Reactive Programming Patterns in JavaScript

Follow along! http://172.30.2.43:8080

Joe Barnes

@joescii

Powered by Lift

rogramming

n JavaScript

tp://172.30.2.43:8080

Joe Barnes

@joescii

- Primarily Java EE from 2004-2013
- Began Reactive Programming in November 2013
- Use RFP regularly at Mentor Graphics
- Yes, the accent is real

This session in a nutshell:

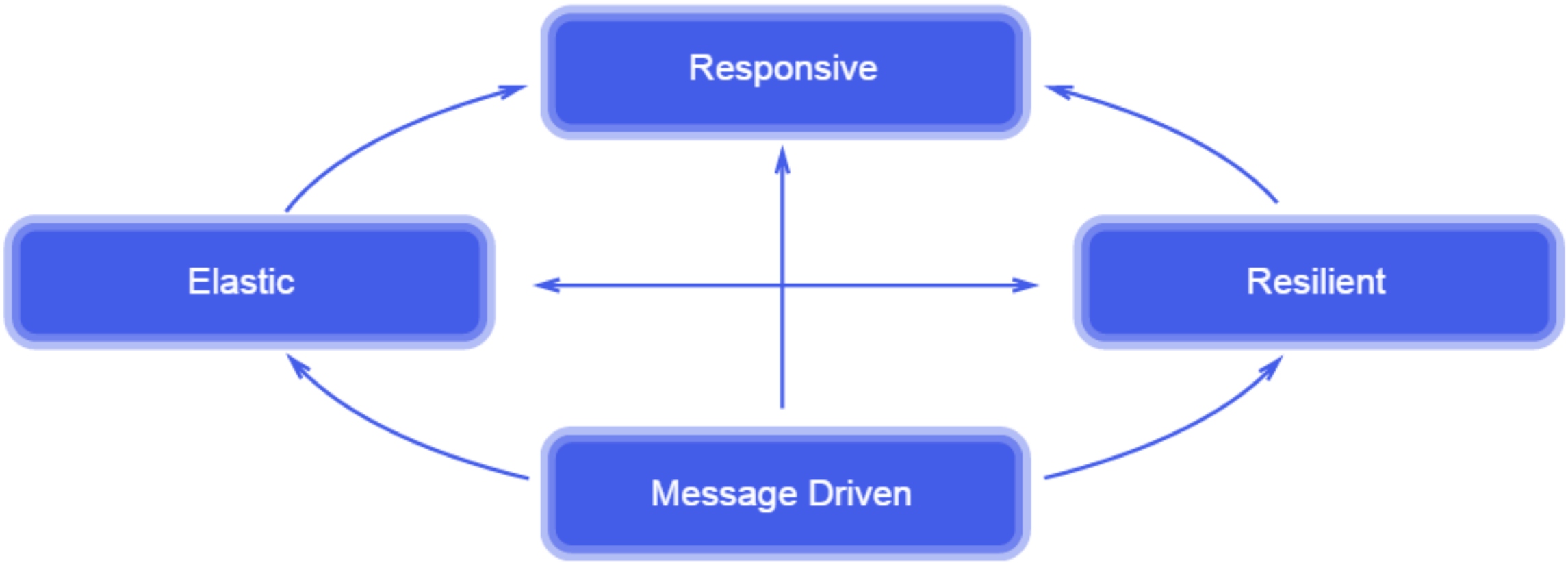
1. Promises (aka Futures)
2. Streams
3. Web Workers (Actors for the browser)

Who heard Venkat's *Reactive Programming* talk?

"Every few years, we come up with a new name for what we've always been doing, and get really excited about it!"

We Are Reactive

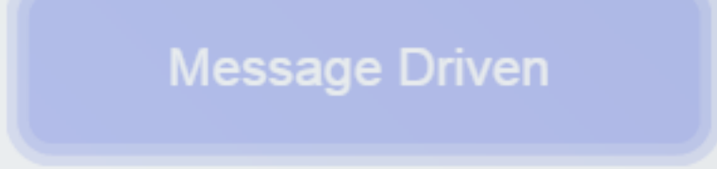
Reactive Manifesto



http://www.reactivemanifesto.org/

How do we go from buzz to practice?

In JavaScript in particular?



http://www.reactivemanifesto.org/

1. Never ever ever ever block!

✓ Use callbacks!

Joe's Stock Shack

Price: $0.00

AAPL 1 USD Quote

~~✓ Use callbacks!~~

✓ Make promises!

Promises are

1. Asynchronous
2. Composable
3. Standardized

The Standard

https://promisesaplus.com/

Basic stuff is standardized, but we used $q from angular in particular.

Three states

1. Pending
2. Fulfilled
3. Rejected

3. Rejected

Promises are like ordering a burger...

1. Place order (receive pending promise/order number)
2. Receive burger (promise fulfilled/number exchanged)
3. Receive apology (promise rejected/no more patties)

Promises are great for *pulling* data...

i.e., sending a message and expecting a response

But what if our data is being *pushed*?

And we don't want callback hell?

Events as a stream →

Observable: producer c

Observable: producer of a stream of events

Subscriber: consumer of a stream of events

Joe's Stock Shack

Price: $0.00

Symbol

1

USD

Quote

Search

What is the difference?

- Stream of events
- List of objects

Streams are like lists, where the elements are events and just so happen to not be available at the same time.

We used RxJS

There is also Bacon.js and Async.js

What if my app has expensive calculations?

How will messages make my app responsive?

Joe's Stock Shack

AAPL ▾ 1 ▾ USD ▾ Chart

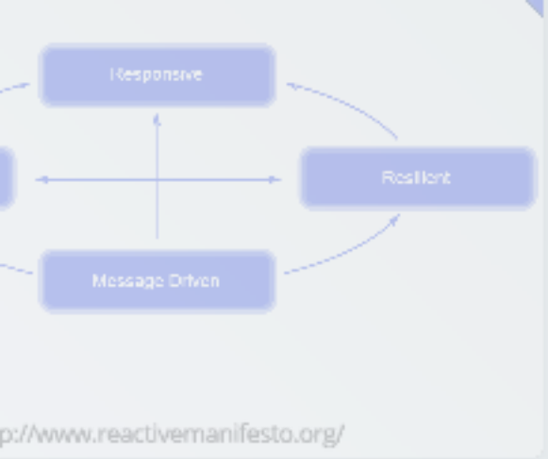
Messages are deep-copied to avoid concurrency concerns

Chrome & Firefox provide *transferable objects*. Like pass by reference.

Web Workers are an implementation of the more general concept known as the *Actor Model* for concurrency.

Recap:

1. Promises: Composable handling of callbacks (pull)
2. Streams: Composable handling of events (push)
3. Web Workers (actors): concurrency in the browser



How do we go from buzz to practice?

ever ever block!

In JavaScript in particular?

Thank you!

Questions, comments, concerns?

ks!

Please fill out the feedback!

Presentation source

~~ks!~~

ises!

Promises are

1. Asynchronous
2. Composable
3. Standardized

The Standard

https://promisesaplus.com/

Basic stuff is standardized, but we used $q from angular in particular.