incentro

Angular Fundamentals Module – Observables

Peter Kassenaar –
info@kassenaar.com

Async services met RxJS / Observables

Reactive programming with asynchronous streams

Async Services

- Statische data ophalen: *synchrone* actie
- Werken via `Http`: *asynchrone* actie
- Werken via `HttpClient`: *Angular 4.3+*
- Angular 1: `Promises`
- Angular 2: `Observables`

Bovendien in Angular 2: ReactiveX library RxJS

ReactiveX Introduction Docs Languages Resources Community

ReactiveX

An API for asynchronou
with observable stream

Choose your platform

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

http://reactivex.io/

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.

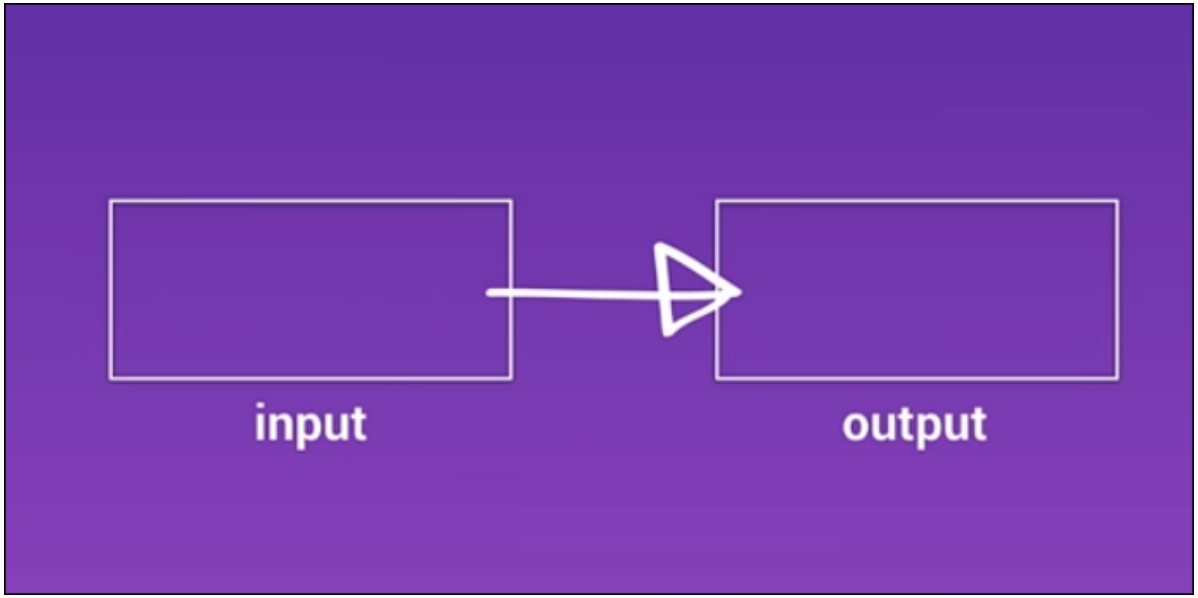
https://auth0.com/blog/2015/10/15/angular-2-series-part-3-using-http/

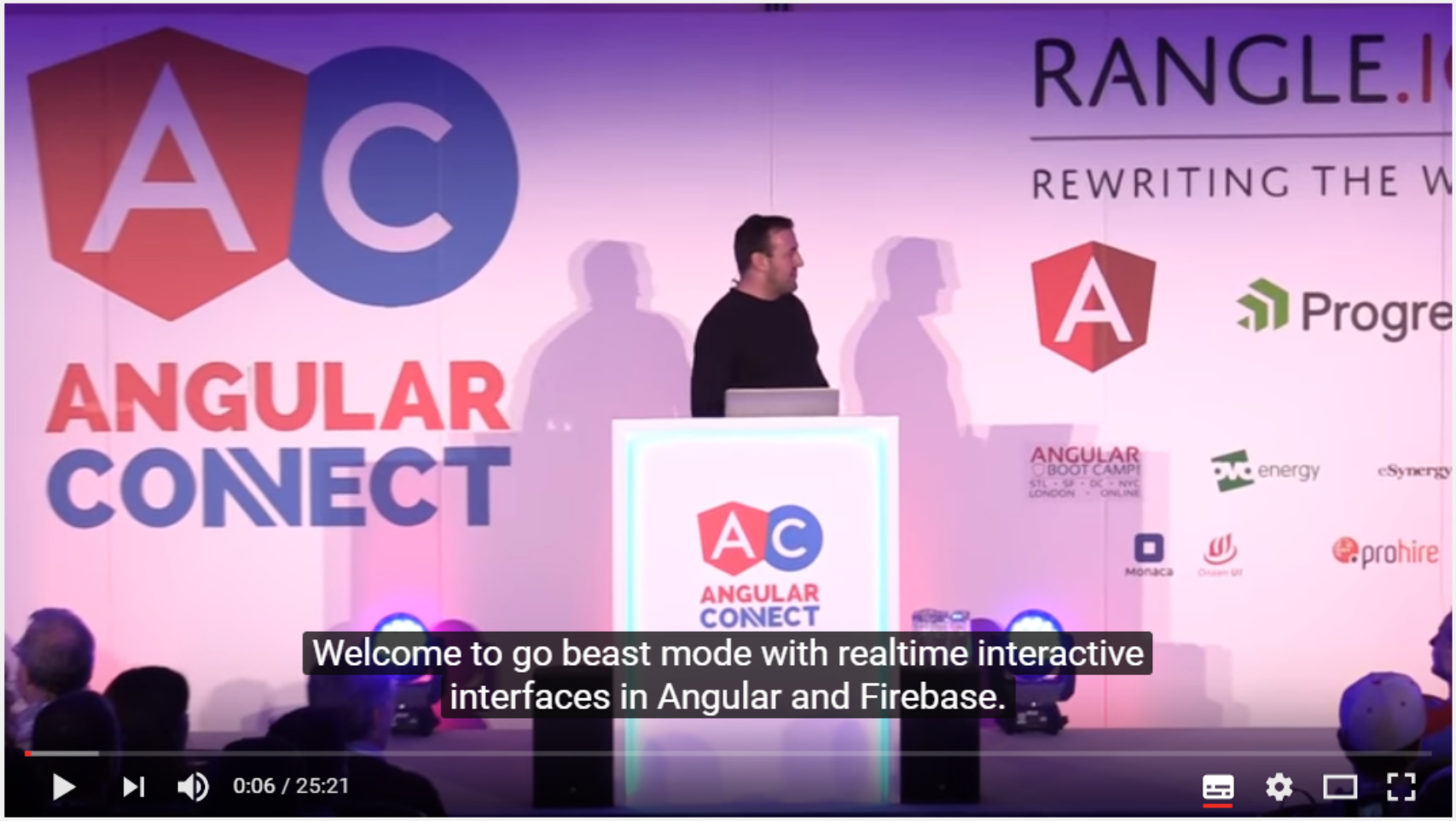
Observables en RxJs

- “Reactive Programming”
  - *“Reactive programming is programming with asynchronous data streams.”*
  - https://gist.github.com/staltz/868e7e9bc2a7b8c1f754
- Observables hebben extra mogelijkheden ten opzichte van Promises
  - Mapping
  - Filtering
  - Combining
  - Cancel
  - Retry
  - …
- Gevolg: géén `.success()`, `.error()` en `.then()` chaining meer!

How do observables work

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



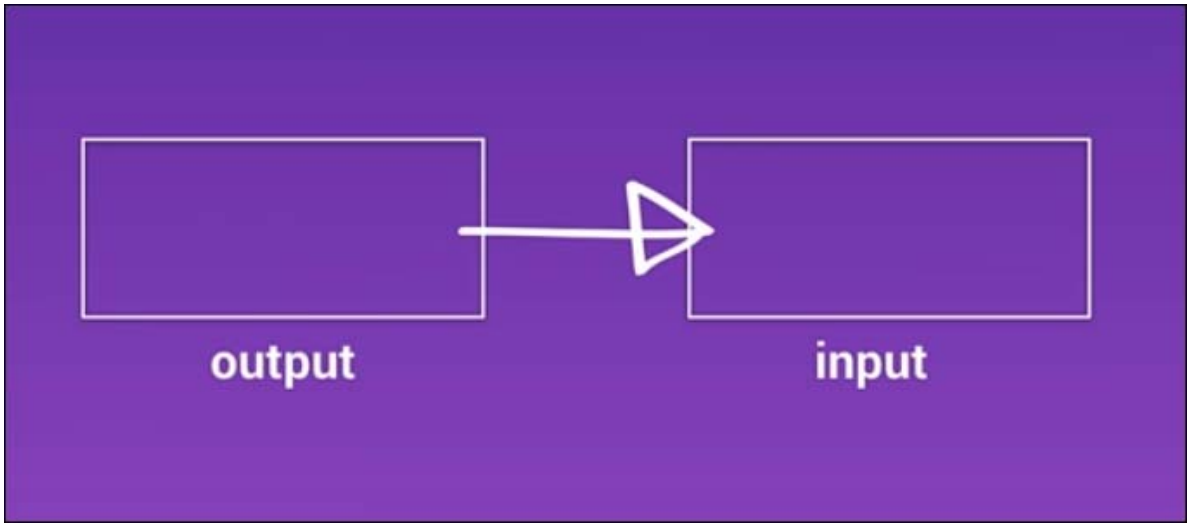


Go beast mode with realtime reactive interfaces in Angular 2 & Firebase | Lukas Ruebbelke

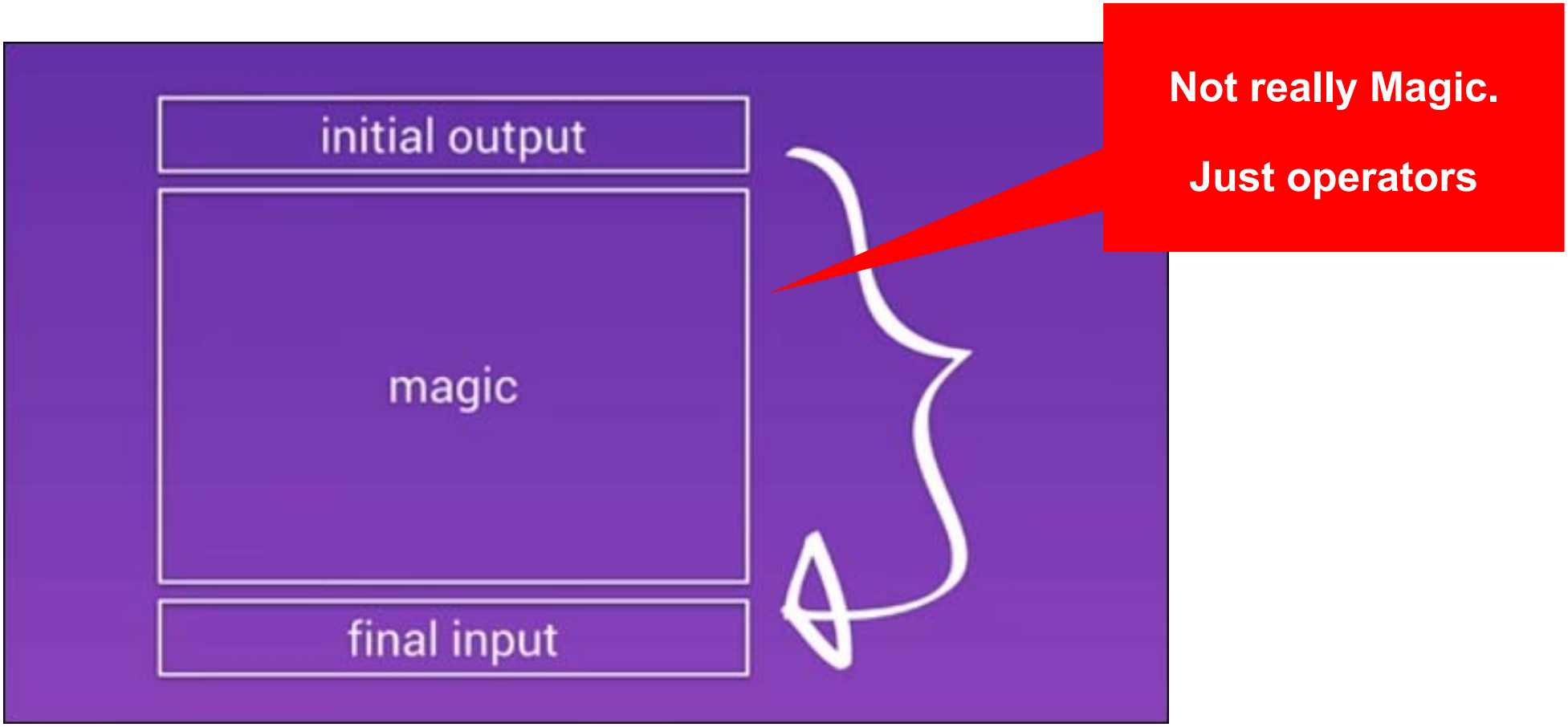
https://www.youtube.com/watch?v=5CTL7aqSvJU

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

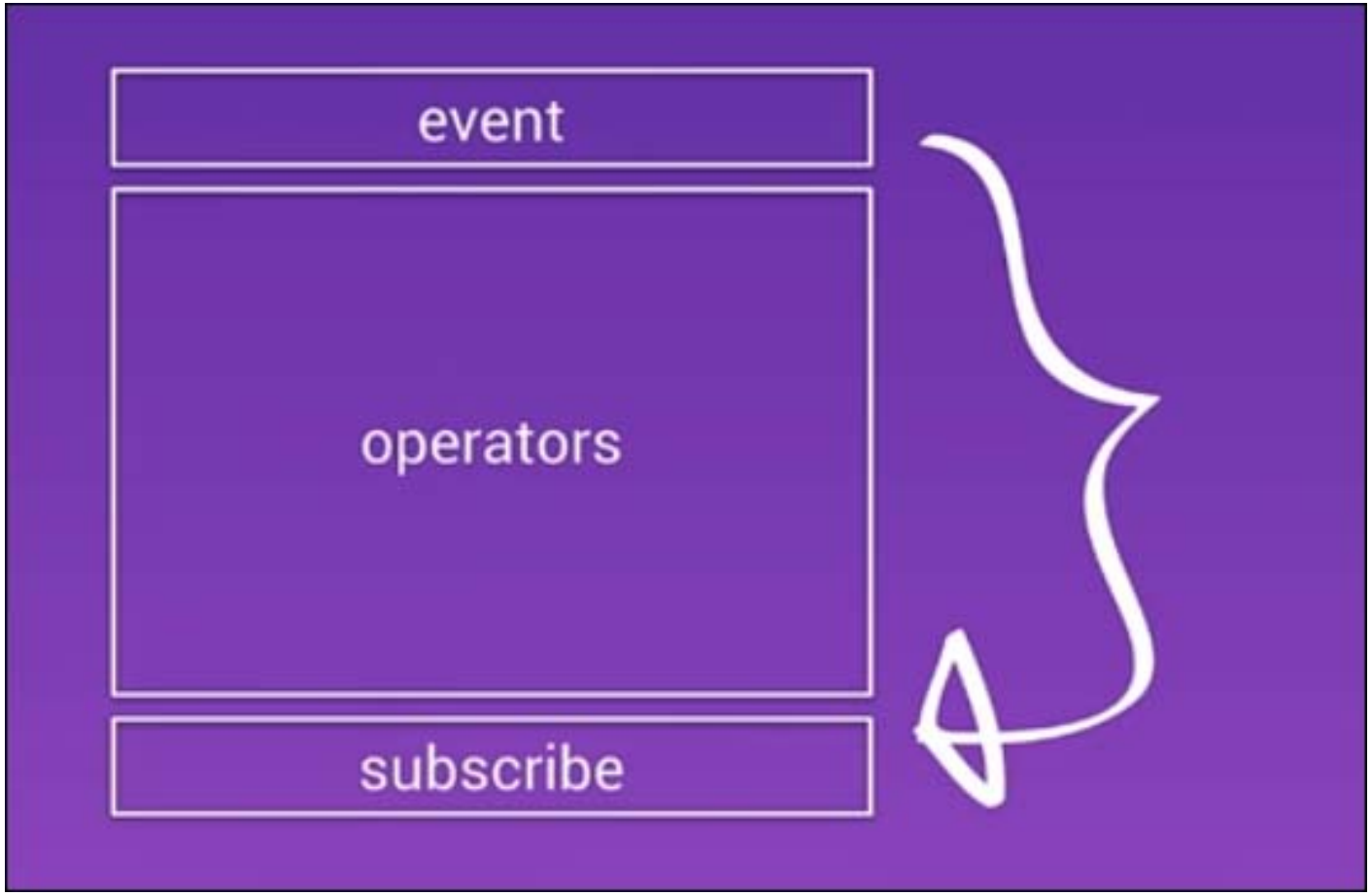
- With Observables -
  - a system, already outputting data,
  - Subscribe to that data
- "trade Output for Input"
- "Push vs. Pull"



"The observable sandwich"



Subscribe to events



In code (HttpModule, Angular 2/4)

```
this.http.get('assets/data/cities.json')
  .map(cities => cities.json())
  .subscribe((result) => {
    //... Do something
  });
```

Initial Output

Operator(s)

Final Input

Ook: importeren HttpModule in @ngModule

- *// Angular Modules*

  ...

- **import** {HttpModule} **from '@angular/http'**;

  // Module declaration

  @NgModule({

  imports : [BrowserModule, HttpModule],

  declarations: [AppComponent],

  bootstrap : [AppComponent],

  providers : [CityService] *// DI voor service*

  })

  export class AppModule {

  }

Angular 4.3/5+: HttpClientModule

- In je @ngModule: `imports : [HttpClientModule]`
- Niet meer `.map(res => res.json())`.
  - Json is de standaard!
- Nieuwe optie: Interceptors
- https://alligator.io/angular/httpclient-intro/ en
- https://alligator.io/angular/httpclient-interceptors/
- Is de standaard in Angular 5
  - `HttpModule` wordt in toekomstige versies verwijderd

Met `HttpClientModule` – geen mapping `.json()`

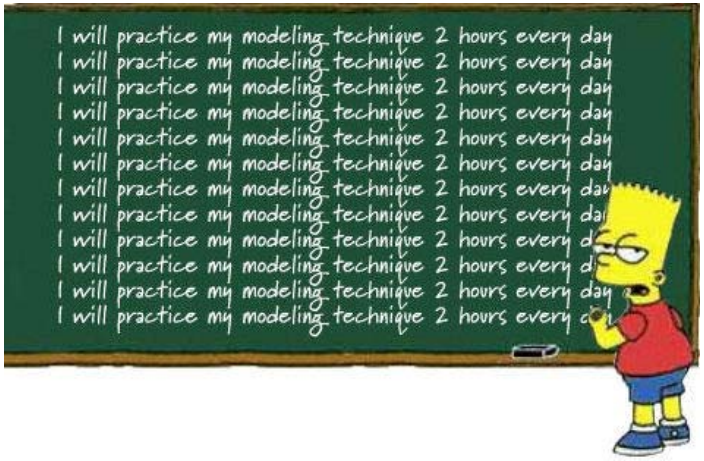
```
this.http.get<City[]>('assets/data/cities.json')
   .subscribe(result => {
      //... Do something
   });
```

Wel: Type opgeven en casting bij de `.get()` call

Oefening

- Bekijk het voorbeeld in `/201_services_http`
- Maak een eigen `.json`-bestand en importeer dit in je applicatie.
- Oefening `5c), 5d)`

Exercise….

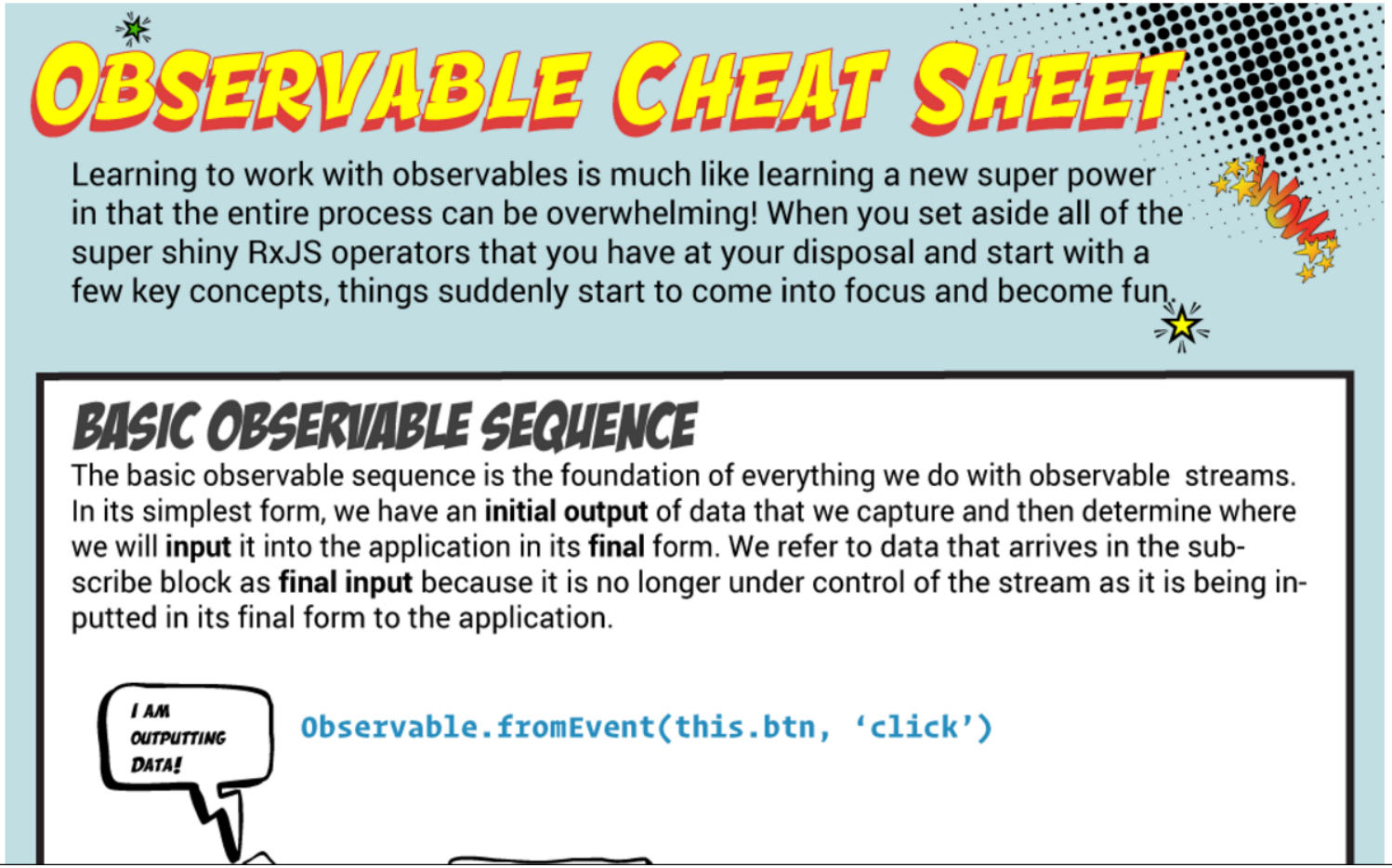


Observable Cheat Sheet

genius to understand.

You can download the full-sized infographic at http://bit.ly/observable-cheat-sheet.

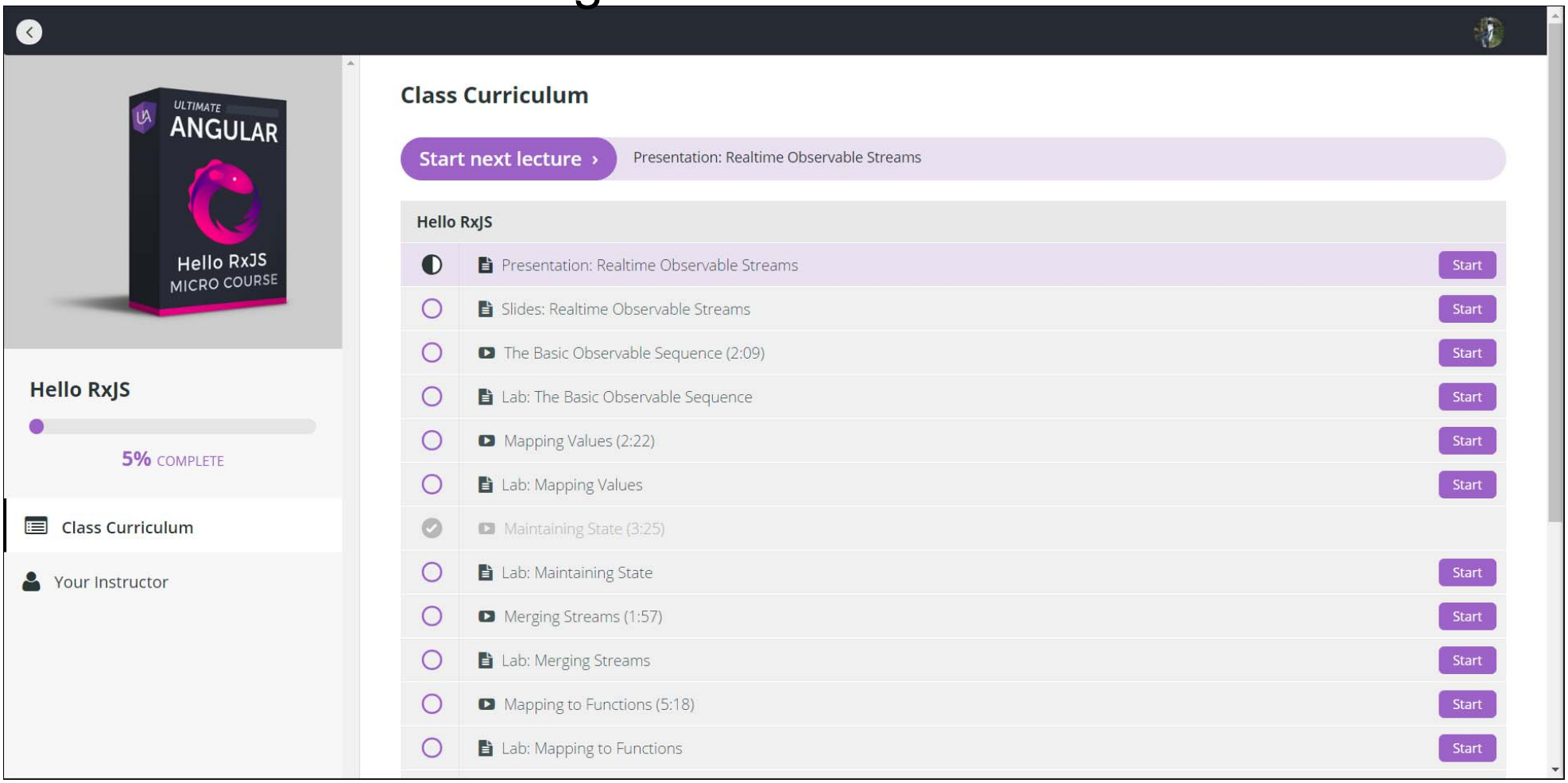
I really hope that you find the infographic helpful. Be sure to drop me a line below if you have any questions or comments. #highFive



http://onehungrymind.com/observable-cheat-sheet/

Hello RxJS

Gratis online training



http://courses.ultimateangular.com/

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/

Type to search

learn-rxjs

LEARN RXJS

Introduction

Operators

- Combination
  - combineAll
  - combineLatest
  - concat
  - concatAll
  - forkJoin
  - merge
  - mergeAll
  - pairwise
  - race
  - startWith
  - withLatestFrom
  - zip
- Conditional

EDIT THIS PAGE

Star 733 Watch 54

Learn RxJS

Clear examples, explanations, and resources for RxJS.

Introduction

RxJS is one of the hottest libraries in web development today. Offering a powerful, functional approach for dealing with events and with integration points into a growing number of frameworks, libraries, and utilities, the case for learning Rx has never been more appealing. Couple this with the ability to utilize your knowledge across nearly any language, having a solid grasp on reactive programming and what it can offer seems like a no-brainer.

But...

Learning RxJS and reactive programming is hard. There's the multitude of concepts, large API surface, and fundamental shift in mindset from an imperative to declarative style. This site focuses on making these concepts approachable, the examples clear and easy to explore, and features references throughout to the best RxJS related material on the web. The goal is to supplement the official docs and pre-existing learning material while offering a new, fresh perspective to clear any hurdles and tackle the pain points. Learning Rx may be difficult but it is certainly worth the effort!

Content

Async pipe

Automatische `.subscribe()` en `.unsubscribe()`

Async Pipe

- Bij `.subscribe()`, eigenlijk ook `.unsubscribe()` aanroepen.
  - Netjes!
  - Bij HTTP-requests niet beslist nodig, bij andere subscriptions wel, in verband met memory leaks.
- Niet meer zelf `.subscribe()` en `.unsubscribe()` aanroepen:
  - Gebruik async pipe van Angular

- In de component:

```
Cities$: Observable<City[]>; // Nu: Observable naar Type
…

ngOnInit() {
   // Call naar de service, retourneert Observable
   this.cities$ = this.cityService.getCities()
}
```

- In de view:

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

Werken met Live API's

- MovieApp
- `Oefeningen\210-services-live`

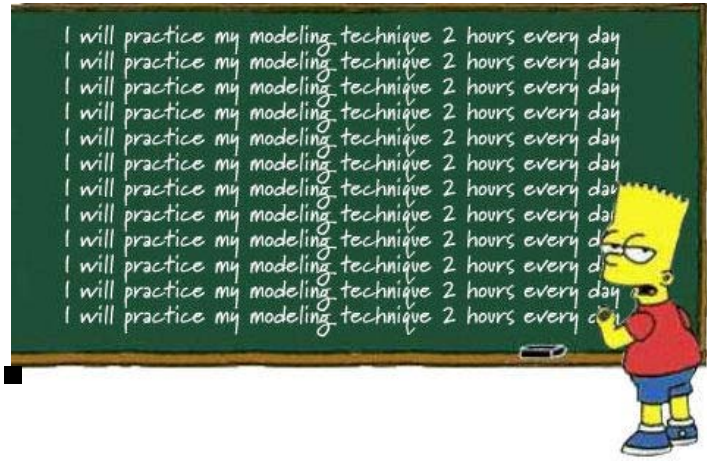
Voorbeeld API's

- https://pokeapi.co/ - Pokemon API
- http://openweathermap.org/API (weerbericht)
- http://filltext.com/ (random NAW-gegevens)
- http://ergast.com/mrd/ - Ergast Motor (F1) API
- http://www.omdbapi.com/ - Open Movie Database
- http://swapi.co/ - Star Wars API
- Zie ook `JavaScript APIs.txt` met meer voorbeelden

Exercise

- 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
  - …

Exercise….



json2ts

generate TypeScript interfaces from JSON

email feedback help

300.jpg"},{"Title":"The Amazing Captain Nemo","Year":"1978","imdbID":"tt0077156","Type":"movie","Poster":"https://images-na.ssl-images-amazon.com/images/M/MV5BMTc4NzExNjcwN15BMl5BanBnXkFtZTYwMTM1Mjg5._V1_SX300.jpg"},{"Title":"Nemo","Year":"1984","imdbID":"tt0087784","Type":"movie","Poster":"https://images-na.ssl-images-amazon.com/images/M/MV5BMTY2NzIwMTgwN15BMl5BanBnXkFtZTcwMjIyMzMzMQ@@._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/MV5BMzAxMzMyODQtNWY0Yy00N2M3LWE5MDQtZDUzNjc1ZGFmMzA4XkEyXkFqcGdeQXVyMzkxMzc4Mw@@._V1_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/

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

https://www.learnrxjs.io/

Type to search

learn-rxjs

LEARN RXJS

Introduction

Operators

- Combination
  - combineAll
  - combineLatest
  - concat
  - concatAll
  - forkJoin
  - merge
  - mergeAll
  - pairwise
  - race
  - startWith
  - withLatestFrom
  - zip
- Conditional

EDIT THIS PAGE

Star 733 Watch 54

Learn RxJS

Clear examples, explanations, and resources for RxJS.

Introduction

RxJS is one of the hottest libraries in web development today. Offering a powerful, functional approach for dealing with events and with integration points into a growing number of frameworks, libraries, and utilities, the case for learning Rx has never been more appealing. Couple this with the ability to utilize your knowledge across nearly any language, having a solid grasp on reactive programming and what it can offer seems like a no-brainer.

But...

Learning RxJS and reactive programming is hard. There's the multitude of concepts, large API surface, and fundamental shift in mindset from an imperative to declarative style. This site focuses on making these concepts approachable, the examples clear and easy to explore, and features references throughout to the best RxJS related material on the web. The goal is to supplement the official docs and pre-existing learning material while offering a new, fresh perspective to clear any hurdles and tackle the pain points. Learning Rx may be difficult but it is certainly worth the effort!

Content

Article - 6 Operators you must know

Sign in / Sign up

Netanel Basal Follow

Jan 24 · 3 min read

RxJS—Six Operators That you Must Know

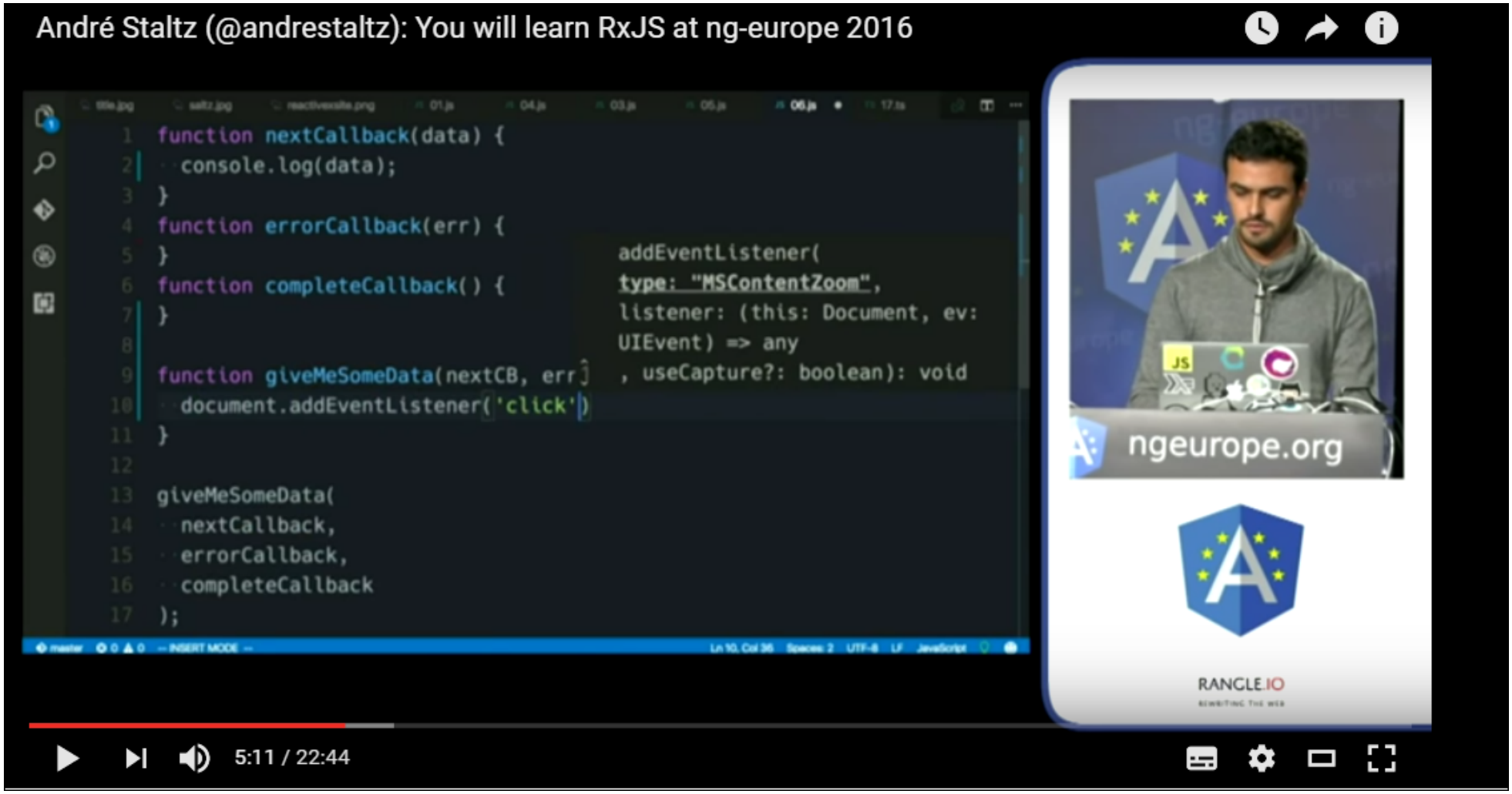
```
class TakeSubscriber<T> extends Subscriber<T> {
  private count: number = 0;

  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



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)

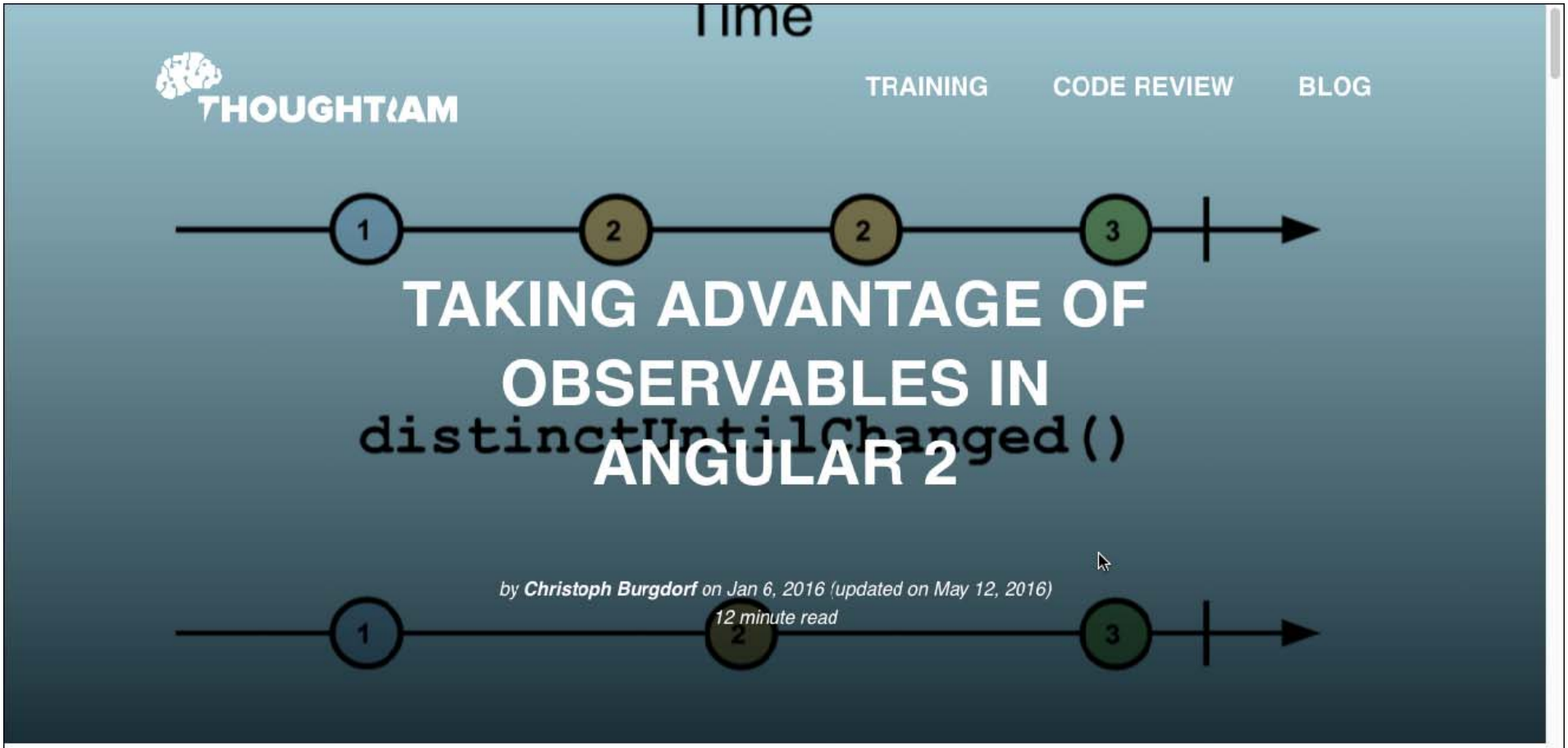
This tutorial as a series of videos

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



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 *lazyness* of Observables. In this article we like to focus on some practical advantages that

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