Reactive Programming:

<https://angular-templates.io/tutorials/about/learn-angular-from-scratch-step-by-step>

<https://blog.logrocket.com/understanding-rxjs-observables/>

Javascript is a single threaded language. This means it has one call stack and one memory heap.

**Asynchronous programming** is a means of parallel **programming** in which a unit of work runs separately from the main application thread and notifies the calling thread of its completion, failure or progress.

To provide better user experience.To make the application more responsive. Reactive programming works on asynchronous data flow.

RXJS library in angular provides reactive programming.

RXJS=Observable+Objerver+Schedulers

**Observable:** Observable are nothing but the data streams. Observable packs the data that can be passed around from one thread to another thread. They basically emit the data periodically or only once in their life cycle based on their configurations. There are various operators that can help observer to emit some specific data based on certain events, but we will look into them in upcoming parts. For now, you can think observables as suppliers. They process and supply the data to other components.

**Observers:** Observers consumes the data stream emitted by the observable. Observers subscribe to the observable using subscribeOn() method to receive the data emitted by the observable. Whenever the observable emits the data all the registered observer receives the data in onNext() callback. Here they can perform various operations like parsing the JSON response or updating the UI. If there is an error thrown from observable, the observer will receive it in onError().

**Schedulers:**Remember that Rx is for asynchronous programming and we need a thread management. There is where schedules come into the picture. Schedulers are the component in Rx that tells observable and observers, on which thread they should run. You can use observeOn() method to tell observers, on which thread you should observe. Also, you can use scheduleOn() to tell the observable, on which thread you should run. There are main default threads are provided in RxJava like Schedulers.newThread() will create new background that. Schedulers.io() will execute the code on IO thread.

Observable returns a stream of values over time while a Promise is only called once and returns a single value.

Variable$ : $ sign is used to indicate that the variable is observable type

**Communicating Between Components with Observable & Subject**

## Observable.subscribe()

The observable subscribe method is used by angular components to subscribe to messages that are sent to an observable.

## Subject.next()

The subject next method is used to send messages to an observable which are then sent to all angular components that are subscribers (a.k.a. observers) of that observable.

## Why Observables are so vital

* Emitting multiple values asynchronously is very easily handled with Observables
* Error handlers can also easily be done inside Observables rather than a construct like promises
* Observables are considered lazy, so in case of no subscription there will be no emission of data values
* Observables can be resolved multiple times as opposed to functions or even promises