Observable : Observables are lazy Push collections of multiple values

Observer: An Observer is a consumer of values delivered by an Observable.

What is a Subscription? : A Subscription is an object that represents a disposable resource, usually the execution of an Observable. A Subscription has one important method, unsubscribe, that takes no argument and just disposes the resource held by the subscription. In previous versions of RxJS, Subscription was called "Disposable".

Subject : A Subject is like an Observable, but can multicast to many Observers. Subjects are like EventEmitters: they maintain a registry of many listeners.

What is a Scheduler? A scheduler controls when a subscription starts and when notifications are delivered. It consists of three components.

- A Scheduler is a data structure. It knows how to store and queue tasks based on priority or other criteria.
- A Scheduler is an execution context. It denotes where and when the task is executed (e.g. immediately, or in another callback mechanism such as setTimeout or process.nextTick, or the animation frame).
- A Scheduler has a (virtual) clock. It provides a notion of "time" by a getter method now() on the scheduler. Tasks being scheduled on a particular scheduler will adhere only to the time denoted by that clock.

Observable operators:

Empty : Creates an Observable that emits no items to the Observer and immediately emits a complete notification.

From: Creates an Observable from an Array, an array-like object, a Promise, an iterable object, or an Observable-like object.

FromEvent: Creates an Observable that emits events of a specific type coming from the given event target

OF : Converts the arguments to an observable sequence.

Range : Creates an Observable that emits a sequence of numbers within a specified range.

Timer : Creates an Observable that starts emitting after an dueTime and emits ever increasing numbers after each period of time thereafter.

Concat : Concatenates multiple Observables together by sequentially emitting their values, one Observable after the other.

Merge: Flattens multiple Observables together by blending their values into one Observable.

GroupBy : Groups the items emitted by an Observable according to a specified criterion, and emits these grouped items as GroupedObservable

Map: Like Array.prototype.map(), it passes each source value through a transformation function to get corresponding output values.

Pluck : Like map, but meant only for picking one of the nested properties of every emitted object.

SwitchMap : Maps each value to an Observable, then flattens all of these inner Observables.

Filter : Like Array.prototype.filter(), it only emits a value from the source if it passes a criterion function.

First: Emits only the first value. Or emits only the first value that passes some test.

Last : Returns an Observable that emits only the last item emitted by the source Observable.

Single : Returns an Observable that emits the single item emitted by the source Observable that matches a specified predicate, if that Observable emits one such item.

Skip : Returns an Observable that skips the first count items emitted by the source Observable.

Take : Emits only the first count values emitted by the source Observable.

Find : Finds the first value that passes some test and emits that.

FindIndex : It's like find, but emits the index of the found value, not the value itself.

IsEmpty : Tells whether any values are emitted by an observable

Count : Tells how many values were emitted, when the source completes.

Max : The Max operator operates on an Observable that emits numbers

Min: The Min operator operates on an Observable that emits numbers