RxJS Operators

Deborah Kurata
CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | blogs.msmvps.com/deborahk/

RxJS Operators

Start the stream

Emits items

Items pass through a set of operations

As an observer

Next item, process it

Error occurred, handle it

Complete, you're done

Stop the stream

RxJS Operators

Start the stream

- Emits items

Items pass through a set of operations

As an observer

- Next item, process it
- Error occurred, handle it
- Complete, you're done

Stop the stream

Items are piped through a set of operators

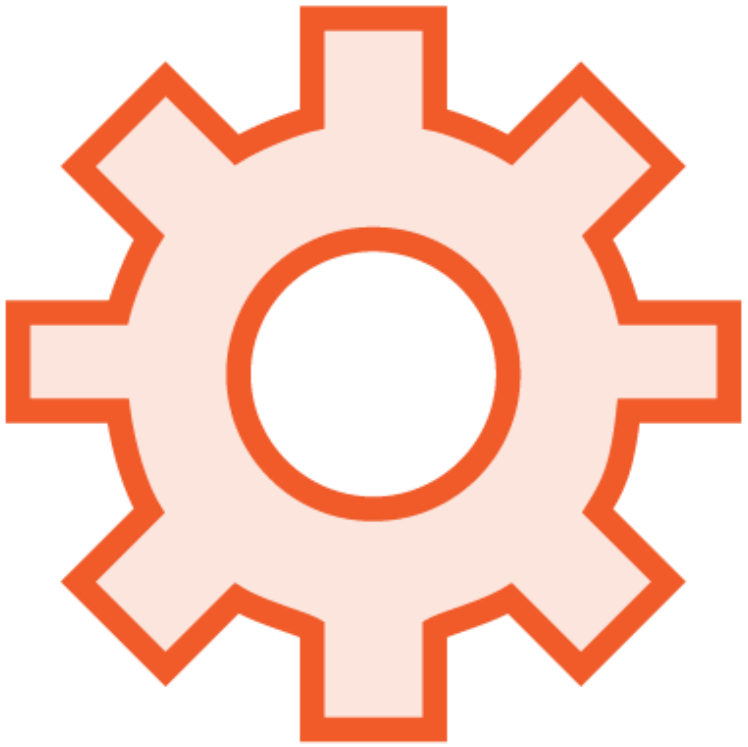
- Fashioned after .NET LINQ operators
- Similar to array methods such as filter and map

Module Overview

RxJS Operators

- Overview
- Documentation
- Examples

RxJS Features

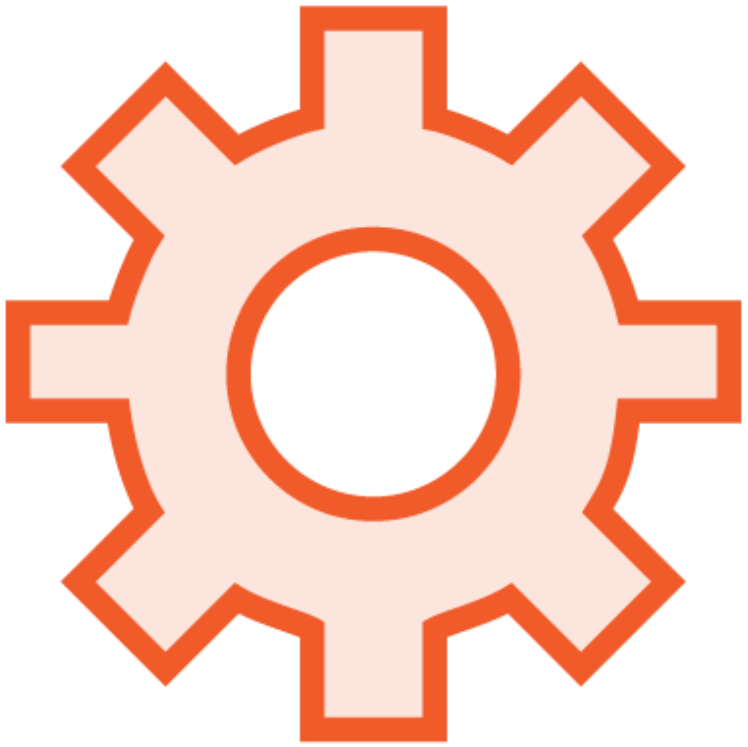


`map`

`tap`

`take`

RxJS Operators



An operator is a function

Used to transform and manipulate items in an Observable stream

Apply operators in sequence using the Observable's pipe method

RxJS Operators

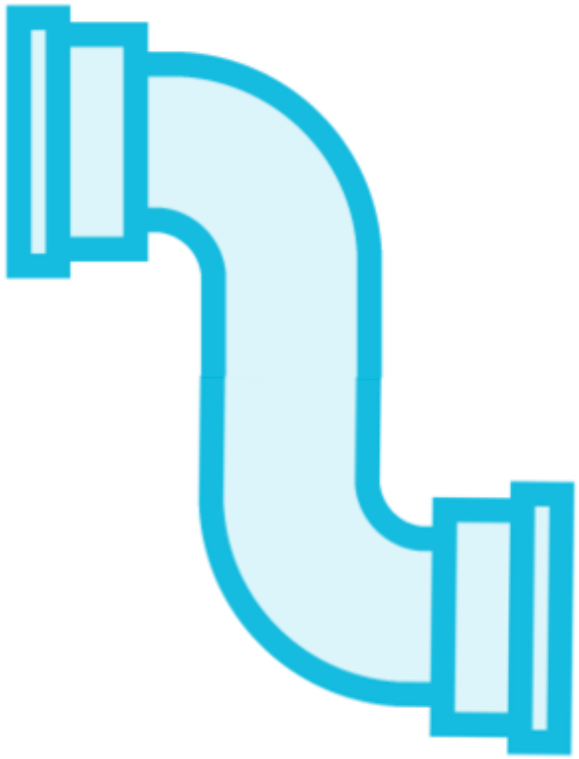
```
of(2, 4, 6)
  .pipe(
     map(item => item * 2),
     tap(item => console.log(item)),
     take(2)
  ).subscribe(console.log);
```

RxJS Operators

- audit
- auditTime
- buffer
- bufferCount
- bufferTime
- bufferToggle
- bufferWhen
- catchError
- combineAll
- ~~combineLatest (deprecated)~~
- ~~concat (deprecated)~~
- concatAll
- concatMap
- concatMapTo
- count
- debounce
- debounceTime
- defaultIfEmpty
- delay
- delayWhen
- dematerialize
- distinct
- distinctUntilChanged
- distinctUntilKeyChanged
- elementAt
- endWith
- every
- exhaust
- exhaustMap
- expand
- filter
- finalize
- find
- findIndex
- first
- flatMap
- groupBy
- ignoreElements
- isEmpty
- last
- map
- mapTo
- materialize
- max
- ~~merge (deprecated)~~
- mergeAll
- mergeMap
- mergeMapTo
- mergeScan
- min
- multicast
- observeOn
- onErrorResumeNext
- pairwise
- ~~partition (deprecated)~~
- pluck
- publish
- publishBehavior
- publishLast
- publishReplay
- ~~race (deprecated)~~
- reduce
- refCount
- repeat
- repeatWhen
- retry
- retryWhen
- sample
- sampleTime
- scan
- sequenceEqual
- share
- shareReplay
- single
- skip
- skipLast
- skipUntil
- skipWhile
- startWith
- subscribeOn
- switchAll
- switchMap
- switchMapTo
- take
- takeLast
- takeUntil
- takeWhile
- tap
- throttle
- throttleTime
- throwIfEmpty
- timeInterval
- timeout
- timeoutWith
- timestamp
- toArray
- window
- windowCount
- windowTime
- windowToggle
- windowWhen
- withLatestFrom
- ~~zip (deprecated)~~
- zipAll

https://rxjs.dev

RxJS Operator: `map`



Transforms each emitted item

```
map(item => item * 2)
```

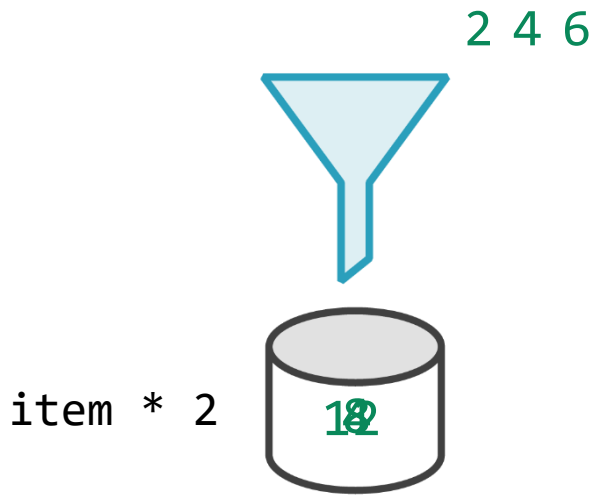
For each item in the source, one mapped item is emitted

Used for

- Making changes to each item

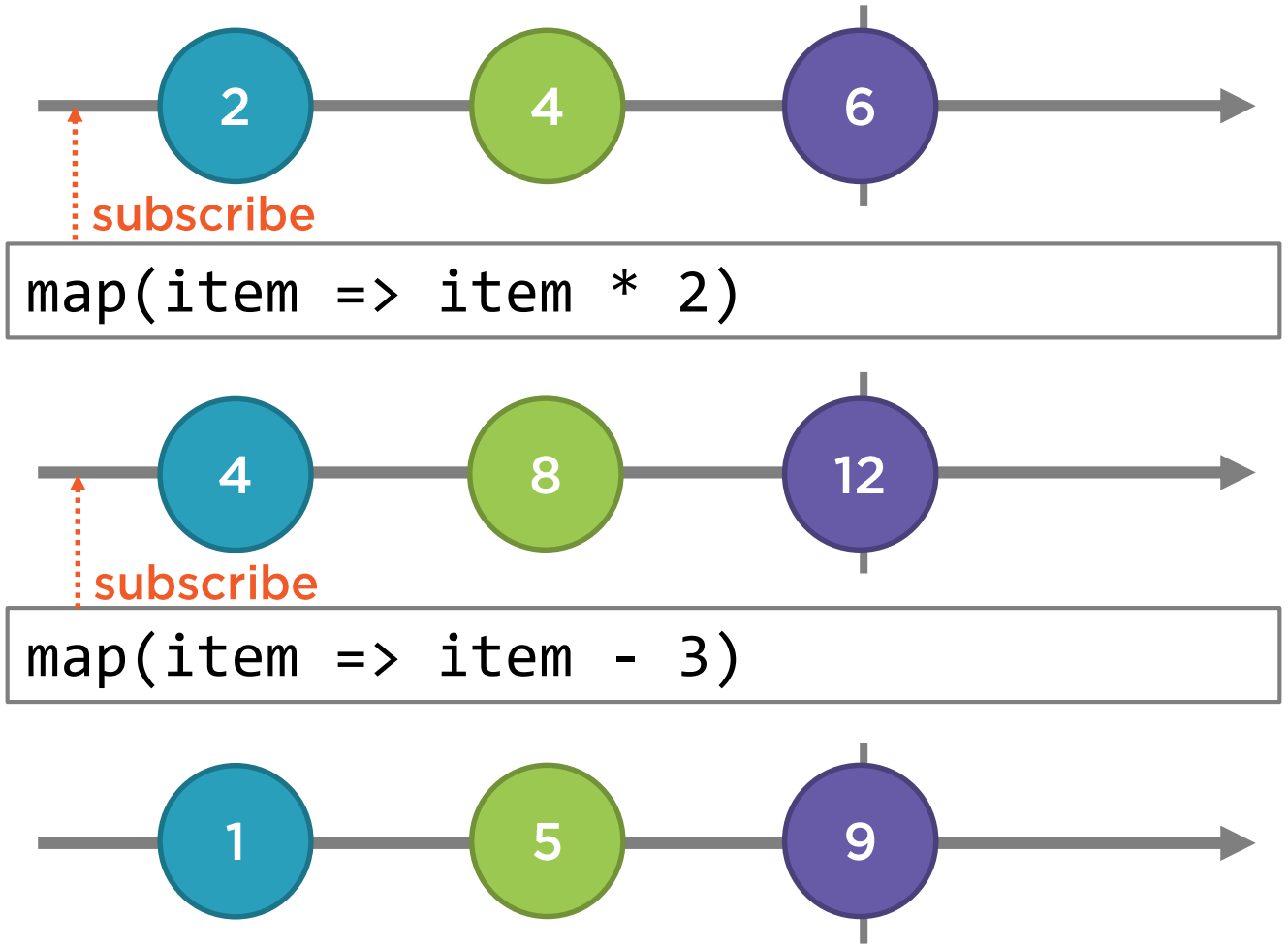
RxJS Operator: **map**

```
of(2, 4, 6)
    .pipe(
      map(item => item * 2)
    ).subscribe(console.log);
```

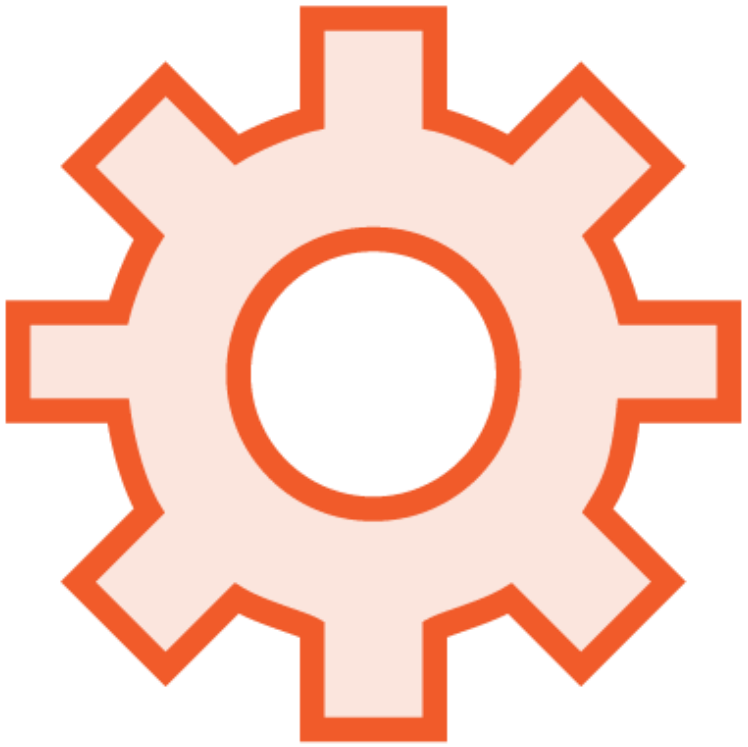


Marble Diagram: `map`

```
of(2, 4, 6)
    .pipe(
      map(item => item * 2),
      map(item => item - 3)
    ).subscribe(console.log);
```



RxJS Operator: `map`



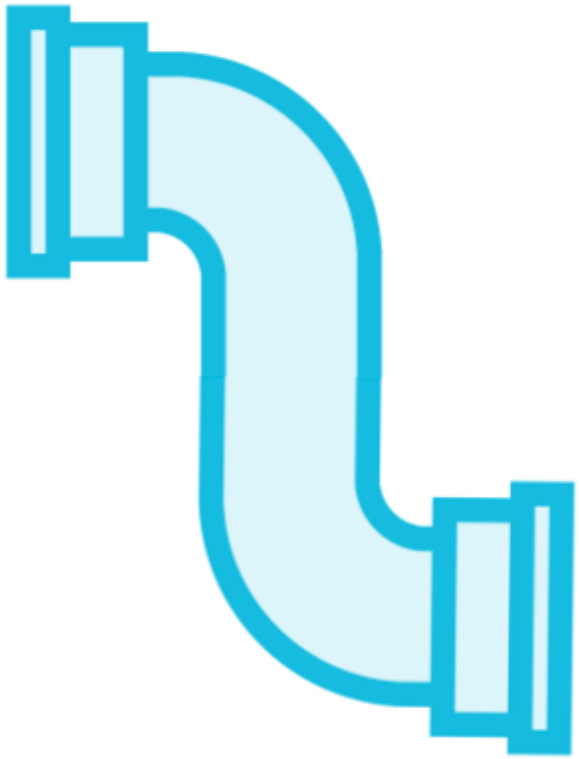
`map` **is a transformation operator**

- Takes in an input stream, subscribes
- Creates an output stream

When an item is emitted

- Item is transformed as specified by a provided function
- Item is emitted to the output stream

RxJS Operator: `tap`



Taps into a stream without modifying it

```
tap(item => console.log(item))
```

Used for

- Debugging
- Performing actions outside of the flow of data

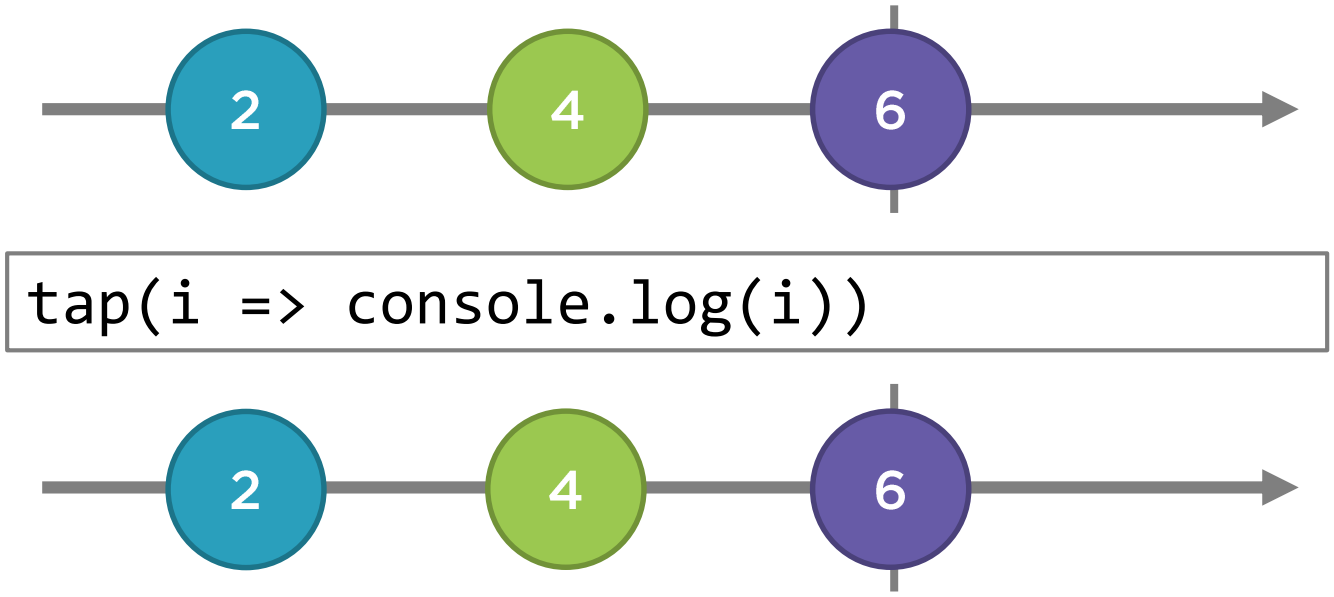
RxJS Operator: `tap`

```
of(2, 4, 6)
    .pipe(
      tap(item => console.log(item)),
      map(item => item * 2),
      tap(item => console.log(item)),
      map(item => item - 3),
      tap(item => console.log(item))
    ).subscribe();
```

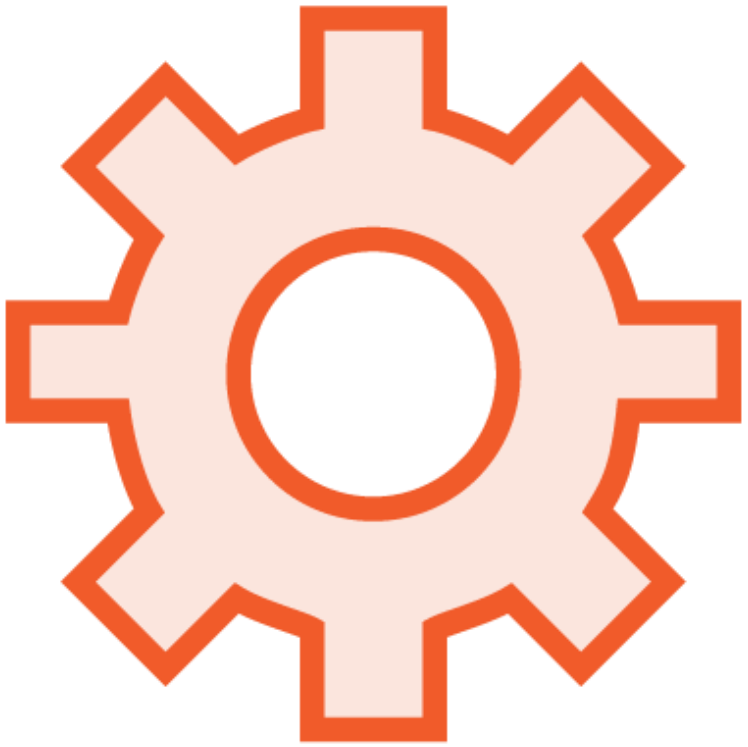
2
4
1
4
8
5
6
12
9

Marble Diagram: `tap`

```
of(2, 4, 6)
    .pipe(
      tap(i => console.log(i))
    ).subscribe(console.log);
```



RxJS Operator: `tap`



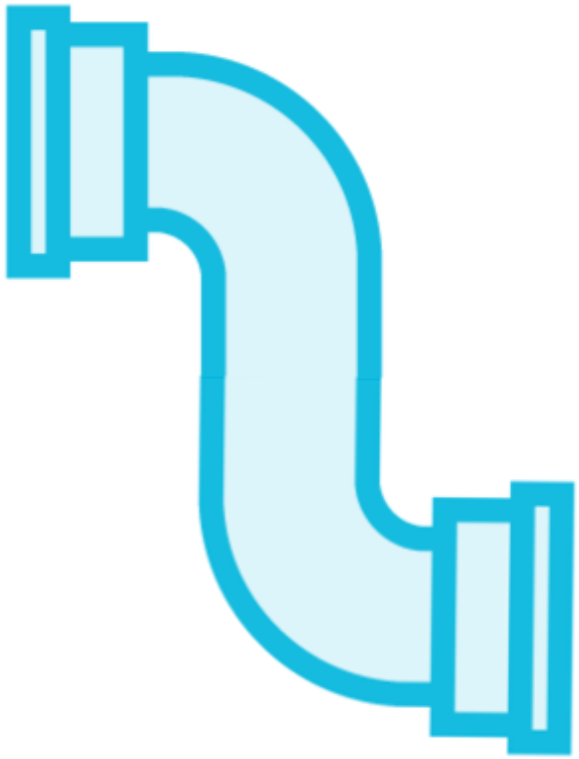
`tap` is a utility operator

- Takes in an input stream, subscribes
- Creates an output stream

When an item is emitted

- Performs a side effect as specified by a provided function
- Item is emitted to the output stream

RxJS Operator: `take`



Emits a specified number of items

```
take(2)
```

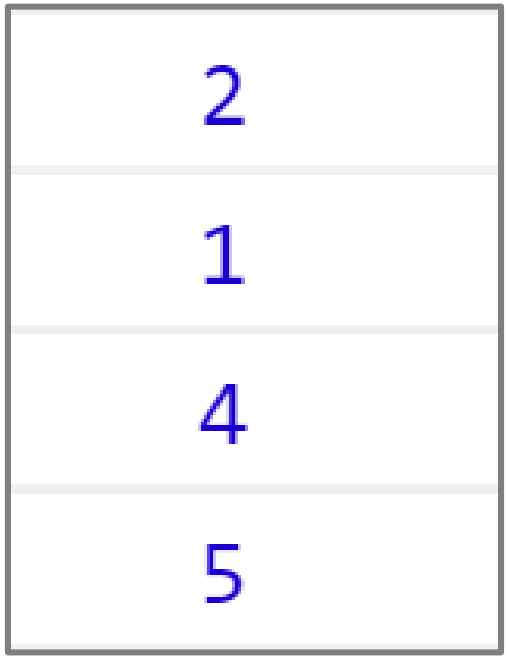
Used for

- Taking a specified number of items
- Limiting unlimited streams

RxJS Operator: take

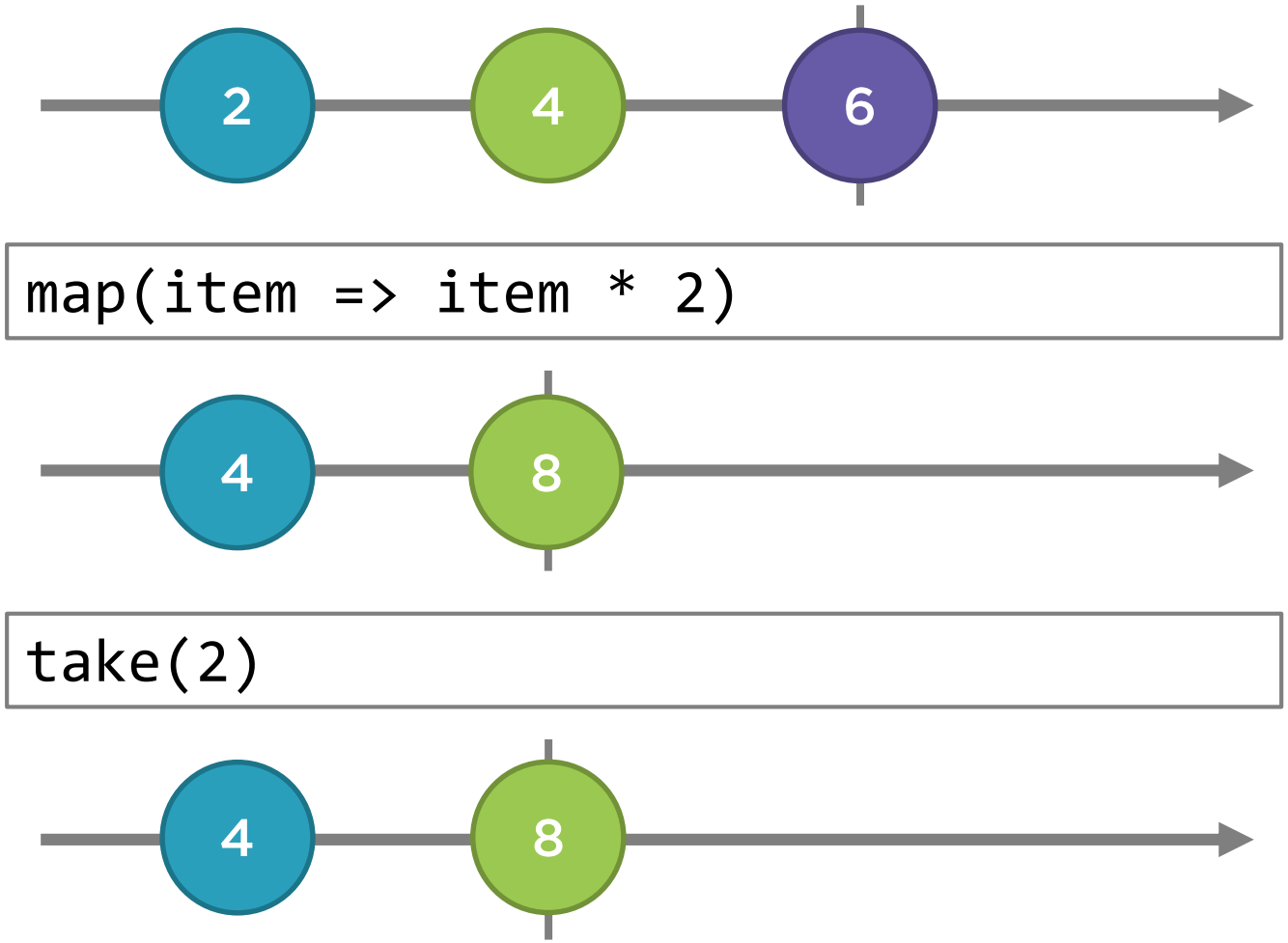
```
of(2, 4, 6)
    .pipe(
      take(2)
    ).subscribe(console.log);  // 2 4
```

```
of(2, 4, 6)
    .pipe(
      tap(item => console.log(item)),
      map(item => item * 2),
      take(2),
      map(item => item - 3),
      tap(item => console.log(item))
    ).subscribe();
```

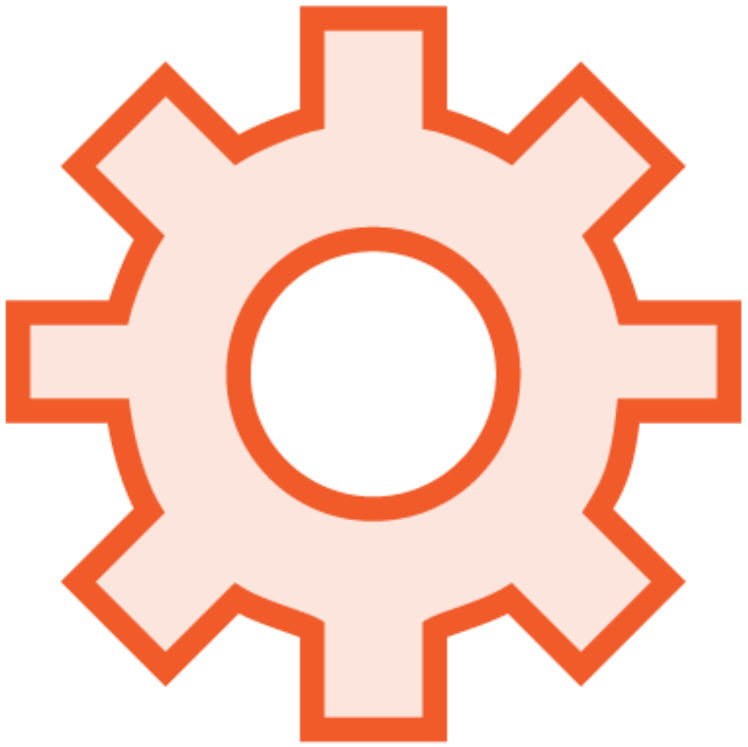


Marble Diagram: `map` and `take`

```
of(2, 4, 6)
    .pipe(
      map(item => item * 2),
      take(2)
    ).subscribe(console.log);
```



RxJS Operator: take



take **is a filtering operator**

- Takes in an input stream, subscribes
- Creates an output stream

When an item is emitted

- Counts the item
  - If <= specified number, emits item to the output stream
  - When it equals the specified number, it completes

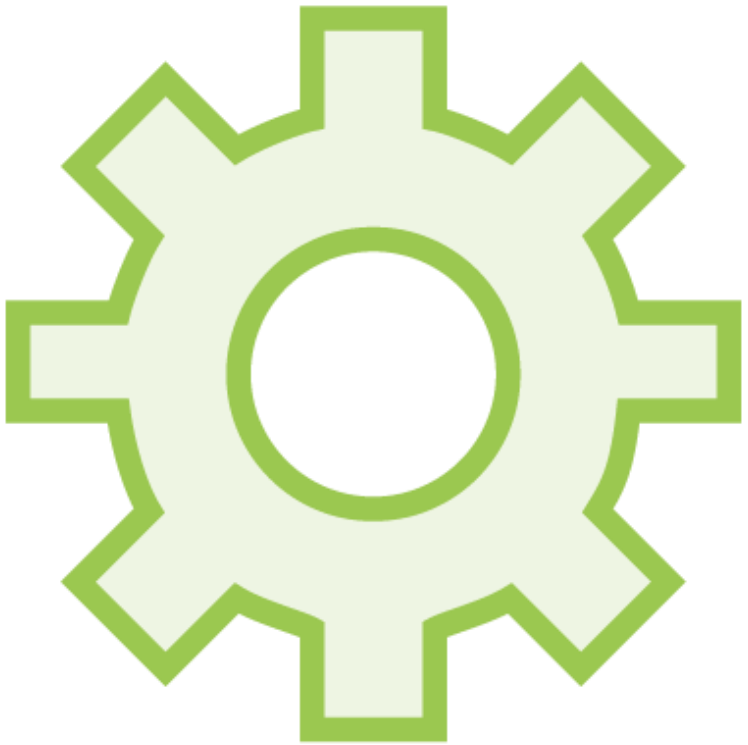
Only emits the defined number of items

Demo

RxJS Operators:

- `map`
- `tap`
- `take`

Operators

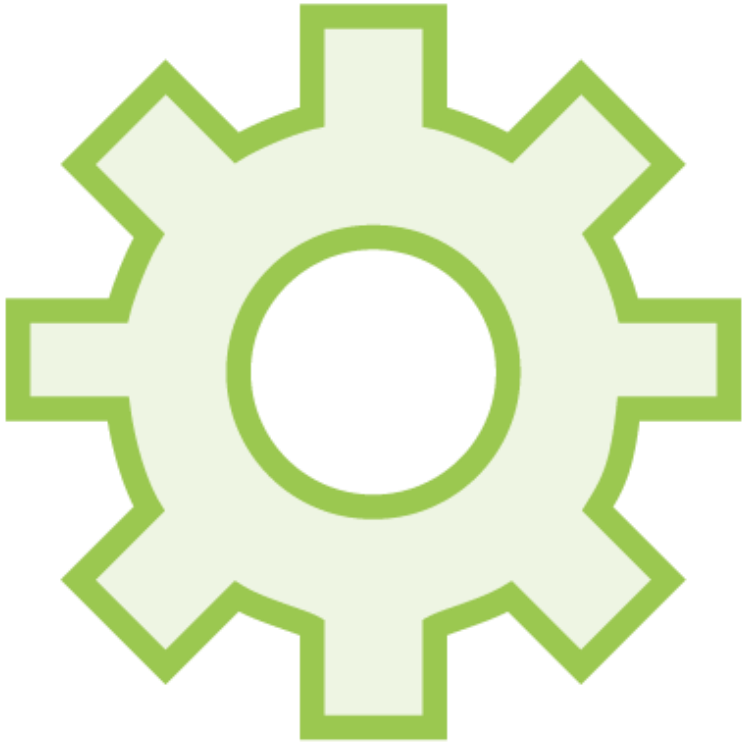


Use the Observable pipe method ...

... To pipe emitted items through a sequence of operators

```
from([20, 15, 10, 5]).pipe(
  tap(item => console.log(item)),
  take(3),
  map(item => item * 2),
  map(item => item - 10),
  map(item => {
    if (item === 0) {
      throw new Error('zero detected');
    }
    return item;
  })
);
```

RxJS Features



map: Transforms each emitted item

```
map(item => item * 2)
```

tap: Taps into the stream without modifying it

```
tap(item => console.log(item))
```

take: Emits the specified number of items and completes

```
take(2)
```