Criando um Observable

Para criar um Observable, basta criar um objeto do tipo Observable. O Observable é um objeto possui um generic. Esse generic informa que tipo de dado o Observable vai retornar. Vamos retornar Strings nesse Observable.

```
import { Component, VERSION } from '@angular/core';
import { Observable } from 'rxjs';

@Component({
  selector: 'my-app',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  obs: Observable<string> = new Observable()
}
```

Esse objeto Observable deve receber uma função como argumento. Essa função deve possuir um parâmetro que será um **observer**. O observer será o elemento que retornará os possíveis dados retornados, o erro ou informar que a execução foi completada. Assim como os Observables, os objetos Observer possuem um generic que informa que tipo de dado será retornado. O tipo de dado do Observable e do Observer devem ser iguais.

```
import { Component, VERSION } from '@angular/core';
import { Observable, Observer } from 'rxjs';

@Component({
  selector: 'my-app',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  obs: Observable<string> = new Observable((observer: Observer<string>) => {

  })
}
```

Como vimos, um Observable pode possuir três possíveis respostas: sucesso, erro e completo. Para enviarmos uma mensagem de sucesso, a partir do observer, devemos

utilizar o método **next()**. Esse método retorna um dado do mesmo que você colocou no generic, nesse caso, Strings. Seu Observable pode ter quantos next você precisar.

```
import { Component } from '@angular/core';
import { Observable, Observer } from 'rxjs';

@Component({
  selector: 'my-app',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  obs: Observable<string> = new Observable((observer: Observer<string>) => {
    observer.next('dado1')
    observer.next('dado2')
    observer.next('dado3')
    observer.next('dado4')

    setTimeout(() => {
      observer.next('dado5')
    }, 2000)
  })
}
```

O método **next** irá retornar esses dados para quem estiver inscrito nesse Observable. Agora, para informarmos que houve um erro no observable, utilizamos o método **error()**. Esse método nos permite enviar qualquer tipo de dado que precisamos retornar.

```
import { Component } from '@angular/core';
import { Observable, Observer } from 'rxjs';

@Component({
  selector: 'my-app',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  obs: Observable<string> = new Observable((observer: Observer<string>) => {

    let hasError = true

    if (hasError) {
      observer.error({
        erro: true,
        mensagem: 'Deu erro, foi mal aí :('
      })
    }
  })
}
```

Estamos retornando um objeto quando ocorre algum erro no nosso observable. Agora, para informar que todos os dados foram enviados e todo o processo já foi completado, utilizamos o método **complete()**. Ele informa se toda a execução do Observable já foi concluída. Ao contrário dos outros métodos, o complete não retorna nenhum dado.

```
import { Component } from '@angular/core';
import { Observable, Observer } from 'rxjs';

@Component({
  selector: 'my-app',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  obs: Observable<string> = new Observable((observer: Observer<string>) => {

    observer.next('Dado 1')
    observer.next('Dado 2')
    observer.next('Dado 3')
    observer.next('Dado 4')

    observer.complete()
  })
}
```

Agora, para utilizarmos nosso observable, precisamos nos inscrever nele. Para isso, a partir do observable, utilizamos o método **subscribe()**. Esse método recebe, como parâmetro, 3 funções. Uma função para quando houver sucesso no observable, uma para quando houver erro e uma para quando for executado o complete, respectivamente.

```
import { Component } from '@angular/core';
import { Observable, Observer } from 'rxjs';

@Component({
  selector: 'my-app',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  obs: Observable<string> = new Observable((observer: Observer<string>) => {

    observer.next('Dado 1')
    observer.next('Dado 2')
    observer.next('Dado 3')
    observer.next('Dado 4')

    observer.complete()
  })

  ngOnInit() {
    this.obs.subscribe(
      (dado) => {
        console.log('Dado retornado', dado)
      },
      (erro) => {
        console.error('Ocorreu um erro', erro)
      },
      () => {
        console.info('Todos os dados foram retornados :)')
      }
    )
  }
}
```

Dessa maneira, podemos tratar todos os possíveis casos que podem acontecer.

Se Desinscrever de um Observable

Assim como podemos nos inscrever em um Observable, também podemos nos desinscrever de um Observable. Para isso, para podermos nos desinscrever de um observable, precisamos nos inscrever antes utilizando o método **subscribe()**. Esse método retorna um objeto do tipo Subscription. Esse objeto nos permite nos desinscrever desse observable. Basta apenas recuperarmos esse objeto e, a partir dele, utilizamos o método **unsubscribe()**

```
import { Component } from '@angular/core';
import { Observable, Observer } from 'rxjs';

@Component({
  selector: 'my-app',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  obs: Observable<string> = new Observable((observer: Observer<string>) => {

    observer.next('Dado 1')
    observer.next('Dado 2')
    observer.next('Dado 3')
    observer.next('Dado 4')

    observer.complete()
  })

  ngOnInit() {
    let sub = this.obs.subscribe(
      (dado) => {
        console.log('Dado retornado', dado)
      },
      (erro) => {
        console.error('Ocorreu um erro', erro)
      },
      () => {
        console.info('Todos os dados foram retornados :)')
      }
    )

    sub.unsubscribe()
  }
}
```

utilizando esse método, ele fará com que nós não recebamos mais os dados retornados.