Observables In Angular

⇒ Observables are construct to which you subscribe to be informed about changes in data

⇒ Observables are that stream of data, whenever a new data piece is emitted, our subscription will know it

In Angular, we already have Observables you only subscribe them, you don't have to create them.

* Observables are added by a package name RxJs

If we take one eg of Observable

Eg → interval (period: 1000). subscribe (
next : count => {
console.log (count); // new value emitted every second
})

interval (period: 1000) is a built in function that gives observable

Let's say we made Interval fn on Home component

We can see value 0, 1, 2 . . . . incrementing In console

If we navigate away to home component and go on other

That doesn't stop observable count.

So Except certain observable like the One we use in http → we get response only once., Other observables keep on running

So to avoid memory leaks, you should stop observable

For that you must unsubscribe observables you're no longer Interested in

Too many Observable ~~is~~ running in your app, you run out of resources and slow down your app and also too much of memory leakage happens

Therefore You must Unsubscribe

To avoid that you can store your Subscription first

* declare private FirstSubscription : Subscription

* import { Subscription } from 'rxjs';

Now we store subscription returned by our subscribe

* this.firstSubscription = interval(period: 1000).
  subscribe(next: count => {
  console.log(count);
  });

Now, after this we can also Implement OnDestroy here

Inside ngOnDestroy(): void {
  this.firstSubscription.unsubscribe();
}

Now whenever we leave that component, we clear subscription and prevent memory leaks.

Remember: All the Angular Package Observable like http, params that are built-in Angular Observable, You don't have to Unsubscribe, Angular does that Automatically

Page
Date

How to create Custom Observable

* import {Observable} from 'rxjs'

* declare const customObservable = Observable.create();

* Observable.create will store an arrow function which will have arguement automatically passed by 'rxjs'

It will look like this :

```
const customObservable = Observable.create((observer) =>
    {setInterval(() => {
        let count = 0;
        Console.log("observable called");
        observer.next(count);
        count++;
    },
    });
```

← It will emit new value

Now you call it using subscribe.

```
customObservable.subscribe(data => {
        console.log(data);
    });
```

* We have more methods like Observer.next (which is used to emit new value) • Observer.error (handles for error notification)
• Observer.complete (handles for complete execute)

Observer.error ⇒ We use it when an error fires up in our observable

It will show our error message in dev tool

Observer.complete ⇒ When your observable complete its action then observer.complete is called, however by default nothing shows in console.log

However If you mention a blank arrow function like this

```
Observer.subscribe(data => {
    console.log(data);
},
    error => {
    alert(error.message);
},
    () => {
    Console.log("completed")
});
}
```

→ This you will see on Observer.complete()

(If observer cancelled due to error)

Otherwise you'll never see observer.complete message in your console

Operators In Observable by 'rxjs'

We have one of Important constructs that is operators of rxjs.

Let's say you have a data coming from httpRequest and you want to change it's syntax.

You can do it directly in your main method But It is alway betted to do these modification prior there use.

→ Every Observable have pipe method and you will use that here to use operators

one operator is map() and filter()
There are more, for that you can explore rxjs

How to Use operators

While subscribing add your operator like this

```
CustomObservabe.pipe(map(data:number) =>
{
  return ('Round:' + (data); })).subscribe
  ((data) => {
              console.log(data);
  });
}
```