



# Angular Fundamentals

## Module 4 – Observables



***Rabobank***



Peter Kassenaar  
[info@kassenaar.com](mailto:info@kassenaar.com)



# **Async services with RxJS/Observables**

Reactive programming  
with asynchronous streams

# Async Services

- Fetching static data: *synchronous* action
- Working via `HttpClient`: *asynchronous* action
- Angular 1: `Promises`
- Angular 2: `Observables`

Angular : ReactiveX library RxJS

[Introduction](#)[Docs ▾](#)[Languages ▾](#)[Resources ▾](#)[Community ▾](#)

# ReactiveX

An API for asynchronous programming  
with observable streams

Choose your platform

[Introduction](#)[Docs ▾](#)[Languages ▾](#)[Resources ▾](#)[Community ▾](#)

## Languages

- [Java: RxJava](#)
- [JavaScript: RxJS](#)
- [C#: Rx.NET](#)
- [C#\(Unity\): UniRx](#)
- [Scala: RxScala](#)
- [Clojure: RxClojure](#)
- [C++: RxCpp](#)
- [Ruby: Rx.rb](#)
- [Python: RxPY](#)
- [Groovy: RxGroovy](#)
- [JRuby: RxJRuby](#)
- [Kotlin: RxKotlin](#)
- [Swift: RxSwift](#)

## ReactiveX for platforms and frameworks

- [RxNetty](#)
- [RxAndroid](#)
- [RxCocoa](#)

<http://reactivex.io/>

### DOCUMENTATION

[Observable](#)[Operators](#)[Single](#)[Subject](#)

### LANGUAGES

[RxJava <sup>ℹ</sup>](#)[RxJS <sup>ℹ</sup>](#)[Rx.NET <sup>ℹ</sup>](#)[RxScala](#)

### RESOURCES

[Tutorials](#)

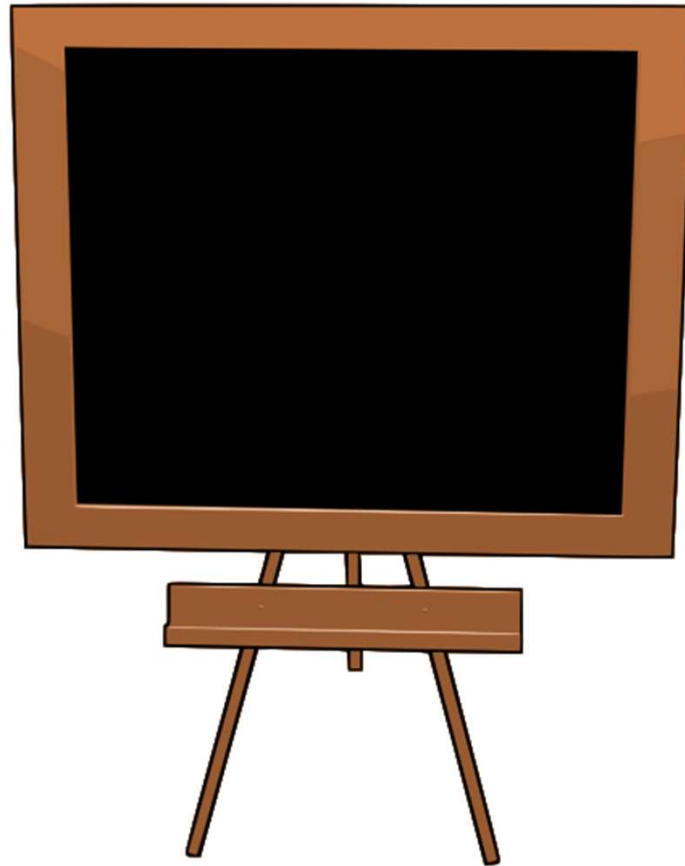
### COMMUNITY

[GitHub <sup>ℹ</sup>](#)[Twitter <sup>ℹ</sup>](#)[Others](#)

# ***The observable design pattern***

*"Understanding the  
observable stream"*

# Explanation



# Why Observables?

*"We can do **much more** with observables than with promises.*

*With observables, we have a whole bunch of operators to pull from, which let us customize our streams in nearly any way we want."*

<https://auth0.com/blog/2015/10/15/angular-2-series-part-3-using-http/>

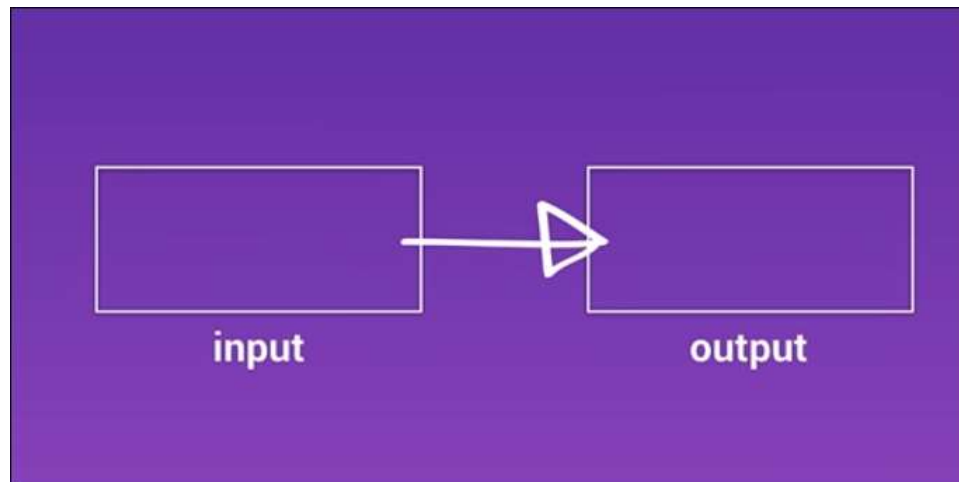
# Observables and RxJs

- “Reactive Programming”
  - *“Reactive programming is programming with asynchronous data streams.”*
  - <https://gist.github.com/staltz/868e7e9bc2a7b8c1f754>
- Observables have a lot of extra options, as opposed to Promises
  - Mapping
  - Filtering
  - Combining
  - Cancel
  - Retry
  - ...
- Consequence: no more `.success()`, `.error()` and `.then()` chaining!



# How do observables work

- First - The Observable Stream
- Later - all 10.000 operators...
- Traditionally:





The image shows a YouTube video player interface. The video content features a man in a black shirt standing at a podium with the Angular Connect logo. Behind him is a large screen with the Angular Connect logo and the text "ANGULAR CONNECT". To the right, there is a banner for "RANGLE.IO" with the tagline "REWRITING THE W". Below this, there are logos for "Progre", "energy", "cSynergy", "prohire", and "MONACA". A subtitle overlay reads: "Welcome to go beast mode with realtime interactive interfaces in Angular and Firebase." The video player controls at the bottom show a play button, a progress bar at 0:06 / 25:21, and icons for volume, settings, and full screen.

ANGULAR CONNECT

RANGLE.IO  
REWRITING THE W

Progre

energy

cSynergy

prohire

MONACA

ANGULAR BOOT CAMP  
STL - SF - DC - NYC  
LONDON - ONLINE

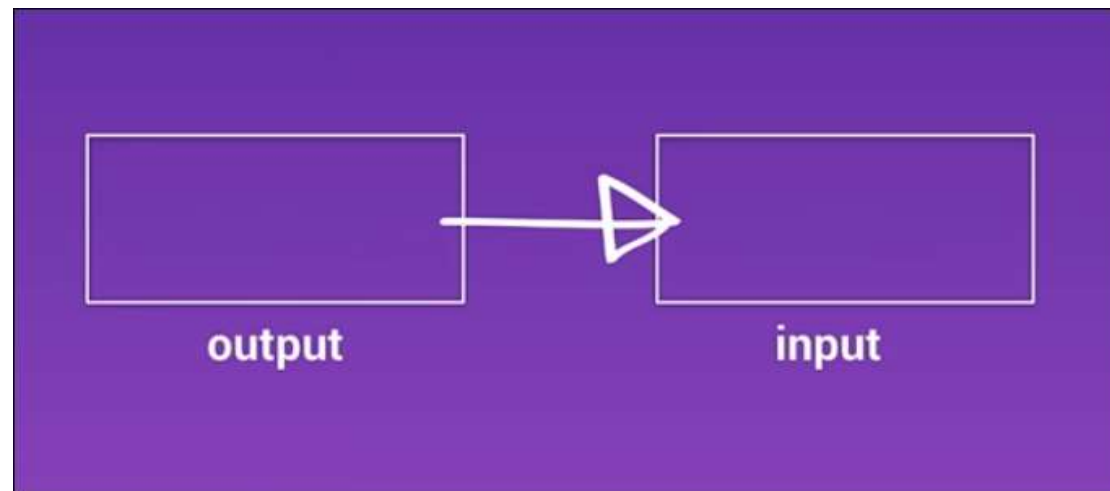
Welcome to go beast mode with realtime interactive interfaces in Angular and Firebase.

0:06 / 25:21

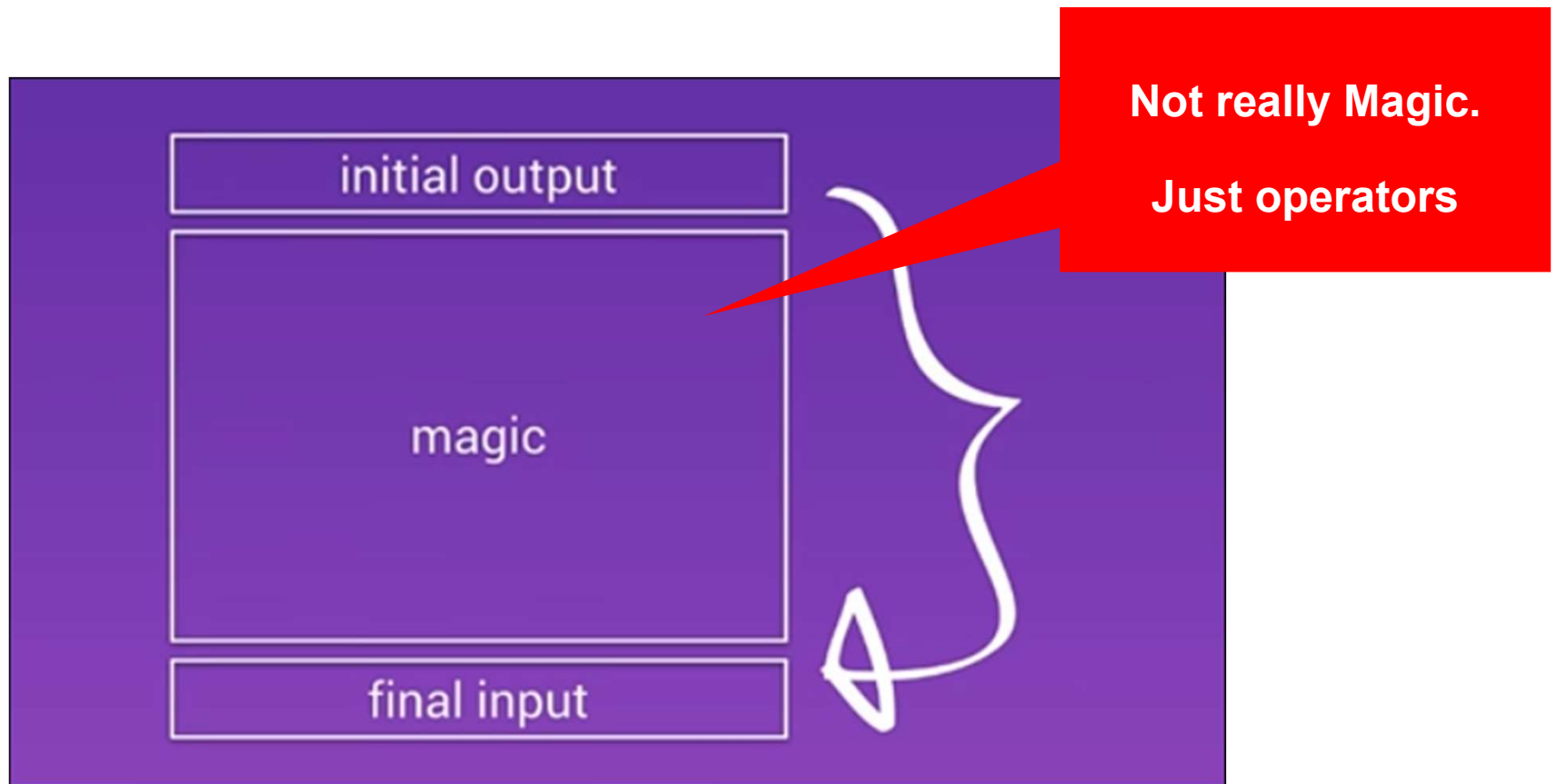
Go beast mode with realtime reactive interfaces in Angular 2 & Firebase | Lukas Ruebbelke

<https://www.youtube.com/watch?v=5CTL7aqSvJU>

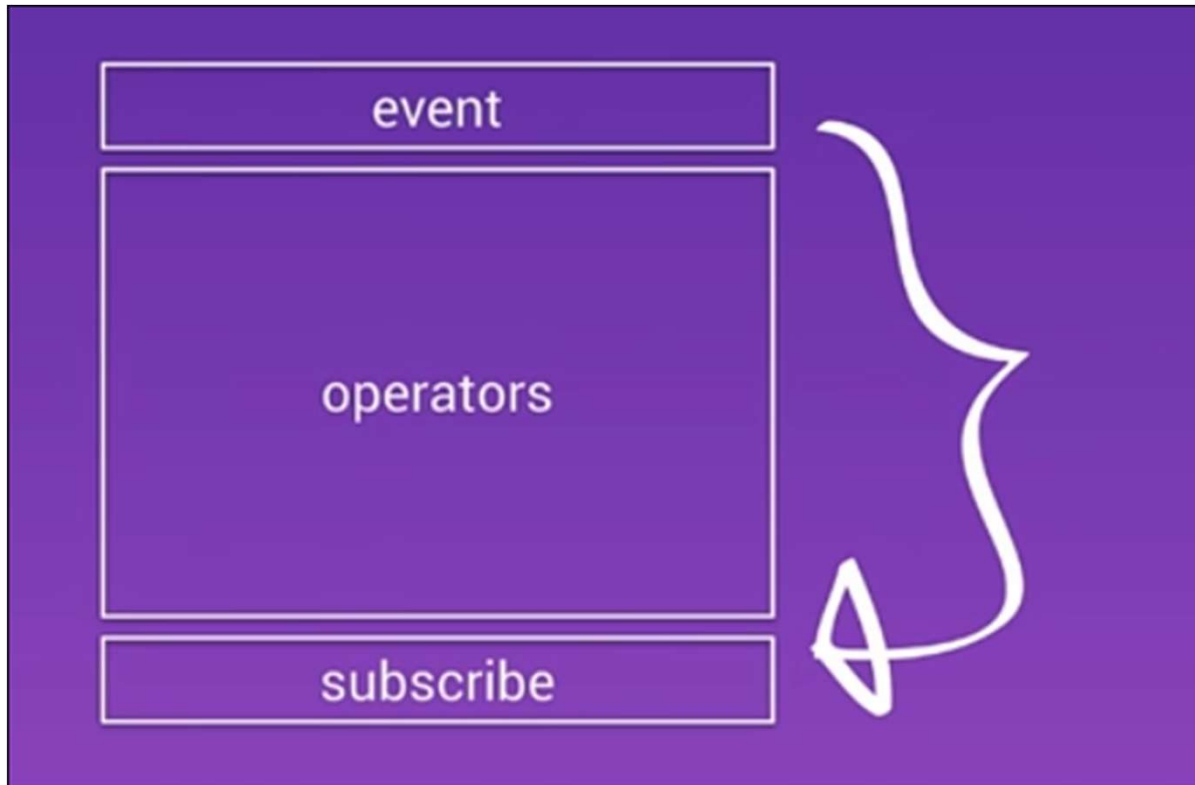
- With Observables -
  - a system, already outputting data,
  - Subscribe to that data
- "trade Output for Input"
- "Push vs. Pull"



# "The observable sandwich"



# "The Observable Sandwich"



**In code, for http-call:**

**Initial Output**

```
this.http.get<City[]>( 'assets/data/cities.json' )  
  .pipe(  
    filter(...),  
    map(...)  
  )  
  .subscribe((result) => {  
    //... Do something  
  });
```

**Optional:  
operator(s)**

**Final Input**

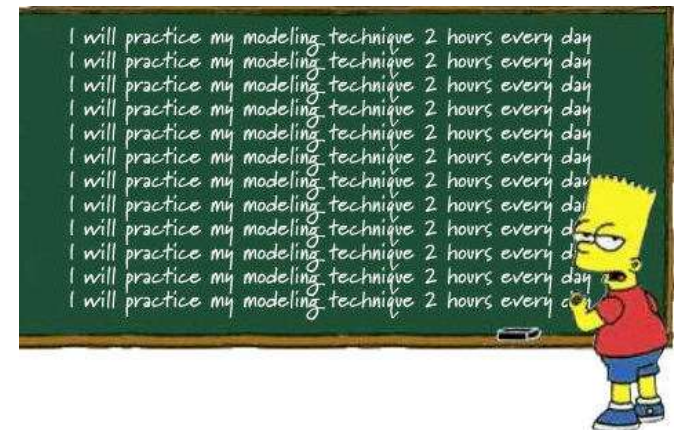
# Import HttpClientModule in @ngModule – don't forget

- *// Angular Modules*  
...
- **import** {HttpClientModule} **from** '@angular/common/http';  
*// Module declaration*  
@NgModule({  
    **imports** : [BrowserModule, HttpClientModule],  
    declarations: [AppComponent],  
    bootstrap : [AppComponent],  
    providers : [CityService] *// DI voor service*  
})  
**export class** AppModule {  
}

# Exercise

- See the example in `/201_services_http`
- Create your own `.json`-file and import it in your application.
- Exercise 5c) , 5d)

## Exercise....







# More on subscriptions

Using parameters inside the subscriber

# Subscribe - only once per block!

- Part of RxJs
- Three parameters:
  - `success()`
  - `error()`
  - `complete()`

```
this.cityService.getCities()  
  .subscribe(cityData => {  
    this.cities = cityData  
  },  
  err => console.log(err),  
  ()=> console.log('Getting cities complete...')  
)
```

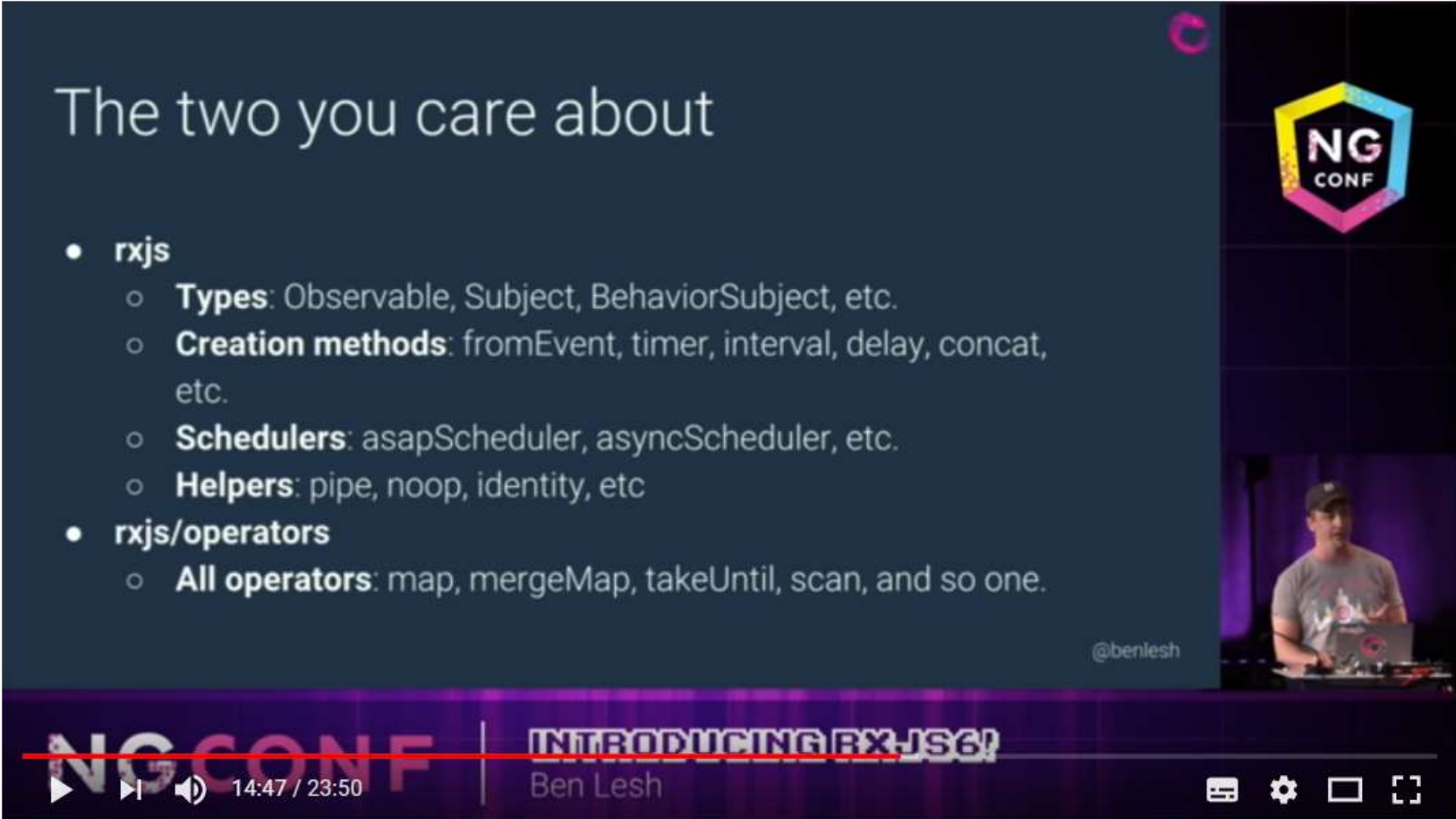


# Pipeable operators

- In RxJS 6.x and up: all operators inside `.pipe()` function
- The parameters of pipe function are the operators!
- Comma-separate different operators.
  - Don't forget `import {...} from 'rxjs/operators';`

```
.pipe(  
  delay(3000),  
  retry(3),  
  map(result => ...),  
  takeUntil(...condition...)  
)
```

# Background info: Ben Lesh on observables in RxJS 6.0



The video player shows a presentation slide with the title "The two you care about". The slide content is as follows:

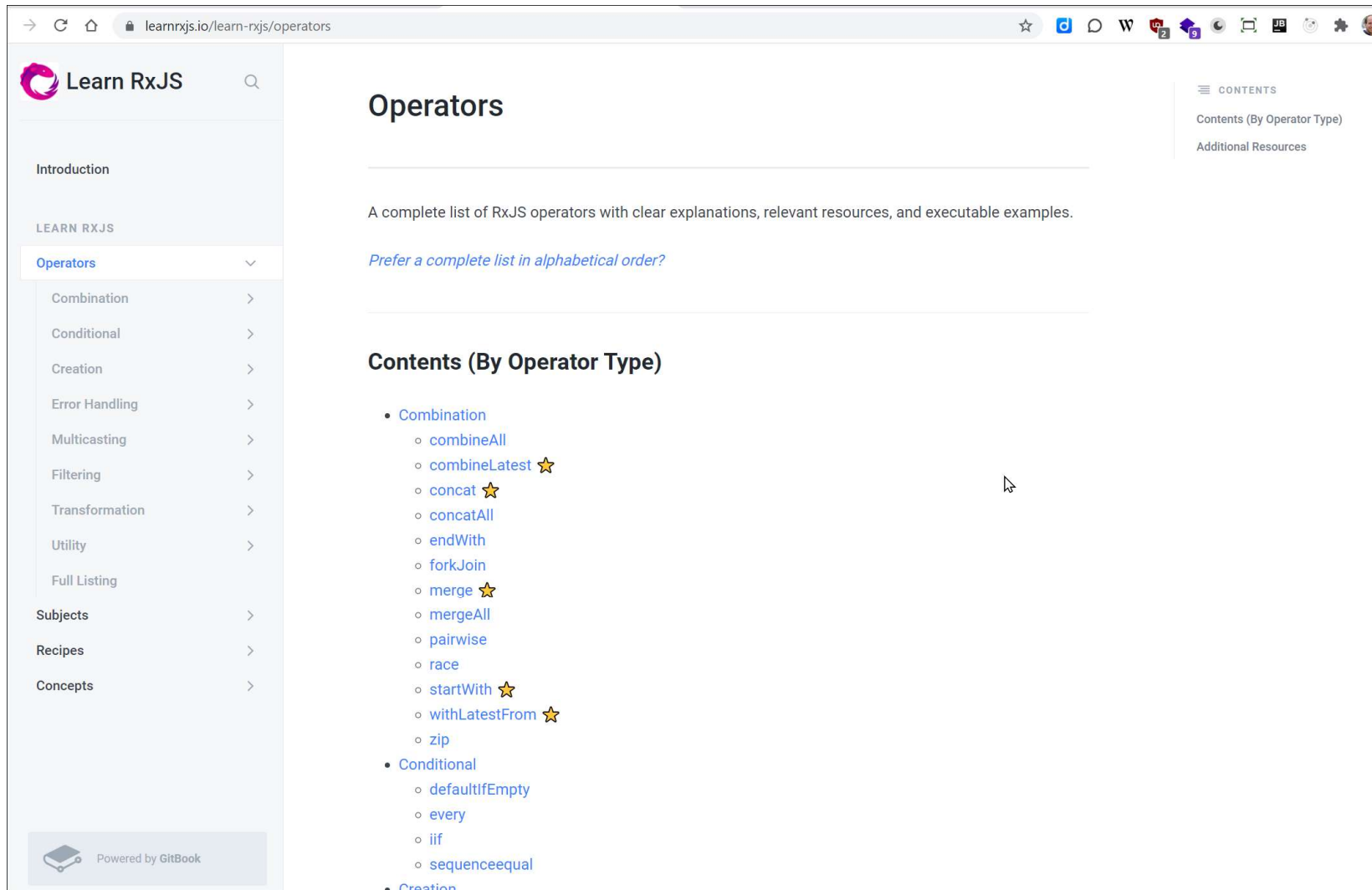
- **rxjs**
  - **Types:** Observable, Subject, BehaviorSubject, etc.
  - **Creation methods:** fromEvent, timer, interval, delay, concat, etc.
  - **Schedulers:** asapScheduler, asyncScheduler, etc.
  - **Helpers:** pipe, noop, identity, etc
- **rxjs/operators**
  - **All operators:** map, mergeMap, takeUntil, scan, and so one.

The video player interface includes a progress bar at 14:47 / 23:50, the title "Introducing RxJS6!", the speaker's name "Ben Lesh", and the Twitter handle "@benlesh". The NG CONF logo is visible in the top right corner of the slide.

Introducing RxJS6! - Ben Lesh

<https://www.youtube.com/watch?v=JCXZhe6KsxQ>

<https://www.learnrxjs.io/>



The screenshot shows the 'Operators' page on the Learn RxJS website. The page has a light blue sidebar on the left with a search bar and a list of navigation items: Introduction, LEARN RXJS, Operators (selected), Combination, Conditional, Creation, Error Handling, Multicasting, Filtering, Transformation, Utility, Full Listing, Subjects, Recipes, and Concepts. The main content area is titled 'Operators' and contains a paragraph: 'A complete list of RxJS operators with clear explanations, relevant resources, and executable examples.' Below this is a link: 'Prefer a complete list in alphabetical order?'. The 'Contents (By Operator Type)' section lists three categories: Combination, Conditional, and Creation. The 'Combination' category lists: combineAll, combineLatest (marked with a star), concat (marked with a star), concatAll, endWith, forkJoin, merge (marked with a star), mergeAll, pairwise, race, startWith (marked with a star), withLatestFrom (marked with a star), and zip. The 'Conditional' category lists: defaultIfEmpty, every, iif, and sequenceEqual. The 'Creation' category is partially visible. On the right side, there is a 'CONTENTS' section with links to 'Contents (By Operator Type)' and 'Additional Resources'. At the bottom left, there is a 'Powered by GitBook' logo.

Learn RxJS

Introduction

LEARN RXJS

Operators

Combination

Conditional

Creation

Error Handling

Multicasting

Filtering

Transformation

Utility

Full Listing

Subjects

Recipes

Concepts

Operators

A complete list of RxJS operators with clear explanations, relevant resources, and executable examples.

[Prefer a complete list in alphabetical order?](#)

Contents (By Operator Type)

- Combination
  - combineAll
  - combineLatest ★
  - concat ★
  - concatAll
  - endWith
  - forkJoin
  - merge ★
  - mergeAll
  - pairwise
  - race
  - startWith ★
  - withLatestFrom ★
  - zip
- Conditional
  - defaultIfEmpty
  - every
  - iif
  - sequenceEqual
- Creation

CONTENTS

[Contents \(By Operator Type\)](#)

[Additional Resources](#)

Powered by GitBook



# Using the `async pipe`

Automagically `.subscribe()` and `.unsubscribe()`

# Async Pipe

- On `.subscribe()`, you actually need to `.unsubscribe()` to avoid memory leaks
  - Best practice
  - Not *\*really\** necessary on HTTP-requests, but you do in other subscriptions.
- No more manually `.subscribe()` and `.unsubscribe()`:
  - Use Angular `async pipe`

- **Component:**

```
cities$: Observable<City[]>; // Now: Observable to Type
```

```
...
```

```
ngOnInit() {  
    // Call service, returns an Observable  
    this.cities$ = this.cityService.getCities()  
}
```

- **View:**

```
<li *ngFor="let city of cities$ | async">
```

<https://blog.angularindepth.com/angular-question-rxjs-subscribe-vs-async-pipe-in-component-templates-c956c8c0c794>

(Background information)



# Working with Live API's

- MovieApp
- `examples\210-services-live`



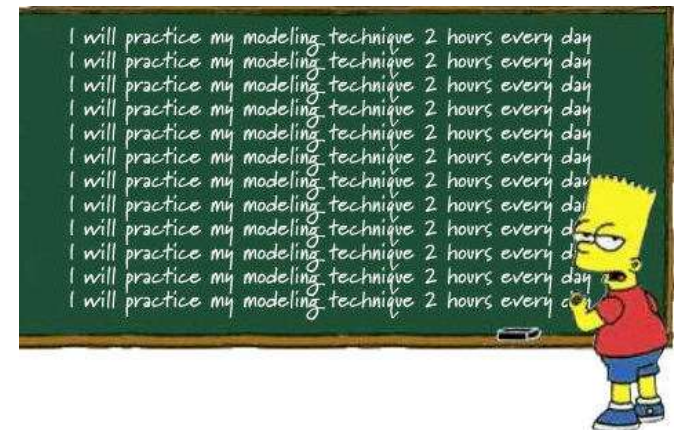
# Example API's

- <https://pokeapi.co/> - Pokemon API
- <http://openweathermap.org/API> (weather forecast)
- <https://jsonplaceholder.typicode.com/> random users, posts, photos
- <http://ergast.com/mrd/> - Ergast Motor (F1) API
- <http://www.omdbapi.com/> - Open Movie Database
- <http://swapi.co/> - Star Wars API
- See also `JavaScript APIs.txt` with other API's.

# Exercise

- Pick one of your own projects, or see for examples:
  - 210-services-live
- Create a small application using one of the API's in the file `JavaScript API's.txt`, using RxJS-calls, for example
  - Star Wars API
  - OpenWeatherMap API
  - ...
- Exercise 5e)

# Exercise....

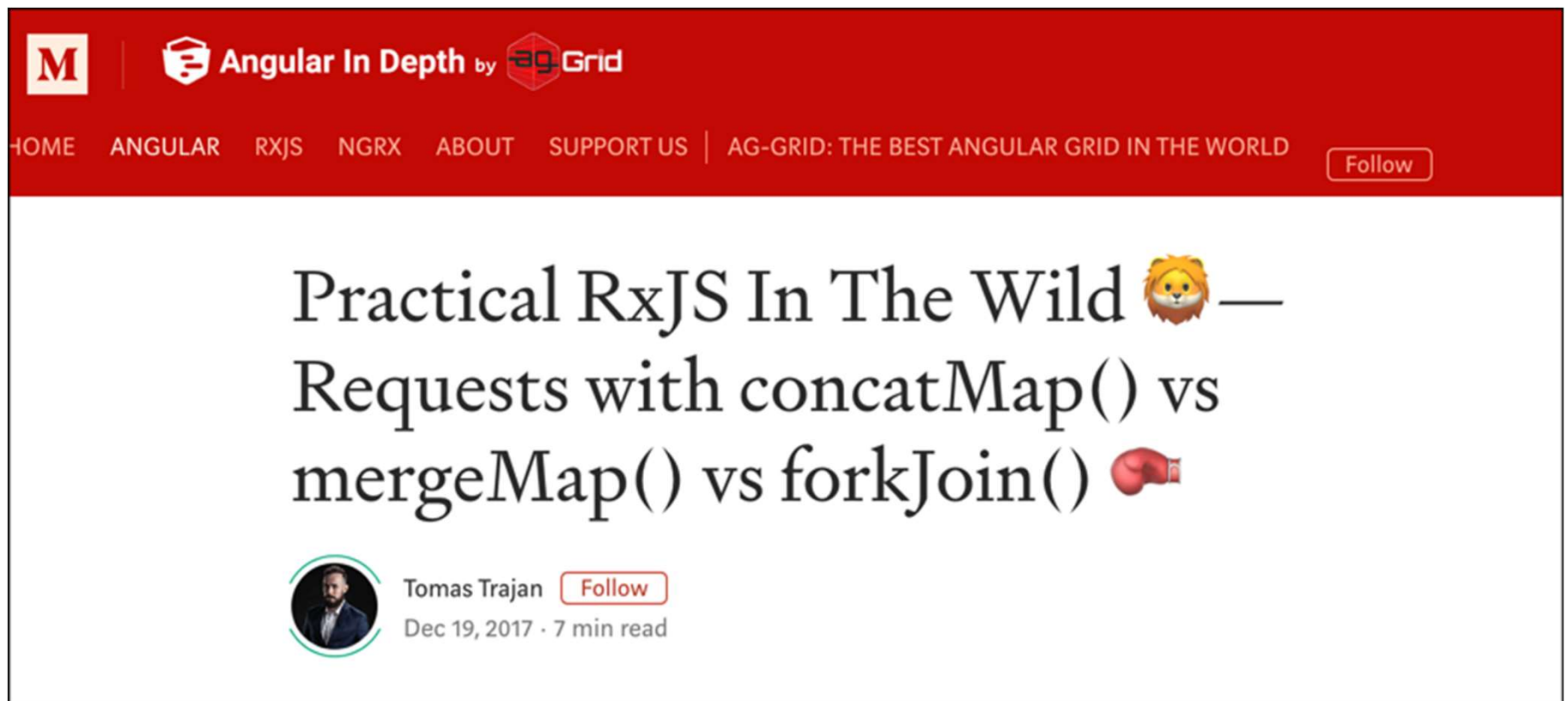




# More info on observables

# Fetching a collection of observables

<https://blog.angularindepth.com/practical-rxjs-in-the-wild-requests-with-concatmap-vs-mergemap-vs-forkjoin-11e5b2efe293>



# Online JSON to TypeScript converter

# json2ts

generate TypeScript interfaces from JSON

[email](#) [feedback](#) [help](#)

```
{
  "Title": "The Amazing Captain Nemo",
  "Year": "1978",
  "imdbID": "tt0077156",
  "Type": "movie",
  "Poster": "https://images-na.ssl-images-
amazon.com/images/M/MV5BMTc4NzExNjcwN15BMTI5BanBnXkFtZTYwMTM1Mjg5_V1_SX300.jpg"},
  {"Title": "Nemo",
  "Year": "1984",
  "imdbID": "tt0087784",
  "Type": "movie",
  "Poster": "https://images-na.ssl-images-
amazon.com/images/M/MV5BMTY2NzlwMTgwN15BMTI5BanBnXkFtZTcwMjlyMzMzMzMQ@@_V1_SX300.jpg"},
  {"Title": "Captain
Nemo",
  "Year": "1975",
  "imdbID": "tt0453375",
  "Type": "movie",
  "Poster": "https://images-na.ssl-images-
amazon.com/images/M/MV5BM2JmOTRlMGQtODMxNy00YmRkLWI1OWEtMmQ2YjZiZmQzZGU5XkEyXkFqcGdeQXVyNDUxNjc5NjY@._V1_
SX300.jpg"},
  {"Title": "Finding Nemo",
  "Year": "2003",
  "imdbID": "tt0401422",
  "Type": "game",
  "Poster": "N/A"},
  {"Title": "Making
'Nemo'",
  "Year": "2003",
  "imdbID": "tt0387373",
  "Type": "movie",
  "Poster": "N/A"},
  {"Title": "Finding Nemo Submarine
Voyage",
  "Year": "2007",
  "imdbID": "tt1319713",
  "Type": "movie",
  "Poster": "https://images-na.ssl-images-
amazon.com/images/M/MV5BMzAxMzMyODQtNWY0Yy00N2M3LWE5MDQtZDUzNjc1ZGFmMzA4XkEyXkFqcGdeQXVyMzcxMzc4Mw@@_V
1_SX300.jpg"},
  {"Title": "Little Nemo: The Dream
Master",
  "Year": "1990",
  "imdbID": "tt0206895",
  "Type": "game",
  "Poster": "N/A"}],
  "totalResults": "31",
  "Response": "True"}
```

generate TypeScript

```
declare module namespace {

  export interface Search {
    Title: string;
    Year: string;
    imdbID: string;
    Type: string;
    Poster: string;
  }
}
```


<http://json2ts.com/>

# In VS Code? Use this extension!

The screenshot shows the Visual Studio Code interface. On the left, the 'EXTENSIONS: MARKETPLACE' sidebar is open, displaying a search for 'paste json'. The 'Paste JSON as Code' extension by 'quicktype' is highlighted at the top of the list. Below it, other extensions like 'JSON Tools', 'JSON to TS', 'Paste Image', 'Paste and Indent', 'Prettify JSON', 'Sort JSON objects', 'JSON Schema to JSON T...', and 'json2ts' are listed. The main panel shows the details for the 'Paste JSON as Code' extension. It features the 'quicktype' logo, the extension name 'Paste JSON as Code', the publisher 'quicktype.quicktype', and statistics: 63,602 installs and a 5/5 rating. The description states: 'Copy JSON, paste as Go, TypeScript, C#, C++ and more.' There is an 'Install' button. Below the description, tabs for 'Details', 'Contributions', 'Changelog', and 'Dependencies' are visible. A section titled 'Visual Studio Marketplace' shows 'v10.0.24', 'installs 63602', and 'rating 5/5 (5)'. A note says 'Supports C#, Go, C++, Java, TypeScript, Swift, Elm, and JSON Schema.' A paragraph explains that 'quicktype' infers types from sample JSON data and outputs strongly typed models and serializers. At the bottom, a preview window shows the extension in use. It displays a JSON file named 'pokedex.json' with a search bar containing '>cop|'. A dropdown menu is open, showing options like 'Copy', 'Copy Line Down', 'Copy Line Up', 'Copy With Syntax Highlighting', 'Developer: Inspect TM Scopes', 'File: Clear Recently Opened', and 'File: Copy Path of Active File'.

<https://marketplace.visualstudio.com/items?itemName=quicktype.quicktype>

# Data Mocking - Mockaroo

 **mockaroo** realistic data generator

?

PRICING

SIGN IN

Need some mock data to test your app?

Mockaroo lets you generate up to 1,000 rows of realistic test data in CSV, JSON, SQL, and Excel formats.

[Need more data? Plans start at just \\$50/year.](#)

Field Name	Type	Options
<div>id</div>	Row Number	blank: 0 % <div>fx</div> ×
<div>first_name</div>	First Name	blank: 0 % <div>fx</div> ×
<div>last_name</div>	Last Name	blank: 0 % <div>fx</div> ×
<div>email</div>	Email Address	blank: 0 % <div>fx</div> ×
<div>gender</div>	Gender	blank: 0 % <div>fx</div> ×
<div>ip_address</div>	IP Address v4	blank: 0 % <div>fx</div> ×

Add another field

# Rows: 1000

Format: CSV

Line Ending: Unix (LF)

Include: ☒ header ☐ BOM

Download Data

Preview

More

Want to save this for later? [Sign up for free.](#)

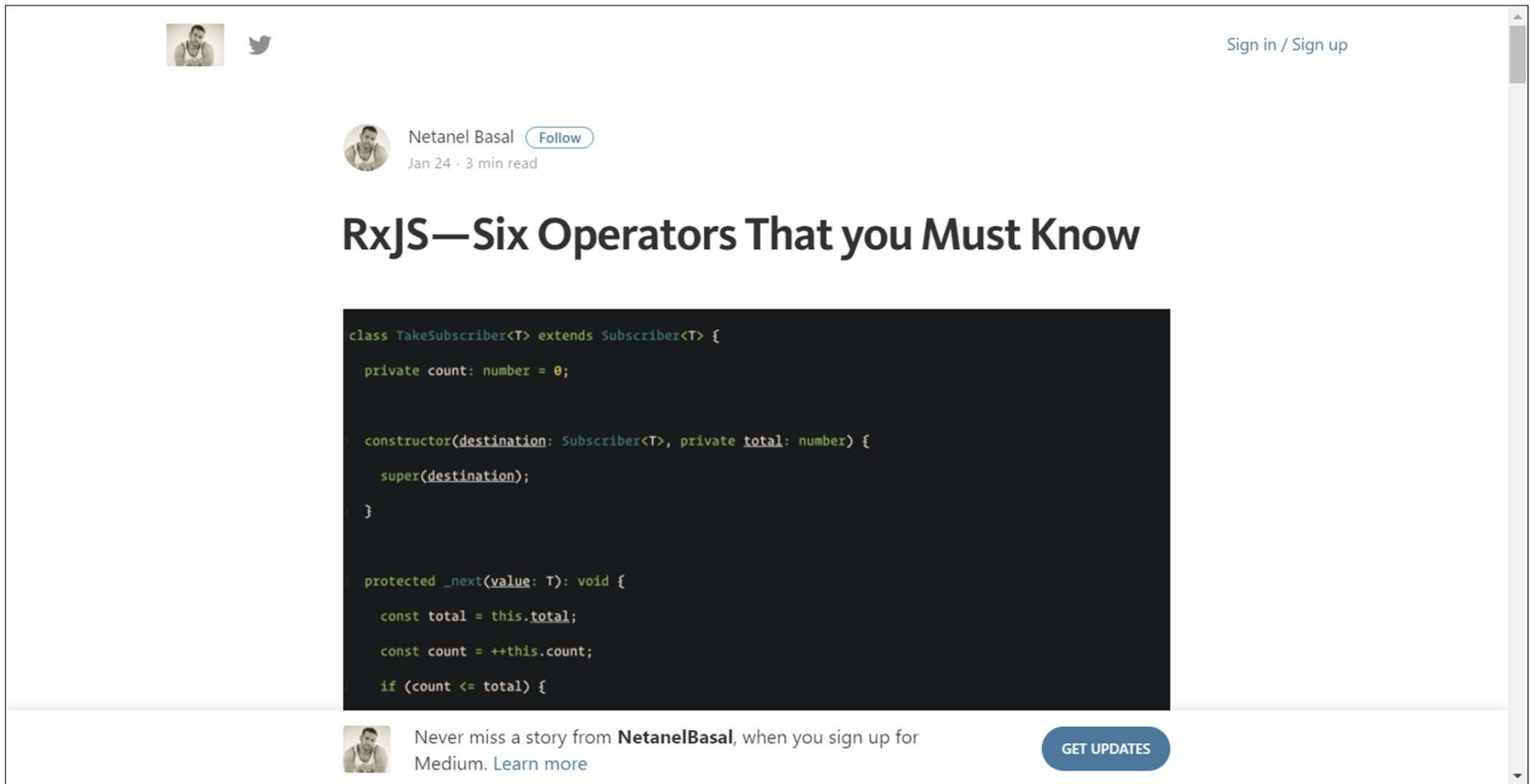
<http://mockaroo.com/>



# Useful operators

- RxJS operators are (mostly) just like Array operators
- Perform actions on a stream of objects
- Grouped by subject
  - Creation operators
  - Transforming
  - Filtering
  - Combining
  - Error Handling
  - Conditional and Boolean
  - Mathematical
  - ...

# 6 Operators you must know



The screenshot shows a Medium article interface. At the top left is a profile picture and a Twitter icon. At the top right is a 'Sign in / Sign up' link. Below the profile picture is the author's name 'Netanel Basal' and a 'Follow' button. The article title is 'RxJS—Six Operators That you Must Know'. The main content is a code block with the following TypeScript code:

```
class TakeSubscriber<T> extends Subscriber<T> {  
  private count: number = 0;  
  
  constructor(destination: Subscriber<T>, private total: number) {  
    super(destination);  
  }  
  
  protected _next(value: T): void {  
    const total = this.total;  
    const count = ++this.count;  
    if (count <= total) {
```

At the bottom of the article preview, there is a small profile picture, the text 'Never miss a story from NetanelBasal, when you sign up for Medium. Learn more', and a 'GET UPDATES' button.

<https://netbasal.com/rxjs-six-operators-that-you-must-know-5ed3b6e238a0#.11of73aox>

# Creating Observables from scratch

## - André Staltz

André Staltz (@andrestaltz): You will learn RxJS at ng-europe 2016

```
1 function nextCallback(data) {  
2   console.log(data);  
3 }  
4 function errorCallback(err) {  
5 }  
6 function completeCallback() {  
7 }  
8  
9 function giveMeSomeData(nextCB, errorCallback, completeCallback) {  
10  document.addEventListener('click', () => {  
11    nextCallback({data: 'clicked'});  
12    errorCallback(null);  
13    completeCallback();  
14  });  
15 }  
16  
17 giveMeSomeData(  
18   nextCallback,  
19   errorCallback,  
20   completeCallback  
21 );
```

addEventListener(  
 type: "MSContentZoom",  
 listener: (this: Document, ev:  
 UIEvent) => any  
 , useCapture?: boolean): void

ng europe  
JS  
ngeurope.org  
RANGLE.IO  
SOMETHING THE WEB

5:11 / 22:44

<https://www.youtube.com/watch?v=uQ1zhJHclvs>

GitHub Gist Search... All gists GitHub New gist

staltz / introrx.md Last active an hour ago

★ Star 10,812 🍴 Fork 1203 ⓘ

Code Revisions 259 Stars 10812 Forks 1203 Embed <script src="https://gist. Download ZIP

The introduction to Reactive Programming you've been missing

introrx.md Raw

## The introduction to Reactive Programming you've been missing

(by @andrestaltz)

---

### This tutorial as a series of videos

If you prefer to watch video tutorials with live-coding, then check out this series I recorded with the same contents as in this article: [Egghead.io - Introduction to Reactive Programming](#).

---

So you're curious in learning this new thing called Reactive Programming, particularly its variant comprising of Rx, Bacon.js, RAC, and others.

Learning it is hard, even harder by the lack of good material. When I started, I tried looking for tutorials. I found only a handful of practical guides, but they just scratched the surface and never tackled the challenge of building the whole architecture

<https://gist.github.com/staltz/868e7e9bc2a7b8c1f754>

# RxMarbles

## RxMarbles

INTERACTIVE DIAGRAMS OF RX OBSERVABLES

TRANSFORMING OPERATORS

- [delay](#)
- [delayWithSelector](#)
- [findIndex](#)
- [map](#)
- [scan](#)
- [debounce](#)
- [debounceWithSelector](#)

COMBINING OPERATORS

- [combineLatest](#)
- [concat](#)
- [merge](#)
- [sample](#)
- [startWith](#)
- [withLatestFrom](#)
- [zip](#)

FILTERING OPERATORS

- [distinct](#)
- [distinctUntilChanged](#)
- [elementAt](#)
- [filter](#)
- [find](#)
- [first](#)

Interactive diagrams of Rx Observables

The diagram illustrates the `merge` operator. It shows two input observables being merged into a single output observable. The first observable has values 20, 40, 60, 80, 100. The second observable has values 1, 1. The merged observable has values 20, 40, 60, 1, 80, 100, 1.

v1.4.1 built on RxJS v2.5.3 by @andrestaltz

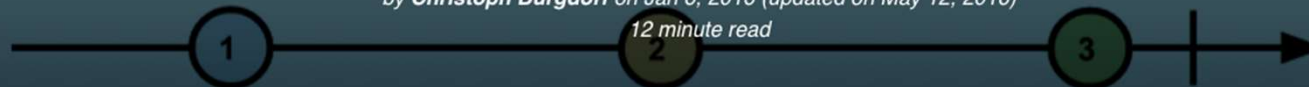
<http://rxmarbles.com/>



# TAKING ADVANTAGE OF OBSERVABLES IN `distinctUntilChanged()` ANGULAR 2

by **Christoph Burgdorf** on Jan 6, 2016 (updated on May 12, 2016)

12 minute read



Some people seem to be confused why Angular 2 seems to favor the Observable abstraction over the Promise abstraction when it comes to dealing with async behavior.

There are pretty good resources about the difference between Observables and Promises already out there. I especially like to highlight this free [7 minutes video](#) by [Ben Lesh](#) on [egghead.io](#). Technically there are a couple of obvious differences like the *disposability* and *laziness* of Observables. In this article we like to focus on some practical advantages that

<http://blog.thoughttram.io/angular/2016/01/06/taking-advantage-of-observables-in-angular2.html>