Higher-order Mapping Operators

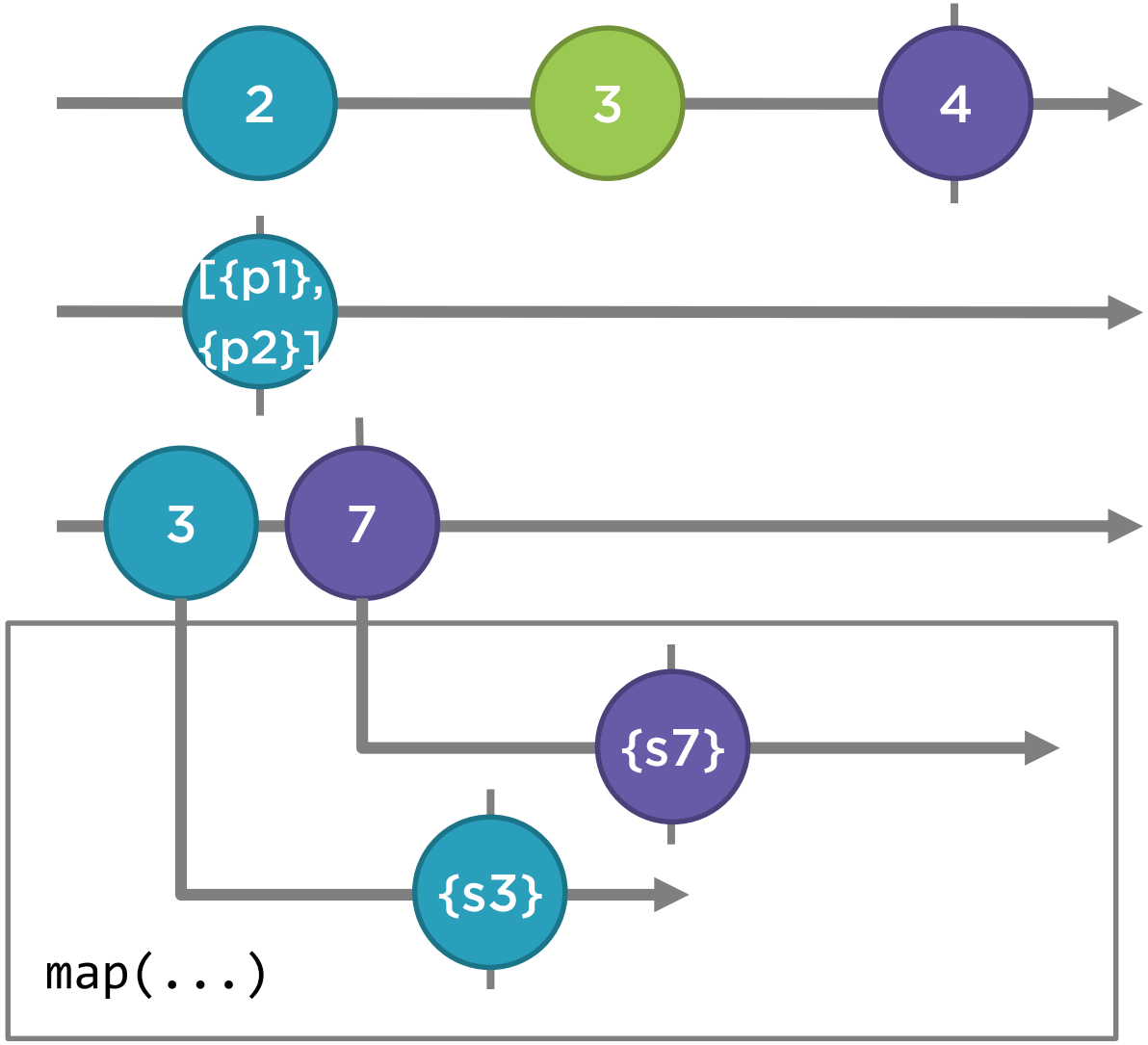
Deborah Kurata
CONSULTANT | SPEAKER | AUTHOR | MVP | GDE
@deborahkurata | blogs.msmvps.com/deborahk/

Observables

```
of(2, 3, 4)
  .subscribe();
```

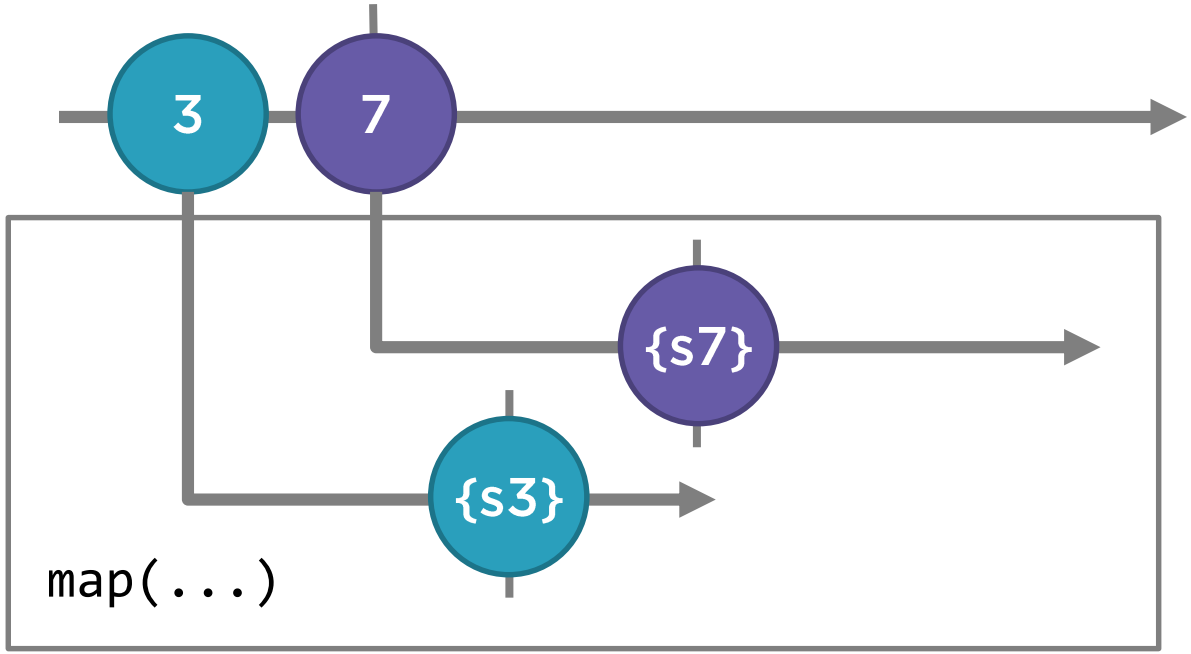
```
this.http.get<Product[]>(this.url)
  .subscribe();
```

```
of(3, 7)
 .pipe(
  map(id => this.http.get<Supplier>
              (`${this.url}/${id}`)
 )).subscribe();
```



Higher-order Observables

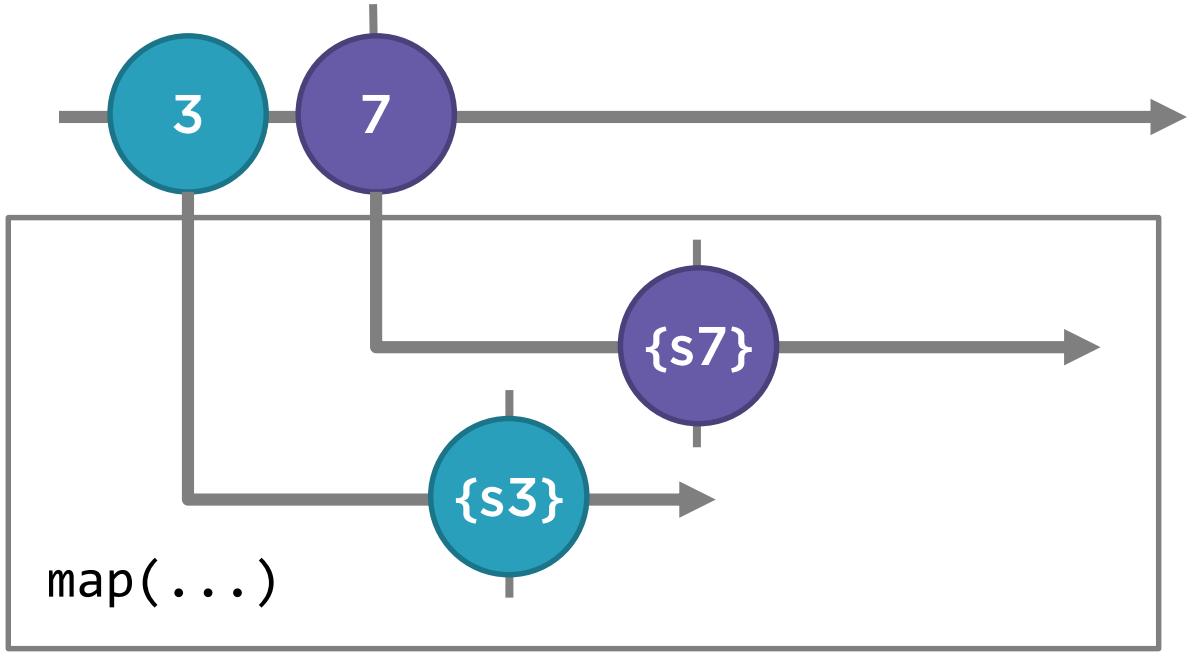
```
of(3, 7)
 .pipe(
  map(id => this.http.get<Supplier>
              (`${this.url}/${id}`)
 )).subscribe();
```



Higher-order Observables

```
of(3, 7)
 .pipe(
  map(id => this.http.get<Supplier>
              (`${this.url}/${id}`)
 )).subscribe();
```

```
of(3, 7)
 .pipe(
  map(id => this.http.get<Supplier>
              (`${this.url}/${id}`)
 )).subscribe(o =>
      o.subscribe()
 );
```

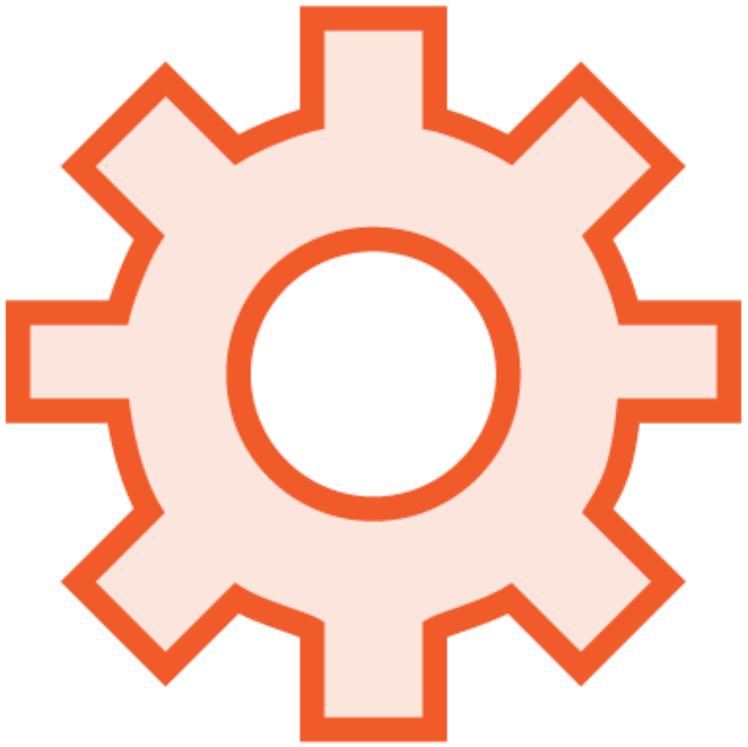


Higher-order mapping operators transform higher-order Observables.

Module Overview

Higher-order mapping operators

RxJS Features



`concatMap`

`mergeMap`

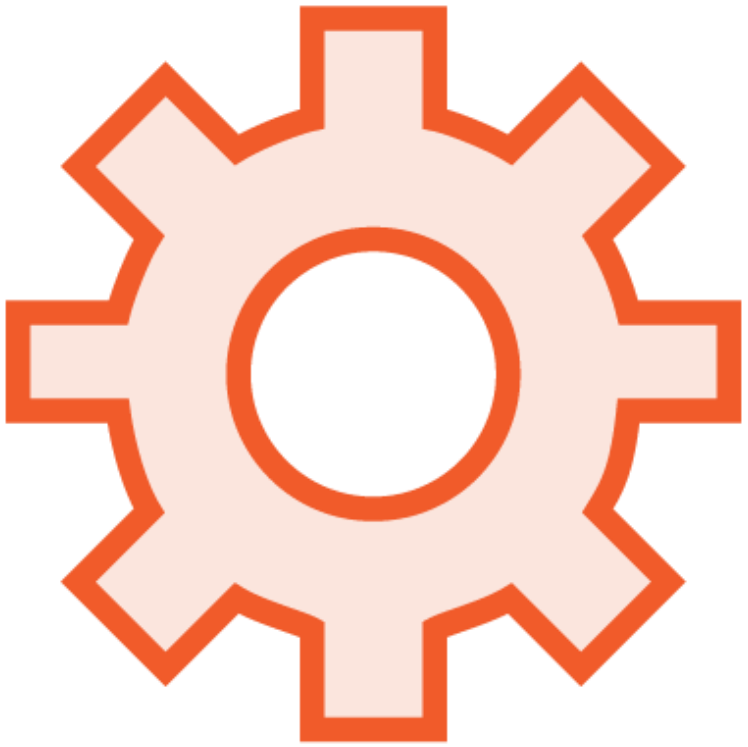
`switchMap`

Higher-order Mapping Operators

```
export interface Product {
  id: number;
  productName: string;
  productCode?: string;
  description?: string;
  price?: number;
  categoryId?: number;
  category?: string;
  supplierIds?: number[];
}
```

```
of(1, 5, 8)
 .pipe(
   map(id => this.http.get<Supplier>(`${this.url}/${id}`))
 ).subscribe(console.log);
```

Higher-order RxJS Mapping Operators



Family of operators: xxxMap()

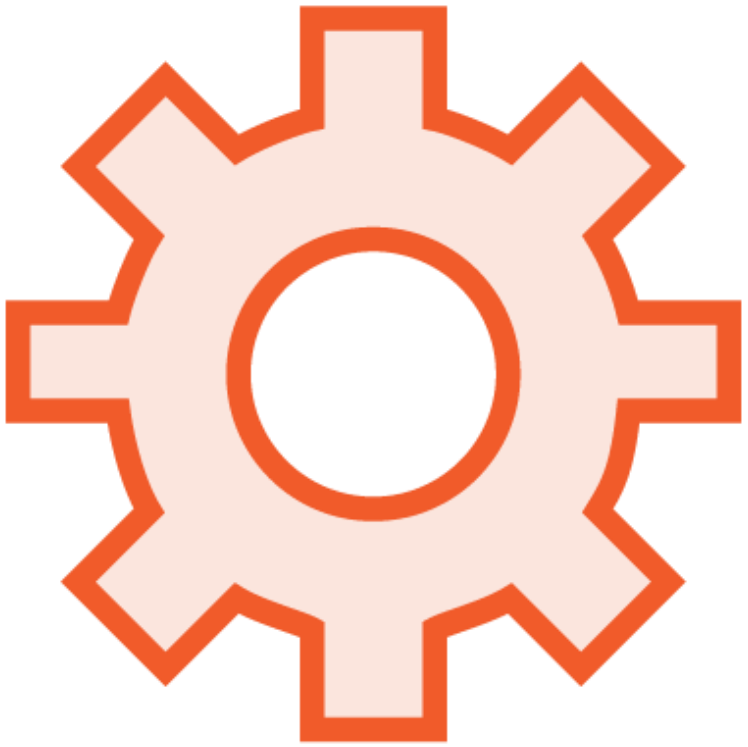
Map each value

- From a source (outer) Observable
- To a new (inner) Observable

Automatically subscribe/unsubscribe from inner Observables

Emit the resulting values to the output Observable

Higher-order RxJS Mapping Operators

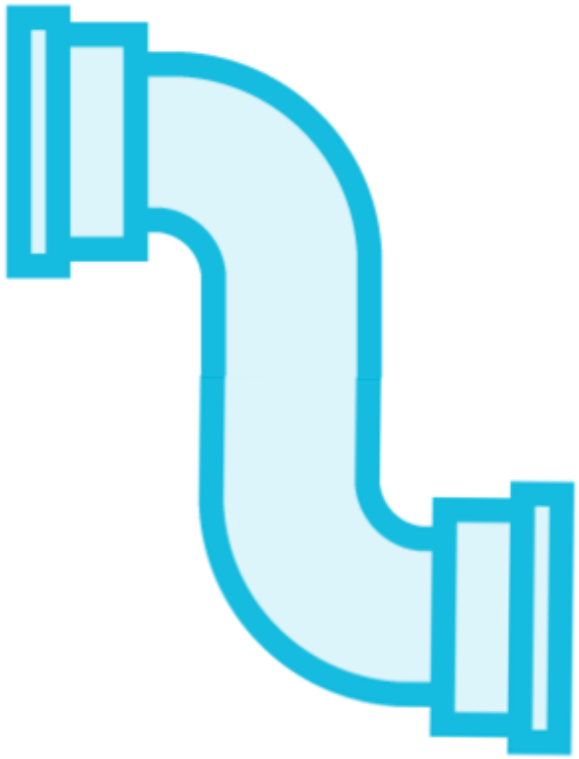


`concatMap`

`mergeMap`

`switchMap`

RxJS Operator: `concatMap`



Higher-order mapping + concatenation

Transforms each emitted item to a new (inner) Observable as defined by a function

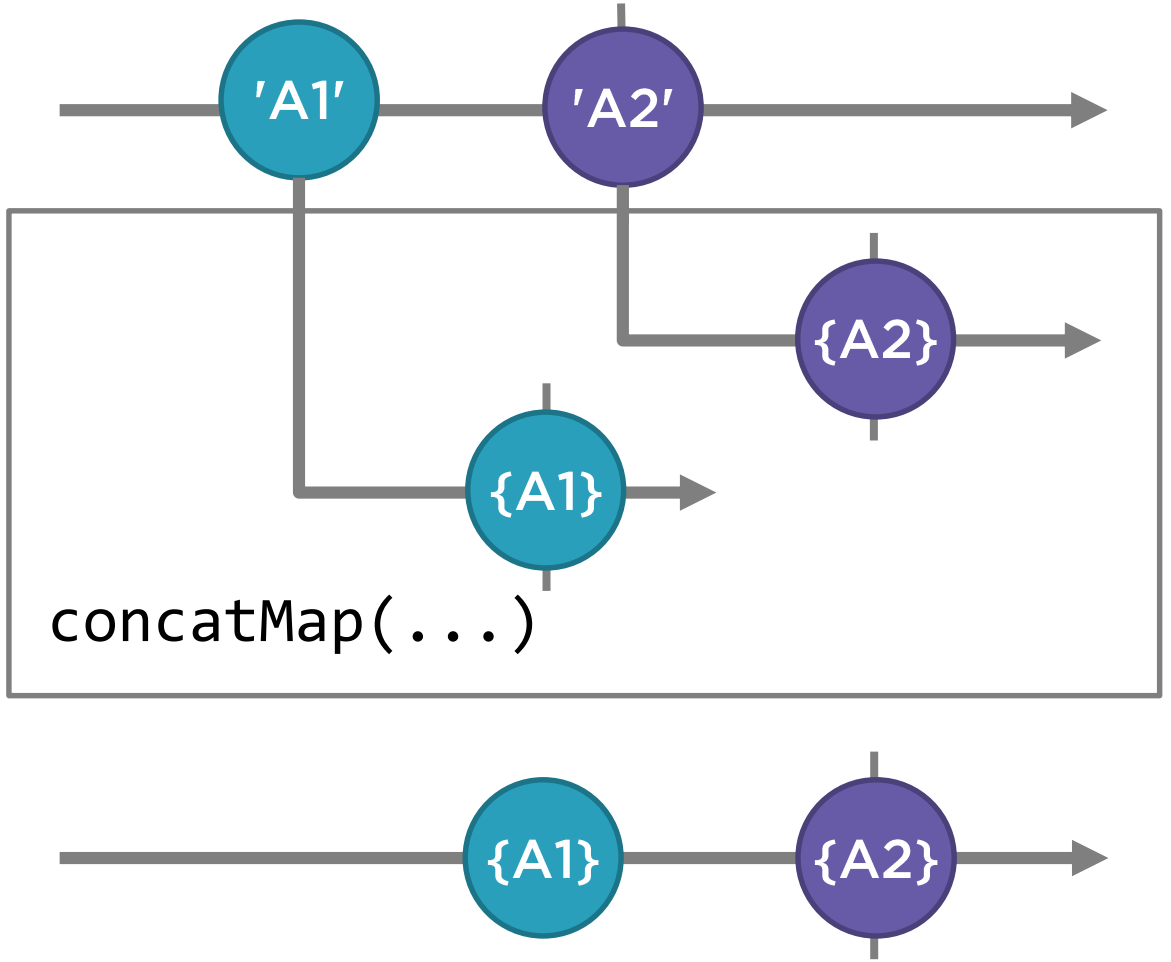
```
concatMap(i => of(i))
```

It waits for each inner Observable to complete before processing the next one

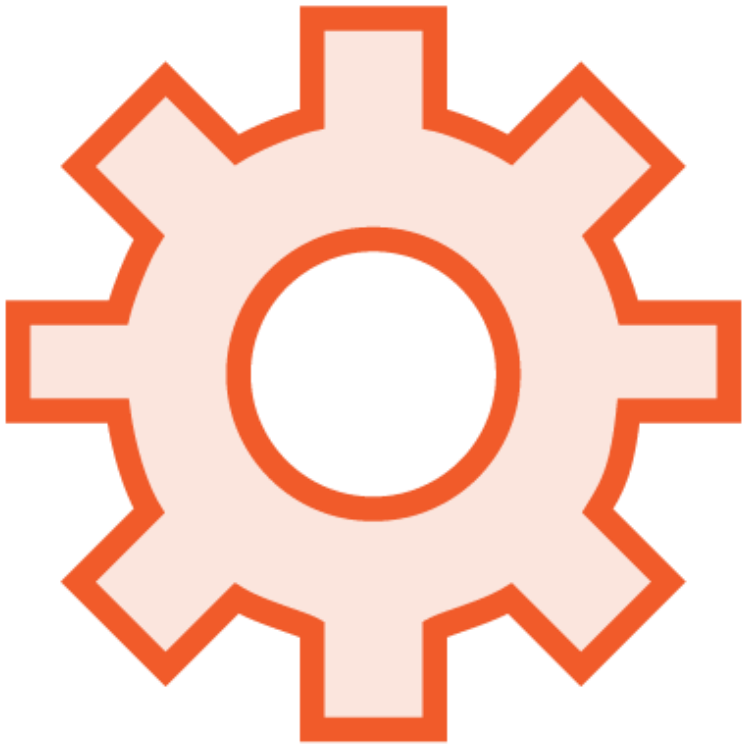
Concatenates their results in sequence

Marble Diagram: concatMap

```
of('A1', 'A2')
 .pipe(
  concatMap(id =>
 this.http.get<Apple>(`${this.url}/${id}`))
 ).subscribe(console.log);
```



RxJS Operator: concatMap



concatMap is a transformation operator

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

When an item is emitted, it's queued

- Item is mapped to a inner Observable as specified by a provided function
- Subscribes to inner Observable
- Waits!
- Inner Observable emissions are concatenated to the output stream
- When inner Observable completes, processes the next item

Use concatMap

To wait for the prior Observable to complete before starting the next one

To process items in sequence

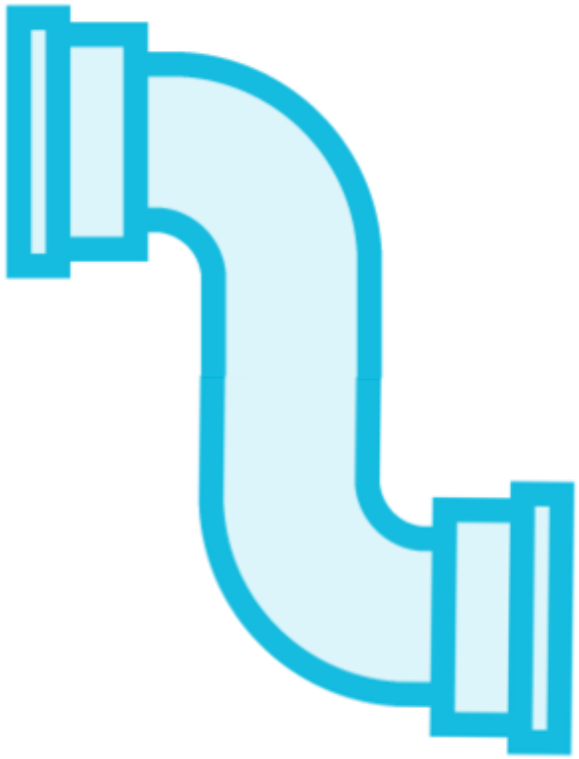
Examples:

- From a set of ids, get data in sequence
- From a set of ids, update data in sequence

Demo

`concatMap`

RxJS Operator: `mergeMap`



Higher-order mapping + merging

Transforms each emitted item to a new (inner) Observable as defined by a function

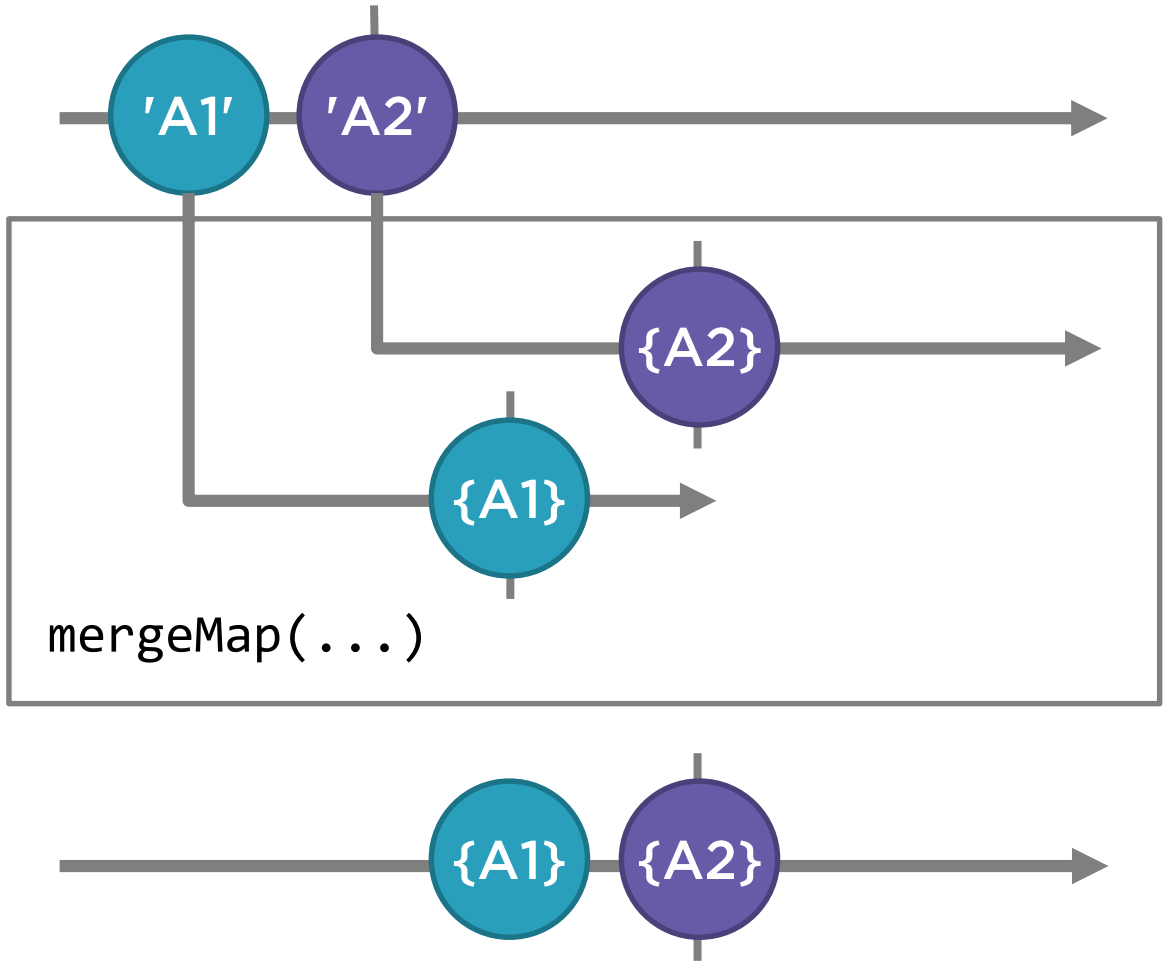
```
mergeMap(i => of(i))
```

It executes inner Observables in parallel

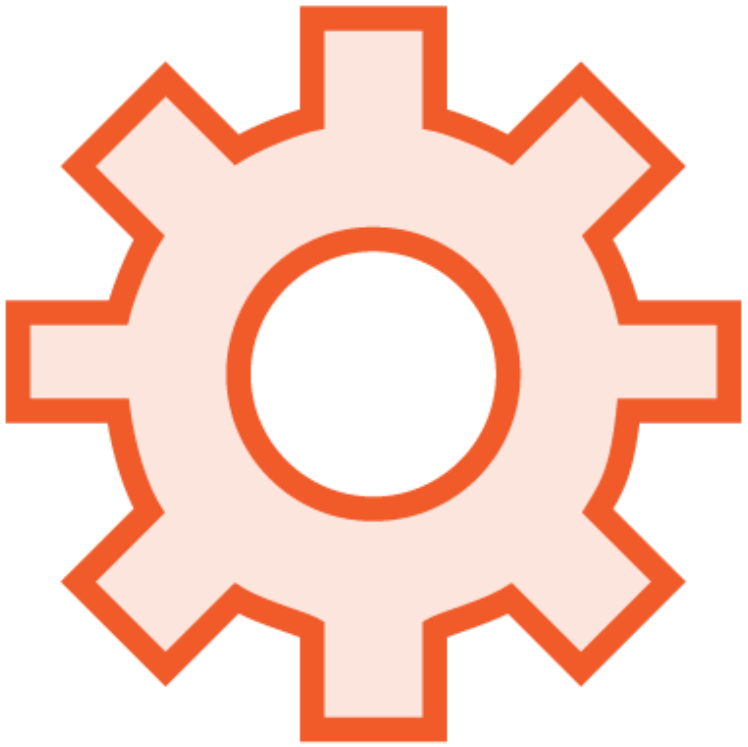
And merges their results

Marble Diagram: `mergeMap`

```
of('A1', 'A2')
 .pipe(
  mergeMap(id =>
 this.http.get<Apple>(`${this.url}/${id}`))
 ).subscribe(console.log);
```



RxJS Operator: `mergeMap (flatMap)`



`mergeMap` **is a transformation operator**

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

When each item is emitted

- Item is mapped to a inner Observable as specified by a provided function
- Subscribes to inner Observable
- Inner Observable emissions are merged to the output stream

Use mergeMap

To process in parallel

When order doesn't matter

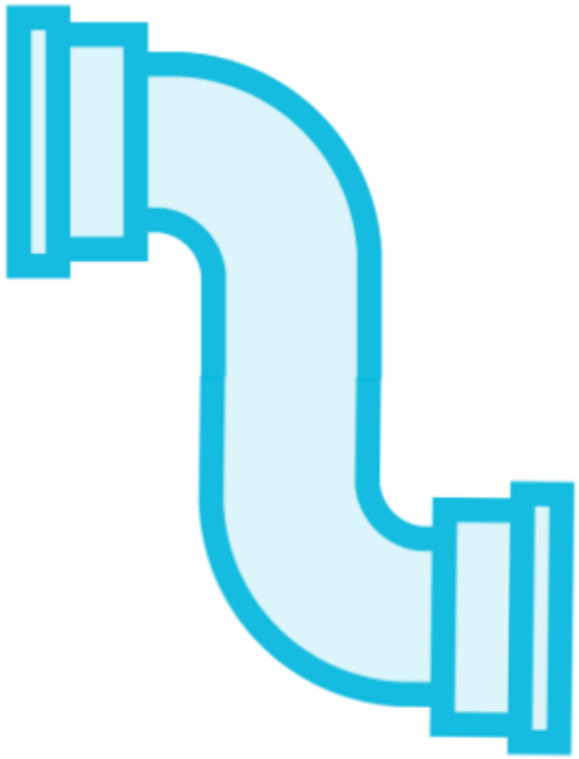
Examples:

- From a set of ids, retrieve data (order doesn't matter)

Demo

`mergeMap`

RxJS Operator: `switchMap`



Higher-order mapping + switching

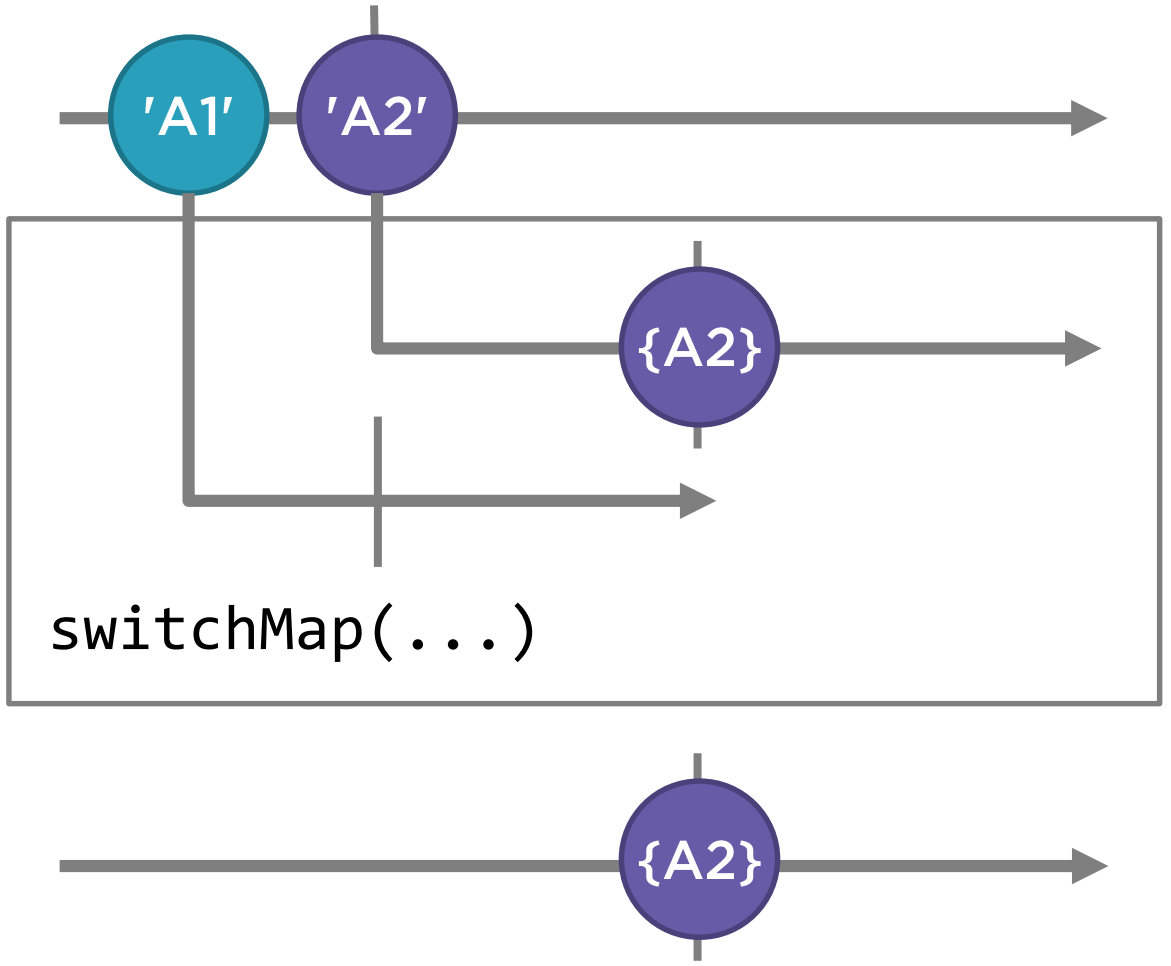
Transforms each emitted item to a new (inner) Observable as defined by a function

```
switchMap(i => of(i))
```

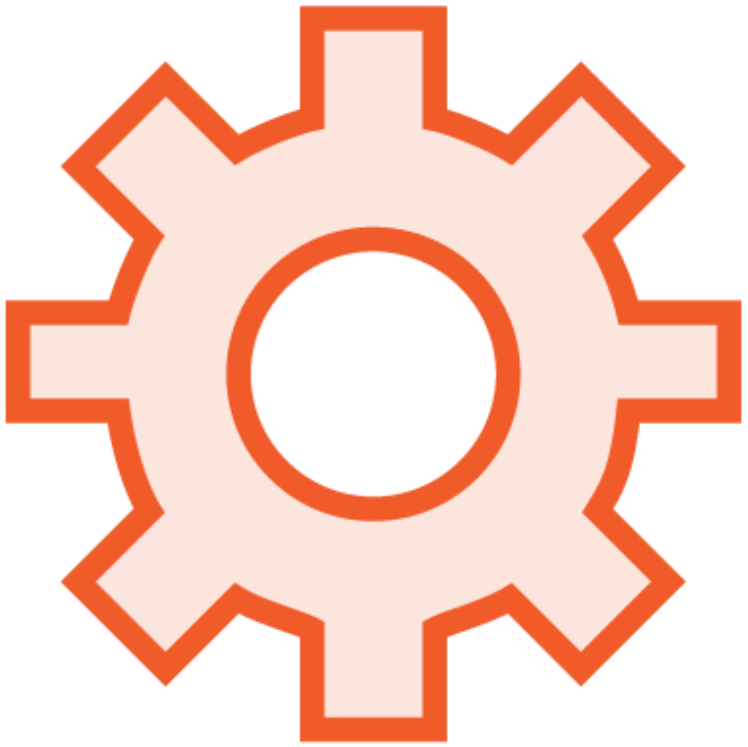
Unsubscribes the prior inner Observable and switches to the new inner Observable

Marble Diagram: `switchMap`

```
of('A1', 'A2')
 .pipe(
  switchMap(id =>
 this.http.get<Apple>(`${this.url}/${id}`))
 ).subscribe(console.log);
```



RxJS Operator: switchMap



switchMap is a transformation operator

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

When each item is emitted

- Item is mapped to an inner Observable as specified by a provided function
- Unsubscribes from prior inner Observable
- Subscribes to new inner Observable
- Inner Observable emissions are merged to the output stream

Use `switchMap`

To stop any prior Observable before switching to the next one

Examples:

- Type ahead or auto completion
- User selection from a list

Demo

`switchMap`

Higher-Order Observable

Source/outer Observable

Inner Observable

```
of('A1', 'A2')
 .pipe(
   mergeMap(id => this.http.get<Apple>(`${this.url}/${id}`))
 );
```

Higher-order mapping operator

Item emitted from outer Observable

{A1} {A2}

Higher-Order Mapping

Use higher-order mapping operators

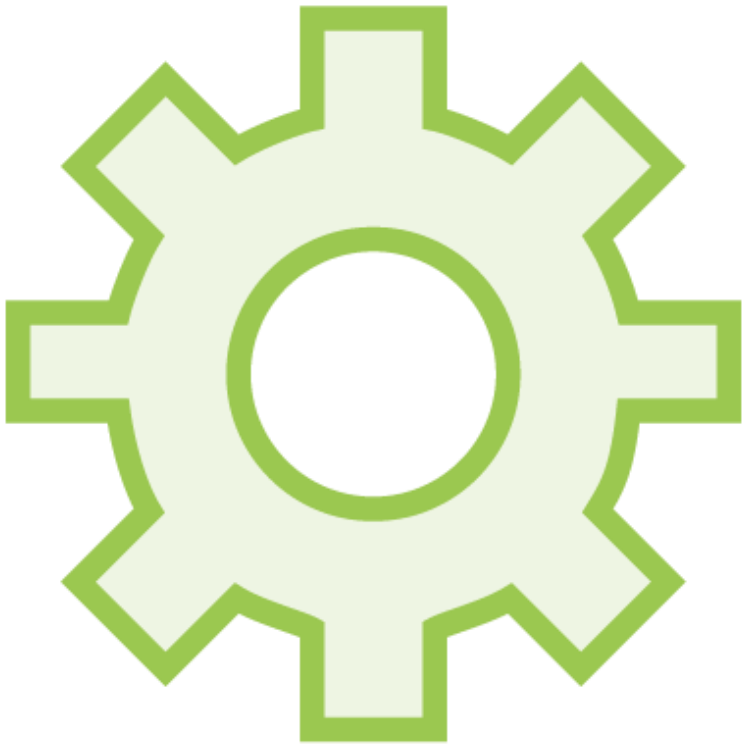
- To map emitted items to a new Observable
- Automatically subscribe to and unsubscribe from that Observable
- And emit the results to the output stream

Higher-order mapping operator functions

- Take in an item and return an Observable

Use instead of nested subscribes

Higher-Order Mapping Operators



`concatMap`

- Waits for inner Observable to complete before processing the next one

`mergeMap`

- Processes inner Observables in parallel

`switchMap`

- Unsubscribes from the prior inner Observable and switches to the new one