Angular

by Jason Swett

Day 3 Overview

- Observables
  - What is an Observable?
  - What is RxJS?
  - Observable vs. Observer
  - Subscribe
  - Subject
  - Transforming Observables
  - Filtering Observables
  - Simplest-Possible-Observer Lab
  - Wikipedia Search Lab

Observables

Helpful Links

- http://reactivex.io/documentation/observable.html
- http://reactivex.io/documentation/operators.html
- https://gist.github.com/staltz/868e7e9bc2a7b8c1f754
- https://medium.com/@benlesh/learning-observable-by-building-observable-d5da57405d87#.w00b5yws4
- https://egghead.io/lessons/rxjs-creating-an-observable
- https://egghead.io/courses/build-an-angular-2-instant-search-component

What is an Observer?

- **An Observer is a piece of behavior that can react to items in a *stream*** (when I say “stream”, just think of a list of items fed one at a time)

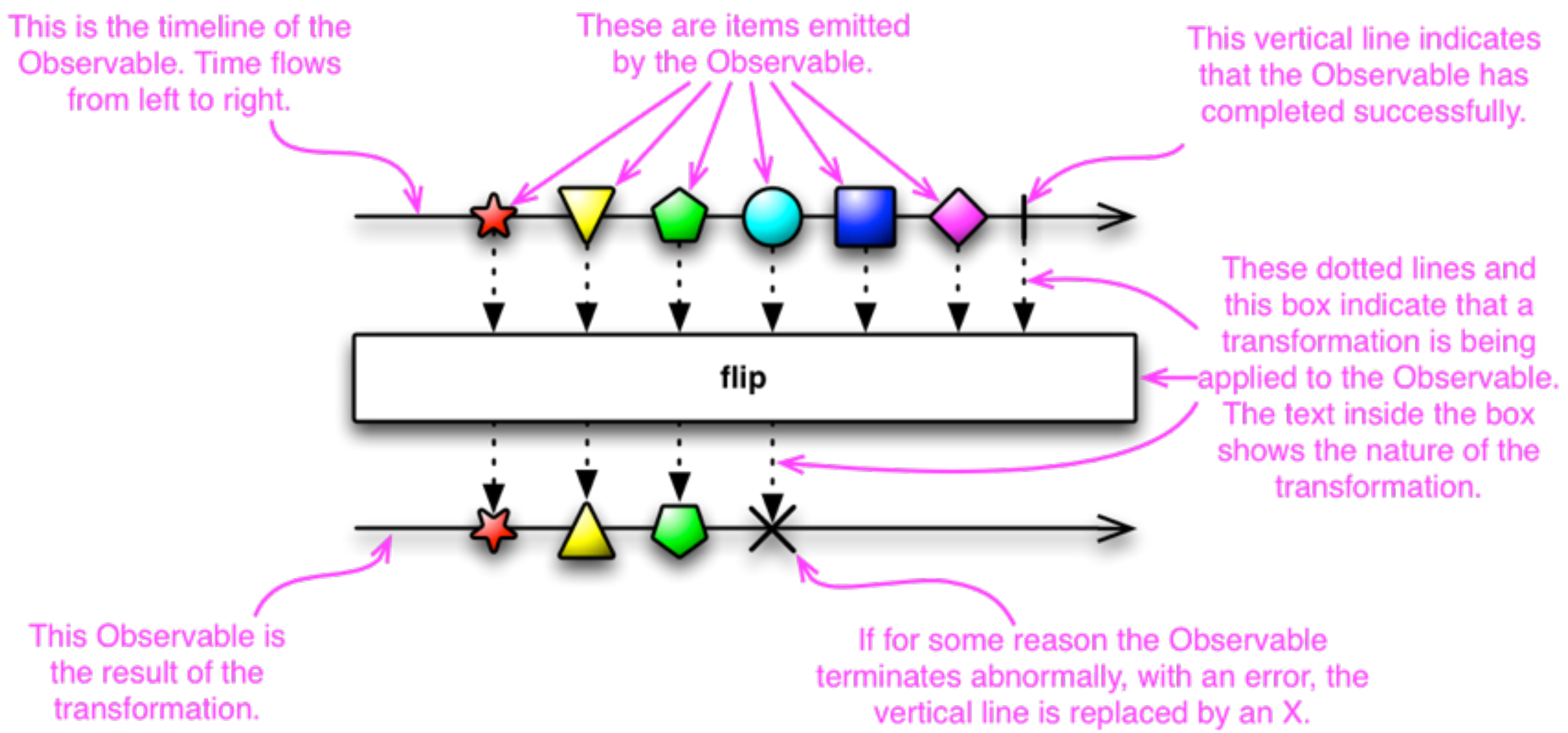
- This is in contrast to promises, which can only deal with a single response

- Part of RxJS

What is RxJS?

- ReactiveX is a language-agnostic API for asynchronous programming with observable streams
- RxJS = Reactive Extensions for JavaScript
- RxJS is a set of libraries
- Independent of Angular, a world unto itself

Observable Diagram


This is the timeline of the Observable. Time flows from left to right.
These are items emitted by the Observable.
This vertical line indicates that the Observable has completed successfully.
These dotted lines and this box indicate that a transformation is being applied to the Observable. The text inside the box shows the nature of the transformation.
flip
This Observable is the result of the transformation.
If for some reason the Observable terminates abnormally, with an error, the vertical line is replaced by an X.

Observable vs. Observer

- The Observable is the thing being watched
- The observer is the thing watching the Observable

Important terms

- An observer ***subscribes*** *to* an Observable
- An Observable ***emits*** items or sends ***notifications*** to its observer(s)
- (The way Observable sends notifications to its observer(s) is by calling the observers' methods, e.g. `observer.next`.)

Observer.create

Function signature for Rx.Observer.create:

```
Rx.Observer.create(
   [onNext],
   [onError],
   [onCompleted]
)
```

Observer: `onNext`

- A function that gets called when a value is emitted from the Observable
- Takes a single argument, the value
- Example:

```
function (x) {
  console.log('Next: ' + x);
}
```

Observer: `onError`

- A function that gets called when an error is raised in the Observable
- Takes a single argument, the error
- Example:

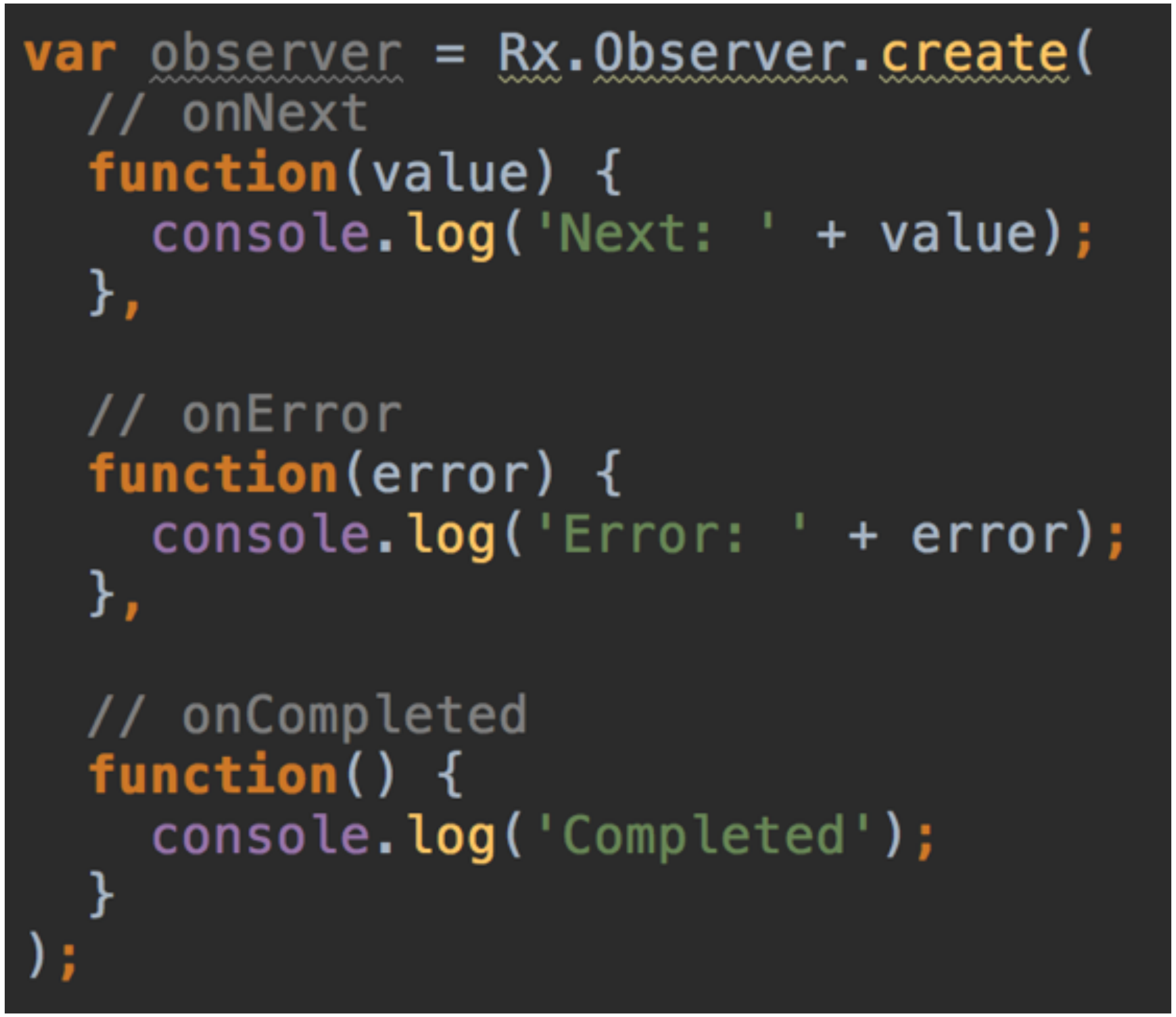
```
function (err) {
  console.log('Error: ' + err);
}
```

Observer: `onCompleted`

- A function that gets called when the Observable is completed
- Takes a single argument, the error
- Example:

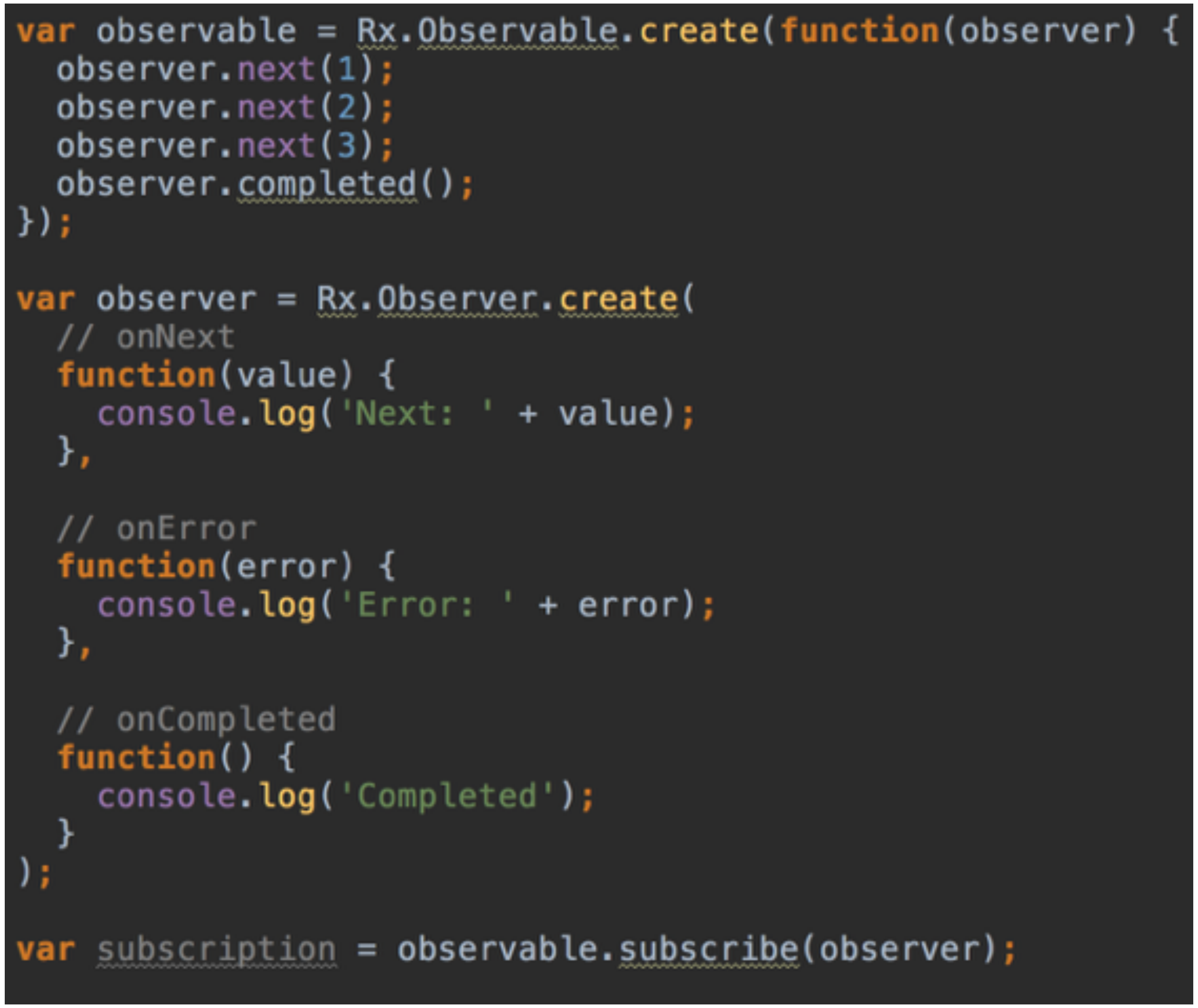
```
function () {
  console.log(‘Completed');
}
```

Observer Example



```
var observer = Rx.Observer.create(
  // onNext
  function(value) {
    console.log('Next: ' + value);
  },

  // onError
  function(error) {
    console.log('Error: ' + error);
  },

  // onCompleted
  function() {
    console.log('Completed');
  }
);
```

Observer Example With Observable



```
var observable = Rx.Observable.create(function(observer) {
  observer.next(1);
  observer.next(2);
  observer.next(3);
  observer.completed();
});

var observer = Rx.Observer.create(
  // onNext
  function(value) {
    console.log('Next: ' + value);
  },

  // onError
  function(error) {
    console.log('Error: ' + error);
  },

  // onCompleted
  function() {
    console.log('Completed');
  }
);

var subscription = observable.subscribe(observer);
```

Subscribe

- "The Subscribe operator is the glue that connects an observer to an Observable."
- Example: `observable.subscribe(observer);`
- `subscribe` is an instance method of Observable. It can take an **observer** as an argument.
- http://reactivex.io/documentation/operators/subscribe.html

Subject

- "A Subject is a sort of bridge or proxy that is available in some implementations of ReactiveX that acts **both as an observer and as an Observable**."
- Example: `term$ = new Subject<string>();`
- We'll make use of a Subject soon

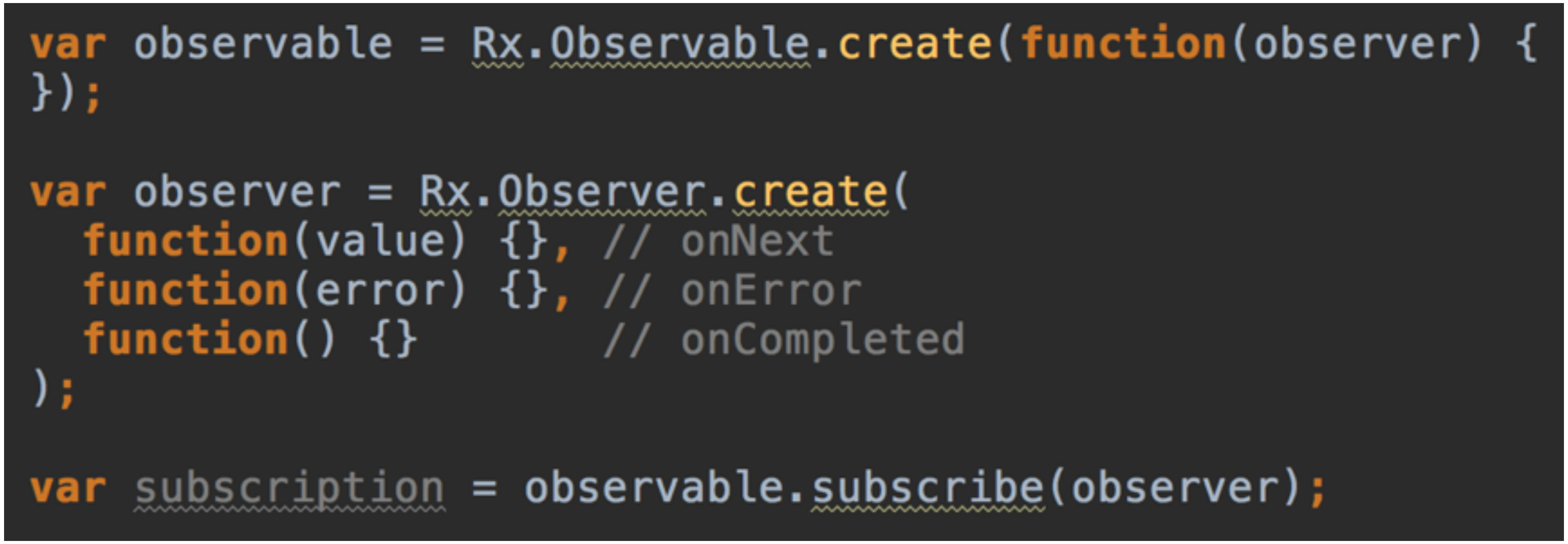
Transforming Observables

- Buffer — periodically gather items from an Observable into bundles and emit these bundles rather than emitting the items one at a time
- FlatMap — transform the items emitted by an Observable into Observables, then flatten the emissions from those into a single Observable
- GroupBy — divide an Observable into a set of Observables that each emit a different group of items from the original Observable, organized by key
- **Map — transform the items emitted by an Observable by applying a function to each item** (most common in my experience)
- Scan — apply a function to each item emitted by an Observable, sequentially, and emit each successive value
- Window — periodically subdivide items from an Observable into Observable windows and emit these windows rather than emitting the items one at a time
- http://reactivex.io/documentation/operators.html

Filtering Observables

- **Debounce — only emit an item from an Observable if a particular timespan has passed without it emitting another item**
- **Distinct — suppress duplicate items emitted by an Observable**
- ElementAt — emit only item n emitted by an Observable
- Filter — emit only those items from an Observable that pass a predicate test
- First — emit only the first item, or the first item that meets a condition, from an Observable
- IgnoreElements — do not emit any items from an Observable but mirror its termination notification
- Last — emit only the last item emitted by an Observable
- Sample — emit the most recent item emitted by an Observable within periodic time intervals
- Skip — suppress the first n items emitted by an Observable
- SkipLast — suppress the last n items emitted by an Observable
- Take — emit only the first n items emitted by an Observable
- TakeLast — emit only the last n items emitted by an Observable
- http://reactivex.io/documentation/operators.html

Observer Lab: Fill in the Blanks

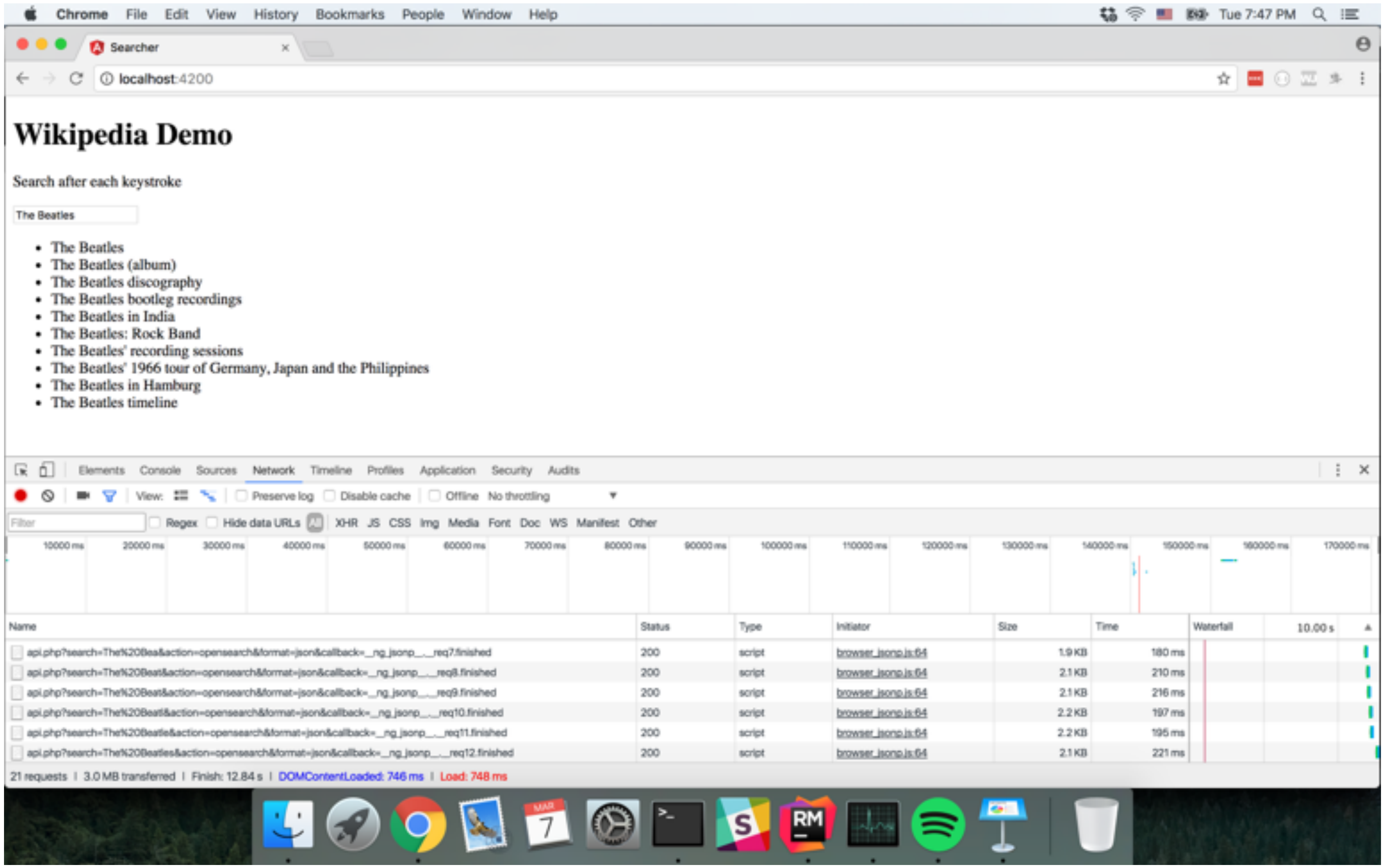


```javascript
var observable = Rx.Observable.create(function(observer) {
});

var observer = Rx.Observer.create(
  function(value) {}, // onNext
  function(error) {}, // onError
  function() {}        // onCompleted
);

var subscription = observable.subscribe(observer);
```

Observer Lab Instructions

1. `fill-in-observable-blanks` folder in labs
2. Get the `onNext` callback to log something to the console
3. Get the `onCompleted` callback to log something to the console
4. Get the `onError` callback to log something to the console

Wikipedia Search Lab



Wikipedia Demo

Search after each keystroke

The Beatles

- The Beatles
- The Beatles (album)
- The Beatles discography
- The Beatles bootleg recordings
- The Beatles in India
- The Beatles: Rock Band
- The Beatles' recording sessions
- The Beatles' 1966 tour of Germany, Japan and the Philippines
- The Beatles in Hamburg
- The Beatles timeline

Name	Status	Type	Initiator	Size	Time
api.php?search=The%20Bea&action=opensearch&format=json&callback=__ng_jsonp____req7.finished	200	script	browser_jsonp.js:64	1.9 KB	180 ms
api.php?search=The%20Beat&action=opensearch&format=json&callback=__ng_jsonp____req8.finished	200	script	browser_jsonp.js:64	2.1 KB	210 ms
api.php?search=The%20Beat&action=opensearch&format=json&callback=__ng_jsonp____req9.finished	200	script	browser_jsonp.js:64	2.1 KB	216 ms
api.php?search=The%20Beatl&action=opensearch&format=json&callback=__ng_jsonp____req10.finished	200	script	browser_jsonp.js:64	2.2 KB	197 ms
api.php?search=The%20Beatle&action=opensearch&format=json&callback=__ng_jsonp____req11.finished	200	script	browser_jsonp.js:64	2.2 KB	195 ms
api.php?search=The%20Beatles&action=opensearch&format=json&callback=__ng_jsonp____req12.finished	200	script	browser_jsonp.js:64	2.1 KB	221 ms

21 requests | 3.0 MB transferred | Finish: 12.84 s | DOMContentLoaded: 746 ms | Load: 748 ms

debounceTime

- “only emit an item from an Observable if a particular timespan has passed without it emitting another item”
- Can be used to prevent the front-end from clobbering the server with too many requests which would put an unnecessary load on the server
- Example:

```
this.myService.getList()
  .debounceTime(500)
  .subscribe(res => console.log(res));
```

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

distinctUntilChanged

- "suppress duplicate items emitted by an Observable"
- Similarly to `debounceTime`, can be used to prevent the front-end from clobbering the server with too many requests which would put an unnecessary load on the server
- Example:

```
this.myService.getList()
  .distinctUntilChanged()
  .subscribe(res => console.log(res));
```

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

Wikipedia Search Lab

1. Start with the Wikipedia search project (`wikipedia-search` directory)

2. Change the search input from calling `search` to calling `next` on an observer that calls `search`

3. Add `debounceTime` to prevent too many requests from being fired

4. Add `distinctUntilChanged` to further prevent unnecessary requests

Class Review

Day 1 Review

- Intro to Angular
  - Why Angular?
  - Lab: Angular Hello World
- TypeScript
  - ES5/ES6/TypeScript Differences
  - Decorators
  - Lab: ES5 to ES6 to TypeScript

Day 1 Review

- Directives
  - Structural Directives
  - Attribute Directives
  - @Input
  - Lab: Write a Directive
- Modules
  - Imports
  - Declarations
  - Providers
  - Bootstrap
- Dependency Injection

Day 1 Review

- Components
  - Lab: Write a Component
- Data Binding
  - Interpolation
  - Property Binding
  - Variables
  - Template Statements
  - Two-Way Binding
  - Lab: Data Binding

Day 2 Review

- Angular CLI
  - Lab: Contact Management App (Part 1)
  - Lab: Contact Management App (Part 2)
- Services
  - Lab: Refactor Contact List to Use Service
- Routing
  - Lab: Add Routing to Contact Management App

Day 2 Review

- Testing with Jasmine
  - Benefits of Automated Testing
  - Jasmine Overview
  - Jasmine Syntax
  - Lab: Number Adder
- Jasmine + Angular
  - Lab: Slug Service Test

Day 3 Review

- Observables
  - What is an Observable?
  - What is RxJS?
  - Observable vs. Observer
  - Subscribe
  - Subject
  - Transforming Observables
  - Filtering Observables
  - Simplest-Possible-Observer Lab
  - Wikipedia Search Lab

Thanks

- Thank you for spending these three days with me
- Thank you for this opportunity to share my knowledge and experience
- Thank you for making me feel welcome in The Netherlands, **my new favorite country on the planet**!

Evaluation Time

https://goo.gl/gEZ9sd