

Deborah Kurata Consultant | Speaker | Author | MVP | GDE

@deborahkurata



Start

Emits items

Item passes through a set of operations

As an observer

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

Stop





Each emitted item can be piped through a set of operators

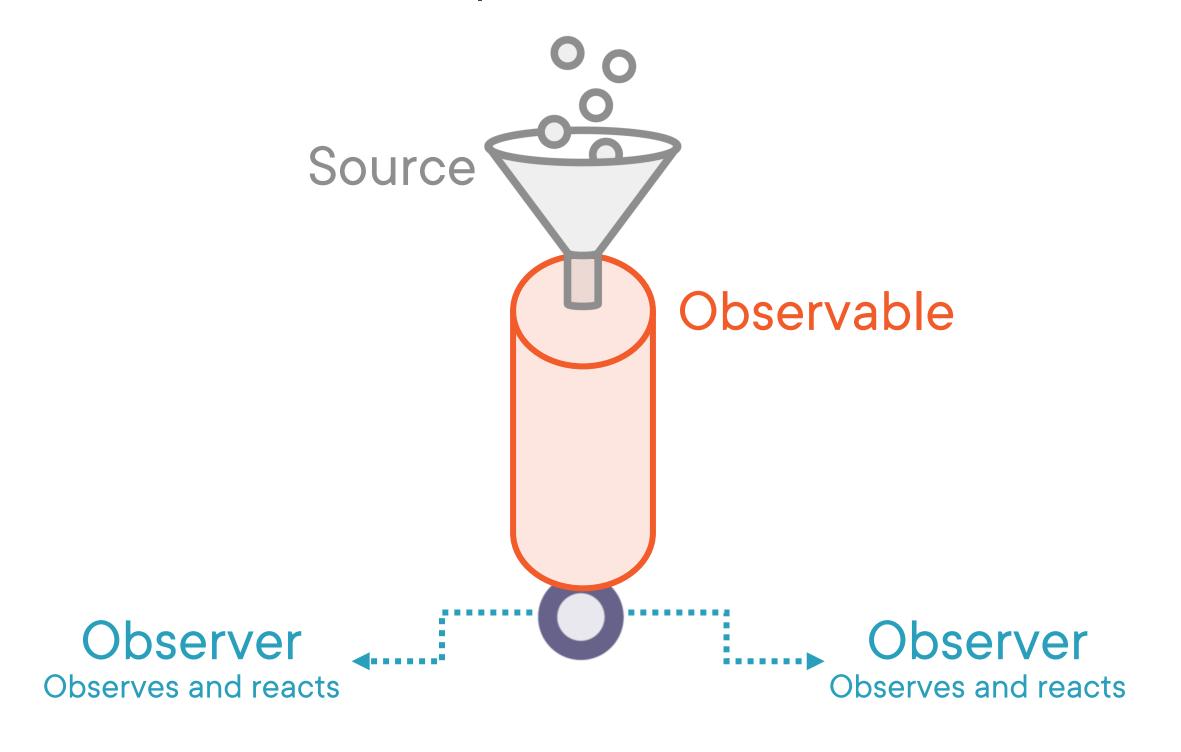
- Transform, filter, process, ...
- Delay, timeout, ...

Fashioned after .NET LINQ operators

Similar to array methods such as filter and map

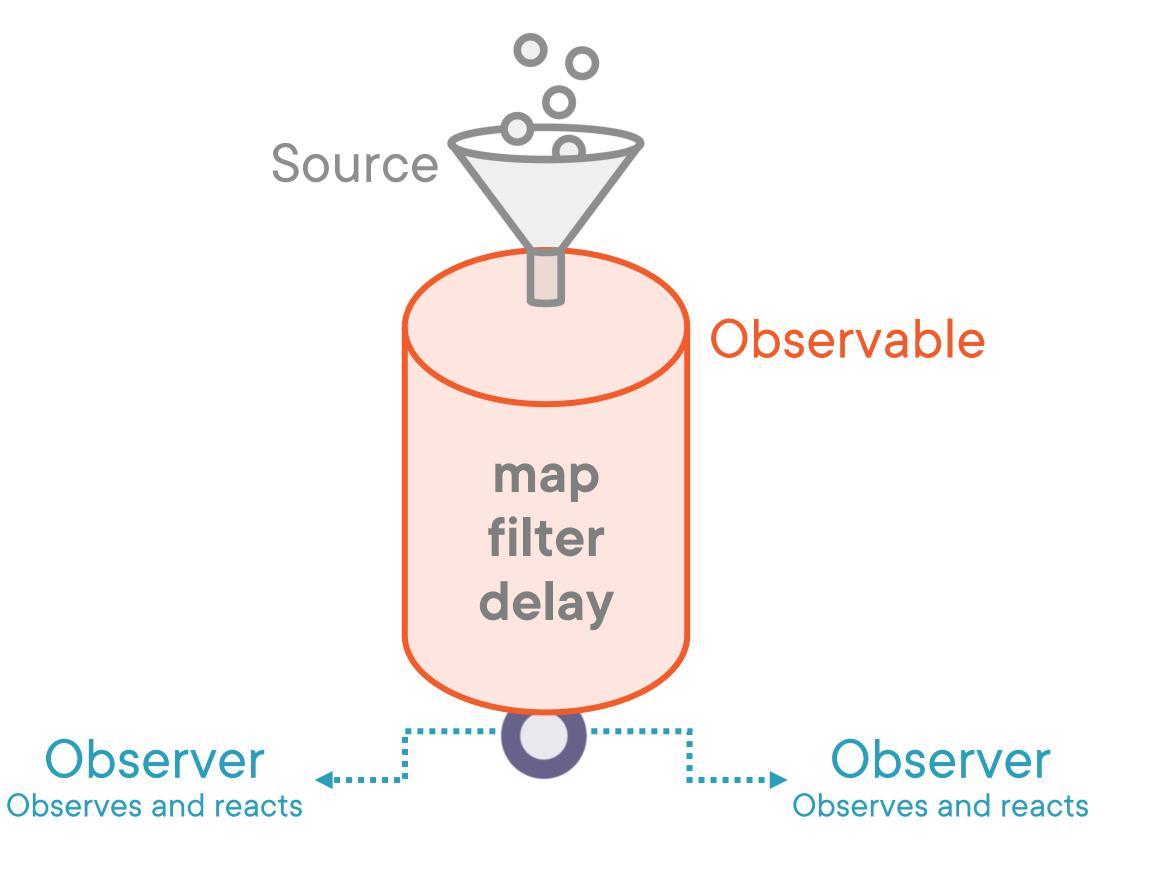


Operators





Operators





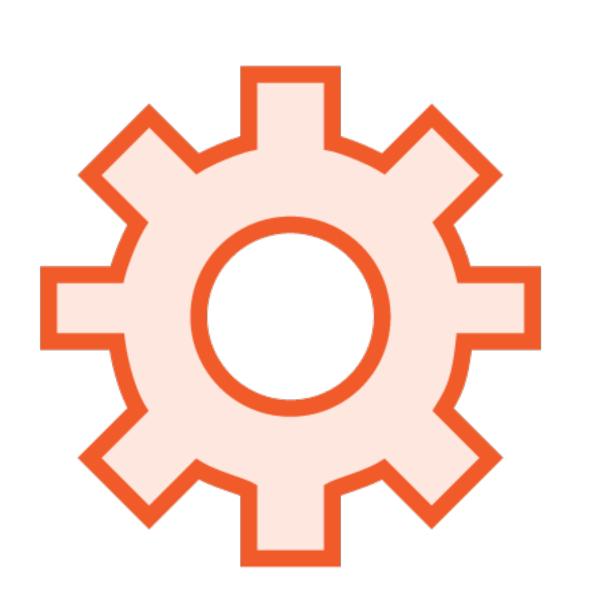
Module Overview



RxJS operators

- Overview
- Documentation
- Examples
- Internals

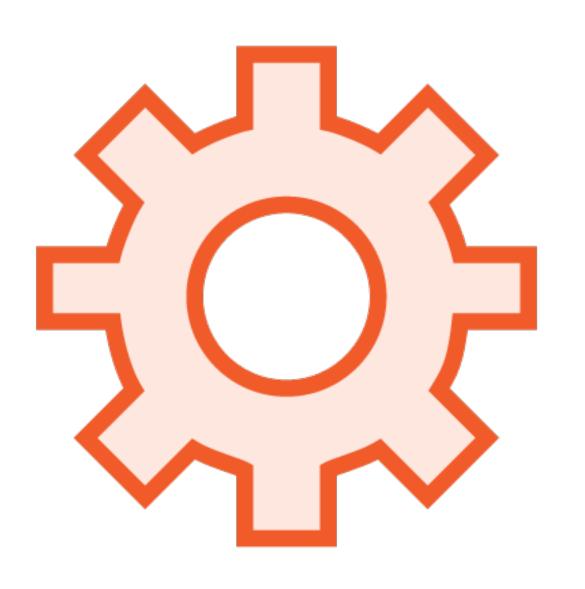
RxJS Features



map

tap

take



An operator is a function

Used to transform and manipulate emitted items

Apply operators in sequence using the Observable's pipe() method

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

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

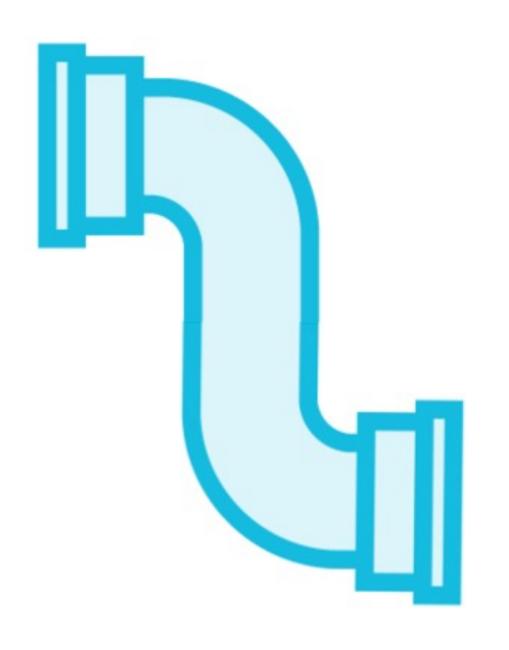
F	audit	F	auditTime	F	buffer
F	bufferCount	F	bufferTime	F	bufferToggle
F	bufferWhen	F	catchError	K	combineAll (deprecated)
F	combineLatest (deprecated)	F	combineLatestAll	F	combineLatestWith
F	concat (deprecated)	F	concatAll	F	concatMap
F	concatMapTo	F	concatWith	F	connect
F	count	F	debounce	F	debounceTime
F	defaultIfEmpty	F	delay	F	delayWhen
F	dematerialize	F	distinct	F	distinctUntilChanged
F	distinctUntilKeyChanged	F	elementAt	F	endWith
F	every	K	exhaust (deprecated)	F	exhaustAll
F	exhaustMap	F	expand	F	filter
F	finalize	F	find	F	findIndex
F	first	K	flatMap (deprecated)	F	groupBy
F	ignoreElements	F	isEmpty	F	last
F	map	F	тарТо	F	materialize
F	max	F	merge	F	mergeAll
F	mergeMap	F	mergeMapTo	F	mergeScan
F	mergeWith	F	min	F	multicast (deprecated)

F observeOn	■ onErrorResumeNext	F pairwise
partition (deprecated)	F pluck (deprecated)	publish (deprecated)
publishBehavior (deprecated)	F publishLast (deprecated)	publishReplay (deprecated)
F race (deprecated)	F raceWith	F reduce
refCount (deprecated)	F repeat	repeatWhen
F retry	retryWhen	F sample
F sampleTime	F scan	F sequenceEqual
F share	F shareReplay	F single
F skip	F skipLast	F skipUntil
skipWhile	F startWith	F subscribeOn
F switchAll	F switchMap	■ switchMapTo
F switchScan	F take	F takeLast
F takeUntil	F takeWhile	F tap
F throttle	F throttleTime	■ throwIfEmpty
F timeInterval	F timeout	F timeoutWith
F timestamp	F toArray	F window
▼ windowCount	F windowTime	▶ windowToggle
■ windowWhen	■ withLatestFrom	F zip (deprecated)
F zipAll	F zipWith	

https://rxjs.dev



RxJS Operator: map



Transforms each emitted item

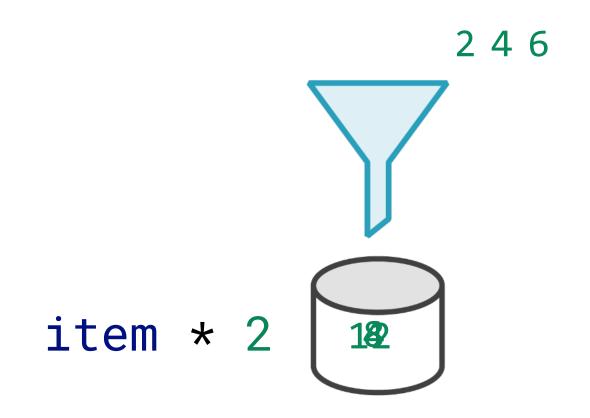
For each item emitted in, one mapped item is emitted out

Used for

- Making changes to each item

RxJS Operator: map

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



Marble Diagram: map

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

Marble Diagram: map

Marble Diagram: map

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

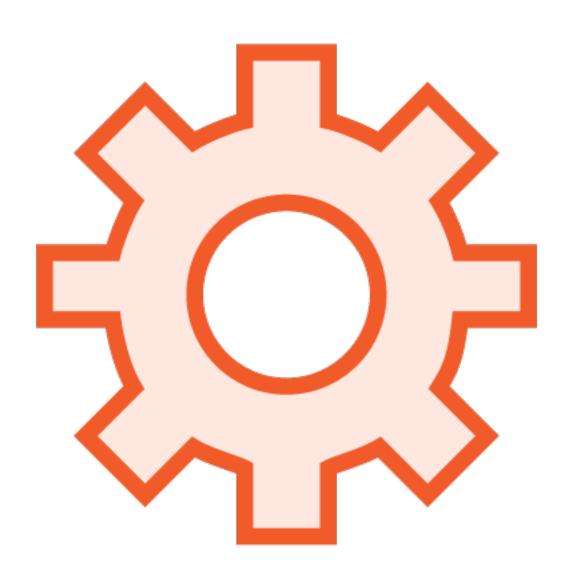
Console
    1
    5
    9
    map(item => item * 2)

map(item => item * 2)

map(item => item - 3)
```



RxJS Operator: map



map is a transformation operator

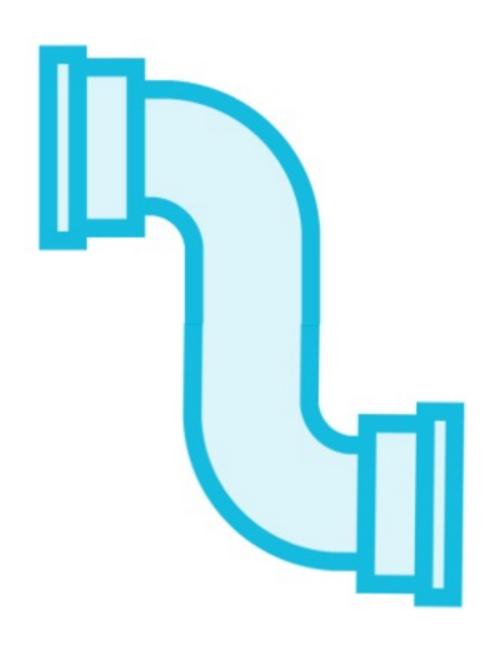
- Subscribes to its input Observable
- Creates an output Observable

When an item is emitted

- Item is transformed as specified by the provided function
- Transformed item is emitted to the output
 Observable



RxJS Operator: tap



Taps into the emissions without affecting the items

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

For each item emitted in, the same item is emitted out

Used for

- Debugging
- Performing actions outside of the flow of data (side effects)



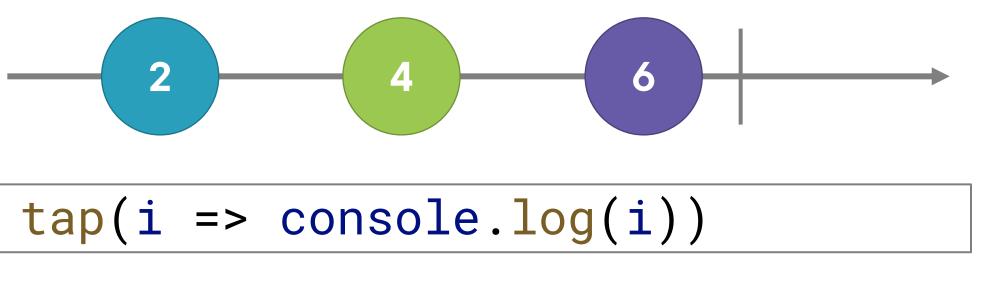
RxJS Operator: tap

```
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

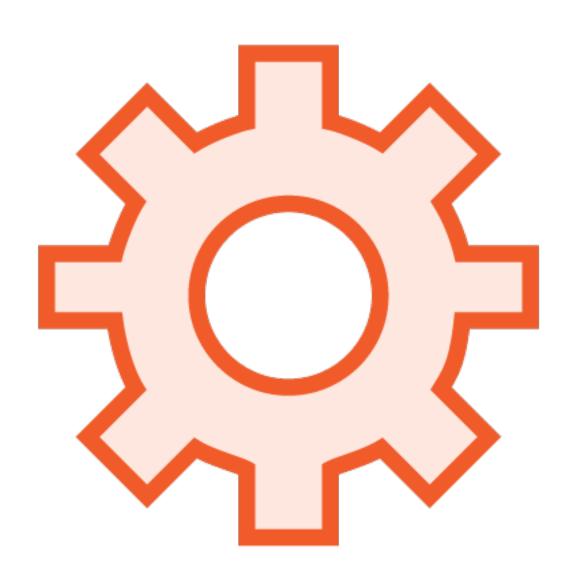
Marble Diagram: tap

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





RxJS Operator: tap



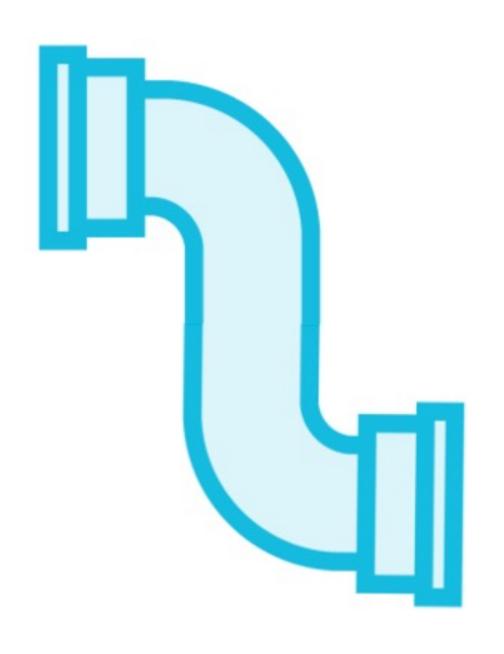
tap is a utility operator

- Subscribes to its input Observable
- Creates an output Observable

When an item is emitted

- Performs a side effect as specified by a provided function
- Original item is emitted to the output
 Observable

RxJS Operator: take



Emits a specified number of items

take(2)

Automatically completes

Used for

- Taking a specified number of items
- Limiting unlimited Observables



RxJS Operator: take

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

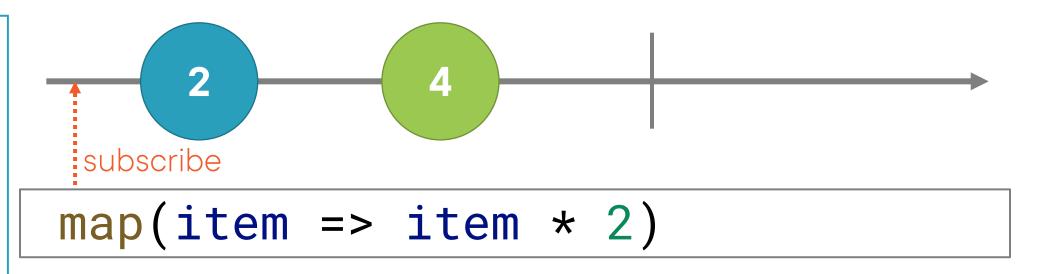
```
of(2, 4, 6)
   .pipe(
     tap(item => console.log(item)),
     map(item => item * 2),
     take(2),
     map(item => item - 3),
     tap(item => console.log(item))
   ).subscribe();
```

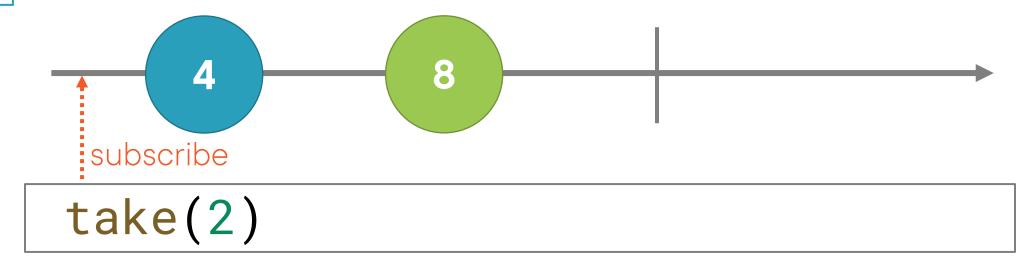
2
 4
 5

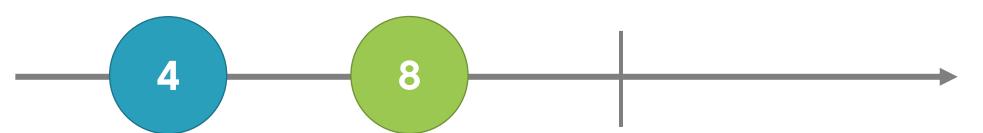


Marble Diagram: map and take

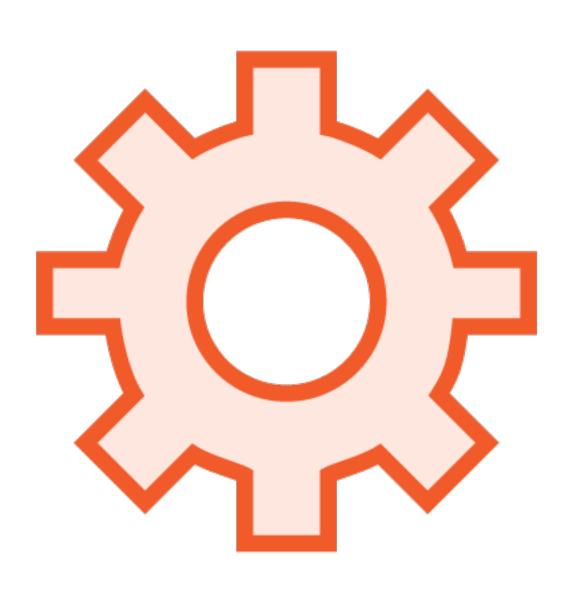
```
of(2, 4, 6)
  .pipe(
    map(item => item * 2)
    take(2)
  )
  .subscribe(x => console.log(x));
```







RxJS Operator: take



take is a filtering operator

- Subscribes to its input Observable
- Creates an output Observable

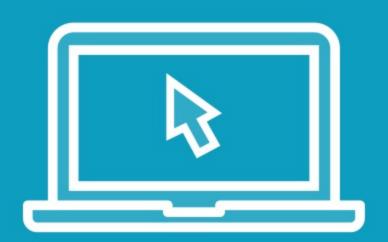
When an item is emitted

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

Only emits the defined number of items



Demo



RxJS operators:

- map
- tap
- take

map Operator Internals

```
import { Observable } from 'rxjs';
export function map(fn) {
return (input) =>
 new Observable(observer => {
   return input.subscribe({
   next: value => observer.next(fn(value)),
   error: err => observer.error(err),
   complete: () => observer.complete()
  });
```

map Operator Internals

```
import { Observable } from 'rxjs';
export function map(fn) {
 return (input) =>
 new Observable(observer => {
   return input.subscribe({
    next: value => observer.next(fn(value)),
    error: err => observer.error(err),
    complete: () => observer complete()
  });
```

- **◄**Function
- **◄Takes an input Observable**
- **◄Creates an output Observable**
- **◄Subscribes to the input Observable**
- **◆Transforms item using provided** function and emits item
- **◄Emits error notification**
- **◄ Emits complete notification**

https://github.com/ReactiveX/rxjs



RxJS Checklist: Operator Basics



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)
);
```

Each operator's output Observable is the input Observable to the following operator

RxJS Checklist: Operators



map: Transforms each emitted item

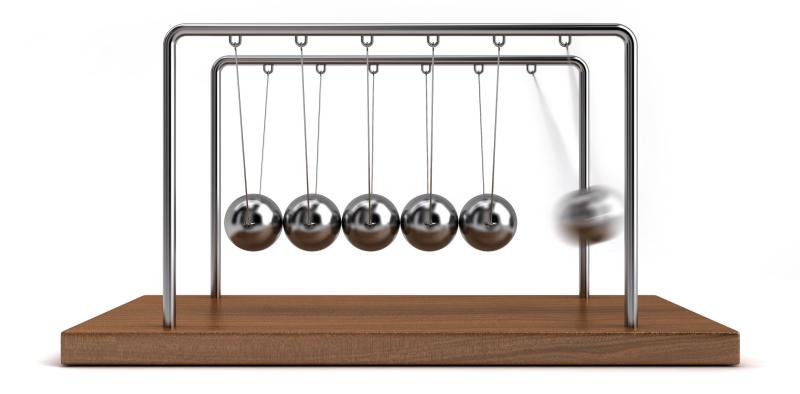
```
map(item => item * 2)
```

tap: Taps into the emitted items without modifying them

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

take: Emits the specified number of items and completes

```
take(2)
```



Coming up next...

Going Reactive