RxJS Subject:

*RxJS subject is a special type of observable that allows values to be multicast to many observers. While plain observables are unicast, subjects are multicast.*

Subject is the equivalent of event emitter and the only way of multicast in a value or event to multiple observers.

Subject implements both observable and observer interfaces

* Every subject is an observable so you can subscribe to it.
* At the same time, every subject is an observer, it means that you have next, error, and complete methods, so you can send values, error subject or completed.

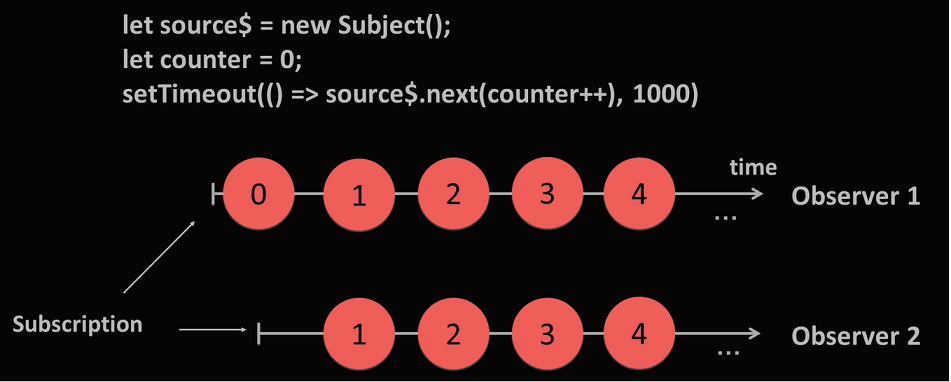
Types of Subjects

* Subject
* ReplaySubject
* BehaviorSubject
* AsyncSubject

These are a few types of subjects, let's review them one by one. :

Subject:

Subject simply creates observable that allows values to be multicast to many observers, each new observer is just starting to receive the next generated values, as shown in the diagram.

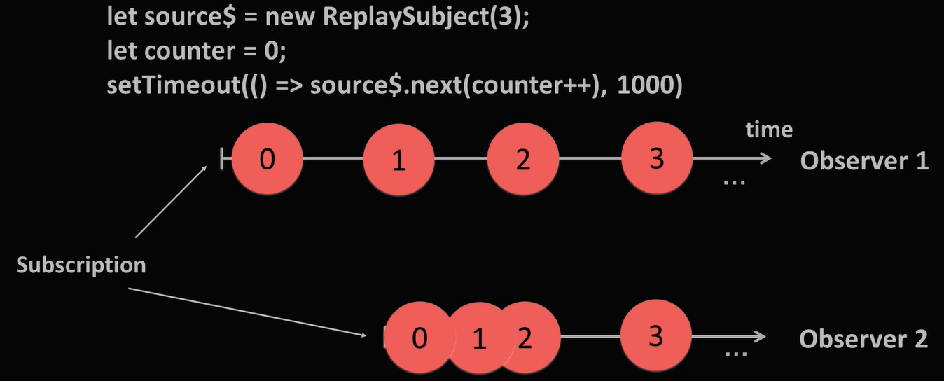


This has behavior as hot observables.

ReplaySubject:

Replay subjects can help with keeping a buffer of previous values that will be emitted to new subscribers.

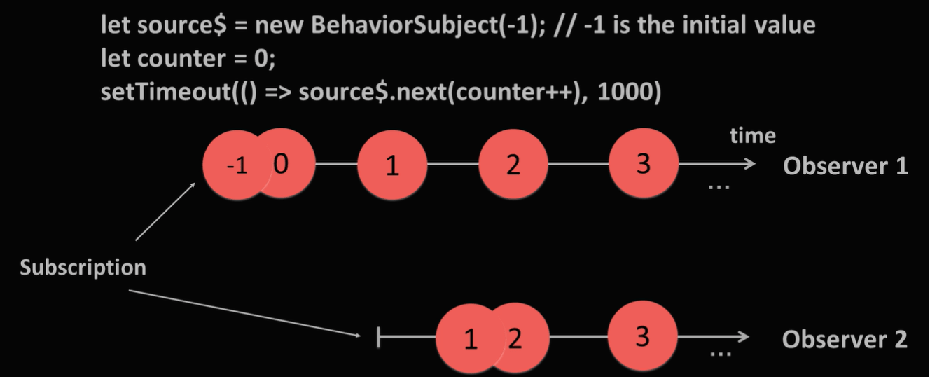
In a simple use case, you just have to provide buffer sites when you create your place subject instance.  
On this diagram, you can see how it works.



As you can see, the second observer received previous values as well.

BehaviorSubject:

* BehaviorSubject is similar to ReplaySubject but with buffer size equals one, so it can replay only one previous item.
* The only difference is that behavior subject has initial value that will be emitted first anyway.
* Stores the latest value emitted to its consumers
* New observer subscribes, it will immediately receive the "current value" from the BehaviorSubject

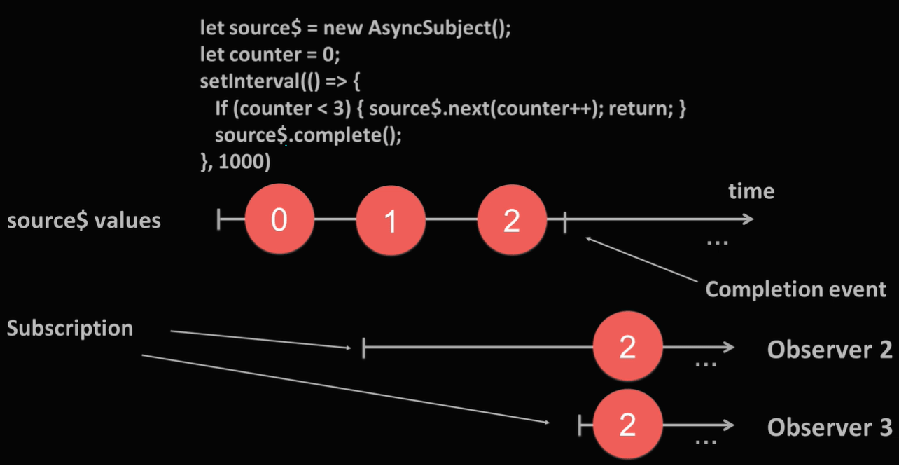
Here is a diagram where you can see how it works.

As you can see, observer 2, got only one previous value of the subscription.

AsyncSubject:

AsyncSubject differs from other types of subjects, it waits until completion and only after that, emits its last value.

Here is a diagram where you can see how it works.



As you can see here, observer 2 and observer 3 both got value only after source completion.

And one more thing to mention about Subject here, If you create a subject and want to provide it only for subscription, so no one else will be able to run next to generate values, you can use method named asObservable(). In that case, you can keep control of who will be able to emit new data.