RxJS in Angular

What is RxJS?

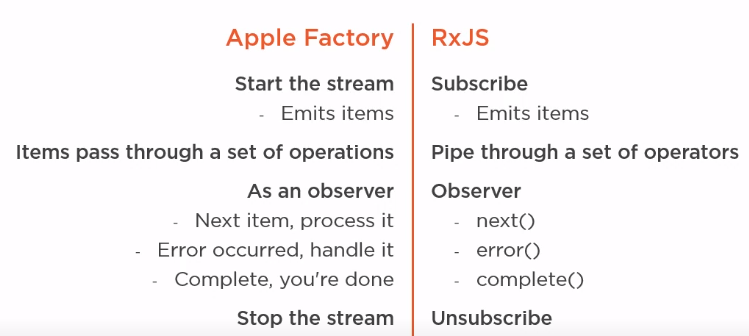
* Reactive extension for JavaScript
* Help us better manage and manipulate the data in our application

What is Reactive development?

* Quick to react to user interactions
* Resilient to the future
  + Error handling
* Reactive to state changes

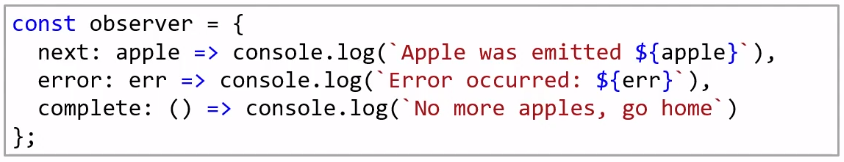
RxJS Terms and Syntax

Processing Observable Streams



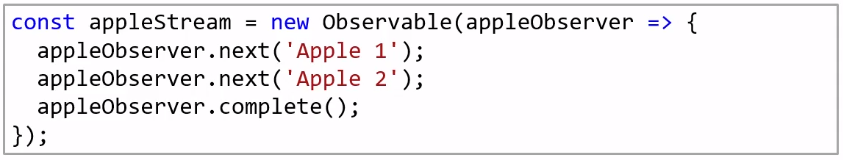
Observer

* In RxJS, we define an observer as an object that monitors the stream
* RxJS Observer observes the stream and responds to its notifications
* Think set of callbacks to observe, handling next, error and complete



Observable stream

* Any stream of data, optionally produced over time
* Observables can be synchronous or asynchronous



Subscription

* We start the stream by calling the subscribe method of the observable
* Must subscribe to start the Observable



Stopping the stream

* Properly stopping each observable stream helps avoid, potential memory leaks in our applications
* **Call the complete method**
  + Automatically unsubscribes
* **Some creation functions also automatically unsubscribe and calls the complete method**
  + of, from and take
* **Any error unsubscribes from the stream but without calling the complete method**
* **Calling the unsubscribe method also stops the stream but it not calls the complete method**

Creating an Observable

* Angular recommends to use built-in creation functions
  + **of**
    - creates an observable using a set of defined values, emitting each value then completing the stream



* + **from**
    - From creates an observable from an array or other data structure, emitting each individual value from that structure and then completing the stream



* + **fromEvent**
    - Creates an observable from any document, object model or DOM event
  + **interval**
    - It creates an observable that emits a sequential number at a defined interval

RxJS Operators

Overview

* RxJS operator is a function
* Used to transform and manipulate items in an Observable stream
* Apply operators in sequence using the Observable’s pipe method
* Documentation -> [www.rxjs.dev](http://www.rxjs.dev)

RxJS Operator: map

* **map is a transformation operator**
  + Takes in an input stream, subscribes
  + Creates an output stream
* **When an item is emitted**
  + Item is transformed as specified by a provided function
  + Item is emitted to the output stream

RxJS Operator: tap

* **Performs an operation that does not affect the stream**
* **Used for:**
  + Debugging
  + Performing actions outside of the flow of data

RxJS Operator: take

* **Emits a specified number of items**
* **Used for:**
  + Taking a specified number of items
  + Limiting unlimited streams

RxJS Operator: catchError

* **catchError is an error handling operator**
  + Takes in an input stream, subscribes
  + Creates an output stream
* **When a source item is emitted**
  + Item is emitted to the output stream
* **If an error occurs**
  + Catches the error
  + Unsubscribes from the input stream
  + Returns a replacement Observable
  + Optionally rethrows the error

RxJS Operator: throwError

* **Creates an observable that emits no items**
* **And immediately emits an error notification**
* **Used for:**
  + Propagating an error

Going Reactive

Working with Async pipe

* Subscribes to the Observable when component is initialized
* Returns each emitted value from that Observable
* When a new item is emitted, the component is marked to be checked for changes and runs change detection
* Unsubscribes when the component is destroyed

RxJS Constant: EMPTY

* **Creates an Observable that emits no items**
* **Immediately emits a complete notification**
* **Used for:**
  + Returning an empty Observable