Rx Intro

O que é Rx?

- Lib que usa sequências observáveis para compor programas assíncronos e baseados em eventos de forma declarativa

Observer pattern

- **Subject** mantém uma lista de **Observers** e notifica todos eles quando um **Evento** acontece

```
NotificationCenter.default.addObserver(self, selector: .method, name: .Event, object: object)

NotificationCenter.default.post(name: .Event, object: someObject)
```

Programação imperativa x declarativa

- **Imperativa:** expressa através de comandos como uma determinada tarefa deve ser executada. Foco no **COMO** fazer

- **Declarativa:** expressa através de comportamentos a tarefa a ser executada. Foco no **O QUÊ** fazer

Programação imperativa x declarativa

Imperativo

```
let numbers = [1, 2, 3, 4, 5, 6]
var even = [Int]()
for n in numbers where (n % 2) == 0 {
    even.append(n)
}
```

Declarativo

```
let numbers = [1, 2, 3, 4, 5, 6]
let even = numbers.filter { ($0 % 2) == 0 }
```

Conceitos

- Observable
- Observer
- Subject

Observable ~ Sequence

- Todo **Observable** é uma **sequência** de eventos
- Eventos podem representar:
  - **Next** (emissão de um elemento)
  - **Completed** (término da sequência)
  - **Error** (término da sequência com erro)

Observable ~ Sequence

- Observable pode conter zero ou mais elementos
- Após emitir **error** ou **completed** o observable não emite mais nenhum evento e termina
- São lazy, ou seja, só emitem após alguém estar observando a sequência

Observer

- Observa eventos emitidos pela sequência e reage a eles de forma assíncrona

Subject

- Atua como Observable e Observer
- PublishSubject
- BehaviorSubject
- ReplaySubject

Code time

Operadores

- Funções aplicadas em uma ou mais sequências e que retornam outra sequência
- Podem ser encadeados para compor lógicas complexas
- Análogos as operações aplicadas em arrays:
  - filter, map, flatMap, reduce, skip .....

Operadores de filtro

- filter
- skip (while, whileWithIndex, until)
- take (while, whileWithIndex, until)
- distinctUntilChanged

More Code