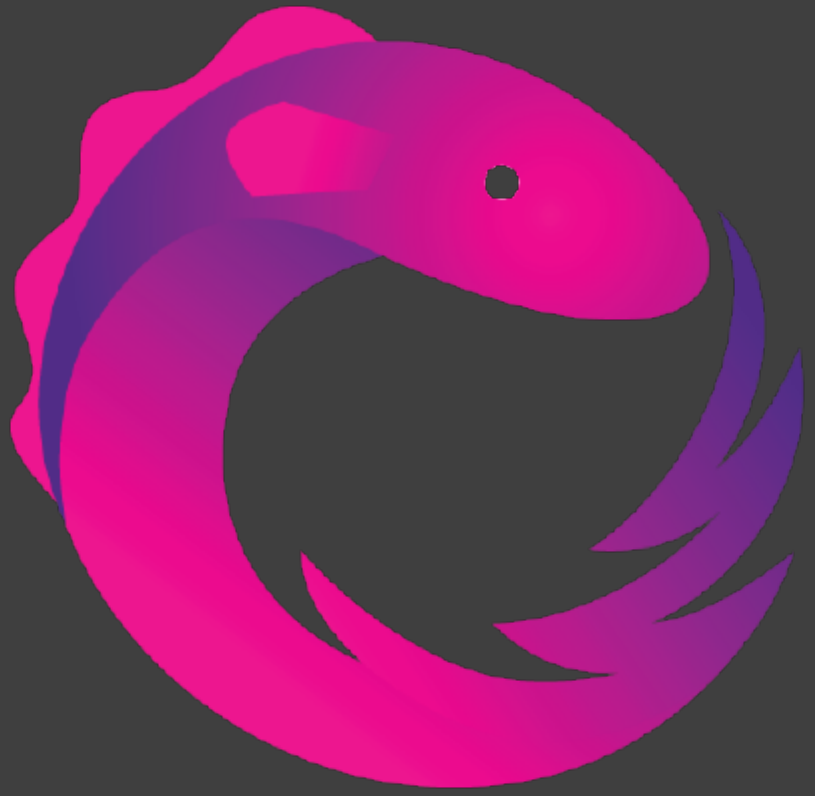
Reactive Extensions



Observables

Cold observables

```
Observable.of(1, 2, 3)
    .subscribe(
        l => console.log(l),
        noop,
        () => console.log('Completed')
    );
```

```
Observable.just(1, 2, 3)
    .doOnComplete(() -> System.out.println("Completed"))
    .subscribe(System.out::println);
```

Output:

1

2

3

Completed

Cold

Hot observables

```
const textSubject = new Subject<string>();
textSubject.subscribe(
    l => console.log(l),
    noop,
    () => console.log('Completed')
);
textSubject.next('foo');
textSubject.next('bar');
```

```
Subject<String> textSubject = PublishSubject.create();
textSubject
        .doOnComplete(() -> System.out.println("Completed"))
        .subscribe(System.out::println);
textSubject.onNext("foo");
textSubject.onNext("bar");
```

Output:

foo

bar

No Completed??

Complete Hot observables

```
const textSubject = new Subject<string>();
textSubject.subscribe(
    l => console.log(l),
    noop,
    () => console.log('Completed')
);
textSubject.complete();
```

```
Subject<String> textSubject = PublishSubject.create();
textSubject
        .doOnComplete(() -> System.out.println("Completed"))
        .subscribe(System.out::println);
textSubject.onComplete();
```

Output:

Completed

Subjects are closed!

Cold vs hot

- Terminates after N amount of values
- Fixed amount of values
- Ends after error OR complete

- Terminates manually by calling onCompleted
- Stays open for input
- Ends after error OR complete

Subscriptions

Subscription/Disposable

```
const textSubject = new Subject<string>();
const subscription = textSubject
    .delay(100)
    .subscribe(
        l => console.log(l),
        noop,
        () => console.log('Completed')
    );
textSubject.next('foo');
subscription.unsubscribe();
```

```
Subject<String> textSubject = PublishSubject.create();
Disposable disposable = textSubject
        .doOnComplete(() -> System.out.println("Completed"))
        .delay(100, TimeUnit.MILLISECONDS)
        .subscribe(System.out::println);
textSubject.onNext("foo");
disposable.dispose();
```

Output: Subscriptions are closed!

Subscriptions

- Cancel the stream action
- Prevent memory leaks
- Closes after error OR complete
- Does not affect the actual stream

RxDocs

Operators

Decision tree

Subject

Schedulers

Marbles