Chaining Observables

Nov. 15. 2018 Younghwan Kim

Observable – typical type

```
func oneObservable() -> Observable<Some> {
    return Observable<Some>.create { observer in

        return Disposables.create()
    }
}
```

Observable – Nested Observable

```
func oneObservable() -> Observable<Some> {
    return Observable<Some>.create { observer in
            nestedObservable()
            .subscribe(onNext: {
                        observer.onNext(Some)
                        observer.onCompleted()
            })
            .disposed()
        return Disposables.create()
    }
}
```

Observable – merge

```
Observable.from([oneObservable, twoObservable]).merge()
.subscribe()
.disposed()
```

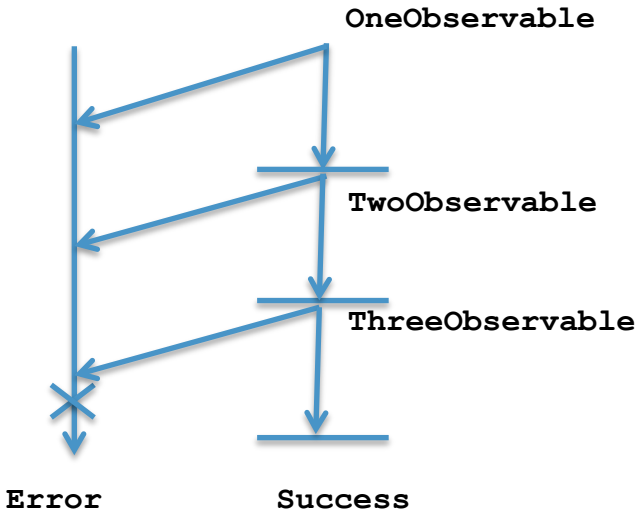
Observable – zip

```
Observable.zip(oneObservable, twoObservable)
.subscribe()
.disposed()
```

Observable – Observable of ZippedObservable

```
Func zippedOneTwoObservable() -> Observable<Some> {
    return Observable<Some>.create { observer in
            Observable.zip(oneObservable, twoObservable)
            .subscribe(onNext: {
                        observer.onNext(Some)
                        observer.onCompleted()
            })
            .disposed()
        return Disposables.create()
    }
}
```

Chaining Observables



Chaining Observables – typical format

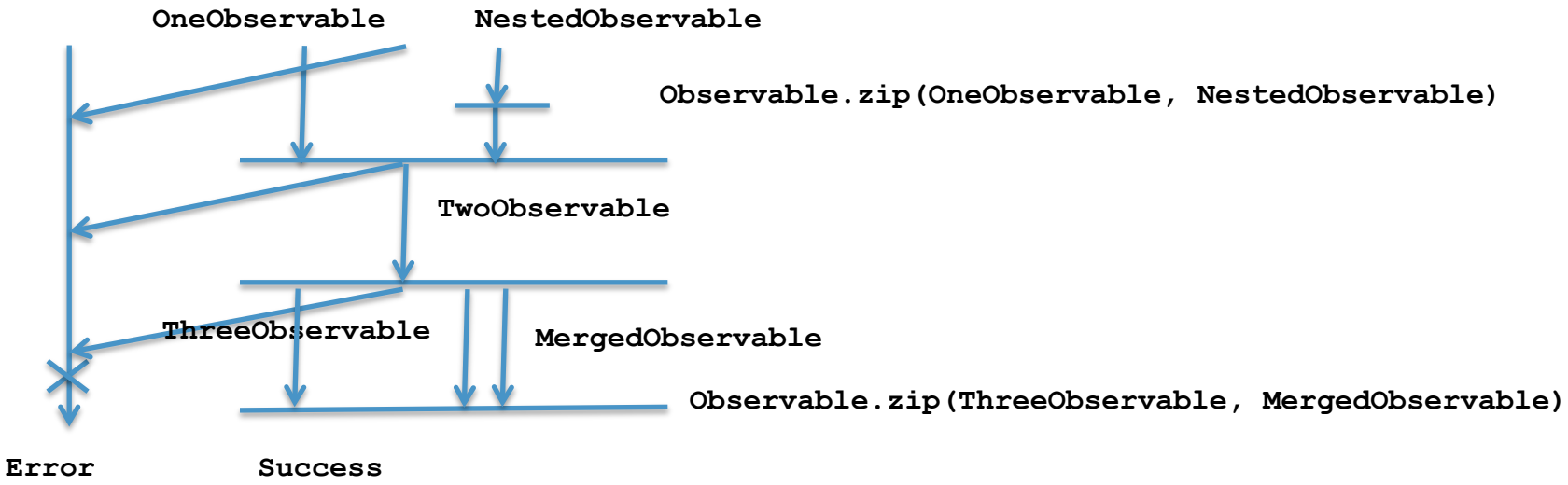
```
OneObservable()
.filter {


}
.flatMapLatest { _ -> Observable<Two> in
          TwoObservable()
}
.filter {


}
.flatMapLatest { _ -> Observable<Three> in
          ThreeObservable()
}
.subscribe()
.disposed()
```

Chaining Observables
- How to use the result of previous observable chain?

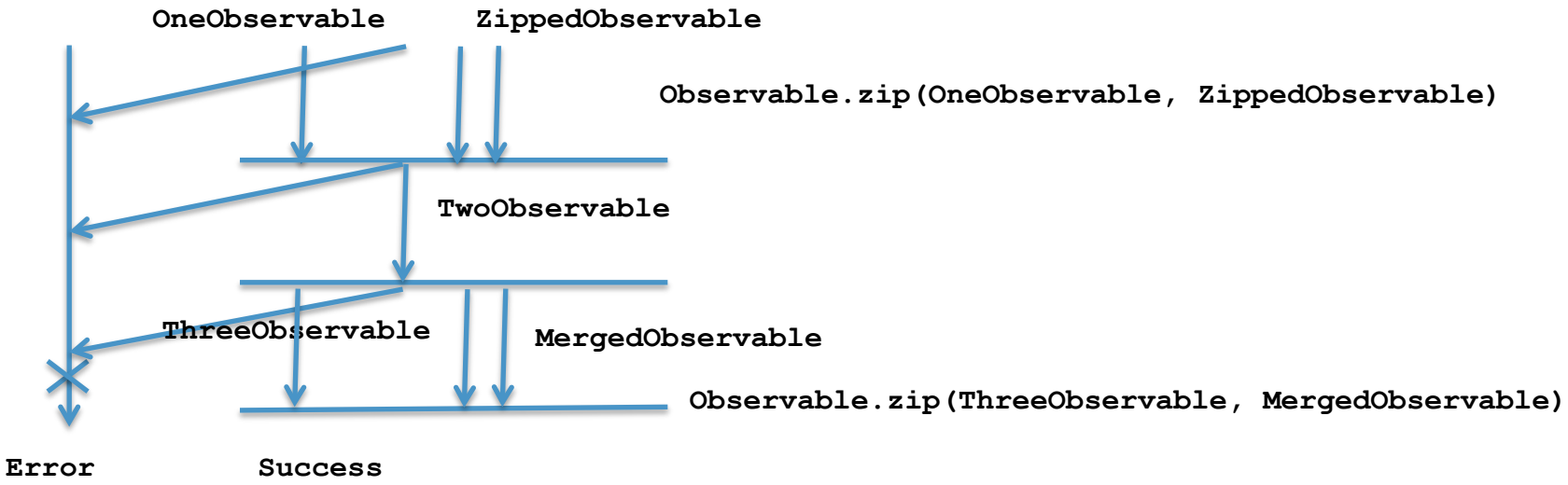
```
OneObservable()
.flatMapLatest { One -> Observable<Two> in
            if Some(One) {
                        return Observable<Two>
            } else {
                        TwoObservable()
            }
}.filter {


}
```

```
.flatMapLatest { _ -> Observable<Three> in
                  ThreeObservable()
}
.subscribe()
.disposed()
```

Chaining Observables – Nested Observable, merge, zip



Chaining Observables – merge, zip



Chaining Observables – merge, zip

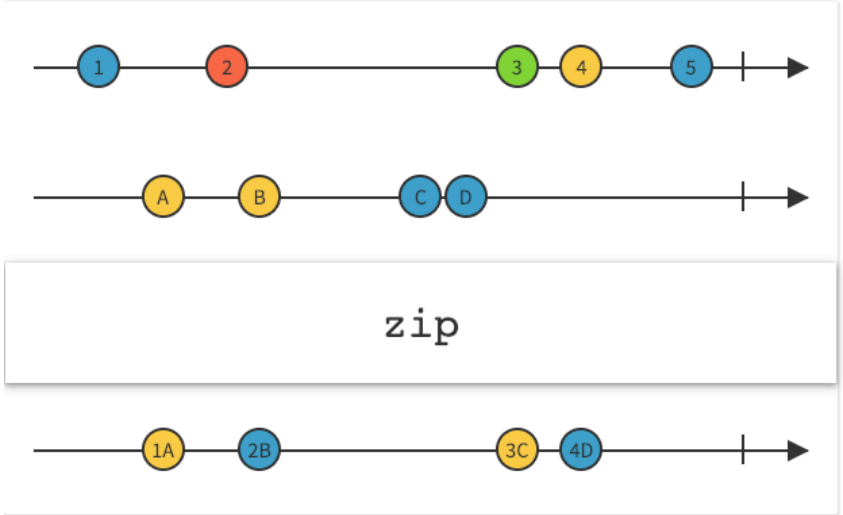
```
Observable.zip(OneObservable(),ZippedObservable())
.filter {


}
.flatMapLatest { _ -> Observable<Two> in
          TwoObservable()
}
.filter {


}
```

```
.flatMapLatest { _ -> Observable<Three> in
          Observable.zip(ThreeObservable(),
MergedObservable())
}
.subscribe()
.disposed()
```

Zip

Zip

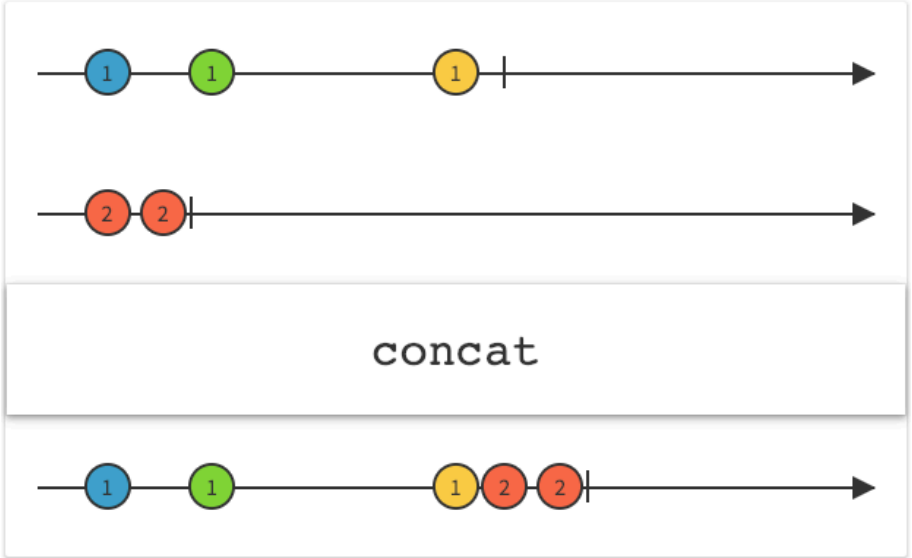
combine the emissions of multiple Observables together via a specified function and emit single items for each combination based on the results of this function



Concat

Concat

emit the emissions from two or more Observables without interleaving them



Merge

Merge

combine multiple Observables into one by merging their emissions