Higher-order Mapping Operators

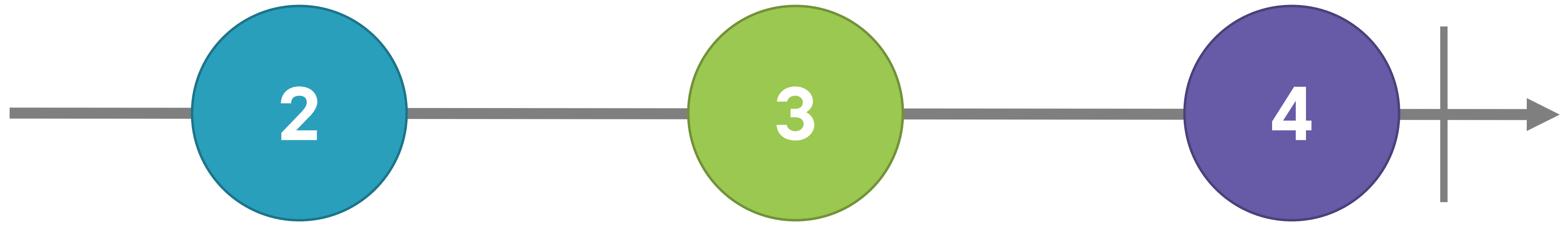
Deborah Kurata
Consultant | Speaker | Author | MVP | GDE

@deborahkurata

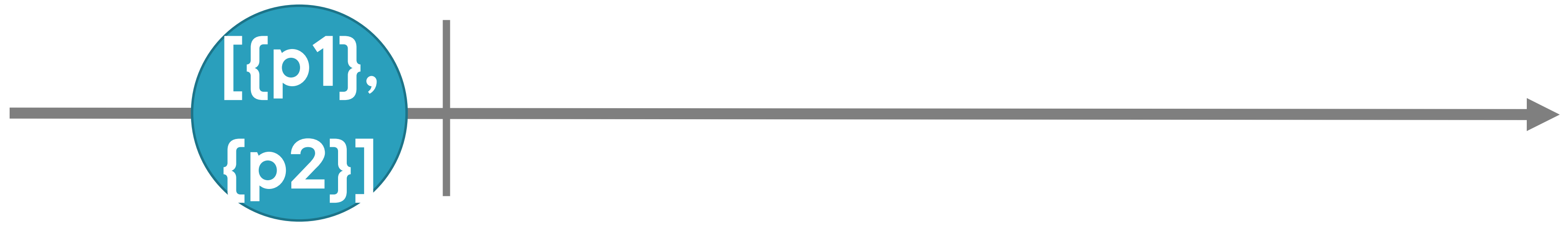
An Observable can emit another Observable

Observables

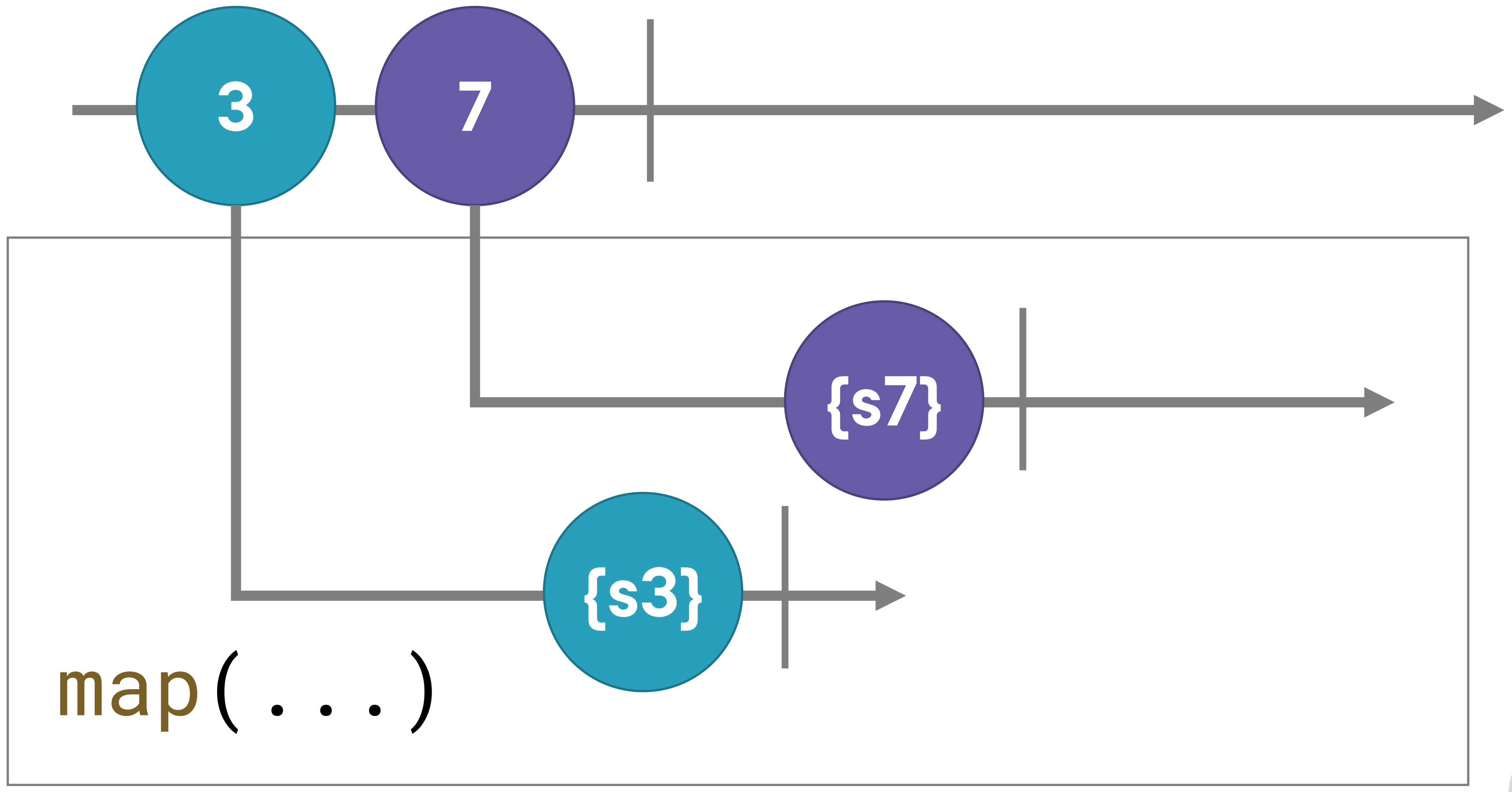
```
of(2, 3, 4)
  .subscribe();
```



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

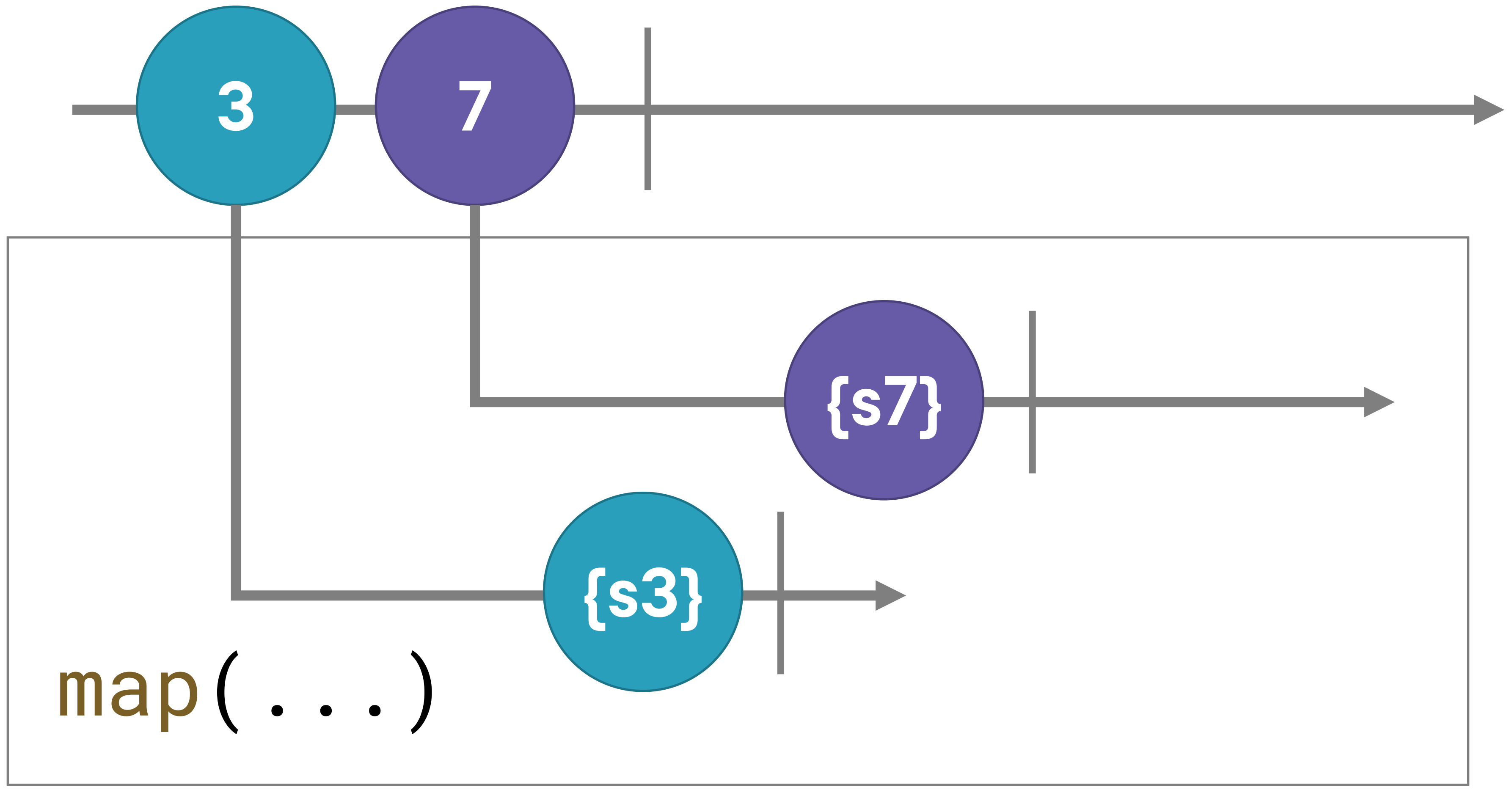


```
of(3, 7)
 .pipe(
  map(id => this.http.get<Supplier>
             (`${this.url}/${id}`)
 )).subscribe();
```



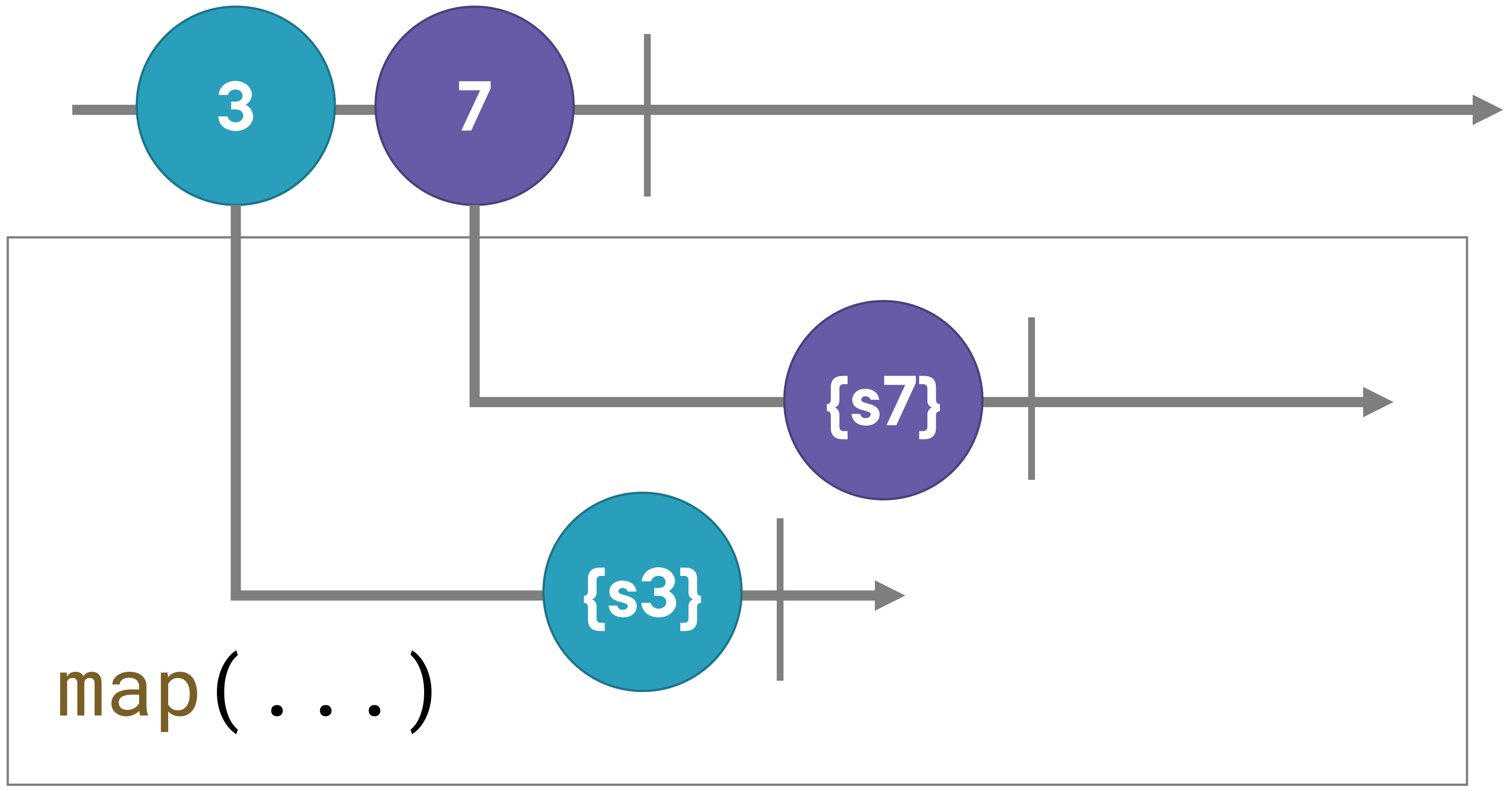
Higher-order Observable

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



Higher-order Observable

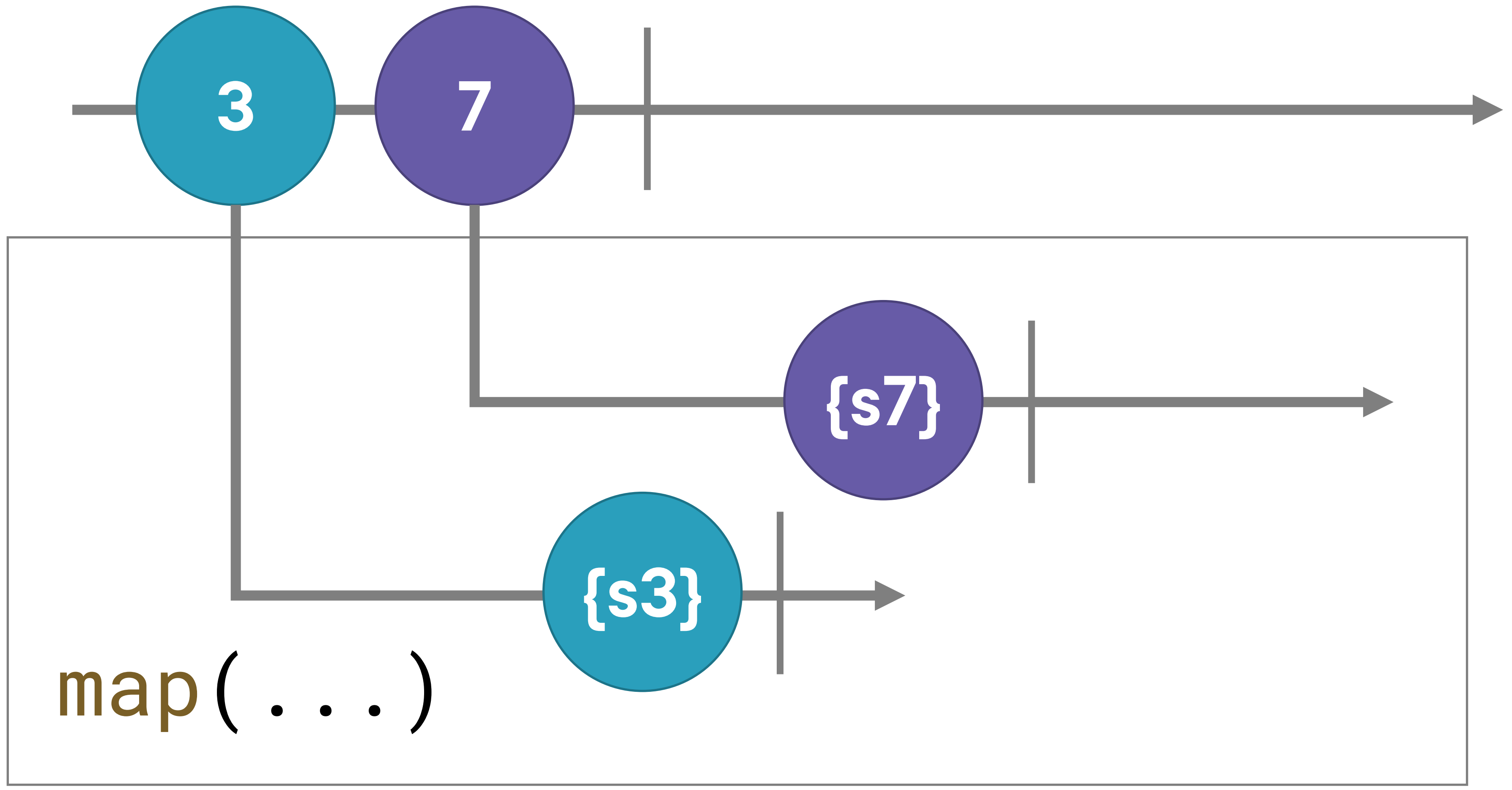
```
of(3, 7)
 .pipe(
  map(id => this.http.get<Supplier>
                 (`${this.url}/${id}`)
 )).subscribe(o => o.subscribe());
```



Higher-order Observable

```
Observable<Observable<Supplier>>
```

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



Higher-order mapping operators flatten higher-order Observables.

`Observable<Observable<T>>` to `Observable<T>`

Module Overview

Higher-order mapping operators

RxJS Features

`concatMap`

`mergeMap`

`switchMap`

Mapping to an Observable

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

Higher-order Mapping Operators

Family of operators: `xxxMap()`

Map each value

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

Automatically subscribe to/unsubscribe from inner Observables

Flatten the result

Emit the resulting values to the output Observable

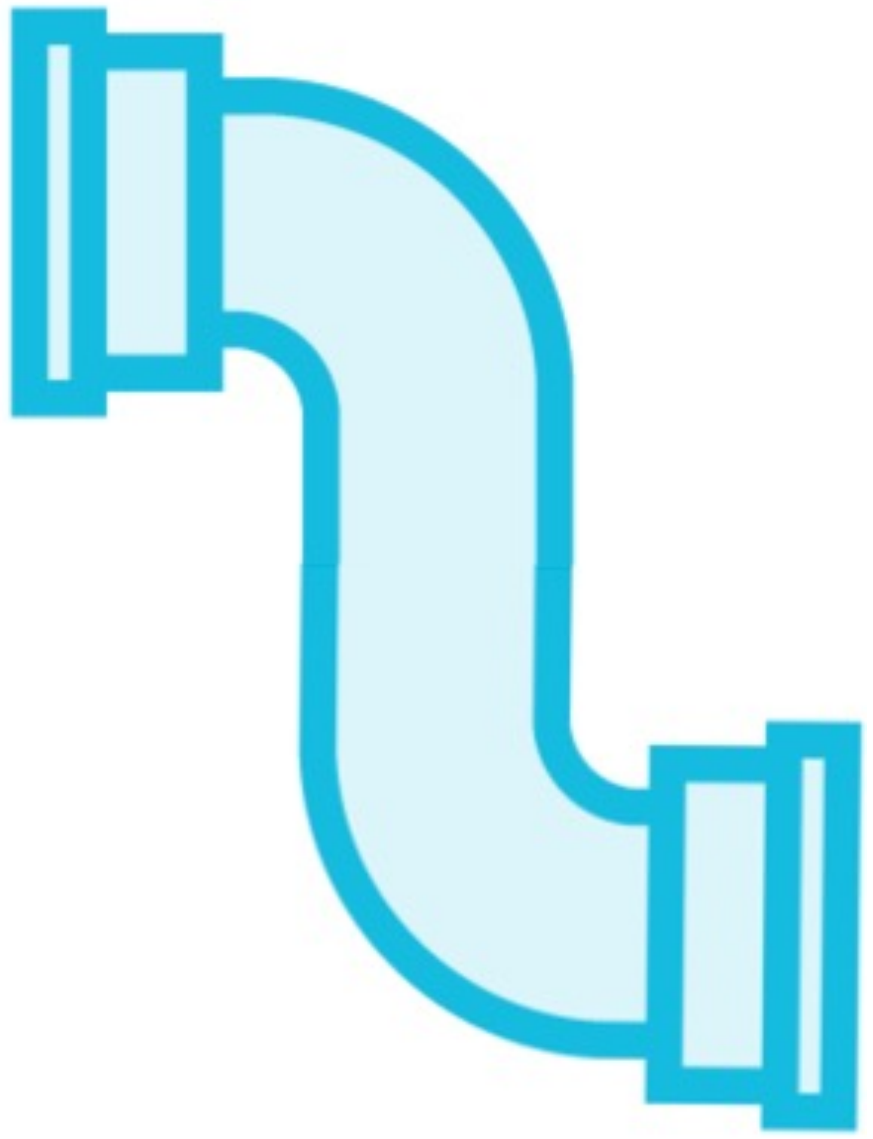
Higher-order RxJS Mapping Operators

`concatMap`

`mergeMap`

`switchMap`

RxJS Operator: `concatMap`



Higher-order mapping + concatenation

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

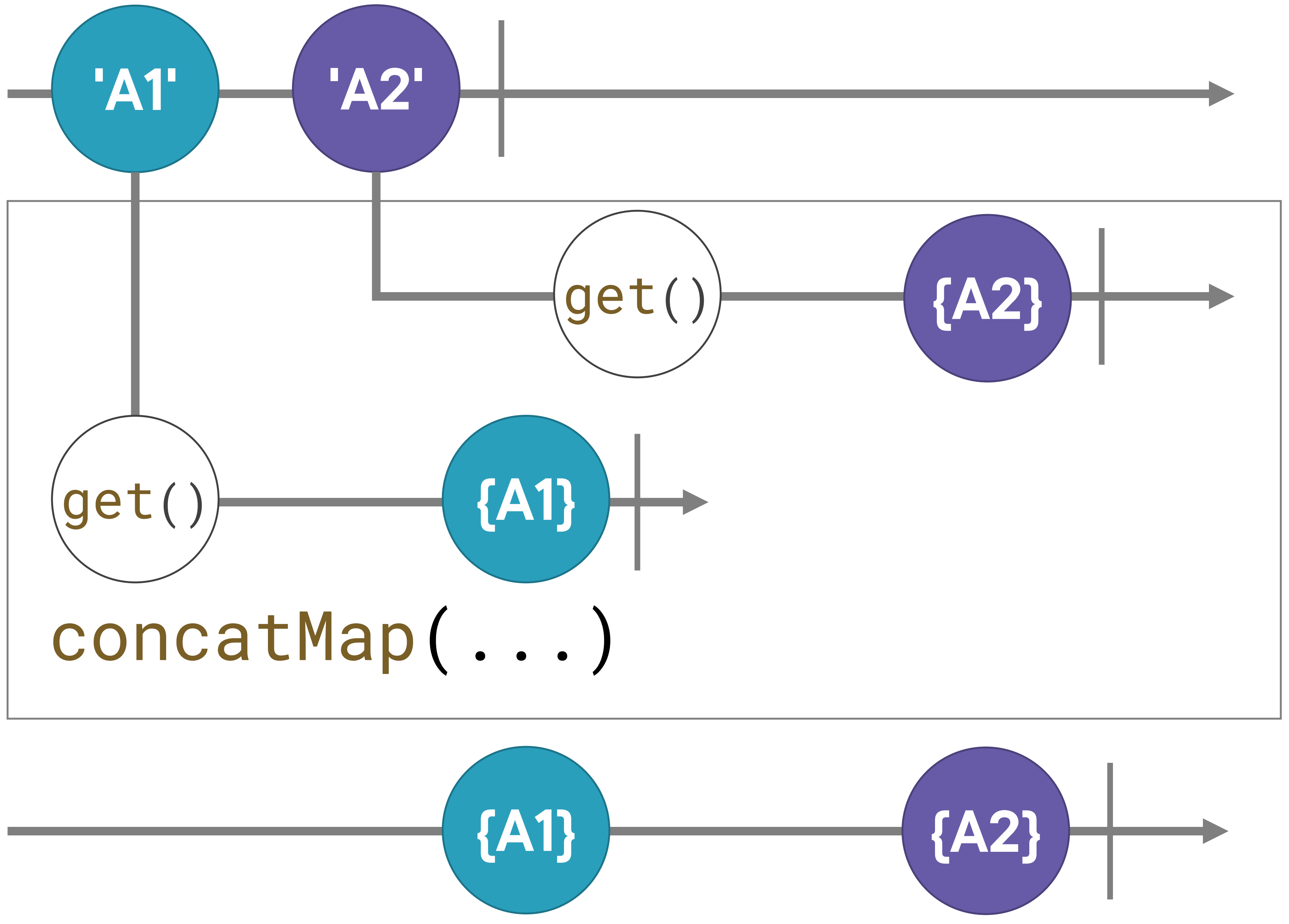
```
concatMap(i => of(i))
```

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



`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

- Subscribes to its input Observable
- Creates an output Observable

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

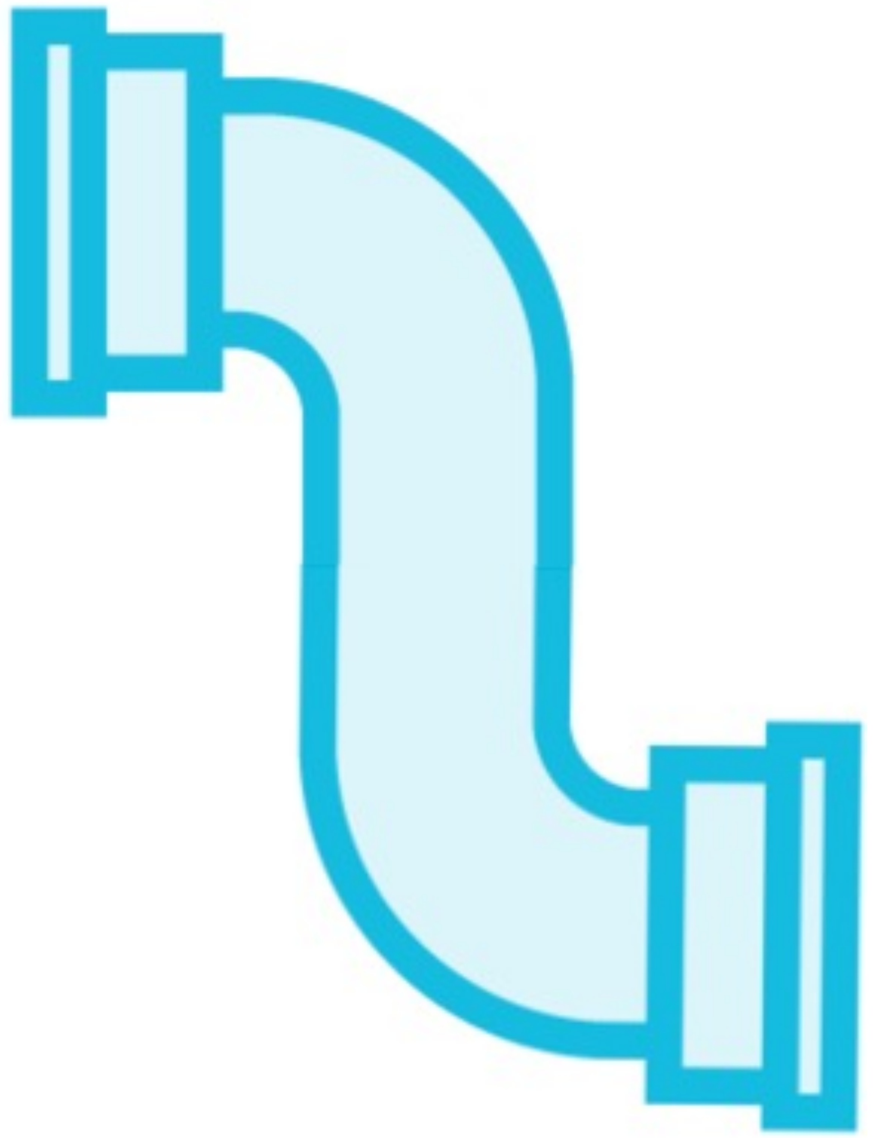
Examples:

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

Demo

`concatMap`

RxJS Operator: `mergeMap`



Higher-order mapping + merging

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

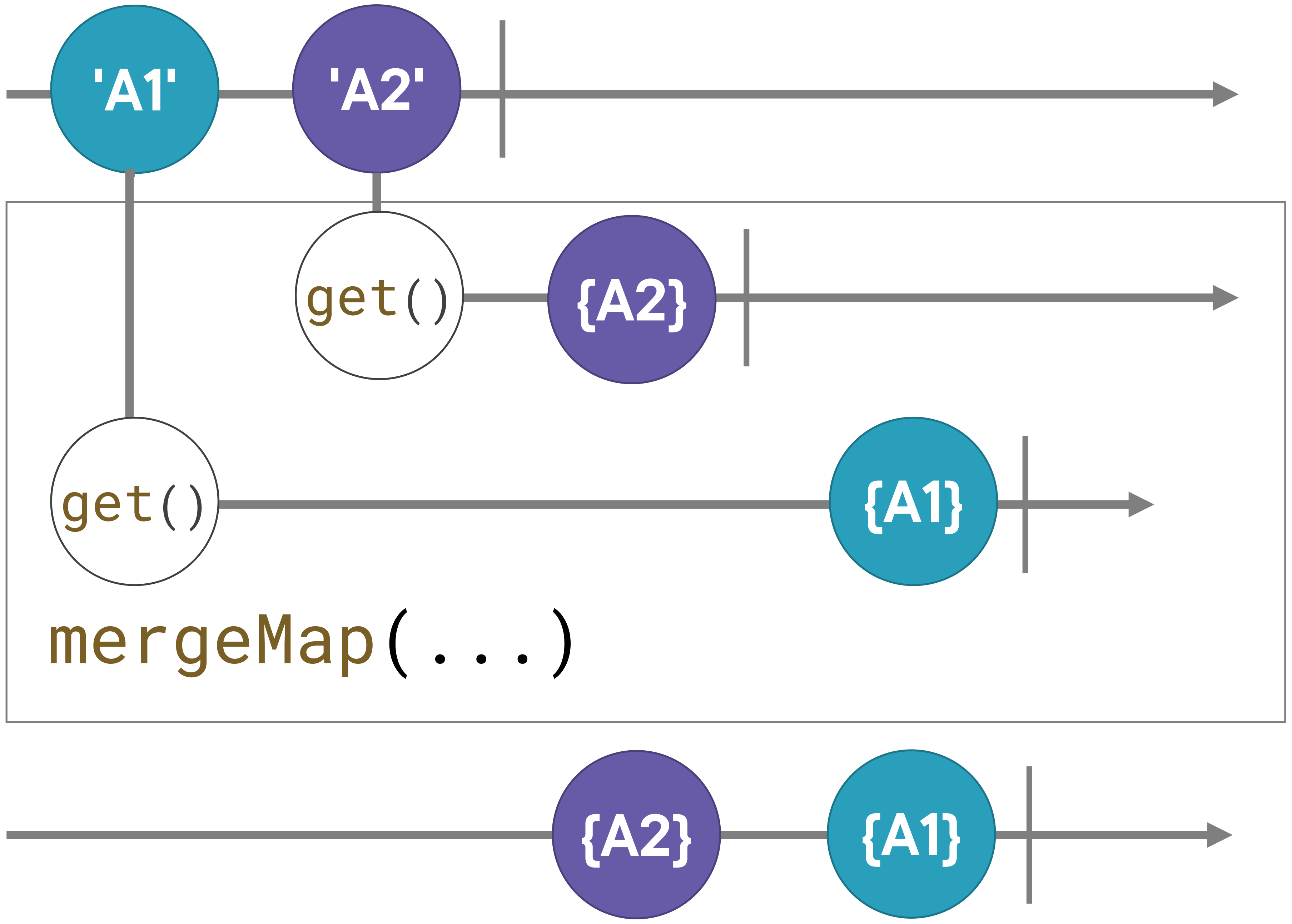
```
mergeMap(i => of(i))
```

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



`mergeMap` -> 800 Meter

Runners start concurrently

They all merge into the lower lanes

The runners complete based on how quickly they finish

RxJS Operator: `mergeMap (flatMap)`

`mergeMap` is a transformation operator

- Subscribes to its input Observable
- Creates an output Observable

When each item is emitted

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

Use `mergeMap`

To process in parallel

When order doesn't matter

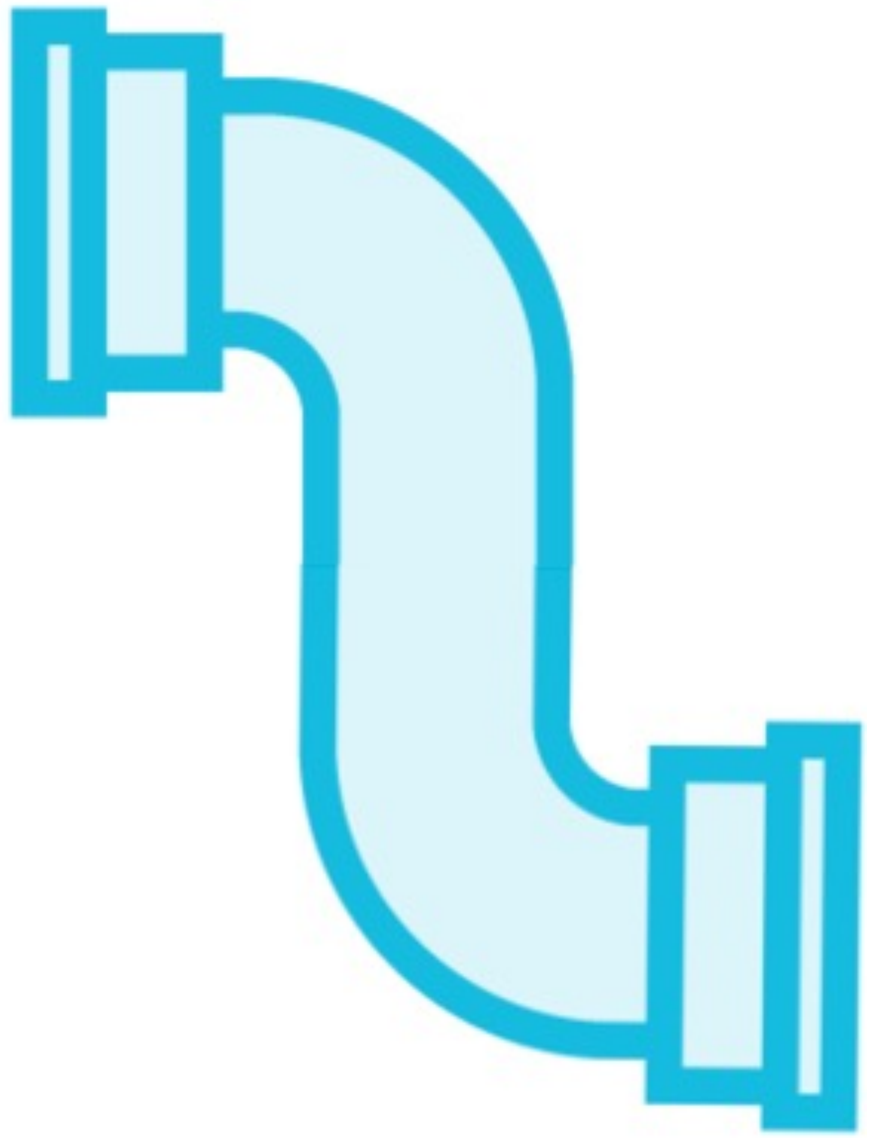
Examples:

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

Demo

`mergeMap`

RxJS Operator: `switchMap`



Higher-order mapping + switching

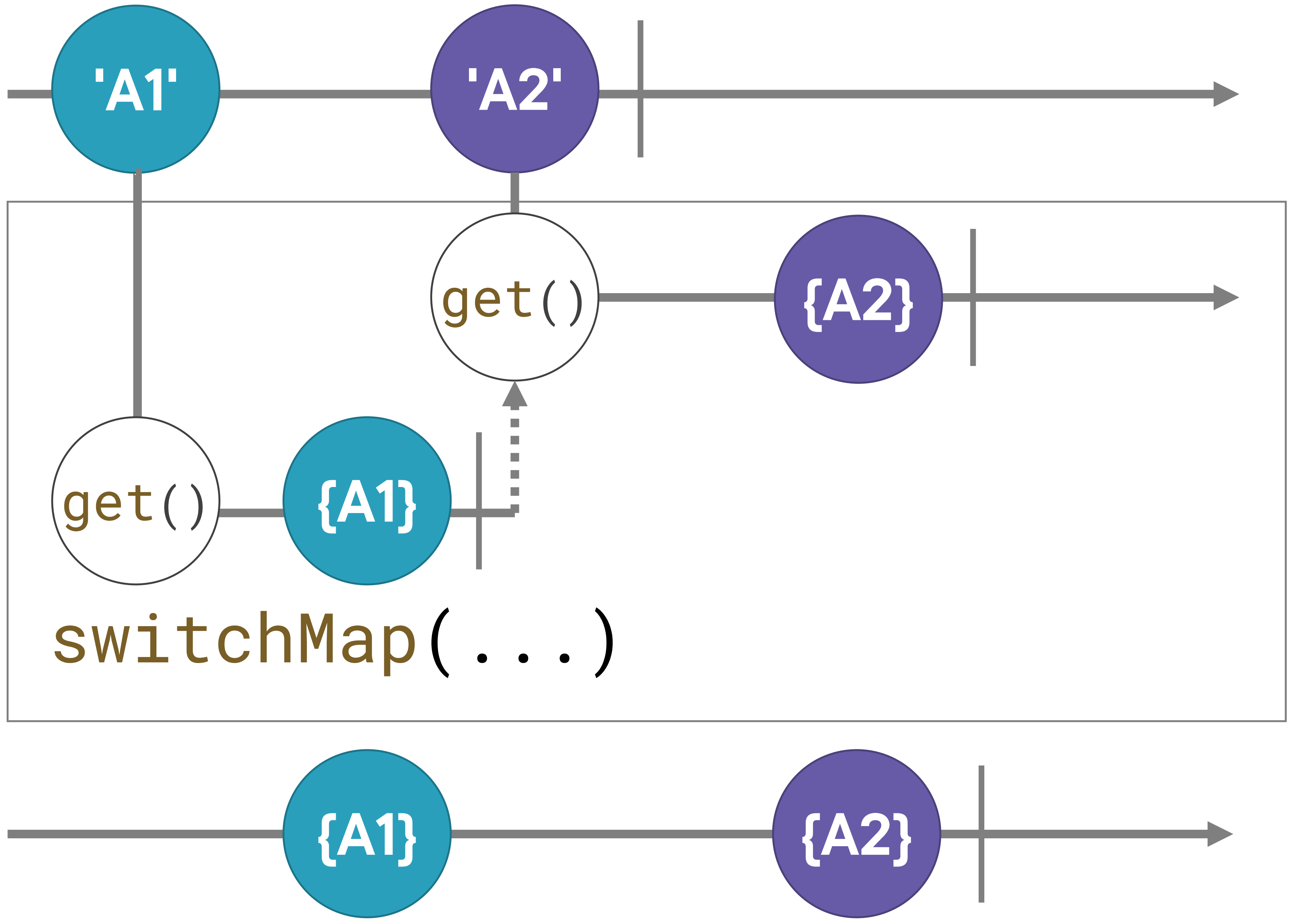
Transforms each emitted item to a new (inner) Observable as defined by a function

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

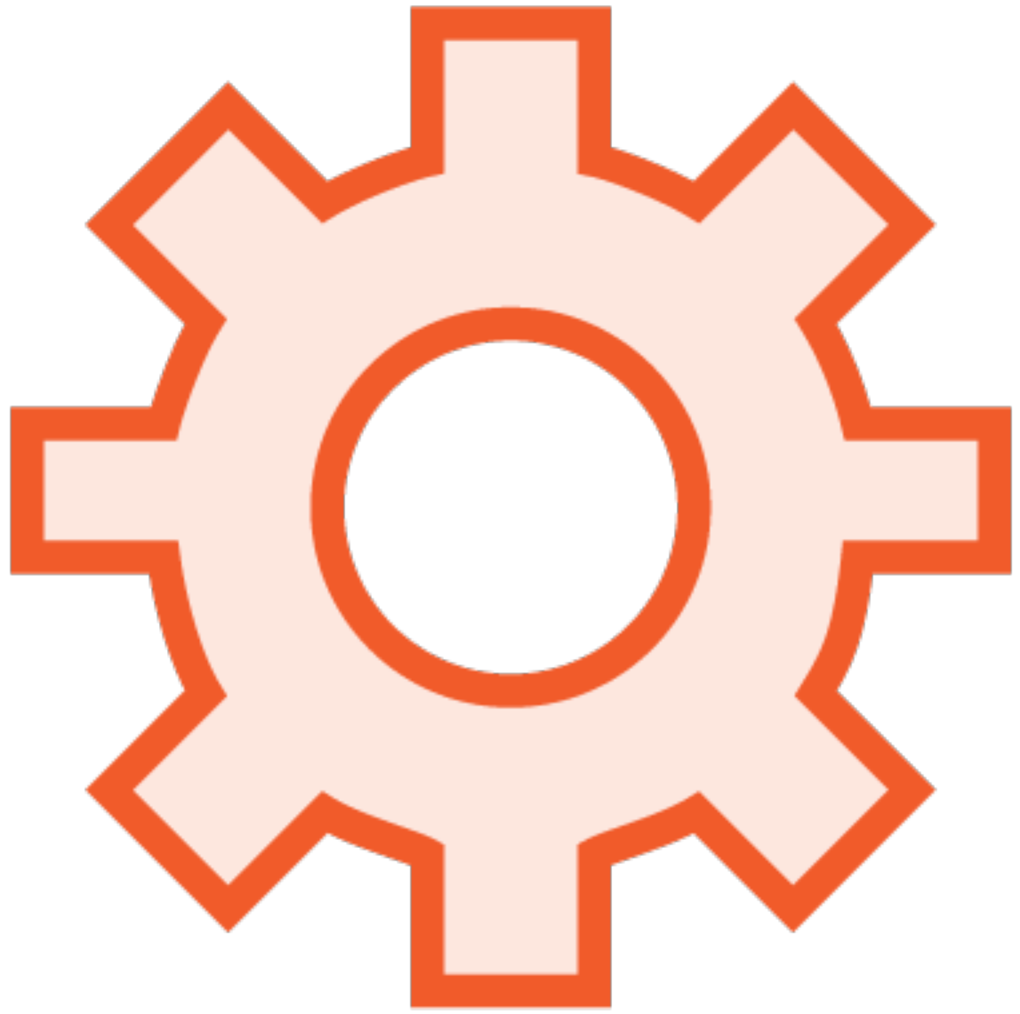


`switchMap` -> Changing Who's Running

The coach changes their mind as to which runner will run

Only one runner will run

RxJS Operator: `switchMap`



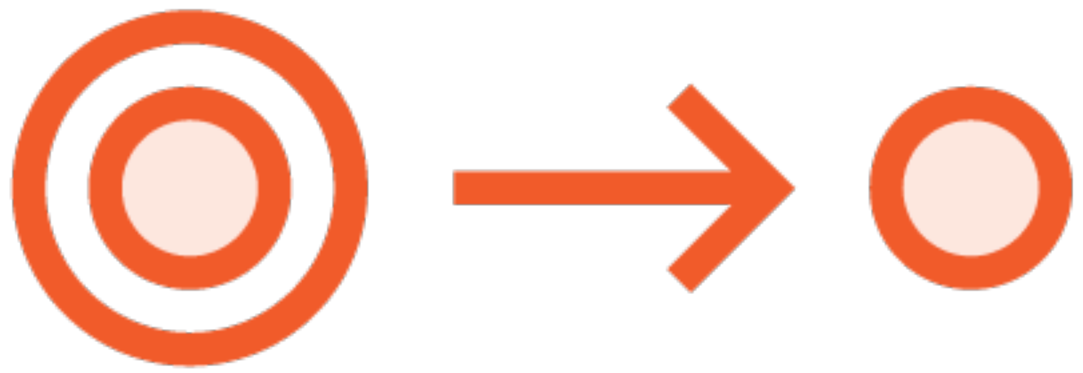
`switchMap` is a transformation operator

- Subscribes to its input Observable
- Creates an output Observable

When an 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
- Inner Observable emissions are merged to the output Observable

Use switchMap



To stop any prior Observable before switching to the next one

Examples:

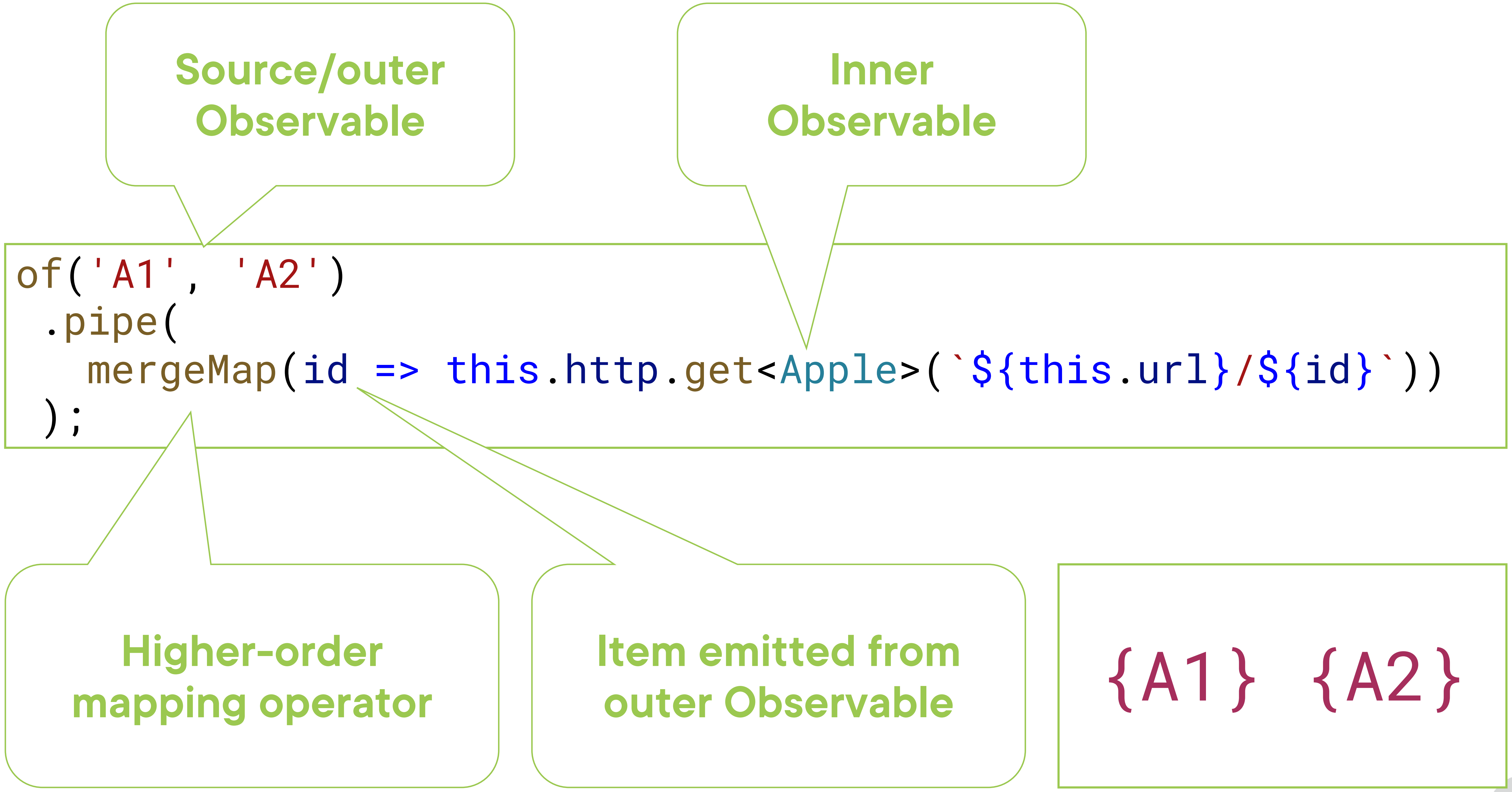
- Type ahead or auto completion
- User selection from a list

Demo

`switchMap`

RxJS Checklist: Higher-order Observable

Observable that emits Observables



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

Source/outer Observable
Inner Observable
Higher-order mapping operator
Item emitted from outer Observable
{A1} {A2}

RxJS Checklist: 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 Observable

Higher-order mapping operator functions

- Take in an item and return an Observable

Use instead of nested subscribes

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

RxJS Checklist: 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

RxJS Checklist: Use Case

```
todosForUser$ = this.userEnteredAction$
  .pipe(
    // Get the user given the username
    switchMap(userName =>
          this.http.get<User>(`${this.userUrl}?username=${userName}`)
      .pipe(
        // Get the todos given the user id
        switchMap(user =>
           this.http.get<ToDo[]>(`${this.todoUrl}?userId=${user.id}`)
        )
      )
    )
  );
```

Coming up next...

Combining All the Streams