Angular Advanced short recap – day #1

Peter Kassenaar –
info@kassenaar.com

Day 1:

- Some Angular CLI tips & tricks
  - Angular 6, `ng add, ng update`
  - Angular Console
- NG applications with multiple modules
  - Using multiple modules in your app
- Basic Routing – Routing parameters
- Lazy loading modules
  - `PreloadAllModules`
  - Custom Loading strategies
  - Configure Router with `ExtraOptions`

Configuring the router

```
let routerConfig: ExtraOptions = {
  enableTracing: true
  preloadingStrategy: PreloadAllModules
};

@NgModule({
  imports: [RouterModule.forRoot(routes, routerConfig)],
  exports: [RouterModule]
})
```

Routing events are actually Observables.

Which means we can subscribe! And do something like:

```
this.router.events
     .subscribe(event =>{
       console.log(' router event: ', event);
  })
```

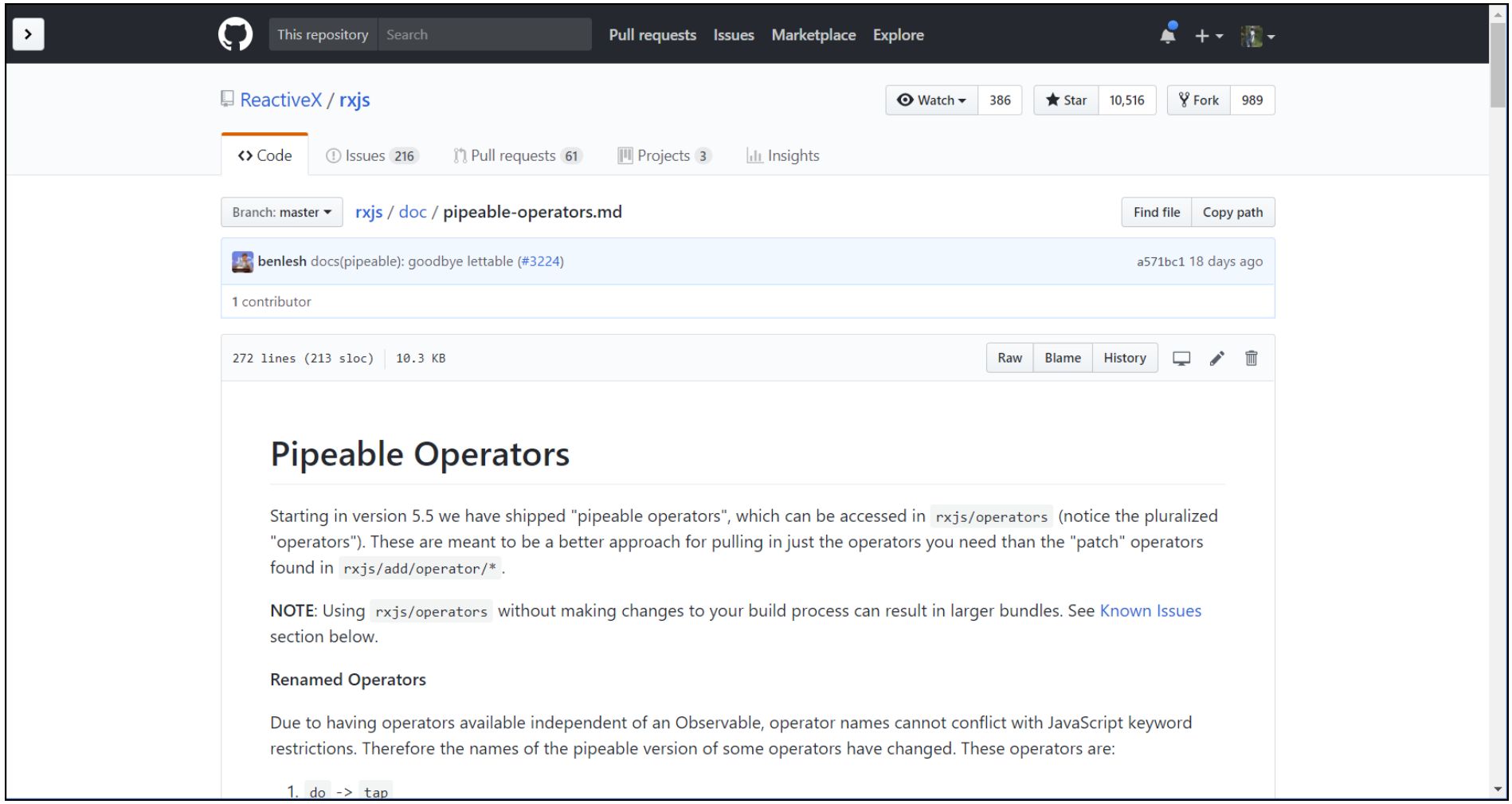
(Don’t forget to inject router:

```
constructor(private router: Router) {  })
```

Or, in a more reactive way of programming:

```
this.router.events.pipe(
    filter(event => event instanceof NavigationEnd),
    map(...),
    . ...
    ).subscribe(event =>{
      console.log(' router event: ', event);
```

RxJS 5.5+: Pipeable operators en `.pipe()`



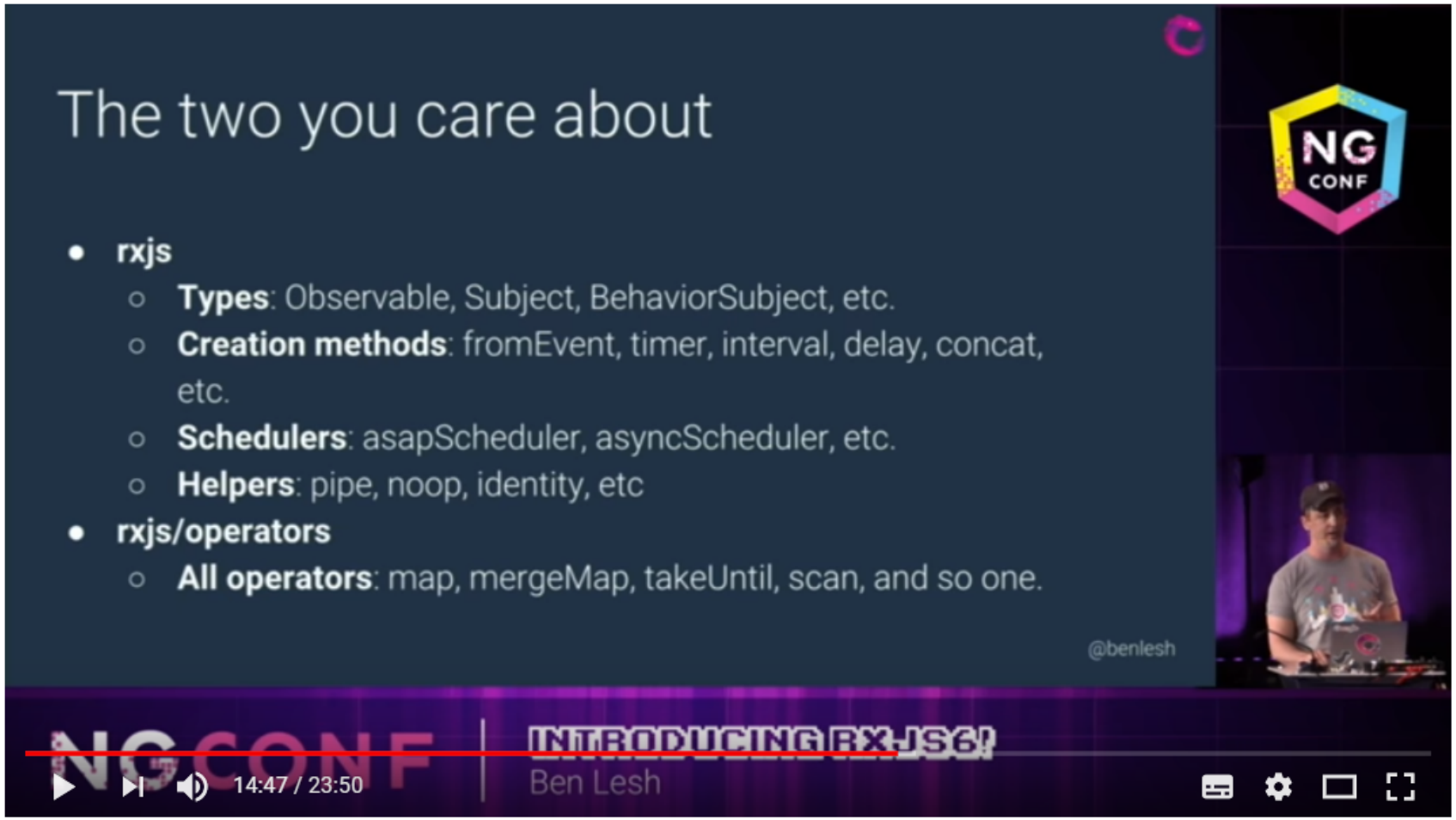
https://github.com/ReactiveX/rxjs/blob/master/doc/pipeable-operators.md

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

```
this.http.get<MyObject[]>(http://some_url')
.pipe(
   delay(3000),
   retry(3)
   map(result => …),
   takeUntil(…condition…)
)
```

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
  - …

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”>
```

Today

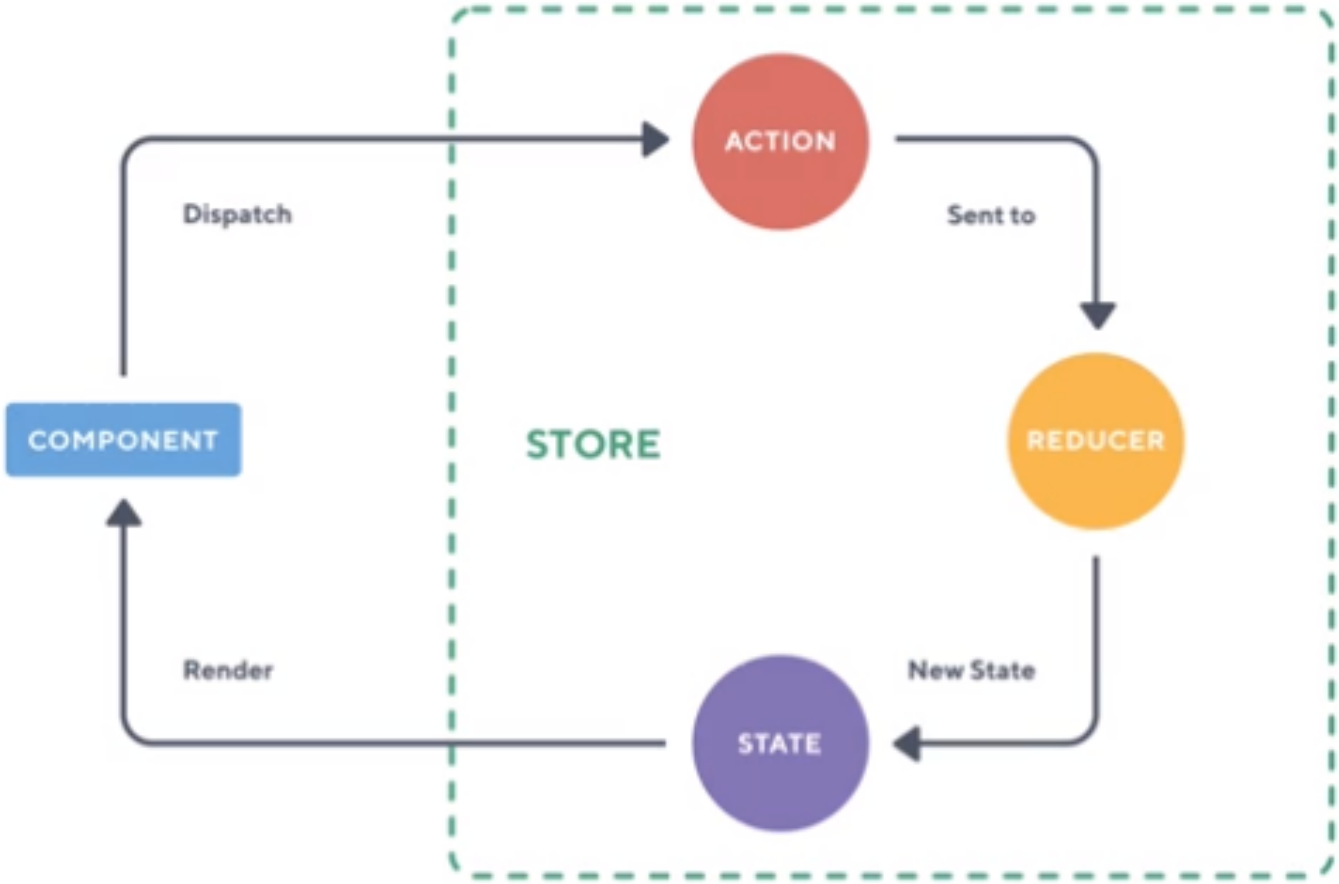
- Content Projection
  - `<ng-content select="…">`
- Smart/View Components
- State Management w/ `@ngrx/store`
  - Store
  - Actions & Action Creators
  - Reducers
  - Payload
  - Store v2.0.0 vs v4.0.0+

tomorrow – Wednesday Sept. 26

- Continue `@ngrx/store`
  - Working with complex types
  - Redux DevTools
  - Working with `Http` (directly)
  - Using `@ngrx/effects` and `@Effect()`
- Angular Elements
- Firebase/PWA/Testing/…

REDUX ARCHITECTURE

One-way dataflow



https://platform.ultimateangular.com/courses/ngrx-store-effects/lectures/3788532