



IK WIL

Angular – Module Observables

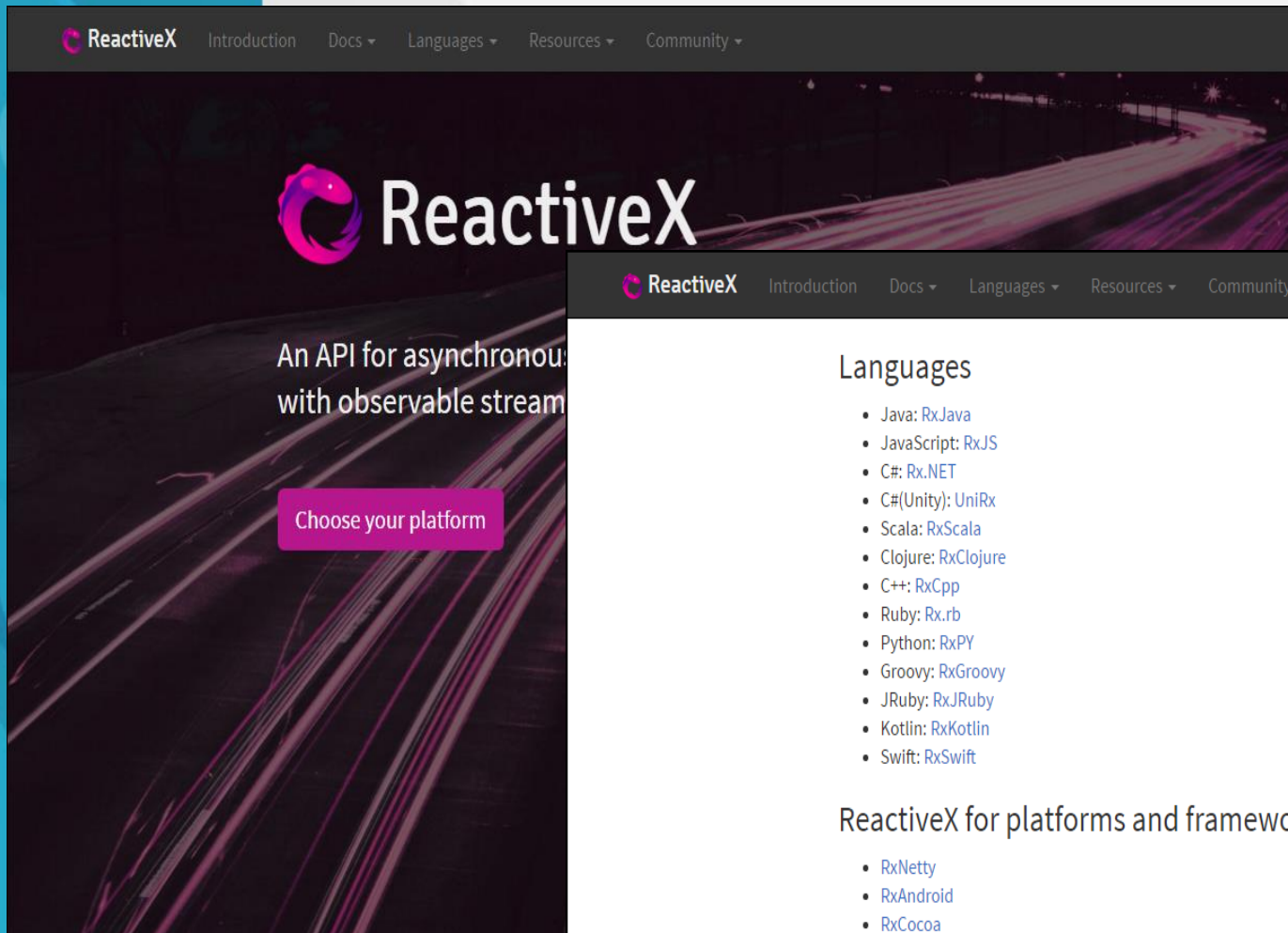
Async services met RxJS/Observables

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`

A screenshot of the ReactiveX website. The top navigation bar is dark grey with the ReactiveX logo and links for Introduction, Docs, Languages, Resources, and Community. The main hero section features a dark background with light streaks, the ReactiveX logo, and the text 'An API for asynchronous with observable stream'. A pink button labeled 'Choose your platform' is positioned below the text. A right-hand sidebar contains a 'Languages' section with a list of supported languages and their respective libraries, and a 'ReactiveX for platforms and frameworks' section with a list of supported platforms and frameworks. The bottom of the page has a dark grey footer with four columns: Documentation, Languages, Resources, and Community, each with a list of links.

ReactiveX Introduction Docs Languages Resources Community

ReactiveX

An API for asynchronous with observable stream

Choose your platform

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](#)

<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.

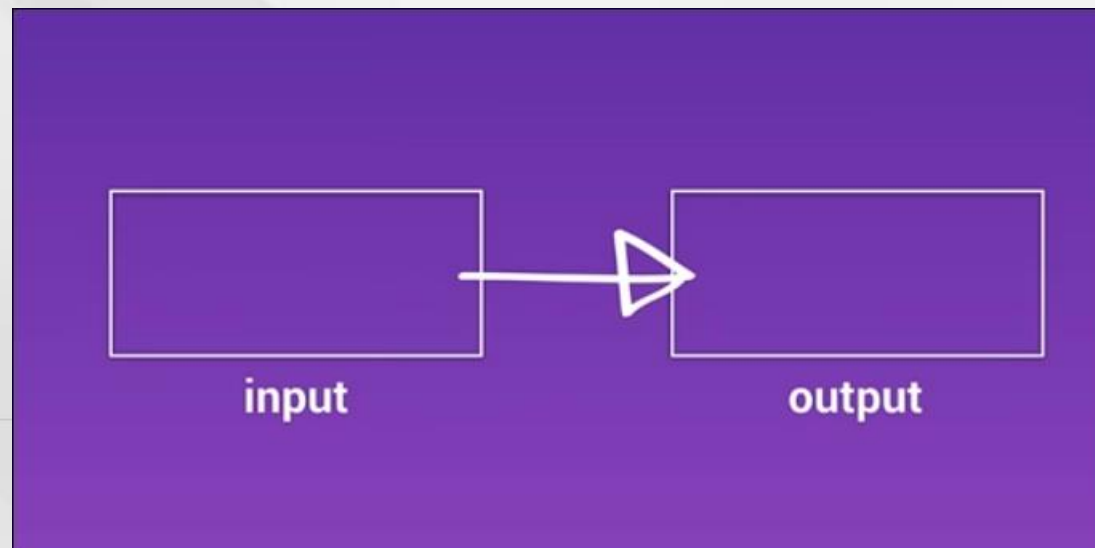
<https://auth0.com/blog/2015/10/15/angular-2-series-part-3-using-http/>

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

How do observables work

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





ANGULAR CONNECT

WELCOME TO GO BEAST MODE WITH REALTIME INTERACTIVE INTERFACES IN ANGULAR AND FIREBASE.

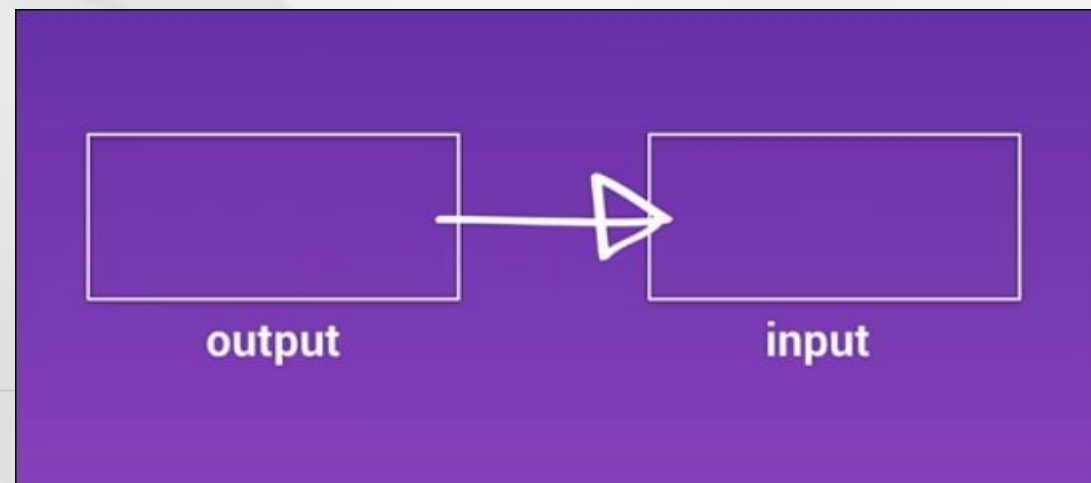
Go beast mode with realtime reactive interfaces in Angular 2 & Firebase | Lukas Ruebbelke

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

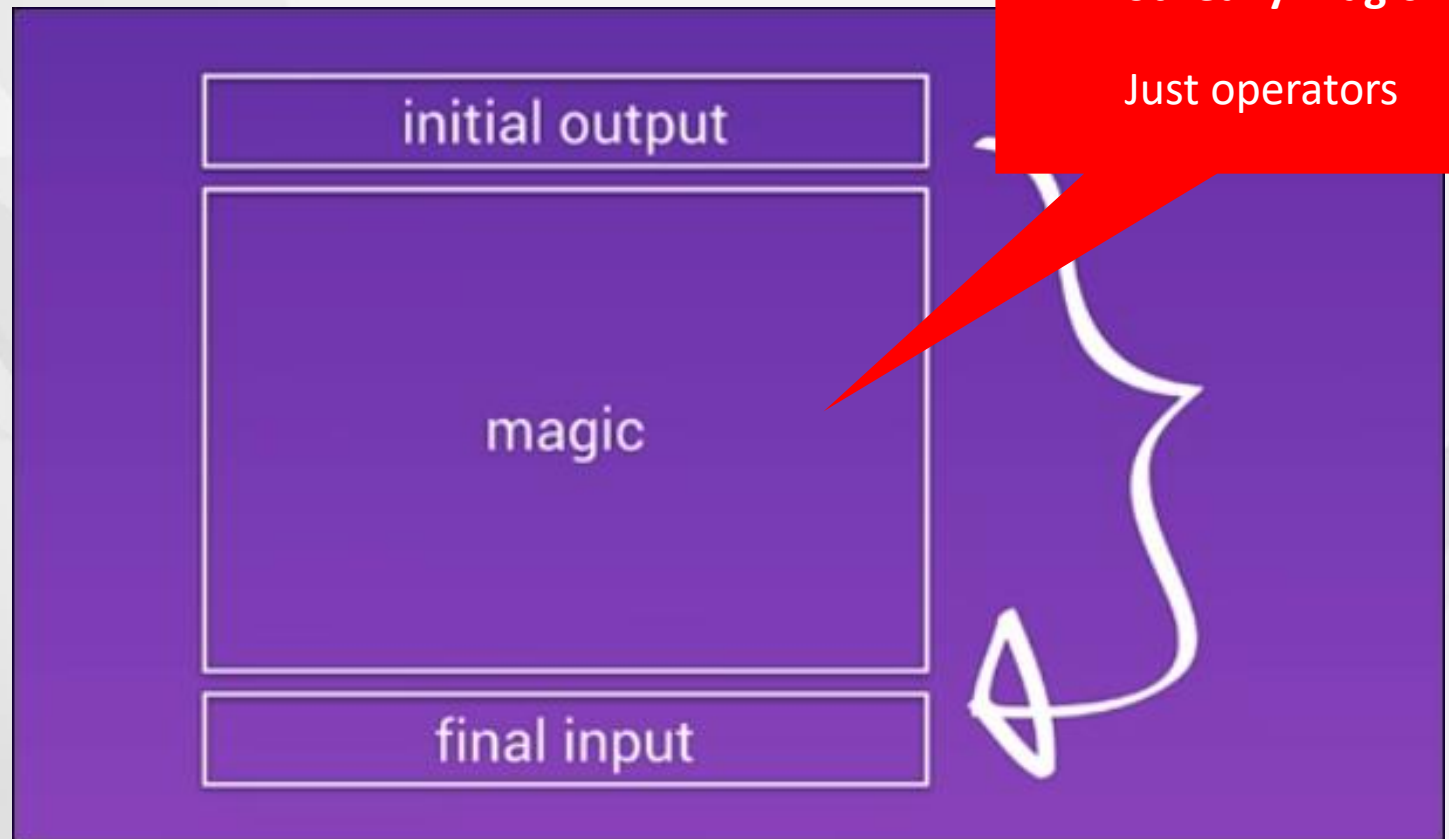
<https://youtu.be/5CTL7aqSvJU?t=4m31s>

→ With Observables -

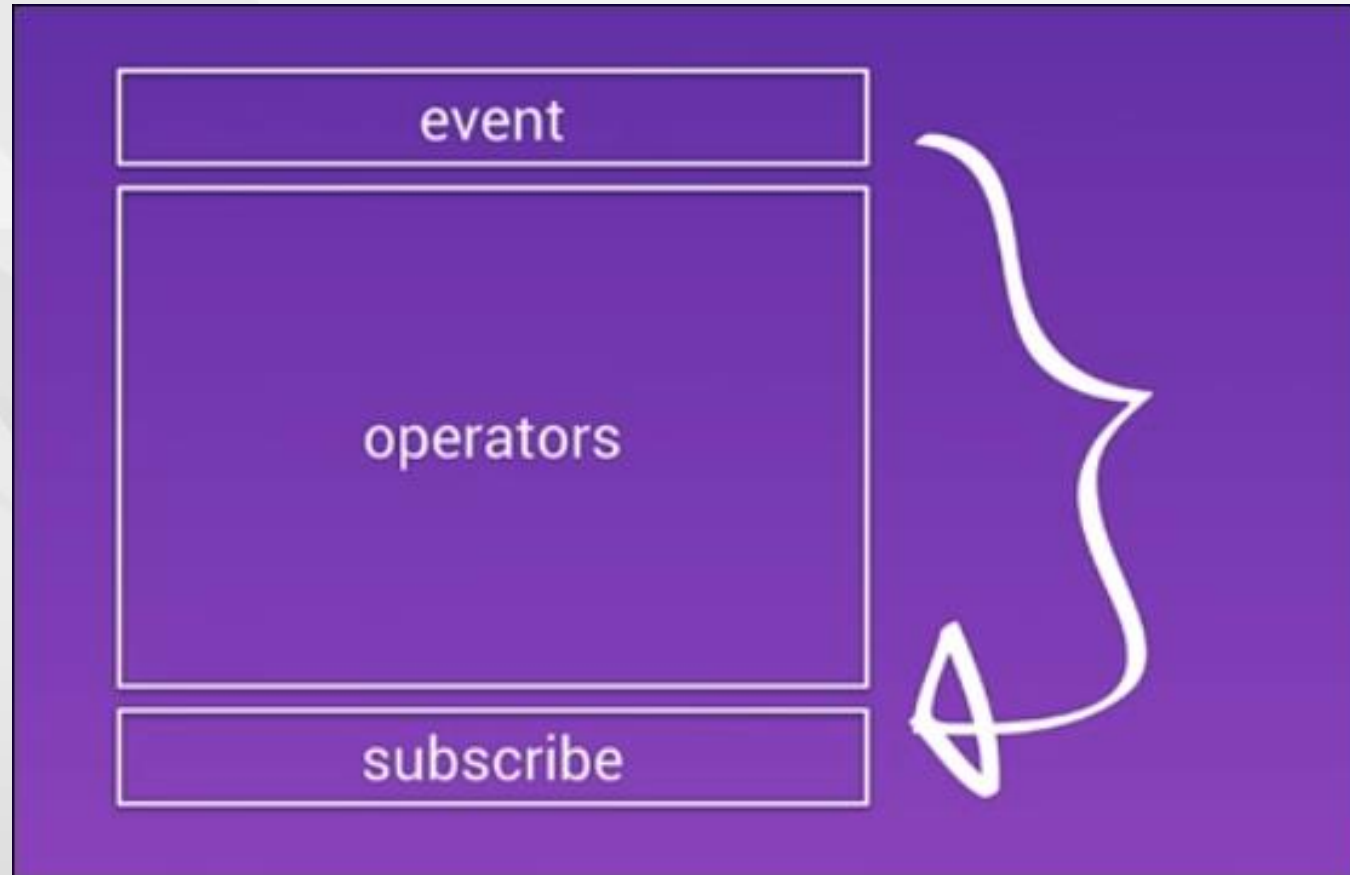
- a system, already outputting data,
- Subscribe to that data
- "trade Output for Input"
- "Push vs. Pull" (bijv. Observable versus Functie)



"The observable sandwich"



Subscribe to events



Initial Output

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

Operator(s)

Final Input

Ook: HttpClientModule in je @NgModule importeren

→ *// 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+: HttpClientModule

- In je @NgModule: imports : [HttpClientModule]
- Niet meer .map(res => res.json()).
 - Json is de standaard!
- Nieuwe optie: [Interceptors](#), voor nu nog te advanced
- HttpClientModule is/wordt de standaard in Angular 5
 - HttpClientModule wordt in toekomstige versies verwijderd

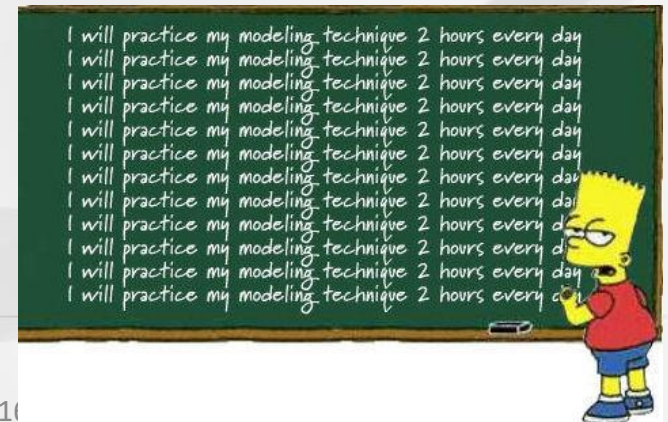
Met HttpClientModule – geen mapping .json()

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

Exercise

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

Exercise....

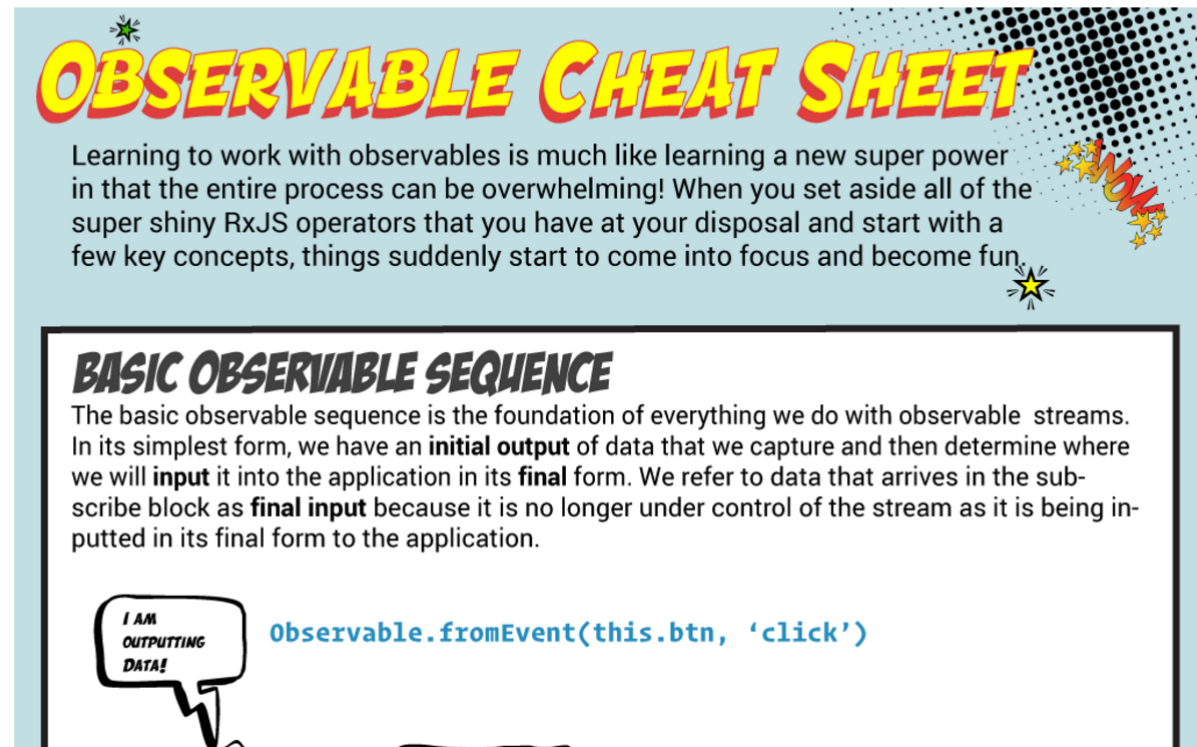


Observable Cheat Sheet

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



The infographic snippet features a light blue background with a black and white halftone pattern in the top right corner. The title 'OBSERVABLE CHEAT SHEET' is written in large, bold, yellow letters with a red outline and a small starburst above the 'O'. Below the title, a paragraph explains that learning to work with observables is like learning a new super power, mentioning RxJS operators. To the right of the text is a 'POW' comic-style sound effect with stars. Below this is a section titled 'BASIC OBSERVABLE SEQUENCE' in bold black letters. The text describes the flow of data from initial output to final input. At the bottom, there is a speech bubble saying 'I AM OUTPUTTING DATA!' and a code snippet: `Observable.fromEvent(this.btn, 'click')`.

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.

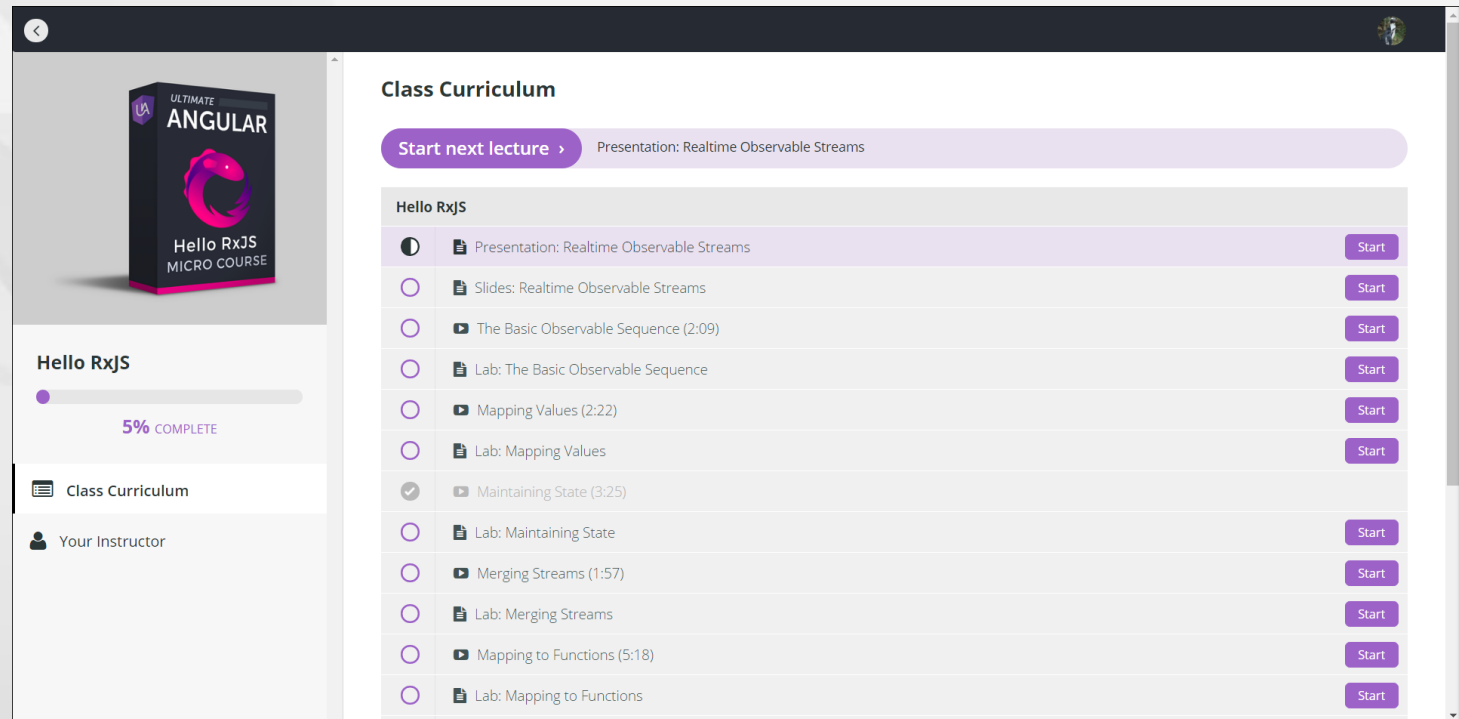
I AM OUTPUTTING DATA!

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

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

Hello RxJS

Gratis online training



The screenshot displays the 'Hello RxJS' course interface. On the left sidebar, there is a course box titled 'Hello RxJS MICRO COURSE' with a progress bar indicating '5% COMPLETE'. Below this, the 'Class Curriculum' section is visible, showing 'Your Instructor'. The main content area, titled 'Class Curriculum', features a 'Start next lecture >' button and a list of course items. The first item, 'Presentation: Realtime Observable Streams', is highlighted in purple. The list includes various presentations, slides, and labs, each with a 'Start' button. The 'Maintaining State (3:25)' item is marked as completed with a checkmark icon.

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/>

Subscribe - only once per block!

- Part of RxJs
- Three parameters:
 - success()
 - error()
 - complete()

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

Observables in een Angular 2-applicatie

→ Importeer Rx in de applicatie

→ Geheel, of alleen de benodigde onderdelen (aanbevolen)

```
import 'rxjs';
```

```
import {Observable} from "rxjs";
```

```
import 'rxjs/add/operator/map';
```

→ Chaining. Resultaat van de een functie dient als invoer voor de volgende functie.

RxJS-operators in de service

```
import {Injectable} from '@angular/core';  
import {Http, Response} from "@angular/http";  
import {Observable} from "rxjs";  
import 'rxjs/add/operator/map';
```

Import operator

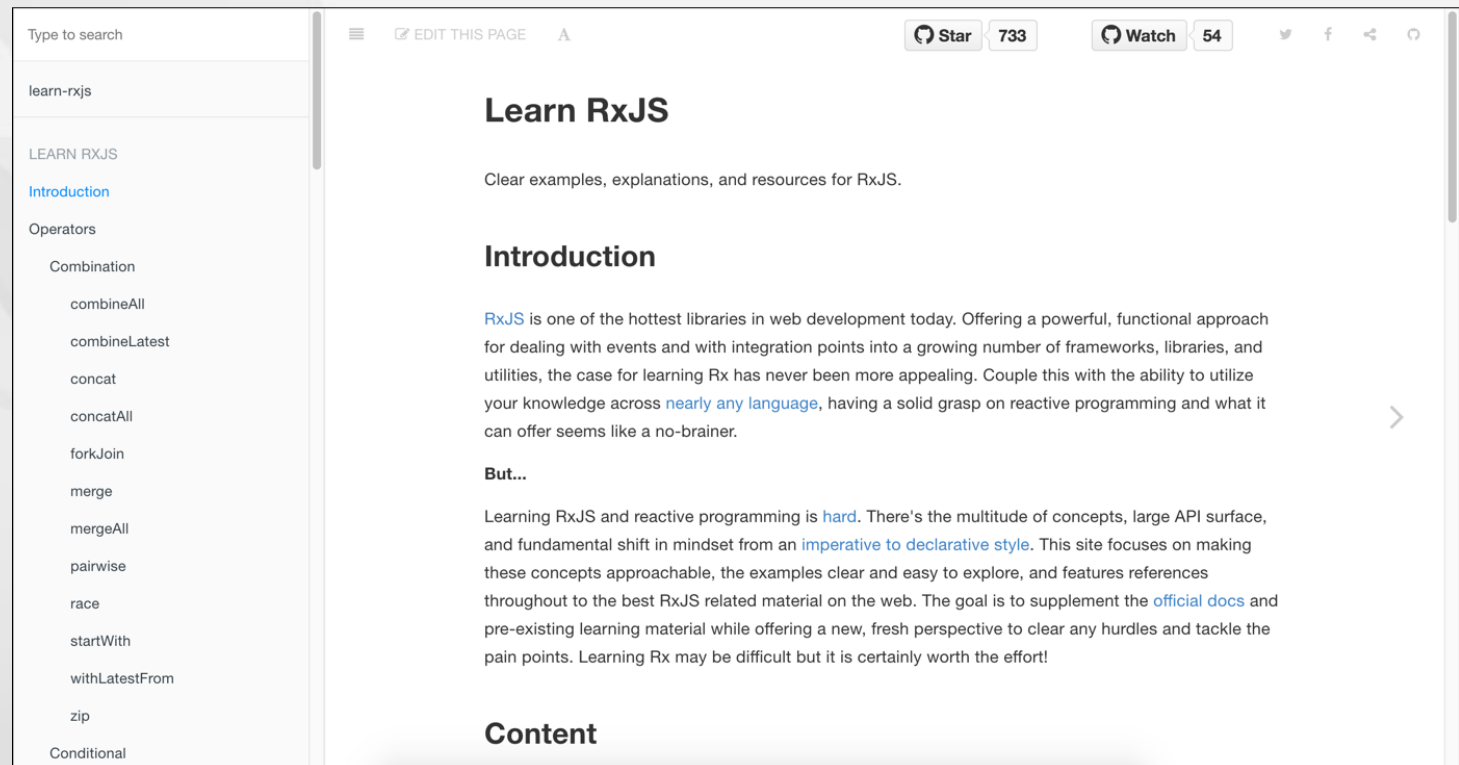
```
@Injectable()  
export class CityService {  
  
    constructor(private http: Http) {  
  
    }  
  
    // retourner alle cities  
    getCities(): Observable<City[]> {  
        return this.http.get('assets/data/cities.json')  
            .map(cities => cities.json());  
    }  
}
```

Transform stream in de service

```
getCities() {  
  if (!this.cities) {  
    this.cityService.getCities()  
      .map(res => res.json())  
      .do(res => console.log(res)) ← extra stap zonder transform  
      .delay(3000)  
      .subscribe(cityData => {  
        this.cities = cityData;  
      },  
        err => console.log(err),  
        () => console.log('Getting cities complete...'))  
  }  
}
```

RxJs-functies

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



The screenshot shows the Learn RxJS website. On the left is a navigation sidebar with a search bar and a list of topics: learn-rxjs, LEARN RXJS, Introduction (highlighted), Operators, Combination (with sub-items: combineAll, combineLatest, concat, concatAll, forkJoin, merge, mergeAll, pairwise, race, startWith, withLatestFrom, zip), and Conditional. The main content area on the right has a title 'Learn RxJS', a subtitle 'Clear examples, explanations, and resources for RxJS.', and a section 'Introduction' which states that RxJS is a popular library for reactive programming. Below this is a 'But...' section discussing the difficulty of learning RxJS and the goal of the site. At the bottom of the main area 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

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

Automatisch `.subscribe()` en `.unsubscribe()`

Async Pipe

- Bij `.subscribe()`, eigenlijk ook `.unsubscribe()` aanroepen.
- Bij HTTP-requests niet beslist nodig.
- Bij andere subscriptions wel, in verband met memory leaks.
- Automatisch `.subscribe()` en `.unsubscribe()`:

→ **Gebruik `async pipe` van Angular**

→ In de component:

```
Cities$: Observable<City[]>; // Nu: Observable  
naar Type  
...  
ngOnInit() {  
    // Call naar de service, levert Observable op  
    this.cities$ = this.cityService.getCities()  
}
```

→ In de view:

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

Werken met Live API's

- MovieApp
- Oefeningen\210-services-live



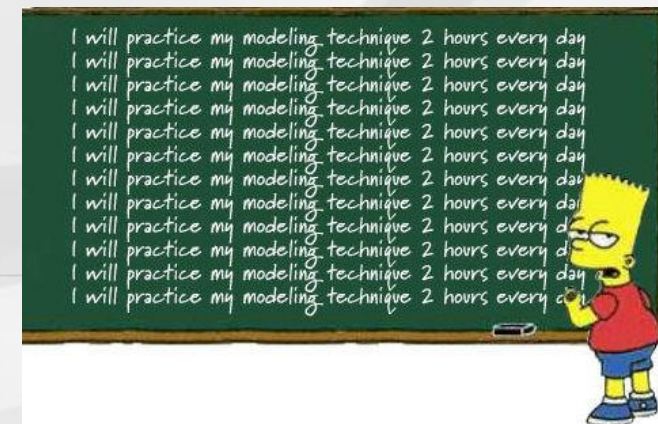
Voorbeeld API's

- <https://pokeapi.co/> - Pokemon API (CrossOrigin fout atm)
- <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
 - ...

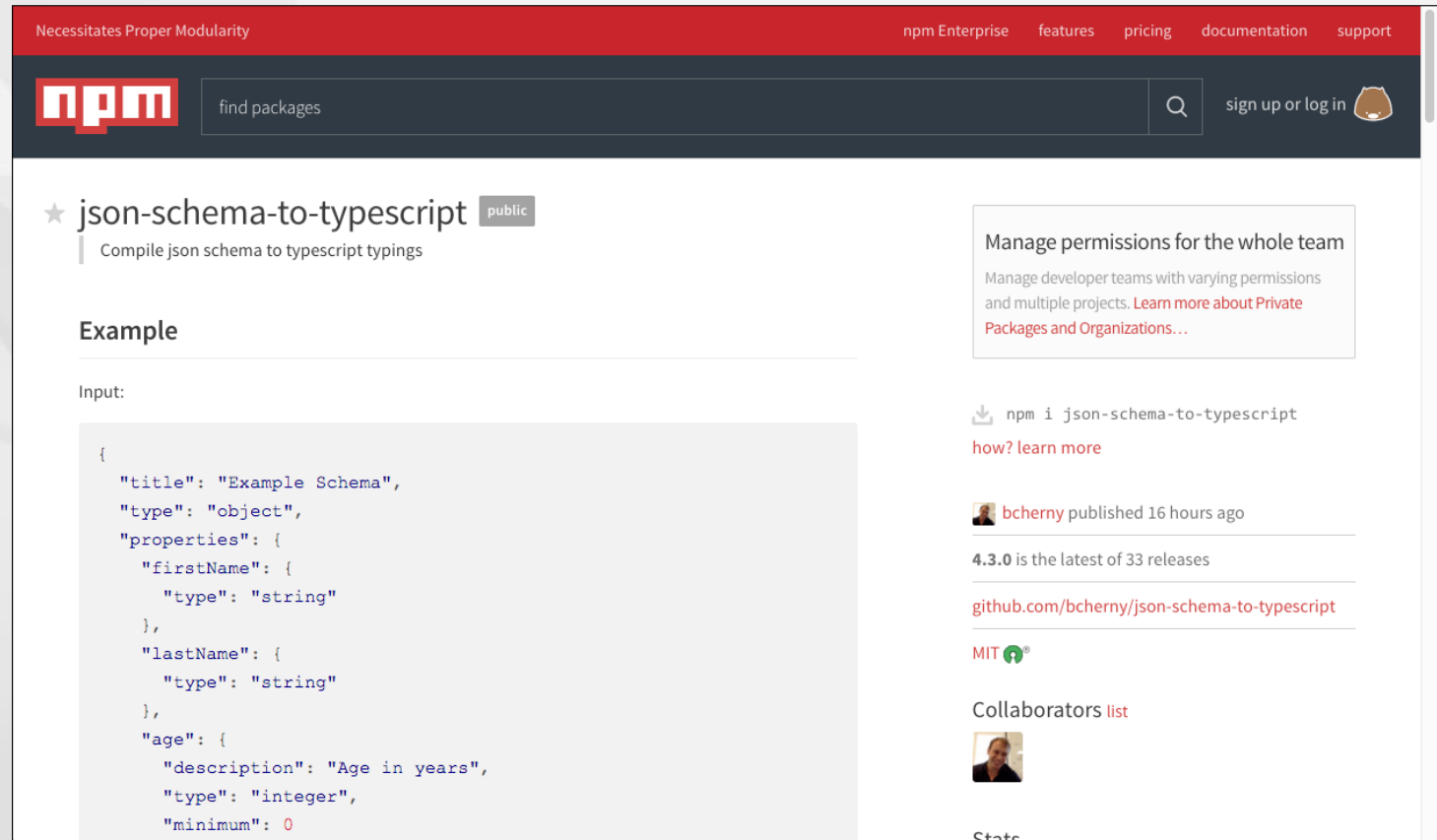
Exercise....



Bonus sheets: more info

Some pointers to more information on the internet

JSON Schema to typescript – server sided



The screenshot shows the npm package page for `json-schema-to-typescript`. The page has a red header with the npm logo and navigation links. The package name is highlighted with a star and a 'public' badge. Below the package name is a description: 'Compile json schema to typescript typings'. An 'Example' section shows a JSON schema input and its corresponding TypeScript output. On the right, there are sections for team management, installation instructions, version information, and collaborators.

Necessitates Proper Modularity

npm Enterprise features pricing documentation support

npm find packages

sign up or log in

★ json-schema-to-typescript public

Compile json schema to typescript typings

Example

Input:

```
{
  "title": "Example Schema",
  "type": "object",
  "properties": {
    "firstName": {
      "type": "string"
    },
    "lastName": {
      "type": "string"
    },
    "age": {
      "description": "Age in years",
      "type": "integer",
      "minimum": 0
    }
  }
}
```

Manage permissions for the whole team

Manage developer teams with varying permissions and multiple projects. [Learn more about Private Packages and Organizations...](#)

npm i json-schema-to-typescript

[how? learn more](#)

bcherny published 16 hours ago

4.3.0 is the latest of 33 releases

github.com/bcherny/json-schema-to-typescript


MIT

Collaborators [list](#)

Stats

<https://www.npmjs.com/package/json-schema-to-typescript>

Data Mocking - 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
<input type="text" value="id"/>	Row Number	blank: <input type="text" value="0"/> % <input type="text" value="fx"/> ×
<input type="text" value="first_name"/>	First Name	blank: <input type="text" value="0"/> % <input type="text" value="fx"/> ×
<input type="text" value="last_name"/>	Last Name	blank: <input type="text" value="0"/> % <input type="text" value="fx"/> ×
<input type="text" value="email"/>	Email Address	blank: <input type="text" value="0"/> % <input type="text" value="fx"/> ×
<input type="text" value="gender"/>	Gender	blank: <input type="text" value="0"/> % <input type="text" value="fx"/> ×
<input type="text" value="ip_address"/>	IP Address v4	blank: <input type="text" value="0"/> % <input type="text" value="fx"/> ×

Add another field

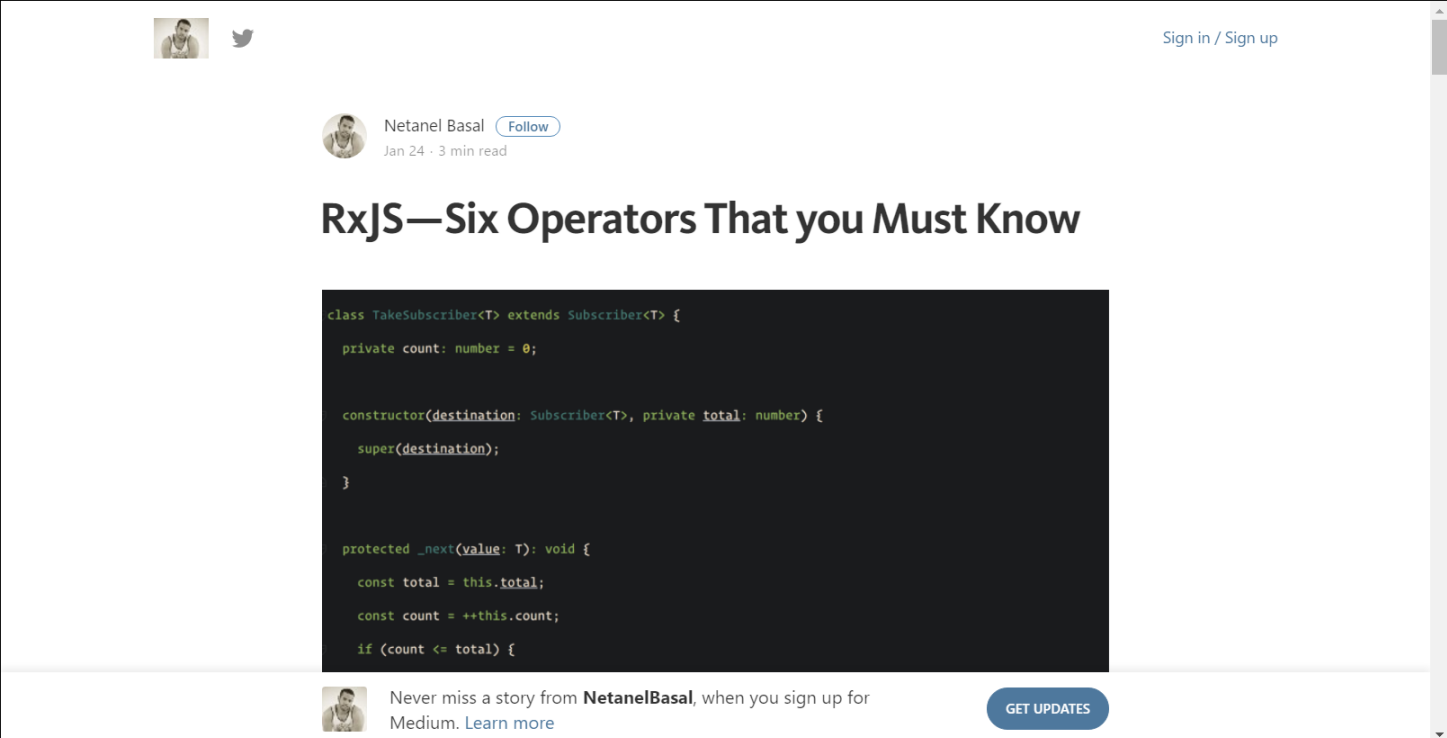
Rows:
Format:
Line Ending:
Include: ☒ header ☐ BOM

Download Data
Preview
More
Want to save this for later? [Sign up for free.](#)

Useful operators

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

6 Operators you “must know”



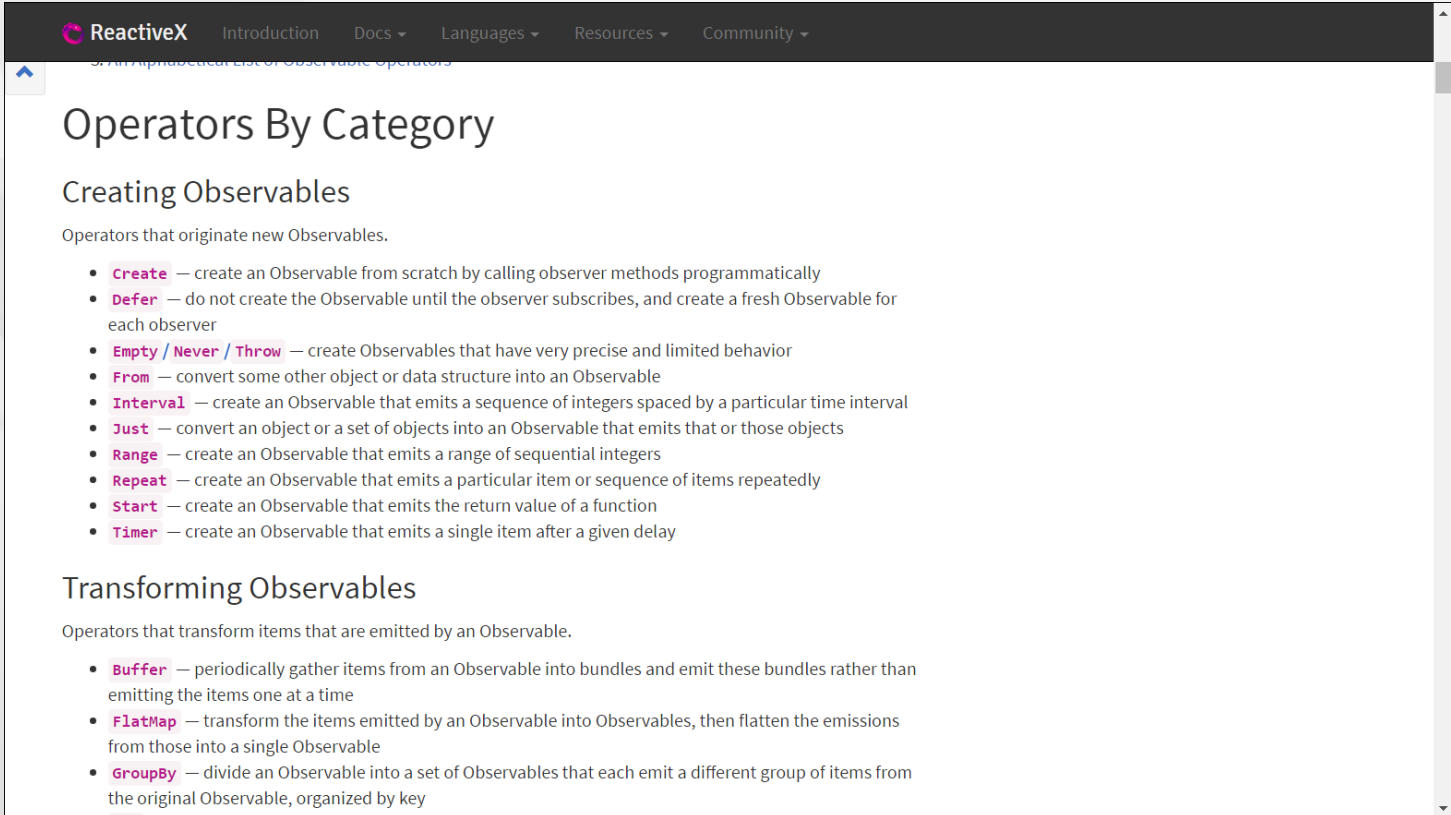
The screenshot shows a Medium article interface. At the top right, there are links for 'Sign in / Sign up'. Below this is the author's profile for 'Netanel Basal' with a 'Follow' button and a timestamp 'Jan 24 · 3 min read'. The article title is 'RxJS—Six Operators That you Must Know'. The main content is a code block with the following TypeScript code:

```
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) {
```

At the bottom of the article preview, there is a note: 'Never miss a story from NetanelBasal, when you sign up for Medium. Learn more' and a 'GET UPDATES' button.

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

Documentation at reactivex.io



The screenshot shows the ReactiveX website's navigation bar with links for Introduction, Docs, Languages, Resources, and Community. The main content area is titled 'Operators By Category' and focuses on 'Creating Observables'. It lists ten operators: Create, Defer, Empty / Never / Throw, From, Interval, Just, Range, Repeat, Start, and Timer, each with a brief description of its function. Below this, the 'Transforming Observables' section lists Buffer, FlatMap, and GroupBy operators.

ReactiveX Introduction Docs Languages Resources Community

Operators By Category

Creating Observables

Operators that originate new Observables.

- **Create** — create an Observable from scratch by calling observer methods programmatically
- **Defer** — do not create the Observable until the observer subscribes, and create a fresh Observable for each observer
- **Empty / Never / Throw** — create Observables that have very precise and limited behavior
- **From** — convert some other object or data structure into an Observable
- **Interval** — create an Observable that emits a sequence of integers spaced by a particular time interval
- **Just** — convert an object or a set of objects into an Observable that emits that or those objects
- **Range** — create an Observable that emits a range of sequential integers
- **Repeat** — create an Observable that emits a particular item or sequence of items repeatedly
- **Start** — create an Observable that emits the return value of a function
- **Timer** — create an Observable that emits a single item after a given delay

Transforming Observables

Operators that transform items that are emitted by an Observable.

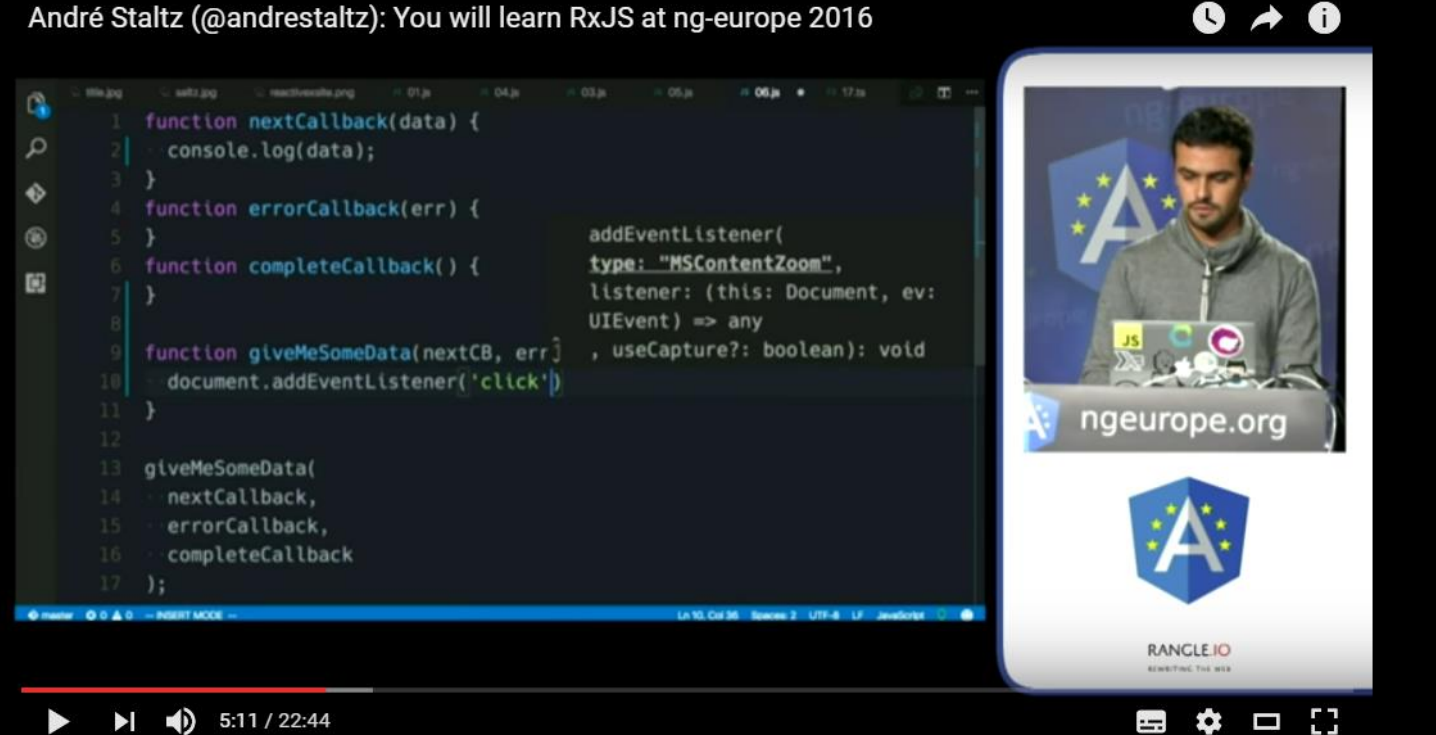
- **Buffer** — periodically gather items from an Observable into bundles and emit these bundles rather than emitting the items one at a time
- **FlatMap** — transform the items emitted by an Observable into Observables, then flatten the emissions from those into a single Observable
- **GroupBy** — divide an Observable into a set of Observables that each emit a different group of items from the original Observable, organized by key

<http://reactivex.io/documentation/operators.html>

Creating Observables from scratch

- André Staltz

André Staltz (@andrestaltz): You will learn RxJS at ng-europe 2016



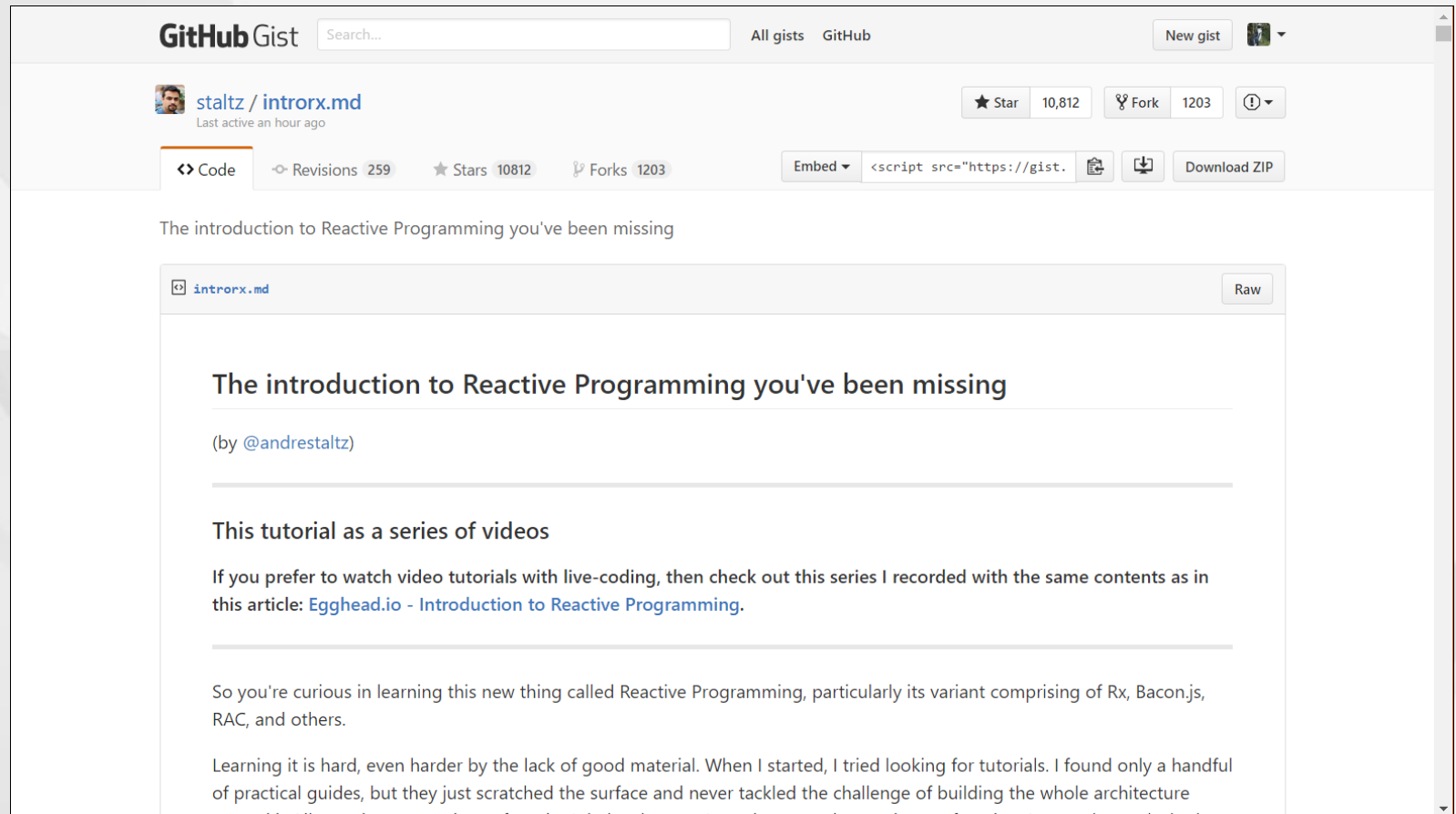
```
1 function nextCallback(data) {  
2   console.log(data);  
3 }  
4 function errorCallback(err) {  
5 }  
6 function completeCallback() {  
7 }  
8  
9 function giveMeSomeData(nextCB, err) {  
10  document.addEventListener('click',  
11  
12  
13  giveMeSomeData(  
14    nextCallback,  
15    errorCallback,  
16    completeCallback  
17  );  
18  
19  addEventListener(  
20    type: "MSContentZoom",  
21    listener: (this: Document, ev:  
22      UIEvent) => any  
23    , useCapture?: boolean): void  
24  )
```

ng-europe.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 has a search bar at the top, navigation links for 'All gists' and 'GitHub', and a 'New gist' button. The user's profile picture and name are shown, along with the file name and a note '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 are also buttons for 'Embed', a script tag, a clipboard icon, a download icon, and 'Download ZIP'. The main content area shows the title 'The introduction to Reactive Programming you've been missing' and the author '(by @andrestaltz)'. Below the title, there is a section 'This tutorial as a series of videos' with a paragraph: '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](#).' Another paragraph follows: 'So you're curious in learning this new thing called Reactive Programming, particularly its variant comprising of Rx, Bacon.js, RAC, and others.' The final visible paragraph starts with '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>

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

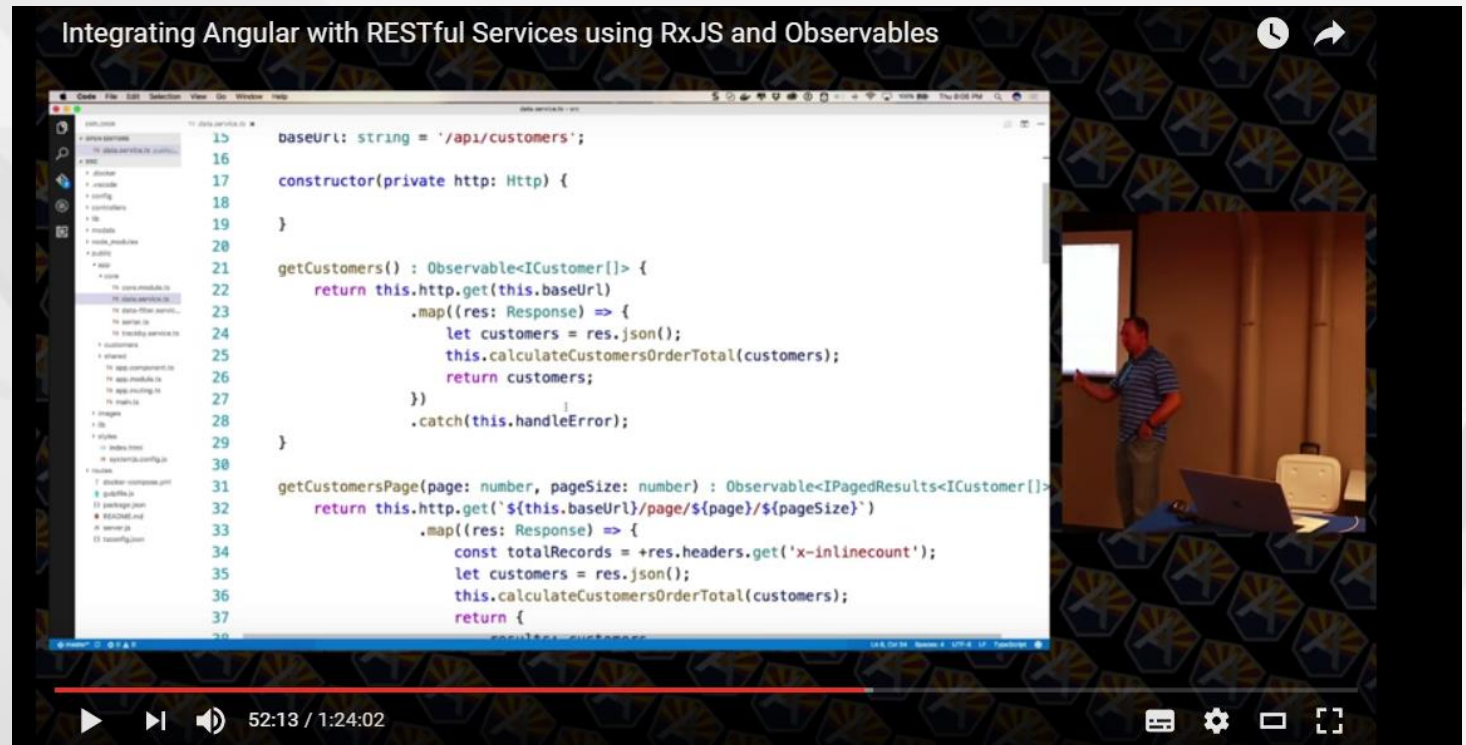
- [distinct](#)
- [distinctUntilChanged](#)
- [elementAt](#)
- [filter](#)
- [find](#)
- [first](#)

merge

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

<http://rxmarbles.com/>

Dan Wahlin on Modules and Observables



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



THOUGHTAM

TRAINING

CODE REVIEW

BLOG

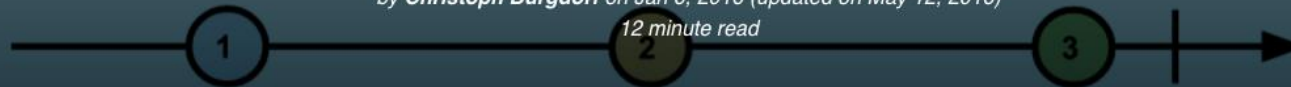


TAKING ADVANTAGE OF OBSERVABLES IN ANGULAR 2

`distinctUntilChanged()`

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.thoughtam.io/angular/2016/01/06/taking-advantage-of-observables-in-angular2.html>