

RxSwift Basics – Day 1

Younghwan Kim



- **Day 1 – Observable, Operator (Filter, Transform, Combine)**
- Day 2 – Subject (flatMap, flatMapFirst, flatMapLatest)
- Day 3 – Two VCs communications with Subject, RxCocoa (Button)
- Day 4 – Sequential, Merged Observable Calls
- Day 5 – RxCocoa, UI Binding (Button, TextField, Label, TableView)



Advanced RxSwift

- Day 1 – Protocol-Oriented Programming, Protocol Extension, Associatetype
- Day 2 – Network Call, Generic Enum
- Day 3 – Binding Track Activity (show / hide ‘Loading’)
- Day 4 – Adding a Reactive Extension to Custom UI Element,
2 Way Binding, Advanced TableView – RxDataSources
- Day 5 – Schedulers (observeOn, subscribeOn),
Unit Test (RxTest, RxBlocking)



Reactive Programming Overview

- Reactive Programming Overview (Jafar Hussain from Netflix)
- <https://www.youtube.com/watch?v=dwP1TNXE6fc>



-What are Reactive Extensions (Video)



- **Use Marble Diagrams(Video)**



Observable Sequence

Observable

- Emits events over time

Observer

- Subscribe to listen events emitted by the observable



Observable Sample - 1

```
@IBAction func observableOfTest(_ sender: UIButton) {  
    Observable.of(1, 2, 3)  
        .debug()  
        .subscribe(onNext: { some in  
            print(some)  
        })  
        .disposed(by: self.disposeBag)  
}  
  
@IBAction func observableFromTest(_ sender: UIButton) {  
    Observable.from([4,5,6])  
        .subscribe(onNext: { some in  
            print(some)  
        })  
        .disposed(by: self.disposeBag)  
}
```




Observable Sample - 2

```
func createTestObservable() -> Observable<Int>{  
    return Observable<Int>.create { observer in  
  
        return Disposables.create()  
    }  
}
```



Observable Sample - 3

```
func observableTestThree() {  
    Observable<Int>.create { observer in  
        observer.onNext(7)  
        observer.onNext(8)  
        observer.onNext(9)  
        observer.onCompleted()  
  
        return Disposables.create()  
    }.subscribe(onNext: { some in  
        print(some)  
    })  
    .disposed(by: self.disposeBag)  
}
```



map vs flatMap - 1

```
let arrOne = [21,22,23]
let arrTwo = [24,25,26]

let arrResultOne
  = arrOne
    .filter { $0 > 22 }
    .map { _ in arrTwo }

for element in arrResultOne {
  print("\(element)")
}
```



map vs flatMap - 2

```
let arrOne = [21,22,23]
let arrTwo = [24,25,26]

let arrResultTwo
  = arrOne
    .filter { $0 > 22 }
    .flatMap { _ in arrTwo }

for element in arrResultTwo {
  print("\(element)")
}
```



map vs flatMap - 3

```
func mapVsflatMap() {  
    let visitors = ["Tom", "Jerry", "Tweety", "Spike"]  
    Observable.from(visitors)  
        .map{ "Hello \($0)" }  
        .subscribe(onNext: { some in  
            print(some)  
        })  
        .disposed(by: self.disposeBag)  
  
    Observable.from(visitors)  
        .flatMap{ Observable.of("Hello \($0)") }  
        .subscribe(onNext: { some in  
            print(some)  
        })  
        .disposed(by: self.disposeBag)  
}
```



Lab - 1

Create Observables

Pikachu, Ash, Misty, Brock, Charmander, Squirtle

1. of
2. from
3. create



Lab - 2

filter

1,2,3,4,5 => less than 3

map

1,2,3,4,5 => 10,20,30,40,50



Lab - 3

merge, concat

Create two observables and use merge and concat operator

```
Observable.of(oneObservable,twoObservable)  
    .merge()
```

```
Observable.from([oneObservable,twoObservable])  
    .concat()
```




Lab - 4

map vs flatMap

Input 1,2,3

Output 1.

[2,3]

[4,6]

[6,9]

Output 2

2

3

4

6

6

9