SpiralTrain
developer courses

Angular Fundamentals Module – Observables

Peter Kassenaar –
info@kassenaar.com

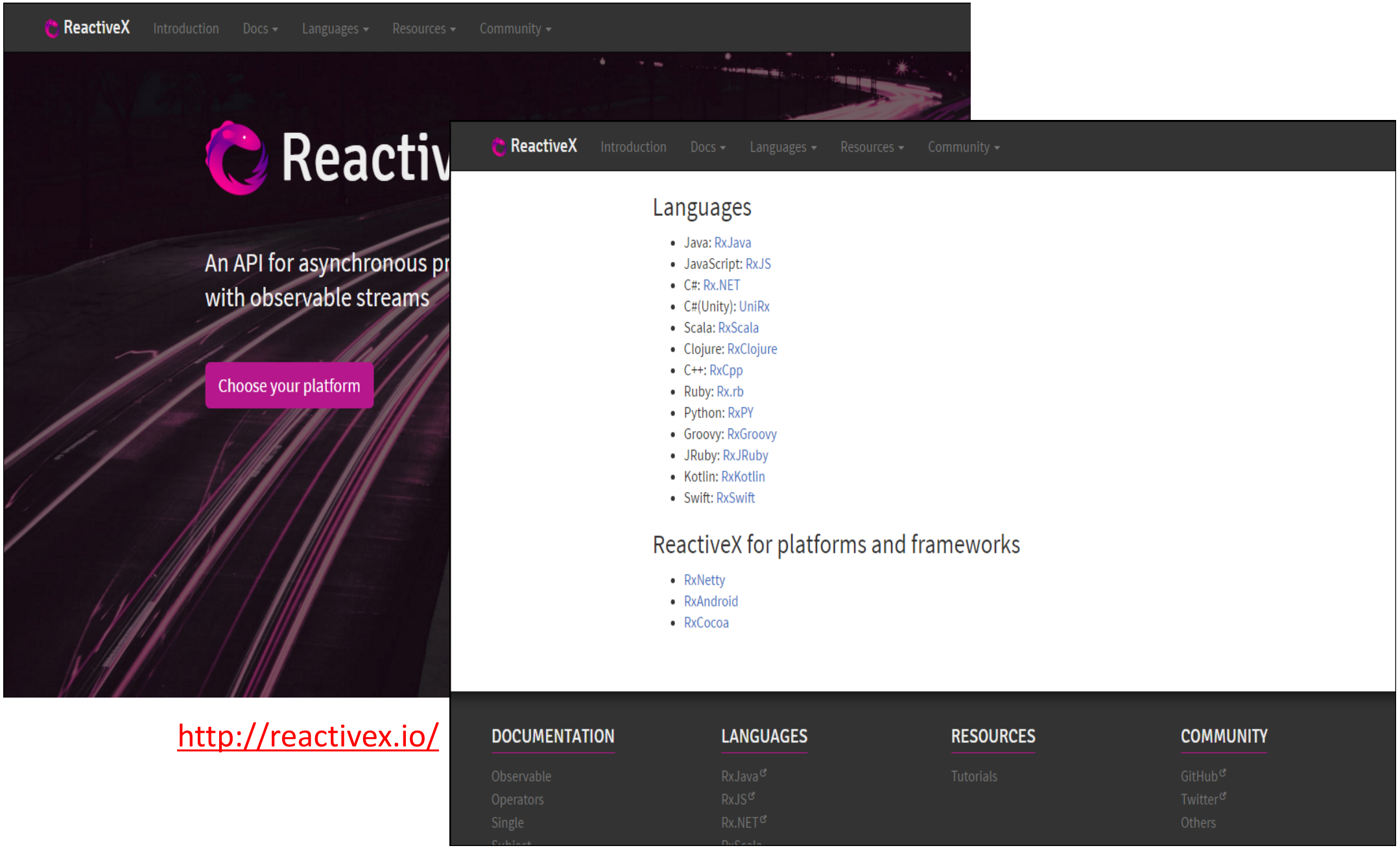
Async services with RxJS / Observables

Reactive programming
with asynchronous streams

Async Services

- Fetching static data: *synchronous* action
- Working via `HttpClient`: *asynchronous* action
- Angular 1: `Promises`
- Angular 2: `Observables`

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.

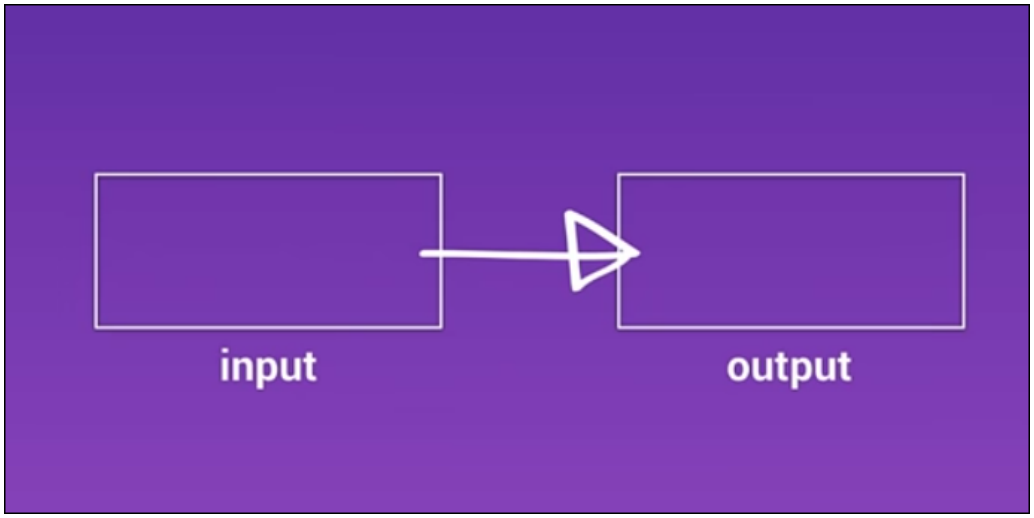
https://auth0.com/blog/2015/10/15/angular-2-series-part-3-using-http/

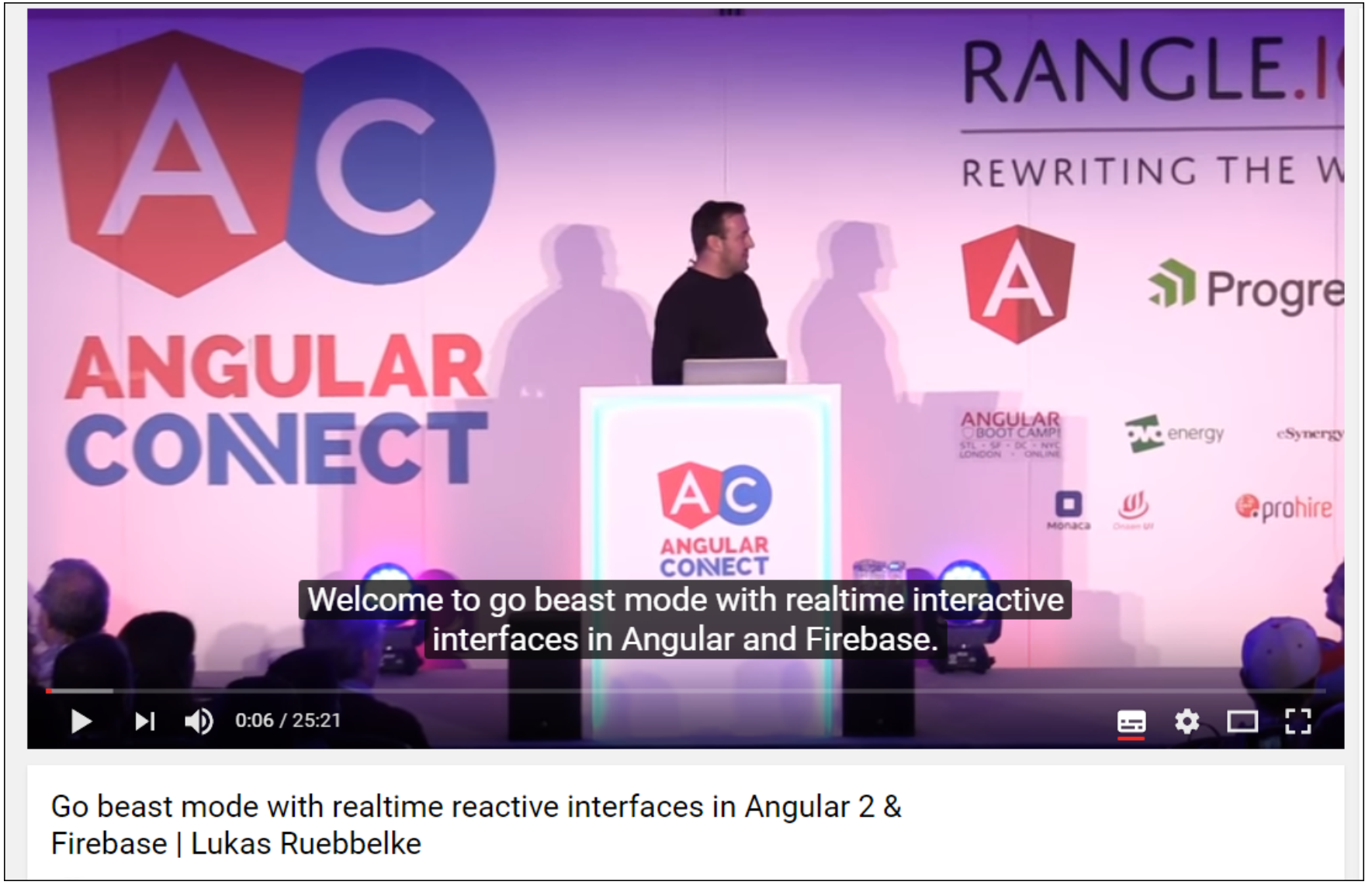
Observables and RxJs

- “Reactive Programming”
  - *“Reactive programming is programming with asynchronous data streams.”*
  - https://gist.github.com/staltz/868e7e9bc2a7b8c1f754
- Observables have a lot of extra options, as opposed to Promises
  - Mapping
  - Filtering
  - Combining
  - Cancel
  - Retry
  - …
- Consequence: no more `.success()`, `.error()` and `.then()` chaining!

How do observables work

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

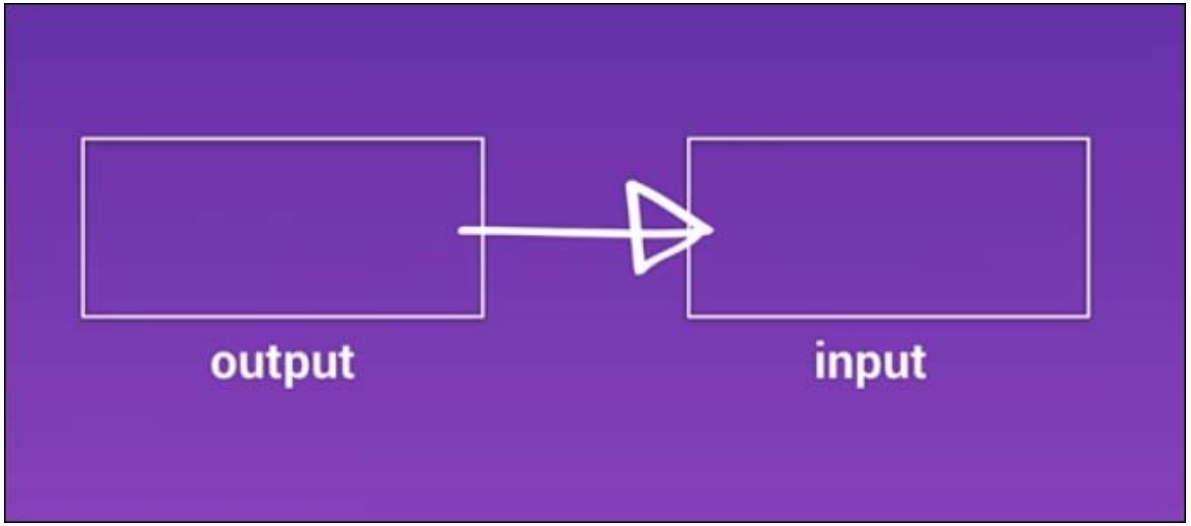




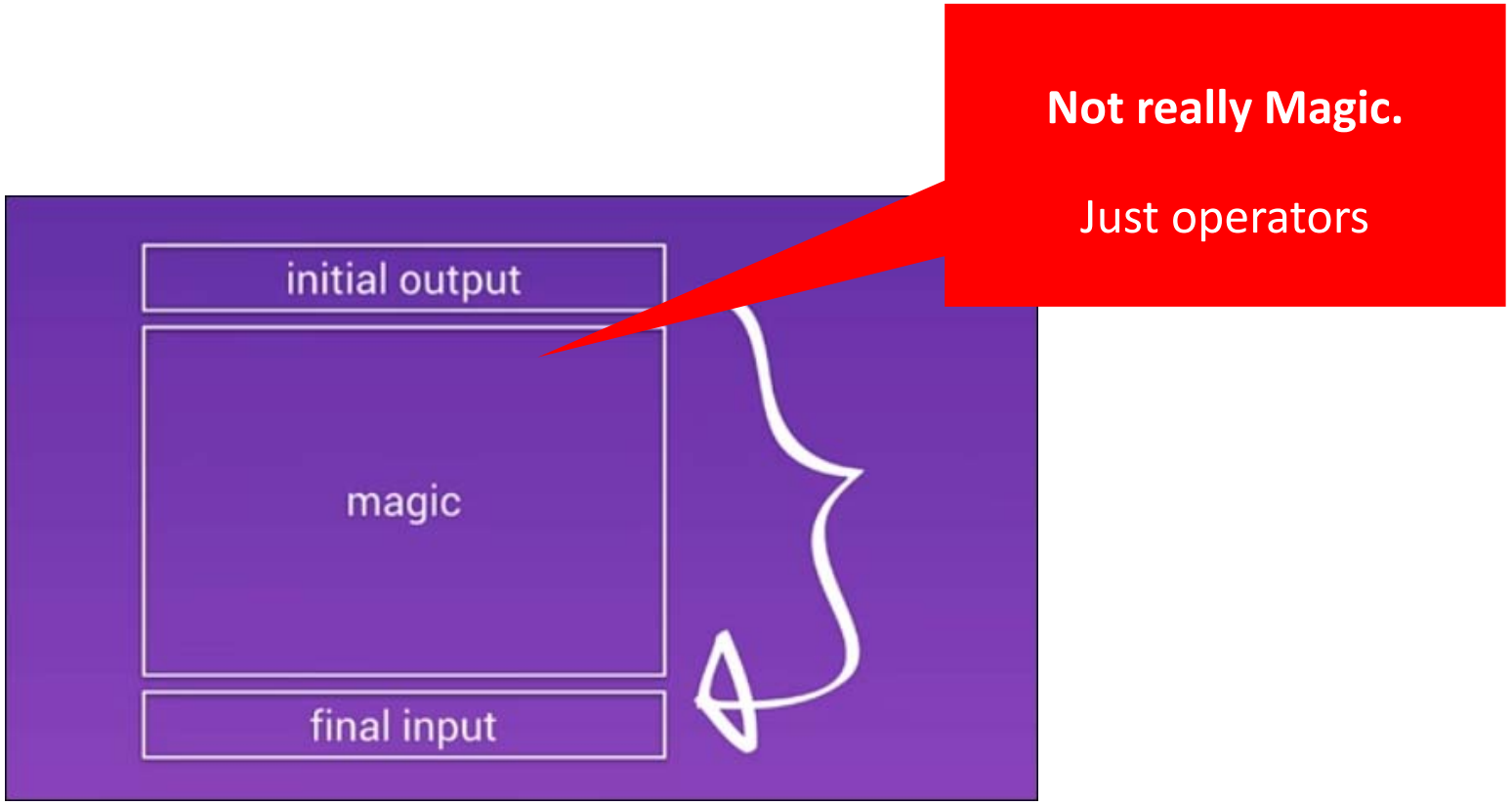
https://www.youtube.com/watch?v=5CTL7aqSvJU

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

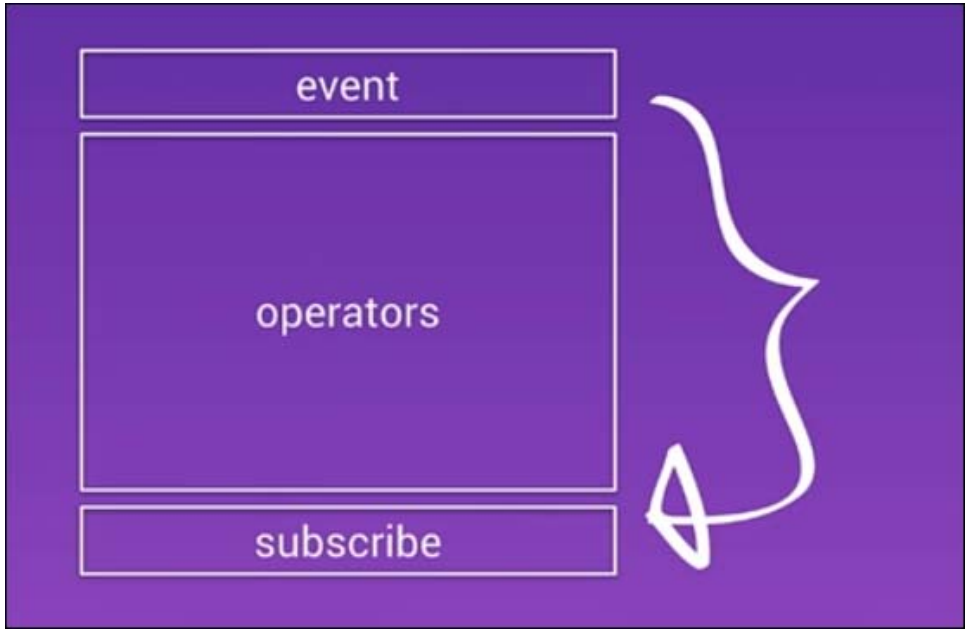
- With Observables -
  - a system, already outputting data,
  - Subscribe to that data
- "trade Output for Input"
- "Push vs. Pull"



"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

Import HttpModule in @ngModule – don't forget

- *// Angular Modules*
  …
- **import** {HttpClientModule} **from '@angular/common/http'**;

```
// Module declaration
@NgModule({
    imports     : [BrowserModule, HttpClientModule],
    declarations: [AppComponent],
    bootstrap   : [AppComponent],
    providers   : [CityService] // DI voor service
})
export class AppModule {
}
```

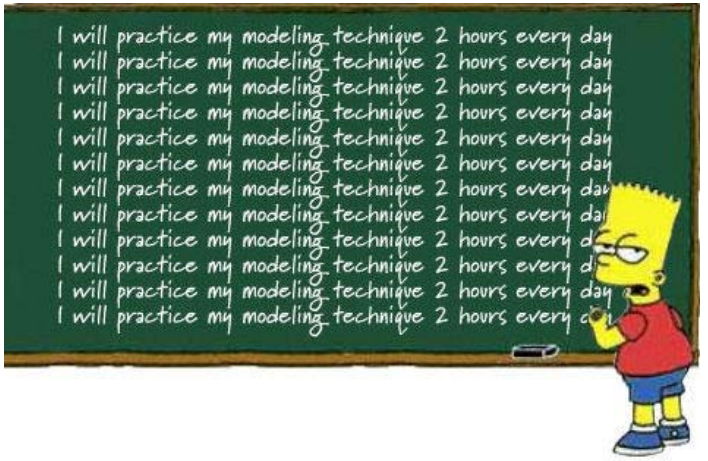
OLD: Angular < 4.3: HttpModule

- In @ngModule: `imports : [HttpModule]`
- Using map-operator `.map(res => res.json())`.
  - Json has to be extracted by using `.map()`.
- `HttpModule` will be removed in future versions!
- Now we have Interceptors (in `HttpClientModule`)
- https://alligator.io/angular/httpclient-intro/ and
- https://alligator.io/angular/httpclient-interceptors/

Exercise

- See the example in `/201_services_http`
- Create your own `.json`-file and import it in your application.
- Exercise `5c), 5d)`

Exercise….

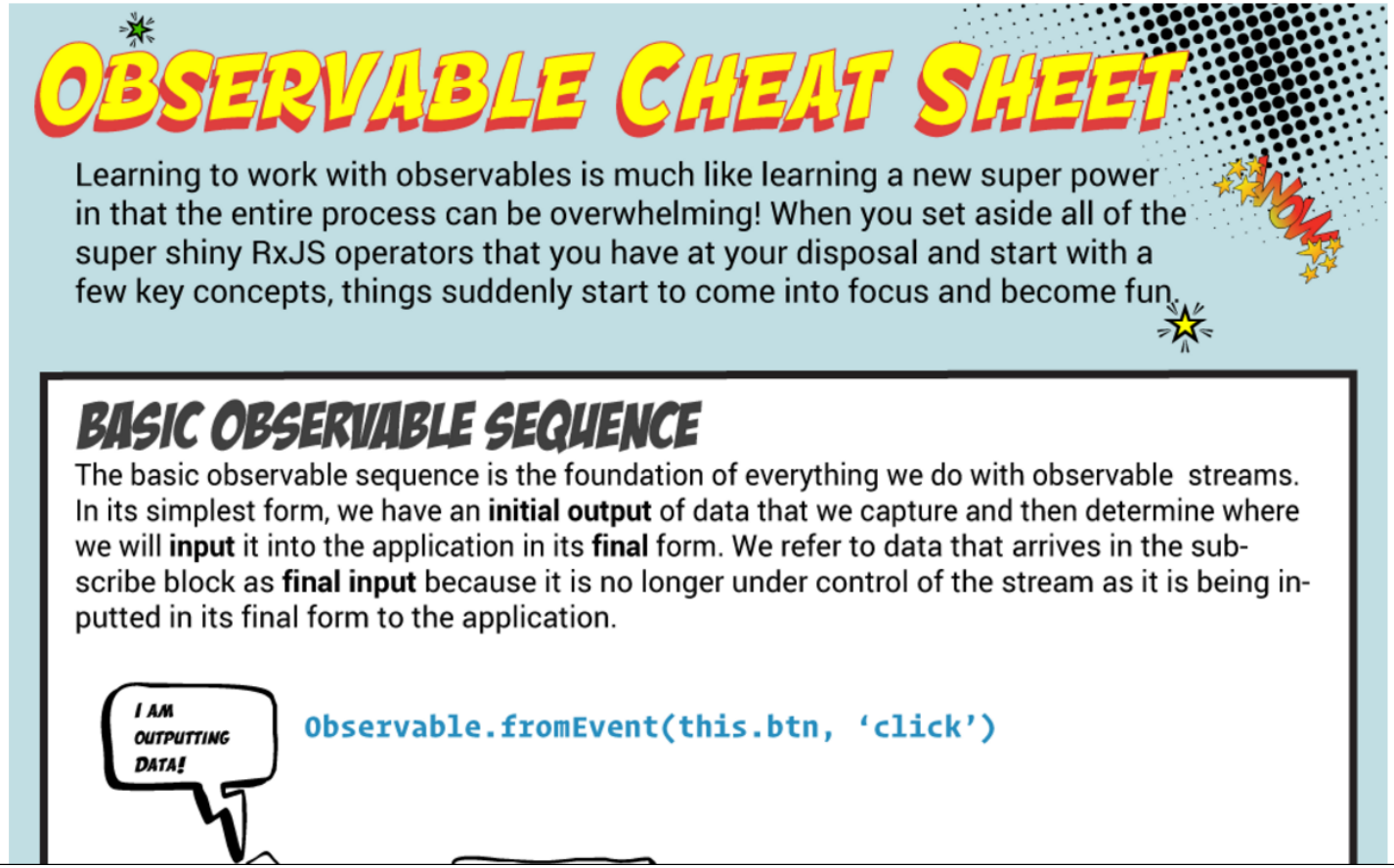


Observable Cheat Sheet

genius to understand.

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

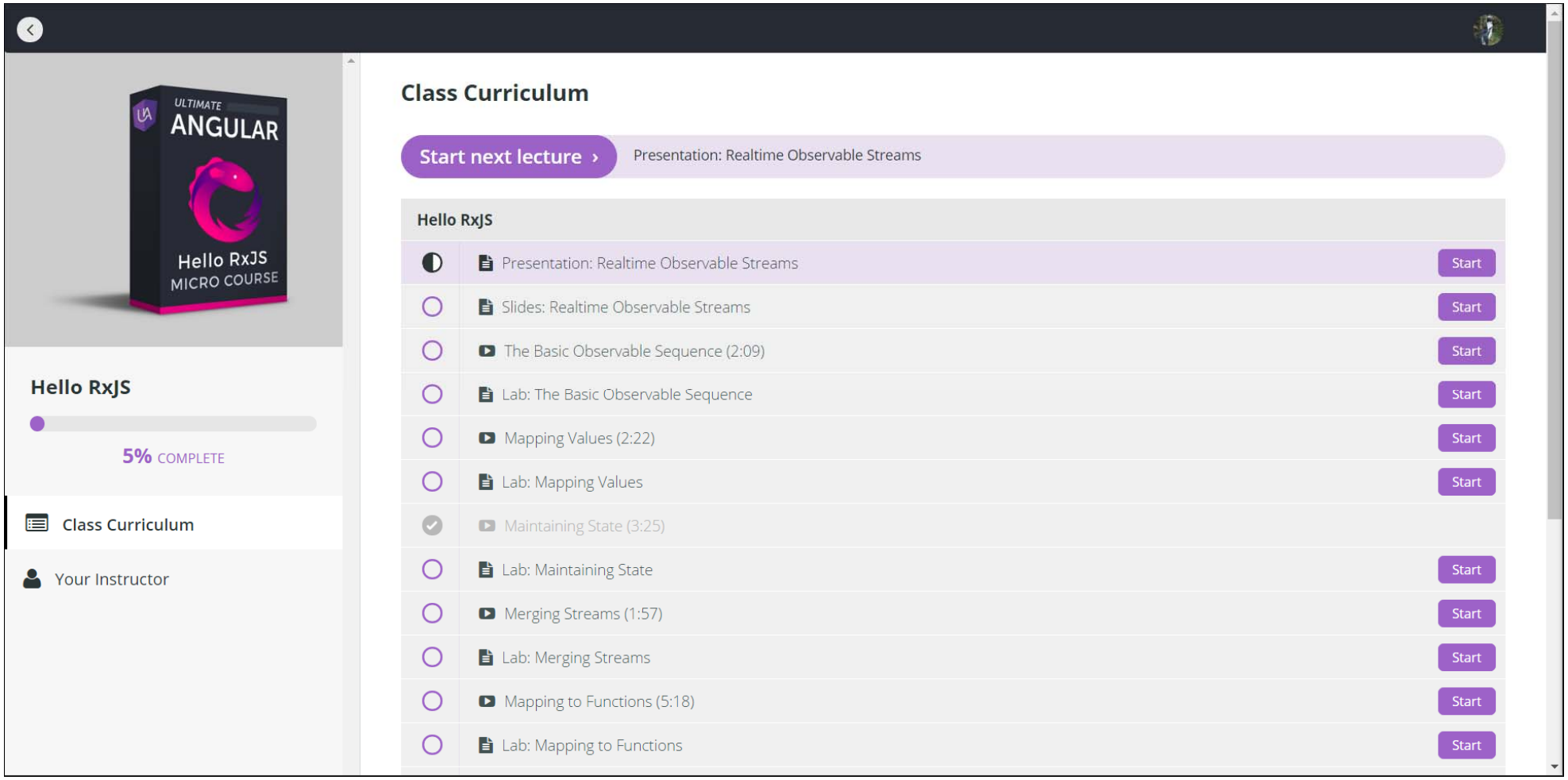
I really hope that you find the infographic helpful. Be sure to drop me a line below if you have any questions or comments. #highFive



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

Hello RxJS

Free online training



http://courses.ultimateangular.com/

Pipeable operators

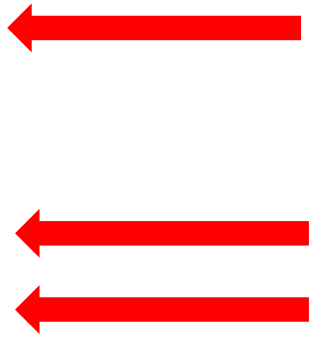
- In RxJS 6.x and up: all operators inside `.pipe()` functie
- The parameters of pipe function are the operators!
- Comma-separate different operators.
  - Don't forget `import {…} from 'rxjs/operators';`

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

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...')
    )
```

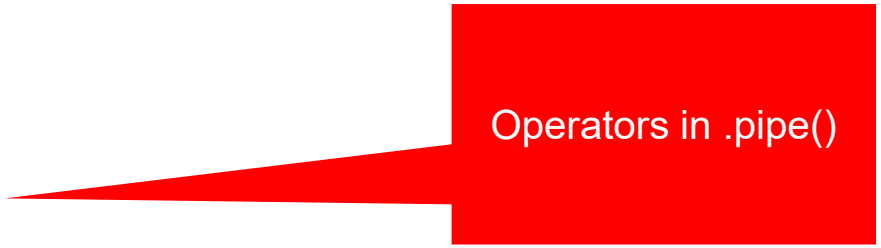


RxJS-operators in the service

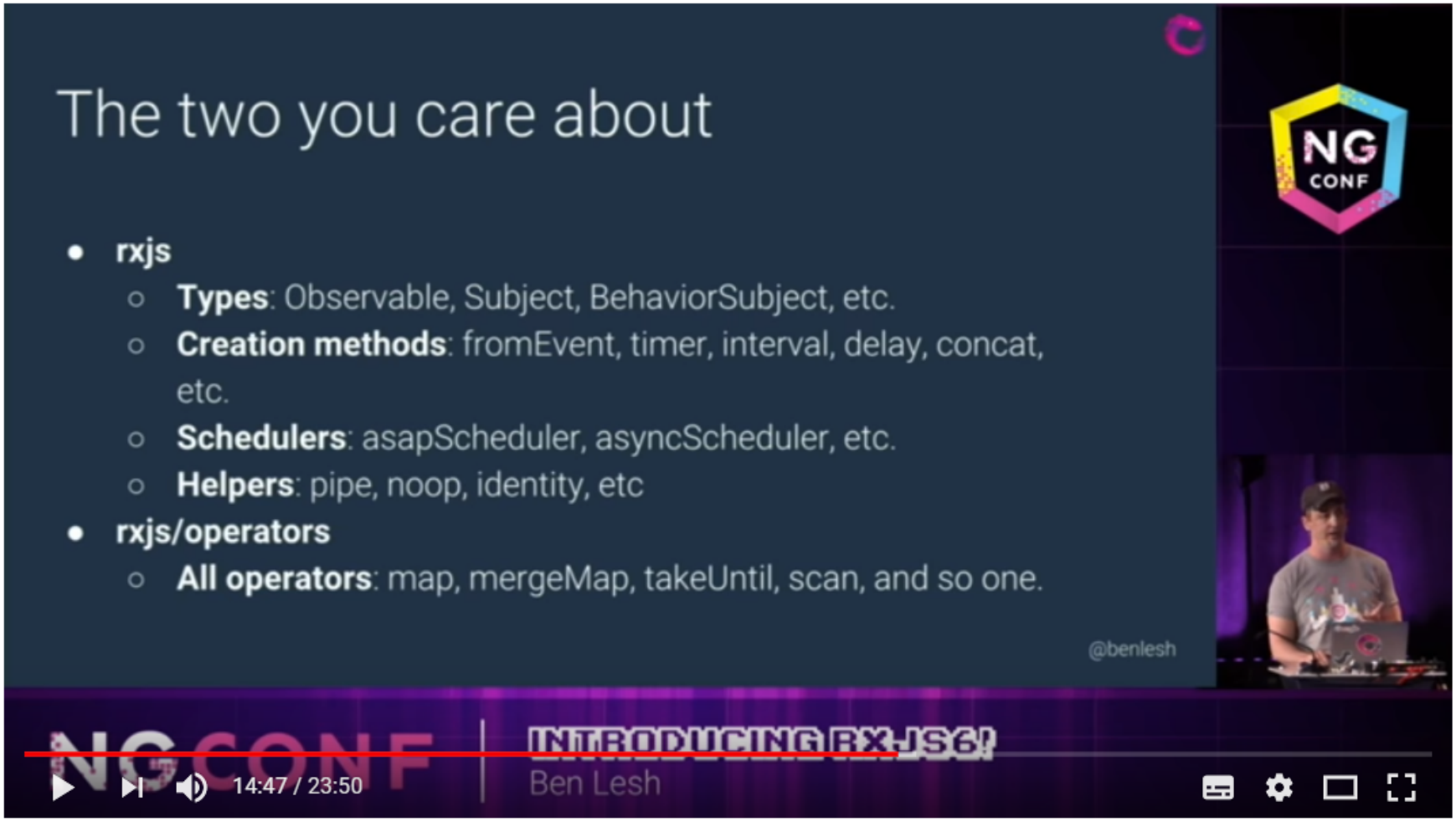
```
import {Injectable} from '@angular/core';
import {HttpClient} from "@angular/common/http";
import {map, delay, takeUntil, …} from "rxjs/operators";

@Injectable()
export class CityService {

   constructor(private http: HttpClient) {

   }

   // retourneer alle cities
   getCities(): Observable<Response> {
      return this.http.get('shared/data/cities.json')
         .pipe(…);
   }

}
```

The result of the first operator is the input of the next operator in the pipeline

```
getCities() {
   if (!this.cities) {
      this.cityService.getCities()
         .pipe(
                delay(3000),
                retry(3)
                 map(result => …),
                takeUntil(…condition…)
         )
      .subscribe(cityData => {
             this.cities = cityData;
       })
   }
}
```



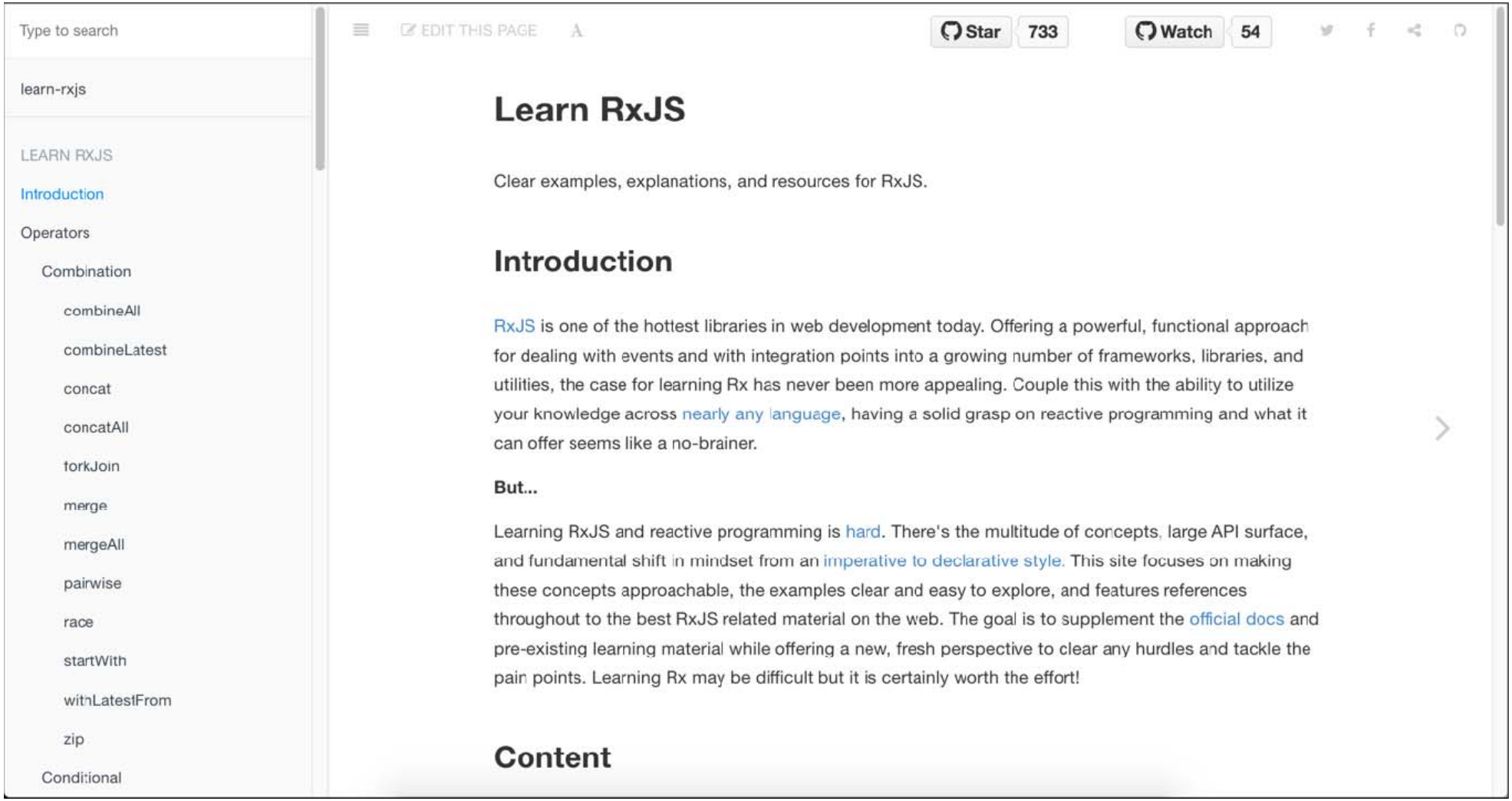
Ben Lesh on observables in RxJS 6.0



Introducing RxJS6! - Ben Lesh

https://www.youtube.com/watch?v=JCXZhe6KsxQ

https://www.learnrxjs.io/



Using the async pipe

Automagically `.subscribe()` and `.unsubscribe()`

Async Pipe

- On `.subscribe()`, you actually need to `.unsubscribe()`
  - Best practice
  - Not *really* necessary on HTTP-requests, but you do in other subscriptions wel, to avoid memory leaks.
- No more manually `.subscribe()` and `.unsubscribe()`:
  - Use Angular async pipe

- Component:

```
cities$: Observable<City[]>; // Now: Observable to Type


…
ngOnInit() {
   // Call service, returns an Observable
   this.cities$ = this.cityService.getCities()
}
```

- View:

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

Working with Live API's

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

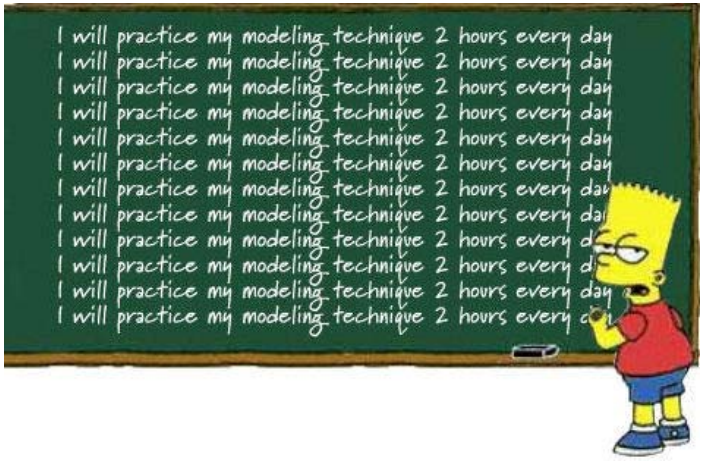
Example API's

- https://pokeapi.co/ - Pokemon API
- http://openweathermap.org/API (weather forecast)
- http://randomuser.me/ (random user data)
- http://ergast.com/mrd/ - Ergast Motor (F1) API
- http://www.omdbapi.com/ - Open Movie Database
- http://swapi.co/ - Star Wars API
- See also `JavaScript APIs.txt` with other API's.

Exercise

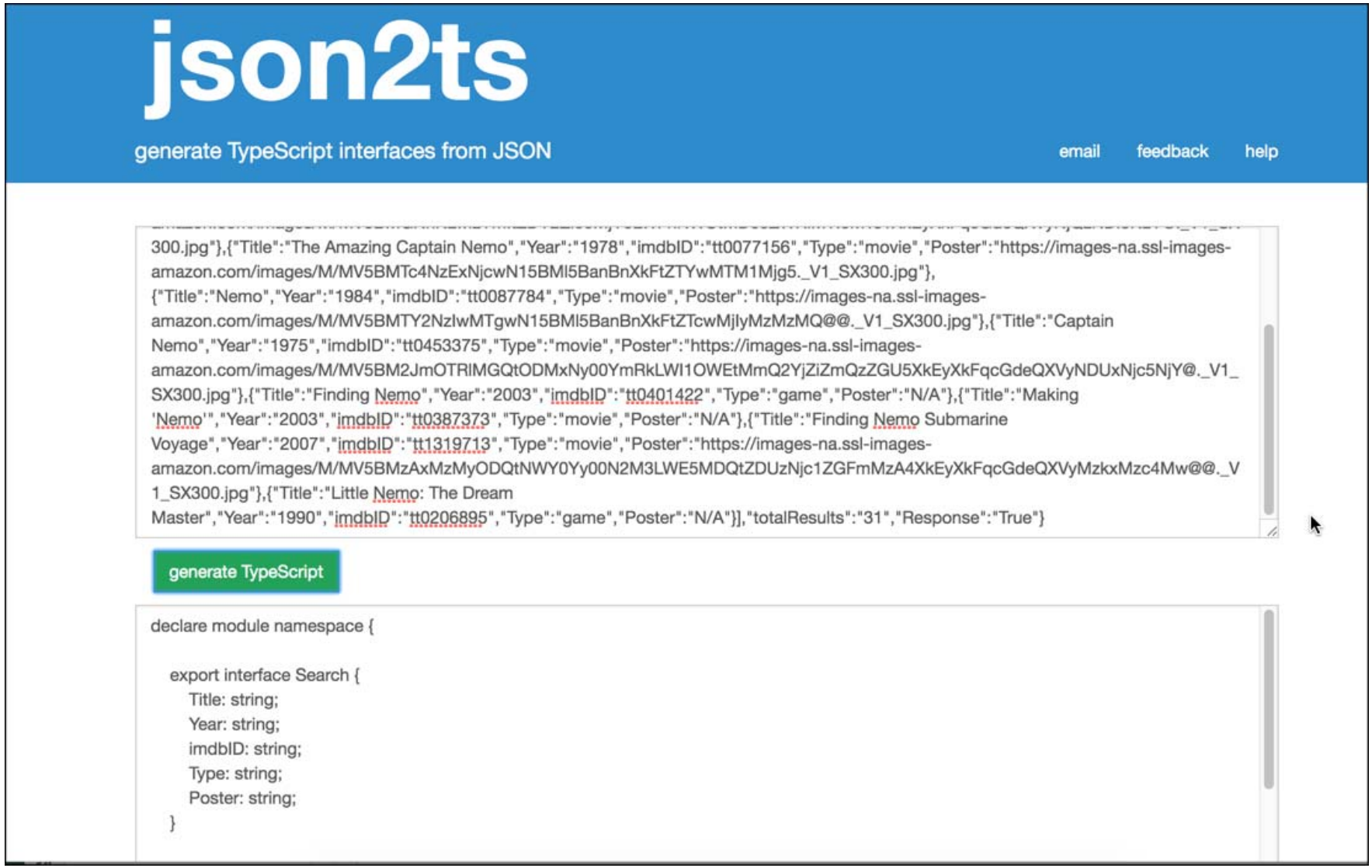
- Pick one of your own projects, or see for examples:
  - `210-services-live`
  - `(502-forms-typeahead` (We'll cover forms later on))
- Create a small application using one of the API's in the file `JavaScript API's.txt`, using RxJS-calls, for example
  - Star Wars API
  - OpenWeatherMap API
  - …
- Exercise 5e)

Exercise….



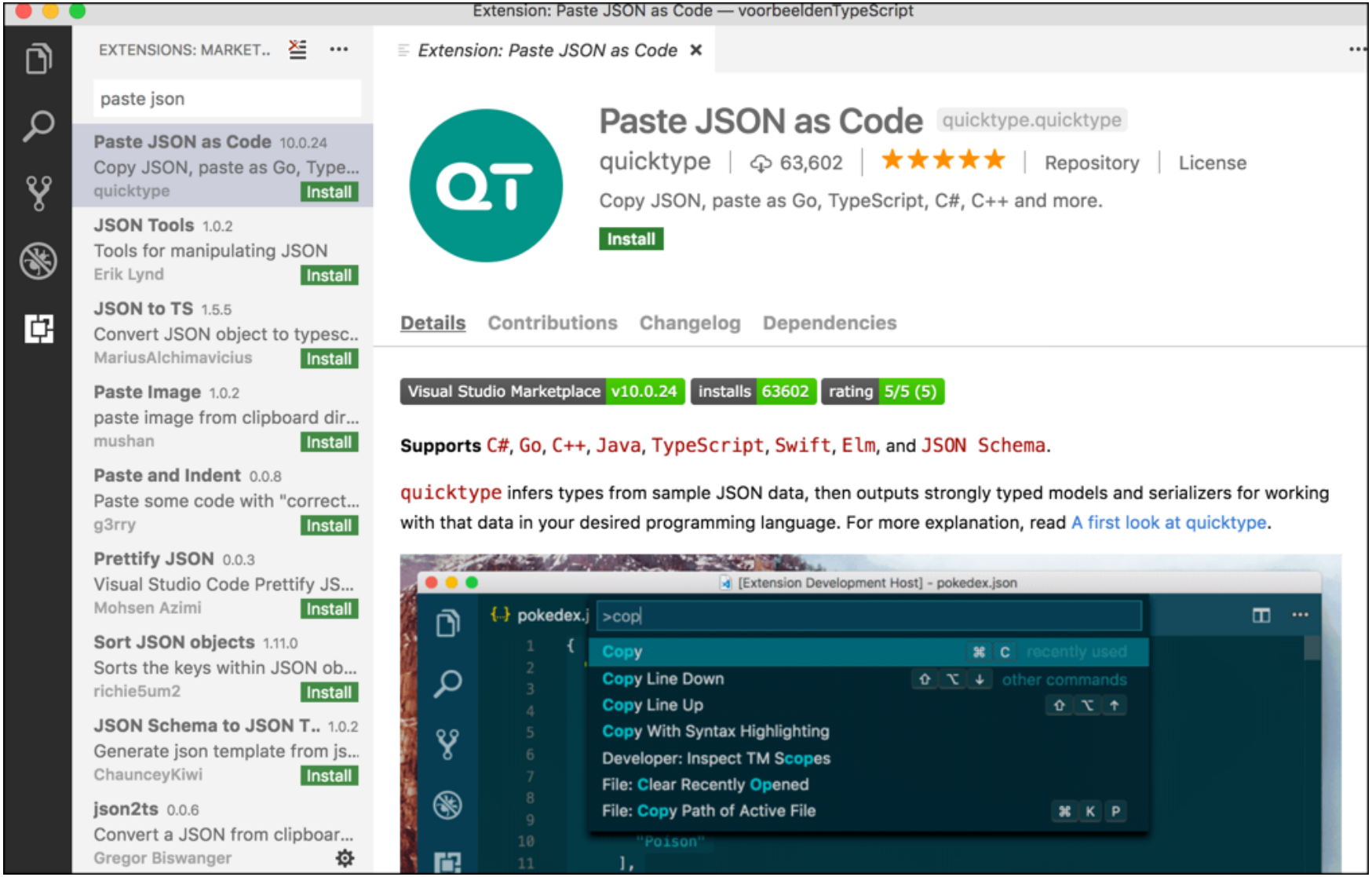
More info on observables

Online JSON to TypeScript converter



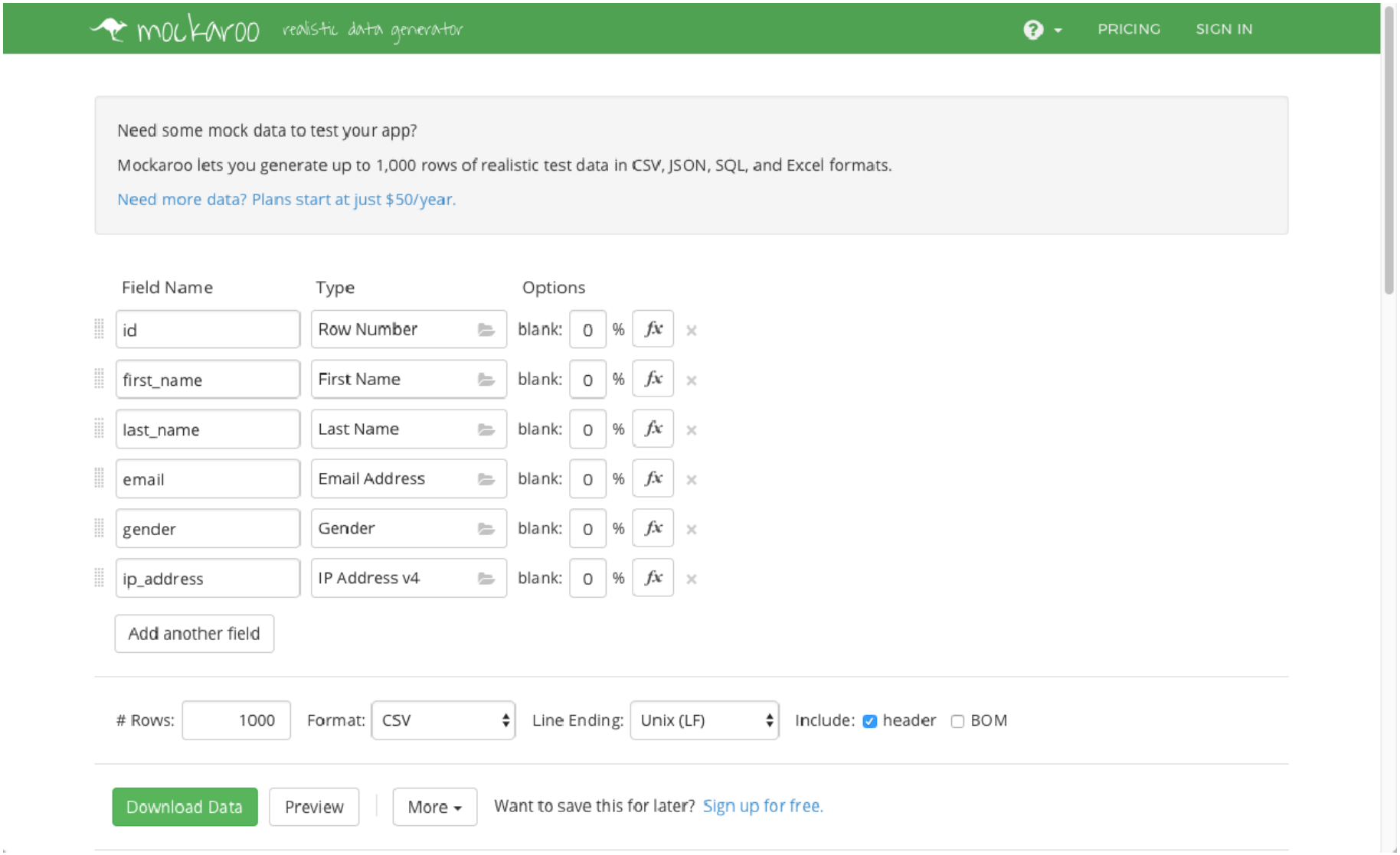
http://json2ts.com/

In VS Code? Use this extension!



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

Data Mocking - Mockaroo

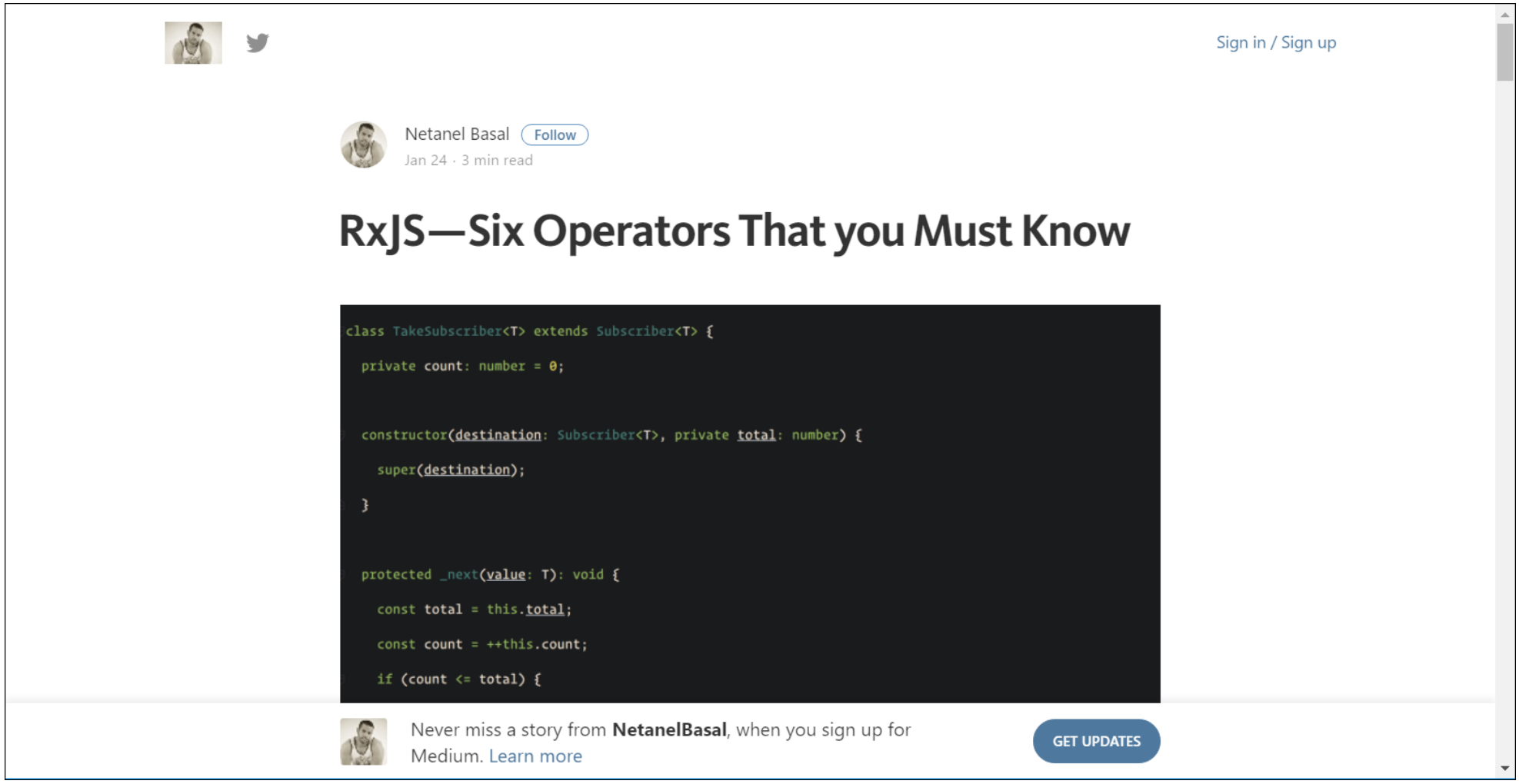


http://mockaroo.com/

Useful operators

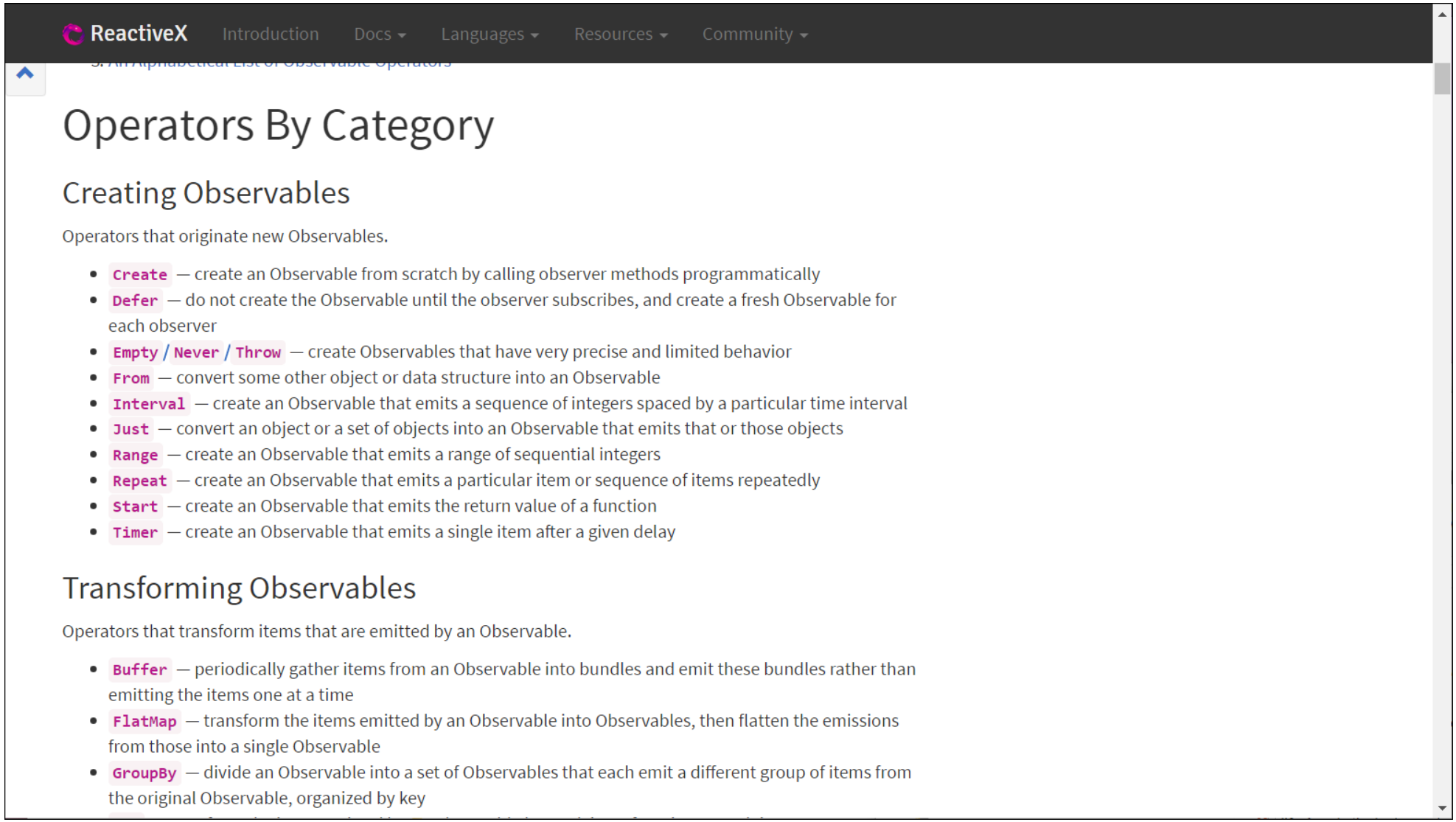
- RxJS operators are (mostly) just like Array operators
- Perform actions on a stream of objects
- Grouped by subject
  - Creation operators
  - Transforming
  - Filtering
  - Combining
  - Error Handling
  - Conditional and Boolean
  - Mathematical
  - ...

6 Operators you must know



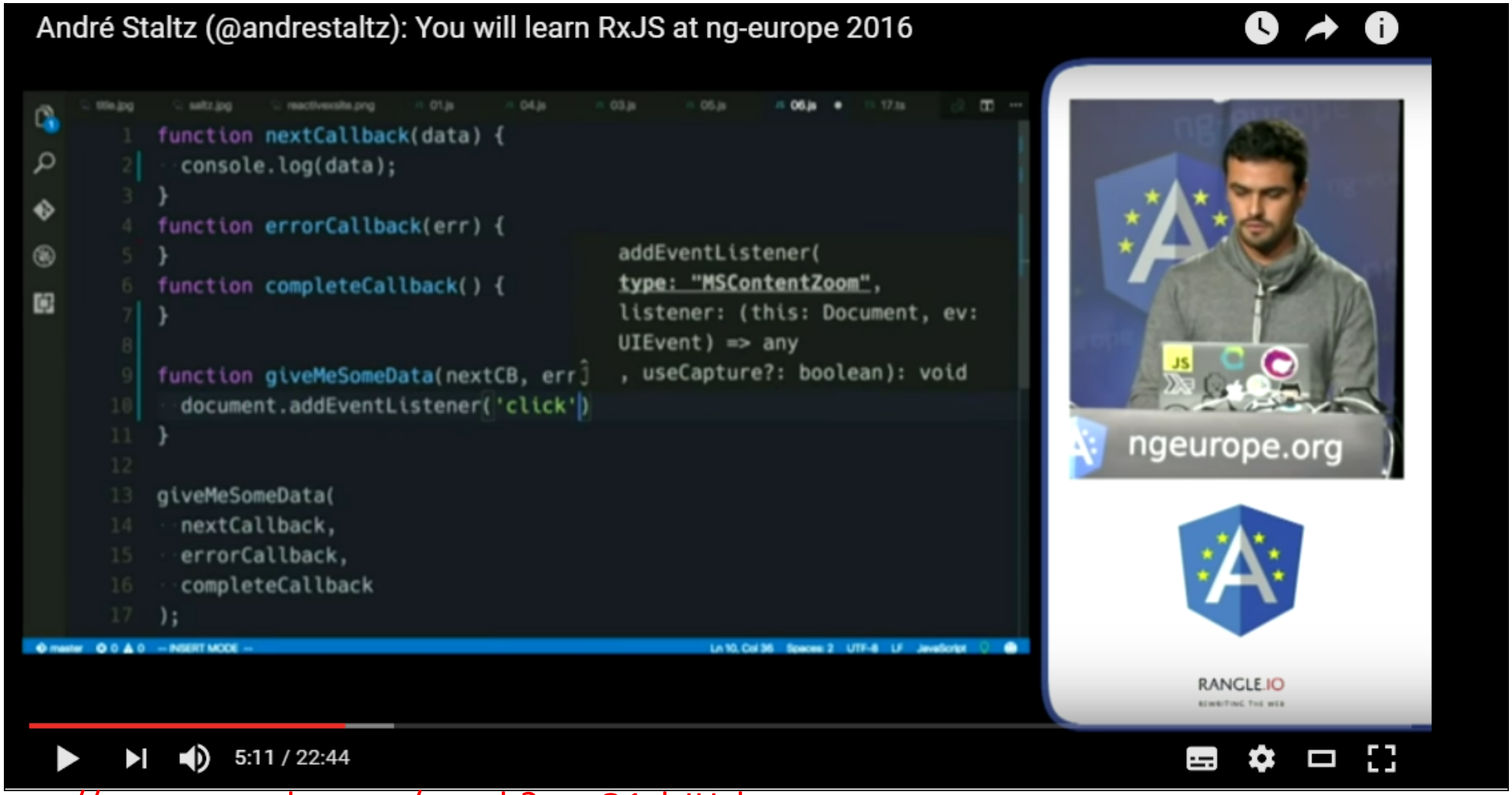
https://netbasal.com/rxjs-six-operators-that-you-must-know-5ed3b6e238a0#.11of73aox

Documentation at reactivex.io

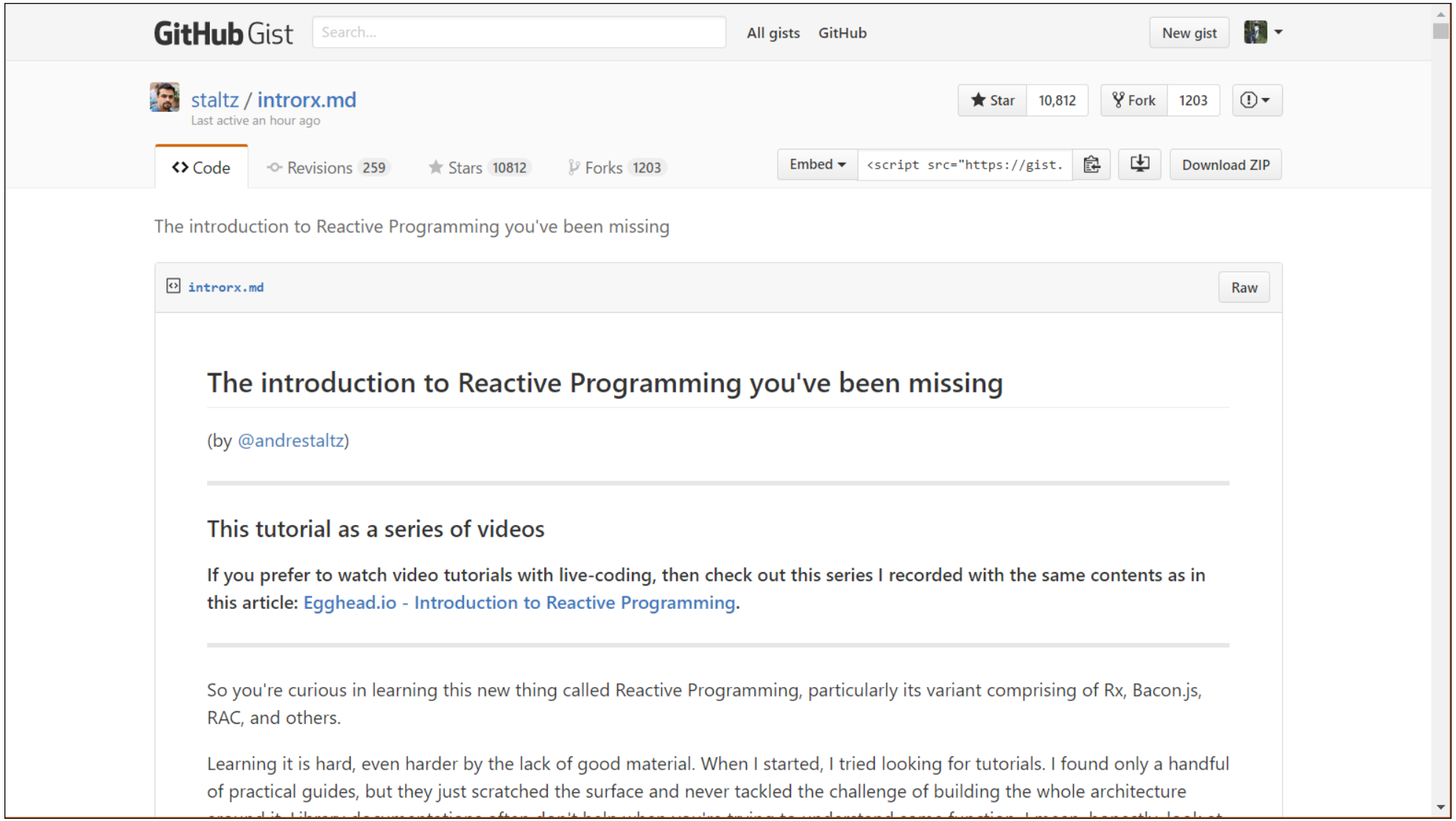


http://reactivex.io/documentation/operators.html

Creating Observables from scratch - André Staltz

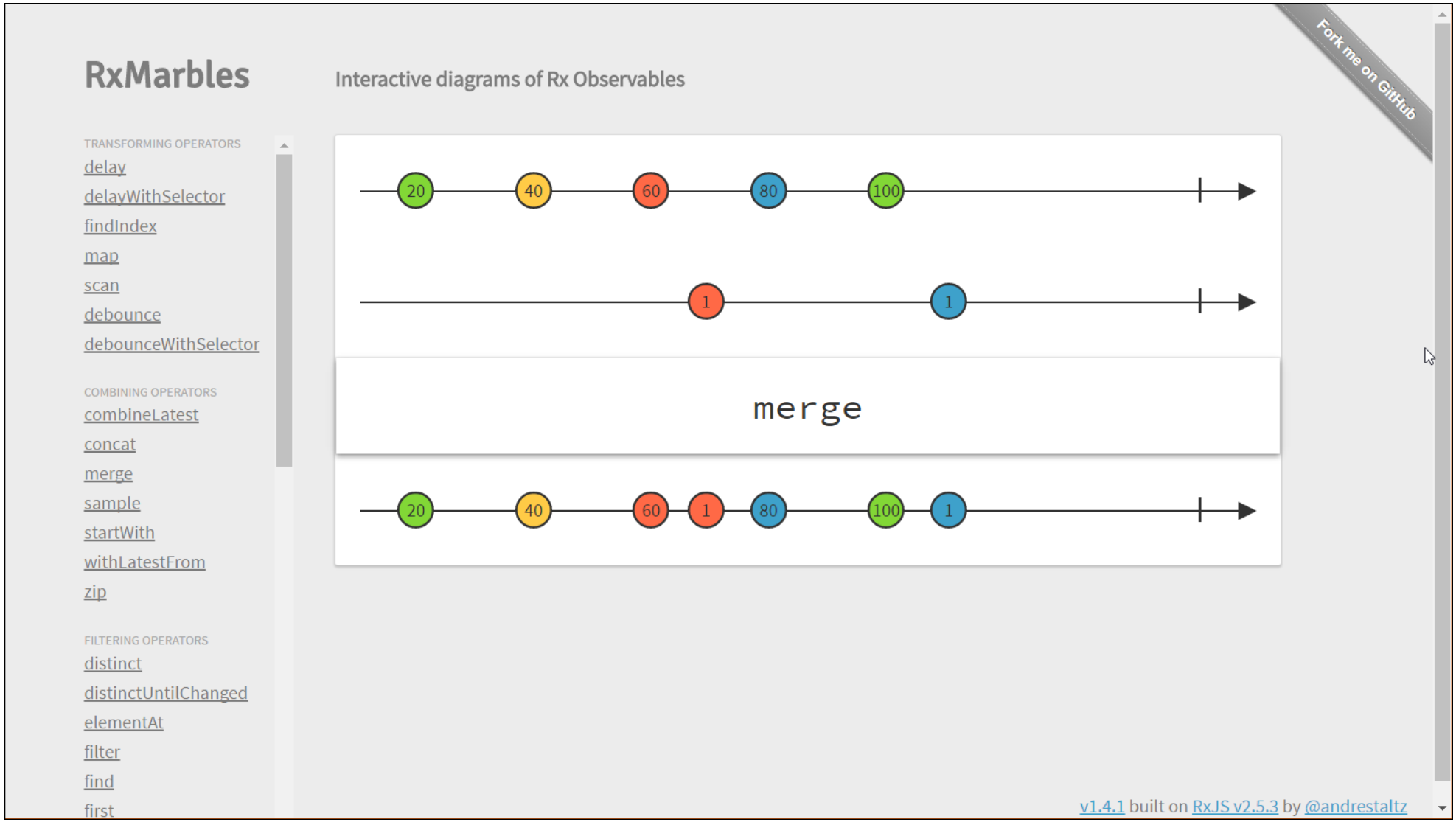


https://www.youtube.com/watch?v=uQ1zhJHclvs



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

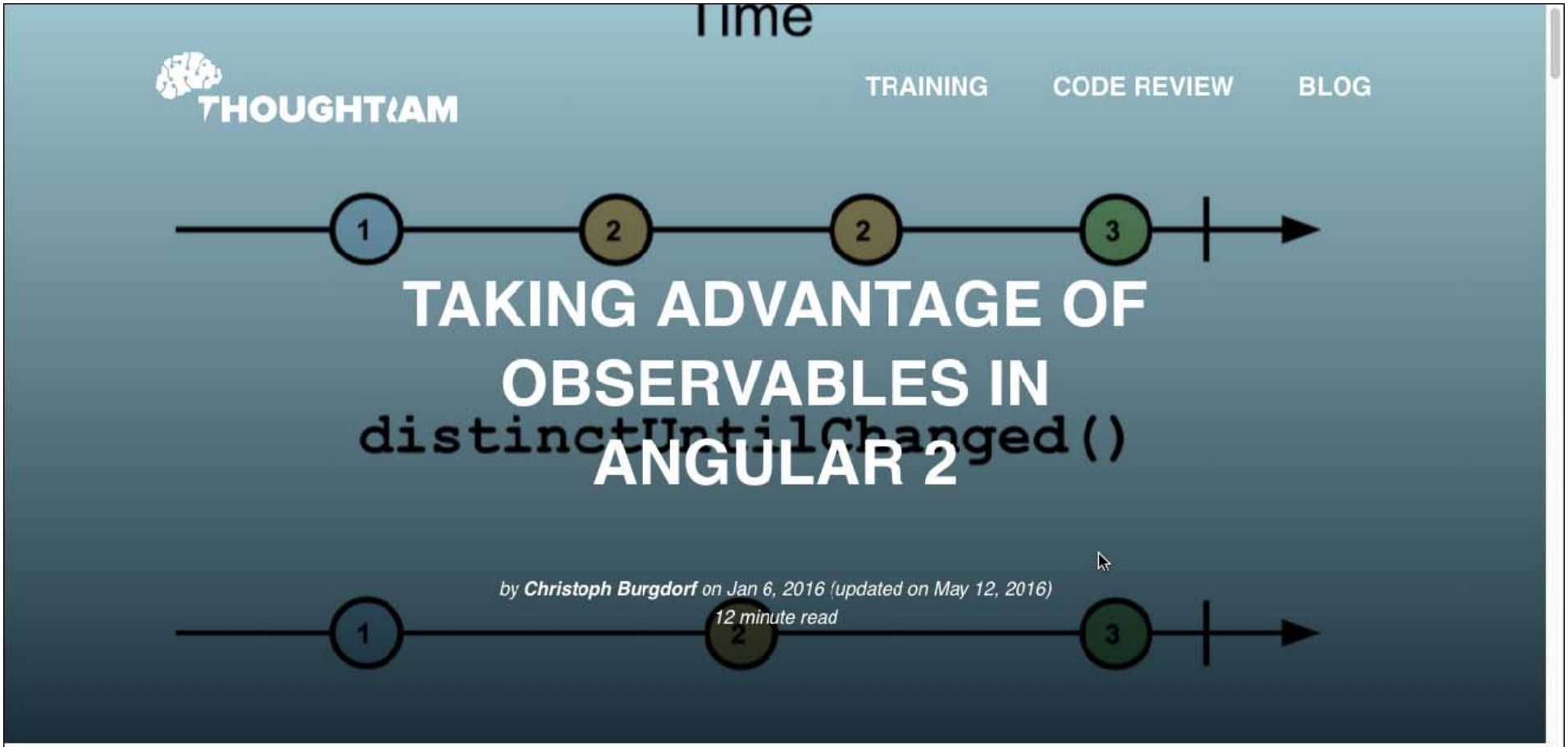
RxMarbles



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