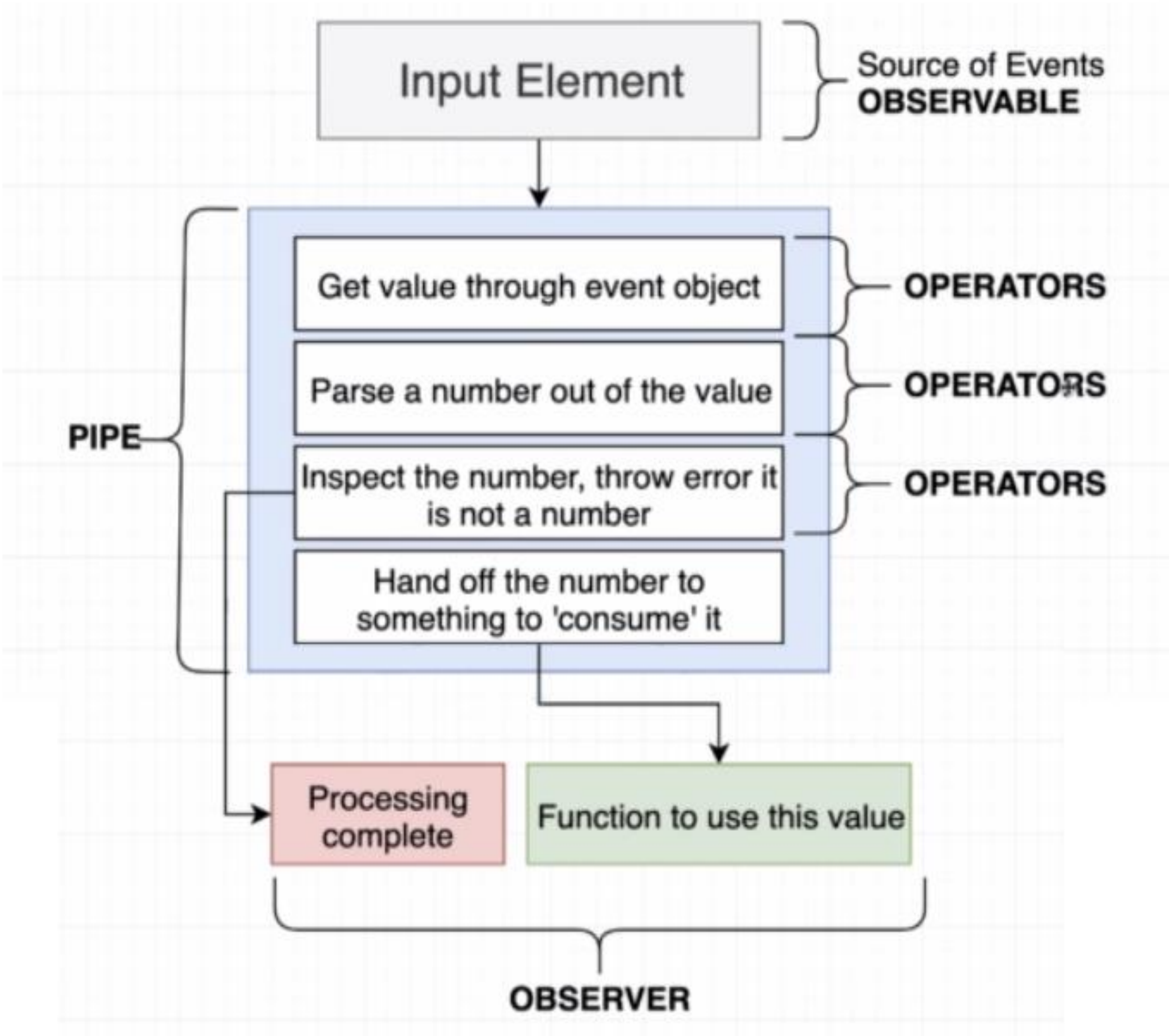
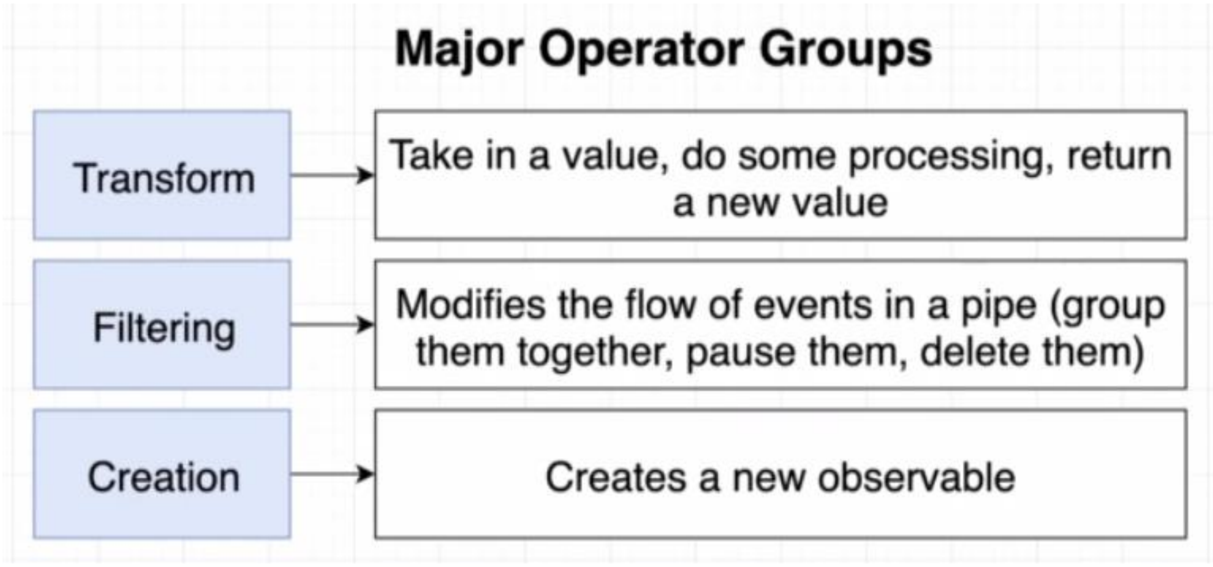
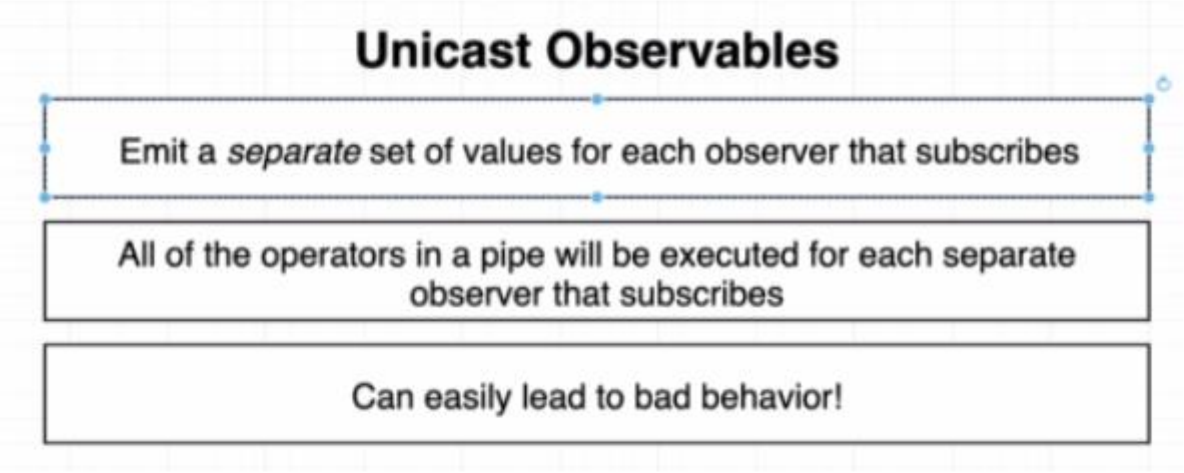
# RxJS Terminology


Input Element
Source of Events
OBSERVABLE
PIPE
Get value through event object
OPERATORS
Parse a number out of the value
OPERATORS
Inspect the number, throw error it is not a number
OPERATORS
Hand off the number to something to 'consume' it
Processing complete
Function to use this value
OBSERVER



## Major Operator Groups

| | |
|---|---|
| Transform | Take in a value, do some processing, return a new value |
| Filtering | Modifies the flow of events in a pipe (group them together, pause them, delete them) |
| Creation | Creates a new observable |



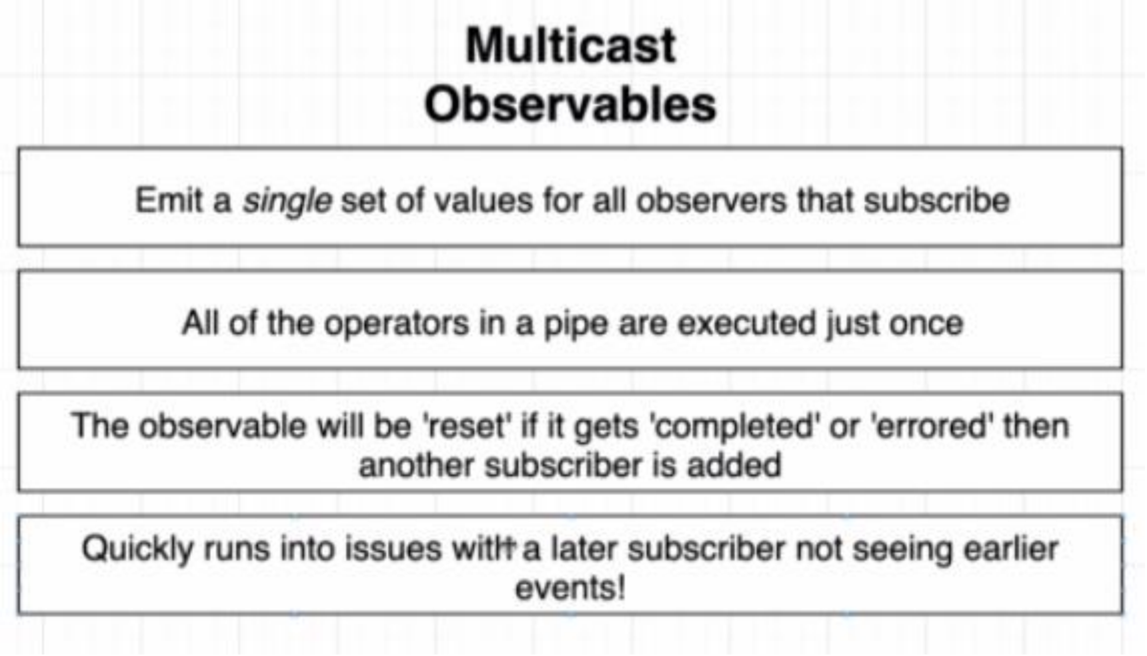
## Unicast Observables

Emit a *separate* set of values for each observer that subscribes

All of the operators in a pipe will be executed for each separate observer that subscribes

Can easily lead to bad behavior!

Unicast Observables are bad: for example when Observable has HttpRequest. For each subscribe call, a new HttpRequest will be called.



## Multicast Observables

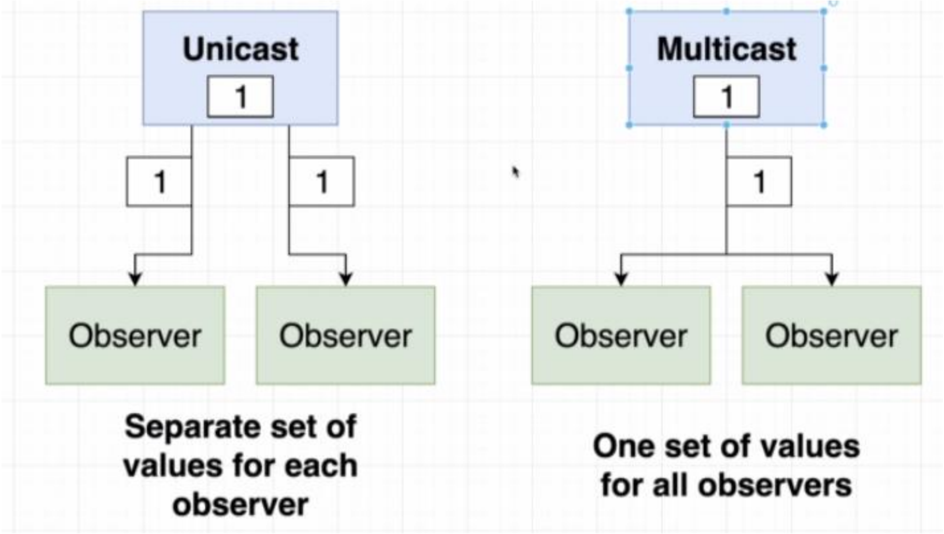
Emit a *single* set of values for all observers that subscribe

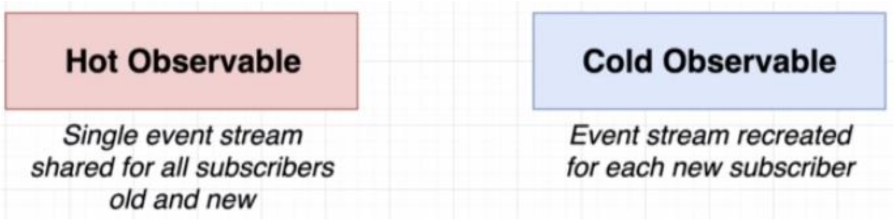
All of the operators in a pipe are executed just once

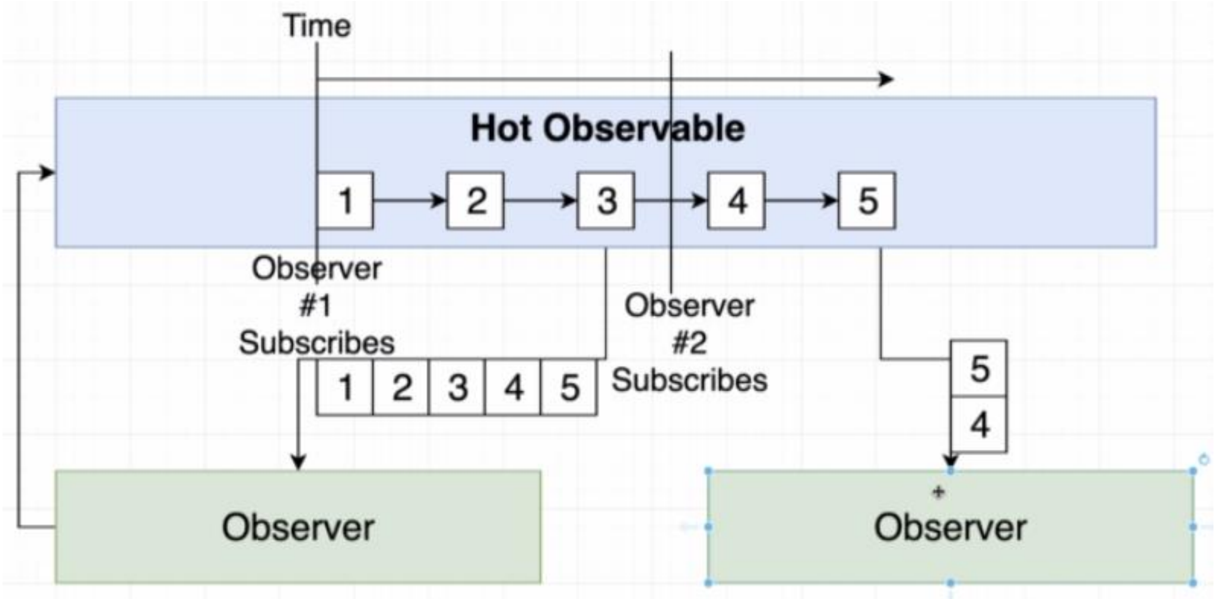
The observable will be 'reset' if it gets 'completed' or 'errored' then another subscriber is added

Quickly runs into issues with a later subscriber not seeing earlier events!

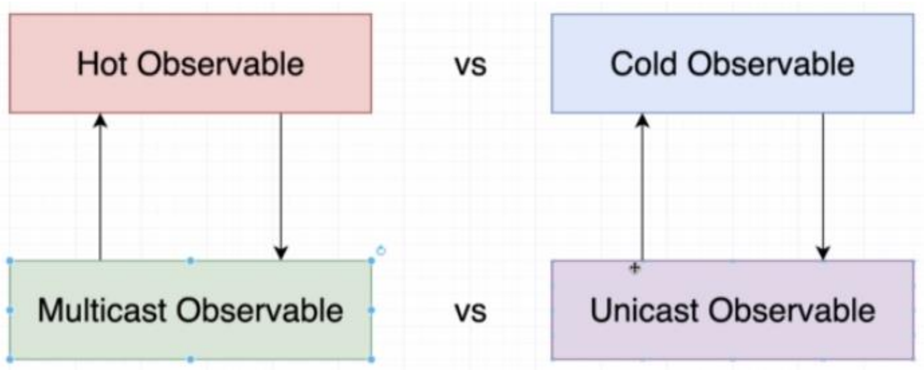
Multicas Observables are using **share()** method inside the **.pipe(...)**


Unicast
1
1
1
Observer
Observer
Separate set of values for each observer
Multicast
1
1
Observer
Observer
One set of values for all observers


Hot Observable
Single event stream shared for all subscribers old and new
Cold Observable
Event stream recreated for each new subscriber



The second Observer starts getting future values compared to Observable #1


Hot Observable
vs
Cold Observable
Multicast Observable
vs
Unicast Observable