnji

https://github.com/ReactiveX/rxjs

Testovi RxJS operatora

https://github.com/jakovljevic-mladen/ng-rx-multi-tab-logged-state-update

Twitter: @jakovljevicMla
GitHub: jakovljevic-mladen