aarixa

> ict and business optimization

Angular Fundamentals
Module 4 – Observables

Peter Kassenaar
info@kassenaar.com

Hoofdstuk 6

p. 138 en verder

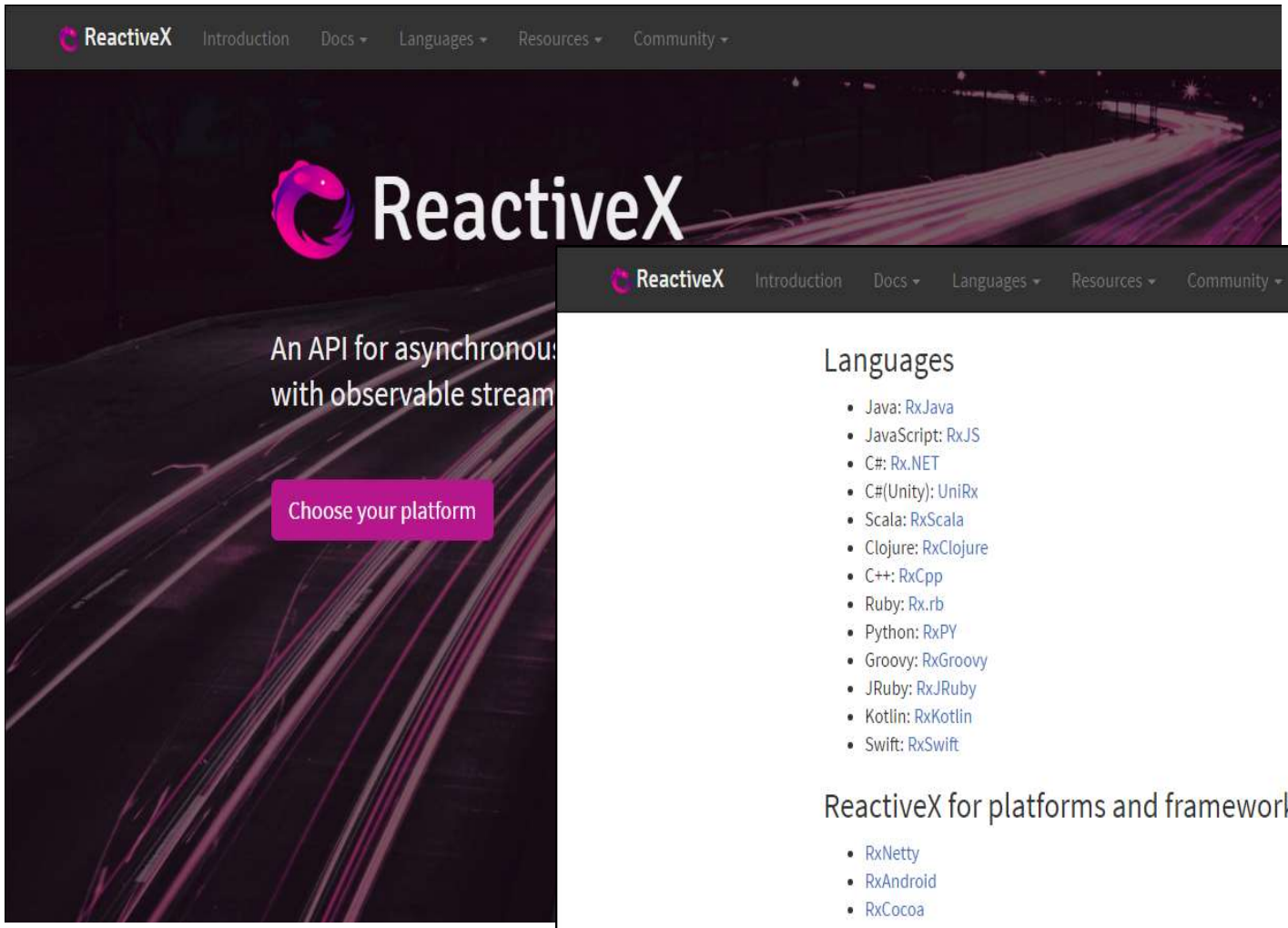
Async services met RxJS/Observables

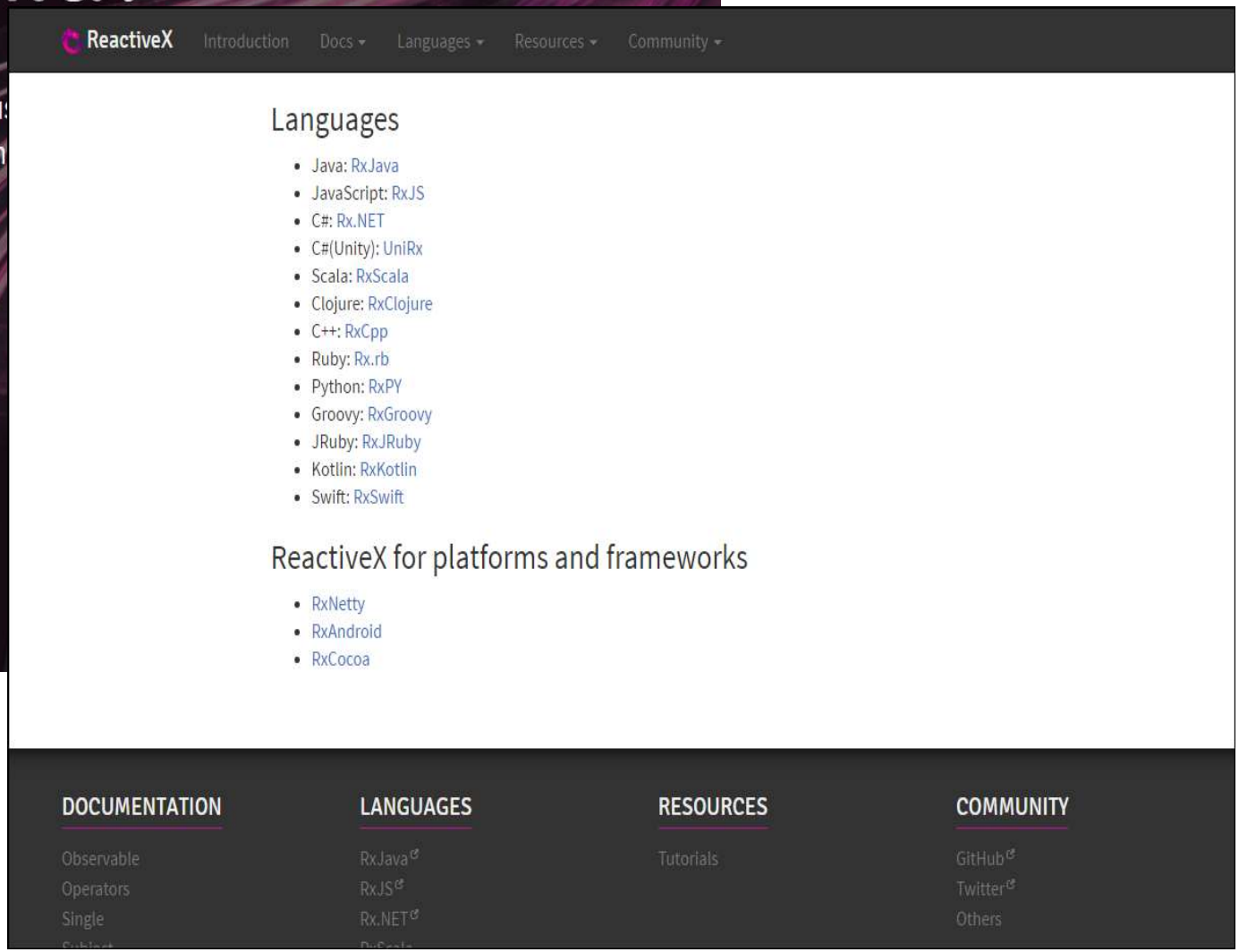
Reactive programming with asynchronous streams

Async Services

- Statische data ophalen: *synchrone* actie
- Data ophalen uit backend: *asynchrone* actie
- Andere frameworks : `Promises`
- Angular: `Observables`

Bovendien in Angular: 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.

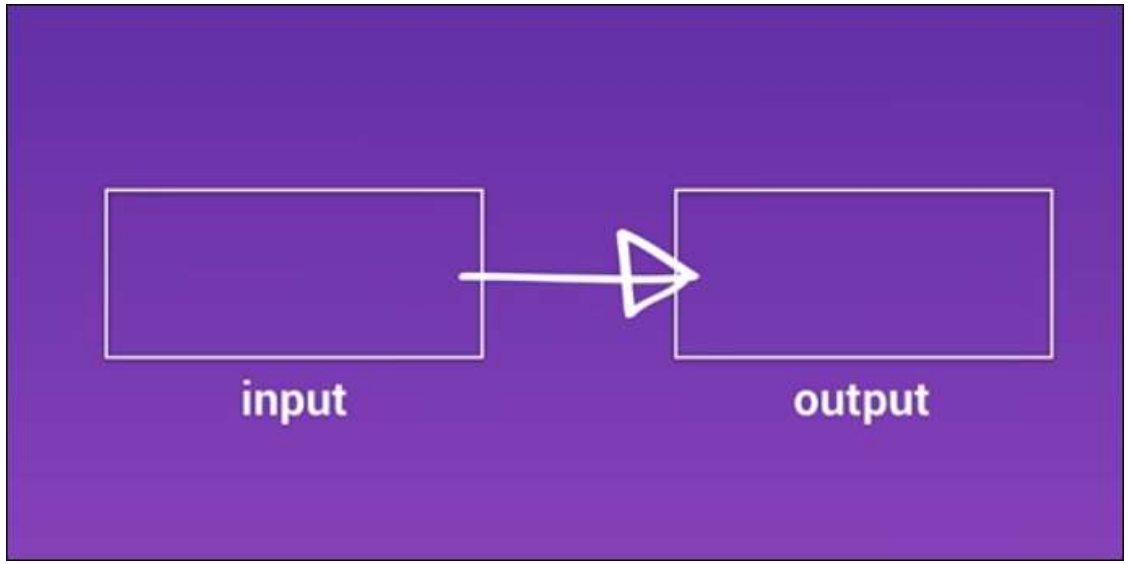
https://auth0.com/blog/2015/10/15/angular-2-series-part-3-using-http/

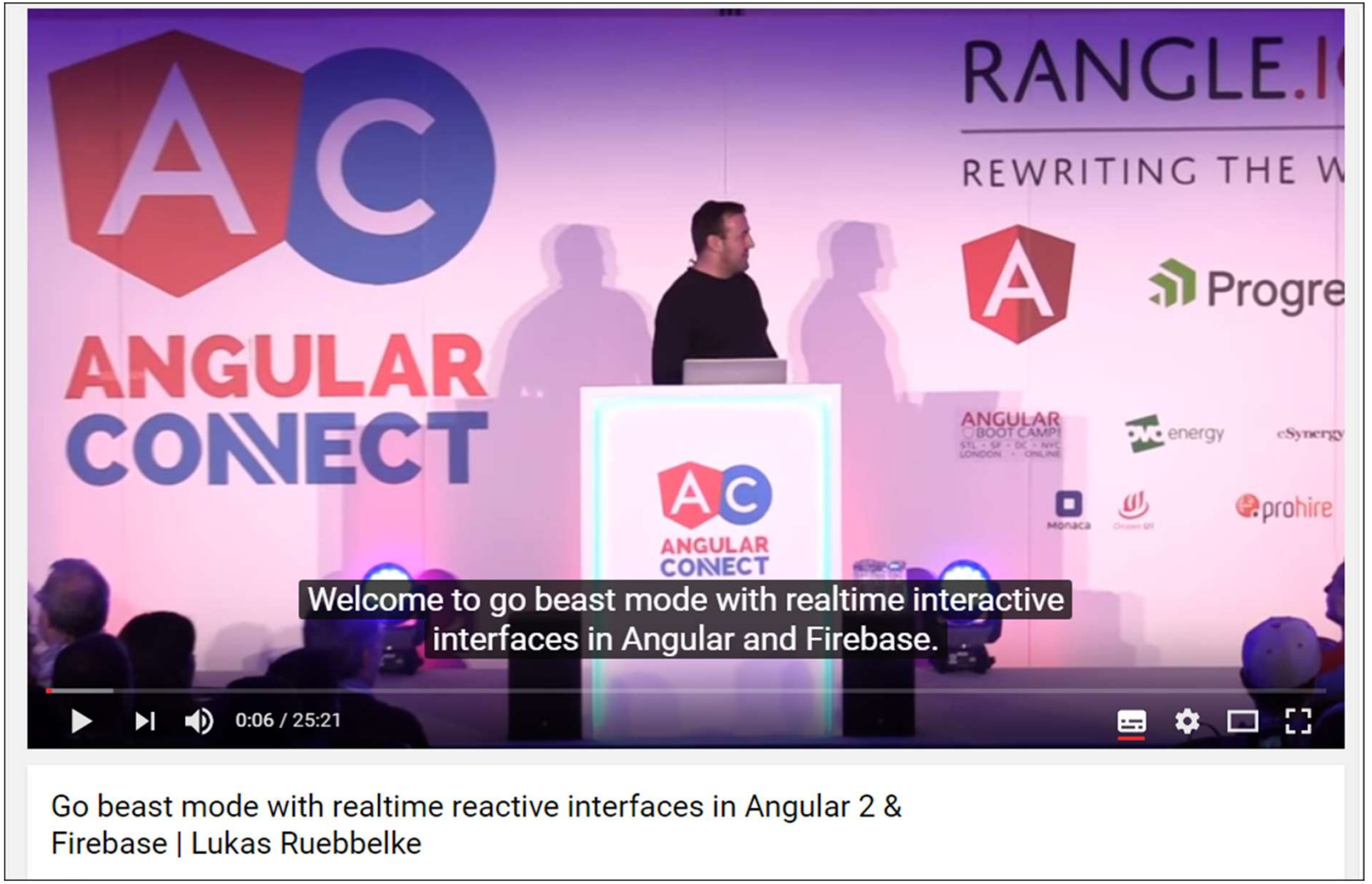
Observables en RxJs

- “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:

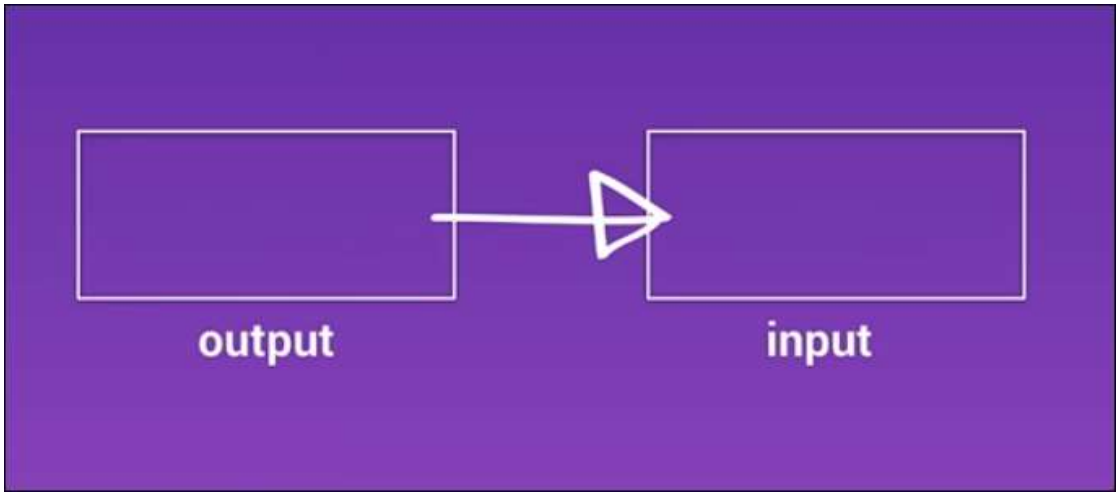




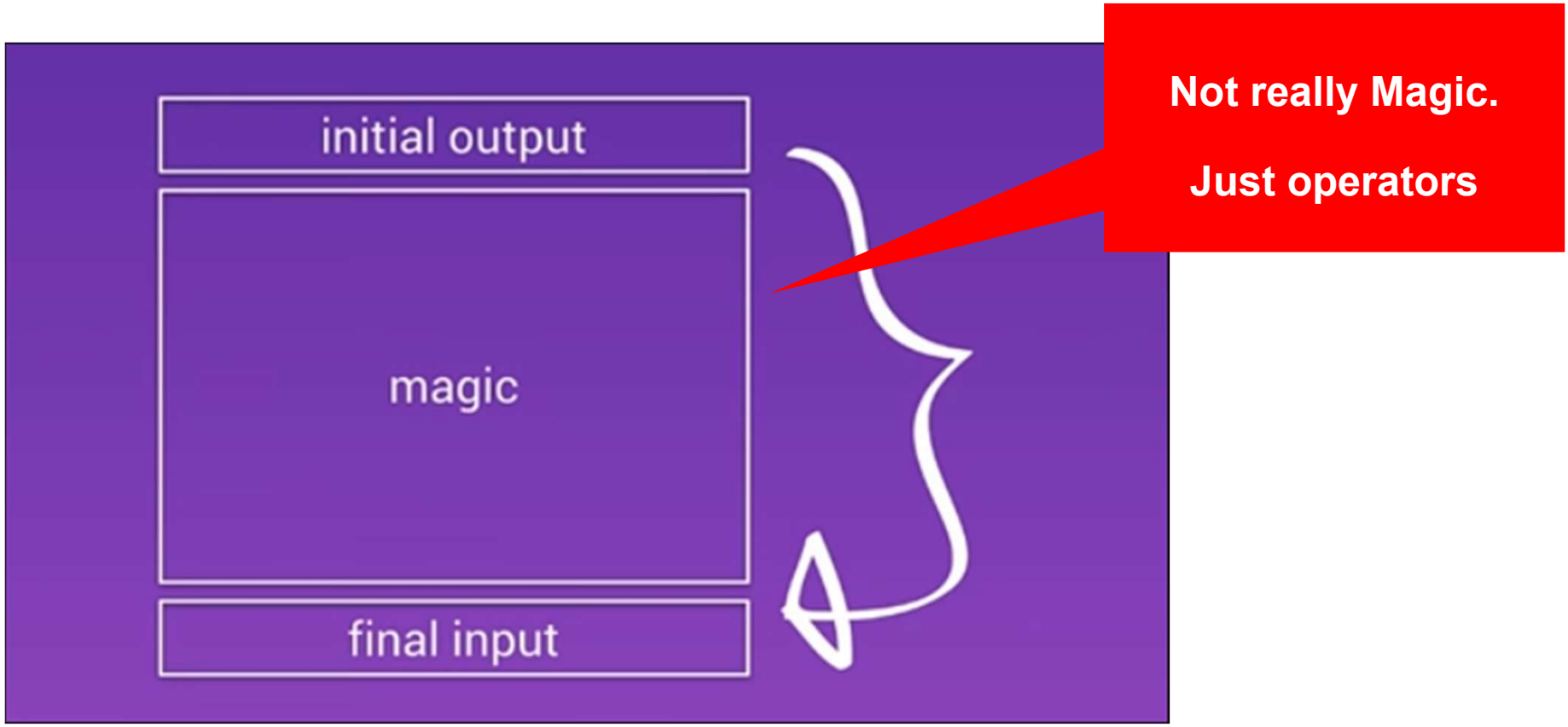
https://www.youtube.com/watch?v=5CTL7aqSvJU

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

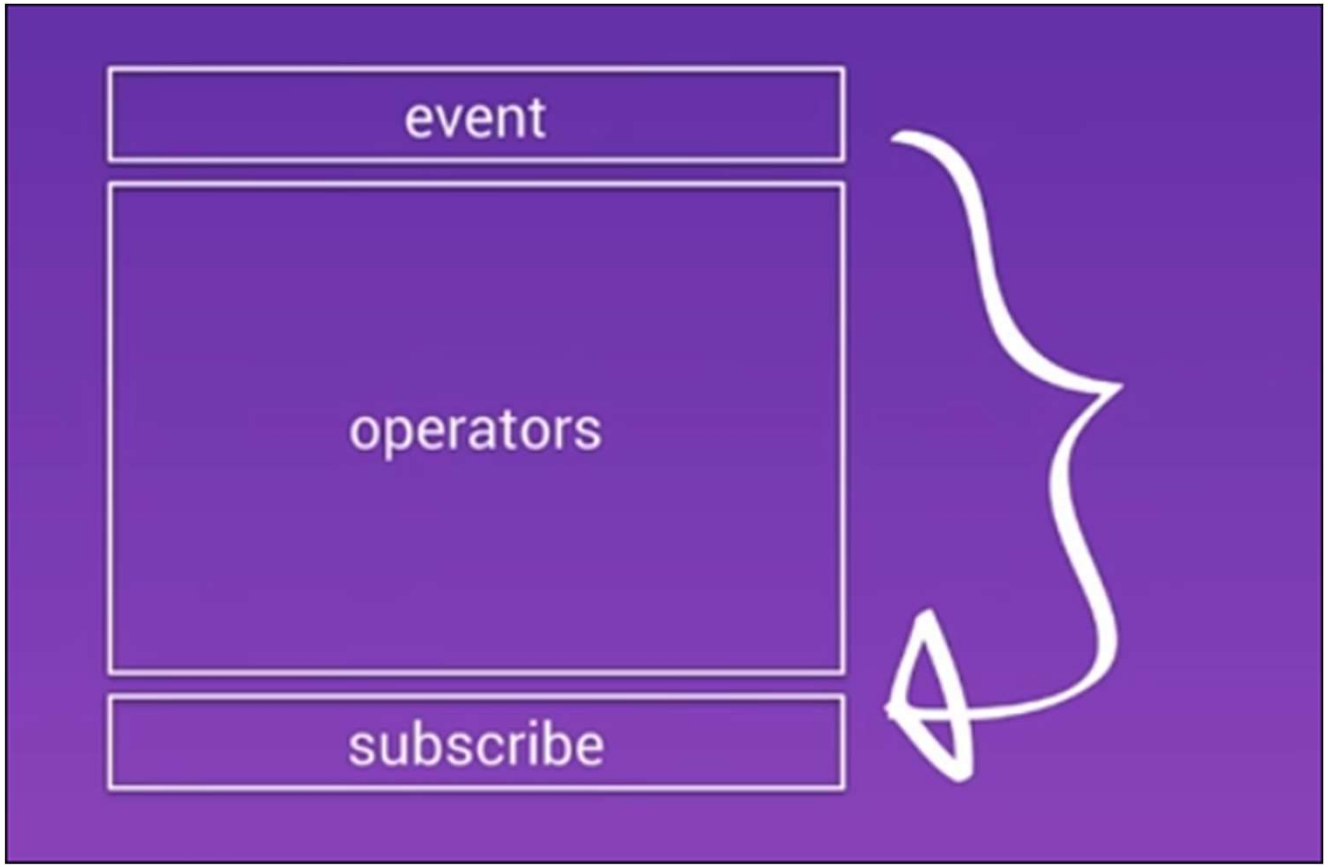
- With Observables -
  - a system, already outputting data,
  - Subscribe to that data
- "trade Output for Input"
- "Push vs. Pull"



"The observable sandwich"



Subscribe to events



In code:

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

  …

  })

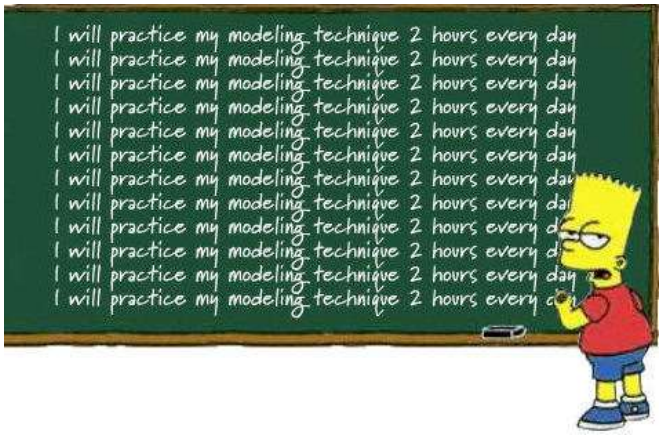
  export class AppModule {

  }

Oefening

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

Exercise….



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…)
)
```

More on subscriptions

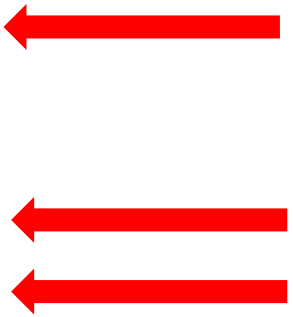
Using parameters inside the subscriber

Subscribe - only once per block!

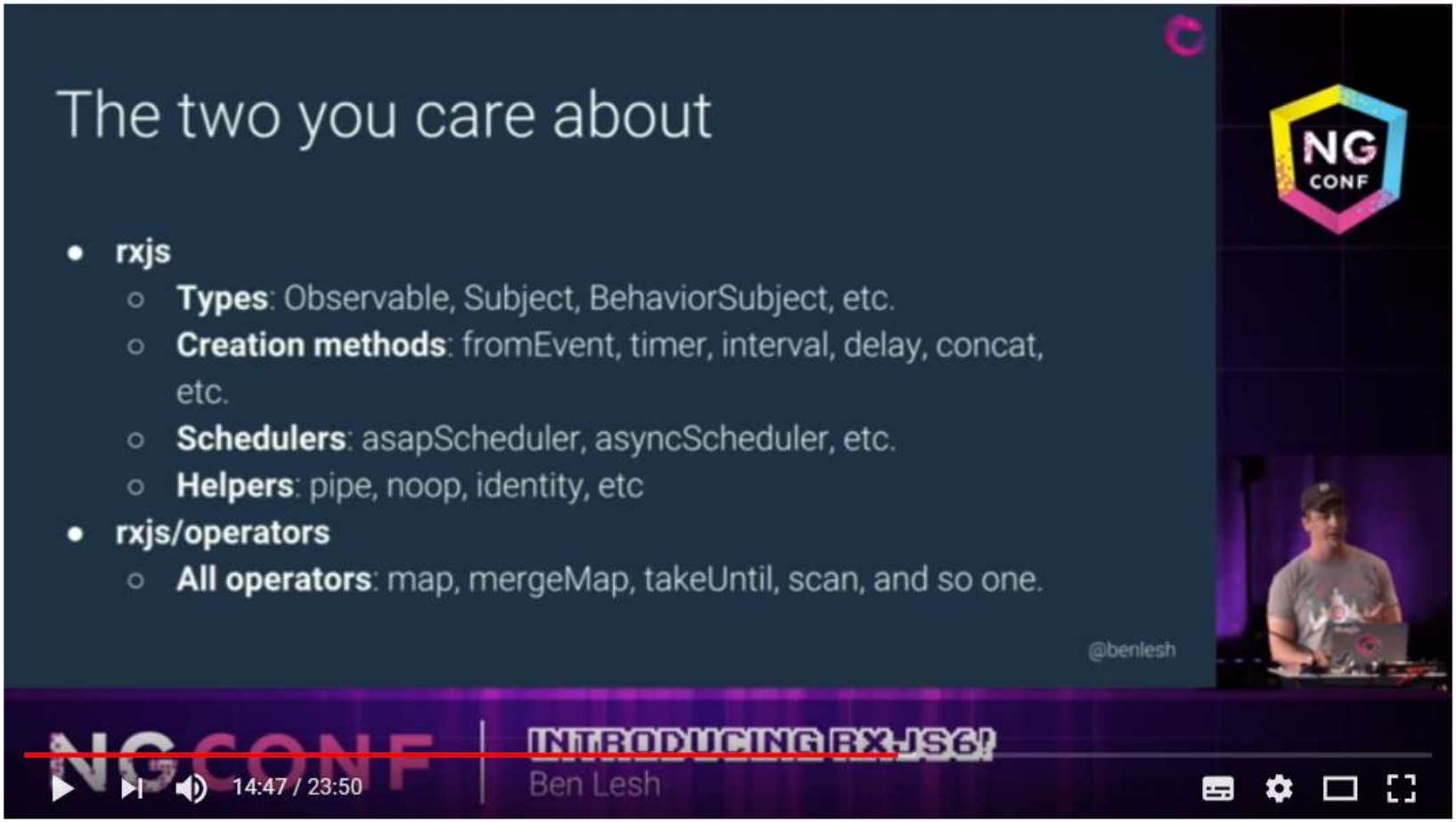
- Three parameters:
  - success()
  - error() – Optioneel!
  - complete() – Optioneel!

```
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



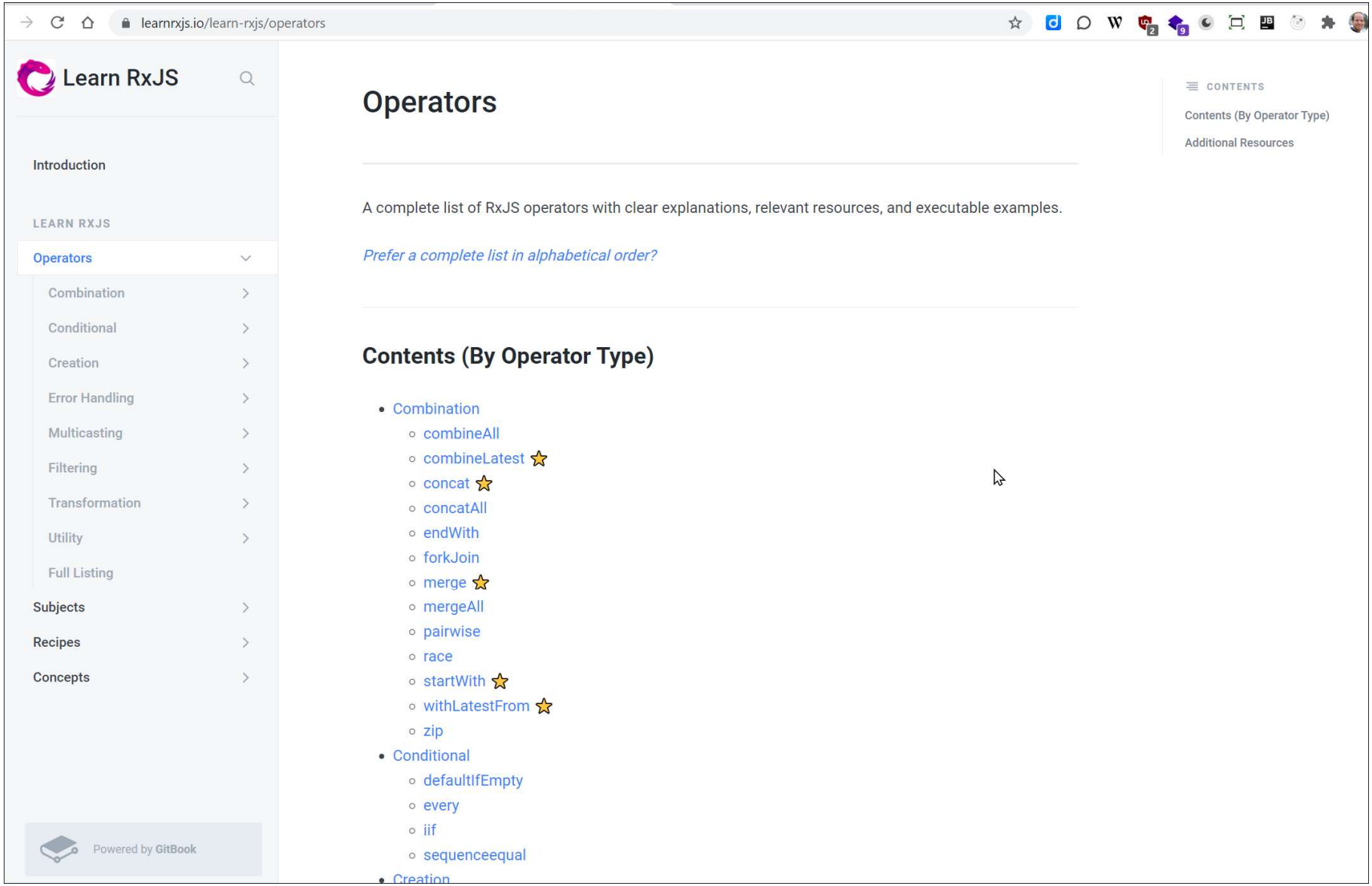
Introducing RxJS6! - Ben Lesh

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/



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

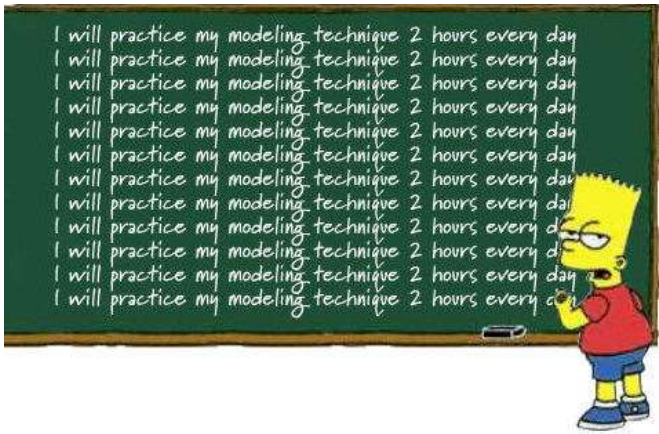
- MovieApp
- `examples\210-services-live`

Voorbeeld API's

- https://pokeapi.co/ - Pokemon API
- http://openweathermap.org/API (weerbericht)
- https://jsonplaceholder.typicode.com/ random users, posts, photos
- http://ergast.com/mrd/ - Ergast Motor (F1) API
- http://www.omdbapi.com/ - Open Movie Database
- https://swapi.dev/ - Star Wars API
- Zie ook `JavaScript APIs.txt` met meer voorbeelden

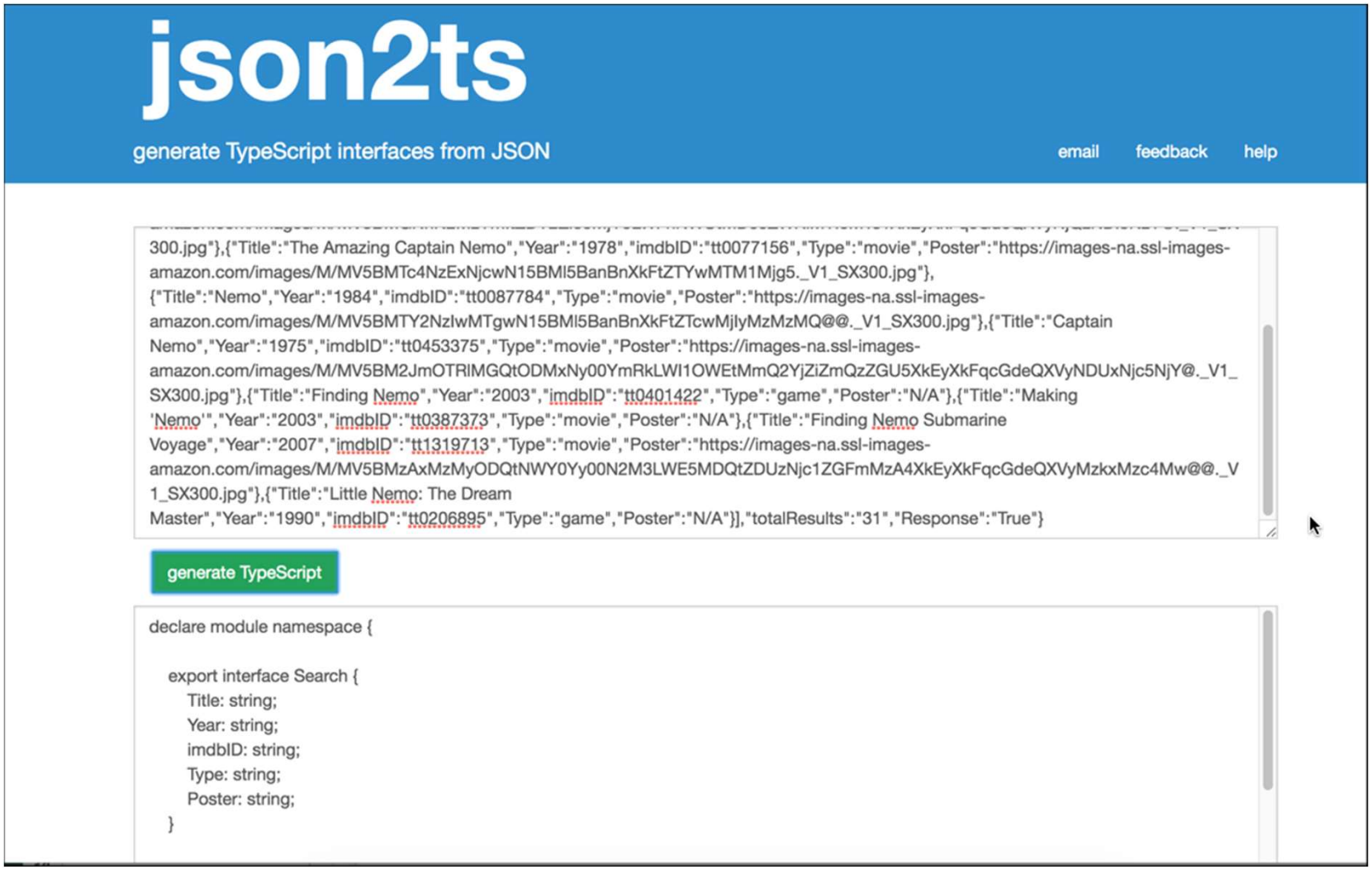
Workshop

- 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[f])



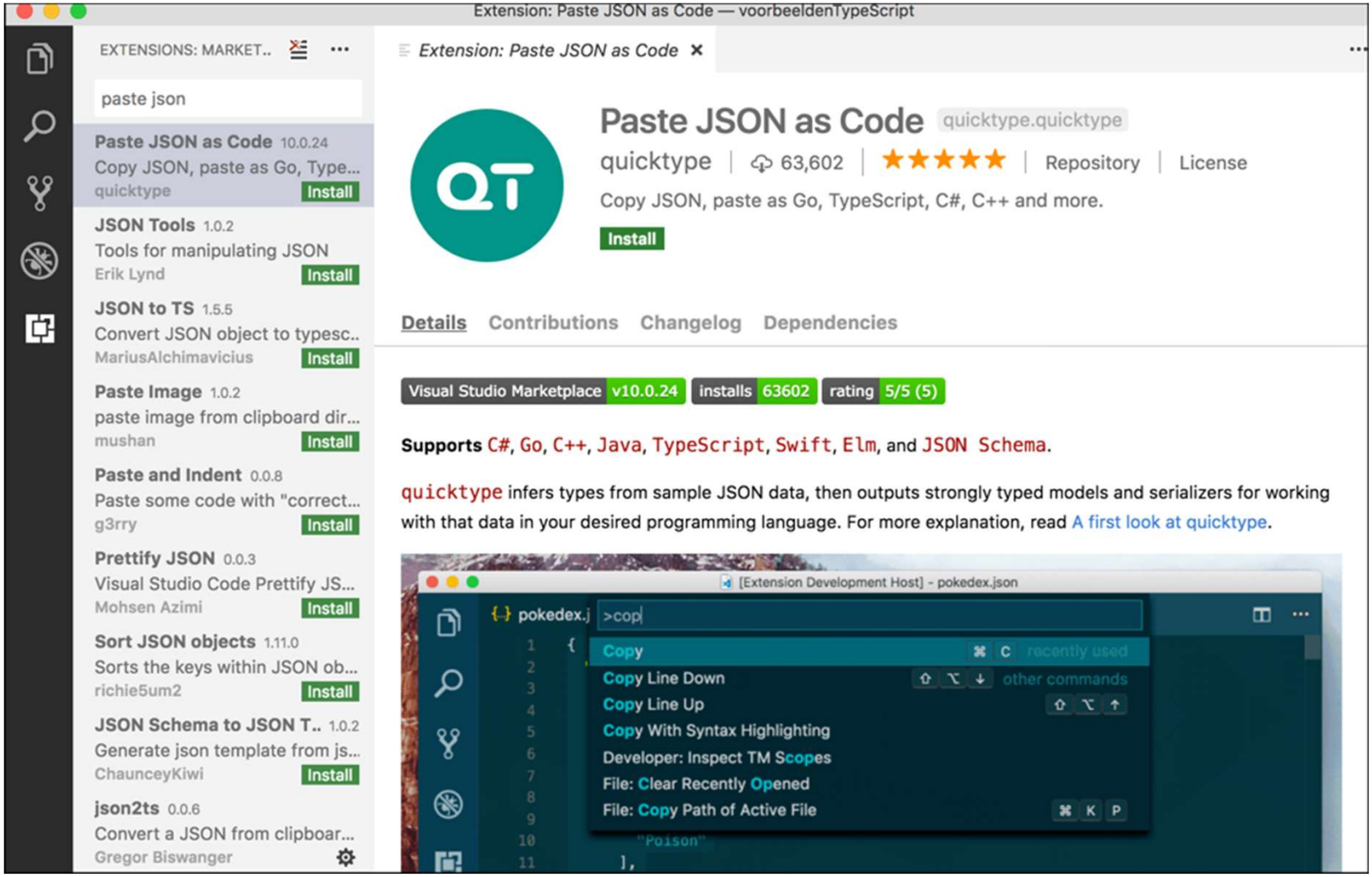
More info on observables

Online JSON to TypeScript converter



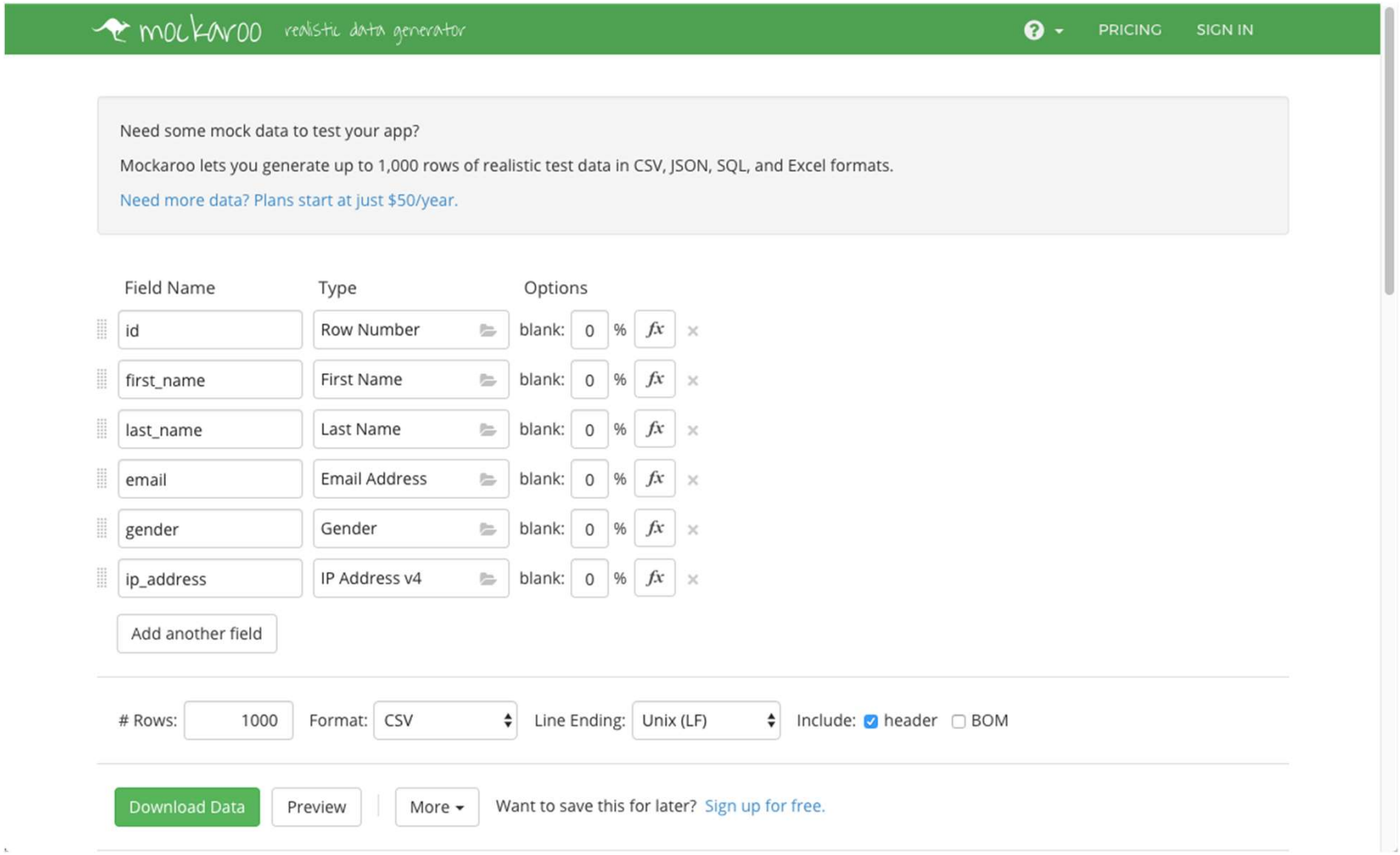
http://json2ts.com/

In VS Code? Use this extension!



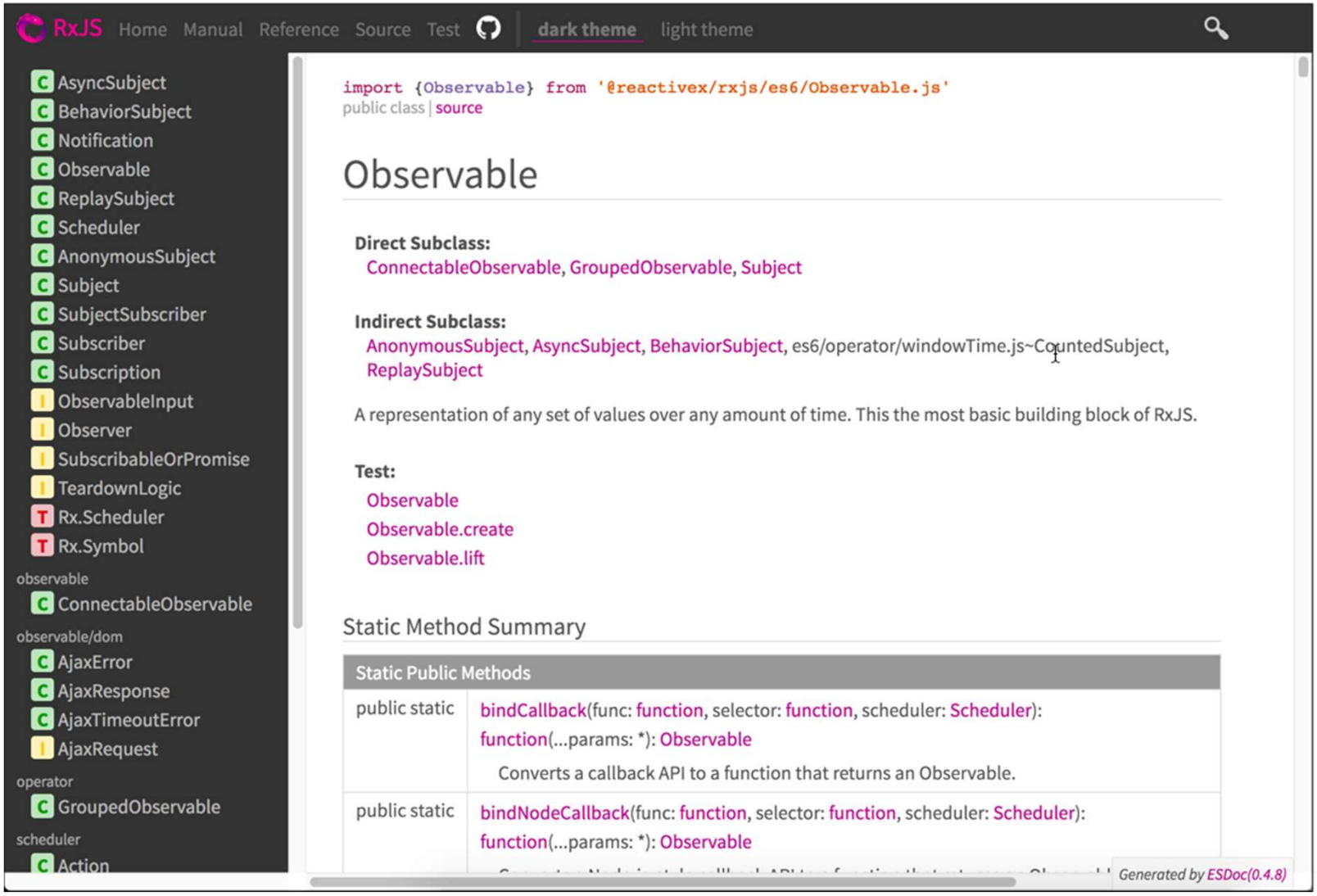
https://marketplace.visualstudio.com/items?itemName=quicktype.quicktype

Data Mocken - Mockaroo



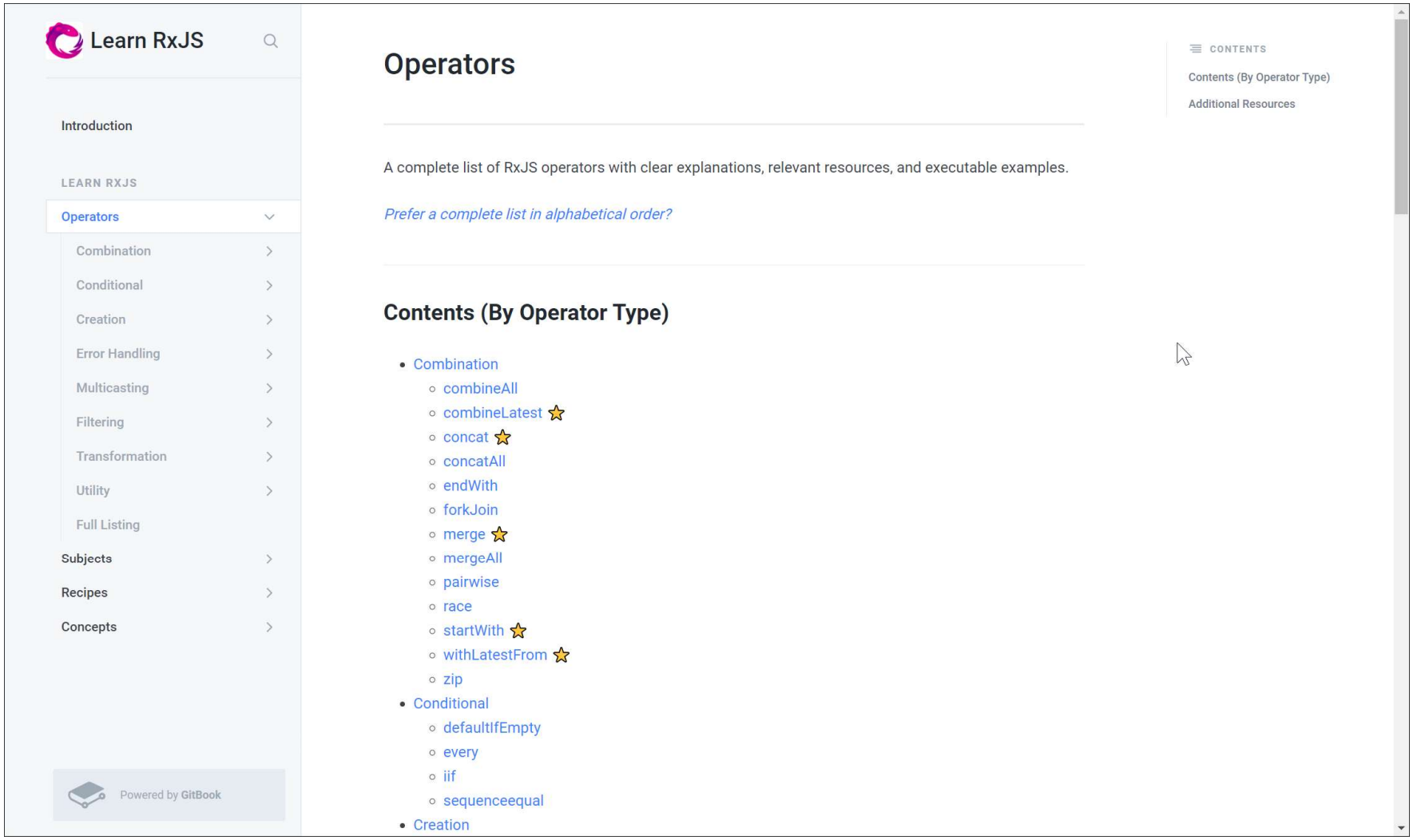
http://mockaroo.com/

Official documentation...

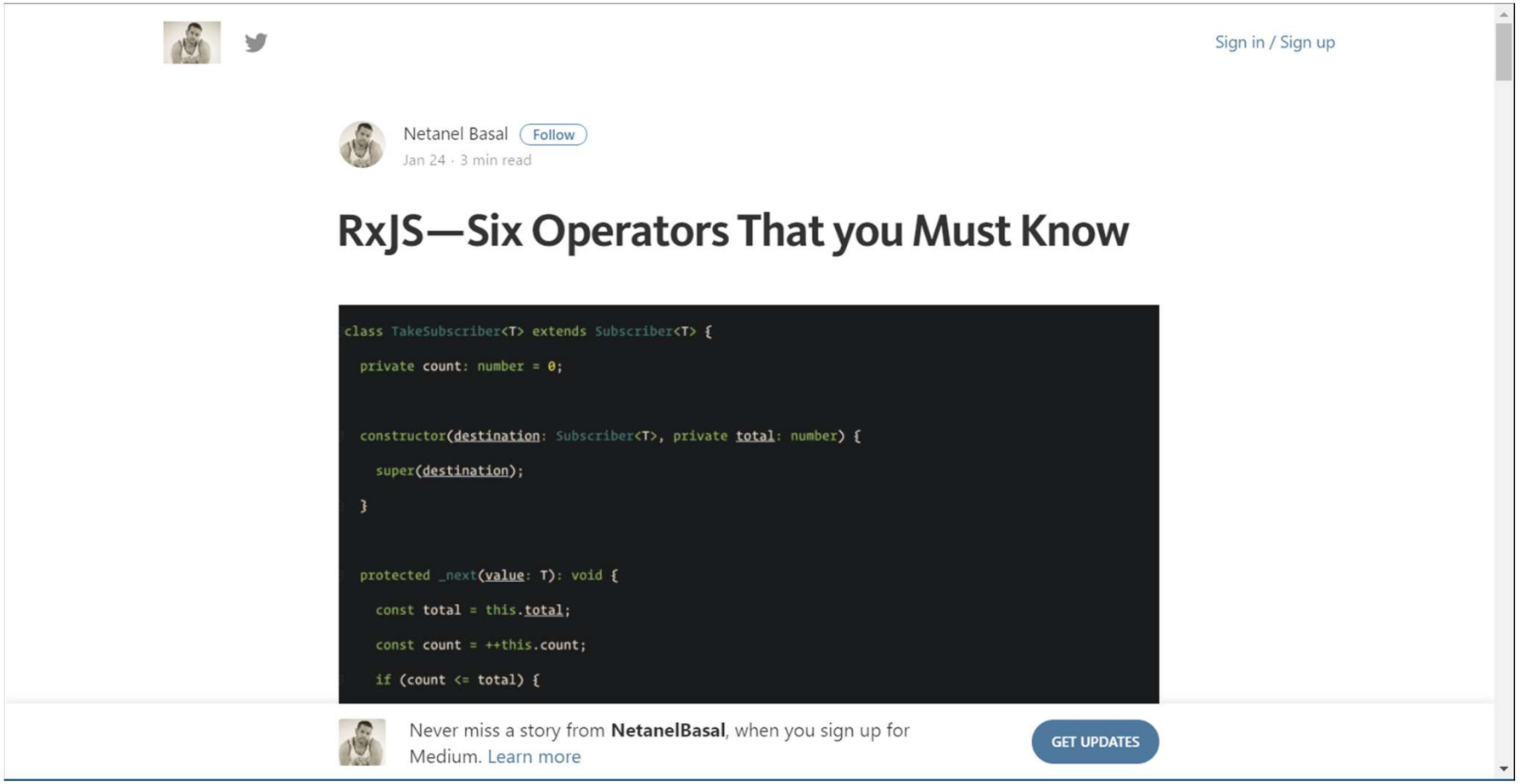


http://reactivex.io/rxjs/class/es6/Observable.js~Observable.html

https://www.learnrxjs.io/

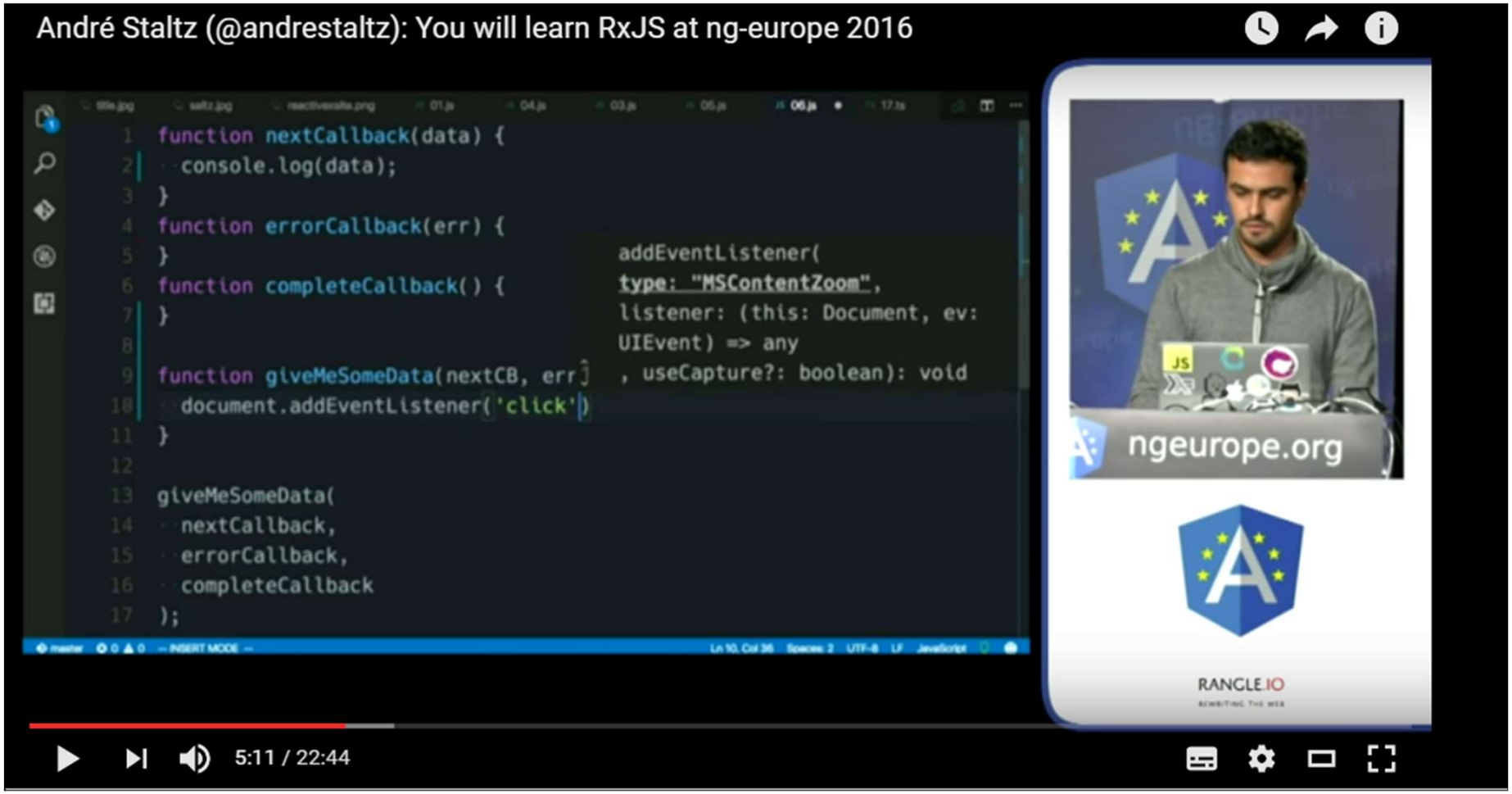


Article - 6 Operators you must know

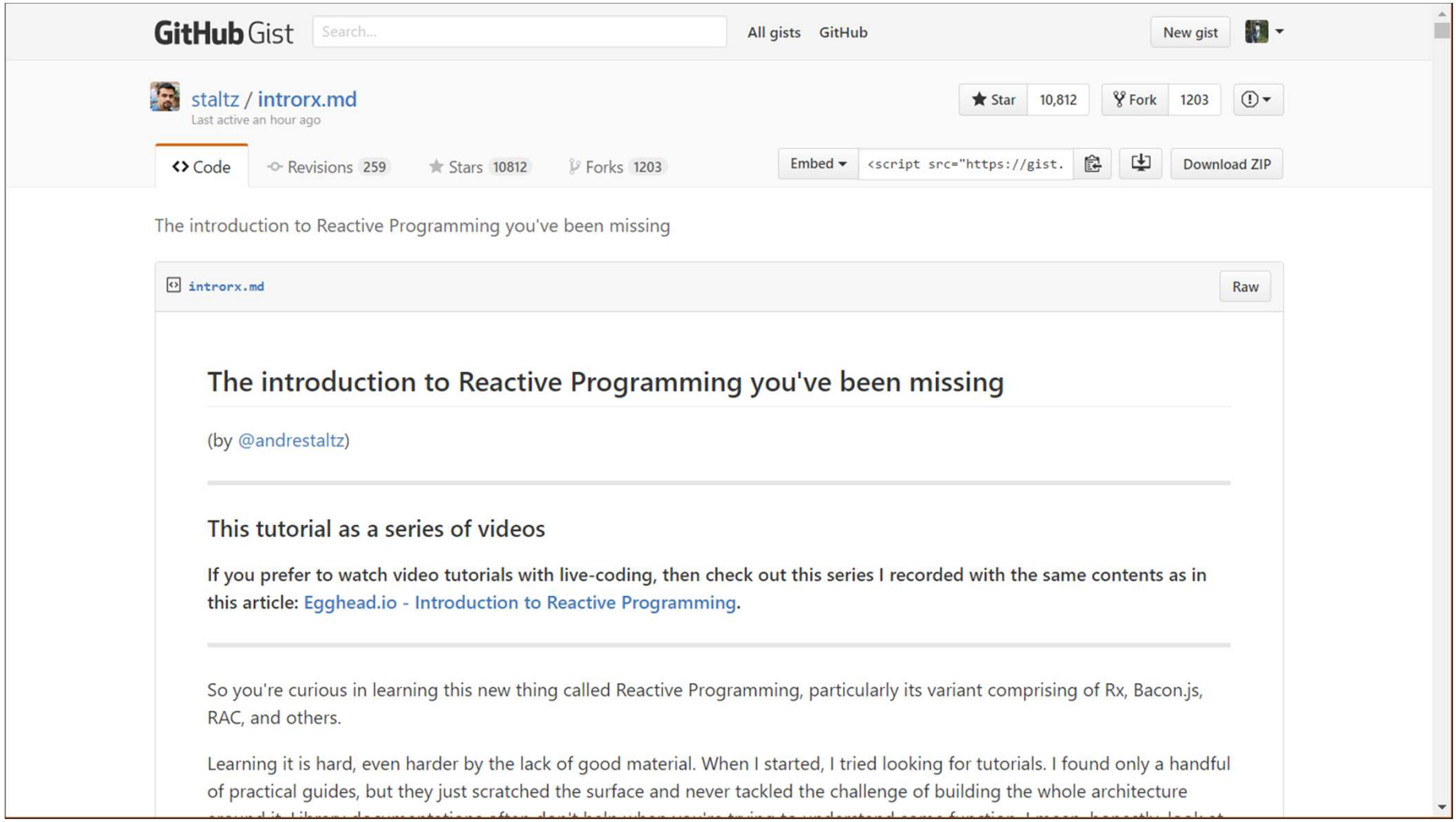


https://netbasal.com/rxjs-six-operators-that-you-must-know-5ed3b6e238a0#.11of73aox

Creating Observables from scratch - André Staltz

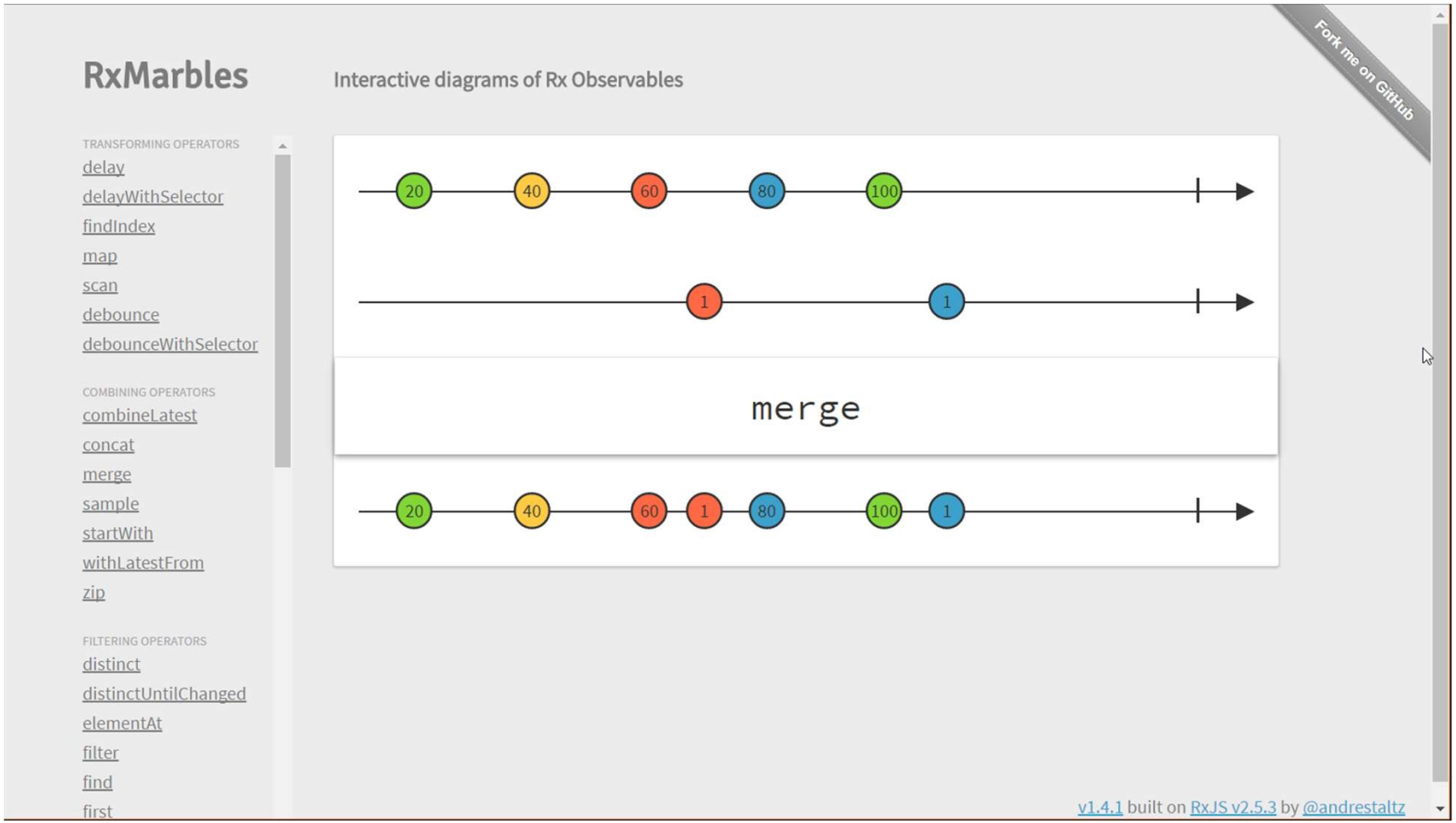


https://www.youtube.com/watch?v=uQ1zhJHclvs



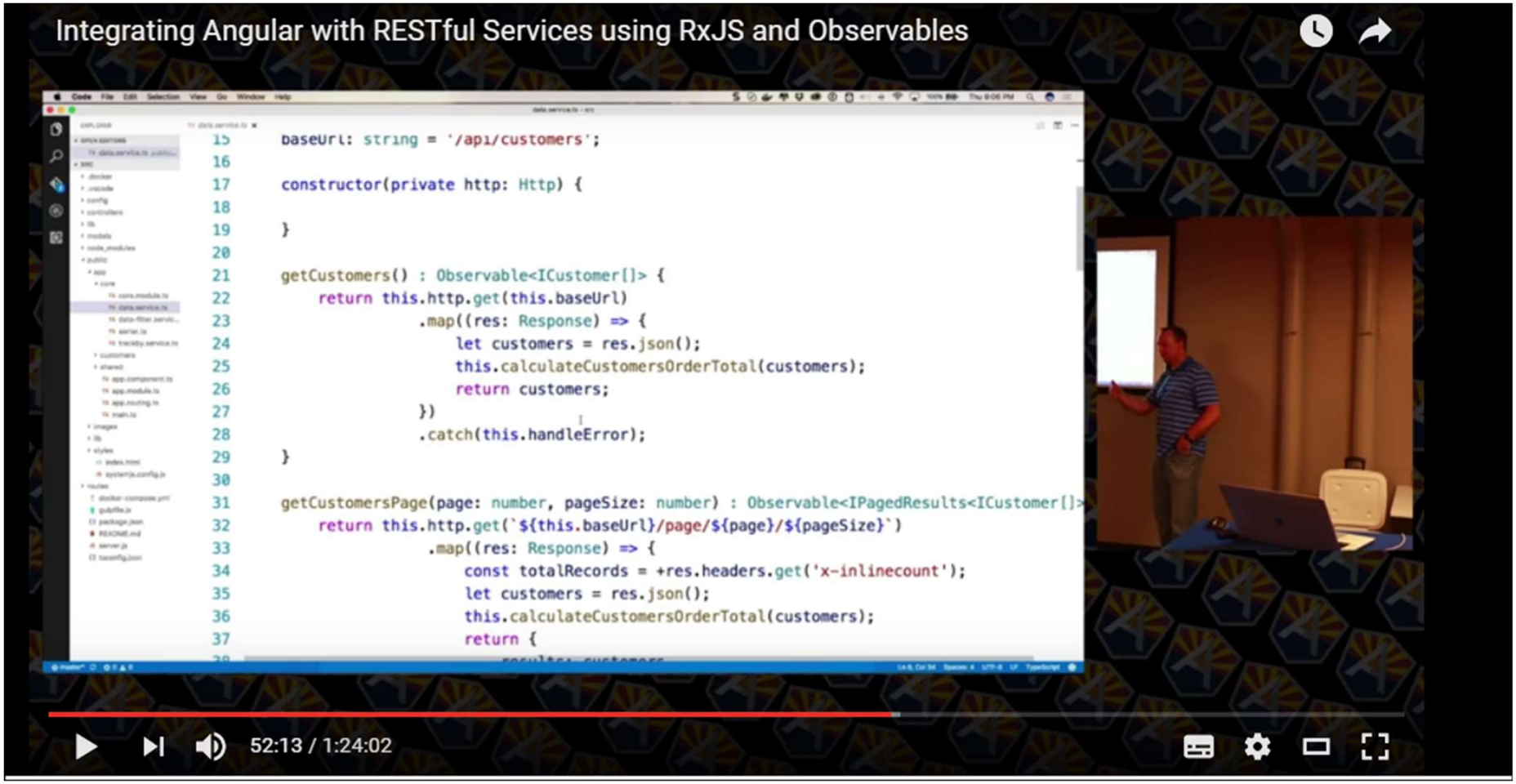
https://gist.github.com/staltz/868e7e9bc2a7b8c1f754

Also by Andre Stalz - RxMarbles

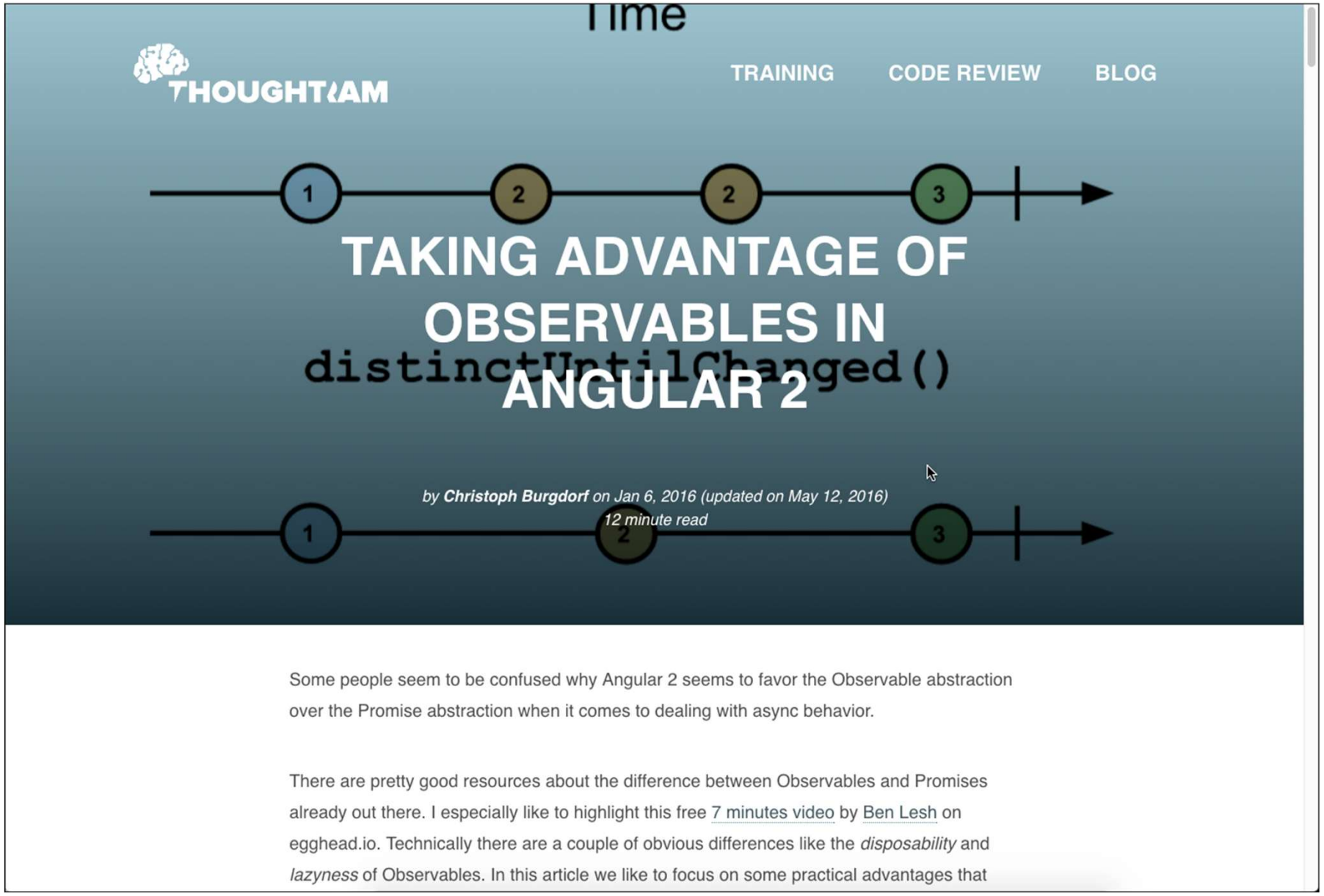


http://rxmarbles.com/

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

Een collectie observables ophalen

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