Services and RxJS

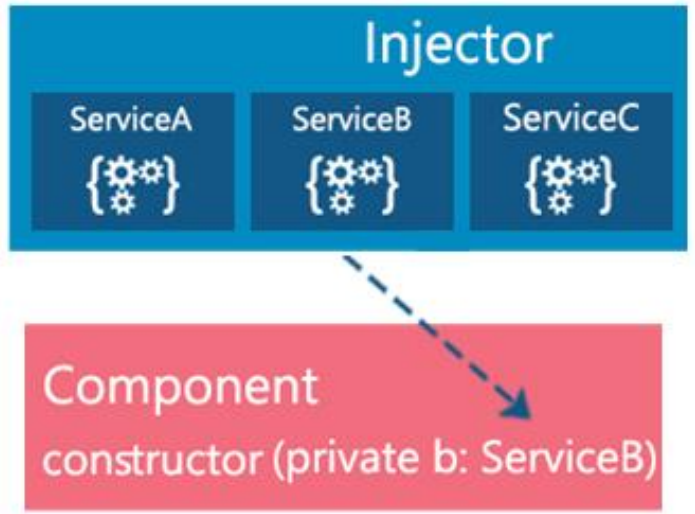
DotNetTricks

Services

- A service is used to share functionalities among components
- A service is injected using Dependency Injection mechanism
- For example:
  - Logging service
  - Data service
  - Application configuration
- A service will be singleton when you register it at *AppModule*

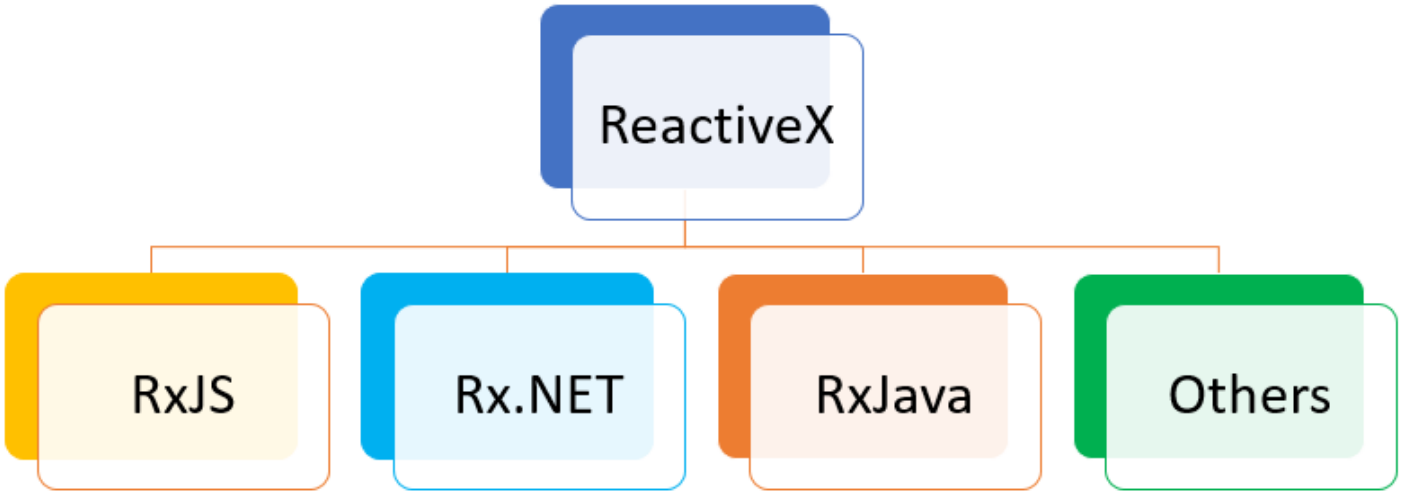
Dependency Injection

- A way to supply a new instance of a class with the fully-formed dependencies it needs
- Most commonly used dependencies are services
- Angular uses dependency injection for providing service instance to a component
- Angular2+ has a Hierarchical Dependency Injection system



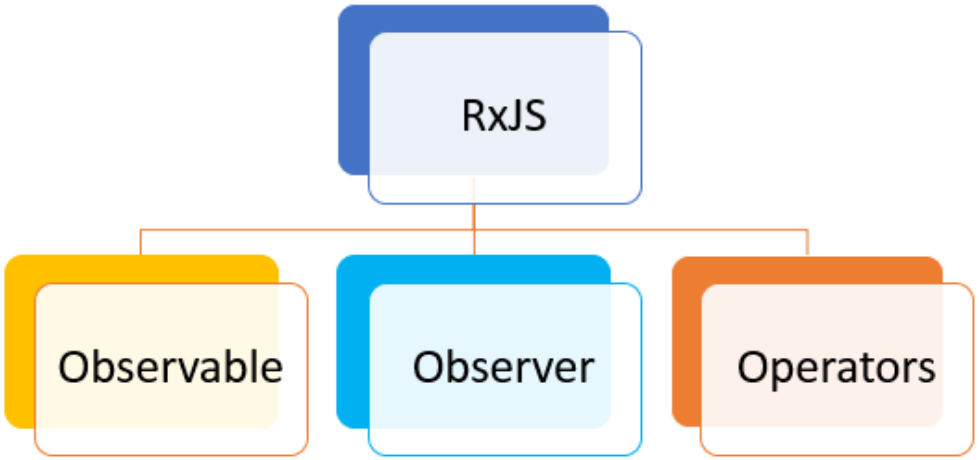
ReactiveX

- An API for asynchronous programming with observable streams
- ReactiveX is a combination of the best ideas from Observer pattern, Iterator pattern and functional programming
- ReactiveX extension are available for :
  - JS
  - .NET
  - Java
  - Scala etc.



RxJS - Reactive Extensions Library

- Angular comes with RxJS library to deal with async data streams
- Make easier to compose asynchronous or callback code
- Supports multiple operators to transform the stream data
- Uses Observable object, Observer object & Observable operators
- Angular uses Observable with HTTP service and Event system



DotNetTricks

Observable (RxJS)

- An observable is just a function
- Takes an observer with *next()*, *error()* and *complete()* methods on it, and returns cancellation logic
- An observable instance starts publishing the values only when someone subscribes to it
- An observable can be subscribed by calling the *subscribe()* method and passing an observer object to receive the notifications

Observer (RxJS)

- An observer is a consumer of values delivered by an observable at subscriber end
- An observer can have a set of callbacks, one for each type of notification delivered by an observable:
  - next
  - error
  - complete

Observable Operators (RxJS)

- Observable operators are just functions
- Takes an observable or collection as a source, do manipulation and returns a new observable

AREA	Operators
Creation	of, from , fromPromise , fromEvent
Combination	concat , merge , startWith , zip
Filtering	filter , take , takeUntil, debounceTime
Transformation	map , mergeMap , scan , switchMap,