



Atos

Global Knowledge.®

# Angular Module 4 - 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

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

Bovendien in Angular 2: 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

Combining

## LANGUAGES

RxJava<sup>®</sup>

RxJS<sup>®</sup>

Rx.NET<sup>®</sup>

RxClojure

## RESOURCES

Tutorials

## COMMUNITY

GitHub<sup>®</sup>

Twitter<sup>®</sup>

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.*

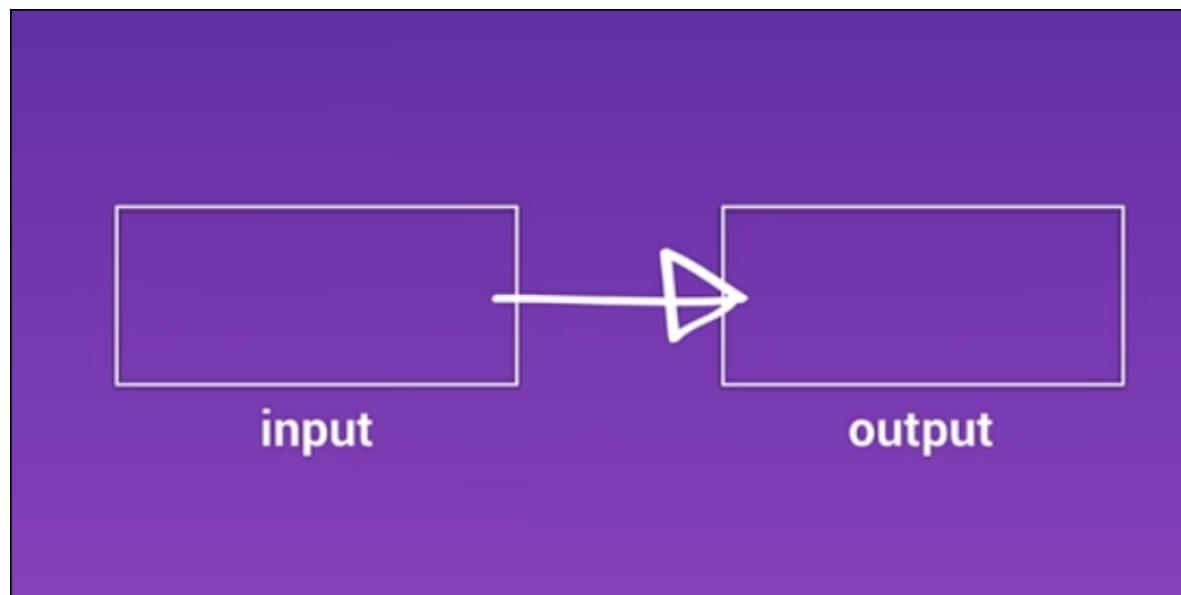
<https://auth0.com/blog/2015/10/15/angular-2-series-part-3-using-http/>

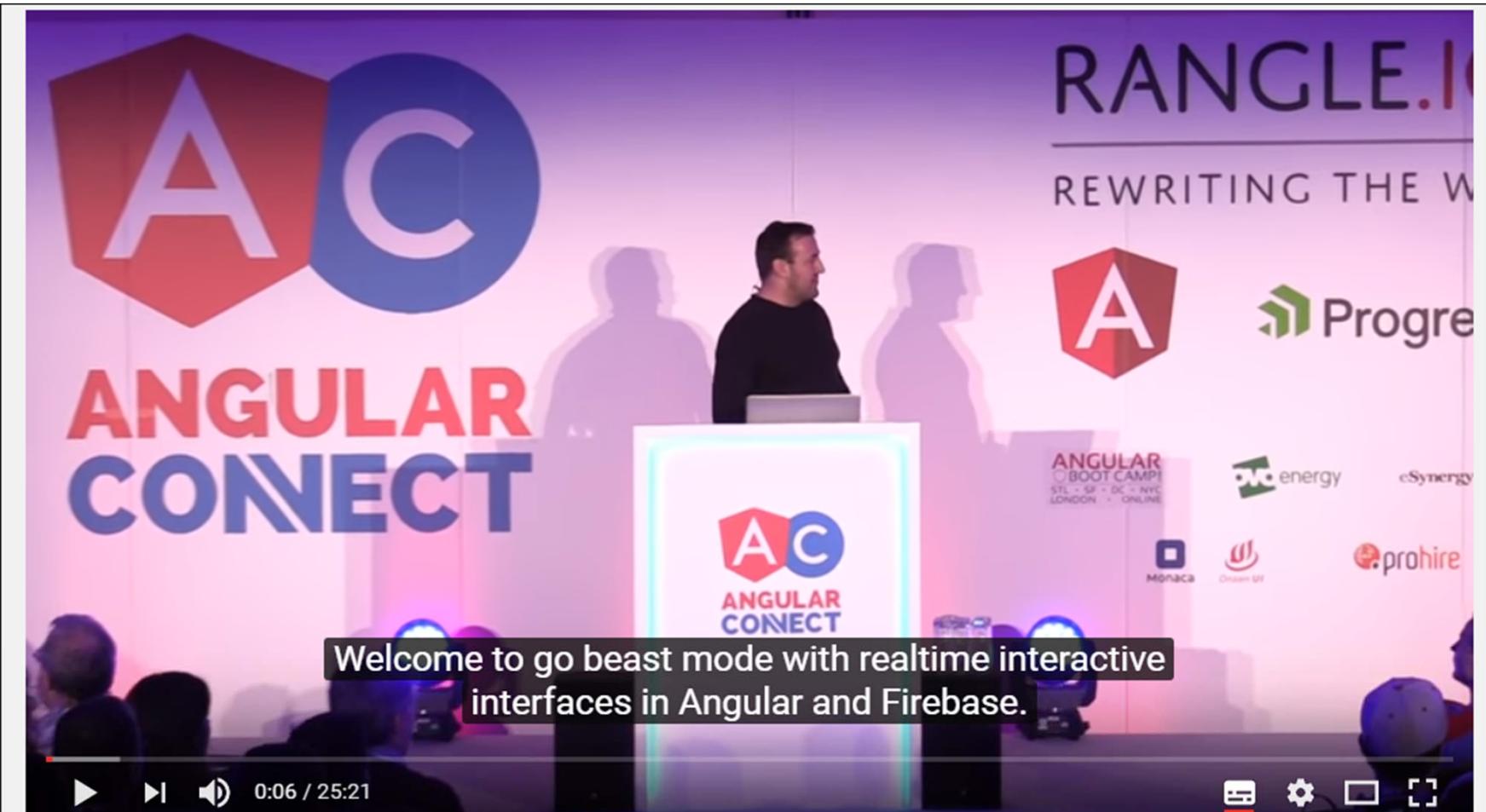
# Observables en RxJs

- “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:

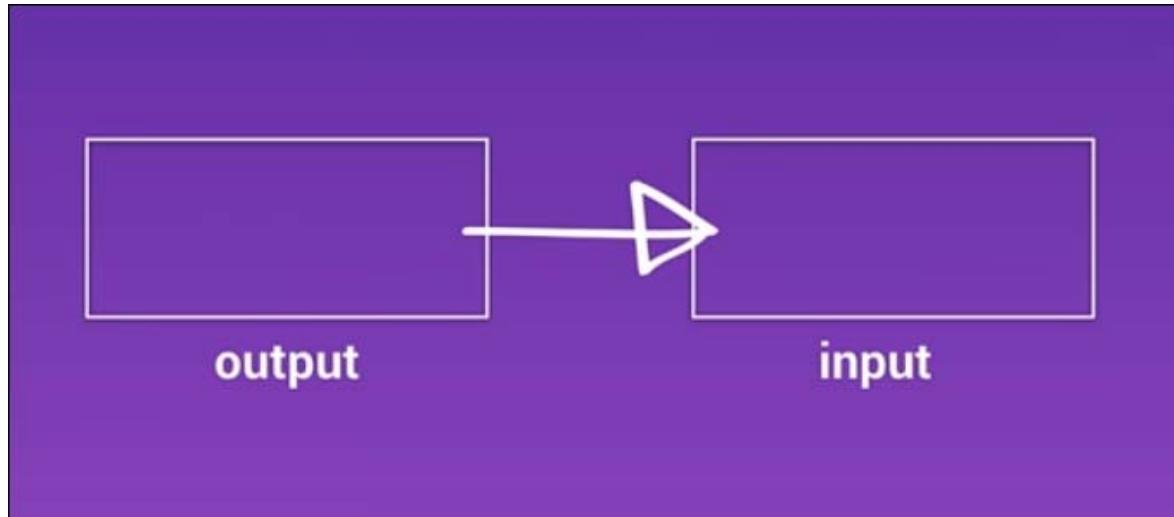




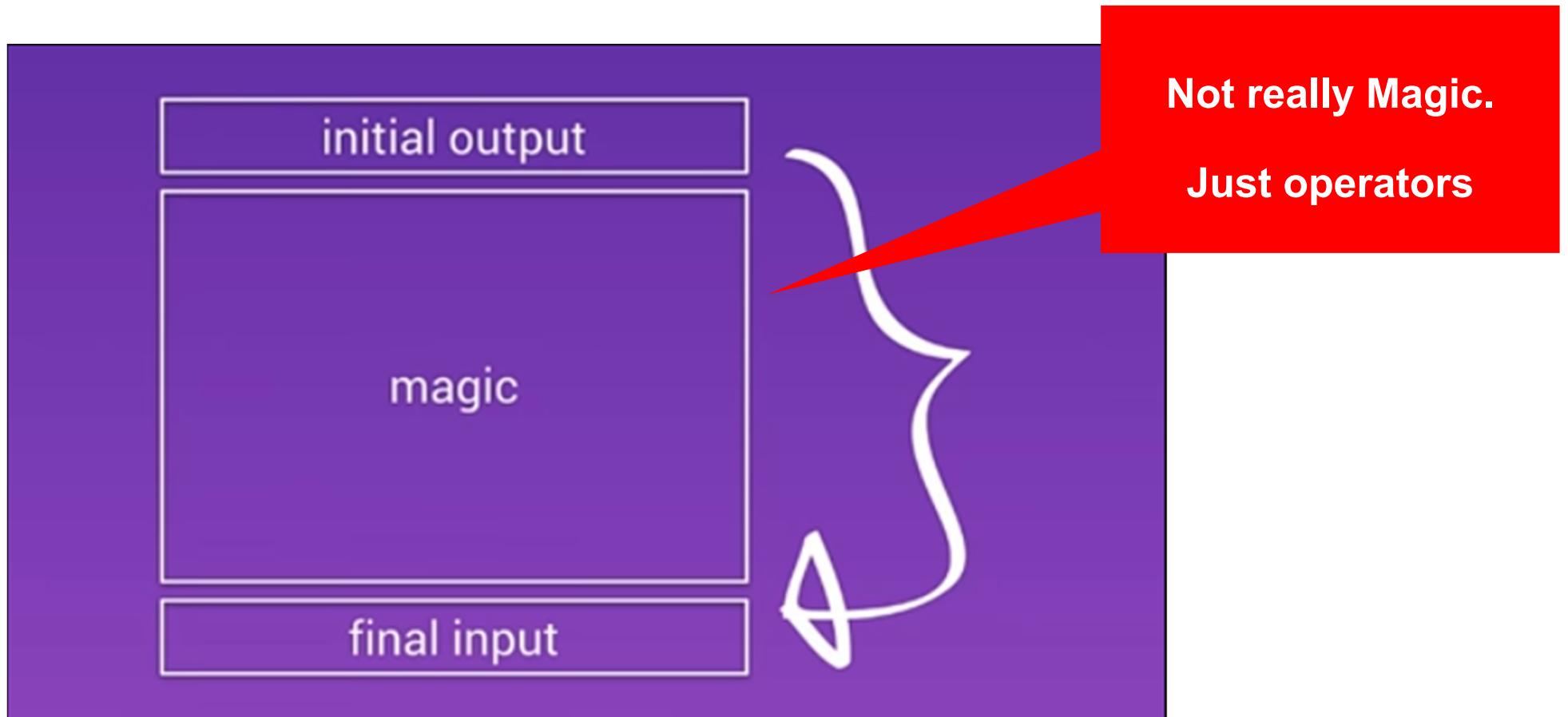
<https://www.youtube.com/watch?v=5CTL7aqSvJU>

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

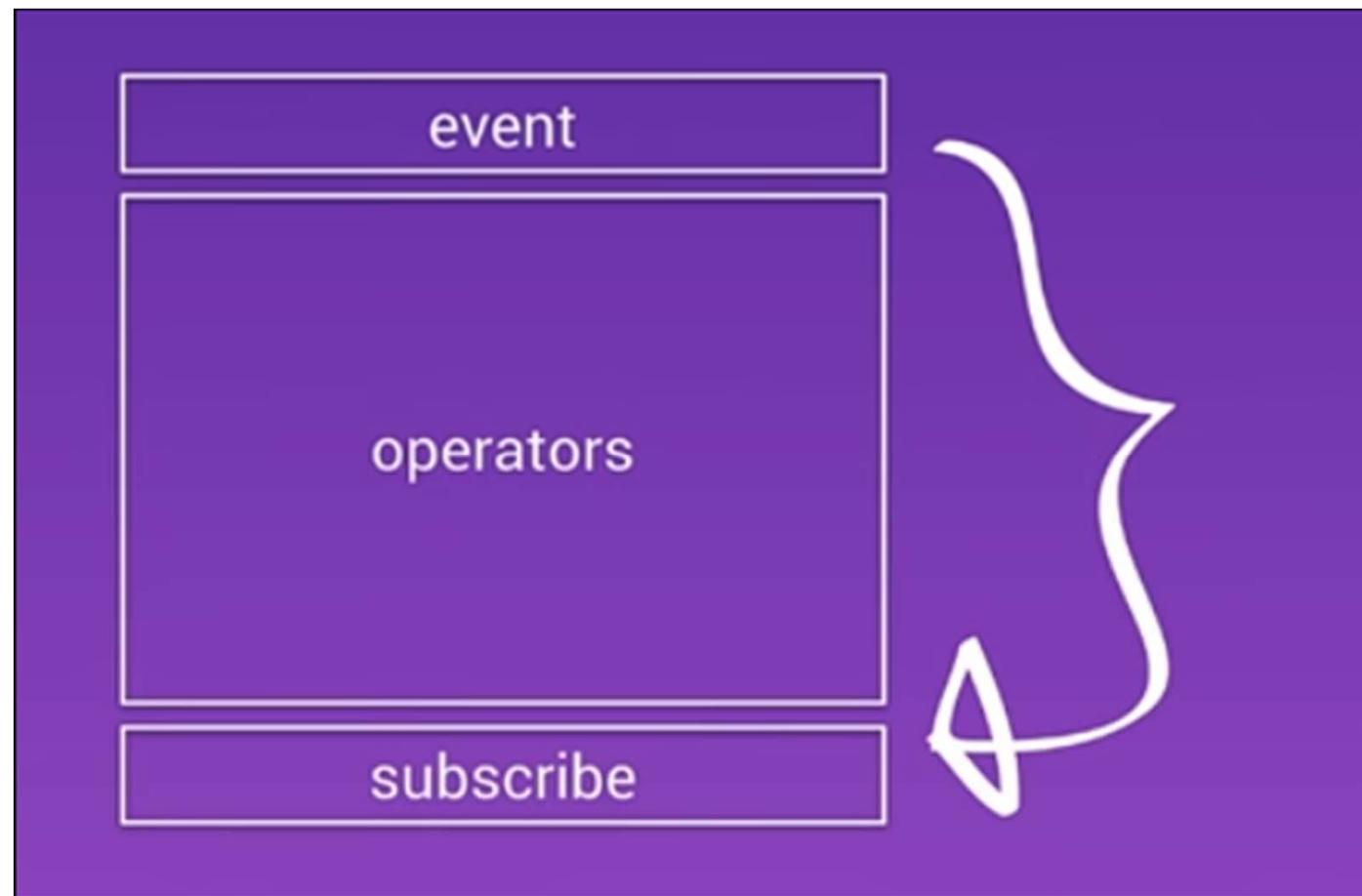
- With Observables -
  - a system, already outputting data,
  - Subscribe to that data
- "trade Output for Input"
- "Push vs. Pull"



# "The observable sandwich"



# Subscribe to events



In code:

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

```

Initial Output

Optioneel:  
operator(s)

Final Input

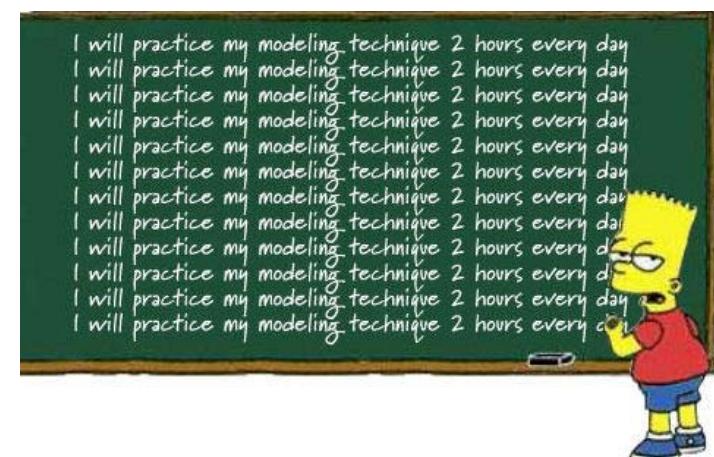
# Ook: importeren HttpClientModule in @NgModule

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

# Oefening

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

# Exercise....

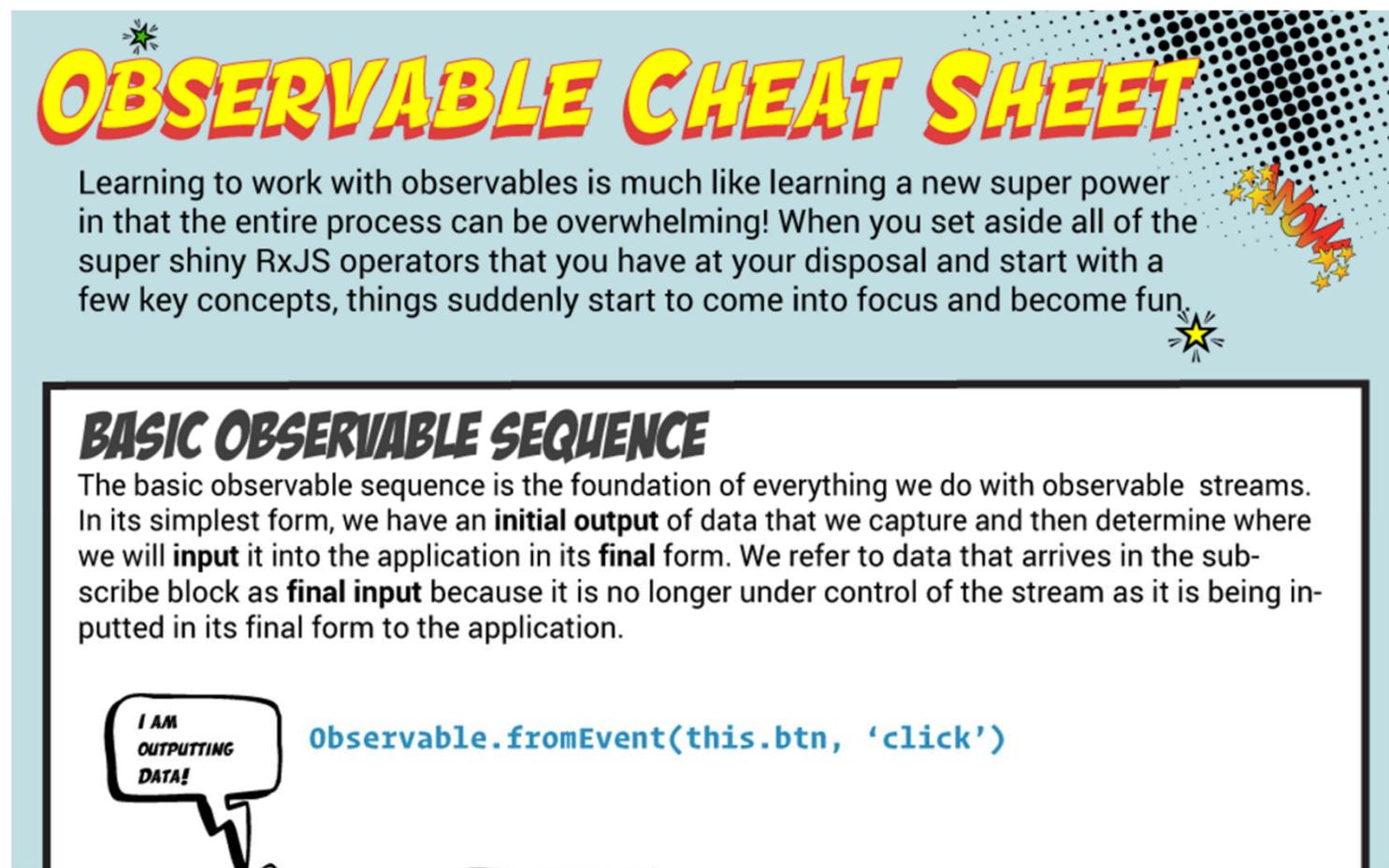


# Observable Cheat Sheet

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

The screenshot shows a web-based learning platform interface for a 'Hello RxJS' micro course. On the left, there's a sidebar with a course thumbnail for 'ULTIMATE ANGULAR' and sections for 'Hello RxJS' (5% complete), 'Class Curriculum', and 'Your Instructor'. The main content area is titled 'Class Curriculum' and shows a list of lessons under the heading 'Hello RxJS'. Each lesson item includes a small icon, the lesson title, and a 'Start' button.

Lesson	Description	Action
1	Presentation: Realtime Observable Streams	Start
2	Slides: Realtime Observable Streams	Start
3	The Basic Observable Sequence (2:09)	Start
4	Lab: The Basic Observable Sequence	Start
5	Mapping Values (2:22)	Start
6	Lab: Mapping Values	Start
7	Maintaining State (3:25)	Start
8	Lab: Maintaining State	Start
9	Merging Streams (1:57)	Start
10	Lab: Merging Streams	Start
11	Mapping to Functions (5:18)	Start
12	Lab: Mapping to Functions	Start

<http://courses.ultimateangular.com/>

# Pipeable operators

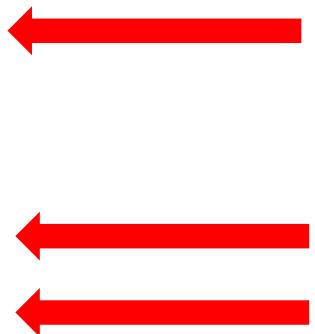
- In RxJS 6.x en hoger: alle operators komen binnen de .pipe() functie
- De parameters van de pipe-functie zijn de operatoren!
- Ze worden met komma's van elkaar gescheiden

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

# Subscribe - only once per block!

- Three parameters:
  - success()
  - error() – Optioneel!
  - complete() – Optioneel!

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



# Ben Lesh on observables in RxJS 6.0

The two you care about

- 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.

@benlesh



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 the homepage of the Learn RxJS website. The left sidebar contains a search bar, a navigation menu with 'learn-rxjs' and 'LEARN RXJS' sections, and a detailed list of RxJS operators under 'Operators'. The main content area features a large title 'Learn RxJS' and a subtitle 'Clear examples, explanations, and resources for RxJS.' Below this is a section titled 'Introduction' with a paragraph about RxJS's popularity and utility. A 'But...' section follows, explaining the challenges of learning RxJS. At the bottom is a 'Content' section.

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
- examples\210-services-live

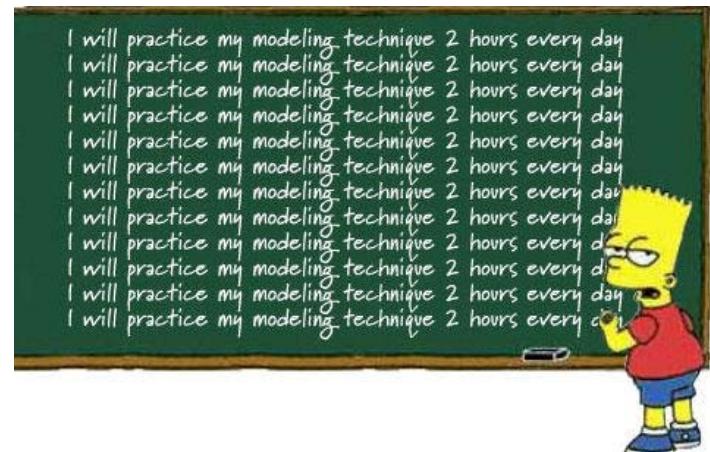


# Voorbeeld API's

- <https://pokeapi.co/> - Pokemon API
- <http://openweathermap.org/API> (weerbericht)
- <http://randomuser.me/> (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

# 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
  - ...
- Exercise : 5<sup>e</sup>)



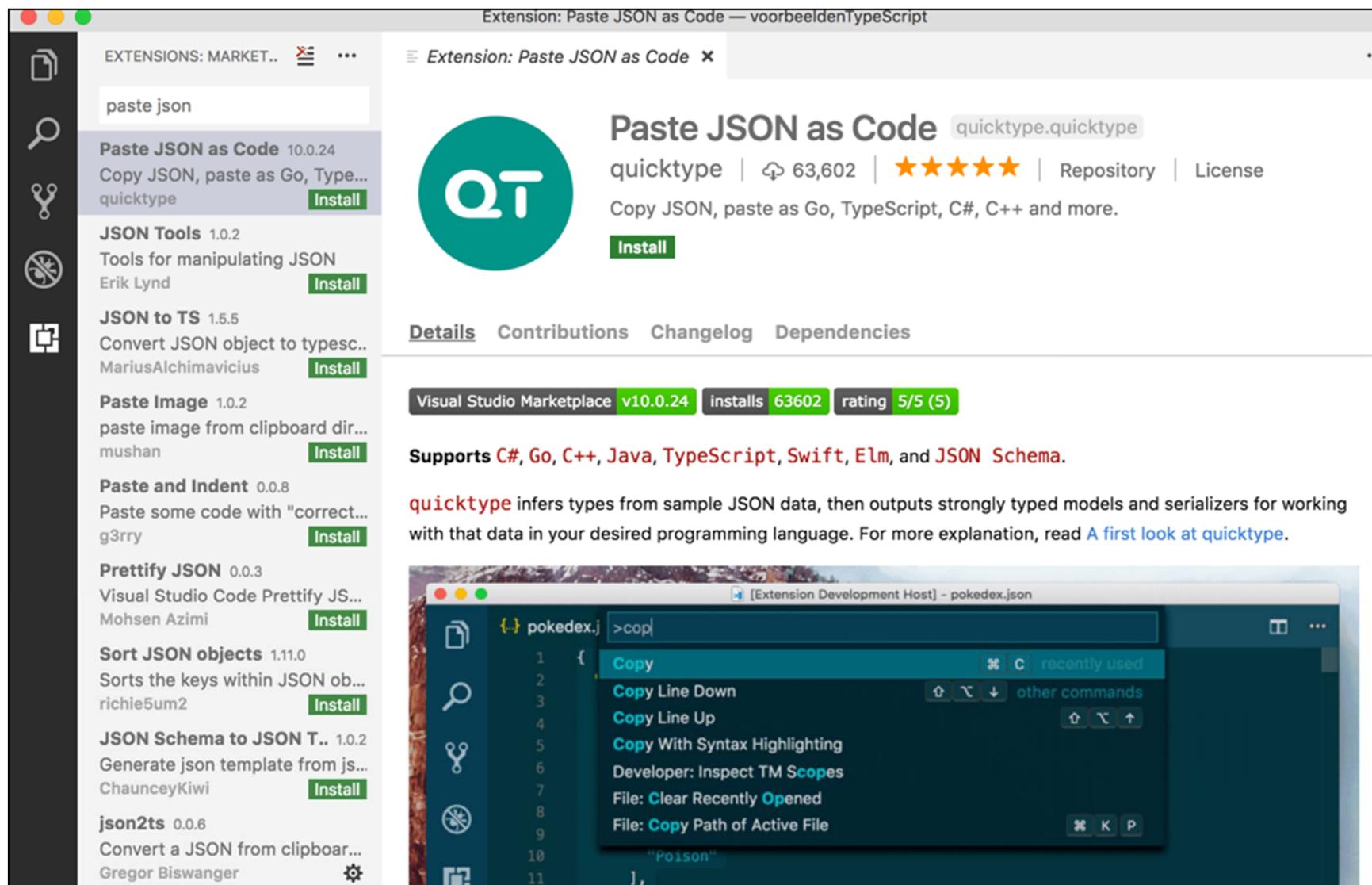
# Online JSON to TypeScript converter

The screenshot shows the json2ts website interface. At the top, the logo "json2ts" is displayed in large white letters on a blue header bar. Below the logo, the text "generate TypeScript interfaces from JSON" is visible. On the right side of the header bar are links for "email", "feedback", and "help". The main content area contains a JSON array representing movie data. Below the JSON, a green button labeled "generate TypeScript" is shown. To the right of the button, the generated TypeScript code is displayed:

```
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!



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

# Data Mocken - Mockaroo

The screenshot shows the Mockaroo web application interface. At the top, there's a green header bar with the Mockaroo logo, a search icon, and links for PRICING and SIGN IN. Below the header, a central message box says "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." The main content area contains a table for defining data fields:

Field Name	Type	Options
id	Row Number	blank: 0 % <input type="button" value="fx"/> <input type="button" value="x"/>
first_name	First Name	blank: 0 % <input type="button" value="fx"/> <input type="button" value="x"/>
last_name	Last Name	blank: 0 % <input type="button" value="fx"/> <input type="button" value="x"/>
email	Email Address	blank: 0 % <input type="button" value="fx"/> <input type="button" value="x"/>
gender	Gender	blank: 0 % <input type="button" value="fx"/> <input type="button" value="x"/>
ip_address	IP Address v4	blank: 0 % <input type="button" value="fx"/> <input type="button" value="x"/>

Below the table is a button labeled "Add another field". Further down, there are filters for "# Rows" (set to 1000), "Format" (set to CSV), "Line Ending" (set to Unix (LF)), and "Include" (checkboxes for "header" checked and "BOM" unchecked). At the bottom, there are three buttons: "Download Data" (green), "Preview" (gray), and "More" (gray dropdown). A link "Want to save this for later? Sign up for free." is also present.

<http://mockaroo.com/>

# Official documentation...

The screenshot shows the RxJS official documentation for the Observable class. The top navigation bar includes links for Home, Manual, Reference, Source, Test, a dark/light theme switch, and a search icon. The left sidebar contains a tree view of RxJS classes and interfaces, such as AsyncSubject, BehaviorSubject, Notification, Observable, ReplaySubject, Scheduler, AnonymousSubject, Subject, SubjectSubscriber, Subscriber, Subscription, ObservableInput, Observer, SubscribableOrPromise, TeardownLogic, Rx.Scheduler, and Rx.Symbol. The main content area starts with a code snippet showing the Observable class definition from es6/Observable.js:

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

## 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 is the most basic building block of RxJS.

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

### Static Method Summary

Static Public Methods	
public static	<code>bindCallback(func: function, selector: function, scheduler: Scheduler): function(...params: *): Observable</code> Converts a callback API to a function that returns an Observable.
public static	<code>bindNodeCallback(func: function, selector: function, scheduler: Scheduler): function(...params: *): Observable</code> Converts a node-style callback API to a function that returns an 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 homepage of the Learn RxJS website. The left sidebar contains a search bar, a navigation menu with 'learn-rxjs' and 'LEARN RXJS' sections, and a detailed list of RxJS operators under 'Operators'. The main content area features a large title 'Learn RxJS' and a subtitle 'Clear examples, explanations, and resources for RxJS.' Below this is a section titled 'Introduction' with a paragraph about RxJS's popularity and utility. A 'But...' section follows, explaining the challenges of learning RxJS. At the bottom is a 'Content' section.

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

# Article - 6 Operators you must know

The screenshot shows a Medium article page. At the top right are 'Sign in / Sign up' and social sharing icons for LinkedIn and Twitter. Below the header is the author's profile picture, name 'Netanel Basal', a 'Follow' button, and the date 'Jan 24 · 3 min read'. The main title 'RxJS—Six Operators That you Must Know' is displayed prominently. A large black box contains the code for a 'TakeSubscriber' class. At the bottom, there's a call-to-action for signing up to receive updates.

```
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) {  
            destination.next(value);  
        }  
    }  
}  
export { TakeSubscriber };
```

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

The screenshot shows a video player interface. On the left, there is a code editor window displaying a JavaScript file with the following code:

```
function nextCallback(data) {
  console.log(data);
}

function errorCallback(err) {
}

function completeCallback() {
}

function giveMeSomeData(nextCB, errCB) {
  document.addEventListener('click', () => {
    nextCB();
    errCB();
  });
}

giveMeSomeData(
  nextCallback,
  errorCallback,
  completeCallback
);
```

On the right, there is a video feed of André Staltz speaking at a conference. He is wearing a grey hoodie and is looking down at a laptop. The background features a blue banner with the 'ng-europe' logo. Below the video, there is a URL: [ngeurope.org](https://ngeurope.org). The video player has standard controls at the bottom: play/pause, volume, and progress bar.

<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.github.com/staltz/868e7e9bc2a7b8c1f754.js"> 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

RxMarbles Interactive diagrams of Rx Observables

Fork me on GitHub

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](#)

merge

```
graph LR; S1[20] --- S2[40] --- S3[60] --- S4[80] --- S5[100]; S6[1] --- S7[1]; S1 --- S6; S2 --- S7; S3 --- S8[1]; S4 --- S9[80]; S5 --- S10[100]; S6 --- S11[1];
```

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

<http://rxmarbles.com/>

# Dan Wahlin on Modules and Observables

The screenshot shows a video player interface. On the left is a code editor window titled "data.service.ts" displaying Angular code for a service. On the right is a video frame showing a man speaking at a podium.

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-inlinelcount');
35        let customers = res.json();
36        this.calculateCustomersOrderTotal(customers);
37        return {
38          results: customers,
39          totalRecords
40        };
41      })
42      .catch(this.handleError);
43  }
44
45  calculateCustomersOrderTotal(customers: ICustomer[]): void {
46    // ...
47  }
48
49  handleError(error: any): Observable<never> {
50    // ...
51  }
52
53  private calculateCustomersOrderTotal(customers: ICustomer[]): void {
54    // ...
55  }
56
57  private handleError(error: any): Observable<never> {
58    // ...
59  }
60
61  private calculateCustomersOrderTotal(customers: ICustomer[]): void {
62    // ...
63  }
64
65  private handleError(error: any): Observable<never> {
66    // ...
67  }
68
69  private calculateCustomersOrderTotal(customers: ICustomer[]): void {
70    // ...
71  }
72
73  private handleError(error: any): Observable<never> {
74    // ...
75  }
76
77  private calculateCustomersOrderTotal(customers: ICustomer[]): void {
78    // ...
79  }
80
81  private handleError(error: any): Observable<never> {
82    // ...
83  }
84
85  private calculateCustomersOrderTotal(customers: ICustomer[]): void {
86    // ...
87  }
88
89  private handleError(error: any): Observable<never> {
90    // ...
91  }
92
93  private calculateCustomersOrderTotal(customers: ICustomer[]): void {
94    // ...
95  }
96
97  private handleError(error: any): Observable<never> {
98    // ...
99  }
100
101 private calculateCustomersOrderTotal(customers: ICustomer[]): void {
102   // ...
103 }
104
105 private handleError(error: any): Observable<never> {
106   // ...
107 }
```

▶ ⏸ 🔍 52:13 / 1:24:02

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



THOUGHTRAM

Time

TRAINING

CODE REVIEW

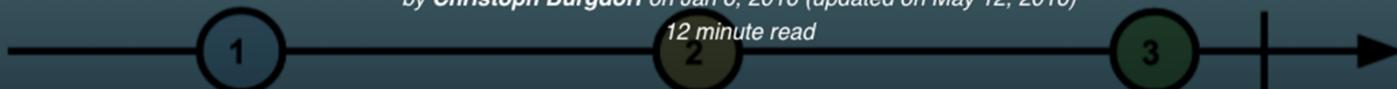
BLOG



# 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 *laziness* 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>

# Een collectie observables ophalen

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

The screenshot shows a blog post on the Angular In Depth website. The header features the 'Angular In Depth' logo with 'by AG-Grid'. Below the header is a navigation bar with links for HOME, ANGULAR, RXJS, NGRX, ABOUT, SUPPORT US, and a mention of AG-GRID. A 'Follow' button is also present. The main content area displays the title 'Practical RxJS In The Wild' followed by a lion emoji. Below the title is the subtitle 'Requests with concatMap() vs mergeMap() vs forkJoin()' followed by a boxing glove emoji. The author's profile picture, name 'Tomas Trajan', and a 'Follow' button are shown. The post was published on 'Dec 19, 2017 · 7 min read'.

Practical RxJS In The Wild —  
Requests with concatMap() vs  
mergeMap() vs forkJoin()

Tomas Trajan [Follow](#)  
Dec 19, 2017 · 7 min read