Transforming Operators

Outline

1. Getting started
2. Transforming elements
3. Transforming inner observables

1. Getting Started

- “Transforming operators” is one of the most important categories of operators in RxSwift.

- Using transforming operators all the time, to prepare data coming from an observable for use by you subscriber.

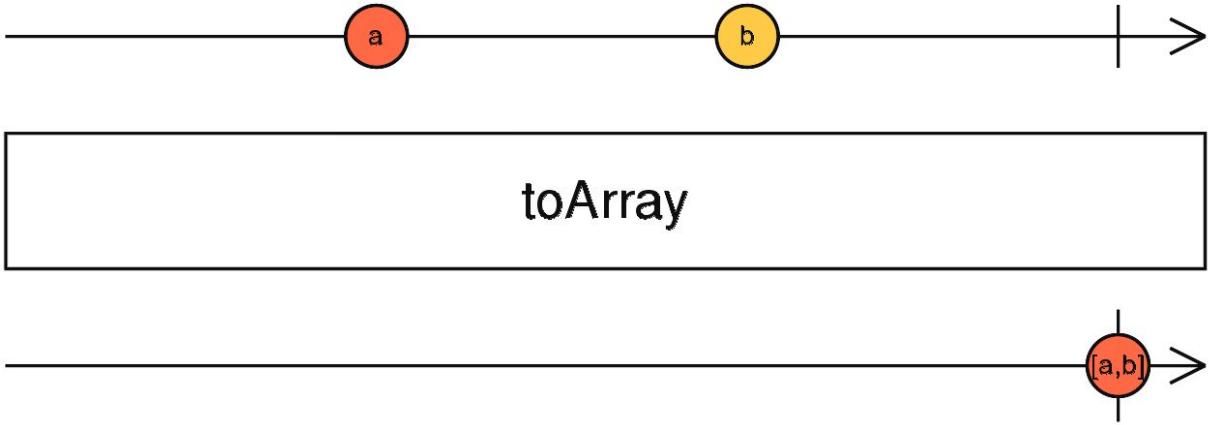
- Once again, there are parallels between transforming operators in RxSwift and the Swift standard library, such as map(_:) and flatMap(_:)

2. Transforming elements

1. toArray()
2. map()

2.1. toArray()

- toArray() will convert an observable sequence of elements into an array of those elements once observable completes and emit a .next event containing that array to subscribers.



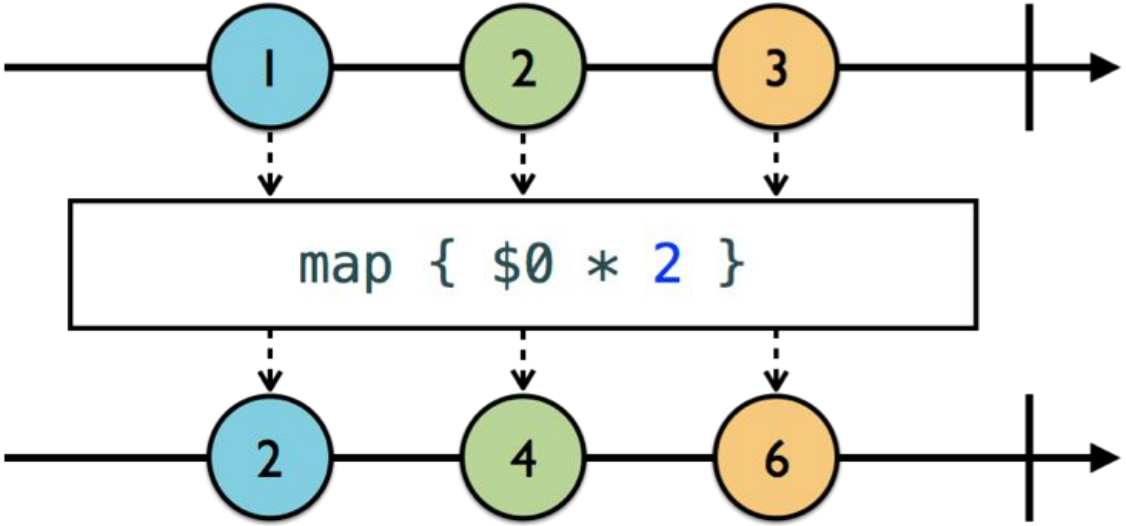
2.1. toArray()

- Example:

```
example(of: "toArray") {
    let disposeBag = DisposeBag()

    Observable.of(1, 2, 3, 4)
        .toArray()
        .subscribe(onSuccess: {
            print($0)
        })
        .disposed(by: disposeBag)
}
/*
--- Example of: toArray ---
[1, 2, 3, 4]
*/
```

2.2. map()

- RxSwift's map() operator works just like Swift's standard map, except it operates on observables.



2.2. map()

- Example:

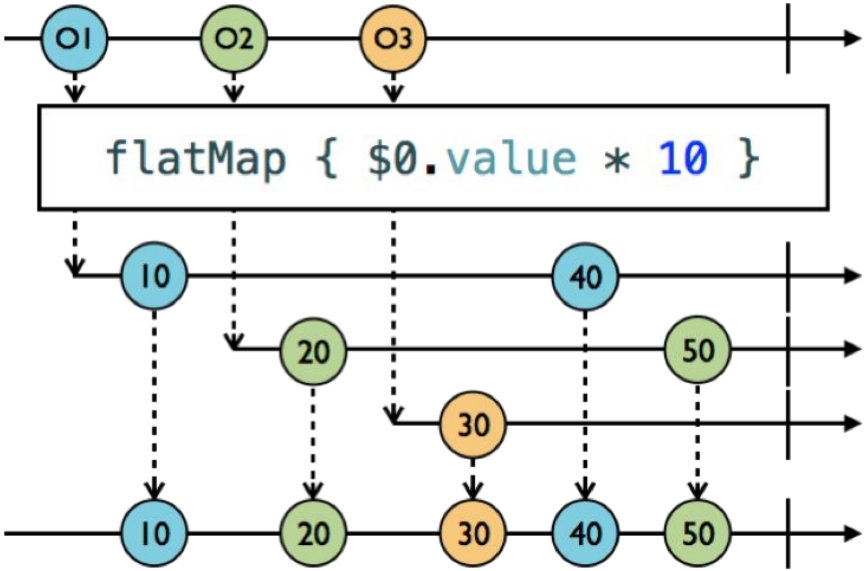
```
example(of: "map") {
  let disposeBag = DisposeBag()

  let formatter = NumberFormatter()
  formatter.numberStyle = .spellOut

  Observable.of(123, 4, 56)
    .map {
      formatter.string(for: $0) ?? ""
    }
    .subscribe(onNext: {
      print($0)
    })
    .disposed(by: disposeBag)
}
```

3. Transforming inner observables

1. flatMap()
2. flatMapLatest()

3.1. flatMap()

- Projects each element of an observable sequence to an observable sequence and merges the resulting observable sequences into one observable sequence



3.1. flatMap()

- Example:

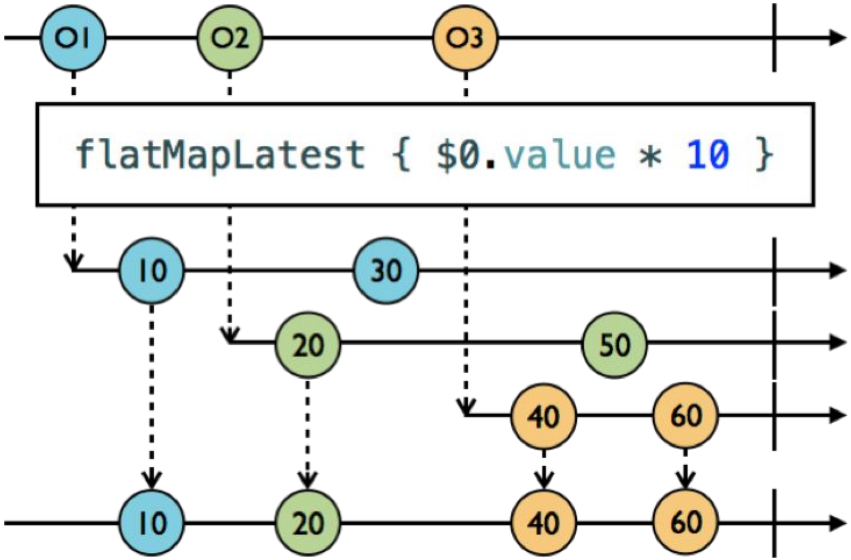
```
example(of: "flatMap") {
    let disposeBag = DisposeBag()

    struct Student {
        var name: String
        var score: BehaviorSubject<Int>
    }

    let studentA = Student(name: "Mr.A", score: BehaviorSubject(value: 5))
    let studentB = Student(name: "Mr.B", score: BehaviorSubject(value: 10))
    let studentC = Student(name: "Mr.C", score: BehaviorSubject(value: 15))

    Observable.of(studentA, studentB, studentC)
        .flatMap { element in
            return element.score.asObservable()
        }
        .subscribe(onNext: { score in
            print("Student's score \(score)")
        })
        .disposed(by: disposeBag)

    studentA.score.onNext(25)
    studentB.score.onNext(30)
    studentC.score.onNext(35)
}
/*
--- Example of: flatMap ---
Student's score 5
Student's score 10
Student's score 15
Student's score 25
Student's score 30
Student's score 35
*/
```

3.2. flatMapLatest()

- Projects each element of an observable sequence into a new sequence of observable sequences and then transforms an observable sequence of observable sequences into an observable sequence producing values only from the most recent observable sequence.

3.2. flatMapLatest()

- `flatMapLatest(_:)` is actually a combination of two operators, `map(_:)` and `switchLatest(_:)`.



3.2. flatMapLatest()

- Example:

```
example(of: "flatMap") {
    let disposeBag = DisposeBag()

    struct Student {
        var name: String
        var score: BehaviorSubject<Int>
    }

    let studentA = Student(name: "Mr.A", score: BehaviorSubject(value: 5))
    let studentB = Student(name: "Mr.B", score: BehaviorSubject(value: 10))
    let studentC = Student(name: "Mr.C", score: BehaviorSubject(value: 15))

    Observable.of(studentA, studentB, studentC)
        .flatMap { element in
            return element.score.asObservable()
        }
        .subscribe(onNext: { score in
            print("Student's score \(score)")
        })
        .disposed(by: disposeBag)

    studentA.score.onNext(25)
    studentB.score.onNext(30)
    studentC.score.onNext(35)
}
/*
--- Example of: flatMap ---
Student's score 5
Student's score 10
Student's score 15
Student's score 35
*/
```

Question & Answer?