



Global Knowledge®

# Angular Fundamentals

## Observables

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

### WORLDWIDE LOCATIONS

BELGIUM CANADA COLOMBIA DENMARK EGYPT FRANCE IRELAND JAPAN KOREA MALAYSIA MEXICO NETHERLANDS NORWAY QATAR  
SAUDI ARABIA SINGAPORE SPAIN SWEDEN UNITED ARAB EMIRATES UNITED KINGDOM UNITED STATES OF AMERICA



# **Async services met RxJS/Observables**

Reactive programming with asynchronous streams

# Async Services

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

Also Angular : ReactiveX library `RxJS`



An API for asynchronous  
with observable streams

Choose your platform

<http://reactivex.io/>

## 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

### 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.*

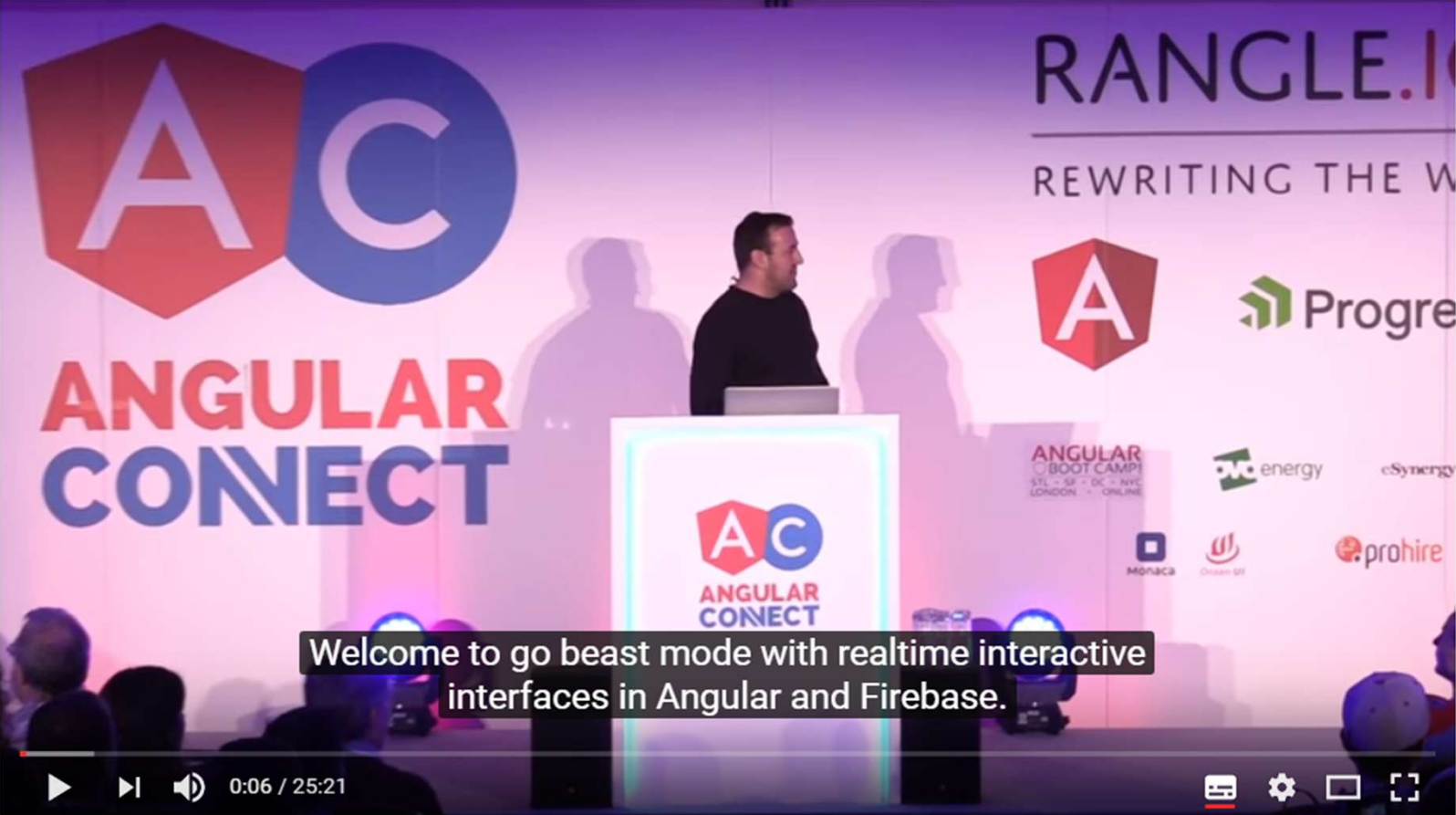
*With observables, we have a whole bunch of operators to pull from, which let us customize our streams in nearly any way we want.*

<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!





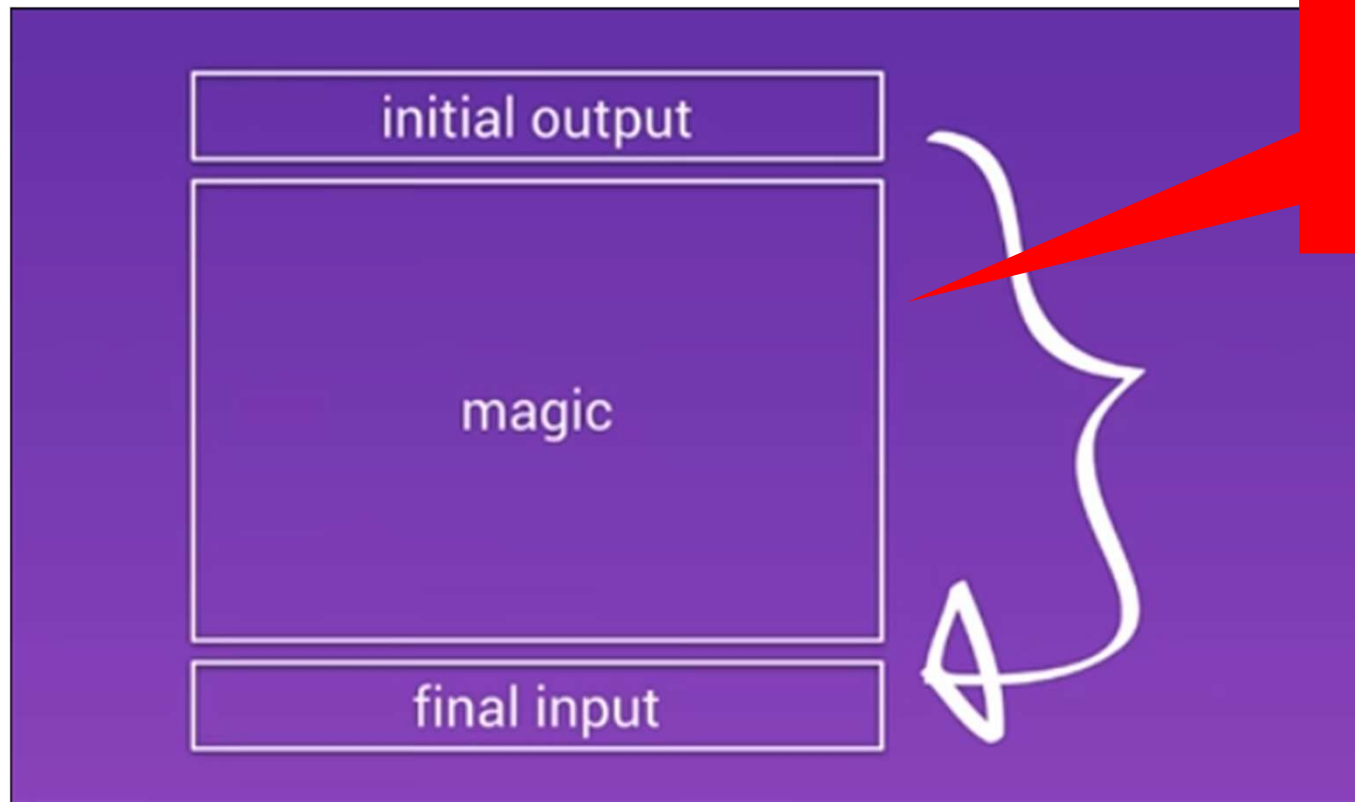
A screenshot of a YouTube video player. The video shows a man standing at a podium with the Angular Connect logo on it. Behind him is a large screen displaying the Angular Connect logo (a red shield with 'A' and a blue circle with 'C') and the text 'ANGULAR CONNECT'. To the right, the screen also shows 'RANGLE.I', 'REWRITING THE W', and several logos including 'Progre', 'energy', 'eSynergy', 'prohire', 'MONACA', and 'Dream UI'. A subtitle at the bottom of the video frame reads: 'Welcome to go beast mode with realtime interactive interfaces in Angular and Firebase.' The video player interface at the bottom shows a play button, a progress bar at 0:06 / 25:21, and icons for subtitles, settings, and full screen.

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

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

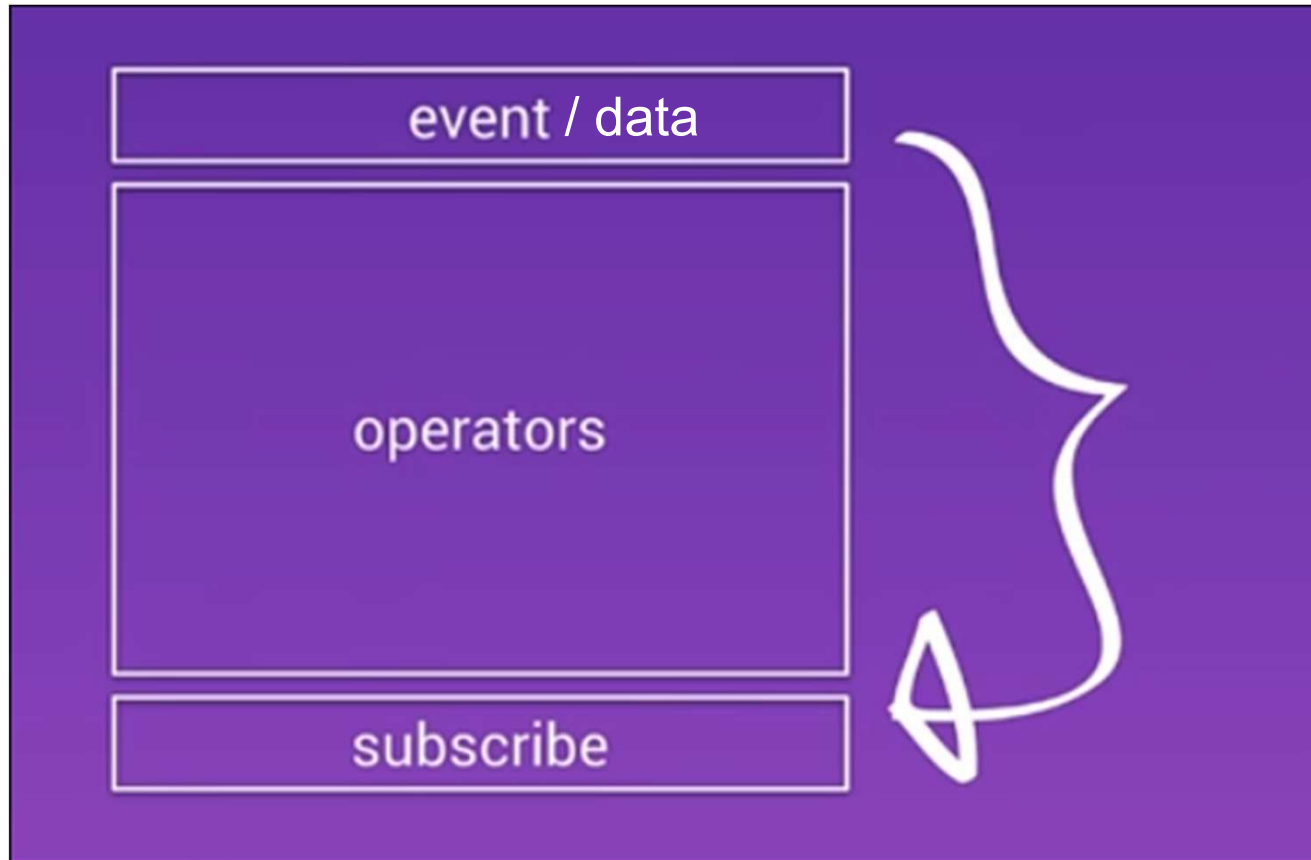
<https://youtu.be/5CTL7aqSvJU?t=4m31s>

# "The observable sandwich"



**Not really Magic.  
Just operators**

# Subscribe to events



## Most common usage of observables: http

Modern applications are moduleless

provide the imported `httpClient()`

```
// app.config.ts
import ...;
import {provideHttpClient} from '@angular/common/http';

export const appConfig: ApplicationConfig = {
  providers: [
    provideHttpClient(),
    ...
  ]
};
```

## In classical applications

When (still) using ngModules,

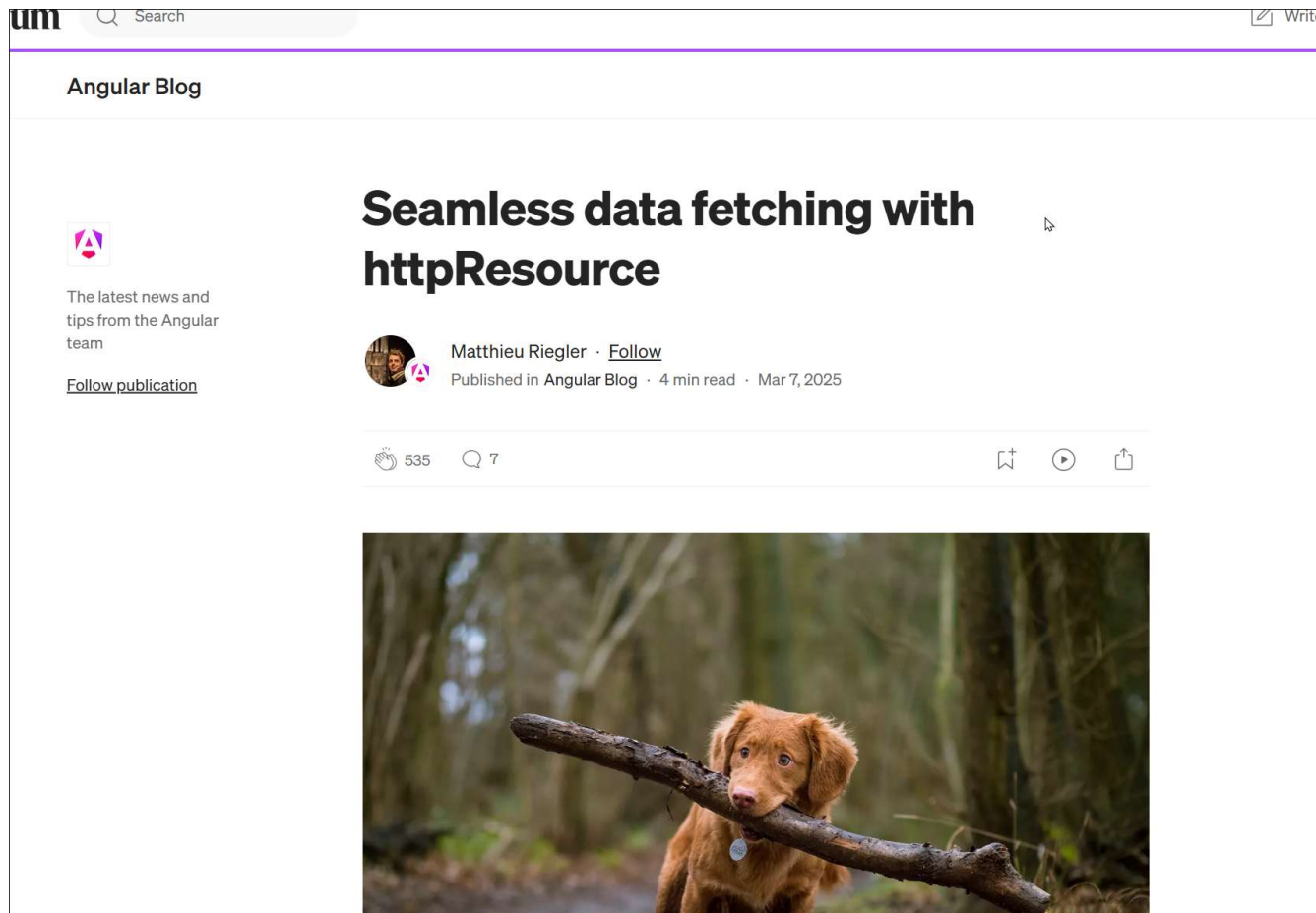
`import HttpClientModule()`

```
// Angular Modules
import ...
import {HttpClientModule} from "@angular/common/http";

// Module declaration
@NgModule({
  imports      : [BrowserModule, HttpClientModule],
  declarations: [AppComponent],
  bootstrap    : [AppComponent],
})
export class AppModule {
}
```

# In even newer applications (v19.2+)

- Angular `httpResource` available.
- <https://blog.angular.dev/seamless-data-fetching-with-httpresource-71ba7c4169b9>



# Classical: inject http in Component / service

- Create an `http` variable, of type `HttpClient`
- Use Constructor Injection, or `inject()` function

```
export class AppComponent {  
  ...  
  constructor(private http: HttpClient) {}  
}
```

```
export class AppComponent {  
  // using inject()  
  private http = inject (HttpClient);  
}
```

<https://angular.dev/guide/http>

## Making an http call, for instance

Initial Output

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

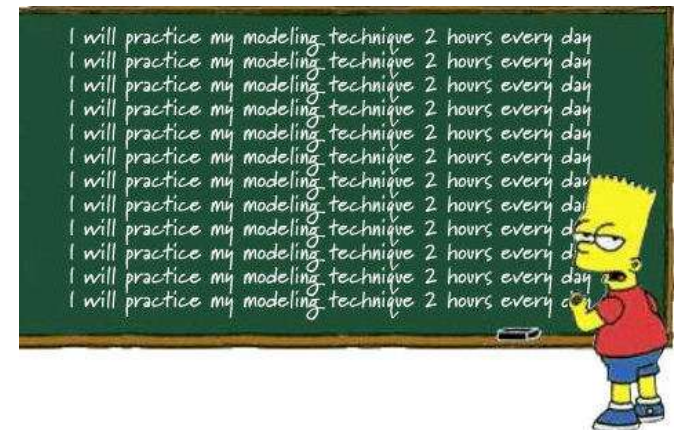
Optioneel:  
operator(s)

Final Input



# Workshop

- You know how to import `HttpClientModule`
- You know about the 'observable sandwich'
- See the example in `/201_services_http`
- Create your own `.json`-file and import it in your application.





# 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...')  
)
```

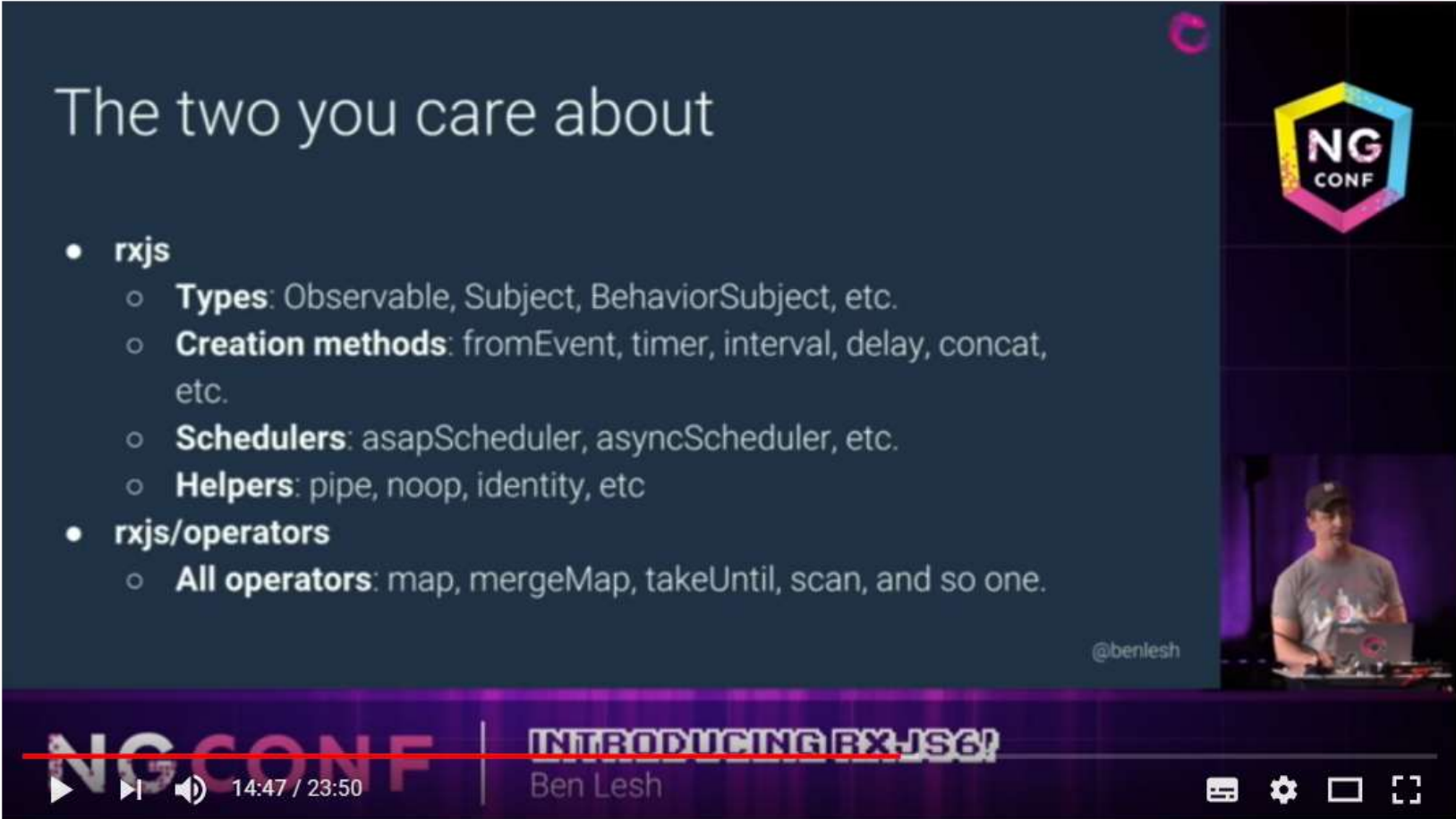


# Pipeable operators

- 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...)  
)
```

# Ben Lesh on observables in RxJS 6.0



The video player shows a presentation slide titled "The two you care about" with the following content:

- **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 the NGCONF logo, the title "INTRODUCING RXJS6!", the speaker's name "Ben Lesh", a progress bar at 14:47 / 23:50, and standard playback controls. A small inset video shows Ben Lesh at a laptop.

Introducing RxJS6! - Ben Lesh

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

# Useful operators

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

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

The screenshot shows a web browser at the URL `learnrxjs.io/learn-rxjs/operators`. The page has a light blue sidebar on the left with the 'Learn RxJS' logo and a search icon. The sidebar contains a menu with '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?'. Further down is a section titled 'Contents (By Operator Type)' which lists several categories: 'Combination' (with sub-items like `combineAll`, `combineLatest`, `concat`, `concatAll`, `endWith`, `forkJoin`, `merge`, `mergeAll`, `pairwise`, `race`, `startWith`, `withLatestFrom`, `zip`), 'Conditional' (with sub-items like `defaultIfEmpty`, `every`, `iif`, `sequenceEqual`), and 'Creation'. A right sidebar contains 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

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`

# Where to change/update your code:

## Component:

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

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

## View:

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

## Service:

No changes necessary

## OR: using the modern `@for` syntax with `async`

For instance, retrieving `users` instead of `cities`:

```
users$: Observable<any[]> | undefined;  
ngOnInit(){  
  ...  
  this.users$ = this.http.get<any[]>  
    ('https://jsonplaceholder.typicode.com/users');  
}
```



```
@for (user of users$ | async; track user.id) {  
  <div>  
    {{ user.name }}  
  </div>  
}
```

# Working with Live API's

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

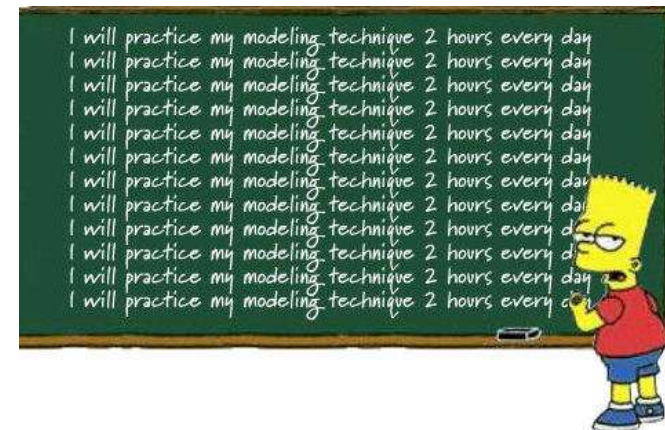


# Example API's

- <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.dev/> - Star Wars API
- See also `JavaScript APIs.txt` with other API's.

# Workshop

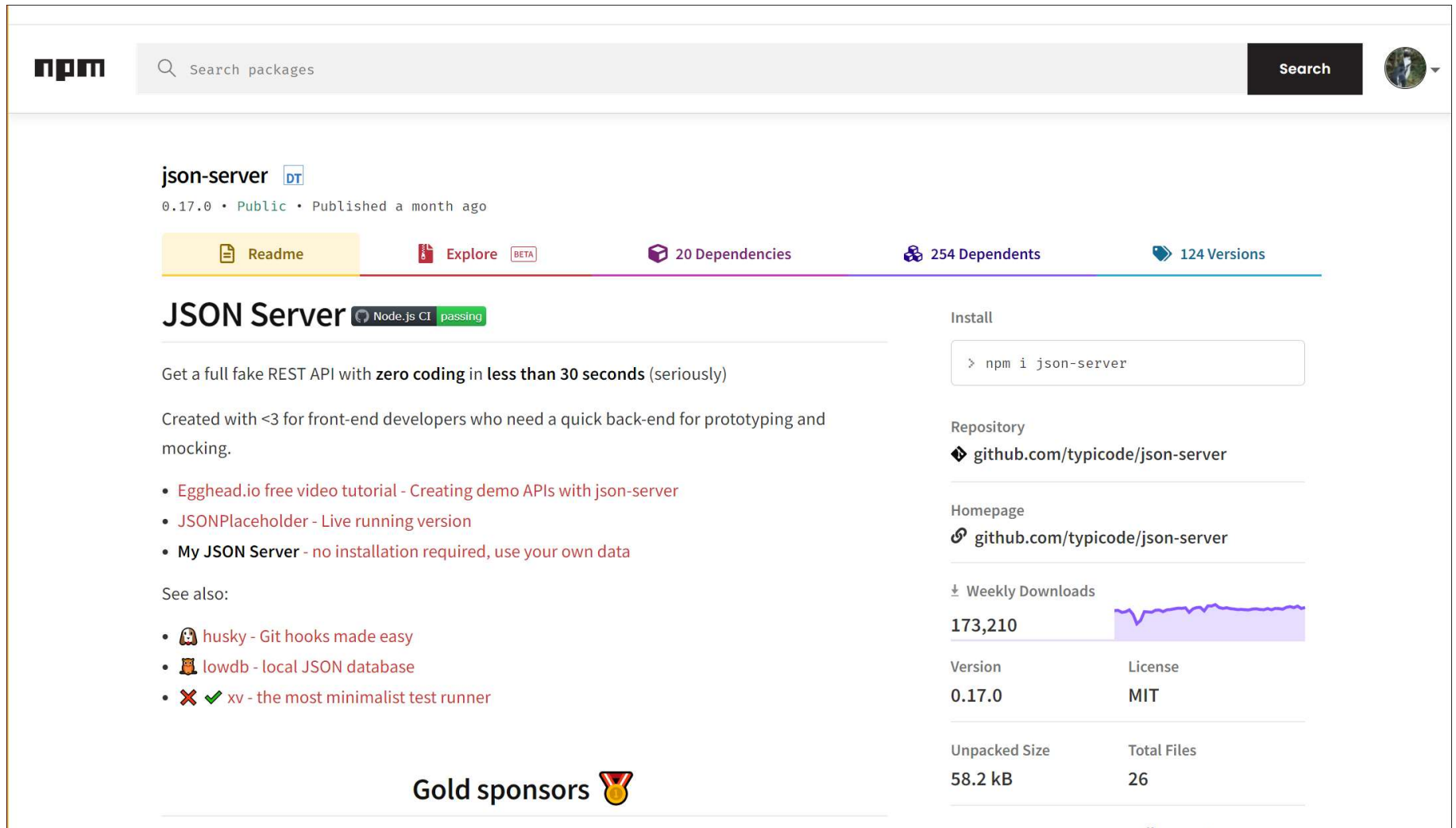
- Pick one of your own projects, or see for instance:
  - `../210-services-live`
- Create a small application using one of the API's in the file `JavaScript API's.txt`, using RxJS-calls, for example
  - Pokemon API
  - Kenteken API
  - OpenWeatherMap API
  - ...





# More info on observables

# Creating your own (fake) API – json server



The screenshot shows the npm package page for **json-server**. The page includes a search bar at the top, the package name **json-server** with a **DT** badge, and version information **0.17.0 • Public • Published a month ago**. Navigation links include **Readme**, **Explore** (with a **BETA** badge), **20 Dependencies**, **254 Dependents**, and **124 Versions**.

The main section is titled **JSON Server** with a **Node.js CI** badge and a **passing** status. The description states: "Get a full fake REST API with **zero coding** in **less than 30 seconds** (seriously)". It also mentions: "Created with <3 for front-end developers who need a quick back-end for prototyping and mocking."

Links provided include:

- [Egghead.io free video tutorial - Creating demo APIs with json-server](#)
- [JSONPlaceholder - Live running version](#)
- [My JSON Server - no installation required, use your own data](#)

Under "See also:",

- [husky - Git hooks made easy](#)
- [lowdb - local JSON database](#)
- [xv - the most minimalist test runner](#)

At the bottom, it says "Gold sponsors" with a medal icon.

On the right side, the **Install** section shows the command: `> npm i json-server`. Below it, the **Repository** is [github.com/typicode/json-server](https://github.com/typicode/json-server), and the **Homepage** is also [github.com/typicode/json-server](https://github.com/typicode/json-server).

The **Weekly Downloads** section shows a graph and the number **173,210**.

Version	License
0.17.0	MIT

Unpacked Size	Total Files
58.2 kB	26

Below the table, there are sections for **Issues** and **Pull Requests**.

<https://www.npmjs.com/package/json-server>



# GET, POST, PUT, DELETE with json-server

- **Example:** `../205-services-http-crud`
  - 1. `npm install`
  - 2. `npm run json-server`
  - 3. `npm start | ng serve`

## CRUD-operations on cities: using package json-server

1 - Groningen

2 - Hengelo

3 - Den Haag

4 - Enschede

5 - Heerlen

new city...

Add City

Clear

### Selected city: Groningen



Hide

Delete

City Name

Groningen

Province

Groningen

Highlights

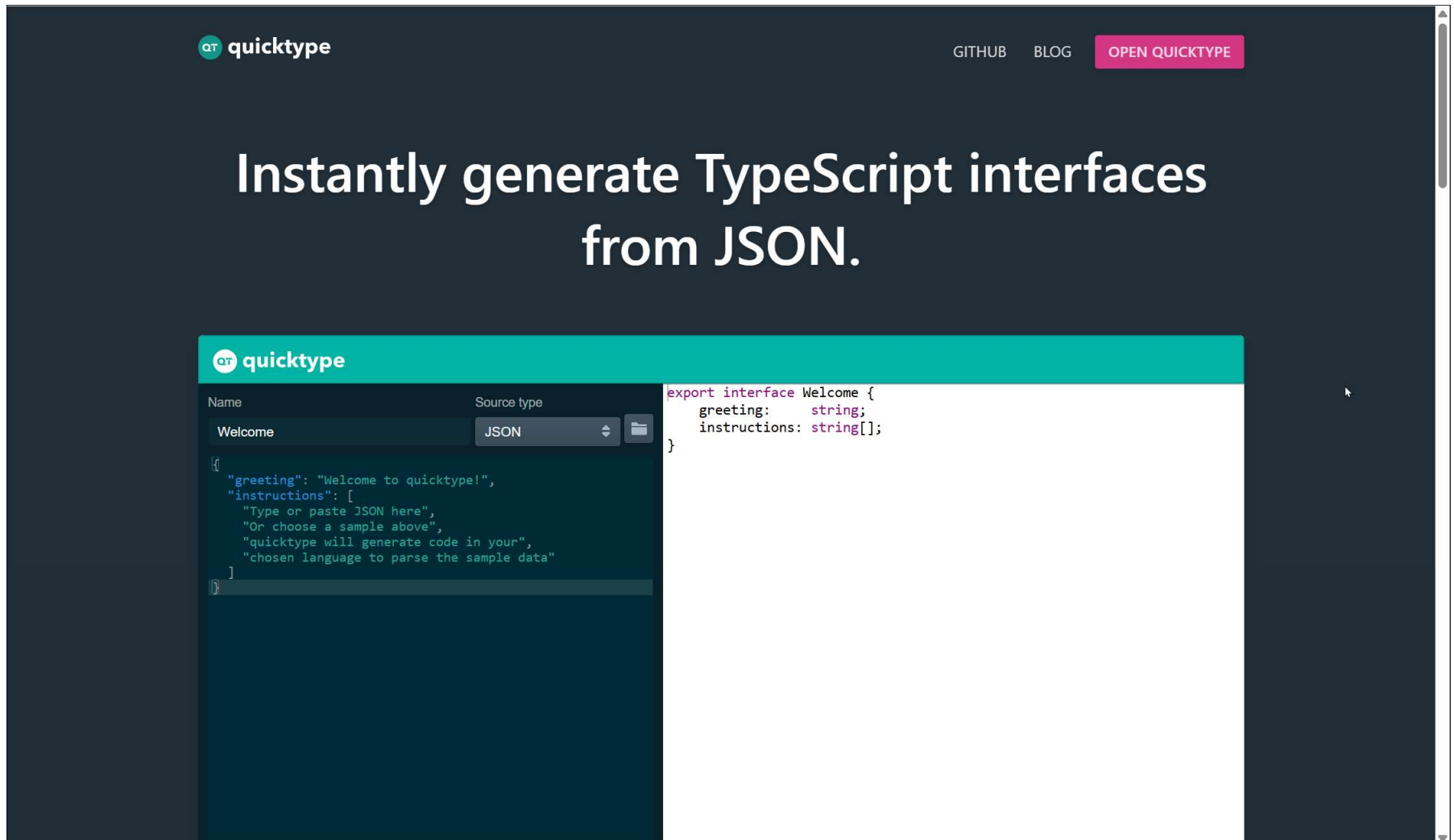
Martinitoren

Update

Cancel

[Info on json-server](#)

# Creating TypeScript from json



The screenshot displays the QuickType website interface. At the top, the 'quicktype' logo is on the left, and 'GITHUB', 'BLOG', and an 'OPEN QUICKTYPE' button are on the right. The main heading reads 'Instantly generate TypeScript interfaces from JSON.' Below this, a preview of the tool's interface is shown. It features a 'Name' field with 'Welcome' and a 'Source type' dropdown set to 'JSON'. The JSON input area contains a sample object with 'greeting' and 'instructions' fields. The output area on the right shows the generated TypeScript interface: 

```
export interface Welcome {  
  greeting: string;  
  instructions: string[];  
}
```

<https://quicktype.io/typescript>

# Official documentation...

The screenshot shows the RxJS official documentation website. The left sidebar contains a navigation menu with categories like 'observable', 'observable/dom', 'operator', and 'scheduler'. The main content area is titled 'Observable' and includes a code snippet at the top: 

```
import {Observable} from '@reactivex/rxjs/es6/Observable.js'
public class | source
```

. Below the title, it lists 'Direct Subclass:' (ConnectableObservable, GroupedObservable, Subject) and 'Indirect Subclass:' (AnonymousSubject, AsyncSubject, BehaviorSubject, es6/operator/windowTime.js~CountedSubject, ReplaySubject). A description states: 'A representation of any set of values over any amount of time. This the most basic building block of RxJS.' Under the 'Test:' section, it lists 'Observable', 'Observable.create', and 'Observable.lift'. The 'Static Method Summary' section contains a table with two rows of static public methods.

**Observable**

**Direct Subclass:**  
ConnectableObservable, GroupedObservable, Subject

**Indirect Subclass:**  
AnonymousSubject, AsyncSubject, BehaviorSubject, es6/operator/windowTime.js~CountedSubject, ReplaySubject

A representation of any set of values over any amount of time. This the most basic building block of RxJS.

**Test:**  
Observable  
Observable.create  
Observable.lift


**Static Method Summary**

Static Public Methods	
public static	<b>bindCallback</b> (func: function, selector: function, scheduler: Scheduler): function(...params: *): Observable Converts a callback API to a function that returns an Observable.
public static	<b>bindNodeCallback</b> (func: function, selector: function, scheduler: Scheduler): function(...params: *): Observable

Generated by ESDoc(0.4.8)

<http://reactivex.io/rxjs/class/es6/Observable.js~Observable.html>

# 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/>

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

The screenshot shows the 'Learn RxJS' website. On the left is a sidebar with a search icon and a menu. The menu includes 'Introduction', 'LEARN RXJS' (with 'Operators' selected), 'Combination', 'Conditional', 'Creation', 'Error Handling', 'Multicasting', 'Filtering', 'Transformation', 'Utility', 'Full Listing', 'Subjects', 'Recipes', and 'Concepts'. At the bottom of the sidebar is a 'Powered by GitBook' logo. The main content area is titled 'Operators' and contains the text: '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?'. Further down is a section titled 'Contents (By Operator Type)' which lists three categories: 'Combination' (with sub-items: combineAll, combineLatest ★, concat ★, concatAll, endWith, forkJoin, merge ★, mergeAll, pairwise, race, startWith ★, withLatestFrom ★, zip), 'Conditional' (with sub-items: defaultIfEmpty, every, iif, sequenceEqual), and 'Creation'. On the right side of the page, there is a 'CONTENTS' menu with links to 'Contents (By Operator Type)' and 'Additional Resources'.

**Learn RxJS**

**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

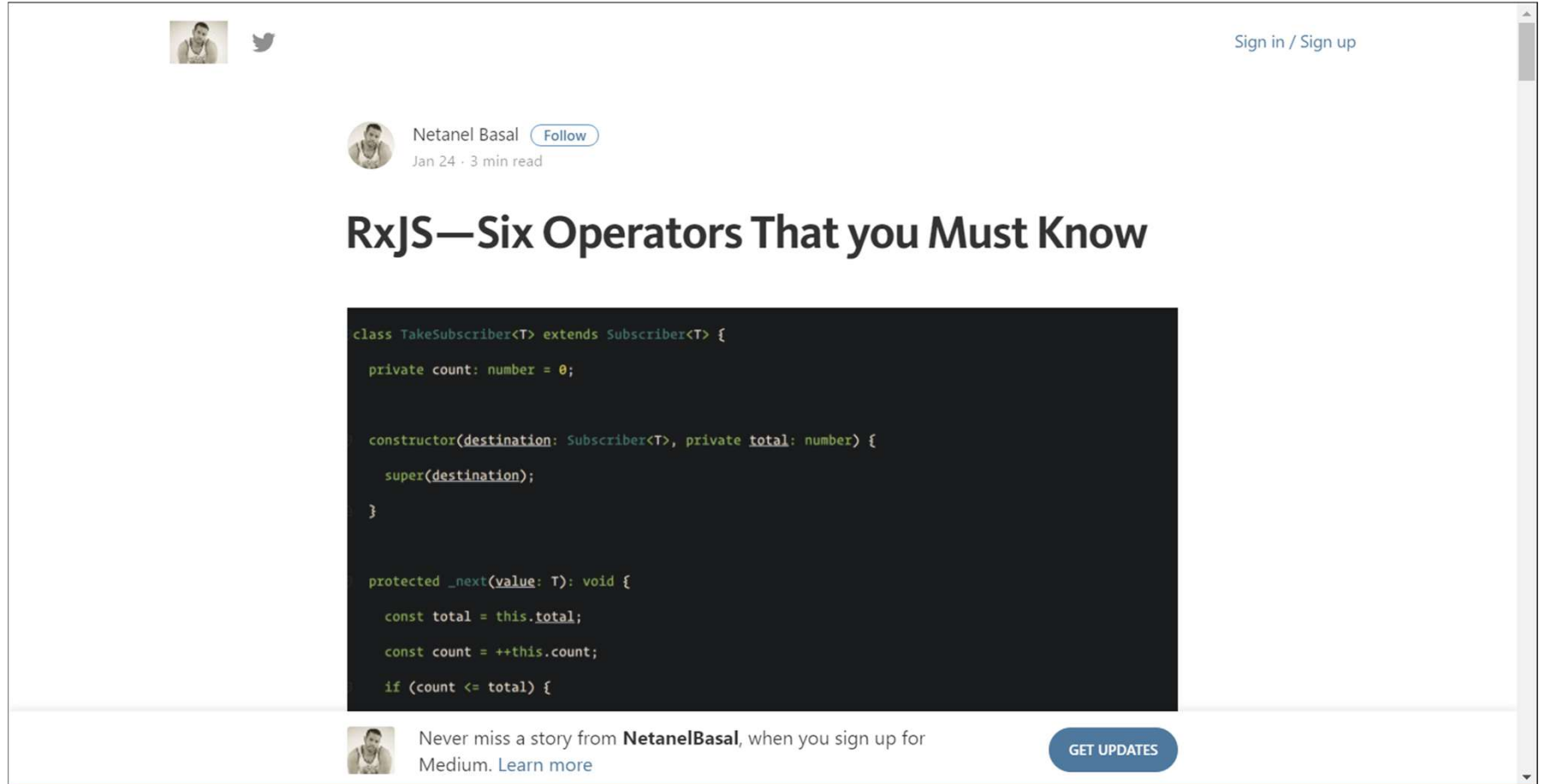
CONTENTS

Contents (By Operator Type)

Additional Resources

Powered by GitBook

# Article - 6 Operators you must know



The screenshot shows a Medium article preview. At the top left, there are social media icons for a profile picture and Twitter. On the top right, it says "Sign in / Sign up". Below this, the author's name "Netanel Basal" is displayed with a "Follow" button and the text "Jan 24 · 3 min read". The article title "RxJS—Six Operators That you Must Know" is prominently displayed. Below the title, a code block is shown with a dark background and light green text, containing a TypeScript class definition for a custom RxJS subscriber. At the bottom of the preview, there is a small profile picture, a text prompt to sign up for updates from NetanelBasal on Medium, and a "GET UPDATES" button.

Sign in / Sign up

Netanel Basal Follow  
Jan 24 · 3 min read

## RxJS—Six Operators That you Must Know

```
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) {
```

Never miss a story from **NetanelBasal**, when you sign up for Medium. Learn more

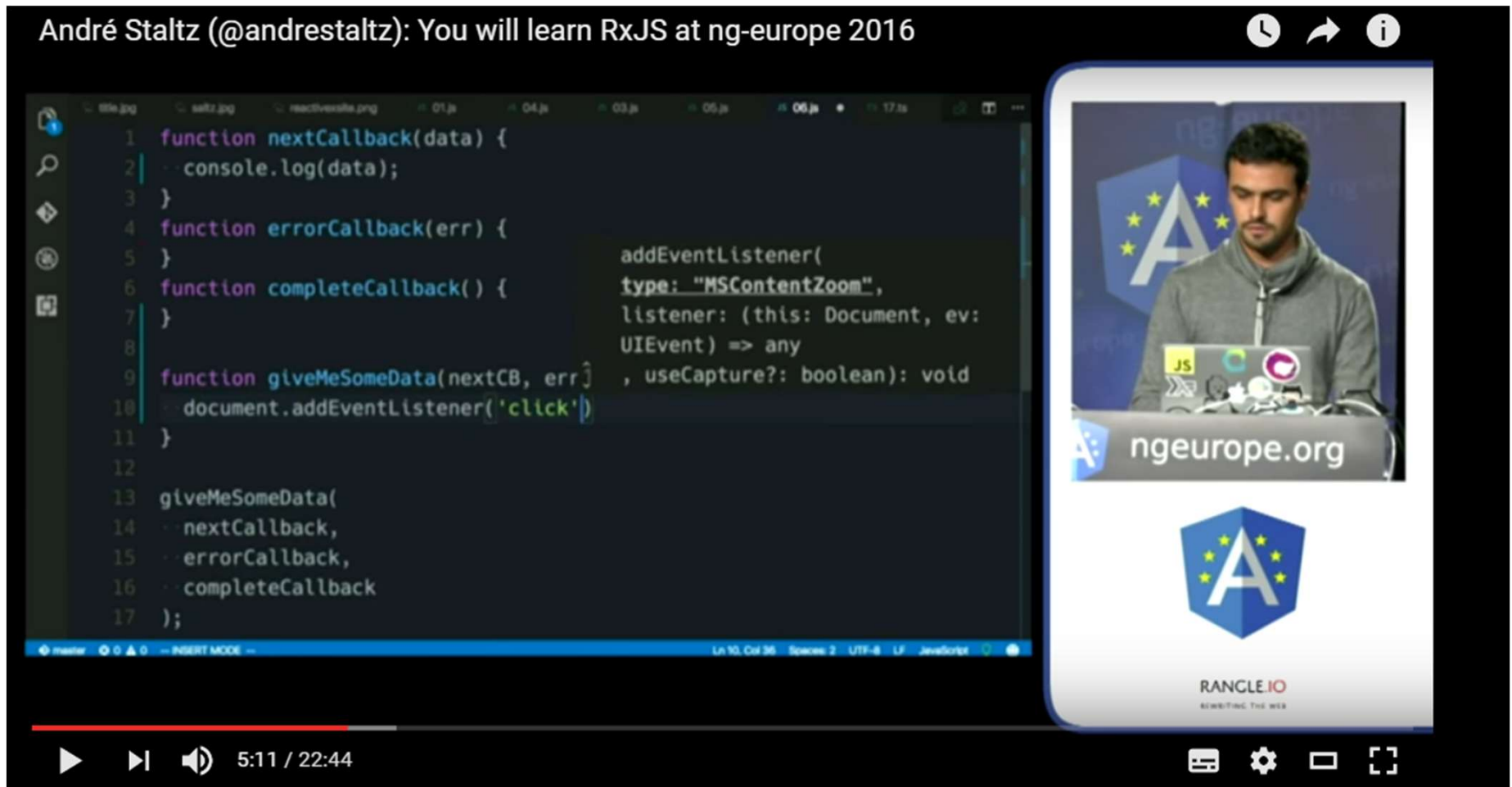
GET UPDATES

<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, err) {
10  document.addEventListener('click')
11 }
12
13 giveMeSomeData(
14   nextCallback,
15   errorCallback,
16   completeCallback
17 );
```

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

ng europe 2016

JS

ngeurope.org

RANGLE.IO  
REWRITING THE WEB

5:11 / 22:44

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

The screenshot shows a GitHub Gist page for a file named `introrx.md` by user `staltz`. The page header includes the GitHub Gist logo, a search bar, and navigation links for "All gists" and "GitHub". The user's profile picture and name are shown, along with the file name and a note that the user was last active an hour ago. On the right, there are buttons for "Star" (10,812), "Fork" (1203), and a dropdown menu. Below this, there are tabs for "Code", "Revisions" (259), "Stars" (10812), and "Forks" (1203). There is also an "Embed" button with a dropdown menu showing a script source, and buttons for "Download ZIP" and "Raw". The main content area displays the text of the `introrx.md` file, which starts with the title "The introduction to Reactive Programming you've been missing" and credits "(by @andrestaltz)". The text continues with "This tutorial as a series of videos" and a paragraph about watching video tutorials with live-coding, mentioning a series recorded with the same contents as in an article on [Egghead.io](https://egghead.io). The text then discusses the difficulty of learning Reactive Programming, particularly its variant comprising of Rx, Bacon.js, RAC, and others, and mentions the lack of good material and the challenge of building the whole architecture.

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](https://egghead.io) - [Introduction to Reactive Programming](#).

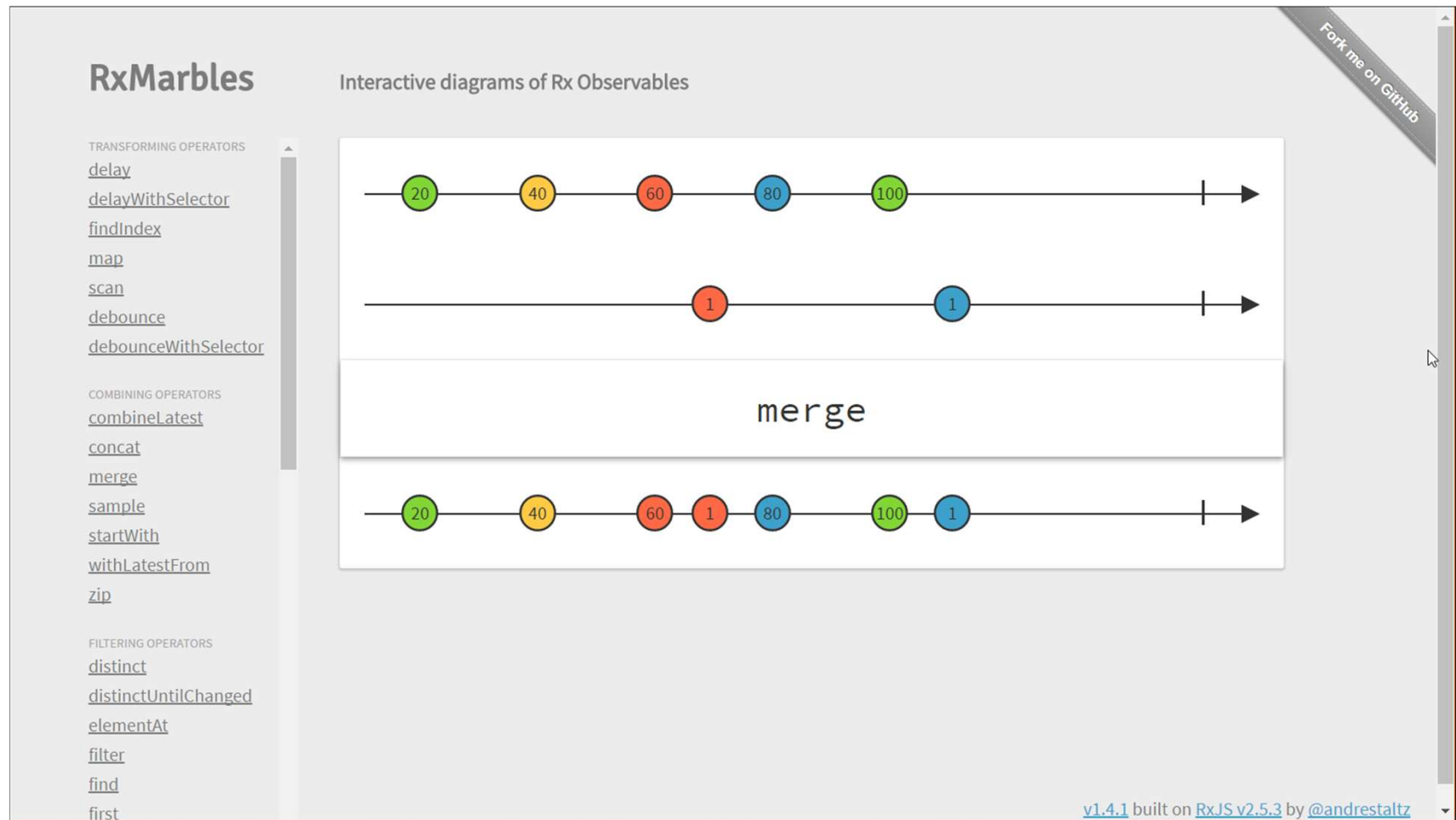
So you're curious in learning this new thing called Reactive Programming, particularly its variant comprising of Rx, Bacon.js, RAC, and others.

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>




# Also by Andre Stalz - RxMarbles



<http://rxmarbles.com/>

# Dan Wahlin on Modules and Observables

Integrating Angular with RESTful Services using RxJS and Observables



```
15 baseUrl: string = '/api/customers';
16
17 constructor(private http: Http) {
18
19 }
20
21 getCustomers() : Observable<ICustomer[]> {
22     return this.http.get(this.baseUrl)
23         .map((res: Response) => {
24             let customers = res.json();
25             this.calculateCustomersOrderTotal(customers);
26             return customers;
27         })
28         .catch(this.handleError);
29 }
30
31 getCustomersPage(page: number, pageSize: number) : Observable<IPagedResults<ICustomer[]>
32     return this.http.get(`${this.baseUrl}/page/${page}/${pageSize}`)
33         .map((res: Response) => {
34             const totalRecords = +res.headers.get('x-inlinecount');
35             let customers = res.json();
36             this.calculateCustomersOrderTotal(customers);
37             return {
38                 results: customers,
```

52:13 / 1:24:02

<https://www.youtube.com/watch?v=YxK4UW4UfCk>