**Promise**

A Promise handles a **single event** when an async operation completes or fails.

Note: There are Promise libraries out there that support cancellation, but

ES6 Promise doesn't so far.

=>

**Observable**

An Observable is like a **Stream** (in many languages) and allows to pass zero or more events where the callback is called for each event.

Often Observable is preferred over Promise because it provides the features of Promise and more. With Observable it doesn't matter if you want to handle 0, 1, or multiple events. You can utilize the same API in each case.

Observable also has the advantage over Promise to be **cancelable**. If the result of an HTTP request to a server or some other expensive async operation isn't needed anymore, the Subscription of an Observable allows to cancel the subscription, while a Promise will eventually call the success or failed callback even when you don't need the notification or the result it provides anymore.

Observable provides **operators** like map, forEach, reduce, ... similar to an array

**Few Problems With Promises**

* A Promise is defined where the data is created, not where it is being consumed.
* As your application gets bigger, Promises become hard to manage.
* What if I want to retry a failed call? Now we are back to callback hell again.

**Then Observables Arrived**

RxJS is all about unifying the ideas of Promises, callbacks and data flow, and making them easier to work with.

An Observable is an array or a sequence of events over time. It has at least two participants. The creator (the data source) and the subscriber (subscription – where data is being consumed).

For example, let data = http.get(‘/api.json’). In Angular, data is going to be an Observable of responses, because the HTTP.get method returns a Promise.

## Get an API Key From Google Console

To interact with the YouTube API, we need to get an API Key from the Google Console, so navigate here and follow the steps to obtain your API Key. You will have to create an app to obtain API credentials for it. This will enable our app to submit API requests to the YouTube API.

## Query the YouTube API

The YouTube API doesn’t return a direct list of results, but rather a group of metadata where it has one property called Items that contains the list of results.

Next, let’s build our form in the src/app/app.component.ts file. Add the following code:

import { Component } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { FormBuilder, Validators, FormGroup } from '@angular/forms';

import { Observable } from 'rxjs/Observable';

import API\_KEY from './api-key';

const API\_URL = 'https://www.googleapis.com/youtube/v3/search';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'Angular RxJs YouTube Searcher';

searchForm: FormGroup;

results: any;

constructor(private formBuilder: FormBuilder, private http: HttpClient) {

this.searchForm = this.formBuilder.group({

search: ['', Validators.required]

});

this.searchForm.controls.search.valueChanges.subscribe(result => console.log(result));

}

}