Observable - An Introduction

- Observables are push based

- Observables are cold [by default]

- Observables can emit multiple values

- Observables can deliver both synchronous and asynchronous values

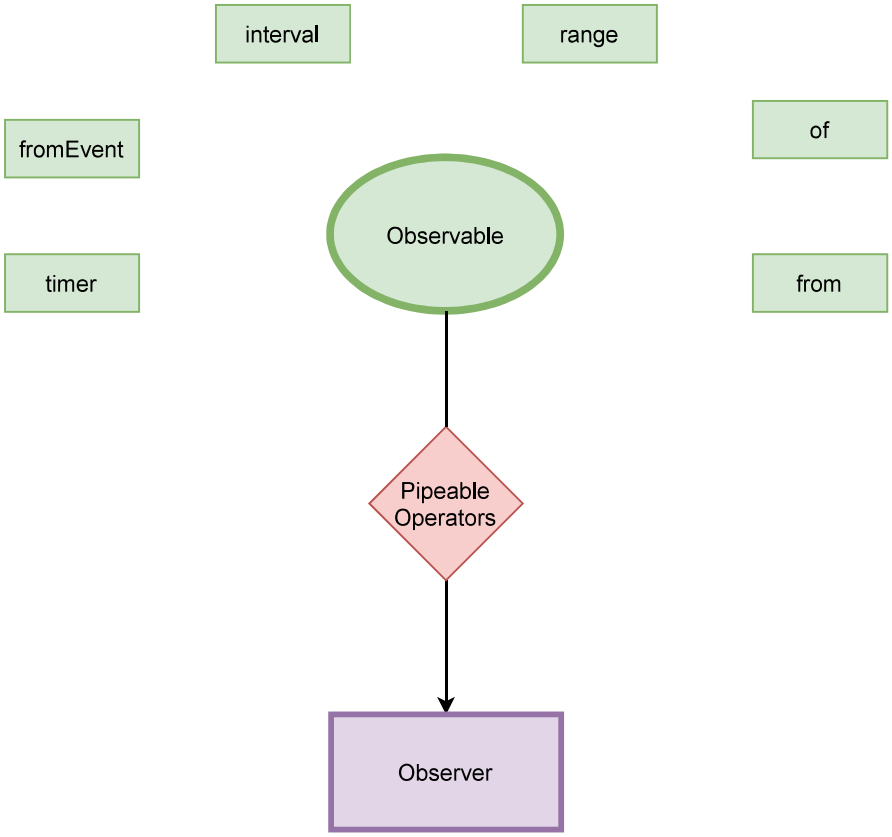
- Observables can be cancelled

Creational Operators

- Stand alone functions to create observables

- Sources can be :
    - **event** : fromEvent(document."click)
    - **request**: of("http://api.github.com/users/octocat")
    - **timer**: interval(1000)
    - **static data** : from ([1,2,3,4,5])
    - combination of other **observable** sources
    - + more...

Creational Operators



Creational Operators

fromEvent	creates observables from DOM events
of	creates observables from static values
from	creates observables from Array, iterators and Promises
interval / timer	Emits item based on a duration

Pipeable Operators

- Operators are the power behind RxJS, letting you more easily compose complex asynchronous code

- Operators can be applied by including them in the pipe() method.

- Operators return a new observable without modifying the input observable.

- A core set of Operators can solve the majority of use case, while others can be pickup up as the situation arises.

Filtering Operators

take	Emits a set number of values from stream
takeWhile	Completes a stream when a condition is met
takeUntil	Completes a stream based on another stream
distinct	Ignores NON unique values
filter	Ignores NOT needed values
reduce	Accumulates data over time
scan	Managing state changed incrementally

Rate Limiting Operators

debounceTime	Takes the latest value after a pause
throttleTime	Ignores values between windows/gap
sampleTime	Sample a stream on a uniform duration
auditTime	Audit a stream after a duration once event occurs

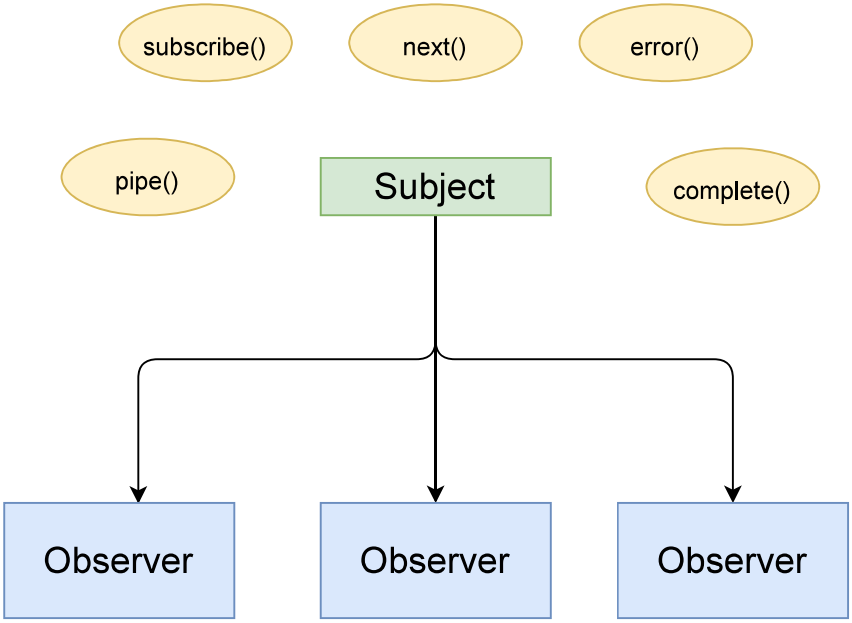
Transforming Operators

mergeMap	Flattening inner observable as they occurs
switchMap	Switch to a new observable on emissions
concatMap	Subscribe to observables in order
exhaustMap	ignore emissions when an inner observable is active

Combination Operators

startWith / endWith	Add values to the stream at start / end
concat	Queue observable emissions
merge	Combines multiple active observables
combineLatest	Receives the latest values from multiple observables on emissions
forkJoin	Receives the latest values from multiple observables on completion

Subjects are both - observable and observer



Types of Subject