Global Knowledge®

Angular Fundamentals

Observables

Peter Kassenaar –
info@kassenaar.com

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`

ReactiveX Introduction Docs Languages Resources Community

ReactiveX

An API for asynchronou
with observable stream

Choose your platform

http://reactivex.io/

ReactiveX 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

DOCUMENTATION

Observable
Operators
Single

LANGUAGES

RxJava
RxJS
Rx.NET

RESOURCES

Tutorials

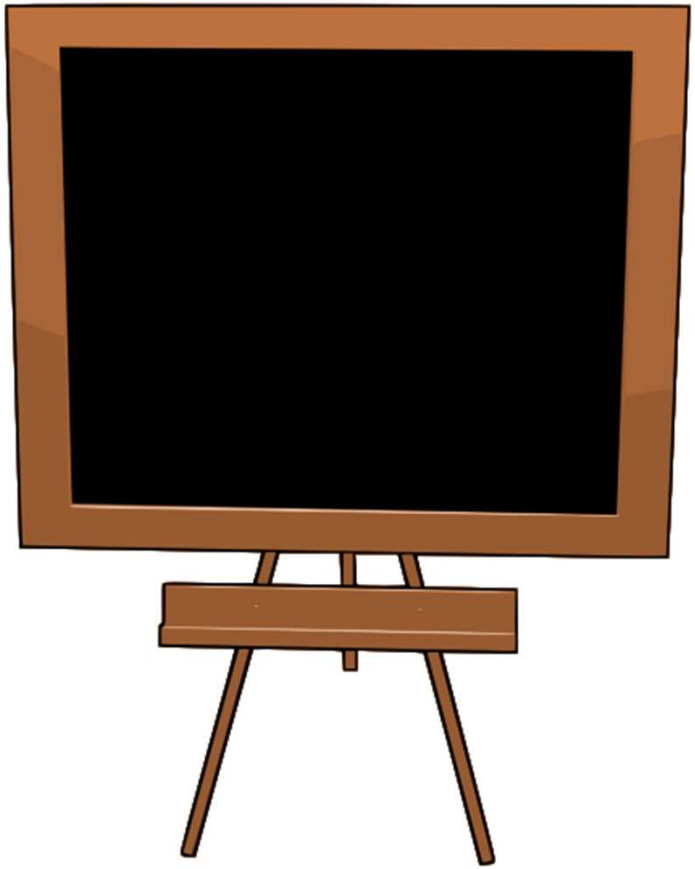
COMMUNITY

GitHub
Twitter
Others

The observable design pattern

"Understanding the observable stream"

Explanation



Why Observables?

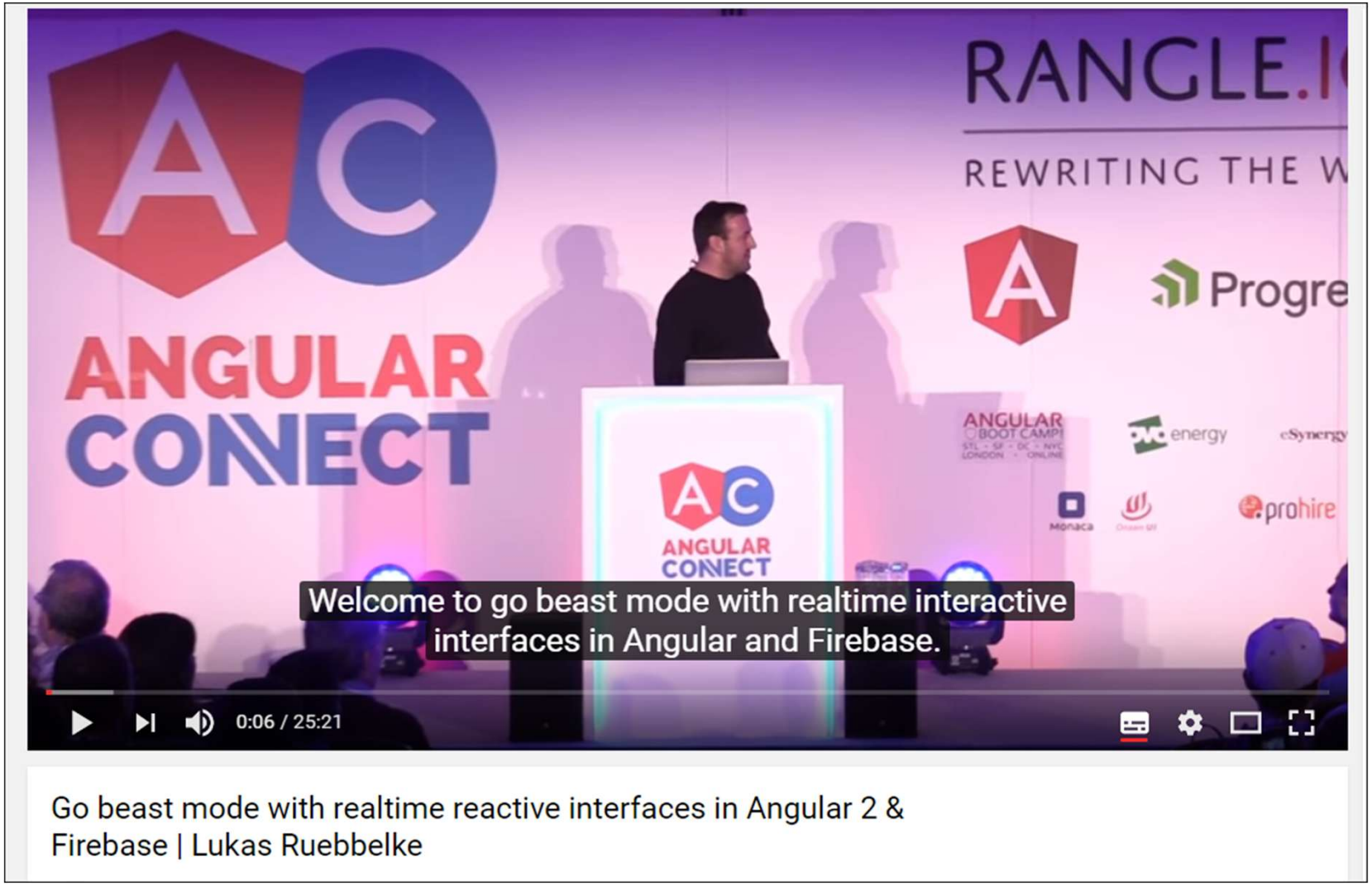
We can do much more with observables than with promises.

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

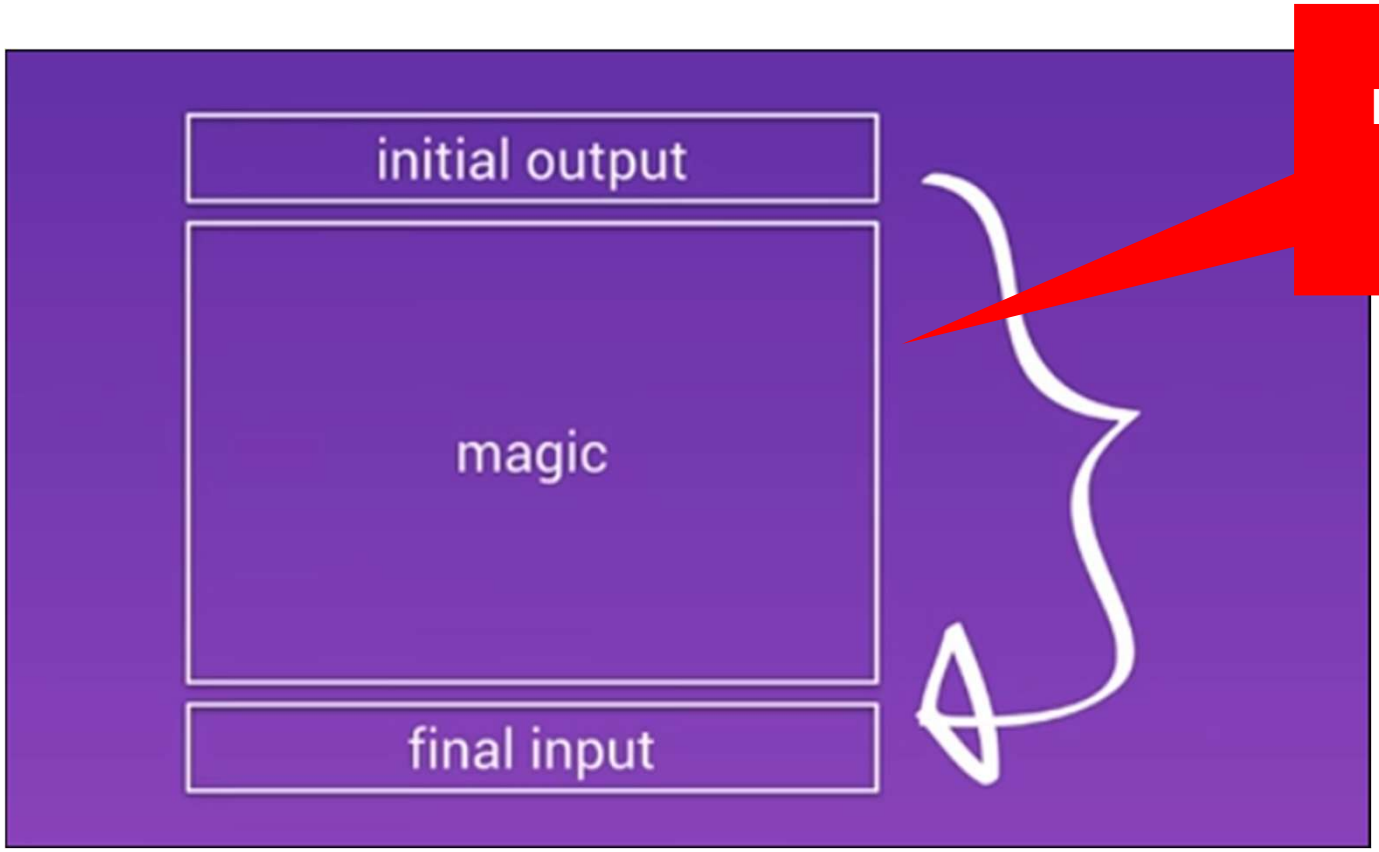
- "Reactive Programming"
  - *"Reactive programming is programming with asynchronous data streams."*
  - https://gist.github.com/staltz/868e7e9bc2a7b8c1f754
- Observables have a lot of extra options, as opposed to Promises
  - Mapping
  - Filtering
  - Combining
  - Cancel
  - Retry
  - …
- Consequence: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



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

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

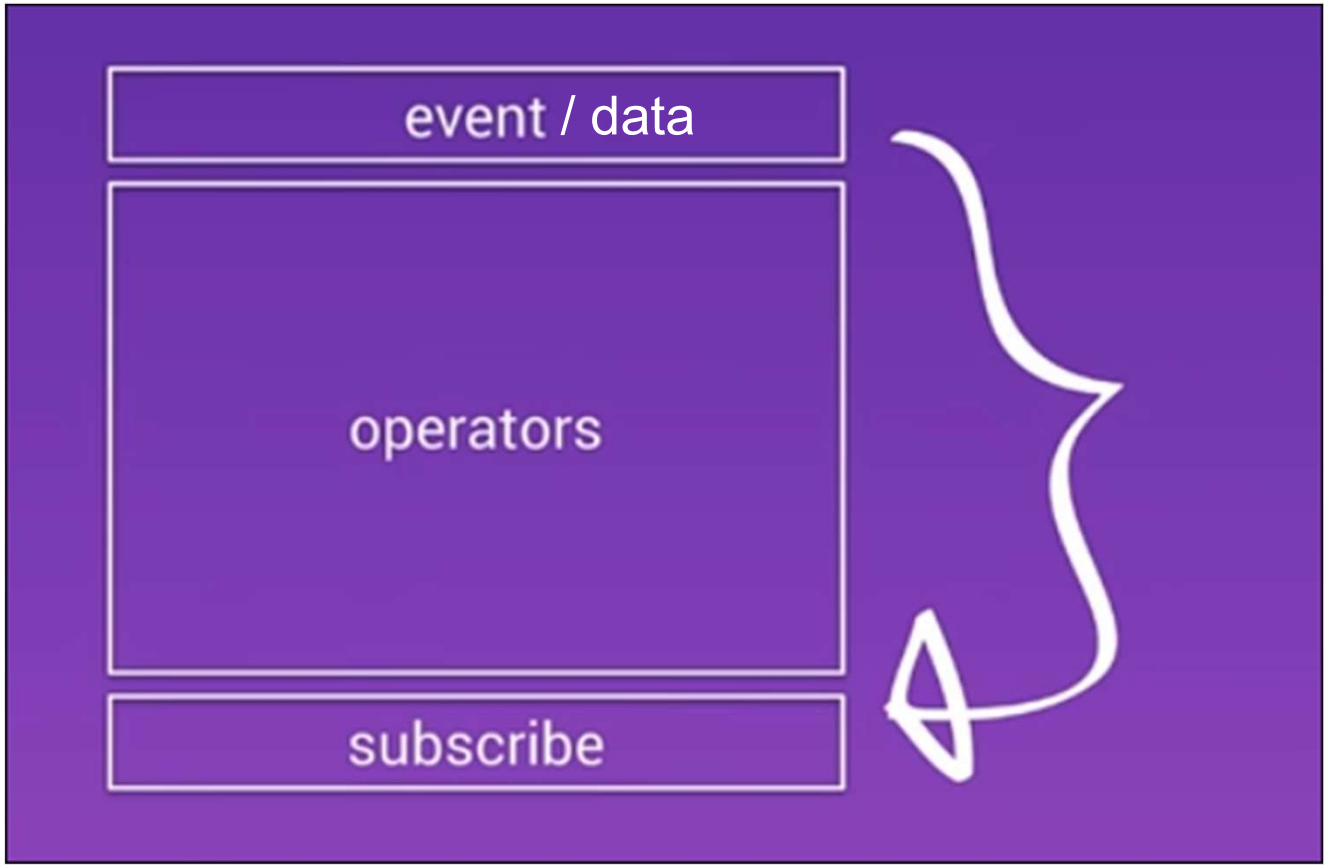
"The observable sandwich"



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,

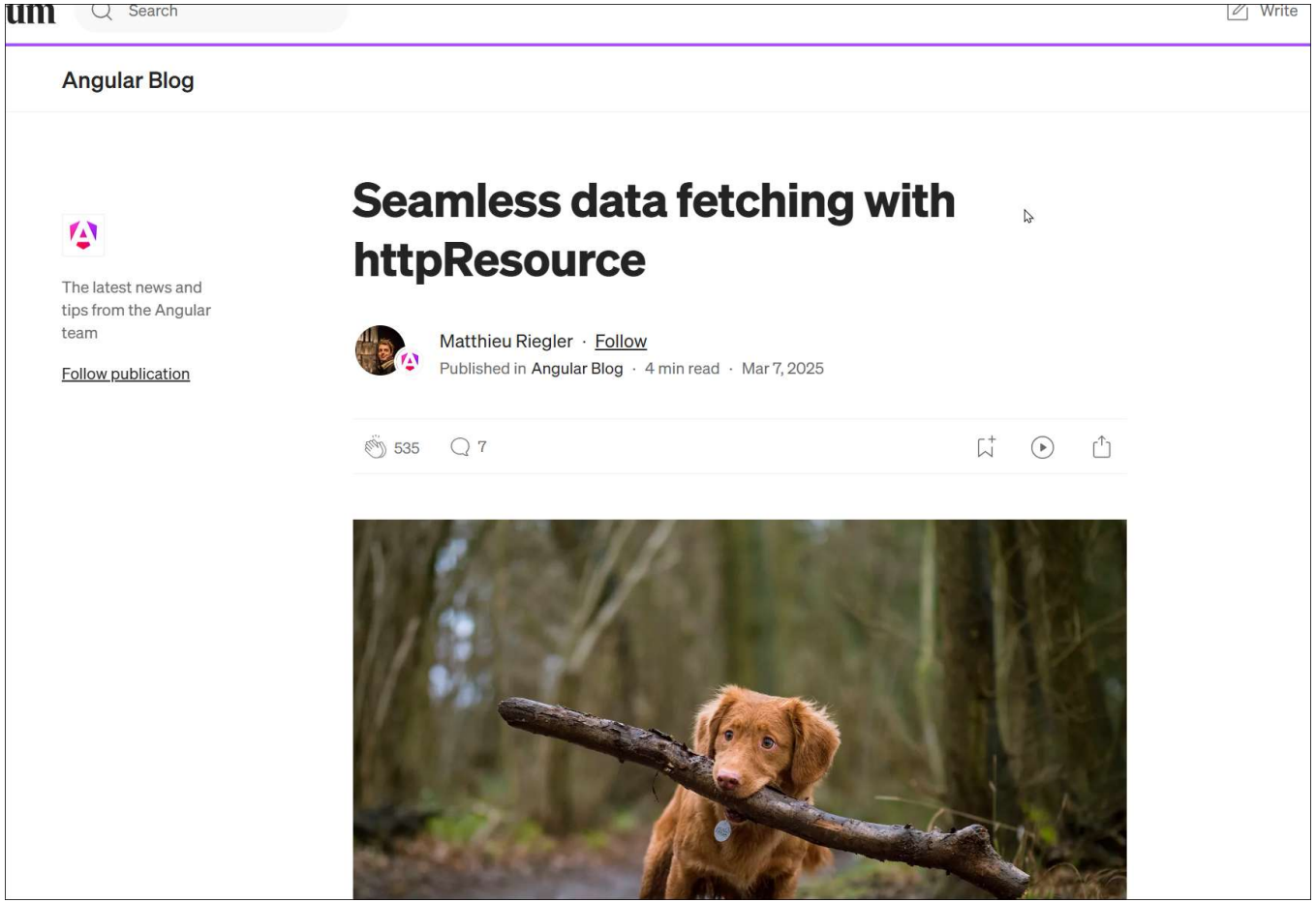
import `httpClientModule()`

```
// 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

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

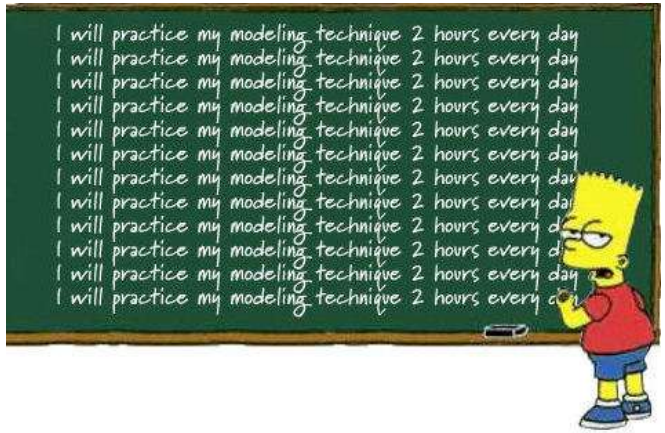
Initial Output

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()`

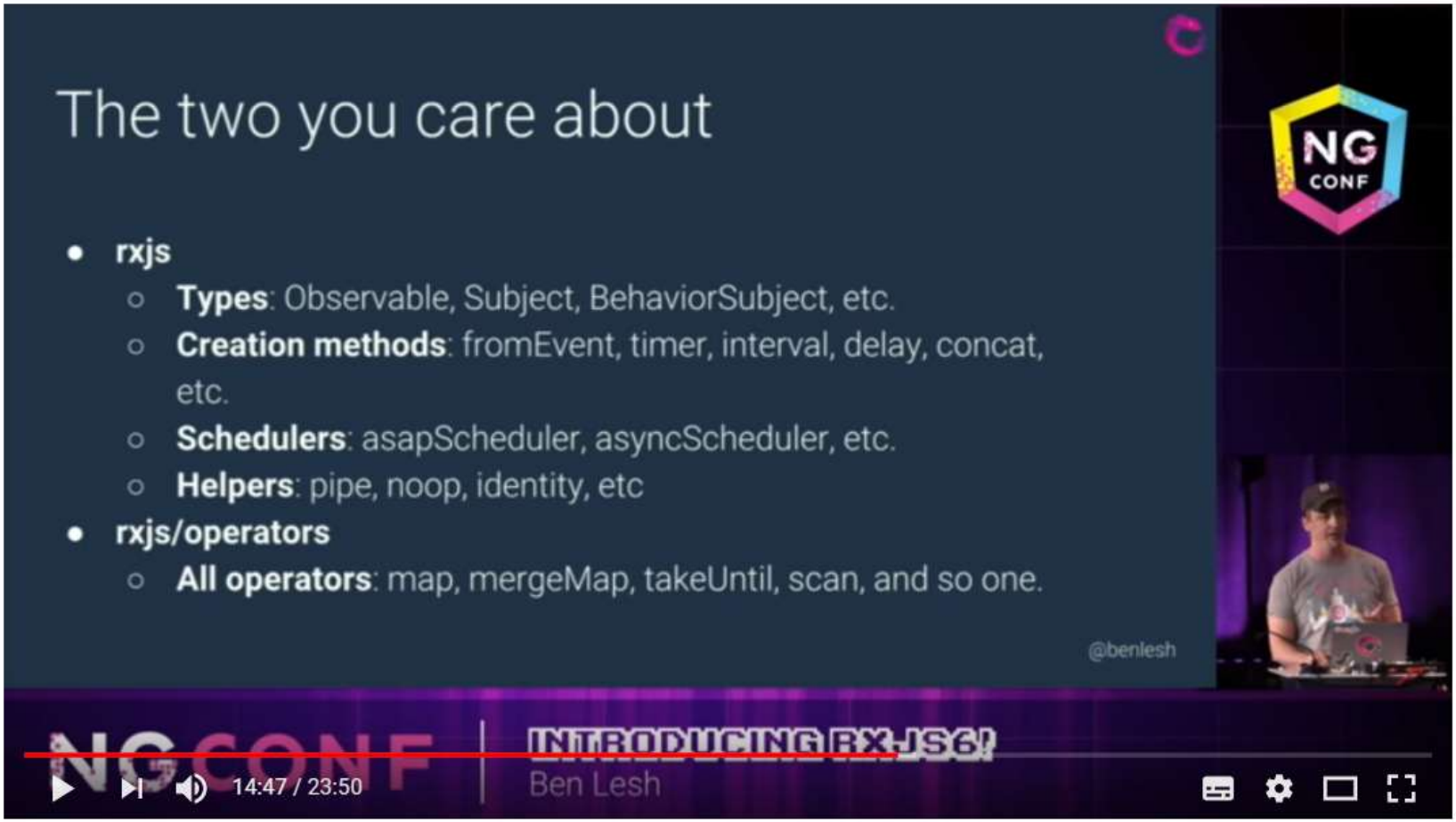
```
this.cityService.getCities()
   .subscribe(cityData => {
         this.cities = cityData
      },
      err => console.log(err),
      ()=> console.log('Getting cities complete...')
   )
```

Pipeable operators

- In RxJS 6.x and up: all operators inside `.pipe()` functie
- 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



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/

learnrxjs.io/learn-rxjs/operators

Learn RxJS

Introduction

LEARN RXJS

Operators
- Combination
- Conditional
- Creation
- Error Handling
- Multicasting
- Filtering
- Transformation
- Utility
- Full Listing

Subjects

Recipes

Concepts

Powered by GitBook

Operators

CONTENTS

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)

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

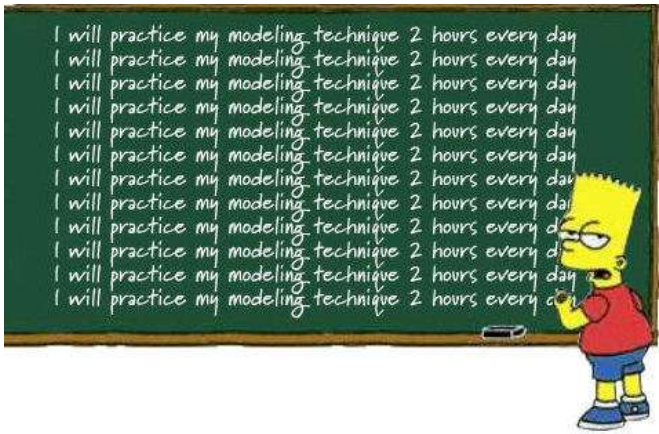
- MovieApp
- `examples\210-services-live`

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

npm

Search packages

Search

json-server

0.17.0 • Public • Published a month ago

Readme | Explore BETA | 20 Dependencies | 254 Dependents | 124 Versions

JSON Server

Node.js CI passing

Get a full fake REST API with **zero coding** in **less than 30 seconds** (seriously)

Created with <3 for front-end developers who need a quick back-end for prototyping and mocking.

- 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

See also:

- husky - Git hooks made easy
- lowdb - local JSON database
- xv - the most minimalist test runner

Gold sponsors

Install

```
> npm i json-server
```

Repository

github.com/typicode/json-server

Homepage

github.com/typicode/json-server

Weekly Downloads

173,210

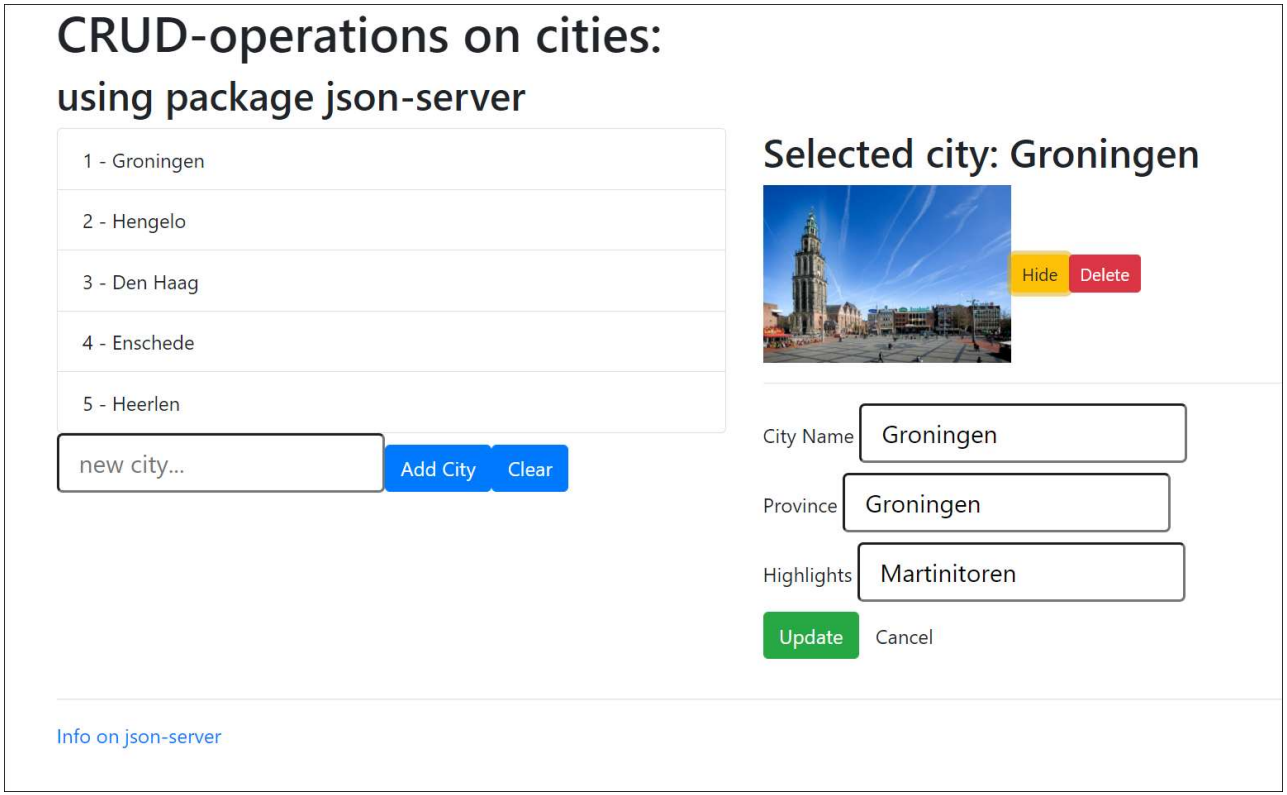
Version	License
0.17.0	MIT

Unpacked Size	Total Files
58.2 kB	26

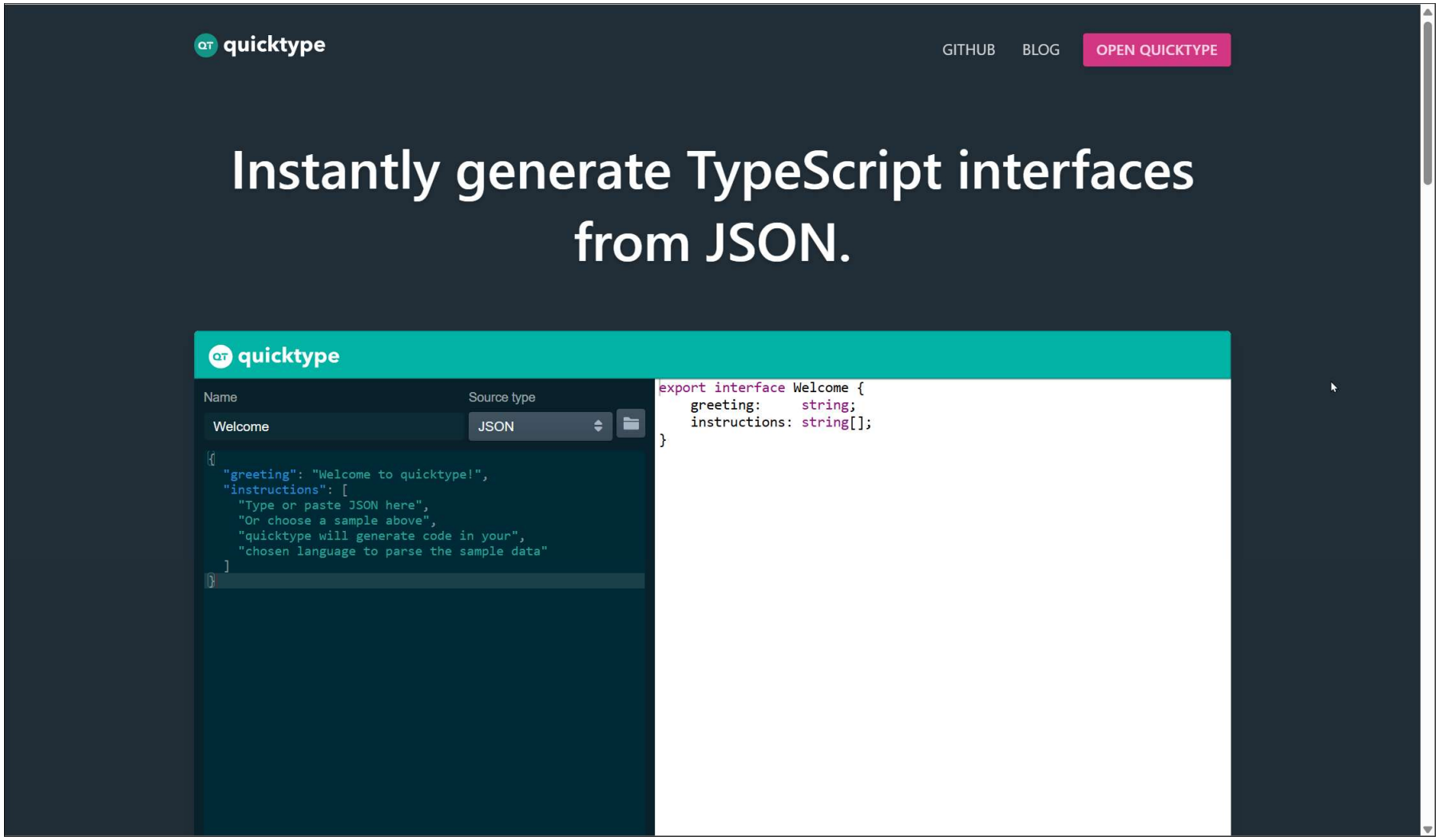
https://www.npmjs.com/package/json-server

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`



Creating TypeScript from json



https://quicktype.io/typescript

Official documentation…

RxJS Home Manual Reference Source Test dark theme light theme

- AsyncSubject
- BehaviorSubject
- Notification
- Observable
- ReplaySubject
- Scheduler
- AnonymousSubject
- Subject
- SubjectSubscriber
- Subscriber
- Subscription
- ObservableInput
- Observer
- SubscribableOrPromise
- TeardownLogic
- Rx.Scheduler
- Rx.Symbol

observable
- ConnectableObservable

observable/dom
- AjaxError
- AjaxResponse
- AjaxTimeoutError
- AjaxRequest

operator
- GroupedObservable

scheduler
- Action

```
import {Observable} from '@reactivex/rxjs/es6/Observable.js'
```

public class | source

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	bindCallback(func: function, selector: function, scheduler: Scheduler): function(...params: *): Observable Converts a callback API to a function that returns an Observable.
public static	bindNodeCallback(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
id	Row Number	blank: 0 % fx
first_name	First Name	blank: 0 % fx
last_name	Last Name	blank: 0 % fx
email	Email Address	blank: 0 % fx
gender	Gender	blank: 0 % fx
ip_address	IP Address v4	blank: 0 % fx

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/

Learn RxJS

Introduction

LEARN RXJS

Operators
- Combination
- Conditional
- Creation
- Error Handling
- Multicasting
- Filtering
- Transformation
- Utility
- Full Listing

Subjects

Recipes

Concepts

Powered by GitBook

Operators

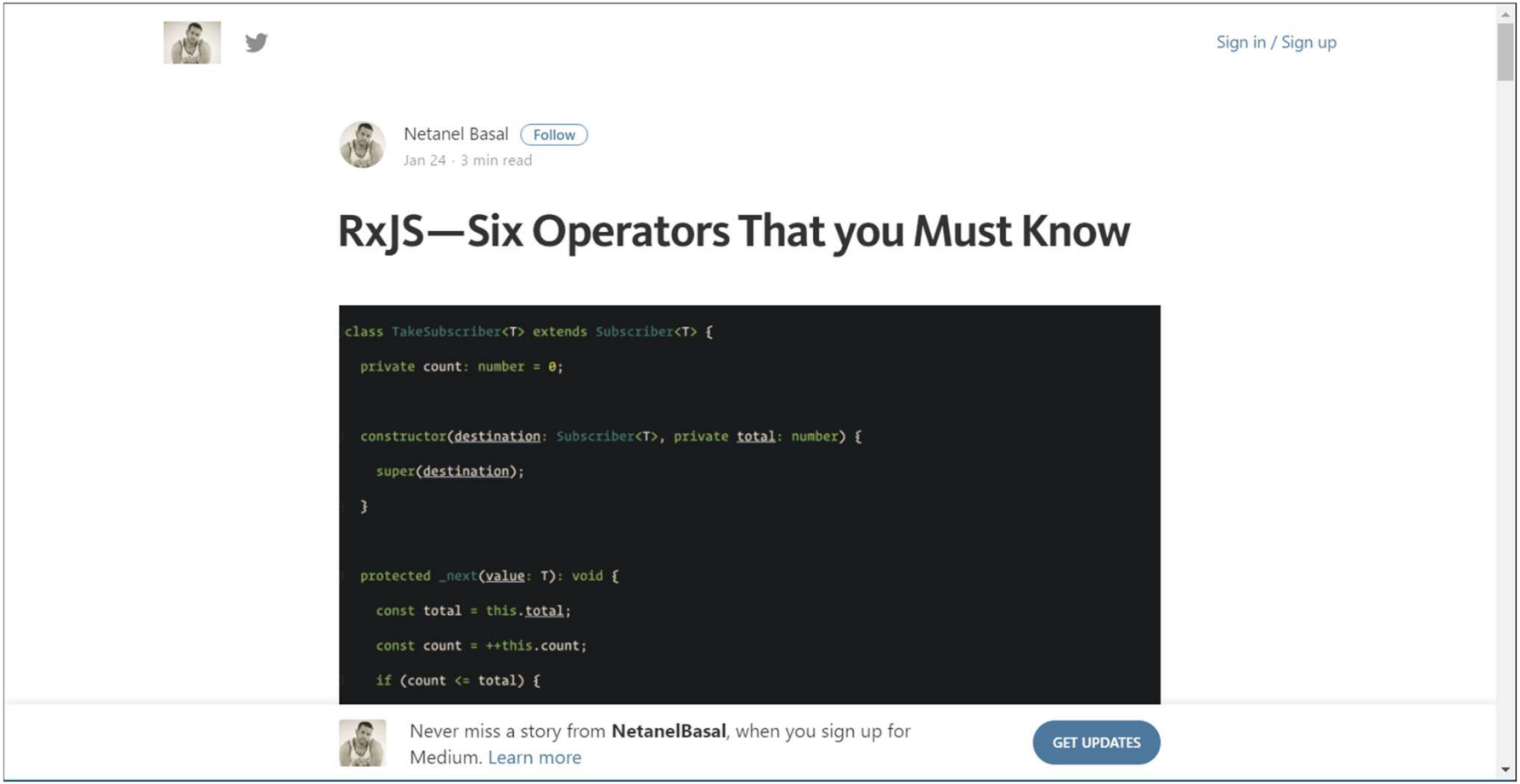
CONTENTS

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

Prefer a complete list in alphabetical order?

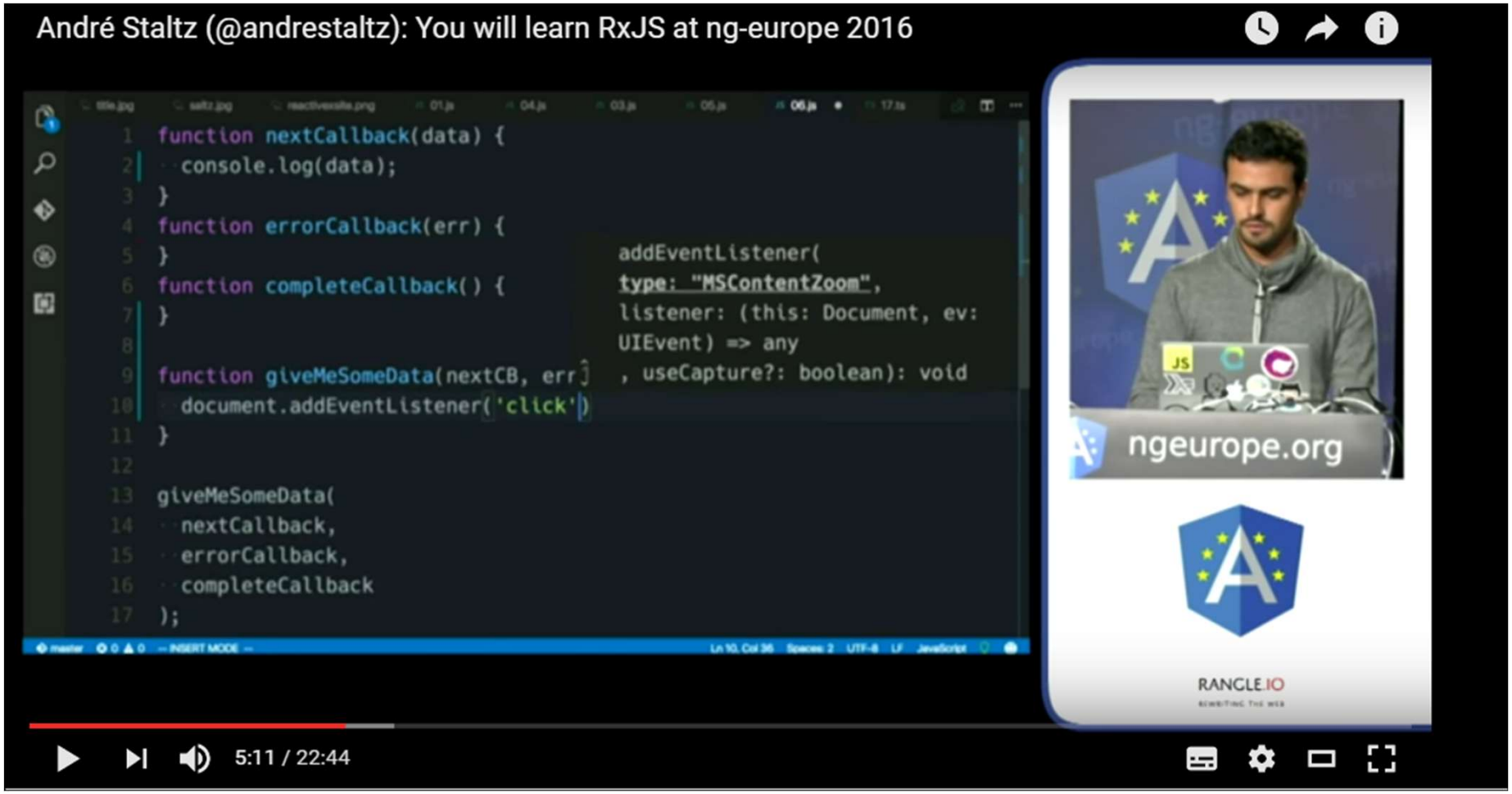
Contents (By Operator Type)

Article - 6 Operators you must know



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

Creating Observables from scratch - André Staltz



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

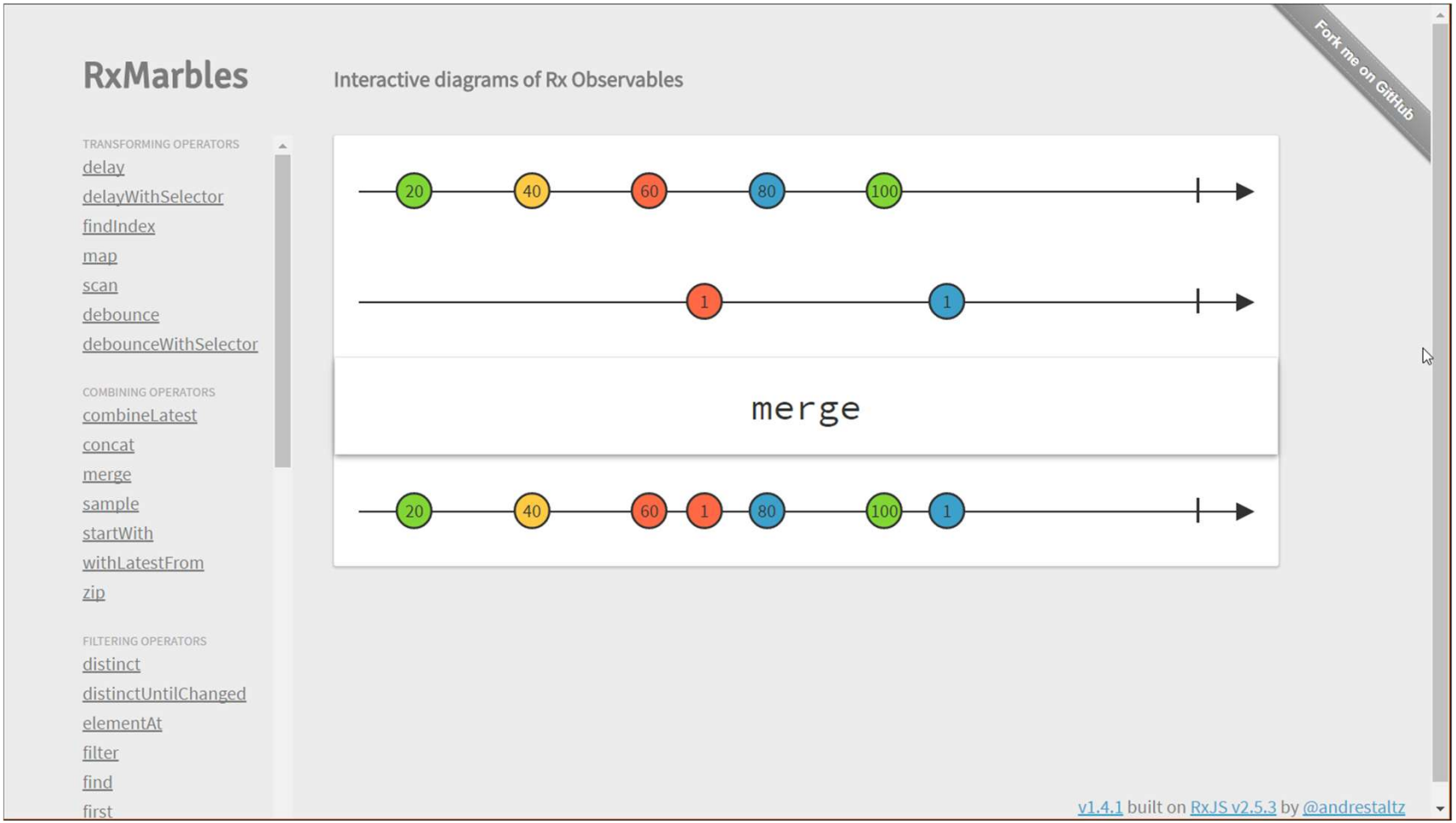
If you prefer to watch video tutorials with live-coding, then check out this series I recorded with the same contents as in this article: Egghead.io - Introduction to Reactive Programming.

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

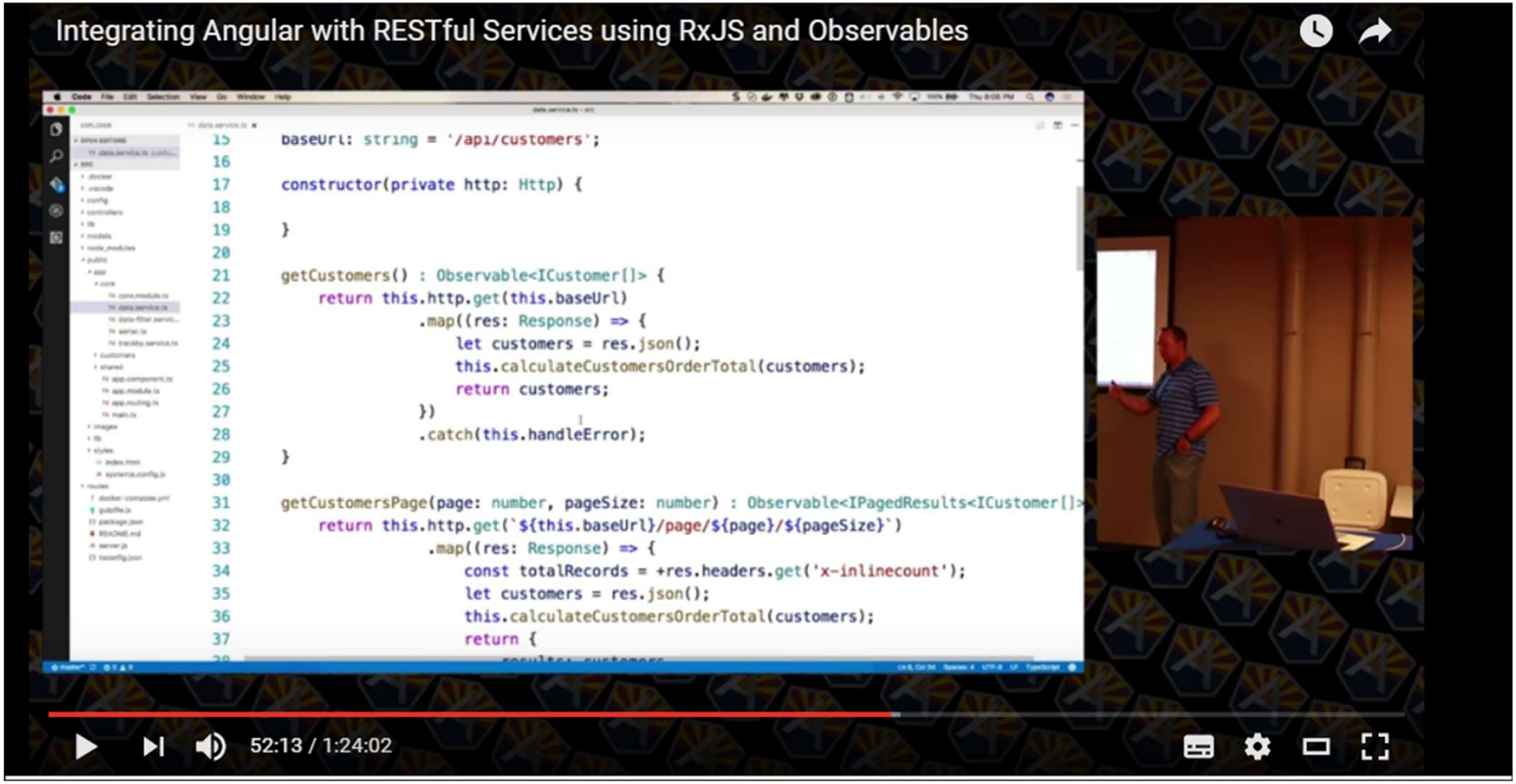
https://gist.github.com/staltz/868e7e9bc2a7b8c1f754

Also by Andre Stalz - RxMarbles



http://rxmarbles.com/

Dan Wahlin on Modules and Observables



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