OBSERVABLE Vs PROMISES

https://www.youtube.com/playlist?list=PL6n9fhu94yhWqGD8BuKuX-VTKqlNBj-m6

To deal with asynchronous data we use either promise or observable

Promise:

---

by default angular 6 http will works as Observable. if you want to change observable to promise we can use method called toPromise()

for this

```
//in service
import { HttpClient } from '@angular/common/http';
import { User } from './user';
import { Injectable } from "@angular/core";

@Injectable()
export class CommonService{
 constructor(private http:HttpClient){}

 loginUser(user:User):Promise<any>{
  return this.http.post("/api/loginuser",user)
  .toPromise()
 }
 registerUser(user:User){
  return this.http.post("/api/registerUser",user);
 }
 listUsers():Promise<any>{
  return this.http.get("/api/listUsers")
  .toPromise()
 }
}
```

```
//in component

import { CommonService } from './common.service';

import { Component } from "@angular/core";
import { User } from "./user";
import { Router } from '@angular/router';
@Component({
 selector: "login",
 templateUrl: "./login.component.html",
 styleUrls: ['../assets/vzbootstrap/styles/bootstrap.min.css', '../assets/vzbootstrap/styles/vzbootstrap-main.min.css', '../assets/styles/styles.css']
})
export class LoginComponent {
 user: User;
 constructor(private cs: CommonService, private router: Router) {
  this.user = new User()
 }
 login() {
     this.cs.loginUser(this.user).then((data) => {
    if (data) {
     console.log(data);
     alert("success");
     this.router.navigate(['/userList']);
    }
   })
  }
 }
 promise emits a single item.
```

Promise->then conatins two callback methods

1.onfulfilled->when the request is completed successfully

2.onrejected->when request gets failed

Observable:

```
//component
import { Component } from "@angular/core";
import { User } from "./user";
import { CommonService } from "./common.service";
@Component({
 selector:"userList",
 templateUrl:"./userList.component.html"
// template:"in User llst"
})
export class userListComponent{
 users:User;
 constructor(private cs:CommonService){
  this.cs.listUsers().subscribe((data)=>{
     this.users = <User>data;
  });

 }
}
```

```
//service
import { Observable } from 'rxjs';
import { HttpClient } from '@angular/common/http';
```

```
import { User } from './user';
import { Injectable } from "@angular/core";

@Injectable()
export class CommonService{
 constructor(private http:HttpClient){}
 loginUser(user:User):Promise<any>{
  return this.http.post("/api/loginuser",user)
  .toPromise()
 }
 registerUser(user:User){
  return this.http.post("/api/registerUser",user);
 }
 listUsers():Observable<any>{
  return this.http.get("/api/listUsers");
 }
}
```

Observable emits multiple items over a period of time

Observable ->subscribe method contains three parameters

1. next:since observable emits multiple items....each item successfully emitted by observable so next callback will be called

for each item next callback will called

and when the last item is emitted and this next method will called and the last item has handled to indicate that observable is complete..then the complete callback will called.

thats way we have two parameters we have next and complete

with promise we have only one callback when the call was fulfilled

2. error:If any error this error callback will be called

3. complete

Since Observable emits multiple items. we have a next method to handle every item..

for last emitted item we have complete method to indicates the emitted values were completed.

Promise:

Promise is not Lazy

Proof:

---

lets open developer tools and click on network tab we can see promise request

for now just remove then mthod from promise then see same in developer tools network area...yes we have a request.

this.cs.loginUser(this.user)

so here we proved promise are not lazy. if you are not using then method also the request is happing.

Observable:

Lazy.An Observable is not called until we subscribe to the observable

proof:

---

lets open developer tools, click on network tab and we can notice here several calls loaded because we subscribed to observable.

for now just remove subscribe from observable code then see same in developer tool we wont get any call request.

this.cs.listUsers();

so here we proved until subscribe we wont get request even.

Promise:

----

promise can not be cancelled

Observable:

----

can be canceled by using unsubscribe() method.

->Observable provides operators like map,forEach,filter,reduce,retry,retryWhen...etc

Observable retry on error:

----

moto is if there is any error we want to resubscribe to application

retry will used for send request for number of times.we have to use pipe for this operation.

```
//component
import { CommonService } from './common.service';
import { retry } from 'rxjs/operators';
import { Component } from "@angular/core";
import { User } from "./user";
import { Router } from '@angular/router';
@Component({
 selector:"login",
 templateUrl:"./login.component.html",
 styleUrls: ['../assets/vzbootstrap/styles/bootstrap.min.css','../assets/vzbootstrap/styles/vzbootstrap-main.min.css','../assets/styles/styles.css']
})
export class LoginComponent{
 user:User;
 access:boolean = true;
 constructor(private cs:CommonService,private router:Router){
  this.user = new User()
```

```
}
login(){
 this.access = false;
 localStorage.setItem("user","chandra")
 this.cs.loginUser(this.user)
                //.pipe(retry())//retry the request when server error here we are retried infinate times.
  .pipe(retry(3))//retry the request when server error here we are retried 3 times.
  .subscribe((data)=>{
    if(data){
     this.router.navigate(['/userList']);
    }
  })
 }
}
///common service ////
import { Observable } from 'rxjs';
import { HttpClient } from '@angular/common/http';
import { User } from './user';
import { Injectable } from "@angular/core";
@Injectable()
export class CommonService{
 constructor(private http:HttpClient){}
 /*loginUser(user:User):Promise<any>{
  return this.http.post("/api/loginuser",user)
  .toPromise()
 }*/
 loginUser(user:User):Observable<User>{
  return this.http.post<User>("/api/loginuser",user);
```

```
  }
  registerUser(user:User){
   return this.http.post("/api/registerUser",user);
  }
  listUsers():Observable<User>{
   return this.http.get<User>("/api/listUsers");
  }
}
```

so this retry is requesting immediatly but I want to retrying after some delay. so we will retryWhen()

retryWhen():

---

retryWhen take other function as a parameter and that function takes an Observable and return an Observable.

what kind of Observable we need to pass here ?

ok,here when error occur we can retry for every 1000ms delay

so

```
retryWhen((error)=>error.delay(1000))
```

```
//component
import { CommonService } from './common.service';
import { retry, retryWhen,delay } from 'rxjs/operators';
import { Component } from "@angular/core";
import { User } from "./user";
import { Router } from '@angular/router';
@Component({
 selector:"login",
 templateUrl:"./login.component.html",
 styleUrls: ['../assets/vzbootstrap/styles/bootstrap.min.css','../assets/vzbootstrap/styles/vzbootstrap-main.min.css','../assets/styles/styles.css']
})
```

```
export class LoginComponent{
 user:User;
 access:boolean = true;
 constructor(private cs:CommonService,private router:Router){
  this.user = new User()
 }
 login(){
  this.access = false;
  localStorage.setItem("user","chandra")
  this.cs.loginUser(this.user)
    .pipe(retryWhen((err)=>err.pipe(delay(1000))))
    .subscribe((data)=>{
      if(data){
       this.router.navigate(['/userList']);

      }
    })
  }
 }

 //service//
 import { Observable } from 'rxjs';
 import { HttpClient } from '@angular/common/http';
 import { User } from './user';
 import { Injectable } from "@angular/core";
 @Injectable()
 export class CommonService{
  constructor(private http:HttpClient){}
```

```
 /*loginUser(user:User):Promise<any>{
  return this.http.post("/api/loginuser",user)
  .toPromise()
 }*/
 loginUser(user:User):Observable<User>{
  return this.http.post<User>("http://localhost:3000/api/loginuser",user);
 }
 registerUser(user:User){
  return this.http.post("/api/registerUser",user);
 }
 listUsers():Observable<User>{
  return this.http.get<User>("http://localhost:3000/api/listUsers");
 }
}
```

If we want to print number of attempts in web page then user scan method:

scan:

---

scan function has two parameters

first parameter is other function and it is called accumulater here we need to track error

second parameter is seed

```
//component
import { CommonService } from './common.service';
import { retry, retryWhen, delay, scan } from 'rxjs/operators';
import { Component } from "@angular/core";
import { User } from "./user";
import { Router } from '@angular/router';
@Component({
 selector: "login",
 templateUrl: "./login.component.html",
```

```
 styleUrls: ['../assets/vzbootstrap/styles/bootstrap.min.css', '../assets/vzbootstrap/styles/vzbootstrap-main.min.css', '../assets/styles/styles.css']
})
export class LoginComponent {
 user: User;
 access: boolean = true;
 statusMsg: string;
 constructor(private cs: CommonService, private router: Router) {
  this.user = new User()
 }
 login() {
  this.access = false;
  localStorage.setItem("user", "chandra")
  this.cs.loginUser(this.user)
   .pipe(retryWhen((err) => {
    return err.pipe(scan((retryCount) => {
     retryCount += 1
     if (retryCount < 6) {
      this.statusMsg = "server connection error attempt"+ retryCount;
      console.log(this.statusMsg);
      return retryCount;
     }
     else {
      throw (err);
     }
    }, 0)).pipe(delay(1000))
   }))
   .subscribe((data) => {
    if (data) {
```

```
      this.router.navigate(['/userList']);
     }
    })
  }
}
//service
import { Observable } from 'rxjs';
import { HttpClient } from '@angular/common/http';
import { User } from './user';
import { Injectable } from "@angular/core";
@Injectable()
export class CommonService{
  constructor(private http:HttpClient){}

   /*loginUser(user:User):Promise<any>{
    return this.http.post("/api/loginuser",user)
    .toPromise()
   }*/
   loginUser(user:User):Observable<User>{
    return this.http.post<User>("http://localhost:3000/api/loginuser",user);
   }
   registerUser(user:User){
    return this.http.post("/api/registerUser",user);
   }
   listUsers():Observable<User>{
    return this.http.get<User>("http://localhost:3000/api/listUsers");
   }
  }
```