IK WIL

Angular – Module Observables

vijfhart
IT-OPLEIDINGEN

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	LANGUAGES	RESOURCES	COMMUNITY
Observable	RxJava	Tutorials	GitHub
Operators	RxJS		Twitter
Single	Rx.NET		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.

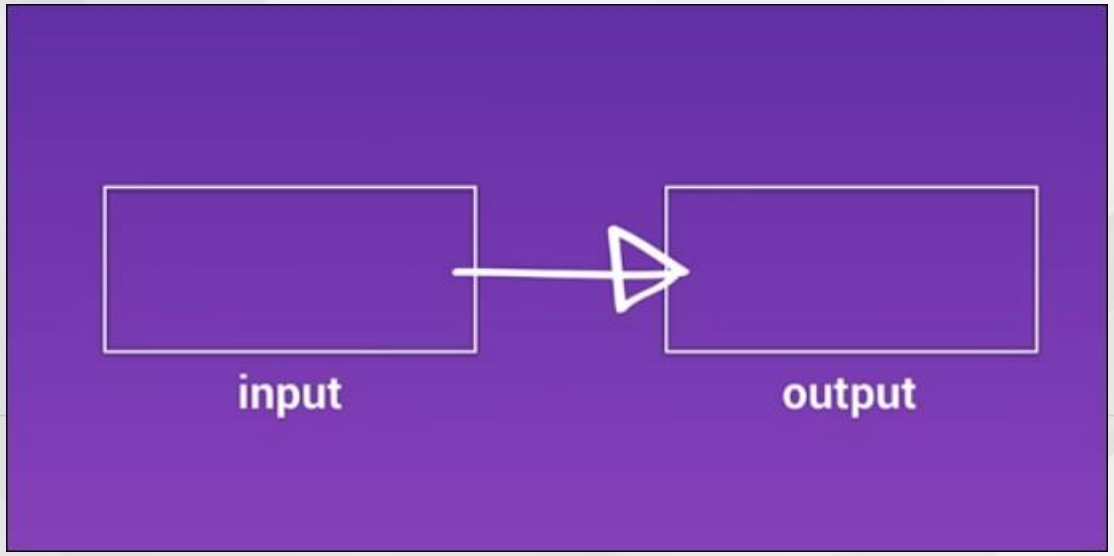
https://auth0.com/blog/2015/10/15/angular-2-series-part-3-using-http/

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

How do observables work

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

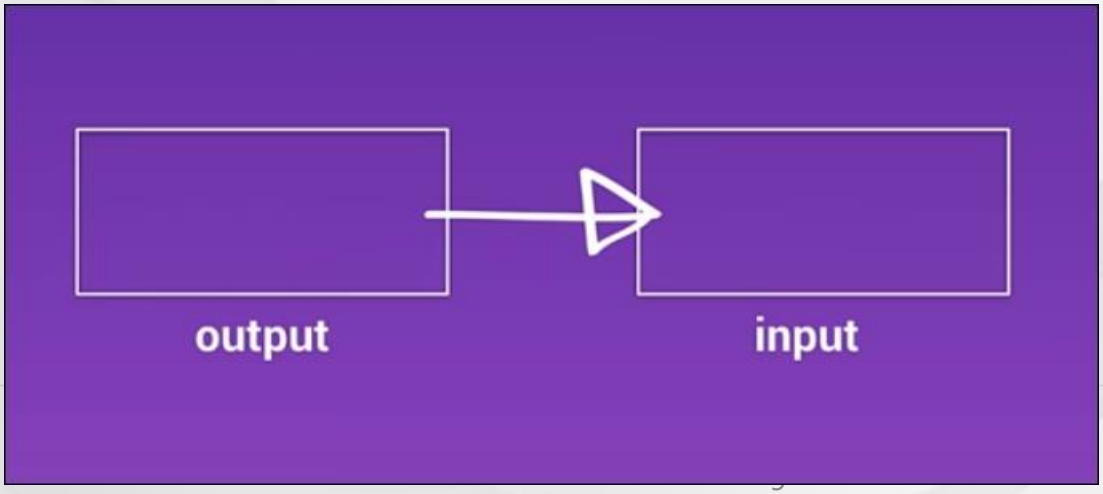


Go beast mode with realtime reactive interfaces in Angular 2 & Firebase | Lukas Ruebbelke

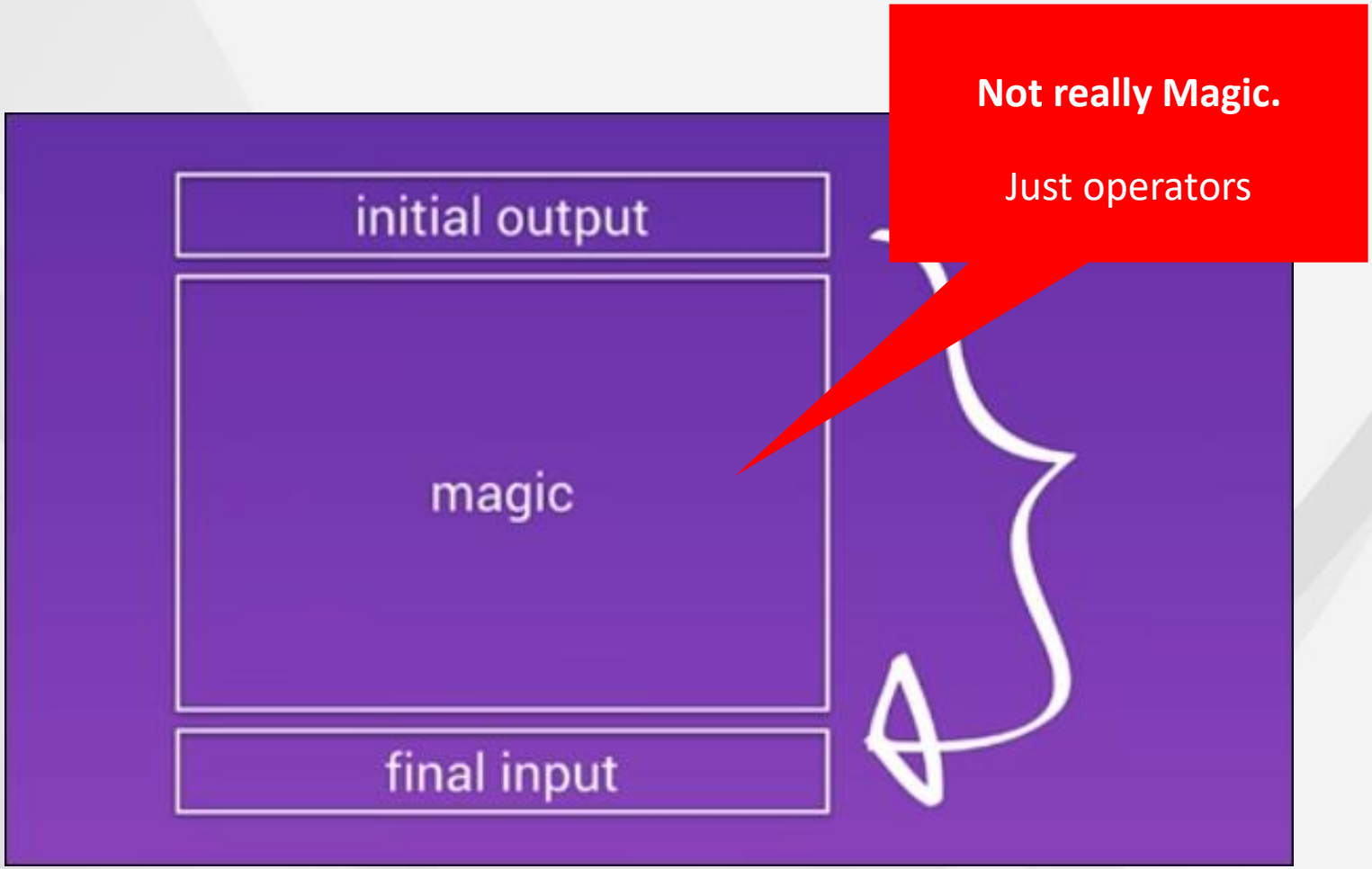
https://www.youtube.com/watch?v=5CTL7aqSvJU

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

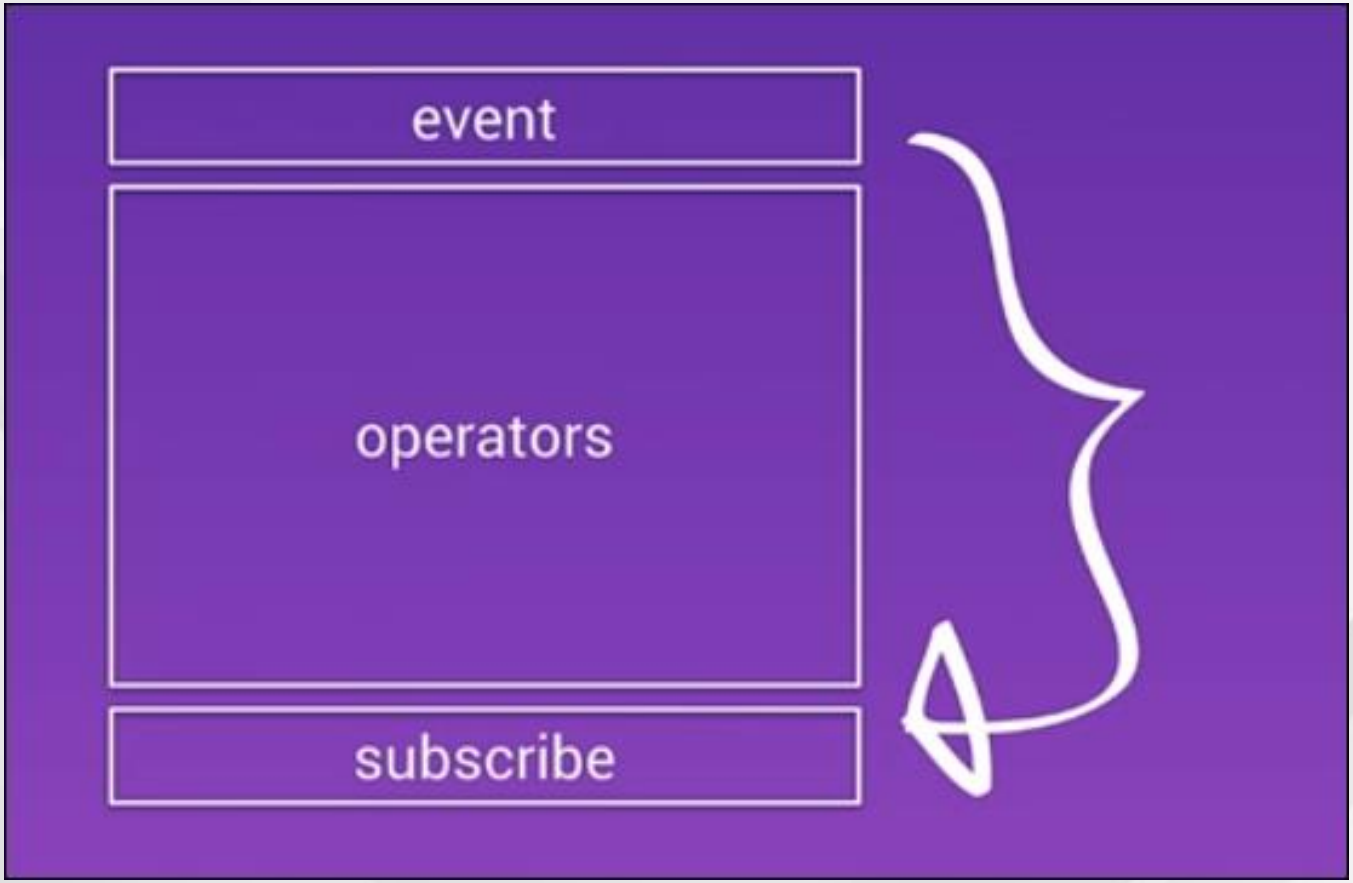
- With Observables -
  - a system, already outputting data,
  - Subscribe to that data
- "trade Output for Input"
- "Push vs. Pull" (bijv. Observable versus Functie)



"The observable sandwich"



Subscribe to events



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

Initial Output

Operator(s)

Final Input

Ook: HttpModule in je @ngModule importeren

```
// 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+: `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
  - `HttpModule` wordt in toekomstige versies verwijderd

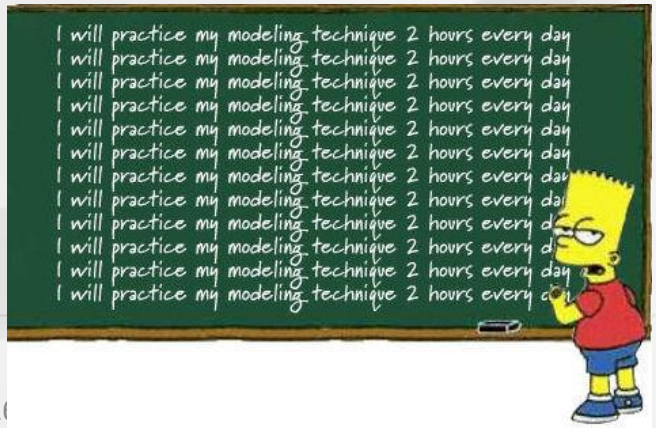
Met `HttpClientModule` – geen mapping `.json()`

```
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….

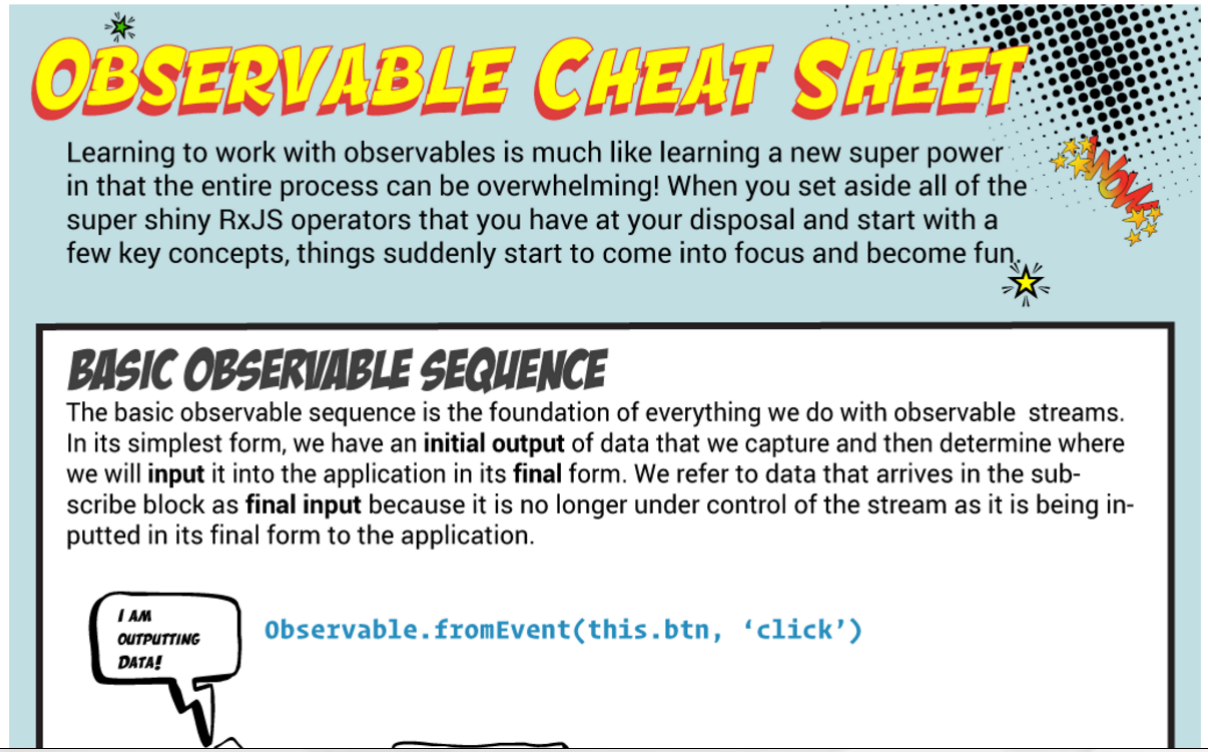


Observable Cheat Sheet

genius to understand.

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

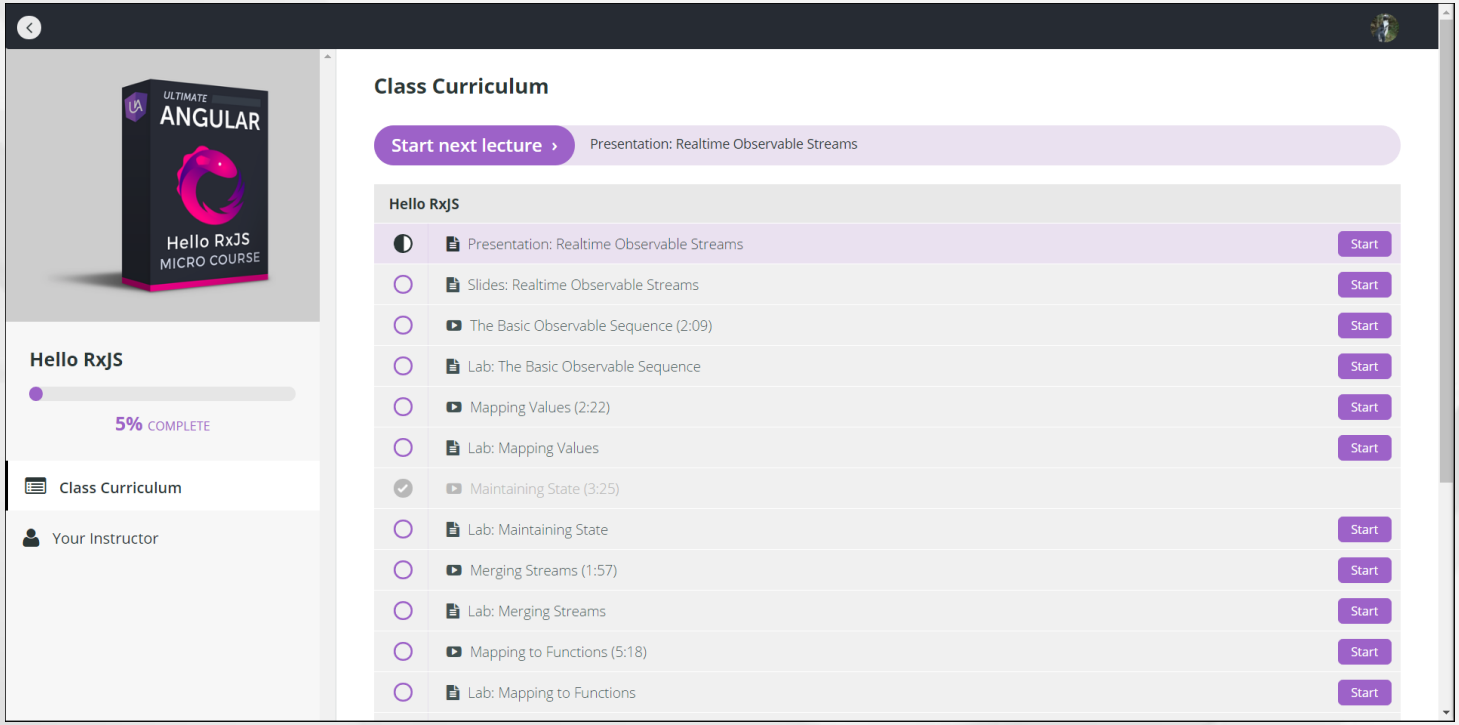
I really hope that you find the infographic helpful. Be sure to drop me a line below if you have any questions or comments. #highFive



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

Hello RxJS

Gratis online training



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';

@Injectable()
export class CityService {

   constructor(private http: Http) {

   }

   // retourneer alle cities
   getCities(): Observable<City[]> {
      return this.http.get('assets/data/cities.json')
         .map(cities => cities.json());
   }
}
```

Import operator

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/

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

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)

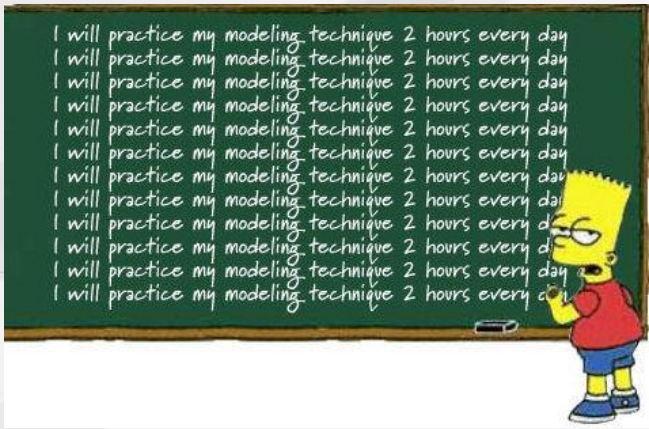
  http://openweathermap.org/API (weerbericht)

- 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

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

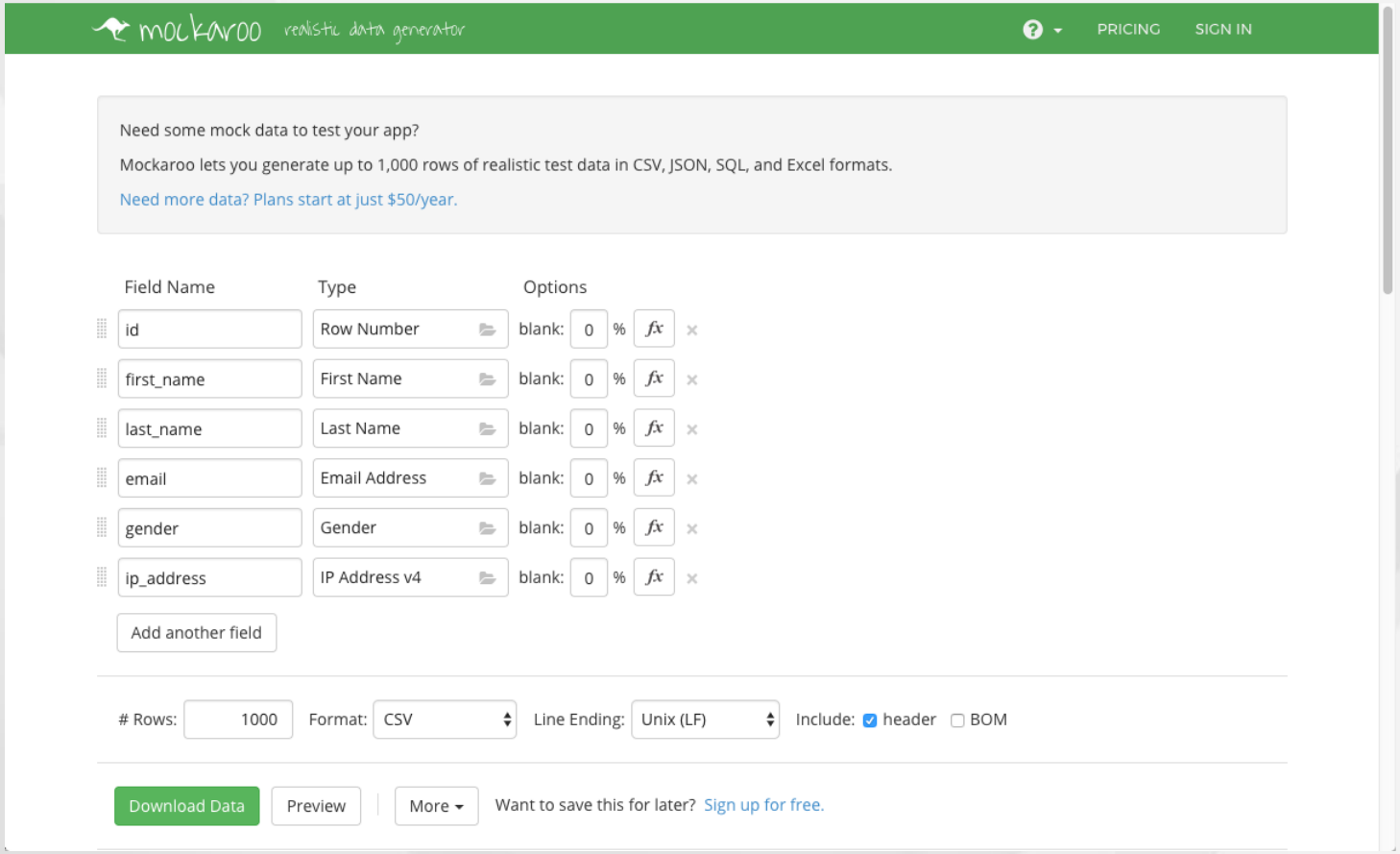
MIT

Collaborators list

Stats

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

Data Mocking - Mockaroo

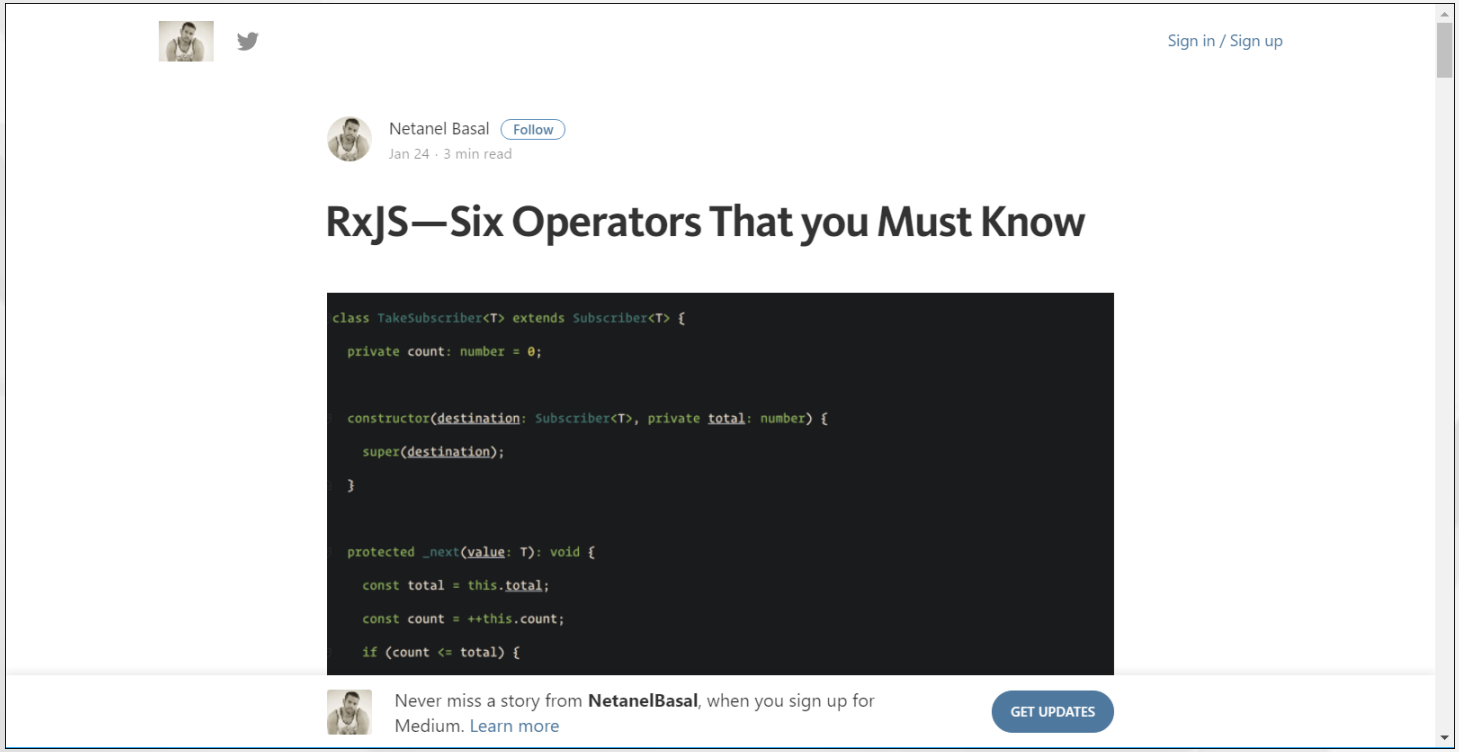


http://mockaroo.com/

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”



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

Documentation at reactivex.io

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

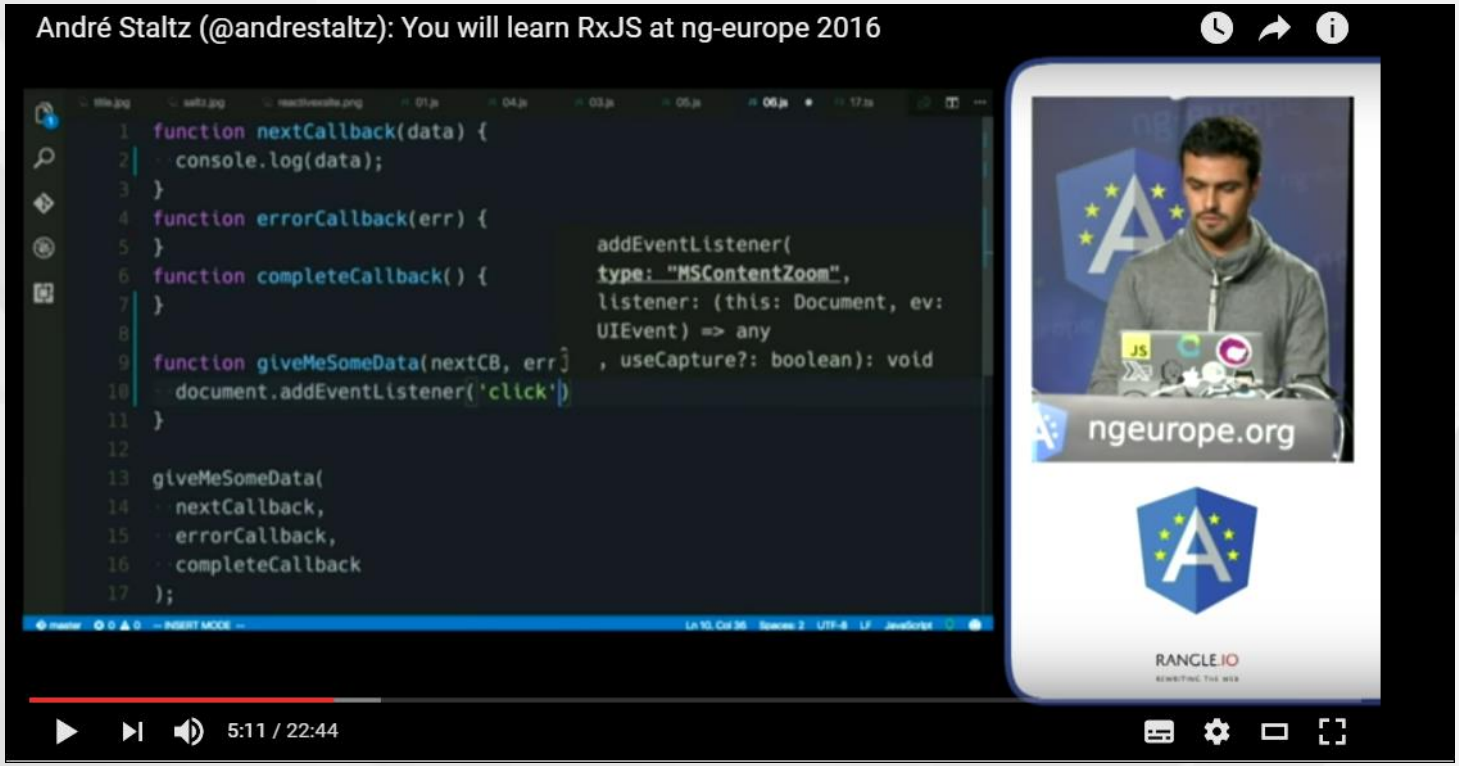
Transforming Observables

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



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

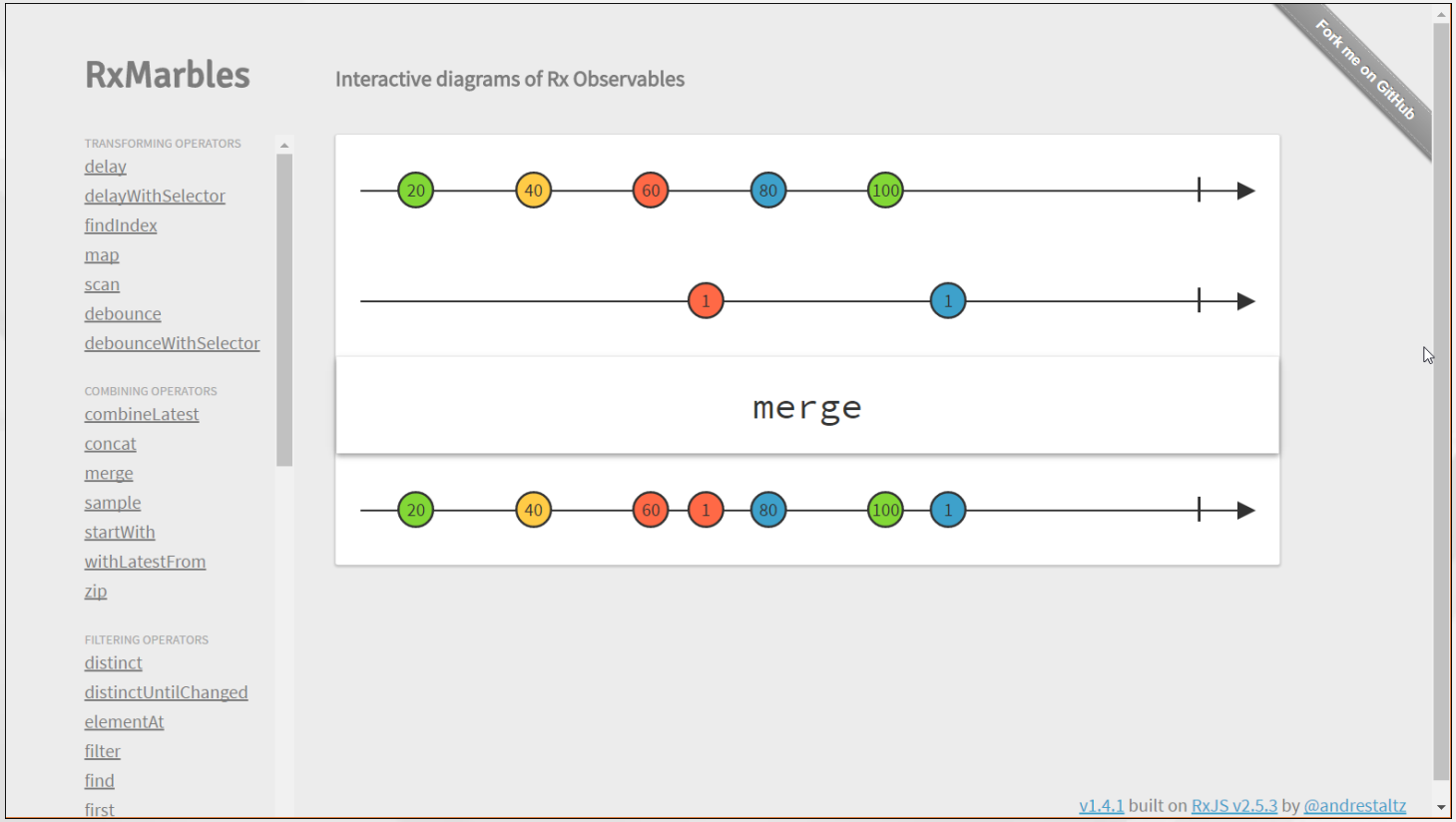
If you prefer to watch video tutorials with live-coding, then check out this series I recorded with the same contents as in this article: Egghead.io - Introduction to Reactive Programming.

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

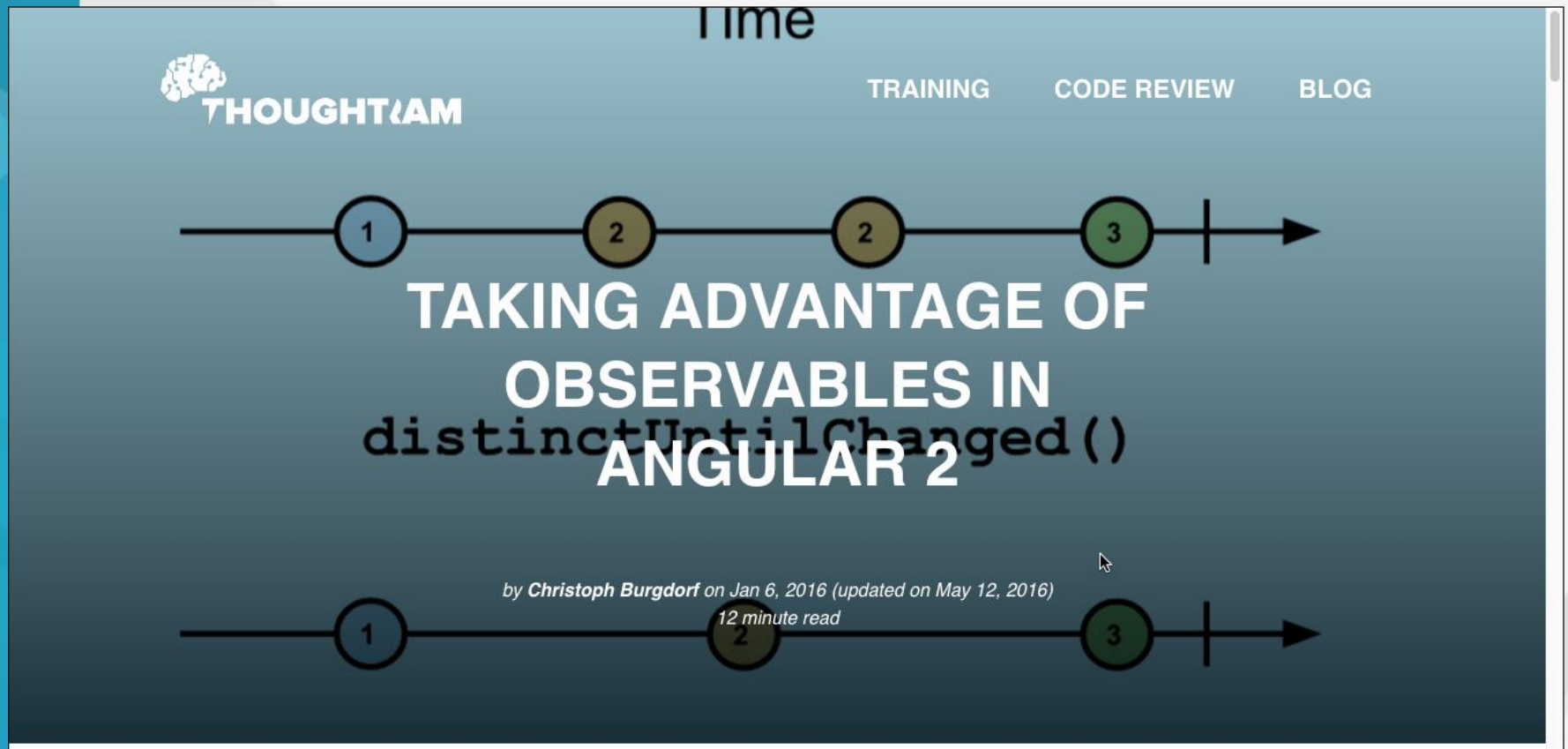
RxMarbles



http://rxmarbles.com/

Dan Wahlin on Modules and Observables

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



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