NEW IN RXJS 7

INTRODUCTION

MARTIN VAN DAM

Frontend Software Engineer @ Philips / CODE.STAR

@MrtnvDam / martin.van.dam@ordina.nl

WHAT IS RXJS

- Reactive programming in the Frontend & Node
- A (better) way to manage data and events within your app.

WHY RXJS

- Functional and Reactive
- Better readable code
- Data flow
- Easier and safer data transformations

WHAT'S NEW IN RXJS 7

- New methods
- New & renamed Operators
- Improved configuration options
- Improved typings

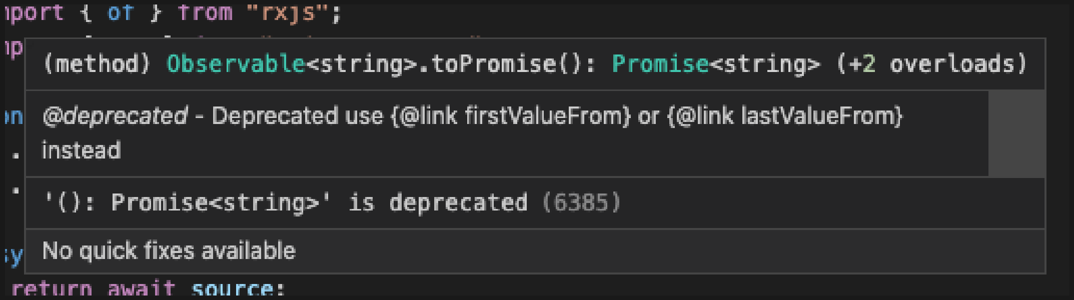
NEW METHOD

`firstValueFrom`

- First value of stream
- Unsubscribes

.toPromise()

```
// before v7
const abc = of("a", "b", "c");
const someValue = await abc.toPromise();
// -> 'c'
```



```
(method) Observable<string>.toPromise(): Promise<string> (+2 overloads)
```

@deprecated - Deprecated use {@link firstValueFrom} or {@link lastValueFrom} instead

```
'(): Promise<string>' is deprecated (6385)
```

No quick fixes available

firstValueFrom

```
const abc = of("a", "b", "c");
const firstValue = await firstValueFrom(abc);
// -> 'a'
// before v7
const empty = of();
const someValue = await empty.toPromise();
// -> undefined (?)
```

firstValueFrom

```js
import { of, firstValueFrom } from "rxjs";

const empty = of();

try {
  await firstValueFrom(empty);
} catch (error) {
  // -> EmptyError: 'no elements in sequence'
}
```

NEW METHOD

`lastValueFrom`

- Last value of stream
- Resolves when completed

lastValueFrom

```
const abc = of("a", "b", "c");
const firstValue = await lastValueFrom(abc);
// -> 'c'
```

lastValueFrom

```
const empty = of();

try {
  await lastValueFrom(empty);
} catch (error) {
  // -> EmptyError: 'no elements in sequence'
}
```

NEW OPERATOR

`concatWith`

- Subscribes to input observable
- Waits until completion and subscribes to second

EXAMPLE

`concatWith`

```
const clicks = fromEvent(document, "click");
const moves = fromEvent(document, "mousemove");

clicks.pipe(
  map(() => "click"),
  concatWith(moves.pipe(map(() => "move")))
);
// -> 'click', 'move', 'move', 'move'
```

NEW OPERATOR

`switchScan`

- Proposed in 2017
- Uses `switchMap` under the hood
- Works like `reduce`

EXAMPLE

`switchScan`

```
const source = of(1, 2, 3);
const example = source.pipe(
  switchScan((acc, value) => {
    return of(acc + value);
  }, 0)
);
// -> 1, 3, 6
```

RENAMED OPERATORS

- Improved consistency & readability
- Deprecated in v7, removed in v8

- `combineLatest` -> `combineLatestWith`
- `zip` -> `zipWith`
- `race` -> `raceWith`

EXTENDED CONFIGURATION

`retry`

- Accepts config object
- `resetOnSuccess` option added

retry

```
// before
const source = fromFetch('https://can-randomly-fail.com');
const example = source.pipe(
  retry(2)
);
```

retry

```
// after
const source = fromFetch('https://can-randomly-fail.com');
const example = source.pipe(
  retry({
    count: 2,
    resetOnSuccess: true, // new!
  })
);
```

EXTENDED CONFIGURATION

`timeout`

- Configuration options extended

timeout

```
// before
const seconds = interval(1000);
const delayed = seconds.pipe(
  timeout(1100)
);
```

timeout

- each
- first
- with
- meta

timeout

```
// after
const seconds = interval(1000);
const delayed = seconds.pipe(
  timeout({
    each: 1100,
    first: 5000,
    with: () => throwError(new NoValueReceivedError()),
    meta: {
      some: 'metadata'
    },
  });
);
```

IMPROVED TYPINGS

groupBy

```
function isPerson(value: Person | Pet): value is Person {
  return value.hasOwnProperty("name");
}

const person: Person = { name: "Judy" };
const pet: Pet = { kind: "cat" };

const o = of(person, pet).pipe(
  groupBy(isPerson),
  mergeMap((group) => {
    if (group.key) {
      const inferred = group; // -> Person
      return inferred;
    } else {
      const inferred = group; // -> Pet
```

DICTONARY SUPPORT

`combineLatest`

```
// Before:
const nums$ = of(1, 2, 3);
const chars$ = of("a", "b", "c");
const bools$ = of(true, false, false);

const example = combineLatest(nums$, chars$, bools$);
// -> [ 3, 'c', true ]
// After:
const nums$ = of(1, 2, 3);
const chars$ = of("a", "b", "c");
const bools$ = of(true, false, false);

const example = combineLatest({
  number: nums$,
  character: chars$,
  boolean: bools$,
});
// -> { number: 3, character: 'c', boolean: true }
```

OTHER NOTEWORTHY CHANGES

- Memory usage reduced
- Improved typings for `filter`, `groupBy`, `combineLatest`
- Improved TestScheduler
- Minor breaking changes
- Bugfixes

WHEN CAN I USE ALL THIS? 😍

THANKS!

And have fun with all the new goodies of RxJS 7! 👐

@MrtnvDam / martin.van.dam@ordina.nl

SOURCES

- RxJS 7 Roadmap
- RxJS Changelog
- RxJS 7 release delay