A Look Inside Observables

Bob Fornal

Entrepreneur

Code-squid provides solid, in-depth frontend training that is supported with real-world code projects. Blessed husband and proud father of two.

Senior Solutions Developer
Leading EDJE, Inc.

Passionate about learning, testing, mentoring, speaking, and personal growth.

What is RxJS

... and what problem does it solve?

- Reactive Extensions - dealing with streams of things.
- Extensions to the JavaScript language around Reactive Programming.
- Reacts to events or signals and contain all state and pass the event to the next step in the chain.

History

Callback-Hell

- Handling multiple requests and accounting for the order-of-response.

Promise-Hell

- Better code-readability with then, catch, finally.

Async/Await

- Improved code-readability, unwraps promises.

Observables

- Push/Pull models ...

	SINGLE	MULTIPLE
Pull	Function	Iterator
Push	Promise	Observable

Essential Concepts

The essential concepts in RxJS what solve async event management are ...

Observable
Represents the idea of an invokable collection of future values or events.

Observer
Is a collection of callbacks that know how to listen to values delivered by the Observable.

Subscription
Represents the execution of an Observable, is primarily useful for cancelling the execution.

Operators
Are pure functions that enable a functional programming style of dealing with collections with operations like map, filter, concat, reduce, etc.

Subject
Is equivalent to an EventEmitter, and the only way of multicasting a value or event to multiple Observers.

Schedulers
Are centralized dispatchers to control concurrency, allowing us to coordinate when computation happens on e.g. setTimeout or requestAnimationFrame or others.

Promises versus Observables

- Observables are lazy whereas promises are not.
- Observables handle multiple values unlike promises.
- Observables are cancelable.
- Observables provide many **operators**.

Subject

An **RxJS Subject** is a special type of **Observable** that allows values to be multicasted to many Observers. While plain Observables are unicast (each subscribed Observer owns an independent execution of the Observable), Subjects are multicast.

- BehaviorSubject
- ReplaySubject
- AsyncSubject
- void Subject


TIME FOR QUESTIONS