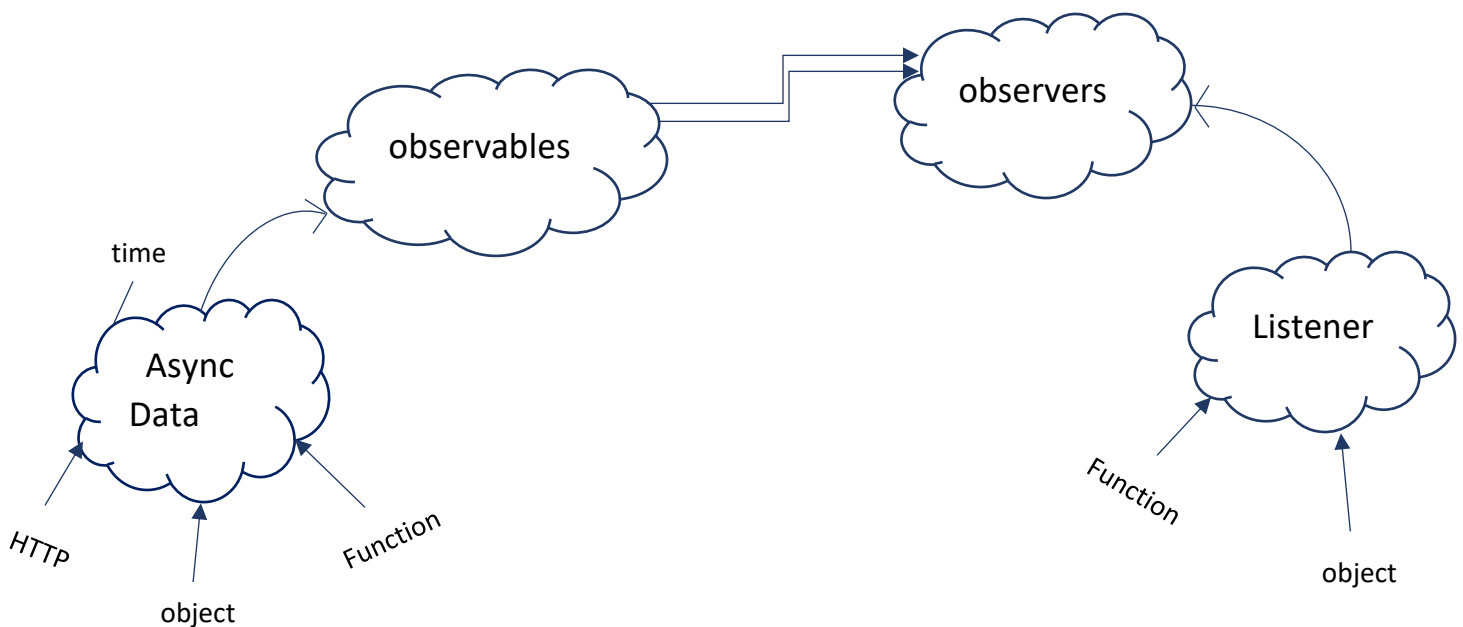


RxJS

RxJS stands for (Reactive Extensions for JavaScript) is a library for reactive programming using observables that makes it easier to compose asynchronous or callback-based code.

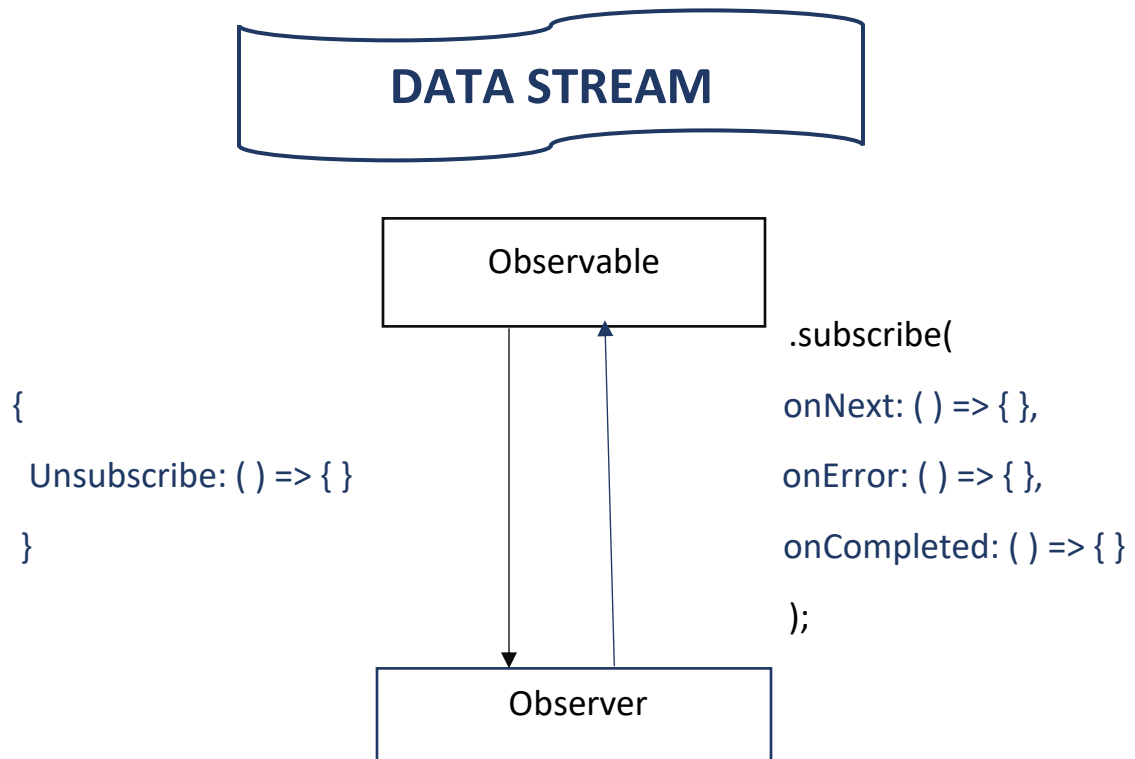
The main goal of RxJS is to handle asynchronous data stream easily.

Observable and Observer



Observables represent async data stream observers represent listener who is actually interested to hear this data stream

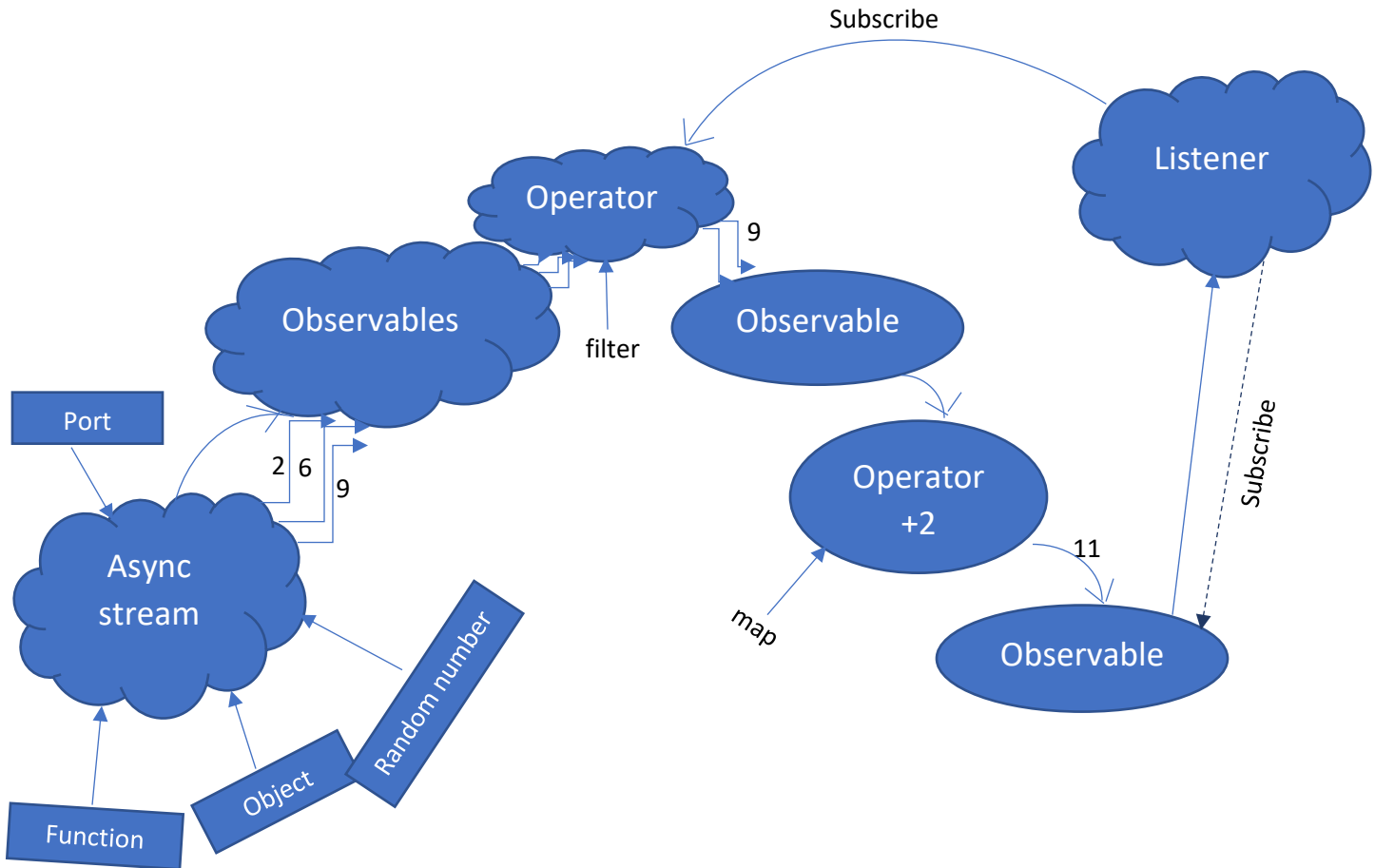
Observables and observers are nothing but they are RxJS objects, observables represent Async data stream and observers is nothing but a function which is subscribe to that observables and listen to that data stream and receive the data stream



The goal of subscribe method is to attach Async data with listener

Operators

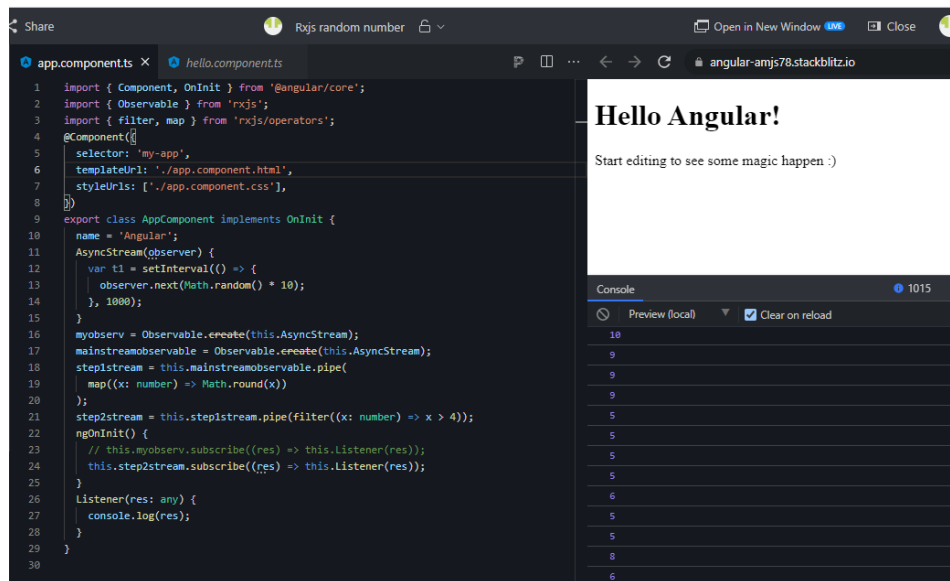
Operators are nothing but they are small pice of logic which actually convert an observables stream to another observables stream



Observables are nothing but they are small piece of logic which you can plugin into the pipe line and you can apply filter, map, sort with the help of RxJS operator

Example

<https://stackblitz.com/edit/angular-amjs78?file=src/app/app.component.ts>



The screenshot shows a web browser window with a StackBlitz editor. The editor has two tabs: 'app.component.ts' and 'hello.component.ts'. The 'app.component.ts' tab is active, showing the following TypeScript code:

```
1 import { Component, OnInit } from '@angular/core';
2 import { Observable } from 'rxjs';
3 import { filter, map } from 'rxjs/operators';
4 @Component({
5   selector: 'my-app',
6   templateUrl: './app.component.html',
7   styleUrls: ['./app.component.css'],
8 })
9 export class AppComponent implements OnInit {
10   name = 'Angular';
11   AsyncStream(observer) {
12     var ti = setInterval(() => {
13       observer.next(Math.random() * 10);
14     }, 1000);
15   }
16   myobserv = Observable.create(this.AsyncStream);
17   mainstreamobservable = Observable.create(this.AsyncStream);
18   step1stream = this.mainstreamobservable.pipe(
19     map((x: number) => Math.round(x))
20   );
21   step2stream = this.step1stream.pipe(filter((x: number) => x > 4));
22   ngOnInit() {
23     // this.myobserv.subscribe((res) => this.Listener(res));
24     this.step2stream.subscribe((res) => this.Listener(res));
25   }
26   Listener(res: any) {
27     console.log(res);
28   }
29 }
30
```

The right side of the editor shows a preview of the application. It displays the text "Hello Angular!" and a message "Start editing to see some magic happen :)". Below the preview is a console window with 1015 entries. The console shows a sequence of numbers: 10, 9, 9, 9, 5, 5, 5, 5, 6, 5, 5, 8, 6.