Angular Fundamentals Module – Observables

Peter Kassenaar –
info@kassenaar.com

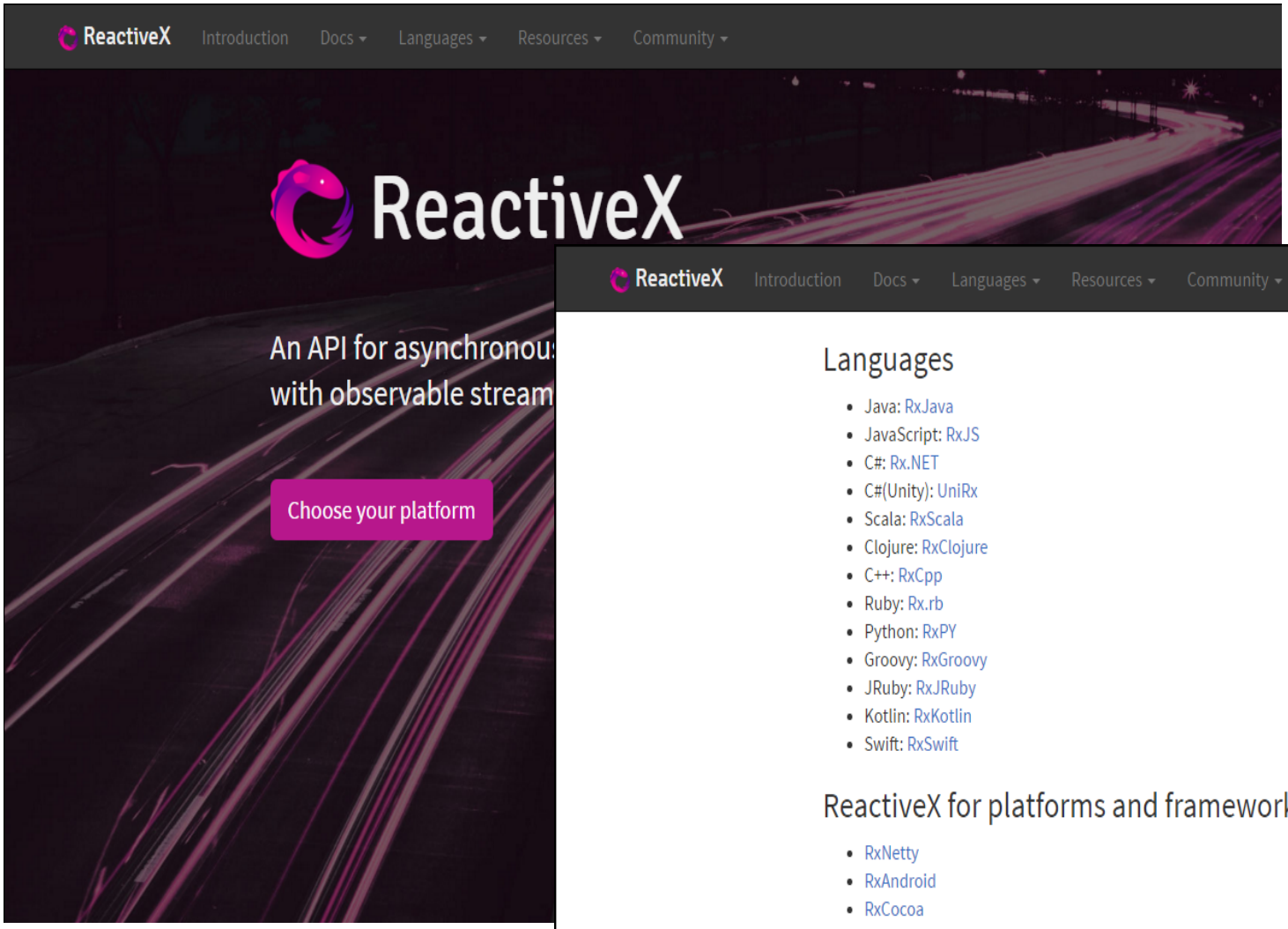
Async services met RxJS/Observables

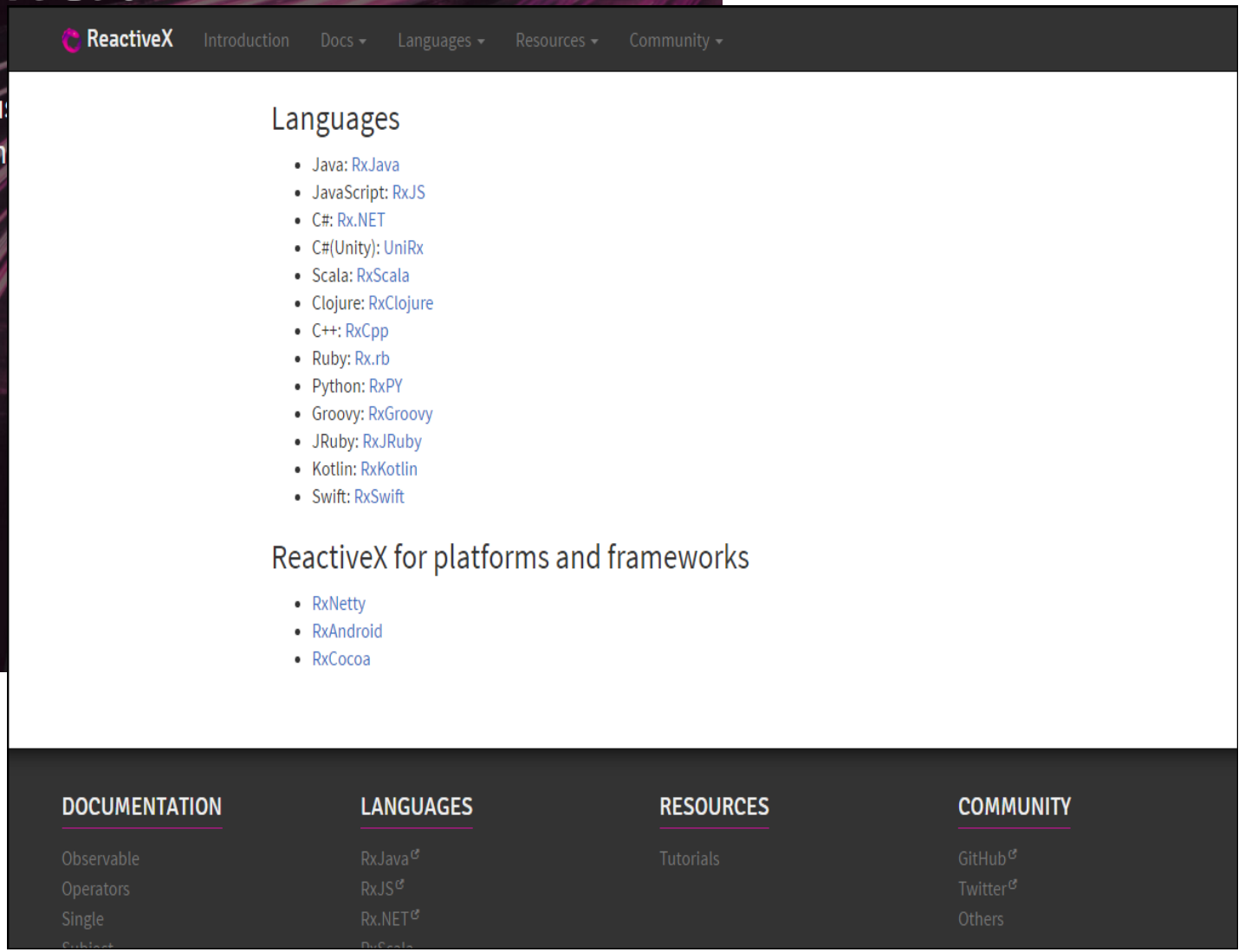
Reactive programming with asynchronous streams

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`





http://reactivex.io/

Why Observables?

We can do much more with observables than with promises.

With observables, we have a whole bunch of operators to pull from, which let us customize our streams in nearly any way we want.

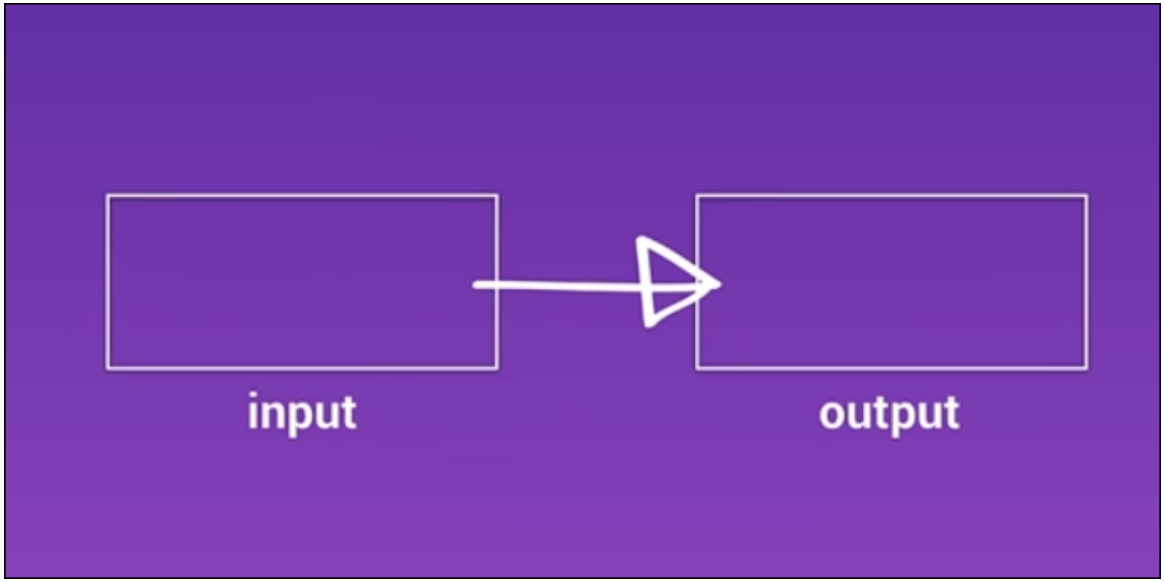
https://auth0.com/blog/2015/10/15/angular-2-series-part-3-using-http/

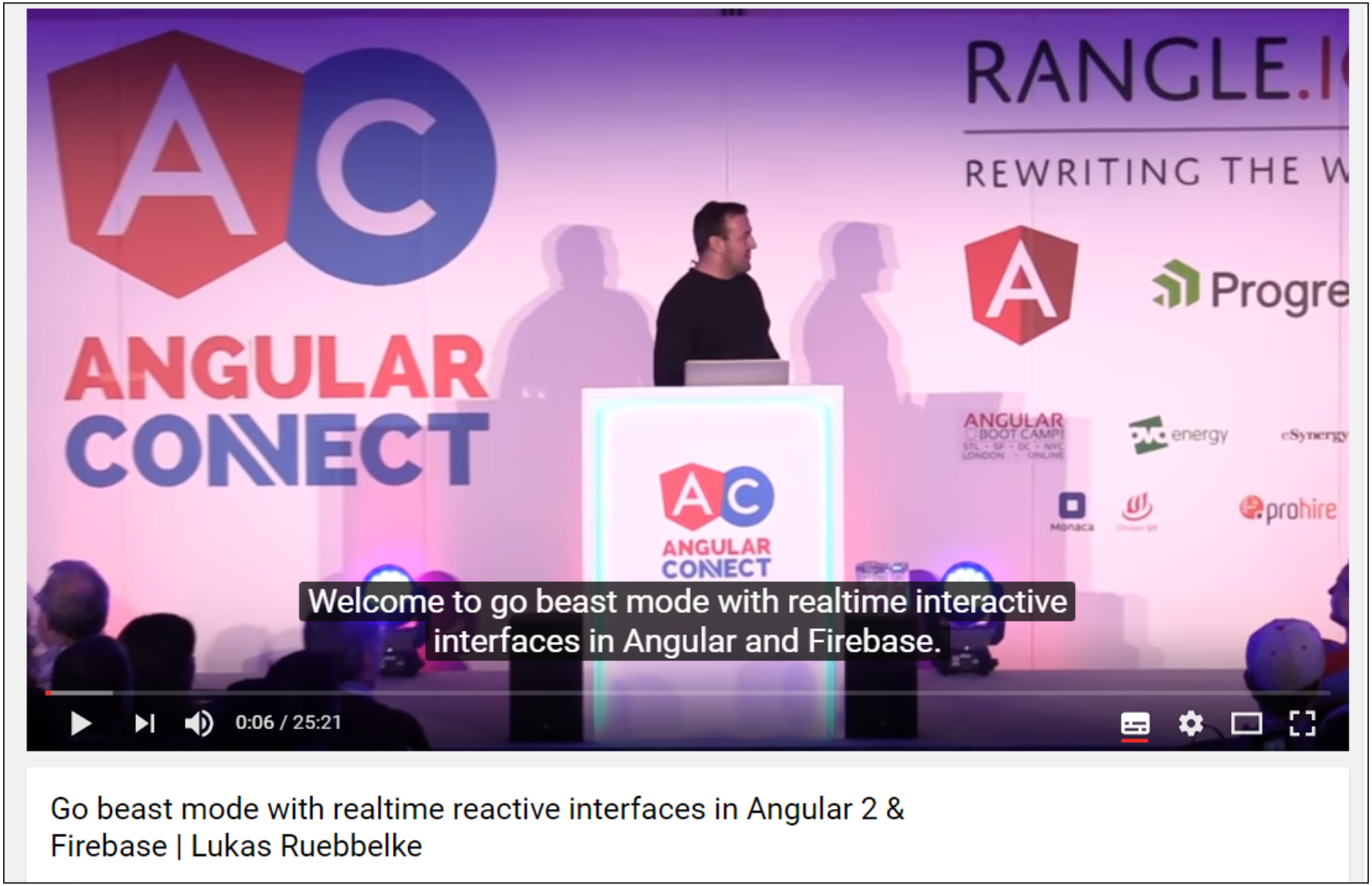
Observables en RxJs

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

How do observables work

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

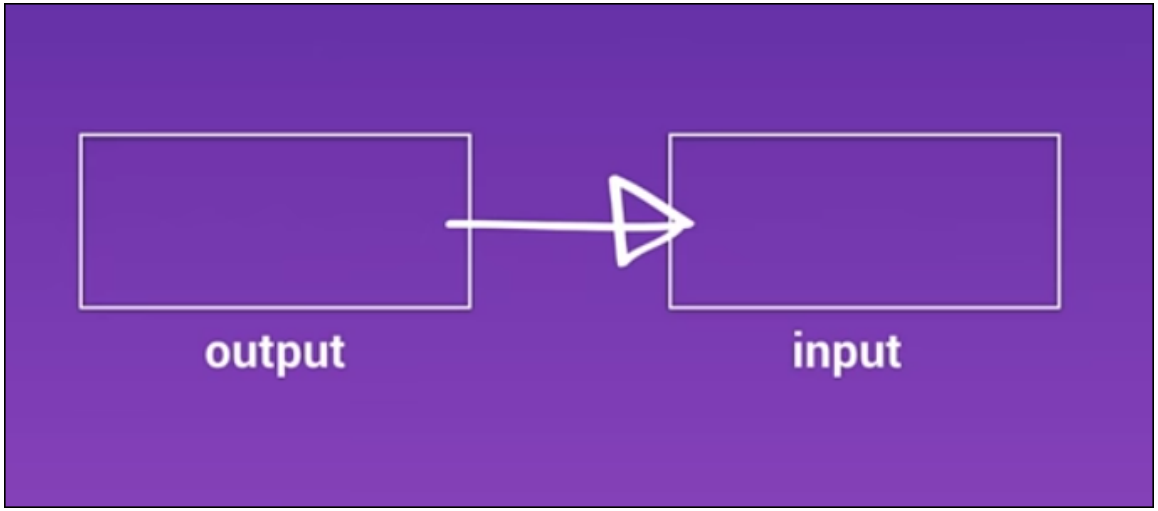




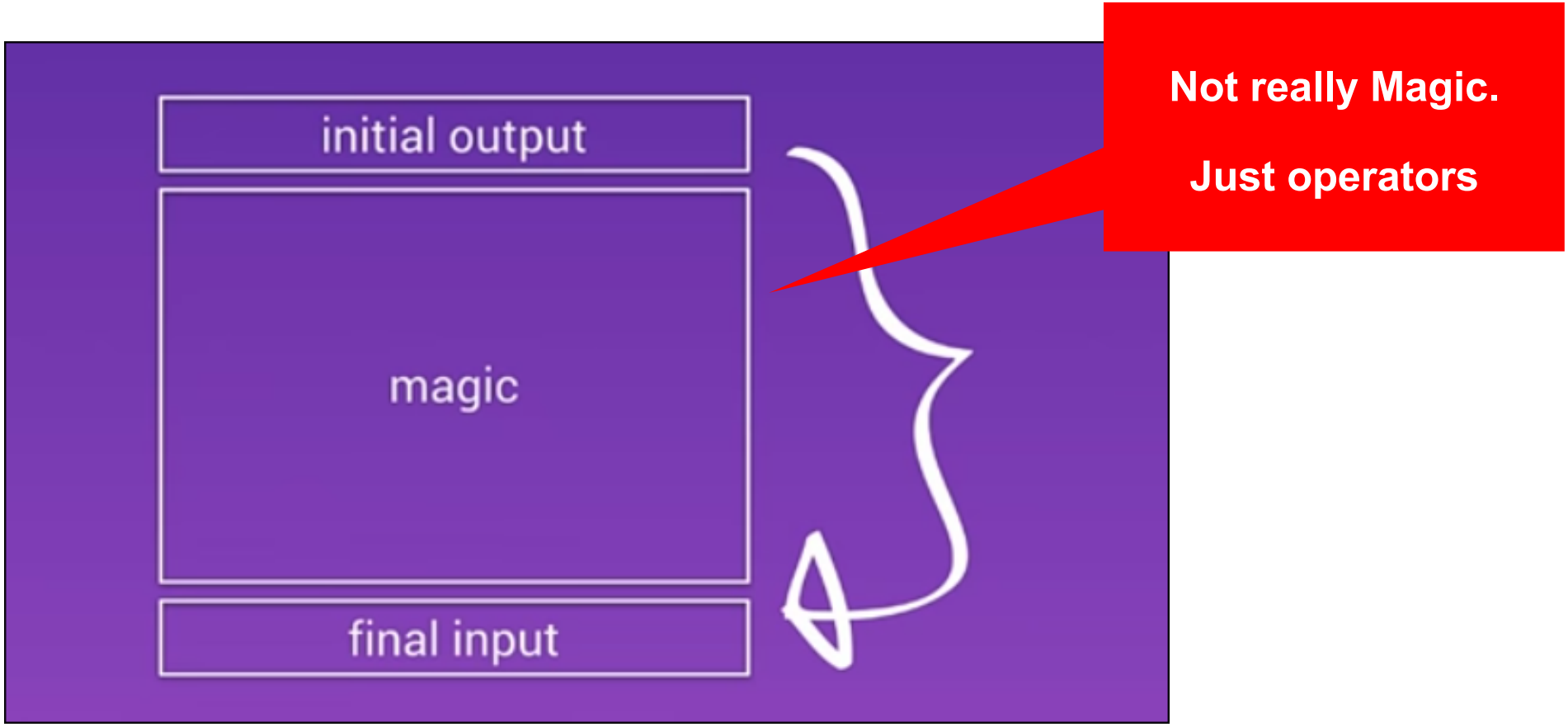
https://www.youtube.com/watch?v=5CTL7aqSvJU

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

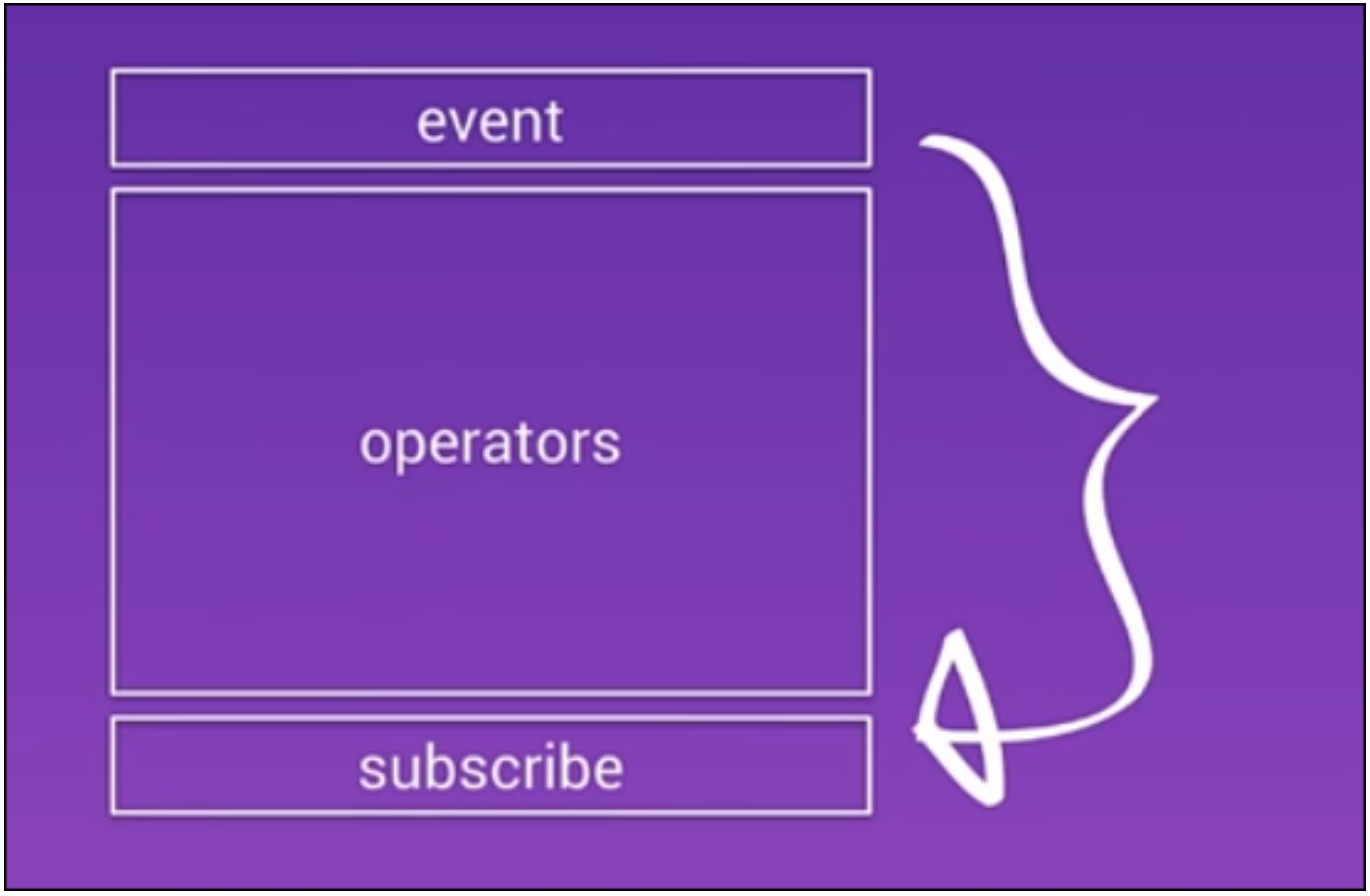
- With Observables -
  - a system, already outputting data,
  - Subscribe to that data
- "trade Output for Input"
- "Push vs. Pull"



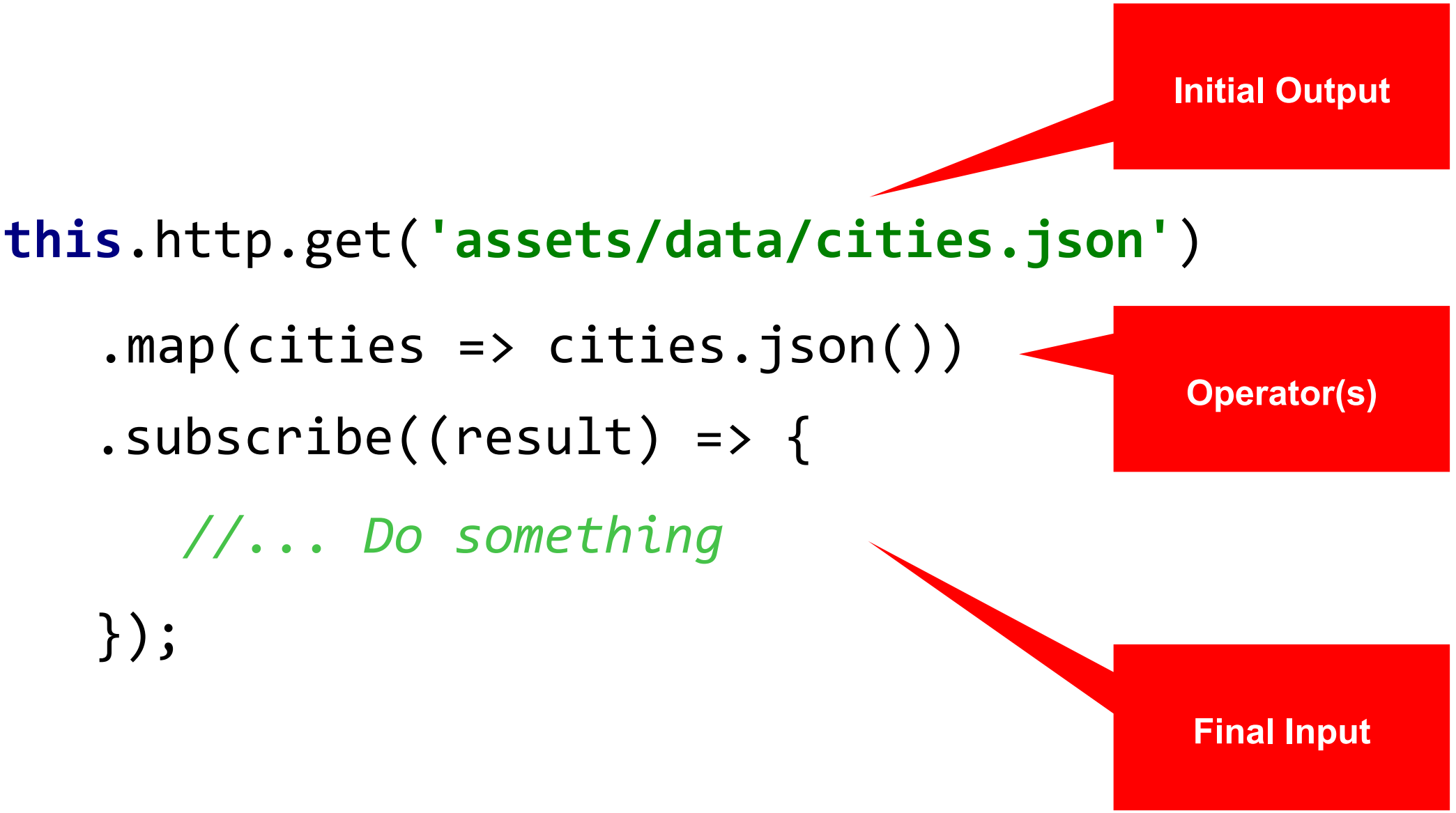
"The observable sandwich"



Subscribe to events



In code


Initial Output
this.http.get('assets/data/cities.json')
.map(cities => cities.json())
Operator(s)
.subscribe((result) => {
//... Do something
});
Final Input

Ook: importeren `HttpModule` in `@ngModule`

- *// Angular Modules*

  …

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

  // Module declaration

  @NgModule({

  imports : [BrowserModule, HttpModule],

  declarations: [AppComponent],

  bootstrap : [AppComponent],

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

  })

  export class AppModule {

  }

Angular 4.3/5+: `HttpClientModule`

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

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

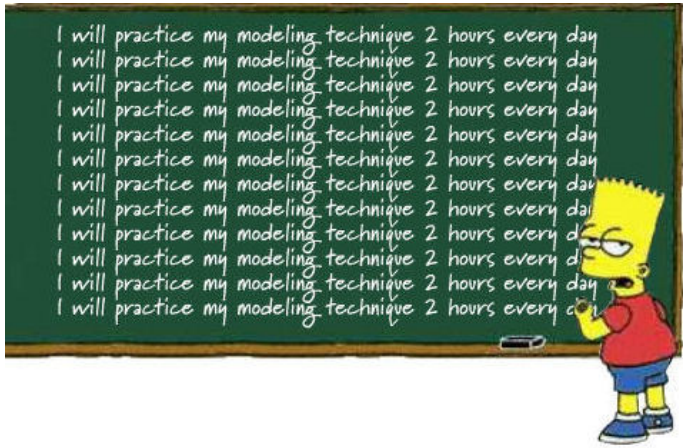
```
this.http.get<City[]>('assets/data/cities.json')
    .subscribe(result => {
        //... Do something
    });
```

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

Oefening

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

Exercise….

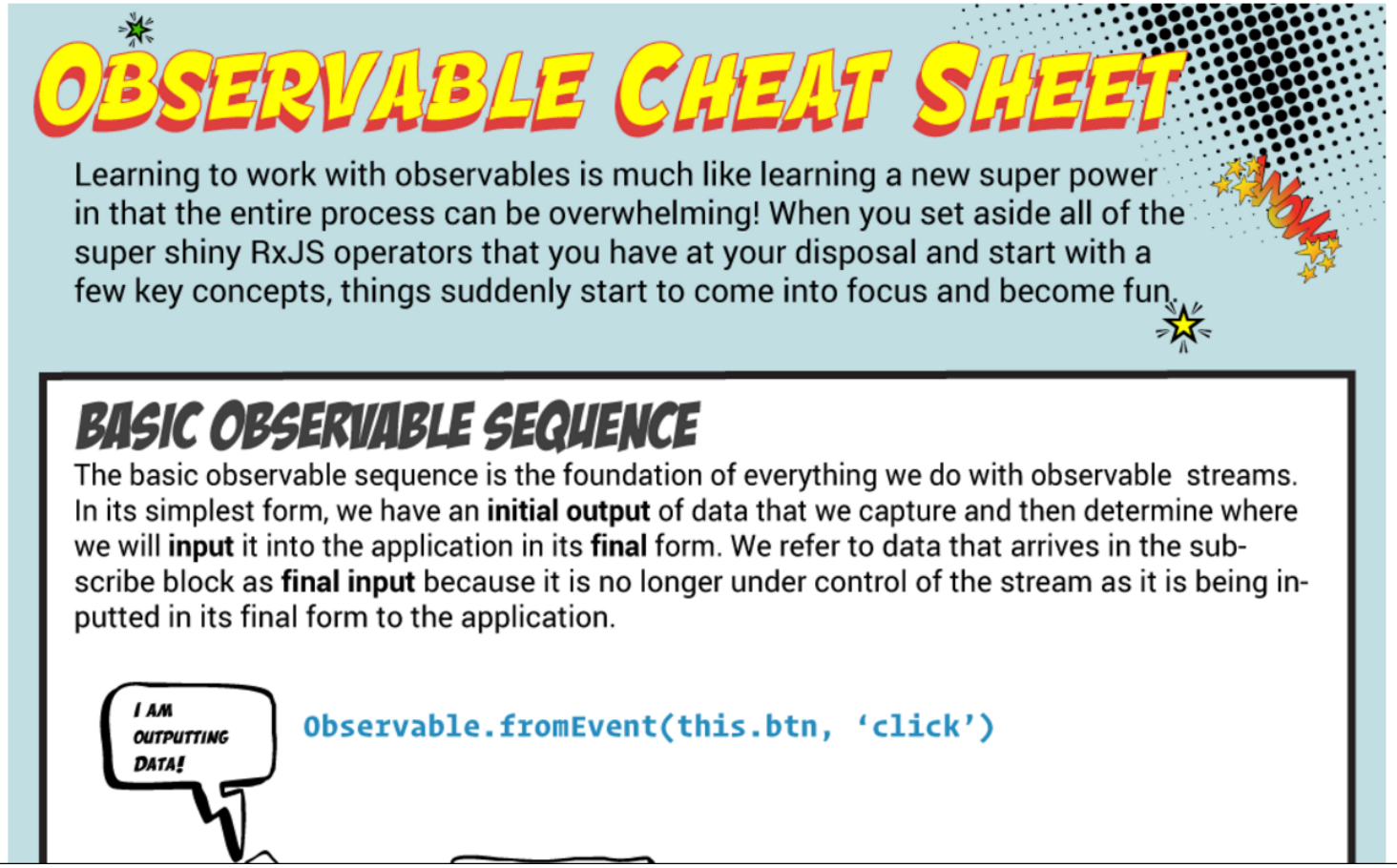


Observable Cheat Sheet

genius to understand.

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

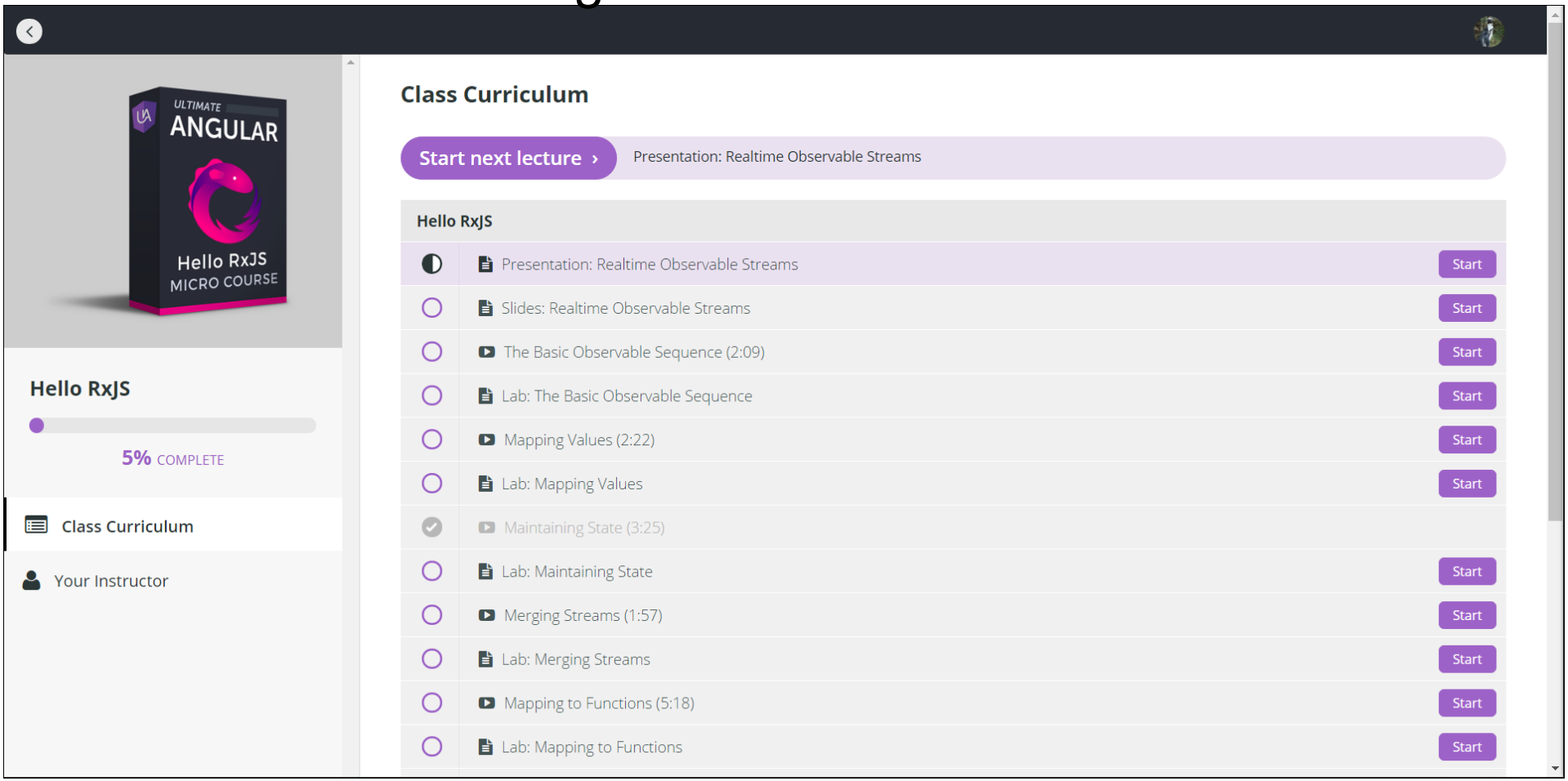
I really hope that you find the infographic helpful. Be sure to drop me a line below if you have any questions or comments. #highFive



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

Hello RxJS

Gratis online training



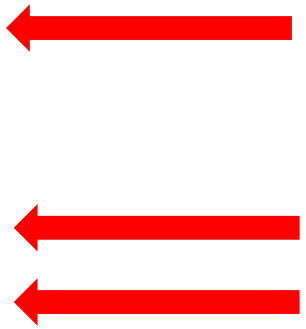
http://courses.ultimateangular.com/

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

```
import 'rxjs/Rx';
import {Observable} from "rxjs";
import 'rxjs/add/operator/map';
```

- Gebruik observable functies als `.map()`, `.filter()` etc.
- Piping. 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<Response> {
        return this.http.get('app/cities.json')
            .map(cities => cities.json());
    }

}
```

Import operator

Transform stream in de service

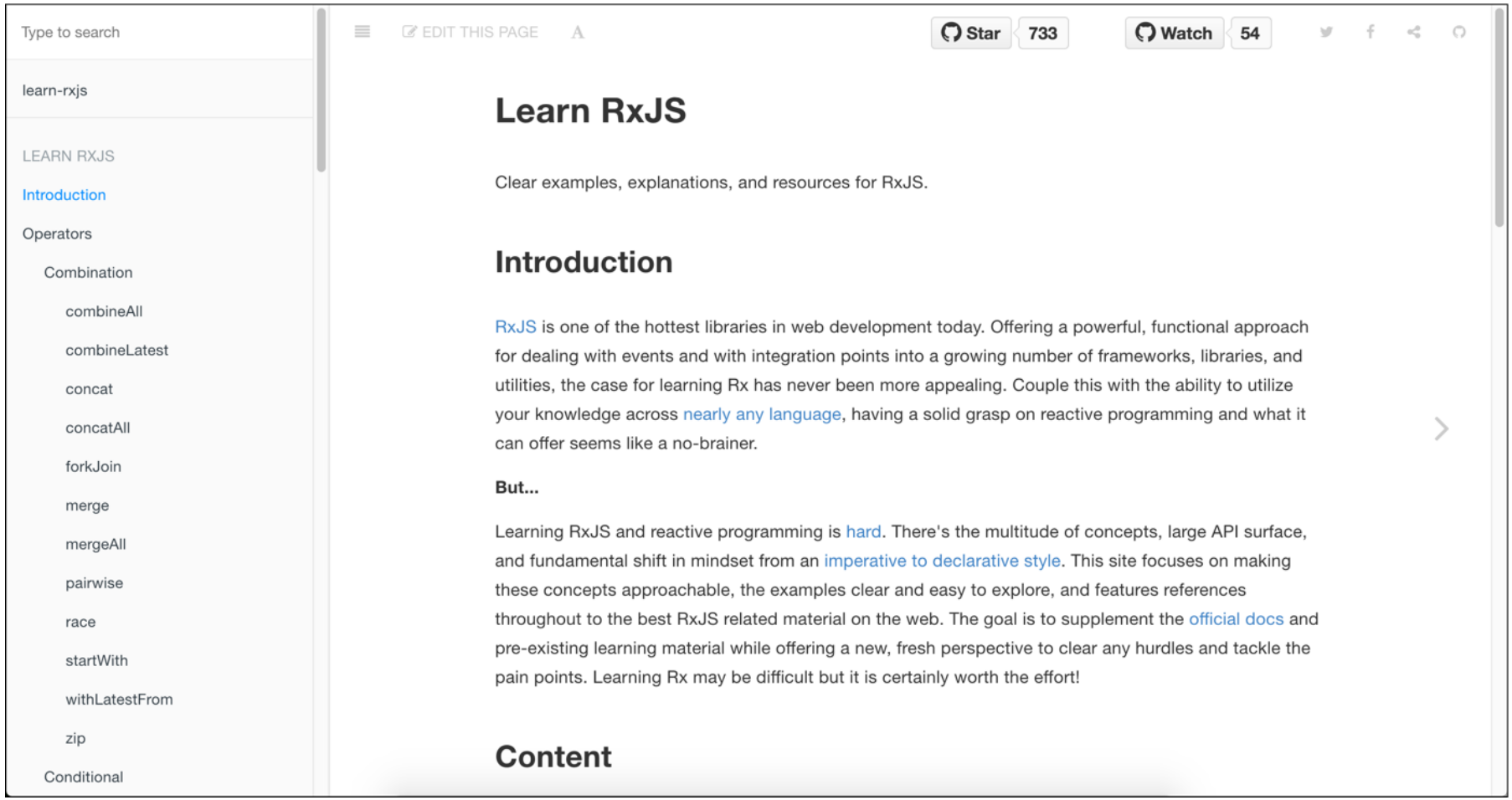
```
getCities() {
   if (!this.cities) {
      this.cityService.getCities()
         .map(res => res.json())
         .delay(3000)
         .subscribe(cityData => {
               this.cities = cityData;
            },
            err => console.log(err),
            ()=> console.log('Getting cities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

RxJs-functies

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/



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”>
```

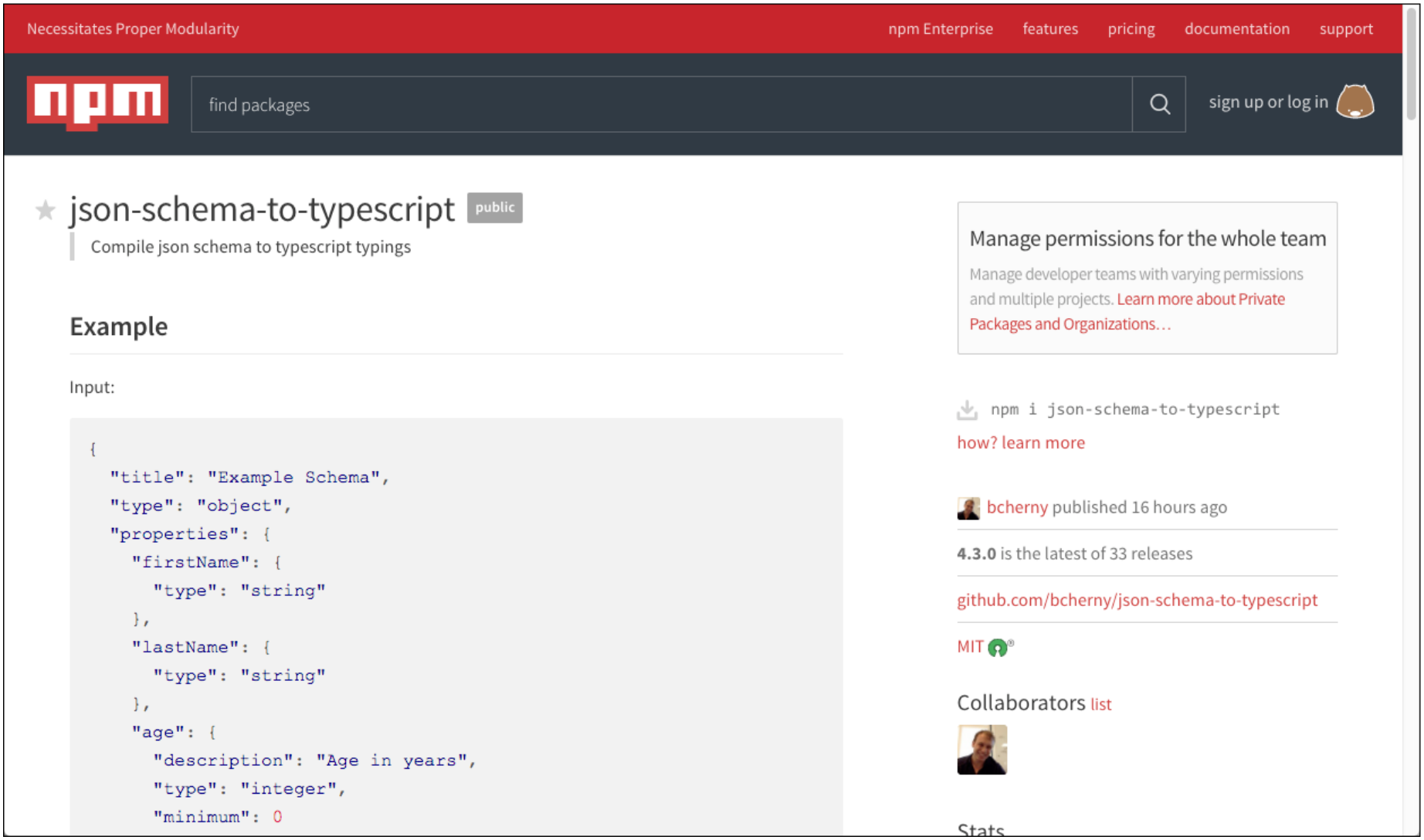
Werken met Live API’s

- MovieApp
- `Oefeningen\09-services-live`

Voorbeeld API’s

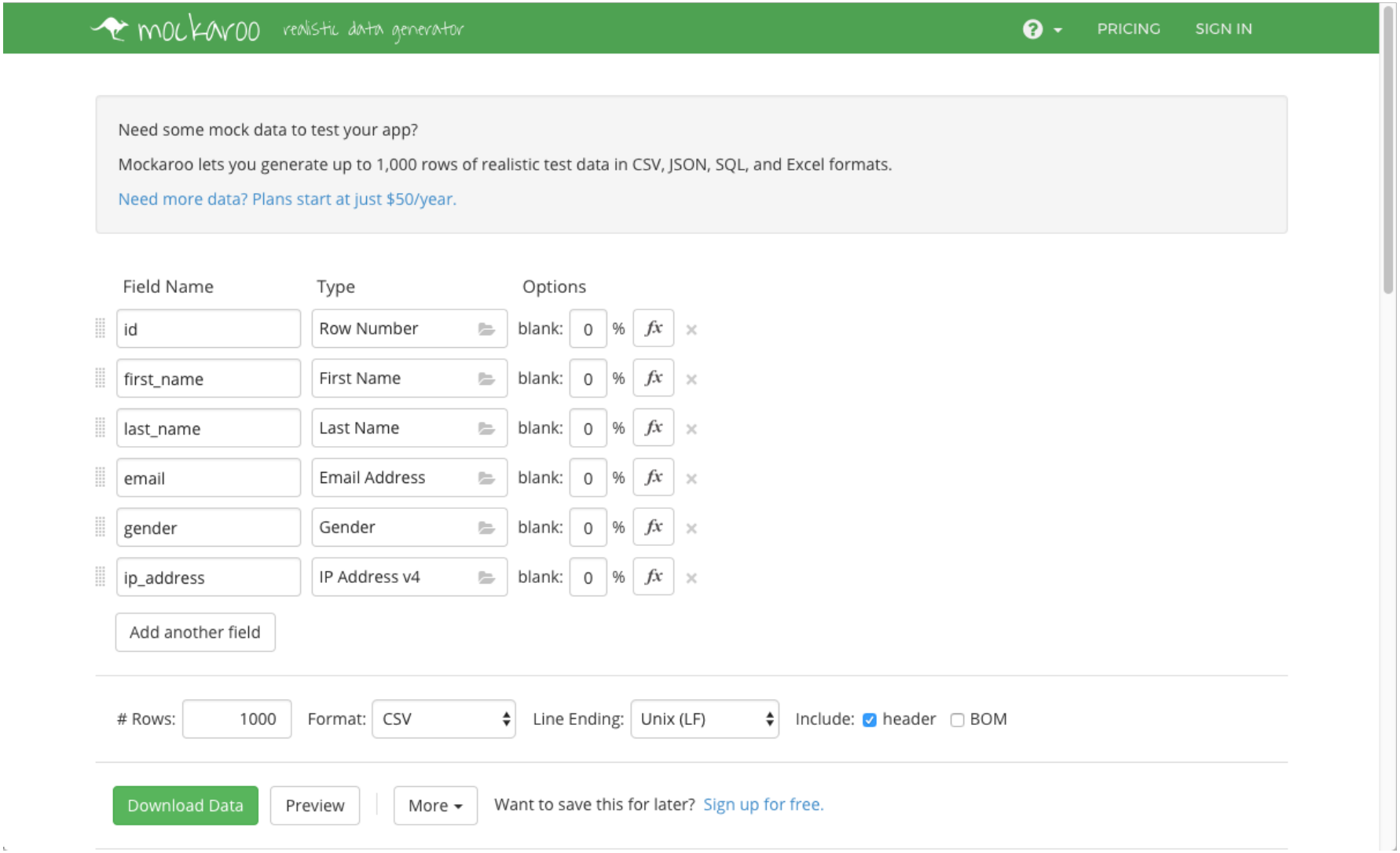
- https://pokeapi.co/ - Pokemon API
- http://openweathermap.org/API (weerbericht)
- http://filltext.com/ (random NAW-gegevens)
- http://ergast.com/mrd/ - Ergast Motor (F1) API
- http://www.omdbapi.com/ - Open Movie Database
- http://swapi.co/ - Star Wars API
- Zie ook `JavaScript APIs.txt` met meer voorbeelden

JSON Schema to typescript – server sided



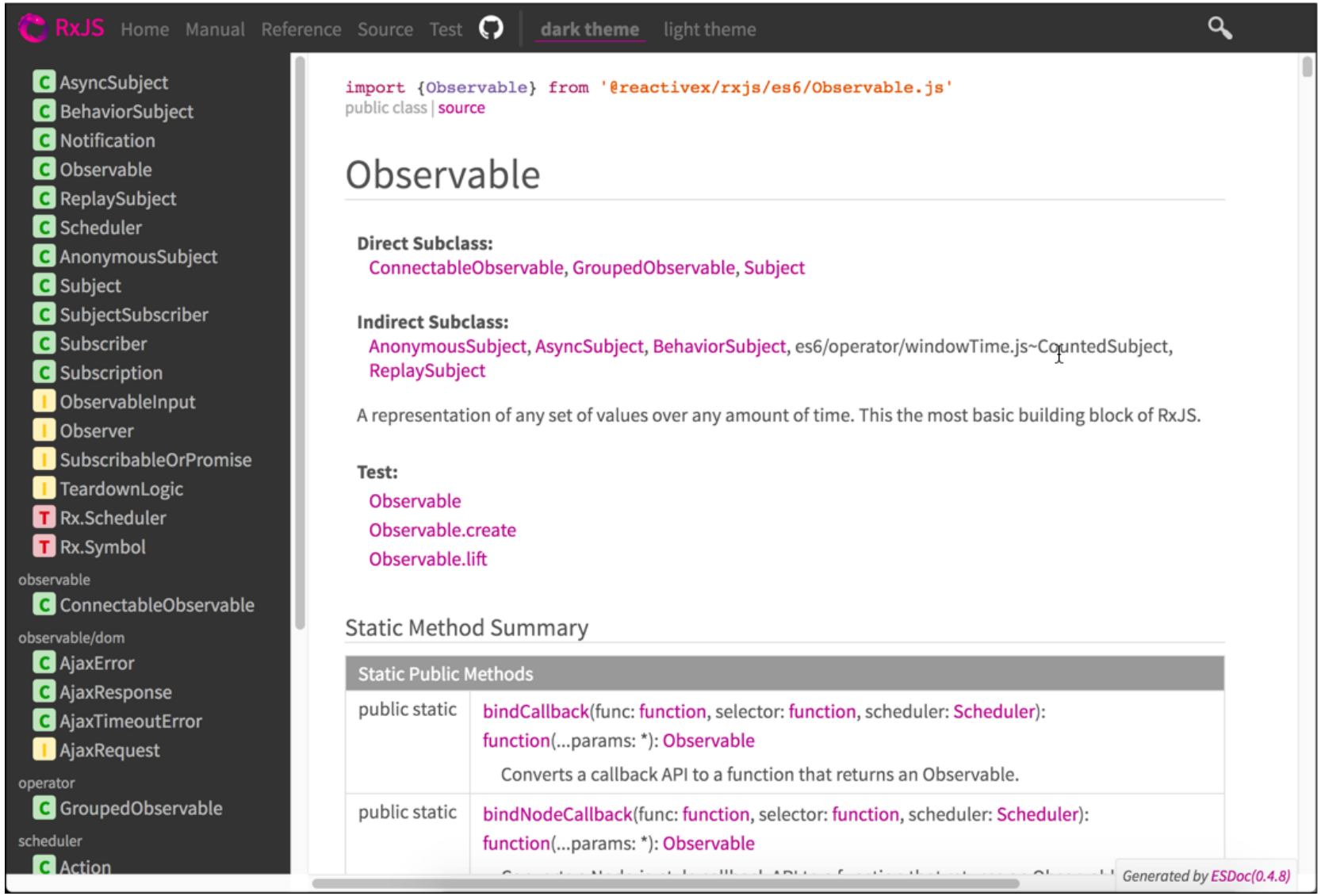
https://www.npmjs.com/package/json-schema-to-typescript

Data Mocken - Mockaroo



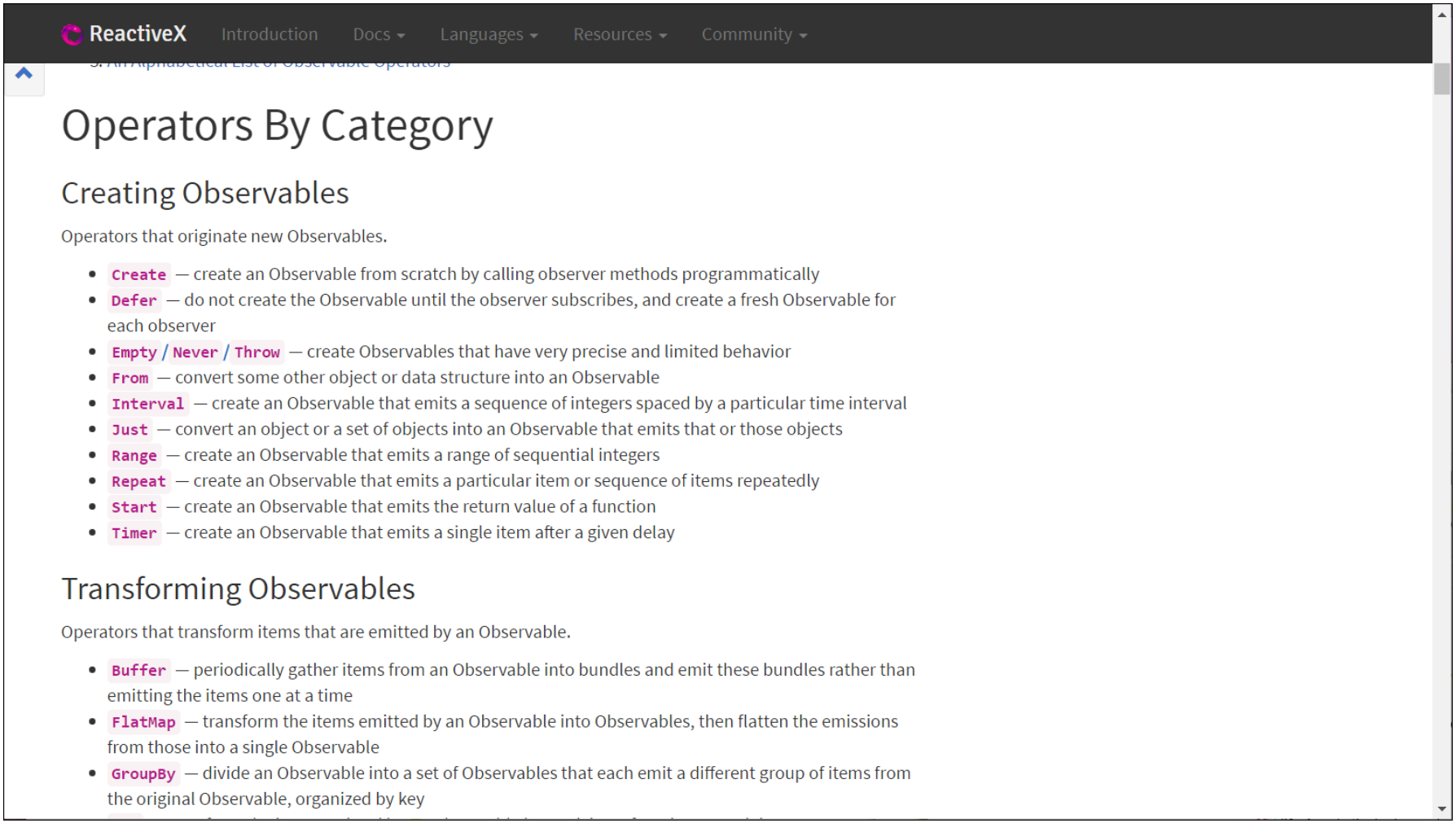
http://mockaroo.com/

Official documentation...



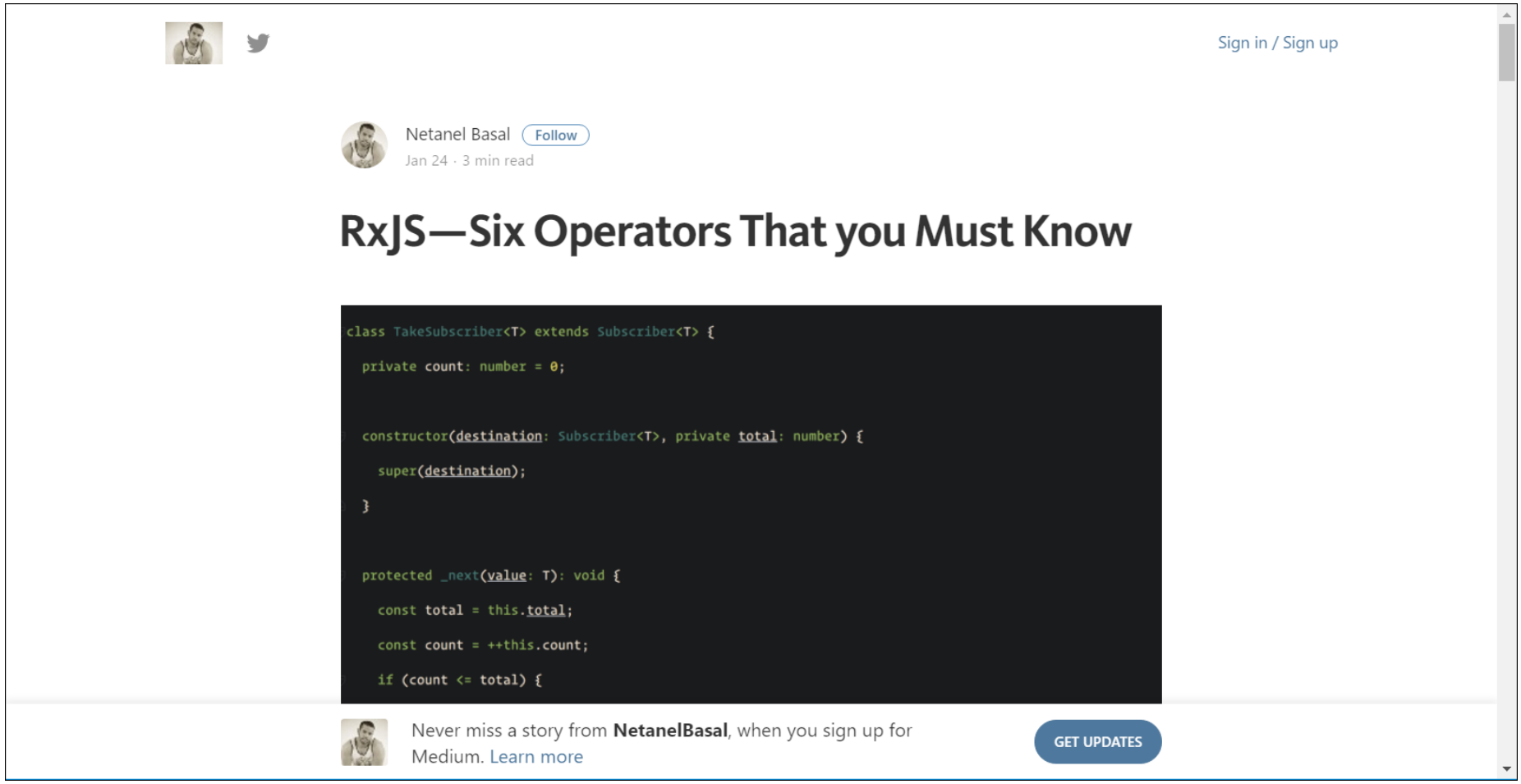
http://reactivex.io/rxjs/class/es6/Observable.js~Observable.html

More documentation



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

Article - 6 Operators you must know



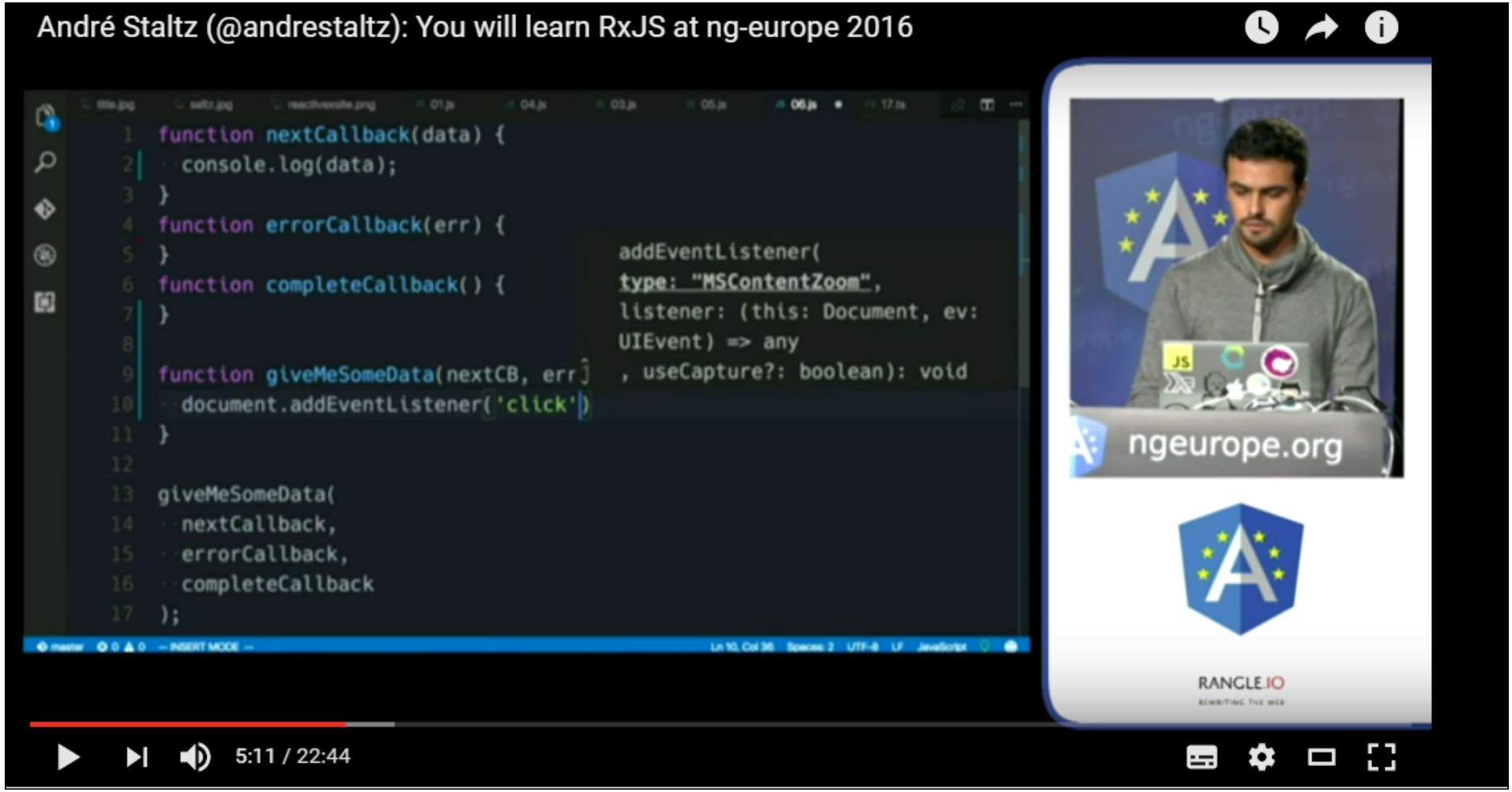
https://netbasal.com/rxjs-six-operators-that-you-must-know-5ed3b6e238a0#.11of73aox

Some Examples

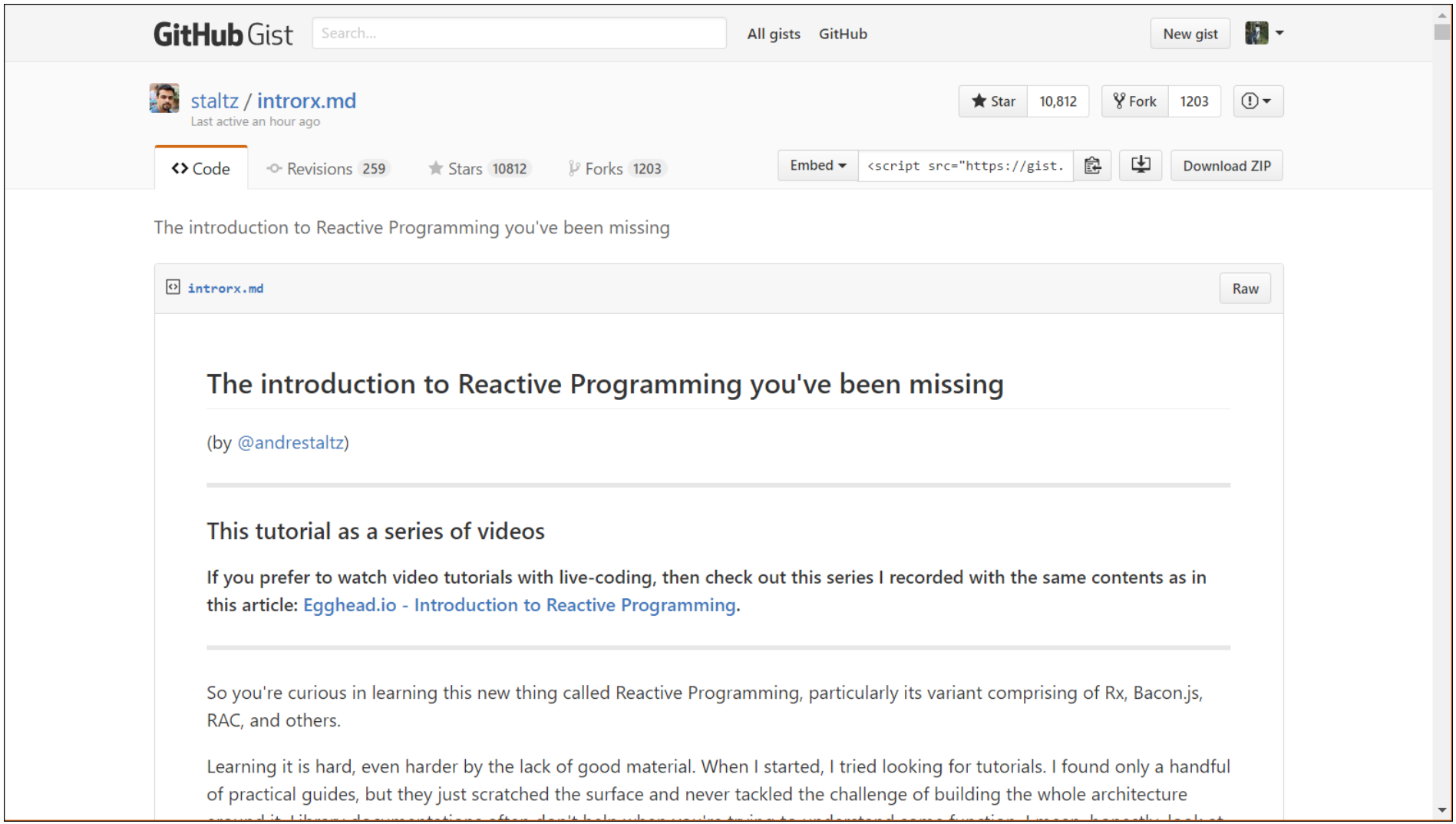
- /09a-services-live
- /09b-services-mapping
- /17c-forms-typeahead

```
this.videos = this.searchYouTube.valueChanges
    .debounceTime(600)                  // wait for 600ms to hit the API
    .map(query => makeURL(query))       // turn keyword into a real youtube-URL
    .switchMap(url => this.http.get(url))  // wait for, and switch to the Observable tha
switchmap)
    .map((res: Response) => res.json())    // map its response to json
    .map(response => response.items);     // unwrap the response and return only the ite
```

Creating Observables from scratch - André Staltz

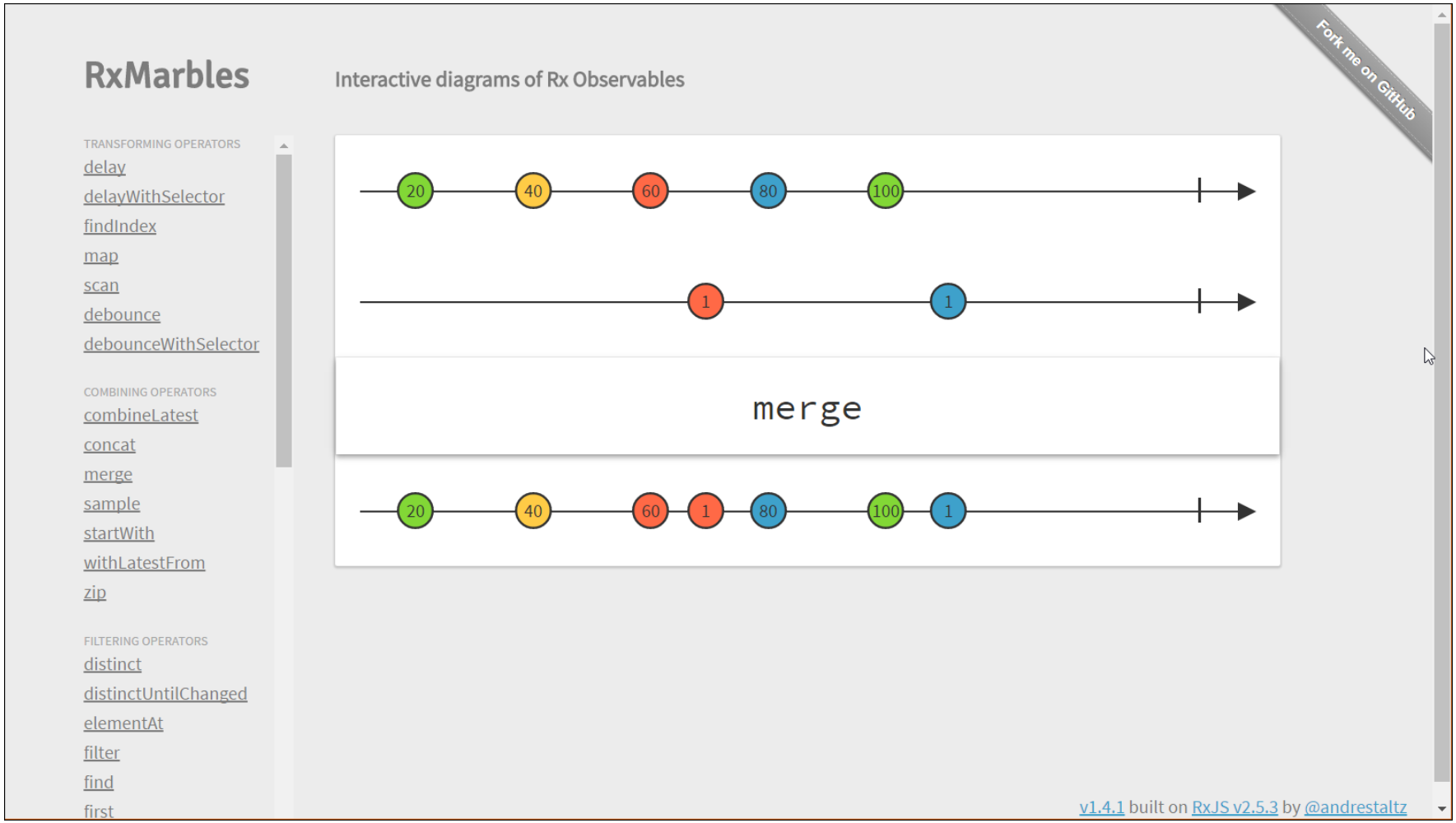


https://www.youtube.com/watch?v=uQ1zhJHclvs



https://gist.github.com/staltz/868e7e9bc2a7b8c1f754

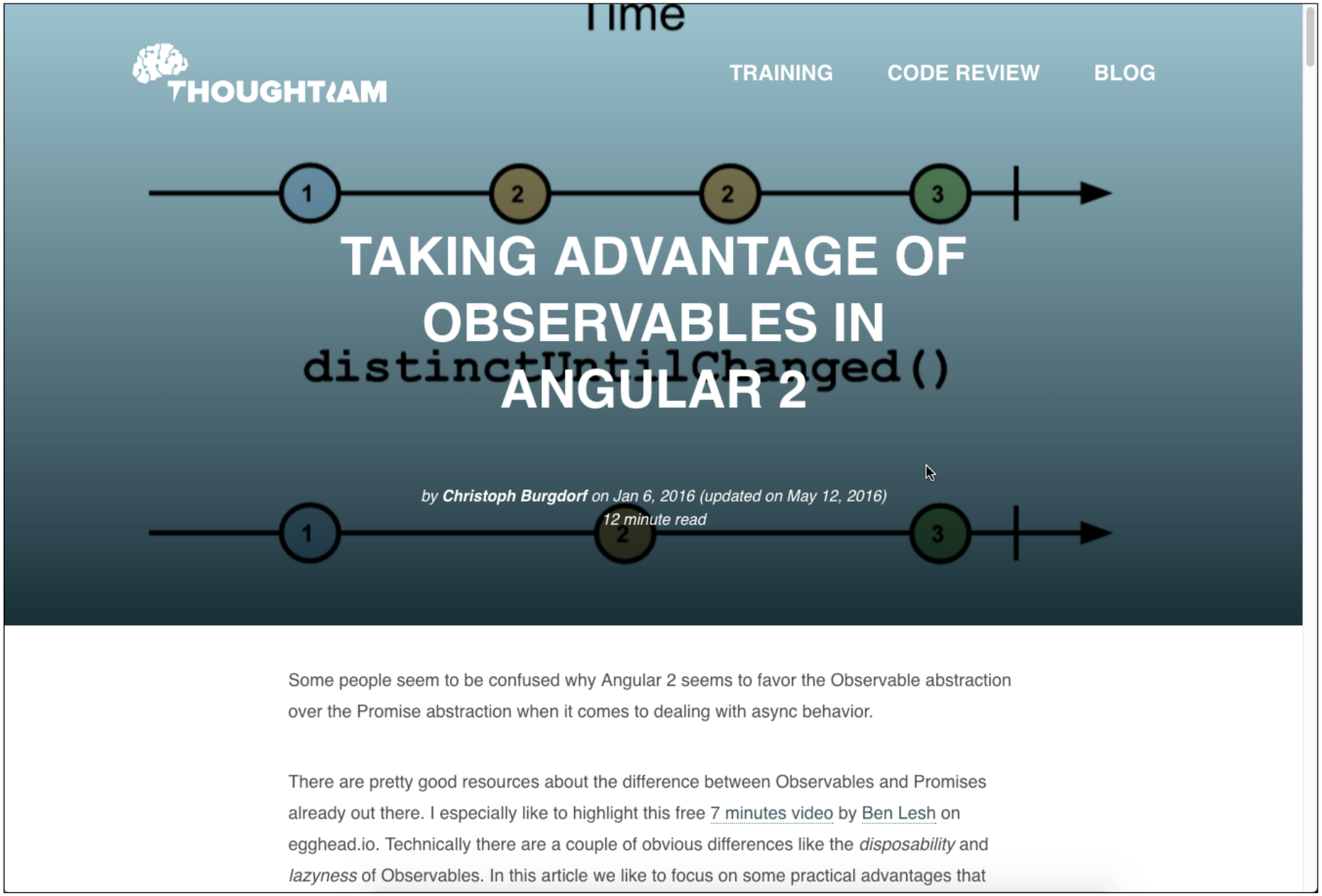
Also by Andre Stalz - RxMarbles



http://rxmarbles.com/

Dan Wahlin on Modules and Observables

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



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