Musical Synthesis with Angular 2 and RxJS

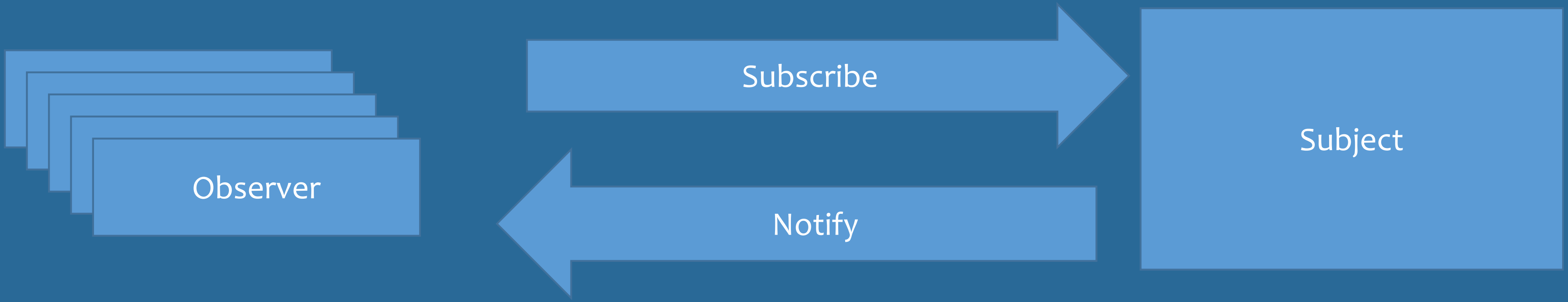
Ken Rimple
Chariot Solutions

Angular NYC

November 15, 2016

Angular 2’s secret weapon: RxJS

- Provides a *reactive* event stream
- Uses the age-old Observer pattern



RxJS is a Functional Library

- Like “Lodash for Events” (Microsoft’s Docs)
- As data arrives, you can
  - Filter it
  - Transform it
  - Mute it
  - Combine it
  - Split it
  - Debounce, limit, or take specific parts of it
- This is because Observables provide *streams* of data

RxJS Observable Basics

```
// provides a stream of numbers
let numberSeq = Observable.of([1, 2, 3, 4, 5]);

// transform them to their square values
numberSeq.subscribe(
   (value) => {
       console.log(value * value);
    }
);
```

So, it’s an iterator… but PUSHED

- Unlike Generators/Iterators…
- The observable provides values for us when they arrive

- RxJS Observables are Reactive
  - Respond to stimuli
  - React accordingly

Observables from Any Source

```
// From a network query
let stream = http.get(‘/foo/bar’);

// From a WebSocket
let stream = Observable.fromWebSocket(…);

// From a generated stream
let stream = new Subject<MusicalNote>();
```

Observable Classes

- **Observable** – provides general utilities to generate Observables from various sources
- **Subject** – an Observable that can generate values AND provide a subscription stream
- **Operator** – something that transforms or processes the stream of data emitted by the observable
- **Subscription** – a reference to a subscription to an Obsevable, used to unsubscribe

Observables –vs- Promises

- Use Promises when
  - You need a single result
  - The source API provides one to use
- Use Observables when
  - You need a stream of data
  - You have one source of information, many listeners / reactors
  - You can take advantage of the functional transforms available to RxJS
- Convert a Promise to an Observable?
  - Why? Promises are easy to use, and for a single result are fine
  - Be *pragmatic*

Observables in Angular 2

- Http client API (strange case, actually)
- The Router API's ActivatedRoute – emits events for changes in
  - Url
  - Query Parameters
  - Params / path
  - etc…
- Form Controls – valueChanges stream
- Component's @Output EventEmitters (are actually subclassed Observables)
- AsyncValidators can be returned as Observables

My Example

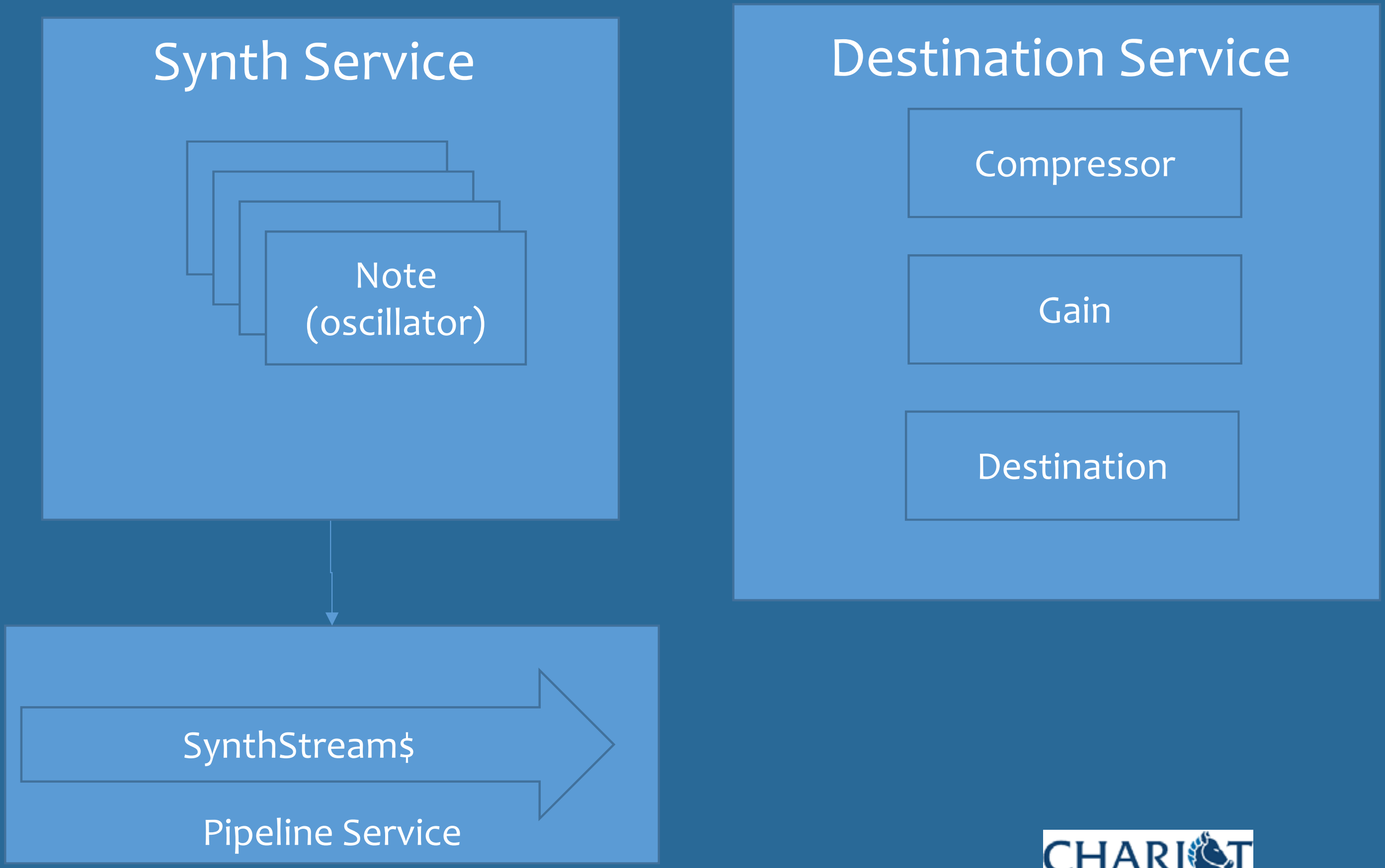
- A dinky little musical synth
  - Notes provided from several sources (web component, MIDI)
  - Creates a stream of note and control values using RxJS
  - Parts of the system subscribe to the stream and modify the synth
- Why???
  - Music geek, synth geek
  - Wanted to show a non-traditional case
  - Wanted to see latency in RxJS-based UI (not seeing it!)
  - Wanted to show various transformations

The Synth – Block Diagram

Inputs

Various Angular 2 Keyboard Components

Touch Board (custom MIDI sketch)



Wiring the Synth – a PipelineService

- Responsible for setting up a synthStream$, a Subject Observable of SynthMessage objects
  - SynthMessages can be
    - SynthNoteOn (we’ve pressed a key)
    - SynthNoteOff (a key was released)
    - VolumeChange (louder, softer)
    - WaveformChange (sine, sawtooth, etc)
    - etc…
- Wires up the synthesizer, destination (output) services

Code sample - wiring

```
begin() {
  this.audioOutputService.setup(…);
  this.synthesisService.setup(…);
  // oh yea, we have a drum machine…
  this.drumPCMTriggeringService.setup(…);
  this.midiInputService.setup()
  …
}
```

Features

- Single stream for all note and control data
- Various services “sip” from the synth data stream and
  - Fire off oscillators (SynthService)
  - Trigger drum machine events (DrumMachineService)
  - Change default waveforms (SynthService)
  - Change audio volume (AudioOutputService)
  - Record a stream of notes and play it back (SequencerService)
- Makes it easy to reconfigure the synthesizer by adding modules to adjust to the changes in the stream

Demonstration

Wrap-up / Q&A

- Reach me at @krimple on Twitter
- Sourcecode online!
  - http://github.com/krimple/nyc-ng2-synth-demo
  - Not perfect, but a starting example of what you can do with Observables and Angular 2!
- Slides online!
  - TBD
- Chariot Solutions Angular 2 training course
  - Onsite, in-person training in Angular 2.x, React, Angular 1, more
  - http://chariotsolutions.com/training
  - (Shameless plug) Chariot Solutions can help you level up in Angular 2 and other technologies such as Scala, Akka, Node.JS, Angular 2, Spring, Spark, more…