Angular 2

Observables, Change Detection, AsyncPipe, Forms, Rx.js

Geoff Filippi / @geofffilippi

Geoff Filippi

Application Architect

Oildex

A cloud service company for oil and gas

- 1 year

Formerly:

Time Warner Cable

- 12 years

Experience

📞

- Worked on streaming media (Voice over IP), 6 years
- 5 million phone customers

Experience

- Worked on video and streaming video, 4 years

Projects

twctv.com

- Video streaming website
    - backbone.js
- Video streaming Set-Top Box (STB) web application

Oildex Projects

- Rewrite 10+-year-old apps
- Angular 1.4
  - New router
  - ES5
- No Angular 2, yet
- No ES6 or Typescript, yet

We will cover

- Fundamentals
- Related concepts
- Angular 2

Fundamentals

A *brief* overview of reactive fundamentals

- Reactive programming
- Functional Reactive Programming (FRP)
- Observer design pattern

Related concepts

- `Object.observe`
- Promises
    - Other Async concepts
- RxJS

Observables in Angular 2

- Forms
- `Http`
- AsyncPipe

Fundamentals

Overview of Reactive Programming

- Data flows
- Propagation of change

Reactive Manifesto

- Responsive
- Resilient
- Elastic
- Message Driven

(FRP) Functional Reactive Programming

- Reactive Programming

and

- Functional Programming

Related Concepts

Observer Design Pattern

- Subject tracks observers
- Subject notifies observers of state changes

Terminology

- Subject, Observable, Producer
- Observer, Consumer

`Object.observe`

Not the same as Observables

- Proposal withdrawn
  - Formerly a Stage 2 Proposal
- `Proxy`

Criticisms of `Object.observe`

- Big changes to Object internals
- Bad performance
- Not well-supported or adopted

But really,

- Ecosystem moved in different direction

`Observable`

- Stage 1 Current Proposal
- Description
- Specification

Differences between Observables and `Object.observe`

Creating an Observable

The hard way

```
let myObservable = new Observable(observer => {
  const token = doAsyncAction((err, value) => {
    if (err) }
      observer.error(err);
    } else {
      observer.next(value);
      observer.complete();
    }
  });

  return () => {
    cancelAsyncAction(token);
  };
{);
```

Compare

- Observables
- Promises
- Events
- callbacks

Promises vs. Observables

- Promise is like a async variable
- Observable is like a async array

Promises vs. Observables

Promises

- Single value
- Cannot be cancelled
- Not lazy
- Good for some AJAX calls
    - Unless you want to cancel them
- Catch, Finally

Promises vs. Observables

Observables

- Streams
- Can be unsubscribed
- Lazy, until you call `.subscribe()`
- Better for events, WebSockets
  - Animations, AJAX you want to cancel
- Can by retried
- Catch, Finally

Bridging Observables

Easy way to get an observable

You can turn other async concepts into an observable using a helper

- Promises
  - `.from()`
- Generator
  - `.from()`
- Callbacks
  - `.fromCallback()`

Bridging Observables

Easy way to get an observable

- Events
  - `.fromEvent()`
- DOM
  - AJAX
  - WebSockets `.fromWebSocket()`
- Other observable implementations
  - bacon.js
  - kefir.js

Bridging Observables

You can even turn non-async concepts into an observable using a helper

- Arrays
  - `.of()`

Working with Promises vs. Observables

- `.then()` becomes `.subscribe()`

Working with Promises vs. Observables

Observables do not need to complete, but...

If you need to do something when an observable completes, pass an optional complete handler

RxJS

- v5.0.0-beta.1
- Observable
- Subject
- Implementation used in Angular 2

Angular 2

Angular 2

- Forms
- `Http`
- AsyncPipe
- Change detection

Forms

Forms are observable

```
this.myForm = fb.group({
  'sku':  ['', Validators.required]
});

this.sku = this.myForm.controls['sku'];

this.sku.valueChanges.subscribe(
  (value: string) => {
    console.log('sku changed to: ', value);
  }
);

this.myForm.valueChanges.subscribe(
  (value: string) => {
    console.log('form changed to: ', value);
  }
```

`Http`

- Cancel requests
- Stream results
  - "type ahead"
- Returns observable

Http

```
getRandomQuote() {
  this.http.get('http://localhost:3001/api/random-quote')
    .retry(3)
    .map(res => res.text())
    .subscribe(
      data => this.randomQuote = data,
      err => this.logError(err),
      () => console.log('Random Quote Complete')
    );
}
```

Pipes

- Like Angular 1 `filter`

AsyncPipe

- Can subscribe to observables

```
timerAsync = Observable.interval(1000).startWith(0);

<h2>Current Total (Async Pipe): {{timerAsync | async}}</h2>
```

Change Detection

New in Angular 2:

- Apps are reactive
- Change detection is directional
- You can skip walking parts of the tree
  - Immutable objects
  - Observables

`ngrx`

Reactive Extensions for Angular 2

`ngrx/store`

Can be used to implement elm/redux-style applications in Angular 2

Questions?