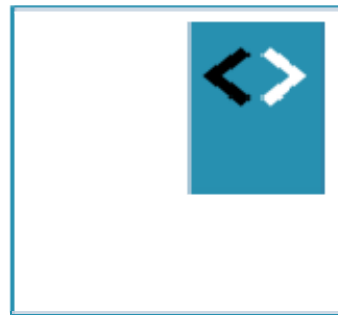




Angular Fundamentals Module – Observables



Peter Kassenaar –
info@kassenaar.com



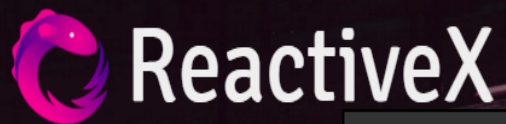
Async services met RxJS/Observables

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



An API for asynchronous
with observable stream

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](#) [🔗]
[RxJS](#) [🔗]
[Rx.NET](#) [🔗]
[RxScala](#)

RESOURCES

[Tutorials](#)

COMMUNITY

[GitHub](#) [🔗]
[Twitter](#) [🔗]
[Others](#)

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.

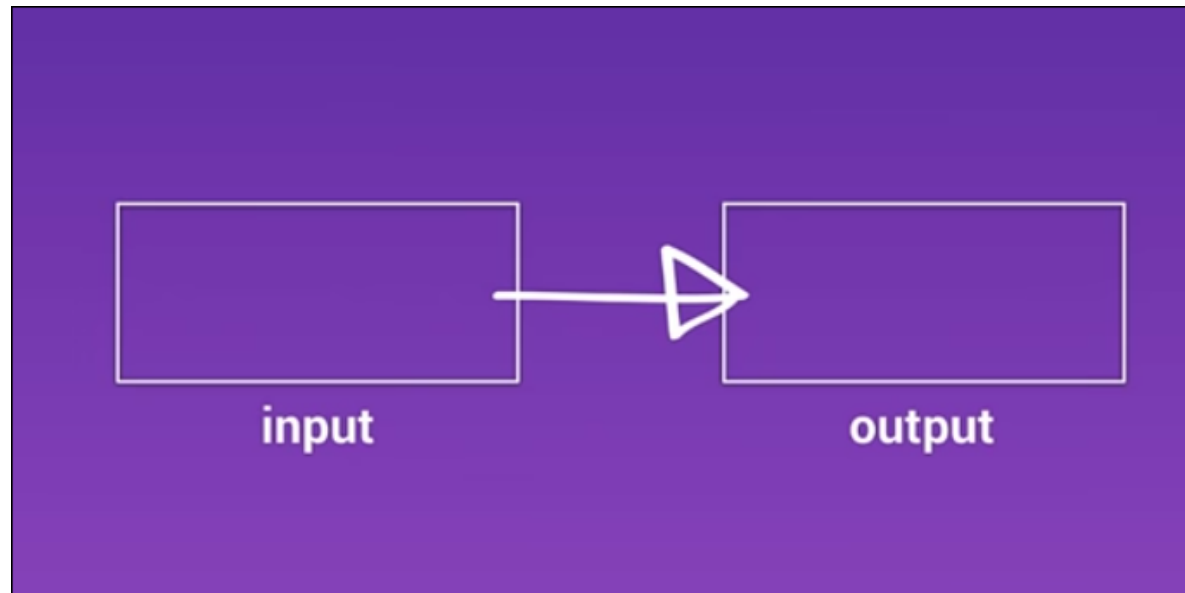
<https://auth0.com/blog/2015/10/15/angular-2-series-part-3-using-http/>

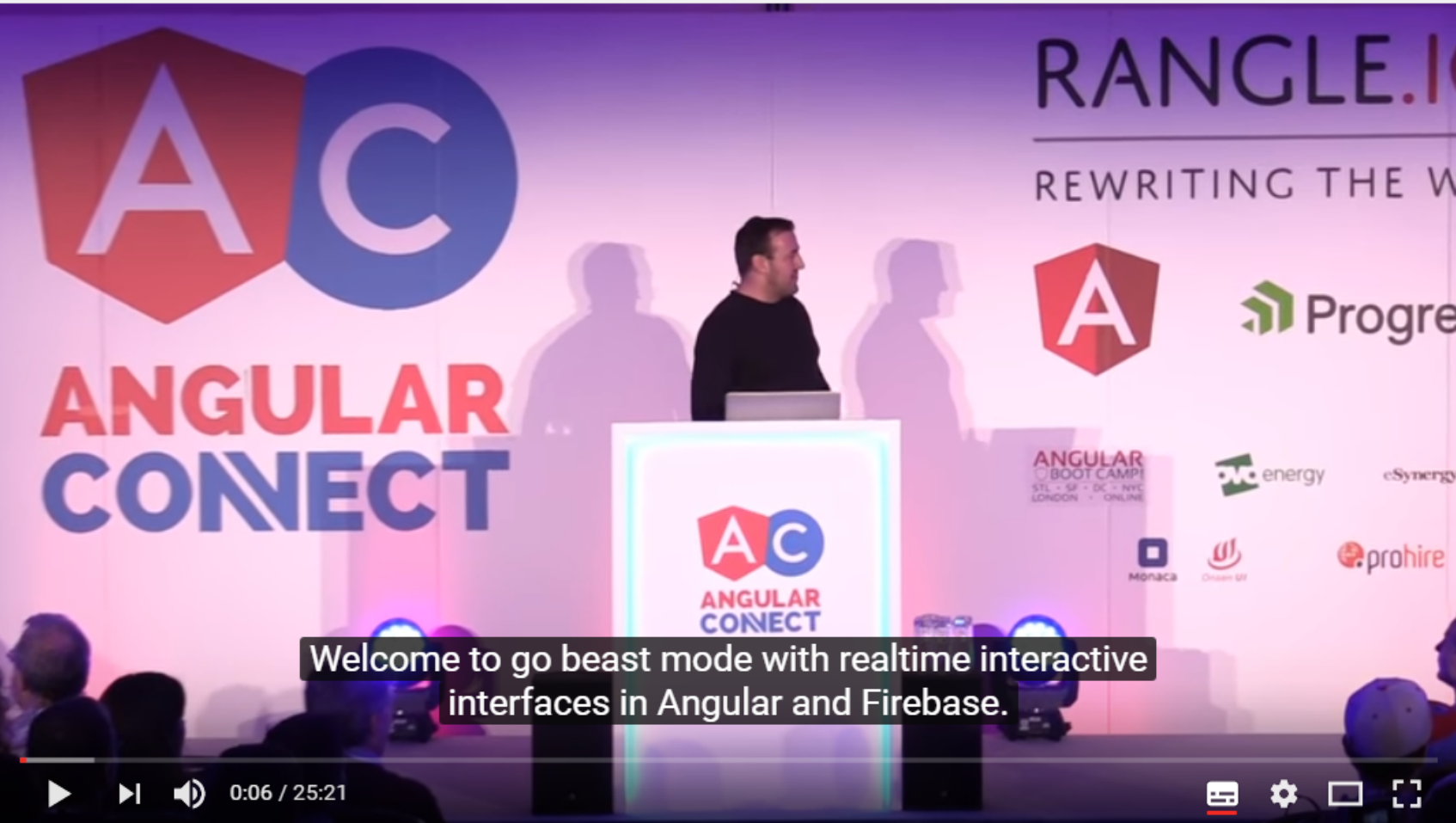
Observables en RxJs

- “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:





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 with the Angular Connect logo and the text "RANGLE.I" and "REWRITING THE W". The screen also displays logos for various companies including Progre, energy, eSynergy, MONACA, and prohire. A subtitle at the bottom of the video reads: "Welcome to go beast mode with realtime interactive interfaces in Angular and Firebase." The video player interface shows a progress bar at 0:06 / 25:21 and various control icons.

Angular Connect

RANGLE.I

REWRITING THE W

Progre

energy

eSynergy

MONACA

prohire

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

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

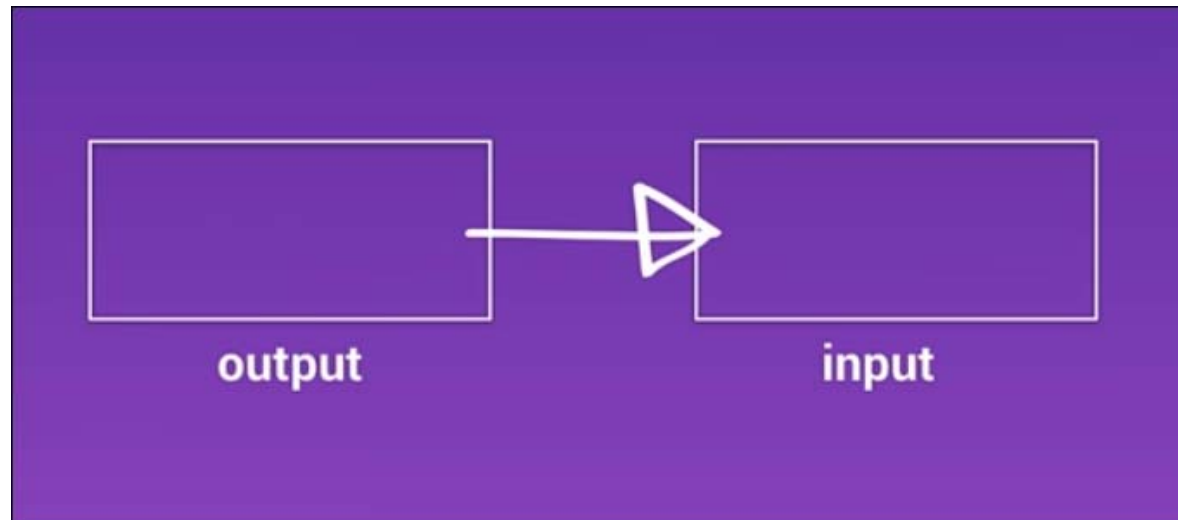
0:06 / 25:21

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

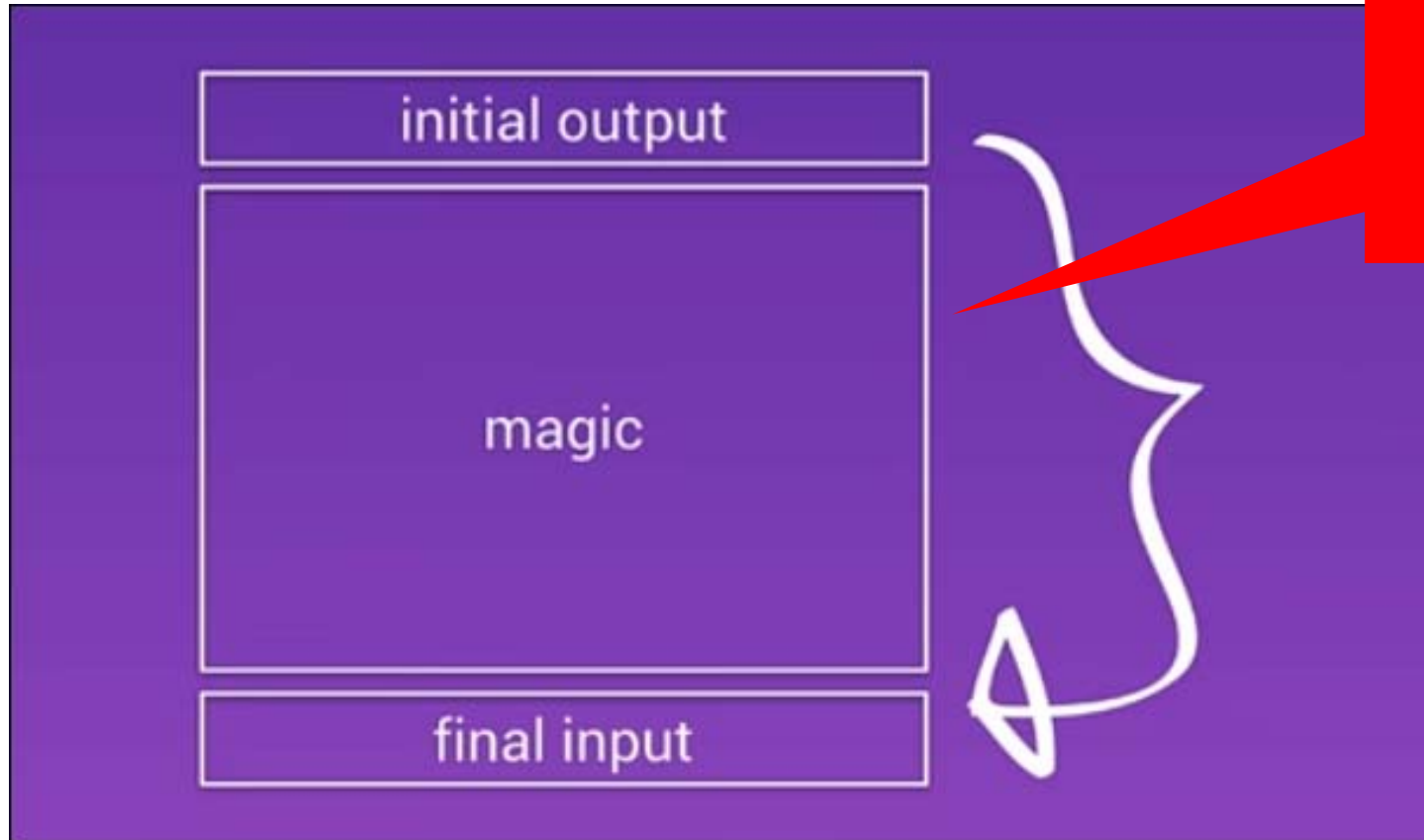
<https://www.youtube.com/watch?v=5CTL7aqSvJU>

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

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

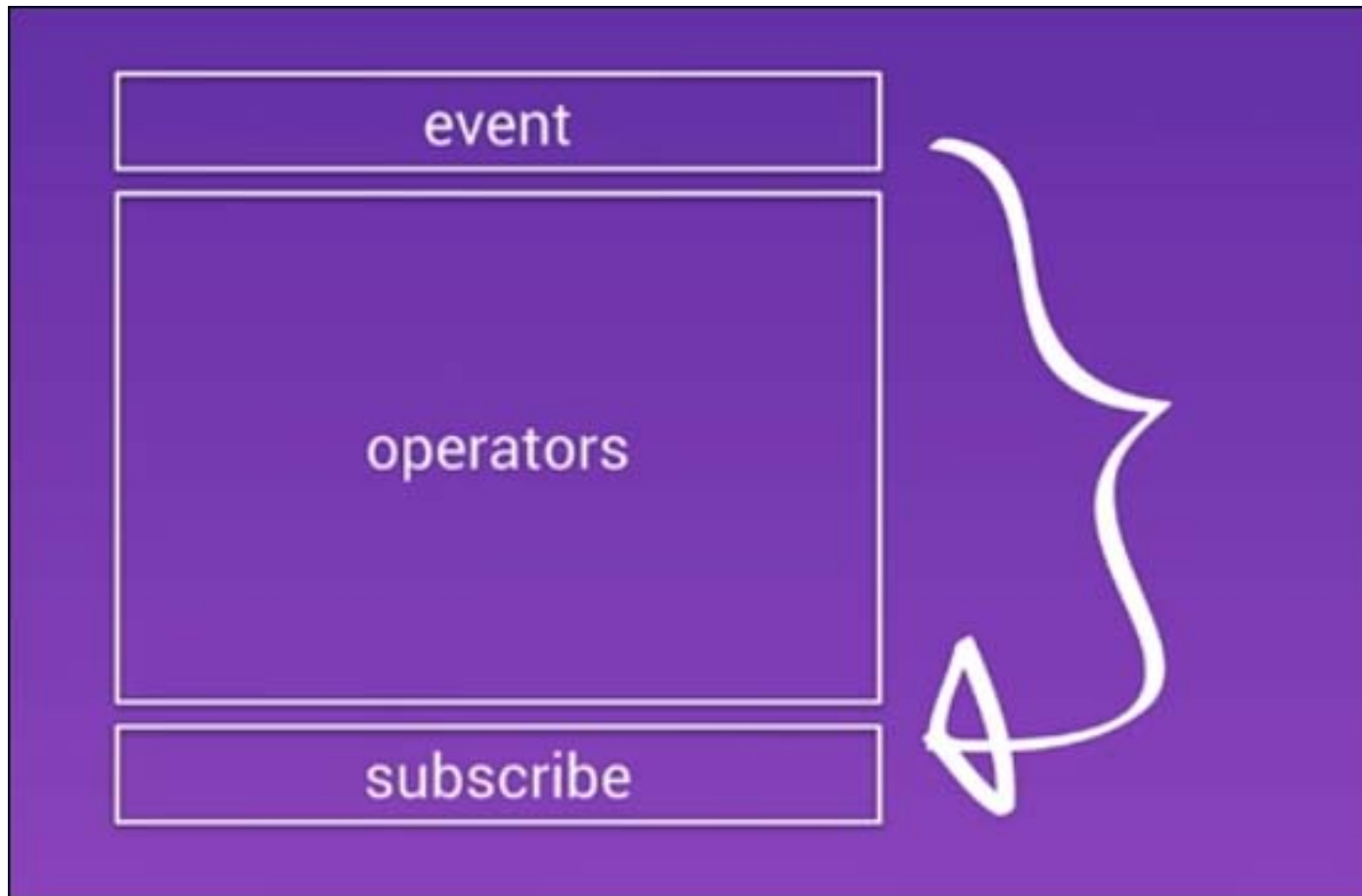


"The observable sandwich"



**Not really Magic.
Just operators**

Subscribe to events



In code (HttpModule, Angular 2/4)

Initial Output

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

Operator(s)

Final Input

Ook: importeren HttpClientModule in @ngModule

- *// Angular Modules*
...
- **import** { HttpClientModule } **from** '@angular/http';
// Module declaration
@NgModule({
 imports : [BrowserModule, HttpClientModule],
 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()

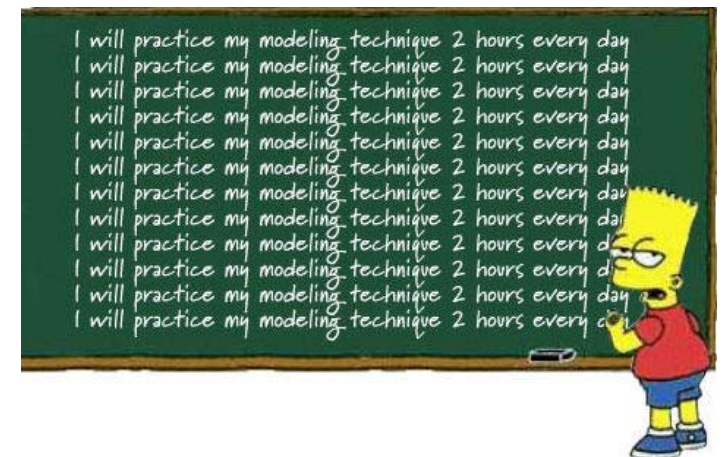
```
this.http.get<City[]>('assets/data/cities.json')  
  .subscribe(result => {  
    //... Do something  
  });
```

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....

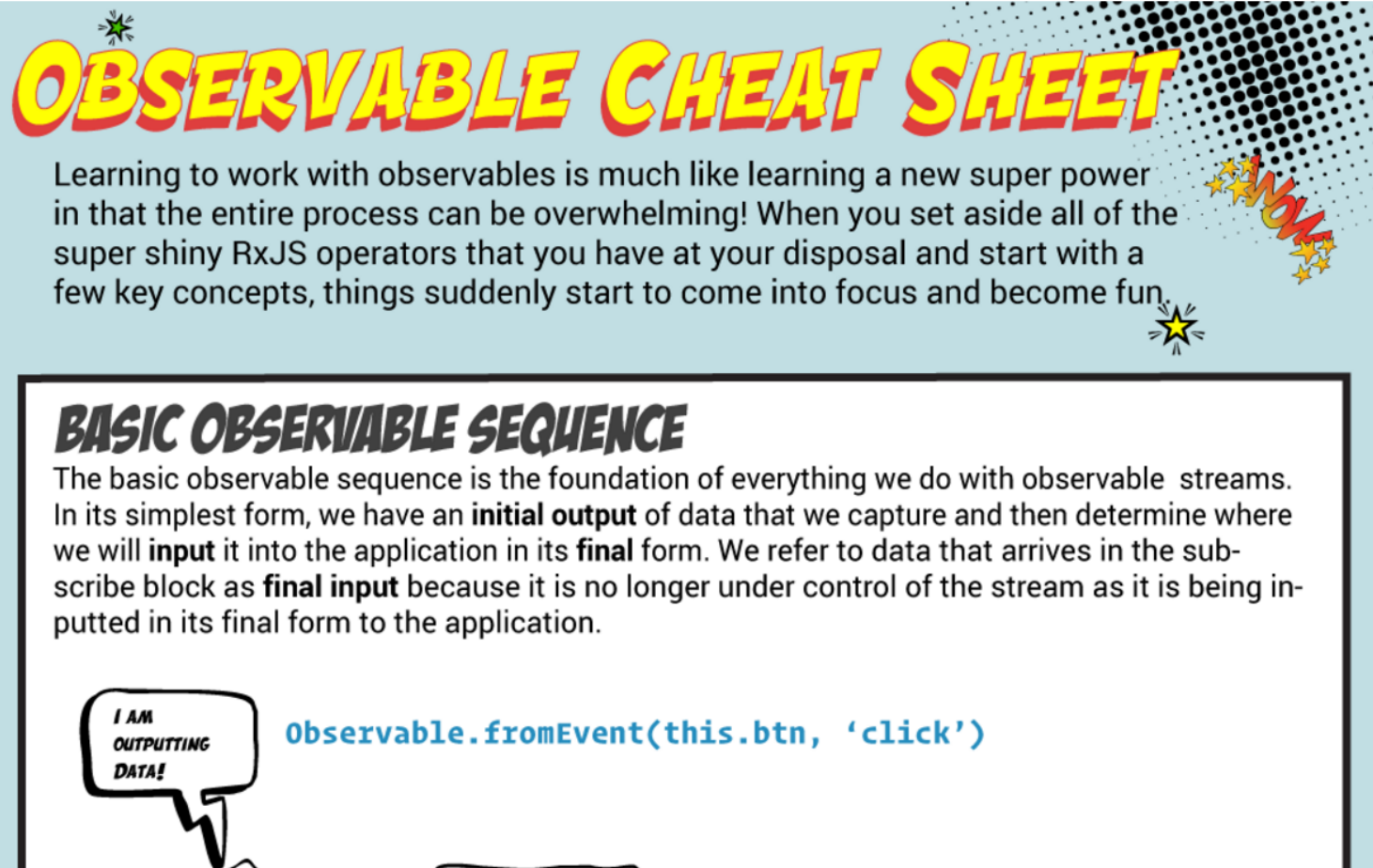


Observable Cheat Sheet

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



OBSERVABLE CHEAT SHEET

Learning to work with observables is much like learning a new super power in that the entire process can be overwhelming! When you set aside all of the super shiny RxJS operators that you have at your disposal and start with a few key concepts, things suddenly start to come into focus and become fun.

BASIC OBSERVABLE SEQUENCE

The basic observable sequence is the foundation of everything we do with observable streams. In its simplest form, we have an **initial output** of data that we capture and then determine where we will **input** it into the application in its **final** form. We refer to data that arrives in the subscribe block as **final input** because it is no longer under control of the stream as it is being inputted in its final form to the application.

I AM OUTPUTTING DATA!

```
Observable.fromEvent(this.btn, 'click')
```

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

Hello RxJS

Gratis online training

The screenshot shows the 'Hello RxJS' micro-course interface. On the left, there's a sidebar with a course box titled 'ULTIMATE ANGULAR Hello RxJS MICRO COURSE', a progress bar at 5% complete, and navigation links for 'Class Curriculum' and 'Your Instructor'. The main area is titled 'Class Curriculum' and features a 'Start next lecture' button followed by 'Presentation: Realtime Observable Streams'. Below this is a list of course items, each with an icon, a title, and a 'Start' button. The items are: 'Presentation: Realtime Observable Streams' (selected), 'Slides: Realtime Observable Streams', 'The Basic Observable Sequence (2:09)', 'Lab: The Basic Observable Sequence', 'Mapping Values (2:22)', 'Lab: Mapping Values', 'Maintaining State (3:25)' (checked), 'Lab: Maintaining State', 'Merging Streams (1:57)', 'Lab: Merging Streams', 'Mapping to Functions (5:18)', and 'Lab: Mapping to Functions'.

Class Curriculum		
Start next lecture > Presentation: Realtime Observable Streams		
Hello RxJS		
<input checked="" type="radio"/>	Presentation: Realtime Observable Streams	Start
<input type="radio"/>	Slides: Realtime Observable Streams	Start
<input type="radio"/>	The Basic Observable Sequence (2:09)	Start
<input type="radio"/>	Lab: The Basic Observable Sequence	Start
<input type="radio"/>	Mapping Values (2:22)	Start
<input type="radio"/>	Lab: Mapping Values	Start
<input checked="" type="radio"/>	Maintaining State (3:25)	
<input type="radio"/>	Lab: Maintaining State	Start
<input type="radio"/>	Merging Streams (1:57)	Start
<input type="radio"/>	Lab: Merging Streams	Start
<input type="radio"/>	Mapping to Functions (5:18)	Start
<input type="radio"/>	Lab: Mapping to Functions	Start

<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/>

The screenshot shows the Learn RxJS website. On the left is a sidebar with a search bar labeled 'Type to search' containing the text 'learn-rxjs'. Below the search bar, the sidebar lists the site's structure: 'LEARN RXJS', 'Introduction' (highlighted in blue), 'Operators', and 'Conditional'. Under 'Operators', a list of operators is shown: 'Combination' (with sub-items 'combineAll', 'combineLatest', 'concat', 'concatAll', 'forkJoin', 'merge', 'mergeAll', 'pairwise', 'race', 'startWith', 'withLatestFrom'), and 'zip'. The main content area on the right has a header 'Learn RxJS' with a sub-header 'Clear examples, explanations, and resources for RxJS.' Below this is the 'Introduction' section, which states that RxJS is a popular library for web development and that the site aims to make learning it easier. It mentions that learning RxJS is 'hard' due to its concepts and API, but the site provides clear examples and references to help. The 'Content' section is partially visible at the bottom.

Type to search

learn-rxjs

LEARN RXJS

[Introduction](#)

Operators

Combination

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

zip

Conditional

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



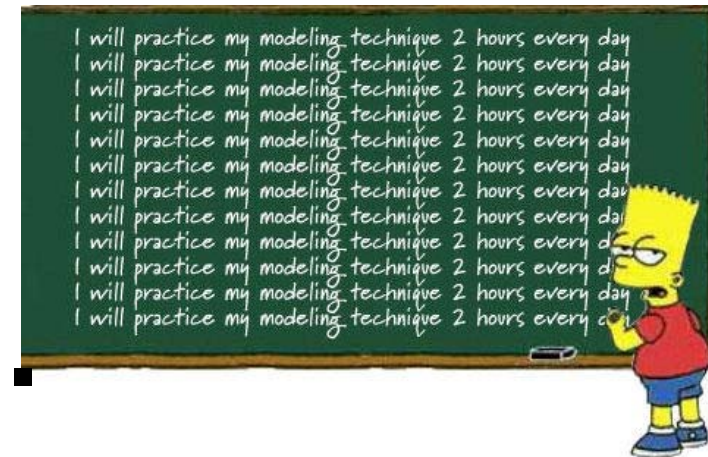
Voorbeeld API's

- <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/MV5BMTY2NzlwMTgwN15BMl5BanBnXkFtZTcwMjlyMzMzMzMQ@@_V1_SX300.jpg"}, {"Title": "Captain  
Nemo", "Year": "1975", "imdbID": "tt0453375", "Type": "movie", "Poster": "https://images-na.ssl-images-  
amazon.com/images/M/MV5BM2JmOTRlMGQtODMxNy00YmRkLWI1OWEtMmQ2YjZlZmQzZGU5XkEyXkFqcGdeQXVyNDUxNjc5NjY@_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/MV5BMzAxMzMyODQtNWY0Yy00N2M3LWE5MDQtZDUzNjc1ZGFmMzA4XkEyXkFqcGdeQXVyMzIxMzc4Mw@@_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/>

Official documentation...

The screenshot shows the RxJS official documentation website. The left sidebar contains a navigation menu with categories like 'AsyncSubject', 'BehaviorSubject', 'Notification', 'Observable', 'ReplaySubject', 'Scheduler', 'AnonymousSubject', 'Subject', 'SubjectSubscriber', 'Subscriber', 'Subscription', 'ObservableInput', 'Observer', 'SubscribableOrPromise', 'TeardownLogic', 'Rx.Scheduler', and 'Rx.Symbol'. Below these are sub-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: 'bindCallback' and 'bindNodeCallback', both taking a function, selector, and scheduler, and returning an Observable. The footer of the page indicates it was 'Generated by ESDoc(0.4.8)'.

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/>

The screenshot shows the Learn RxJS website. On the left is a sidebar with a search bar labeled 'Type to search' containing 'learn-rxjs'. Below the search bar, the sidebar lists 'LEARN RXJS' with links for 'Introduction' (highlighted in blue) and 'Operators'. Under 'Operators', there are sub-sections: 'Combination' (listing combineAll, combineLatest, concat, concatAll, forkJoin, merge, mergeAll, pairwise, race, startWith, withLatestFrom, zip) and 'Conditional'. The main content area has a header with 'EDIT THIS PAGE', 'Star 733', and 'Watch 54' buttons. The title 'Learn RxJS' is prominently displayed. Below it, a subtitle reads 'Clear examples, explanations, and resources for RxJS.' The 'Introduction' section follows, stating that RxJS is a popular library for web development and that the site aims to make learning it easier. A 'But...' section mentions that learning RxJS is 'hard' due to its concepts and API surface, but the site provides a fresh perspective to overcome these challenges. The 'Content' section is partially visible at the bottom.

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](#)

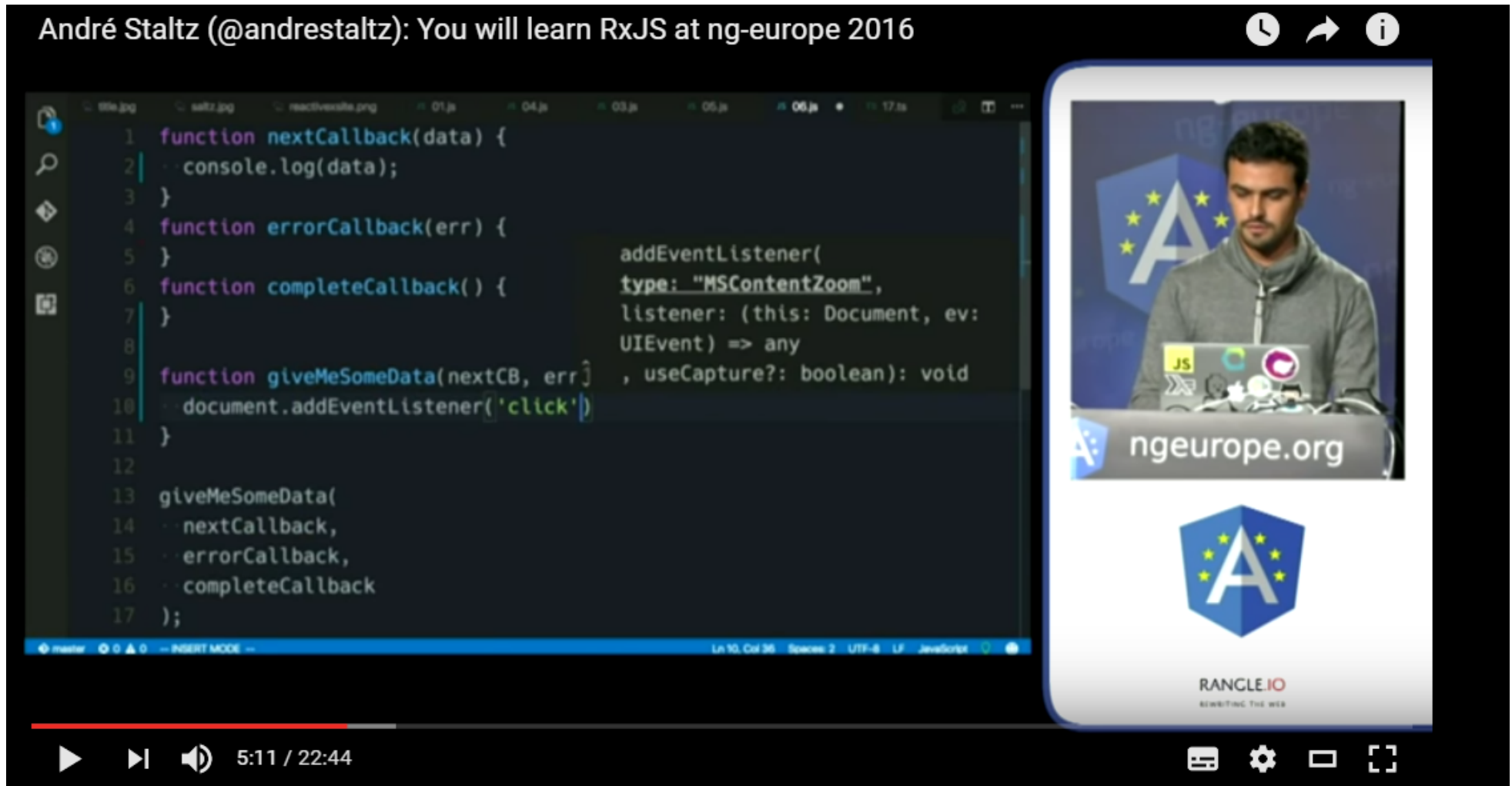
GET UPDATES

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

Creating Observables from scratch

- André Staltz

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  );  
18 }  
19  
20 addEventListener(  
21   type: "MSContentZoom",  
22   listener: (this: Document, ev:  
23     UIEvent) => any  
24   , useCapture?: boolean): void
```

ng-europe.org

RANGLE.IO
REWRITING 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." data-bbox="595 195 735 215">

📄

📄

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 of distributed systems. I found a few detailed books, but they were too expensive and I didn't have time to read them. I decided to create a series of videos that would help you learn the basics of Reactive Programming in a practical way.



THOUGHTTRAM

TRAINING

CODE REVIEW

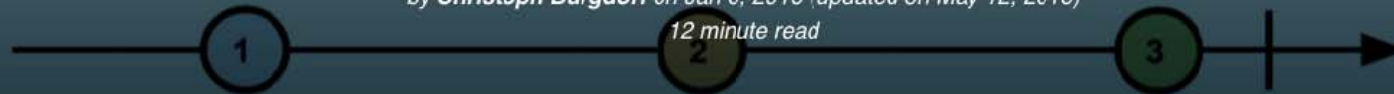
BLOG



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 *lazyness* 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>