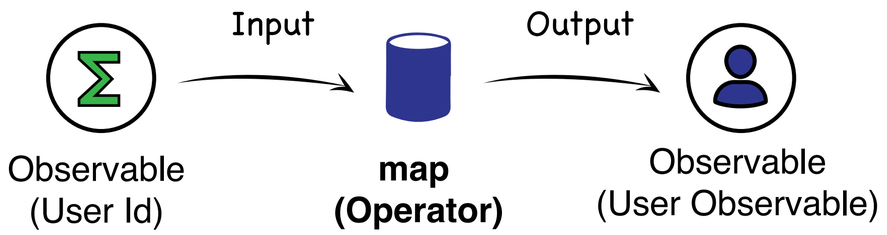
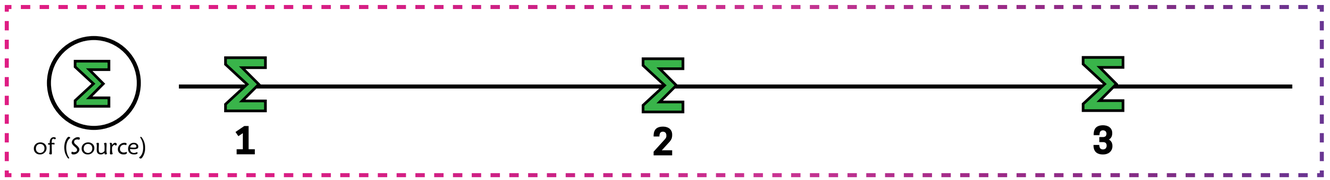
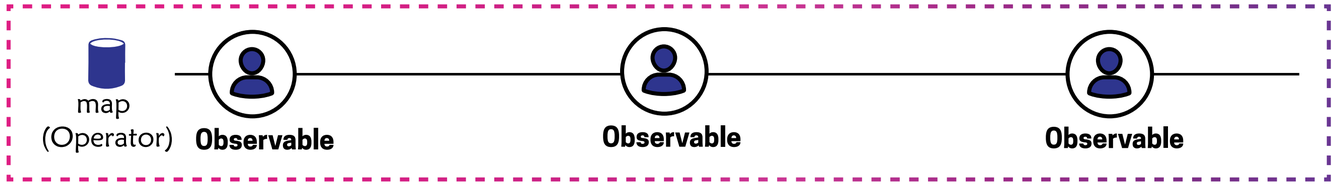
Higher Order Observables







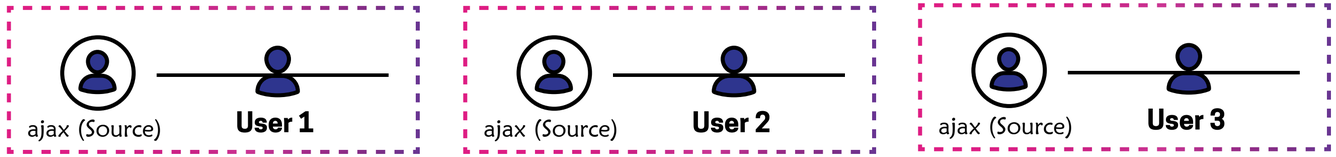
1 Outer Obs

2 Inner Obs

```
const usersObs = of(1, 2, 3)
.pipe(
  map( id => ajax.get(/user/id) )
)

usersObs.subscribe({...})
```

Instead of a direct value,
we get Observable as a value



Higher Order Observable

Observable of Observables

```
usersObs.subscribe(
  user$ => user$.subscribe({..})
)
```

Although it is an anti-pattern, but in upcoming videos
we will see the best approach to handle Higher-Order Observables

Examples

1 Outer Obs

```
const rangesObs = of(1, 2, 3)
.pipe(
  map(n => range(n, 3))
)

rangesObs.subscribe(
  range$ => range$.subscribe({..})
)
```

2 Inner Obs

Output

1, 2, 3

2, 3, 4

3, 4, 5

1 Outer Obs

```
const timersObs = interval(1000)
.pipe(
  map(v => timer(v * 1000))
)

timersObs.subscribe(
  timer$ => timer$.subscribe({..})
)
```

2 Inner Obs

Output

0 (for interval 0 with timer delay of 0)

.....

0 (for interval 1 with timer delay of 1000)

.....

0 (for interval 2 with timer delay of 2000)

.....

1 Outer Obs

```
const usersObs = from( [url1, url2, url3] )
.pipe(
  map(url => ajax.getJSON(url))
)

usersObs.subscribe(
  res$ => res$.subscribe({..})
)
```

2 Inner Obs

Output

Url1 Response

Url2 Response

Url3 Response

Higher Order Mapping Operators

1. concatMap
2. mergeMap
3. switchMap
4. exhaustMap

$$O_H \rightarrow O_I(V_T) \rightarrow V_T$$

Where
O_H is Higher Order Observable
O_I is Inner Observable
T is Transformation
V_T is Transformed Value

Higher Order Flattening Operators

1. concatAll
2. mergeAll
3. switchAll
4. exhaustAll
5. combineLatestAll

$$O_H \rightarrow O_I(V) \rightarrow V$$

Where
O_H is Higher Order Observable
O_I is Inner Observable
V is Value

Computer Baba

AJIT

Youtube Channel https://www.youtube.com/c/ComputerBabaOfficial

Twitter https://twitter.com/akacomputerbaba

Discord Server https://discord.gg/9V4VTDM