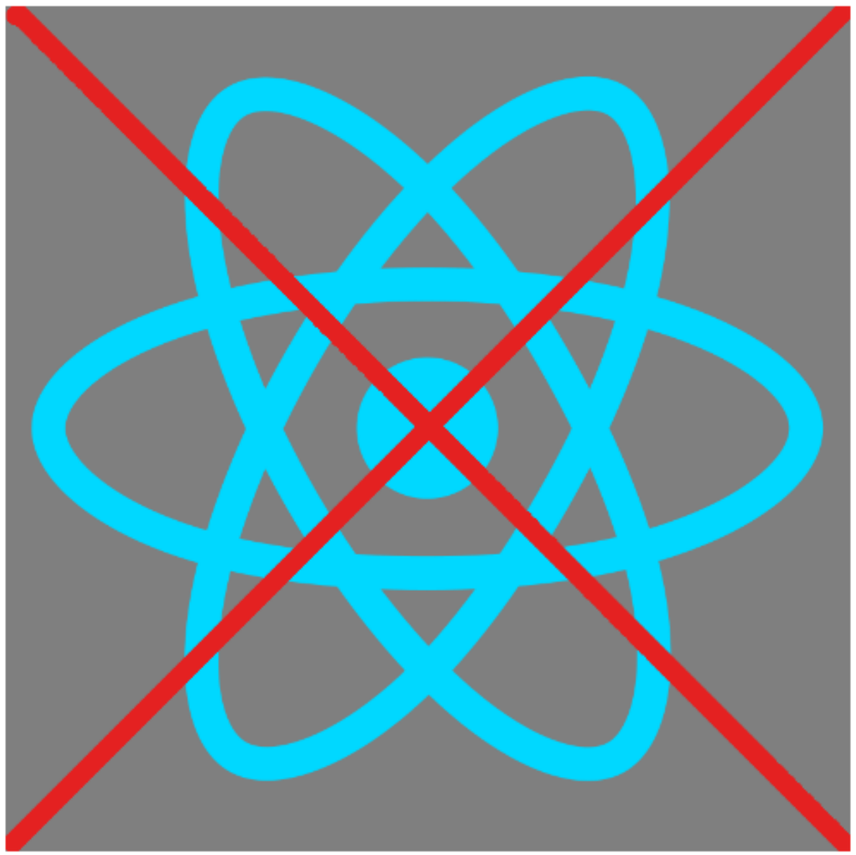

DEVNATION
June 21-25, 2015 Boston, Massachusetts

Brian Leathem @brianleathem

This is not.



Reactive programming with Rx.js

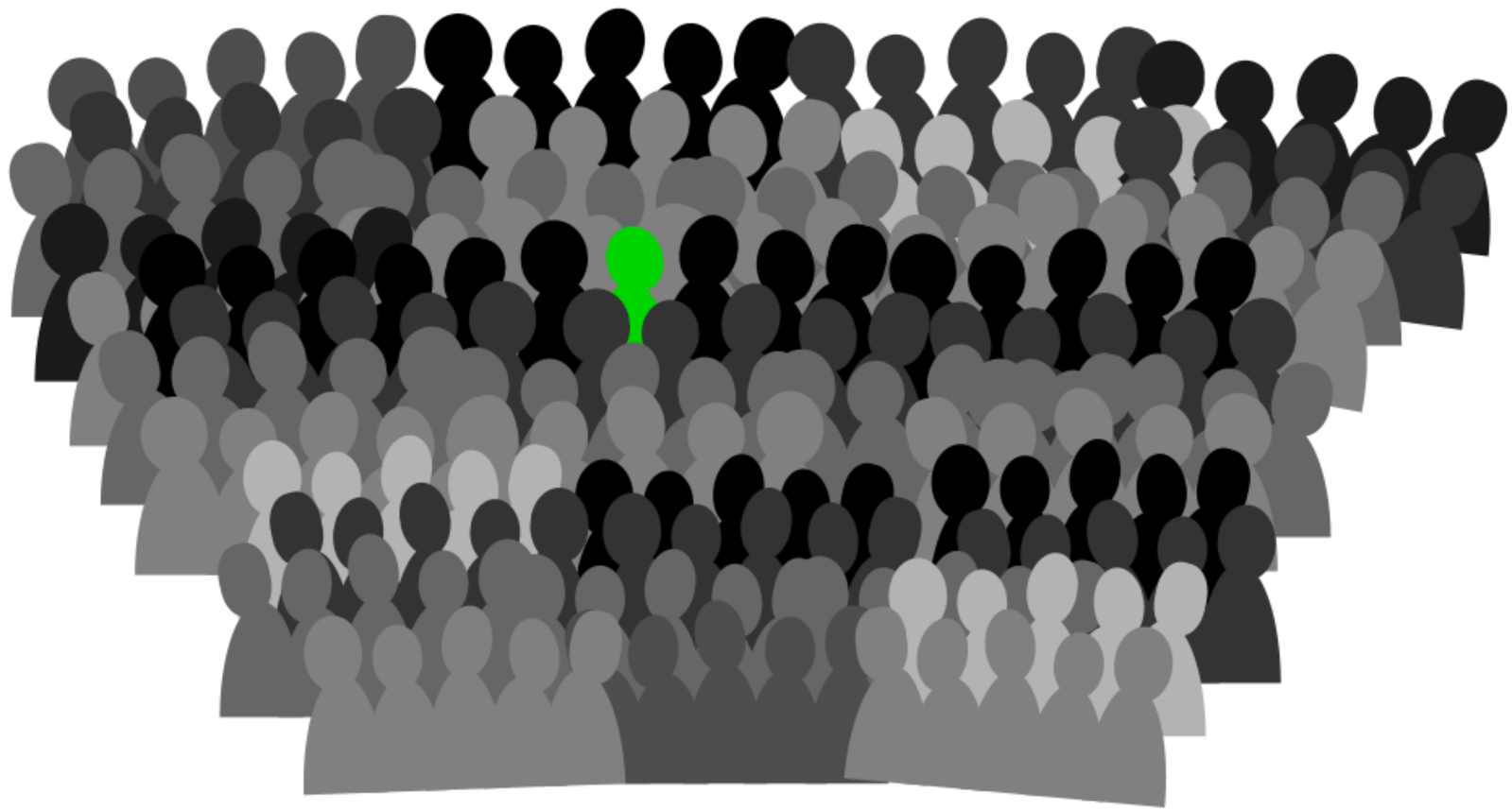
The plan

- Async JavaScript review
- Observables are what?
- Use cases and examples

Who am I?

- Brian Leathem
- Software Engineer @ Red Hat
- Works on developer tools and frameworks

Who are you?



Async JavaScript

Async Javascript

- XHR
- Animations
- Timeout/Interval
- Event Listeners

Callbacks

Asynchronous javascript:

```
function asyncTask(args, callback) {
  // do stuff
  callback()
}
```

Invoking `asyncTask`

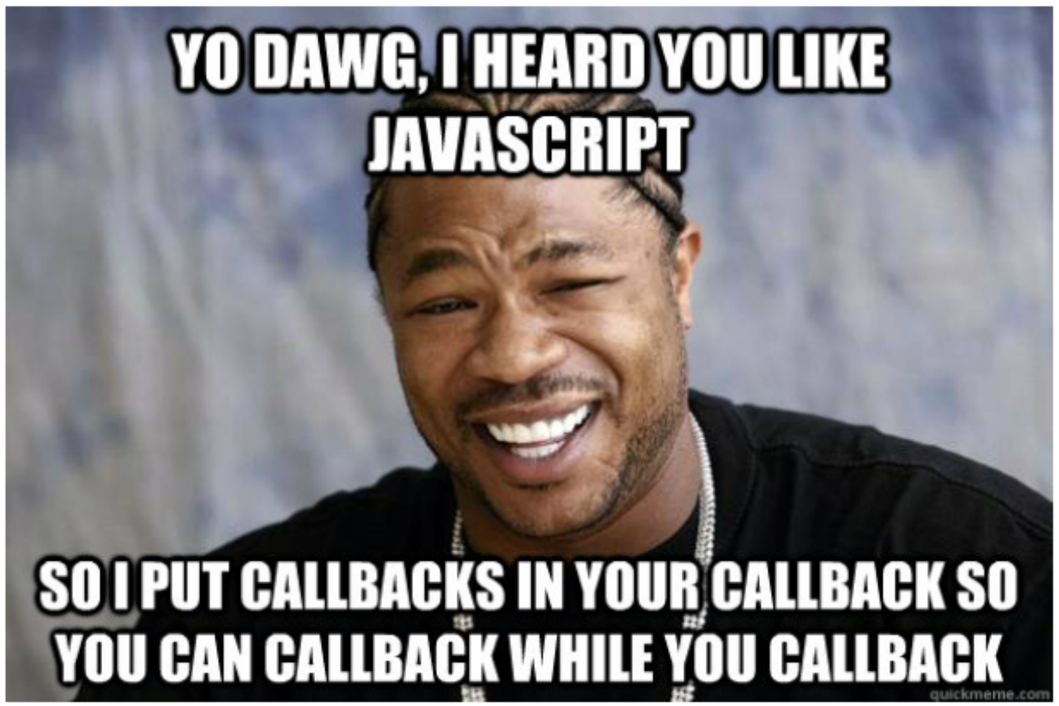
```
asyncTask(args, function() {
  // task is done here!
})
```

Anonymous function:

- simple
- concise
- well-accepted pattern

Nesting async calls - serial

```
asyncTask1(args, function() {
  asyncTask2(args, function() {
    asyncTask1(args, function() {
      // task is done here!
    })
  })
})
```



Simultaneous callbacks - parallel

```
var result1=false; result2=false; // state!!

asyncTask1(function(){a1=true; doAction()}
asyncTask2(function(){a2=true; doAction()}

function doAction() {
  if (a1 && a2) {
    …
  }
}
```

Promises to the rescue!

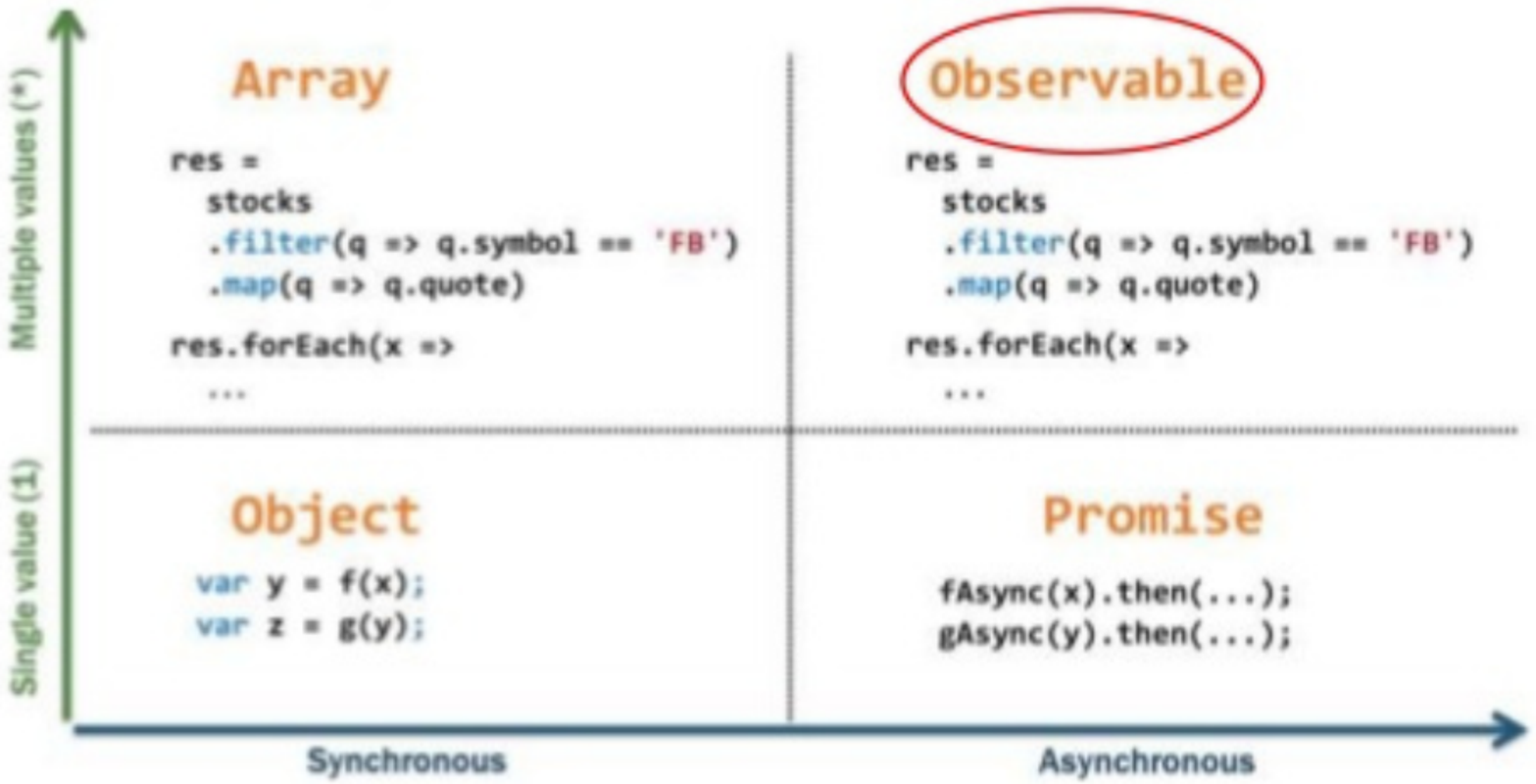
Serial:

```
asyncTask1.then(asyncTask2).then(function() {
  // success
}, function(err) {
  // error
})
```

Parallel:

```
Promise.all(asyncTask1, asyncTask2).then(...)
```

What about events?



From: http://www.slideshare.net/stefanmayer13/functional-reactive-programming-with-rxjs

Observable what?!

Think of an observable as a *collection-in-time*

Same *functional* tools apply

- `forEach`
- `filter`
- `map`
- `...`

Imperative vs. Functional

Iterating over an array

Imperative:

```
for (var i=0; i < a.length; i++) {
  item = a[i];
  // item.doAction()
}
```

Functional:

```
a.forEach(function(item) {
  // item.doAction()

})
```

A Collection

See the Pen Collection | Iden by Brian Leathem (@bleathem) on CodePen.

An Observable

See the Pen Observable by Brian Leathem (@bleathem) on CodePen.

Rx.js

Reactive Extensions for JavaScript

> ...is a set of libraries to compose asynchronous and event-based programs using observable collections and Array#extras style composition in JavaScript

Some basic transforms

- `map`
- `reduce`
- `mergeAll`
- `reduce`
- `zip`

map

```
.map(function(x) {
  return {
    id: x.id
  , color: 'green'
  , size: x.size
  , type: 'square'
  };
});
```

Map each shape

into a green square

of the same size

Collection `map`

See the Pen Operating on a Collection by Brian Leathem (@bleathem) on CodePen.

Observable `map`

See the Pen Map an Observable by Brian Leathem (@bleathem) on CodePen.

mergeAll

```
.map(function(x) {
    var y = _.clone(x);
    y.id = y.id + 80;
    y.color = 'green';

    var z = _.clone(x);
    y.size = y.size / 1.5;
    z.size = z.size / 1.5;

    return [y, z];
})
.mergeAll();
```

Nested Collections

See the Pen Map a nested Collection by Brian Leathem (@bleathem) on CodePen.

Nested Collections `mergeAll`

See the Pen MergeAll a Collection by Brian Leathem (@bleathem) on CodePen.

Observable `mergeAll`

See the Pen [MergeAll an Observable](#) by Brian Leathem (@bleathem) on CodePen.

`flatMap`

`flatMap` is a shorthand for a `map` followed by a `mergeAll`.

```
.flatMap(function(x) {
    var y = _.clone(x);
    y.id = y.id + 80;
    y.color = 'green';

    var z = _.clone(x);
    y.size = y.size / 1.5;
    z.size = z.size / 1.5;

    return [y, z];
});
```

reduce

```
var outputData = inputData
  .reduce(function(acc, x) {
    return {
      id: x.id
    , color: 'green'
    , size: acc.size + x.size
    , type: 'square'
    };
}, {size: 0});
```

Collection `reduce`

See the Pen Reduce a Collection by Brian Leathem (@bleathem) on CodePen.

Observable `reduce`

See the Pen Reduce an Observable by Brian Leathem (@bleathem) on CodePen.

zip

```
var outputData = Rx.Observable.zip(
  input1Data,
  input2Data,
  function(x1, x2) {
    return {
      id: x1.id
    , color: x1.color
    , size: x2.size
    , type: x2.type
    };
  });
```

Observable `zip`

See the Pen Zip an Observable by Brian Leathem (@bleathem) on CodePen.

A Burgeoning Standard

> Observable in JavaScript proposal presented to TC-39 (JS standards committee) today. Advanced to Stage 1 (Proposal). https://t.co/sBuazdM7vR— Jafar Husain (@jhusain) May 29, 2015

Creating Observables

Brute Force:

```
var source = Rx.Observable.create(function (observer) {
  observer.onNext(42);
  observer.onCompleted();

  // Optional: only return this if cleanup is required
  return function () {
    console.log('disposed');
  };
});
```

Example: mousemove

Using the brute force approach:

```
Rx.Observable.create(function(observer) {
  var element = document.getElementById("box1");
  element.addEventListener("mousemove", function(event) {
    observer.onNext(event);
  }, false);
});
```

Example: mousemove

Using the `fromEvent` helper

```
var element = document.getElementById("box1");
Rx.Observable.fromEvent(element, 'mousemove');
```

Consuming Observables

```
Rx.Observable.fromEvent(element, 'mousemove')
  .subscribe(
    function(event) {
      console.log(event);
    },
    function(error) {
      console.log(error);
    },
    function() {
      // stream completed
    }
```

Learn Rx

http://reactive-extensions.github.io/learnrx/

Use Case: jQuery .on()

```
Rx.Observable.fromEvent(element, 'mousemove')
  .filter(function(event) {
    return event.target.classList.contains('myClass');
  })
  .subscribe(...);
```

Example: jQuery .on()

See the Pen Event Listener by Brian Leathem (@bleathem) on CodePen.

Use Case: Drag and Drop

Define the Observables:

```
var dragTarget = document.getElementById('dragTarget');
var mouseup   = Rx.Observable.fromEvent(dragTarget, 'mouseup');
var mousemove = Rx.Observable.fromEvent(document,   'mousemove');
var mousedown = Rx.Observable.fromEvent(dragTarget, 'mousedown');
```

Use Case: Drag and Drop

Manipulate the Observables

```
var mousedrag = mousedown.flatMap(function (md) {
  var startX = md.offsetX, startY = md.offsetY;
  return mousemove.map(function (mm) {
    mm.preventDefault();
    return {
      left: mm.clientX - startX,
      top: mm.clientY - startY
    };
  }).takeUntil(mouseup);
});
```

Use Case: Drag and Drop

Subscribe to Observables

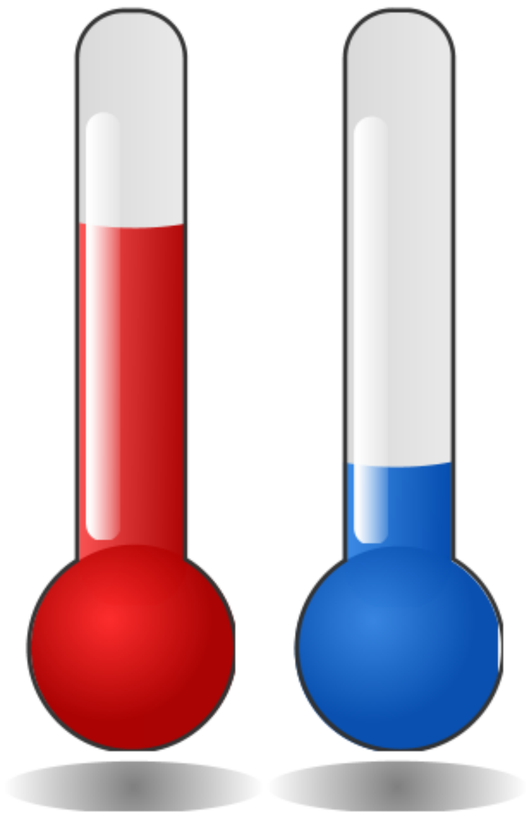
```
var subscription = mousedrag.subscribe(function (pos) {
  dragTarget.style.top = pos.top + 'px';
  dragTarget.style.left = pos.left + 'px';
});
```

Example: Drag and Drop

See the Pen Event Listener by Brian Leathem (@bleathem) on CodePen.

Hot and Cold

- Hot Observable
  - Ongoing; event stream doesn't stop
- Cold Observable
  - No events until you subscribe



```
{...a...b...c.....d..e.....f...}
{1...2...3.....4..5.....6}
```

Ending Observables

```
{...1....2...3..4...5....}.take(3)
```

yields:

```
{1....2...3}
```

Use Case: Autocomplete

```
Rx.Observable.fromEvent($input, 'keyup')
  .map(function (e) {
    return e.target.value; // Project the text from the input
  })
  .filter(function (text) {
    return text.length > 2; // Only if the text is longer than 2 charac
  })
  .debounce(750 /* Pause for 750ms */ )
  .distinctUntilChanged() // Only if the value has changed
  .flatMapLatest(searchWikipedia)
  .subscribe(function (data) {
    // ...
  });
```

Example: Autocomplete

See the Pen Event Listener by Brian Leathem (@bleathem) on CodePen.

Middleware Keynote demo

- UI built with Rx.js (and d3.js)

Demo

Image references

- https://openclipart.org/detail/22519/crowd
- http://www.flickr.com/photos/jdhancock/5845280258/
- http://leslycorazon.wikispaces.com/file/detail/head-silhouette-with-question-mark.png/319199232
- https://openclipart.org/detail/202235/protester
- http://www.slideshare.net/stefanmayer13/functional-reactive-programming-with-rxjs
- https://openclipart.org/detail/170672/weather-icon-hot
- https://openclipart.org/detail/170665/weather-icon-cold

Conclusion

Quite simply:

> Rx.Js allows us to complex asynchronous applications as a composition of very simple functions

Resources

- http://reactive-extensions.github.io/learnrx/ "Learn Rx"
- https://www.youtube.com/watch?v=FAZJsxcykPs --- "Async Javascript at Netflix"
- https://github.com/Reactive-Extensions/RxJS
- http://twitter.com/brianleathem