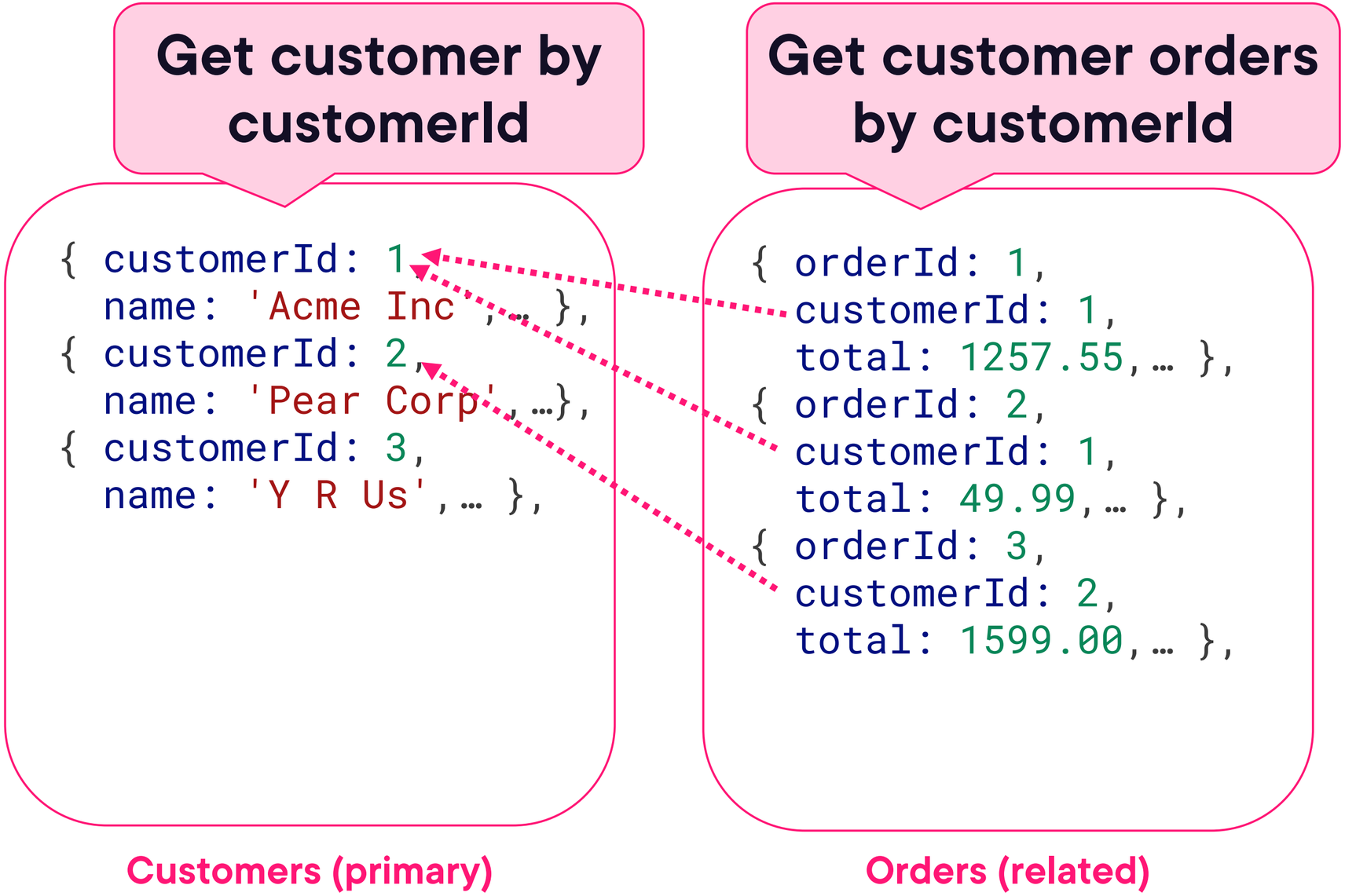
Getting Related Data: `switchMap`, `concatMap` and `mergeMap`

Deborah Kurata
Developer

https://www.youtube.com/@deborah_kurata

Get Each by Id



Get One, Use It to Get the Related Data

Get product by productId

Check if it hasReviews

If so, get product reviews by productId

```
{ productId: 1,
  name: 'Hammer',
  hasReviews: true,…},
{ productId: 2,
  name: 'Saw',
  hasReviews: true,…},
{ productId: 3,
  name: 'Rake',
  hasReviews: false,…},
...
```

Products (primary)

```
{ reviewId: 1,
  productId: 1,
  title: 'Works fine',…},
{ reviewId: 2,
  productId: 1,
  title: 'Dangerous',…},
{ reviewId: 3,
  productId: 2,
  title: 'Great!',…},
...
```

Product Reviews (related)

Get One, Use Its Array to Get the Related Data

Get movie by movieId

Read its cast array (actorIds)

For each actorId, get the actor by id

```
{ movieId: 1,
  name: 'Thor',
  cast: [actorId,… ]},
{ movieId: 2,
  name: 'LOTR',
  cast: [actorId,… ]},
{ movieId: 3,
  name: 'Up',
  cast: [actorId,… ]},
...
```

```
{ actorId: 1,
  name: 'Joe',
  roles: [movieId,… ]},
{ actorId: 2,
  name: 'Jess',
  roles: [movieId,… ]},
{ actorId: 3,
  name: 'Krysta',
  roles: [movieId,… ]},
...
```

https://youtu.be/vtCDRiG_D4?t=2190

Overview

Try getting related data using the `map` operator

Examine higher-order mapping operators:

- `concatMap`
- `mergeMap`
- `switchMap`

Apply a higher-order operator and retrieve related data

Demo

Retrieve related data

- Get one, use it to get the related data

Use editor hover feature to help debug data typing issues


```
getProduct(id: number): Observable<Product> {
  const productUrl = this.productsUrl + '/' + id;
  return this.http.get<Product>(productUrl)
    .pipe(
      tap(() => console.log('In http get by id pipeline')),
      map( (parameter) x: Observable<Product>  s(product)),
      tap(x => console.log(x)),
      catchError(err => this.handleError(err))
    );
}
```

Why?

Tooltips display the data type of any variable or method

Higher-order Observable

Emits an
Observable<Review>

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

Observable<Review>

Higher-order Observable

Outer observable

Inner observable

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

Higher-order Observable

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

Observable<Review>

Review

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

Don't nest subscribes

Key Point

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

Why?

- No easy way to unsubscribe
- Features can't auto subscribe
- Timing issues

Higher-order Mapping Operators

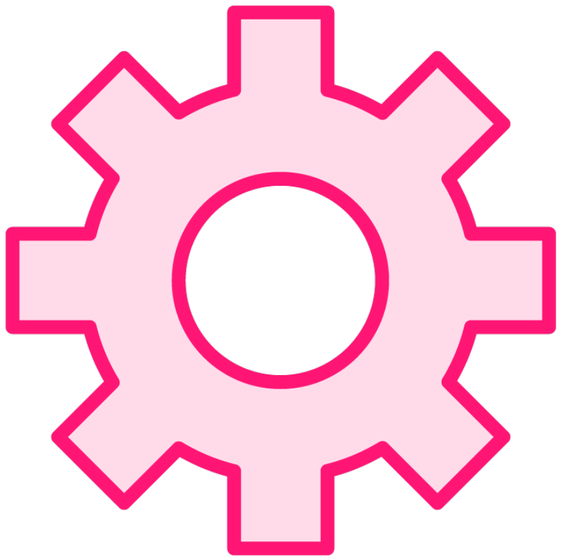
Transform an emitted item to a new observable

Automatically subscribe and unsubscribe

Unwrap (flatten) the result

```
- Observable<Product> -> Product
```

Higher-order Mapping Operators

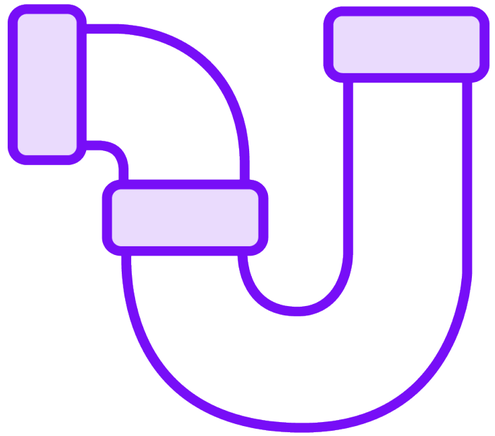


`concatMap`

`mergeMap`

`switchMap`

RxJS Operator: concatMap



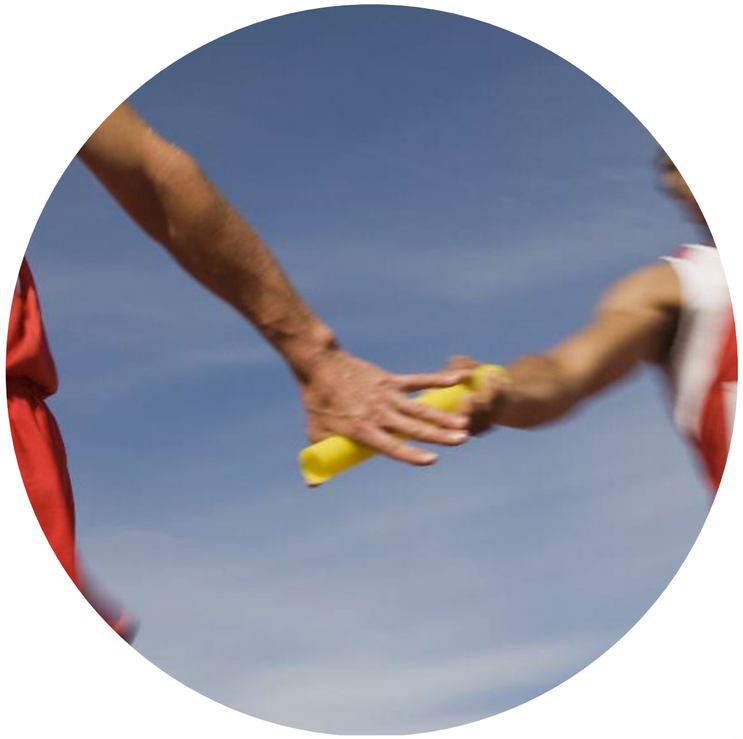
Transforms each emitted item to a new (inner) observable as defined by a function

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

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

Concatenates their results in sequence

`concatMap` -> Relay Race



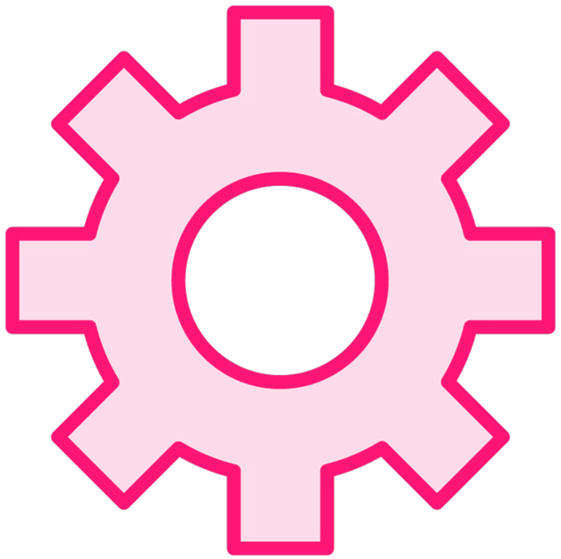
Runners are queued

Only one runner runs at a time

A runner must complete before the next runner can execute

Runners retain their order

RxJS Operator: `concatMap`

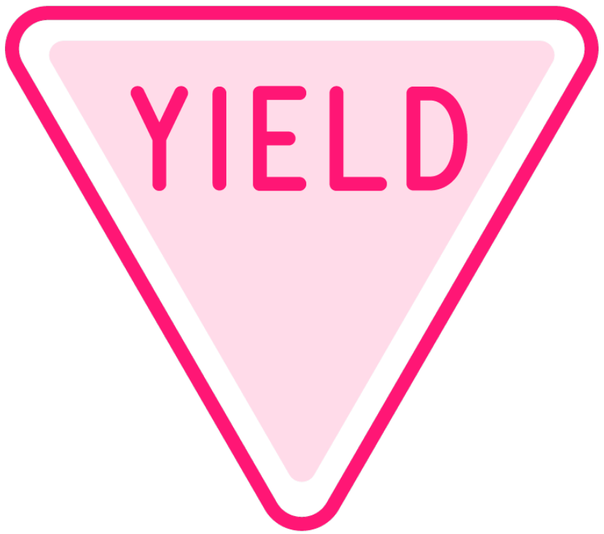


`concatMap` **is a transformation operator**

When an item is emitted, it's queued

- Item is mapped to an inner observable as specified by the provided function
- Subscribes to the inner observable
- Waits!
- Inner observable emissions are concatenated to the output observable
- When the 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

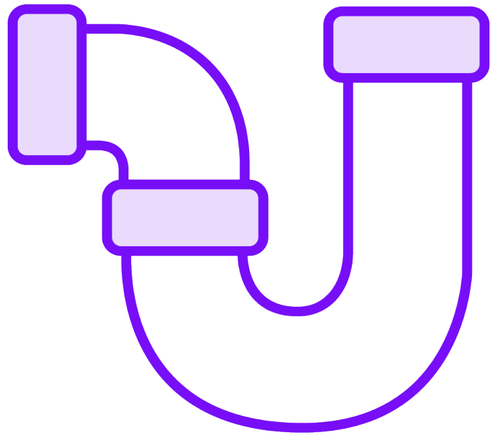
Examples:

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

Demo

`concatMap`

RxJS Operator: `mergeMap`



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

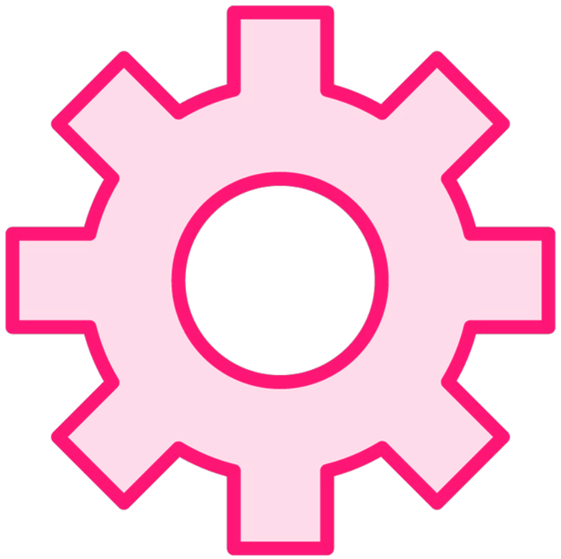
`mergeMap` -> 800 Meter Race

Runners start concurrently

They all merge into the lower lanes

The runners complete based on how quickly they finish

RxJS Operator: `mergeMap`

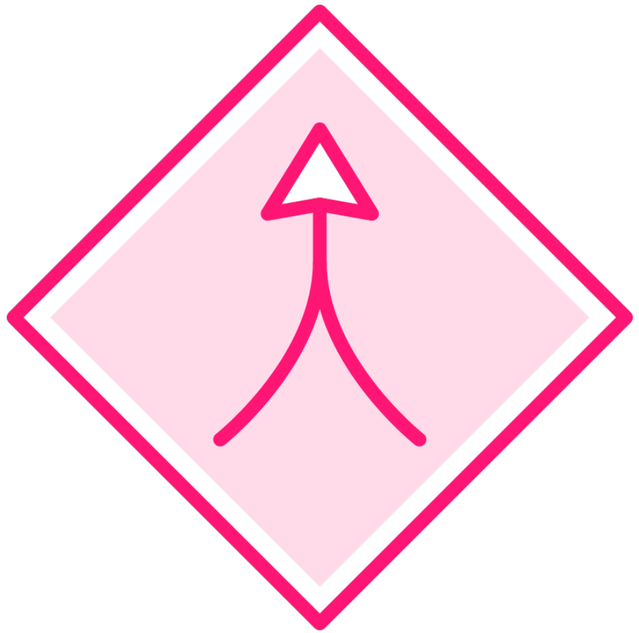


`mergeMap` **is a transformation operator**

When each item is emitted

- Item is mapped to an inner observable as specified by the provided function
- Subscribes to the inner observable
- Inner observable emissions are merged to the output observable in the order they finish

Use `mergeMap`



To process items in parallel

When order doesn't matter

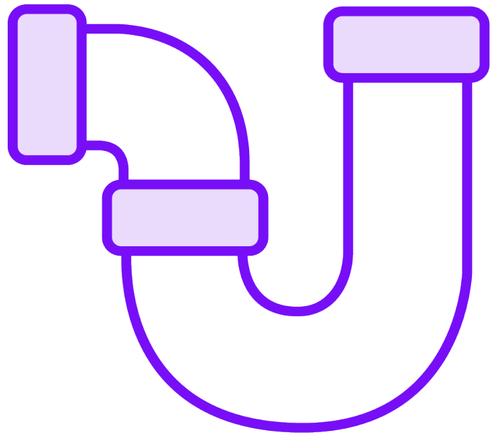
Examples:

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

Demo

`mergeMap`

RxJS Operator: `switchMap`



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

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

It unsubscribes from the prior inner observable

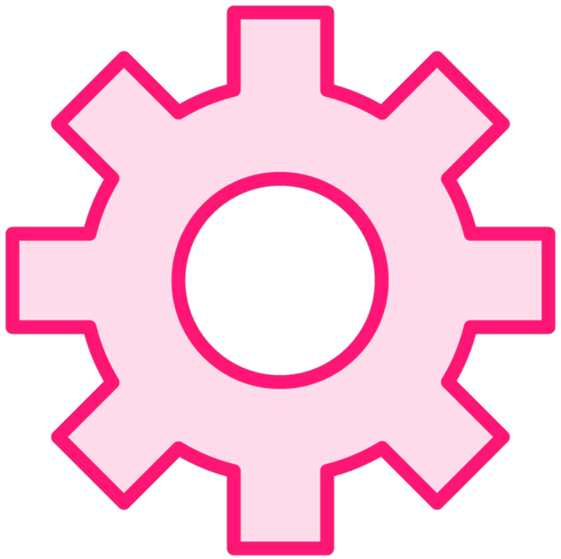
And switches to the new inner observable

`switchMap` -> Changing Who's Running

The coach changes their mind as to which runner will run

Only one runner runs

RxJS Operator: `switchMap`



`switchMap` **is a transformation operator**

When each item is emitted

- Item is mapped to an inner observable as specified by the provided function
- Switches to this inner observable
  - Unsubscribes from any prior inner observable
  - Subscribes to the new inner observable

Use `switchMap`

To stop (cancel) any prior observable before switching to the next one

Examples:

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

Demo

`switchMap`

Demo

Getting related data

- Get one, use it to get the related data

Key Point

Minimize the amount of code in your pipelines

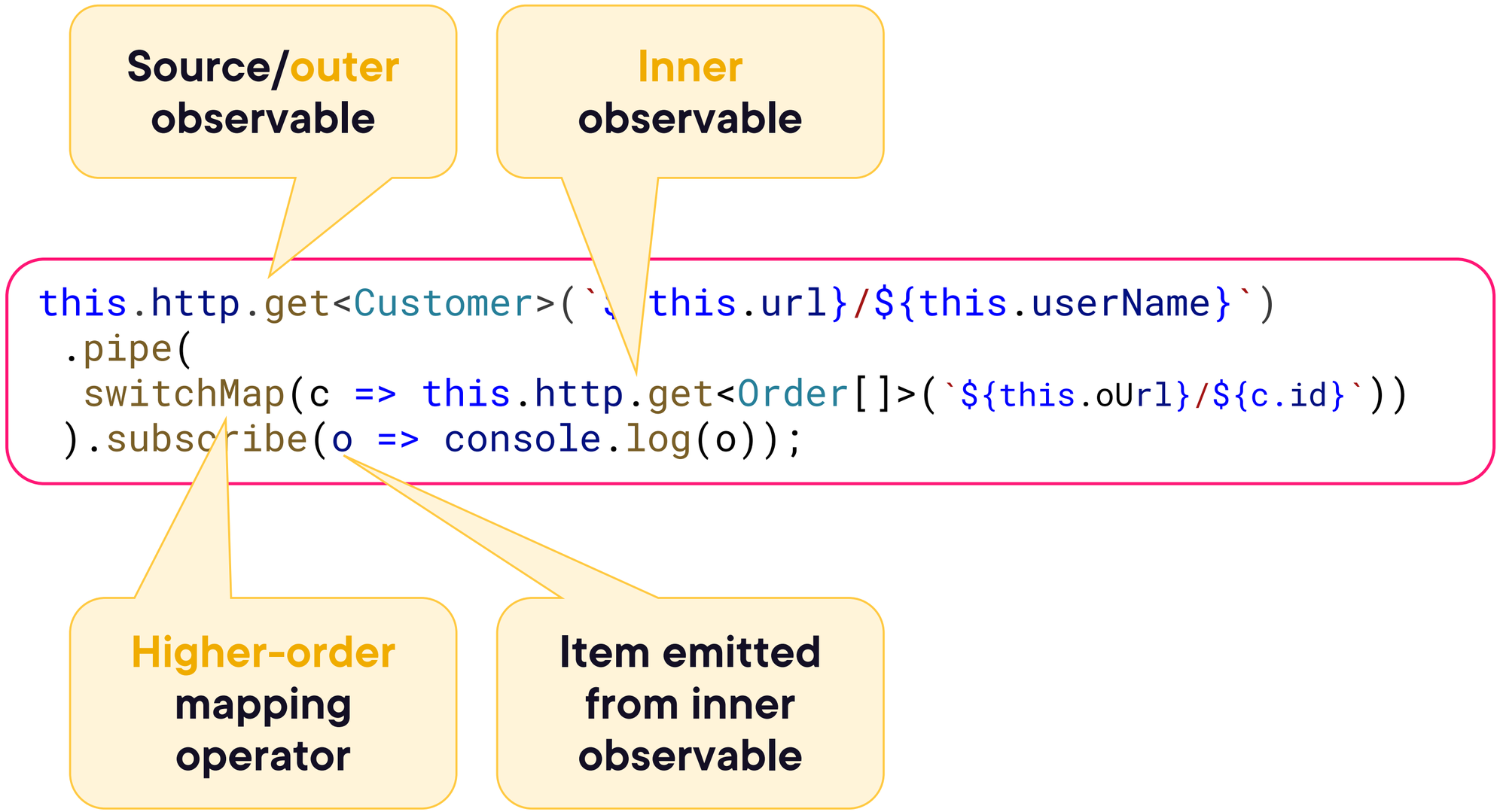
Why?

Easier to read, debug and test
Improves performance

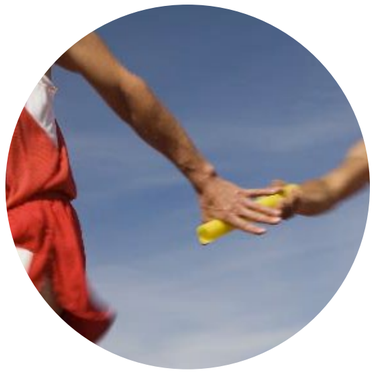
How?

Break up into separate methods and call the methods in the pipeline

Higher-order Mapping Operators



Higher-order Mapping Operators



`concatMap`

- **Waits** for each inner observable to complete before processing the next one

`mergeMap`

- Processes inner observables in **parallel** and **merges** the result

`switchMap`

- **Unsubscribes** from the prior inner observable and **switches** to the new one

range and delay

range(first, count)

- Emits integers
- Starts with the number in the first argument
- Continues for the count specified in the second argument

```
range(1, 5).subscribe(i => console.log(i));
```

delay(amount)

- Delays the emission from the source observable
- By the amount of time specified, in milliseconds

```
range(1, 5) .pipe(
    concatMap(i => of(i).pipe(
      delay(1000)
    ))
  ).subscribe((i) => console.log(i));
```

Retrieving Related Data

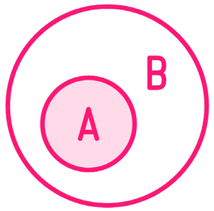
To retrieve data and use that data to retrieve related data:

- Use a higher-order mapping operator
- Automatically subscribe and unsubscribe
- Flatten the result: Emits a `Product` not `Observable<Product>`

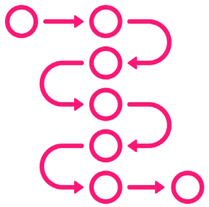
```
getProduct(id: number): Observable<Product> {
  const productUrl = this.productsUrl + '/' + id;
  return this.http.get<Product>(productUrl)
    .pipe(
      switchMap(product => this.getWithReviews(product)),
      catchError(err => this.handleError(err))
    );
}
```

Best Practices

Leverage your tools
Use the hover feature of the editor to help debug data typing or other issues



Don't nest subscribes



Minimize the amount of code in your pipelines

For More Information

StackBlitz code

- https://stackblitz.com/edit/rxjs-signals-m7-deborahk

Demo code

- https://github.com/DeborahK/angular-rxjs-signals-fundamentals

RxJS documentation

- https://rxjs.dev/guide/higher-order-observables

"Why You Shouldn't Nest Subscribes"

- https://medium.com/ngconf/why-you-shouldnt-nest-subscribes-eafbc3b00af2

"switchMap vs concatMap vs mergeMap ... Oh My!"

- https://youtu.be/RSf7DlJXoGQ

"RxJS in Angular: Terms, Tips, and Patterns"

- https://youtu.be/vtCDRiG_D4?t=2190

Up Next:

Using a Declarative Approach