The Power of RxJS

In NativeScript + Angular

What is RxJS ?

Push based primitive in JavaScript

What is RxJS ?

Push based primitive in JavaScript

Ref implementation for TC39 Observable proposal

What is RxJS ?

Push based primitive in JavaScript

Domain specific language that sits on top of JS

Ref implementation for TC39 Observable proposal

What is RxJS ?

Only external dependency to Angular (2x)

Push based primitive in JavaScript

Domain specific language that sits on top of JS

Ref implementation for TC39 Observable proposal

What is RxJS ?

Only external dependency to Angular (2x)

Push based primitive in JavaScript

Domain specific language that sits on top of JS

Ref implementation for TC39 Observable proposal

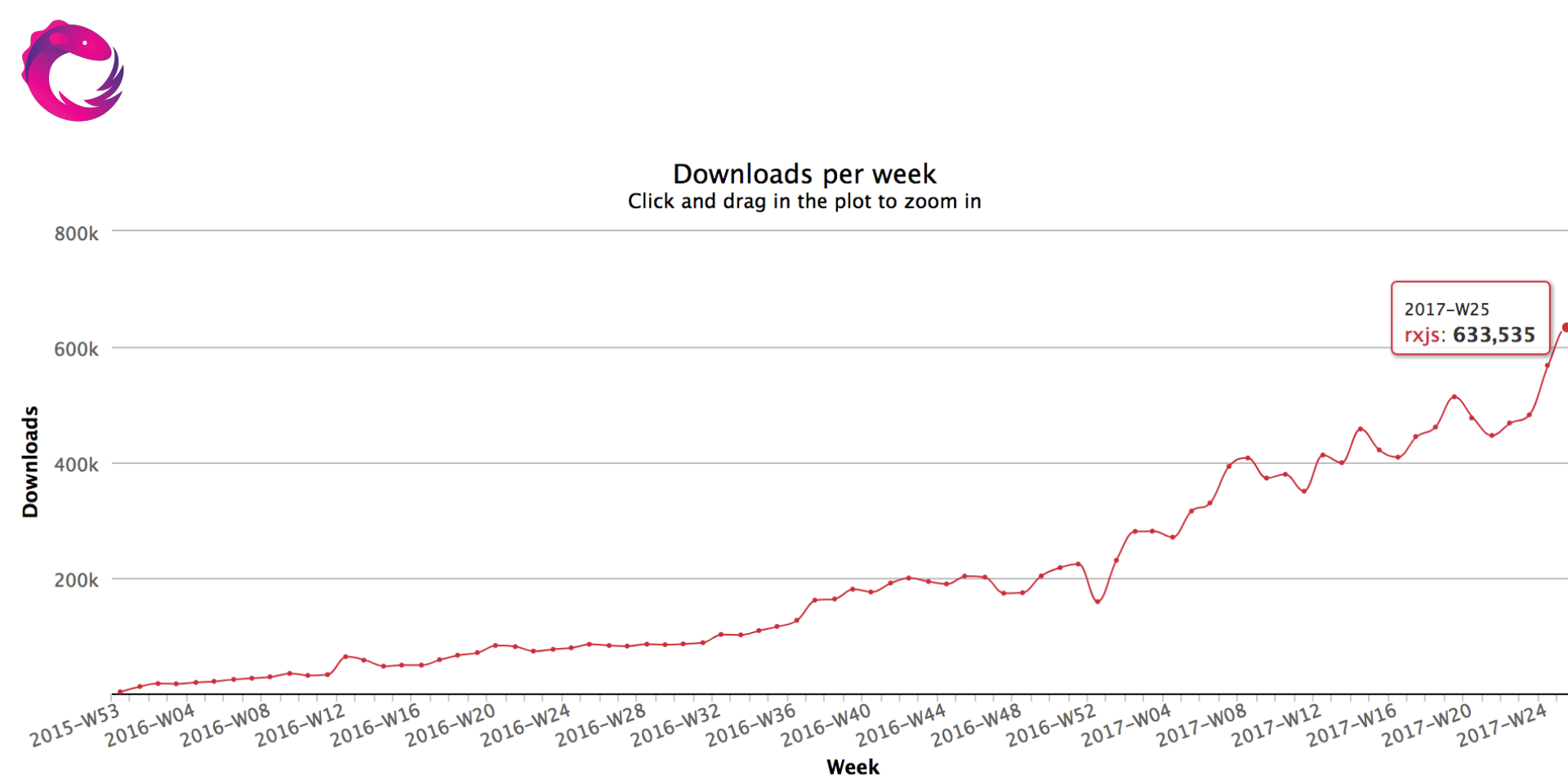
Easy to chain together complex interactions

RxJS Authors

Matt Podwysocki, RxJS 4

Ben Lesh, RxJS5+

RxJS is used in big cos


Downloads per week
Click and drag in the plot to zoom in
800k
600k
400k
200k
Downloads
2017-W25
rxjs: 633,535
2015-W53
2016-W04
2016-W08
2016-W12
2016-W16
2016-W20
2016-W24
2016-W28
2016-W32
2016-W36
2016-W40
2016-W44
2016-W48
2016-W52
2017-W04
2017-W08
2017-W12
2017-W16
2017-W20
2017-W24
Week

RxJS - 2 Major Components

Observables & Observers

RxJS - 2 Major Components

Observables & Observers

Observables give data and observers receive data

RxJS - 2 Major Components

Observables & Observers

Observables give data and observers receive data

Observables are just functions in JavaScript - they don’t do anything until you subscribe to them.

RxJS - 2 Major Components

Observables & Observers

Observables give data and observers receive data

Observables are just functions in JavaScript - they don't do anything until you subscribe to them.

Observables are always observed by observers

RxJS - 2 Major Components

Observables & Observers

Observables give data and observers receive data

Observables are just functions in JavaScript - they don't do anything until you subscribe to them.

Observables are always observed by observers

Subscribing to an Observable sets up a chain of observation using observers.

Yeah but… isn’t it just for async?

Nope! Observables are just push based primitives that push values at you.

Synchronous examples: Observable.of and Observable.from

Observable.of(x) just gives you a back whatever you pass into that argument.

Observable.from converts an array, promise, iterable) -
observable.from([1,2,3])

TRACY LEE | @ladyleet

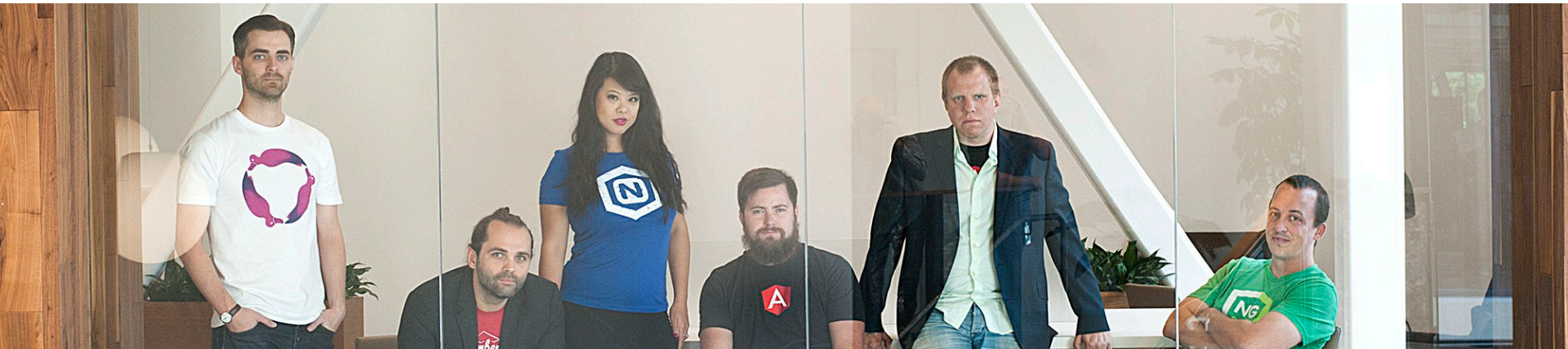
SPEAKER & GOOGLE DEVELOPER EXPERT

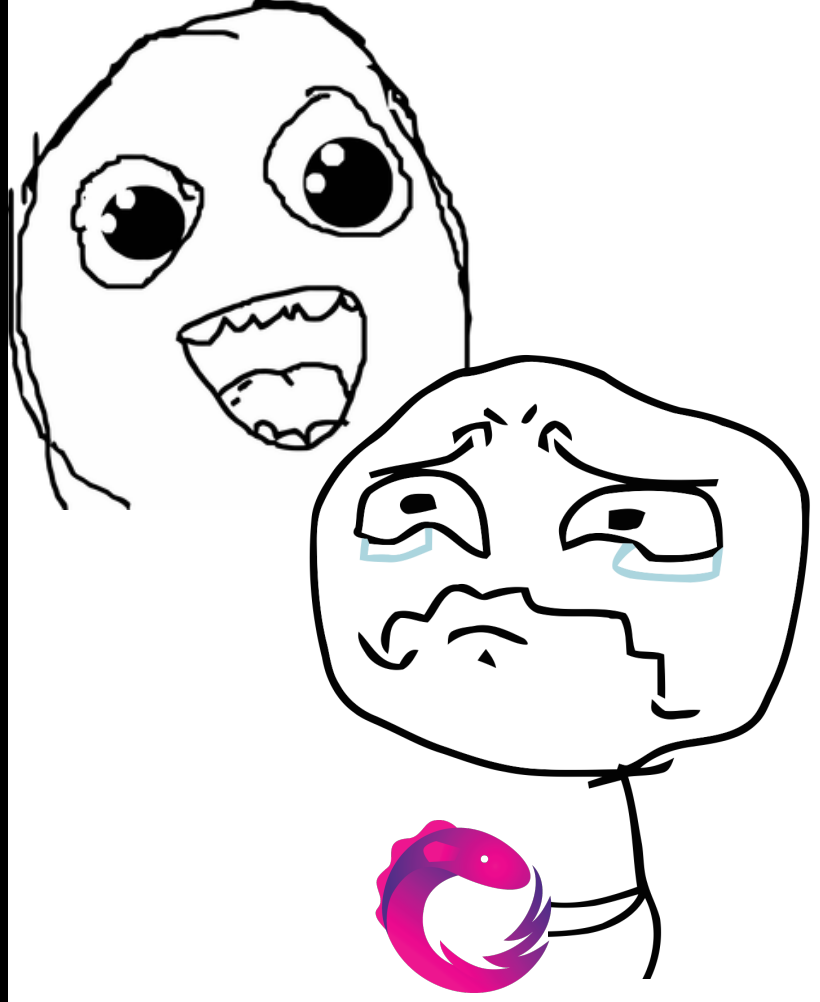
DEV - ANGULAR, REACT, RXJS, REACT NATIVE, NATIVESCRIPT

THIS DOT LABS & THIS DOT MEDIA

TECH WRITER & PODCASTER

MENTOR





Learning RxJS Be Like...

Couldn’t find the right docs!

Learning RxJS Be Like...

What do these operators even do?

Learning RxJS Be Like...

Oops, I forgot to subscribe. (╯°□°)╯ ⁀ ┻━━┻

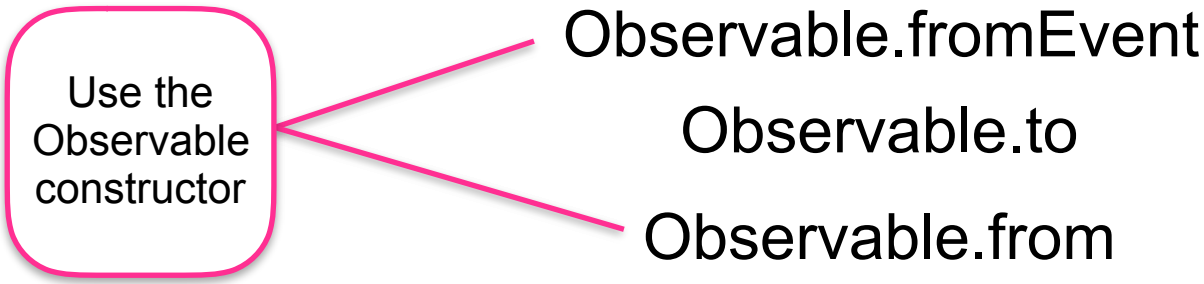
Learning & Using RxJS

- How to create an Observable
- Best practices for importing and using RxJS
- How to choose operators and find documentation
- How to avoid unwanted subscriptions
- The benefits of “same-shapedness”
- What’s coming for RxJS 6 and 7

How to create an Observable

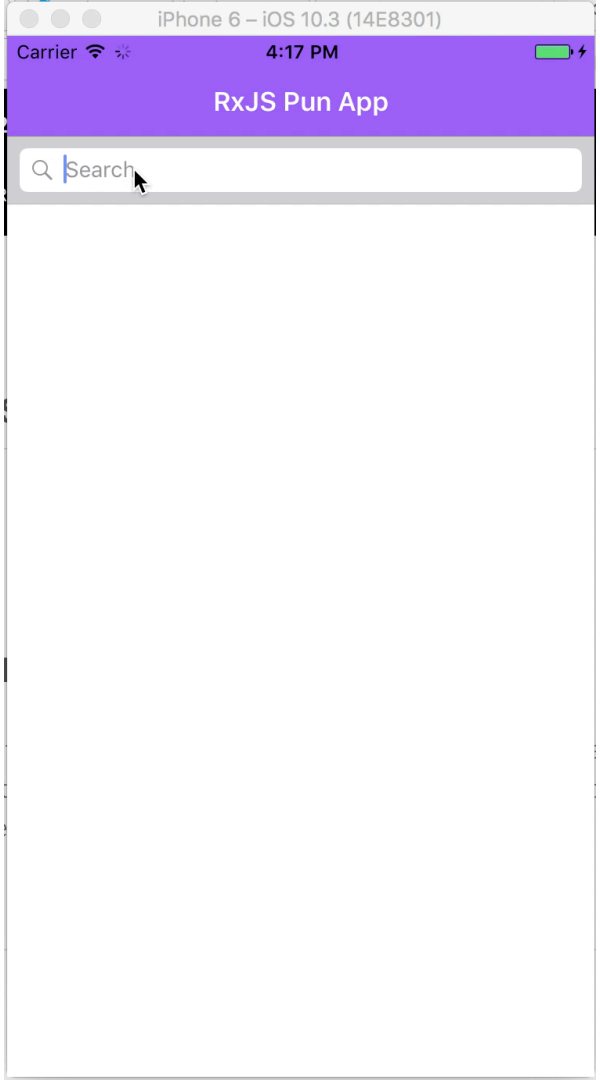
```
const x = new  Observable (observer => {
          observer.next('hi');
    observer.complete();
});
```

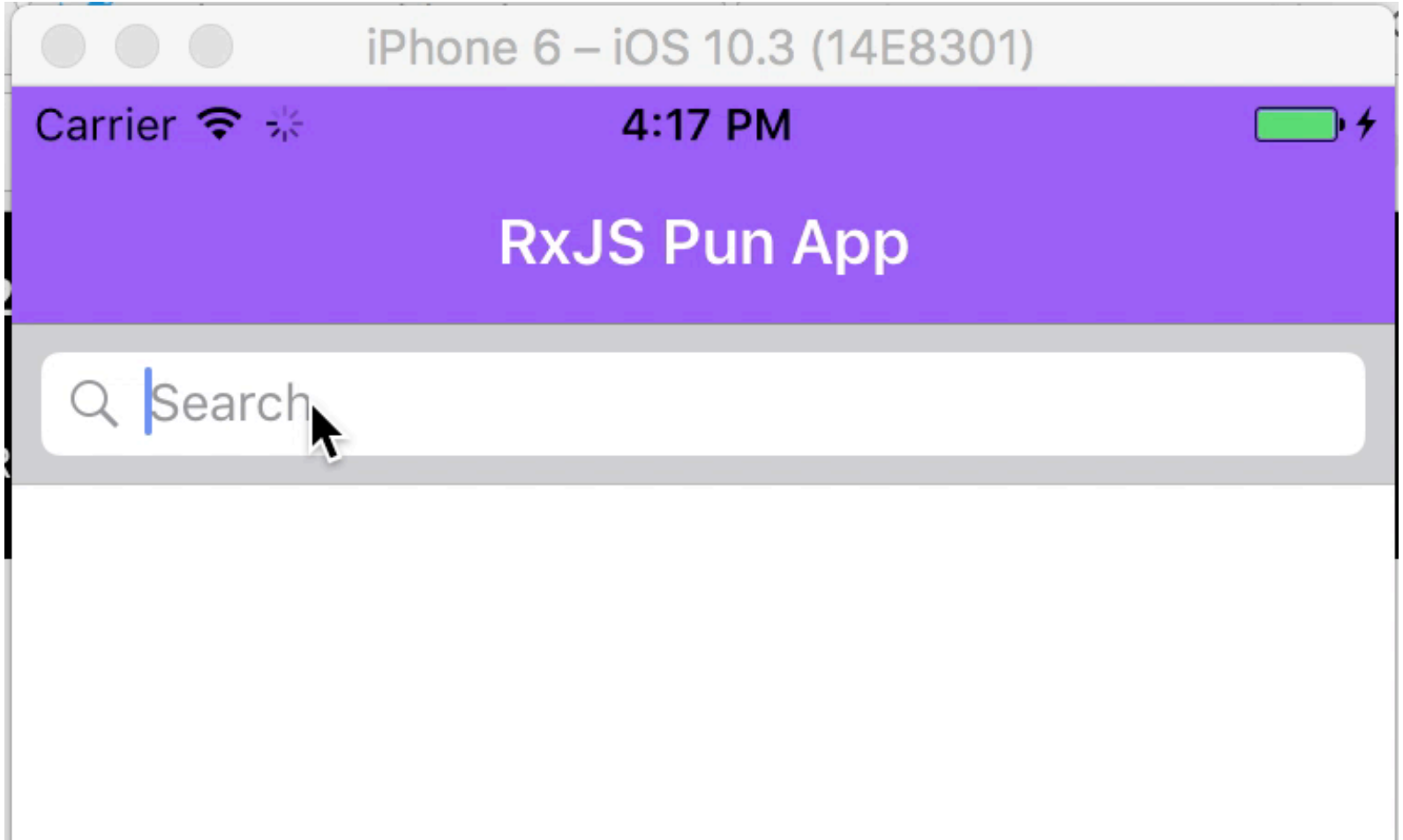
Many other Observable creation methods



Let’s build an app!

(It’s a pun app)


iPhone 6 – iOS 10.3 (14E8301)
Carrier
4:17 PM
RxJS Pun App
Search


iPhone 6 – iOS 10.3 (14E8301)
Carrier
4:17 PM
RxJS Pun App
Search

Lookahead Search

The Idiomatic RxJS Example

Setting up the App

Using NativeScript + Angular

Scaffold a NativeScript App with NativeScript CLI

```
$ tns create <name-of-app> - -ng

$ tns run ios
```

Create a Pun Component

pun.component.ts ✖ | items.component.html | app.css

```
1   import { Component, OnInit } from "@angular/core";
2
3   @Component({
4       selector: "ns-pun",
5       moduleId: module.id,
6       templateUrl: "./pun.component.html",
7   })
8   export class PunComponent implements OnInit {
9       ngOnInit() {
10
11      }
12  }
```

Add Component to Router

app.routing.ts ● | app.module.ts | app.routing.js

```
1  import { NgModule } from "@angular/core";
2  import { NativeScriptRouterModule } from
   "nativescript-angular/router";
3  import { Routes } from "@angular/router";
4
5  import { PunsComponent } from './puns/puns.component';
6
7  const routes: Routes = [
8      { path: "", redirectTo: "/puns", pathMatch: "full" },
9      { path: 'puns', component: PunsComponent}
10 ];
11
12 @NgModule({
13     imports: [NativeScriptRouterModule.forRoot(routes)],
14     exports: [NativeScriptRouterModule]
15 })
16 export class AppRoutingModule { }
```

Configure Main App Component

app.component.html

```
1   <ActionBar
2     title="RxJS Pun App"
3     class="action-bar">
4   </ActionBar>
5   <ScrollView>
6     <page-router-outlet></page-router-outlet>
7   </ScrollView>
```

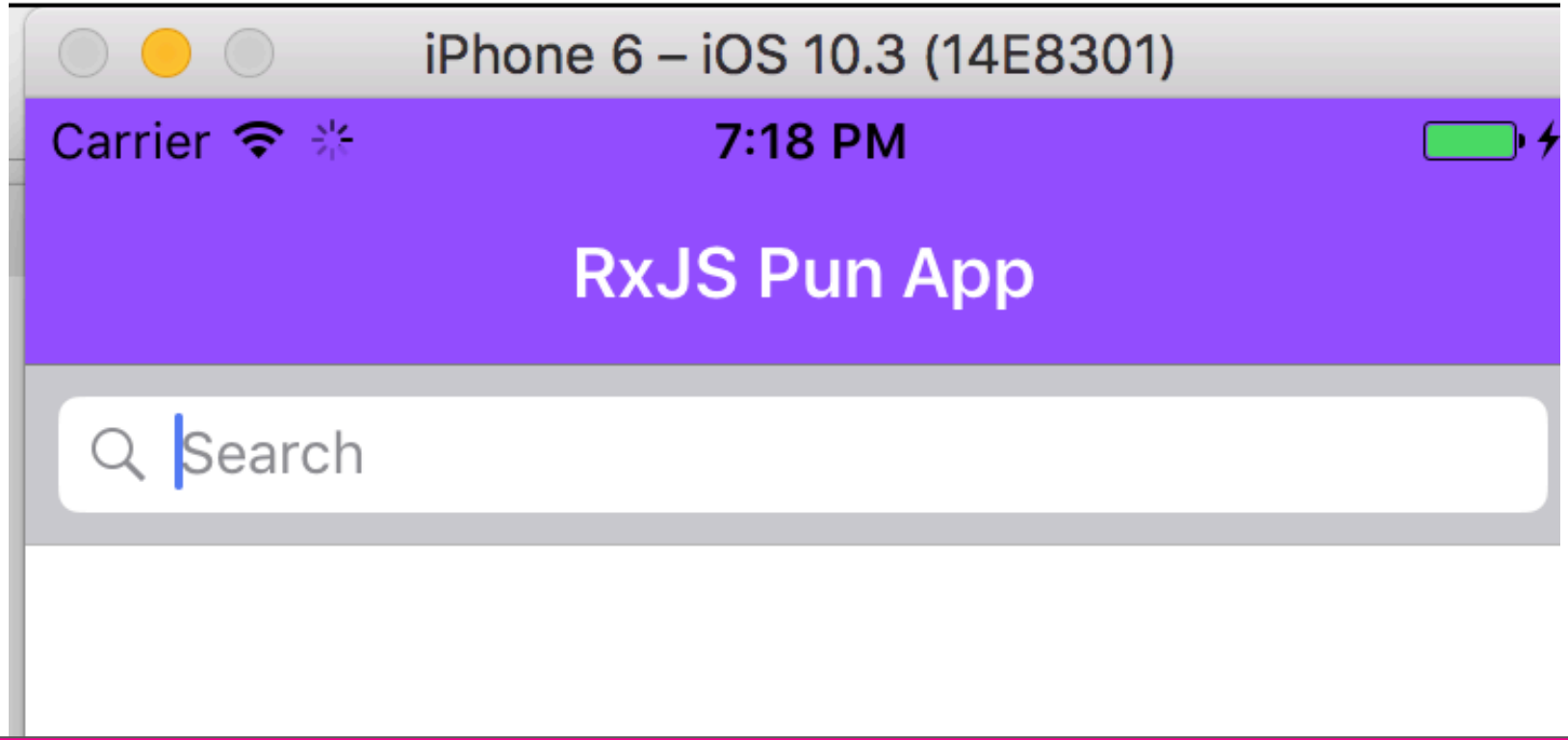
puns.component.html

Add the SearchBar Component

puns.component.html ✕ app.routing.ts

```
1   <StackLayout>
2       <SearchBar
3         id="searchBar"
4         hint="Search"
5         text=""
6         (clear)="onClear"
7         (submit)="onSubmit"
8         class="search-bar">
9       </SearchBar>
10  </StackLayout>
```

Add the SearchBar Component



Get Observable of Text Changes from Submit

Use Subject as an event handler in submit event.

puns.component.html | pun.service.ts | puns.component.ts

```
1   <StackLayout>
2       <SearchBar
3           #keywords
4           id="searchBar"
5           hint="Search"
6           (submit)="keywordInputChange$.next(keywords.text)"
7           class="search-bar">
8       </SearchBar>
9   </StackLayout>
```

Get Observable of Text Changes from Input

Add a template reference variable to get the text value

puns.component.html | pun.service.ts | puns.component.ts

```
1   <StackLayout>
2       <SearchBar
3        #keywords
4        id="searchBar"
5        hint="Search"
6        (submit)="keywordInputChange$.next(keywords.text)"
7        class="search-bar">
8       </SearchBar>
9   </StackLayout>
```

Get Observable of Text Changes from Input

Using the next method to get an Observable of values.

puns.component.html | pun.service.ts | puns.component.ts

```
1   <StackLayout>
2       <SearchBar
3         #keywords
4         id="searchBar"
5         hint="Search"
6         (submit)="keywordInputChange$.next(keywords.text)"
7         class="search-bar">
8       </SearchBar>
9   </StackLayout>
```

Get Observable of Text Changes from Input

Import Subject from RxJS

puns.component.ts | app.routing.ts | app.module.ts

```
1  import { Component, OnInit } from "@angular/core";
2  import { ActivatedRoute } from "@angular/router";
3  import { Subject } from 'rxjs/Subject';
4  import { PunService } from '../services/pun.service';
```

Get Observable of Text Changes from Input

Add the Subject to your component class

```
export class PunComponent implements OnInit {
    keywordsInputChange$ = new Subject<string>();
```

On the Subject of Subjects

- Subjects are Observables
- Subjects are Observers (with next, error and complete)
- Allow us to push values by calling `subject$.next(value)`
- Have all operators on them any observable would
- Multicast, or hot

Importing RxJS

… this is where a lot of people make a mistake.

puns.component.ts | app.routing.ts | app.module.ts

```
1  import { Component, OnInit } from "@angular/core";
2  import { ActivatedRoute } from "@angular/router";
3  import { Subject } from 'rxjs/Subject';
4  import { PunService } from '../services/pun.service';
```

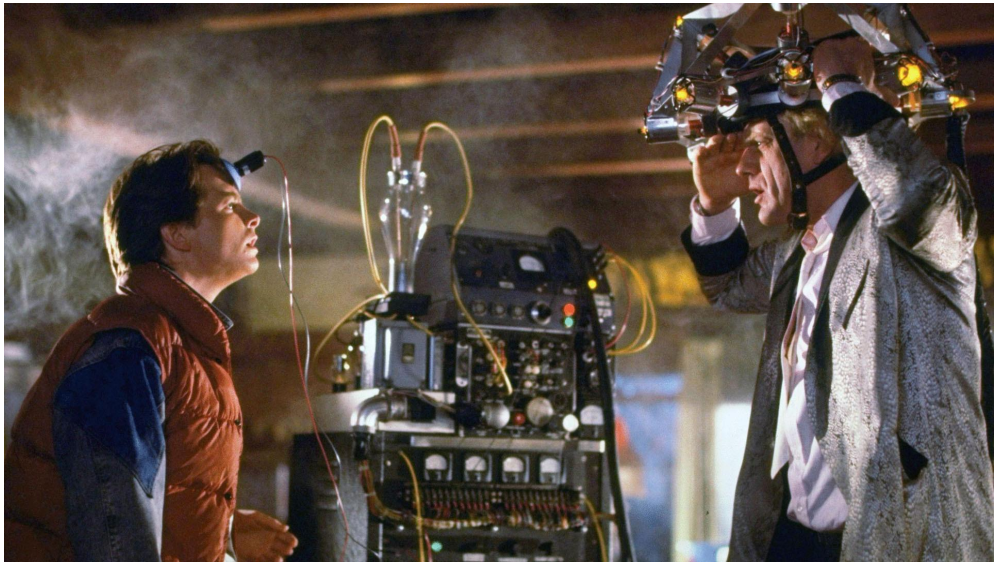
MISTAKE: Importing ALL of RxJS

```
//No
import * as Rx from 'rxjs';

//Wrong
import { Observable, Subject } from 'rxjs';

//Bad
import { Subject } from 'rxjs/Rx';
```

Include Just What You Need

```
//Pull in types you need individually
import { Observable } from 'rxjs/Observable';
import { Subject } from 'rxjs/Subject'
import { async } from 'rxjs/scheduler/async';

//add operators in your app.module.ts
import 'rxjs/add/operator/map';
import 'rxjs/add/operator/filter';
```

Back to Our Pun App



We have an Observable of text input changes - keywordInputChange$

… give us the puns we're looking for!

To Get Our Data, We've Created "PunService"

app.module.ts ● | puns.component.ts

```
33    providers: [
34      PunService
35    ],
```

puns.component.ts ● | app.routing.ts ✕ | app.module.ts

```
11  export class PunsComponent implements OnInit {
12
13    constructor(private puns: PunService) { }
```

For Our Data Source, We Created PunService

```
1   import { Injectable } from '@angular/core';
2   import { Observable } from 'rxjs/Observable';
3
4   export interface Pun {
5     id: number,
6     pun: string,
7     answer: string
8   };
9
10  @Injectable()
11  export class PunService {
12    suggestKeywords(partial: string): Observable<string[]> {
13      const keywords = suggestKeywords(partial);
14      return Observable.of(keywords);
15    }
```

Take Observable of Textbox Changes and Get a List of Suggested Keywords

```
export class PunsComponent implements OnInit {

    keywordInputChange$ = new Subject<string>();

    keyword$ = this.keywordInputChange$
        .switchMap(partial => this.puns.suggestKeywords
        (partial));
```

switchMap

Converts the value to a new Observable, then switches to that Observable (unsubscribing from any previous ones it might have made)

Operators: Why switchMap?

http://reactivex.io/rxjs

RxJS Home Manual Reference Source Test dark theme light theme

- C AsyncSubject
- C BehaviorSubject
- C Notification
- C Observable
- C ReplaySubject
- C Scheduler
- C AnonymousSubject
- C Subject
- C SubjectSubscriber
- C Subscriber
- C Subscription
- I ObservableInput
- I Observer
- I SubscribableOrPromise
- I TeardownLogic
- T Rx.Scheduler
- T Rx.Symbol

» Full reference

Read detailed documentation on each operator

» View examples

Read our tutorials on using RxJS in real applications

Do you need to find an operator for your problem? Start by choosing an option from the list below:

- I have one existing Observable, and...
- I have some Observables to combine together as one Observable, and...
- I have no Observables yet, and...

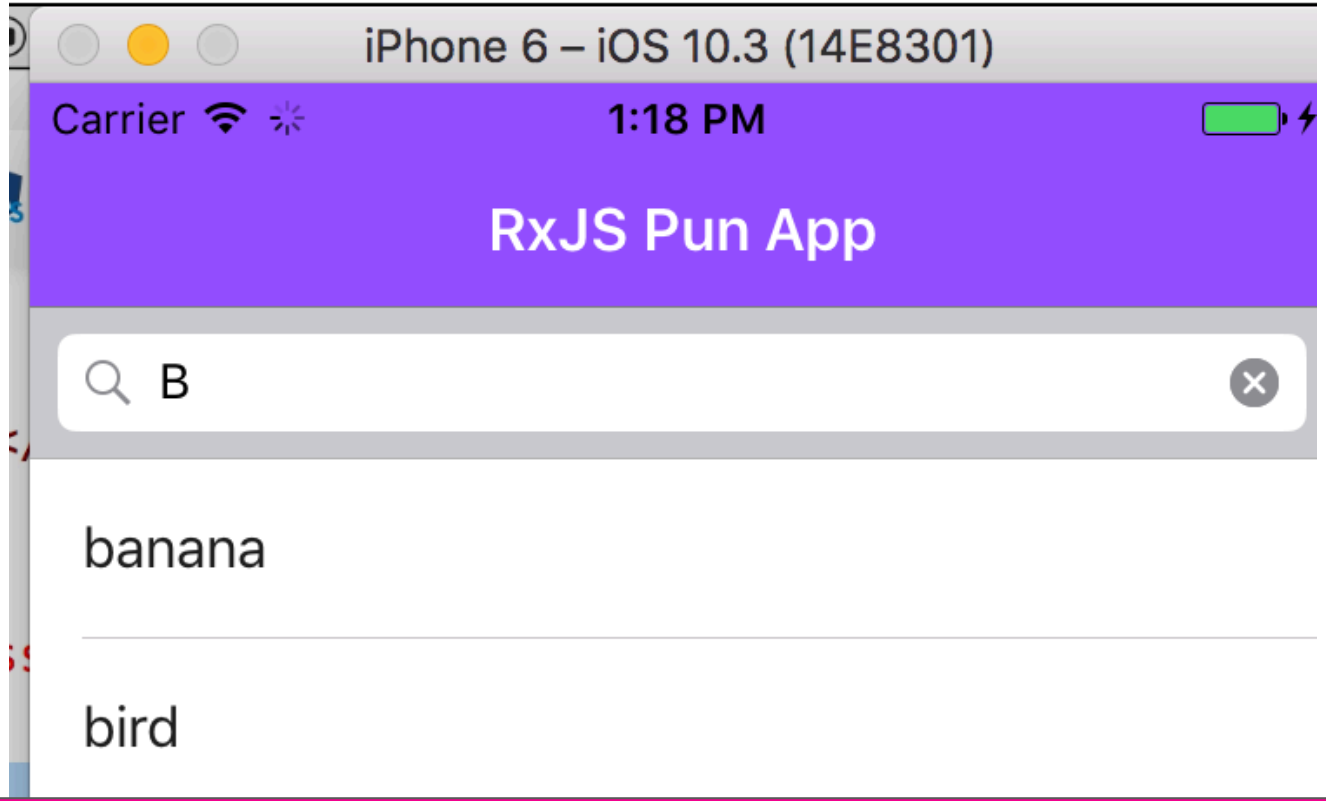
Writing Suggested Keywords to the View

```
<ListView [items]="source" (loaded)="onLoaded(event)"
(itemLoading)="onItemLoading(event)" (itemTap)="onItemTap
(event)" class="list-group">
    <ng-template let-item="item" let-odd="odd"
    let-even="even">
      <StackLayout>
        <Label [text]="item.title"
        class="list-group-item"></Label>

      </StackLayout>
    </ng-template>
</ListView>
```

Writing Suggested Keywords to the View

```
<ListView [items]="keyword$ | async" class="list-group">
    <ng-template let-keyword="item">
        <Label [text]="keyword"
        class="list-group-item"></Label>
    </ng-template>
</ListView>
```

Writing Suggested Keywords to the View



Get List of Puns from List of Keywords

```
export class PunsComponent implements OnInit {

    keywordInputChange$ = new Subject<string>();

    keyword$ = this.keywordInputChange$
        .switchMap(partial => this.puns.suggestKeywords
        (partial))

    pun$ = this.keyword$
        .switchMap(keywords => this.puns.getPuns(keywords));
```

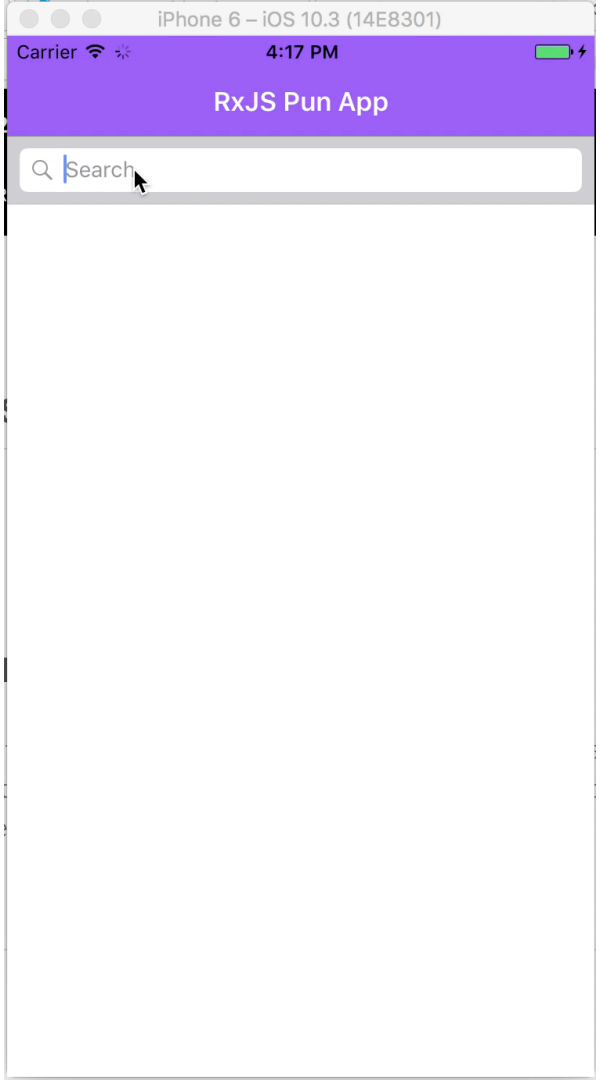
Displaying Puns

```
<ListView [items]="keyword$ | async" class="list-grou
  <ng-template let-keyword="item">
      <Label [text]="keyword"
      class="list-group-item"></Label>
  </ng-template>
</ListView>
```

```
<ListView [items]="pun$ | async" class="li
  <ng-template let-pun="item">
      <Label [text]="pun.pun + ' ' + pun.ans
      class="list-group-item"></Label>
  </ng-template>
</ListView>
```

Let's Share keyword$

```
export class PunsComponent implements OnInit {

    keywordInputChange$ = new Subject<string>();

    keyword$ = this.keywordInputChange$
        .switchMap(partial => this.puns.suggestKeywords
        (partial))
        .share();

    pun$ = this.keyword$
        .switchMap(keywords => this.puns.getPuns(keywords));
```


iPhone 6 – iOS 10.3 (14E8301)
Carrier
4:17 PM
RxJS Pun App
Search

The Benefits of Same Shaped-ness

Let’s Demo!

Learning & Using RxJS

- How to create an Observable
- Best practices for importing and using RxJS
- How to choose operators and find documentation
- How to avoid unwanted subscriptions
- The benefits of “same-shapedness”

What’s Next for RxJS 6 & 7?

What’s Next for RxJS 6 & 7?

T-Rex => RxJS 6/7

~30kb => ~3kb

(minified, bundled, and gzipped)

How did it get so small?

- Implemented a lot of operators in terms of other operators
- Turn operators into one line vs 40-50 lines of code
- Ex. reduce operator is just scan and take last
- Ex. toArray is just a specialty of reduce

Patch Operators vs Lettable Operators

- Patch operators
  - import ‘rx/add/operator/*map*’
  - Libraries and code breaking
  - Add operators on to the prototype
  - No tree shaking
- Lettable operators
  - aka operator functions
  - Higher order functions
  - Yay tree shaking
  - Yay functional programming

Lots more to come…

Resources

https://github.com/ladyleet/rxjs-nativescript-ddays

https://github.com/ladyleet/rxjs-test

http://reactivex.io/rxjs

tracy@thisdot.co

twitter.com/ladyleet

Thank You!