RxJs Subjects

RxJs is used a lot in the **Angular** ecosystem, thus, it is important to know.

Subjects are one of the main concepts of RxJS.

A Subject is an **Observable**, a special type of Observable. Subjects can be **multicast** to multiple **observers**.

Subject

The **Subject**'s ecosystem is the following:

1) You **create** the Subject Observable.

2) You **subscribe** to the Subject.

3) You feed values thru the **.next()** method and **ONLY** then we are **emitting** those values.

Please look at the next page for a code example.

Subject

This is the code example of a Subject Observable.

```ts
export class TestSubjectsComponent implements OnInit {
  //step 1, we are creating the SubjectObservable
  thisIsMySubject = new Subject<number>();

  constructor() {}

  ngOnInit(): void {
    //This doesn't do anything because we aren't subscribed. :(
    this.thisIsMySubject.next(5);

    //step 2, we are subscribing to it, It will NOT emit any data when we subscribe.
    //But it will emit data once we feed it values thru the .next() method in the next step.
    this.thisIsMySubject.subscribe((value) => console.log(value));

    //step 3, we are  feeding it values and we are emiting data. :)
    //Expected output: 10
    this.thisIsMySubject.next(10);
  }
}
```

BehaviorSubject

The **BehaviorSubject**'s ecosystem is the following:
1) You **create** the BehaviorSubject Observable and give it a **value**.

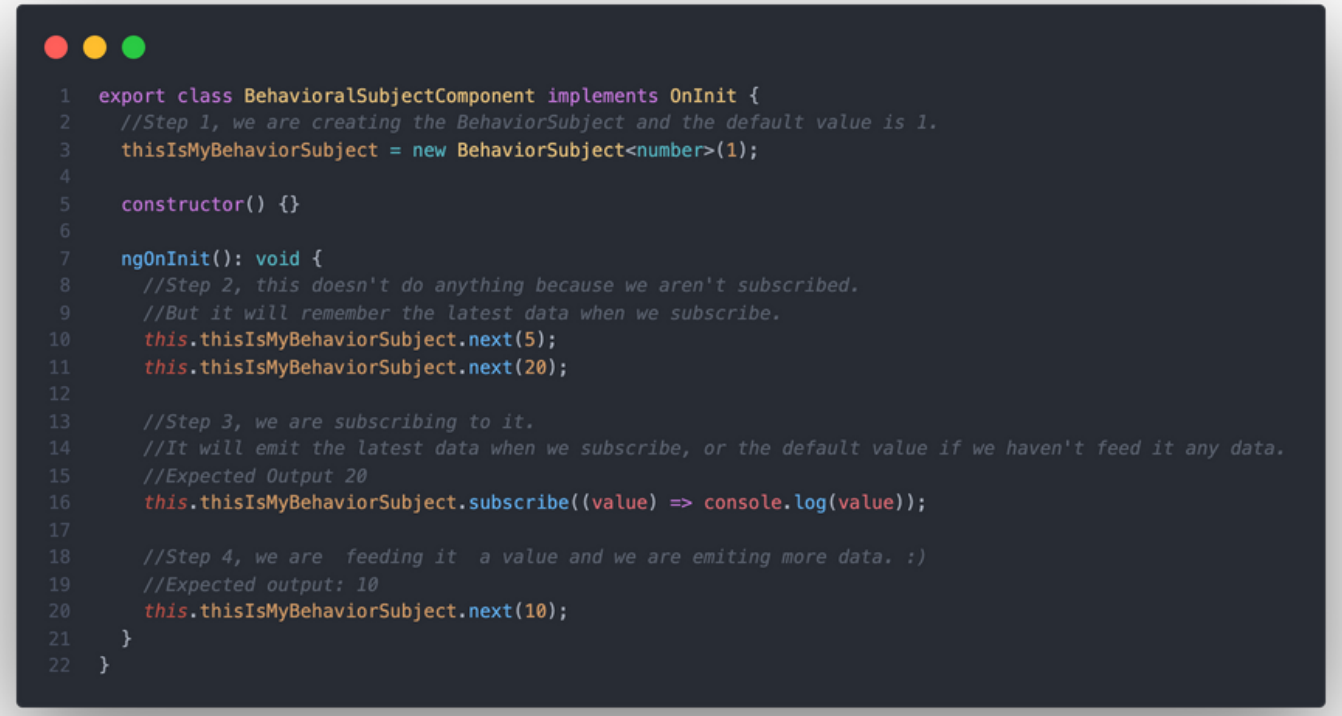
2) You can feed values with **.next()** however it will **NOT** emit because we aren't subscribed.

3)You **subscribe** to the BehaviorSubject and it will **EMIT** the latest value fed or the default value.

4) You can still feed values and it will emit them.

BehaviorSubject

This is the code example of a BehaviorSubject Observable.



```ts
export class BehavioralSubjectComponent implements OnInit {
  //Step 1, we are creating the BehaviorSubject and the default value is 1.
  thisIsMyBehaviorSubject = new BehaviorSubject<number>(1);

  constructor() {}

  ngOnInit(): void {
    //Step 2, this doesn't do anything because we aren't subscribed.
    //But it will remember the latest data when we subscribe.
    this.thisIsMyBehaviorSubject.next(5);
    this.thisIsMyBehaviorSubject.next(20);

    //Step 3, we are subscribing to it.
    //It will emit the latest data when we subscribe, or the default value if we haven't feed it any data.
    //Expected Output 20
    this.thisIsMyBehaviorSubject.subscribe((value) => console.log(value));

    //Step 4, we are  feeding it  a value and we are emiting more data. :)
    //Expected output: 10
    this.thisIsMyBehaviorSubject.next(10);
  }
}
```

ReplaySubject

The ReplaySubject's ecosystem is the following:
1) You **create** the ReplaySubject's Observable and configure it.

2) You can feed values with **.next()** but it will **NOT** emit because we aren't subscribed.

3) You **subscribe** to the ReplaySubject and it will **EMIT** the data given the configuration.

4) You can still feed it values and it will emit them.

ReplaySubject

This is the code example of a ReplaySubject Observable.

```
export class ReplaySubjectComponent implements OnInit {
  //Step 1, we are creating the ReplaySubject.
  //The first configuration that Replay subject takes is the bufferSize. Default is infinite.
  //The bufferSize will determine how many values are stored in the buffer.
  //The second configuration is the windowTime, The amount of time to hold a value in the buffer before removing it from the buffer.
  thisIsMyReplaySubject = new ReplaySubject<number>(2);

  constructor() {}

  ngOnInit(): void {
    //Step 2, this doesn't emit because we aren't subscribed.
    //But it will remember ALL the data when we subscribe.
    this.thisIsMyReplaySubject.next(2);
    this.thisIsMyReplaySubject.next(5);
    this.thisIsMyReplaySubject.next(20);

    //Step 3, we are subscribing to it.
    //It will emit the configured data when we subscribe.
    //Expected Output 5  then 20.
    this.thisIsMyReplaySubject.subscribe((value) => console.log(value));

    //Step 4, we are  feeding it  a value and we are emiting more data. :)
    //Expected output: 10
    this.thisIsMyReplaySubject.next(10);
  }
}
```

AsyncSubject

The **AsyncSubject's** ecosystem is the following:

1) You **create** the **AsyncSubject** Observable.

2) You **subscribe** to the Subject.

3) You feed values thru the **.next()** method.

4)It will emit the latest value once the AsyncSubject completes thru the **.complete()** method.

AsyncSubject

This is the code example of an AsyncSubject Observable.

```
export class AsyncSubjectComponent implements OnInit {
  //Step 1 , create the AsyncSubject
  thisIsMyAsyncSubject = new AsyncSubject<number>();

  constructor() {}

  ngOnInit(): void {
    //Step 2, we feed it some values.
    this.thisIsMyAsyncSubject.next(2);

    //Step 3, we are subscribing to it.
    this.thisIsMyAsyncSubject.subscribe((value) => console.log(value));

    //Step 4, we are  feeding it  more values.
    this.thisIsMyAsyncSubject.next(10);
    this.thisIsMyAsyncSubject.next(5);

    //AsyncSubject will emit data once the subject has been completed with the .complete() method.
    //Expected Output: 5 , which is the lastest value fed.
    this.thisIsMyAsyncSubject.complete();
  }
}
```