Rxjs operators code

import { HttpClient, HttpClientModule } from '@angular/common/http';

import { Component, inject } from '@angular/core';

import { FormControl, ReactiveFormsModule } from '@angular/forms';

import { RouterOutlet } from '@angular/router';

import { Observable, Subject, catchError, concatMap, debounceTime, exhaustMap, filter, forkJoin, interval, mergeMap, observable, switchMap, take, takeUntil, throwError } from 'rxjs';

@Component({

selector: 'app-root',

standalone: true,

imports: [RouterOutlet , ReactiveFormsModule, HttpClientModule],

templateUrl: './app.component.html',

styleUrl: './app.component.scss'

})

export class AppComponent {

title = 'rxjs-operators';

gotData:boolean = false;

searchControl = new FormControl();

constructor(private http: HttpClient){}

ngOnInit(){

// let sub = new Subject();

// let observable = interval(1000);

// observable.pipe(

// filter(i => i > 0),

// takeUntil(sub)

// ).subscribe((res)=>{

// console.log(res)

// })

// setTimeout(()=>{

// if(this.gotData){

// sub.next('');

// }

// }, 5000)

//this.funDebounceTime();

//this.funSwitchMergeConcatExhaustMaps();

//this.funForkJoin();

this.funCatchThrowError();

}

funDebounceTime(){

this.searchControl.valueChanges.pipe(

debounceTime(1000)

).subscribe((res)=>{

console.log(res)

})

}

funSwitchMergeConcatExhaustMaps(){

let observable = interval(500);

observable.pipe(

filter(i=>i>0),

take(5),

exhaustMap((id)=> this.http.get("https://jsonplaceholder.typicode.com/posts/"+id))

).subscribe((res)=>{

console.log(res)

})

}

funForkJoin(){

const observable1 = new Observable((observer)=>{

setTimeout(()=>{

observer.next("First Observable")

observer.complete();

}, 1000)

});

const observable2 = new Observable((observer)=>{

setTimeout(()=>{

observer.next("Second Observable")

observer.complete();

}, 5000)

})

forkJoin(

observable1,

observable2

).subscribe((res)=>{

console.log(res)

})

}

funCatchThrowError(){

return this.http.get("https://jsonplaceholder.typicode.com/psts/1").pipe(

catchError(this.handleError)

).subscribe((res)=>{

console.log(res)

}, (err)=>{

console.log(err)

} , ()=>{

console.log("Completed")

})

}

handleError(){

return throwError("SOmething went wrong, please try again");

}

}