Desbravando o Mundo da programação reativa

Erich de Souza Oliveira

- 30 anos (10 anos trabalhando em desenvolvimento)
- CTO Winnin (Plataforma de curadoria de vídeo)
- Autor de “Mastering Reactive Javascript” (PacktPub)
- Viciado em Javascript, mas com experiência em Java e Scala.
- Interesse em programação reativa e arquitetura de microsserviços
- @oliveira_erich (Twitter)
- @ericholiveira (Medium)
- ericholiveira.com

Programação reativa

- O que é?
- Onde pode ser utlizado?
- Como usar?
- Observable/Operator/Observer
- Vantagens
- Backpressure
- Principais operadores
- Referências

O que é?

“Um paradigma de programação para lidar com fluxo de dados e propagação das alterações. Torna possível expressar fluxos de dados estáticos (arrays) e dinâmicos (eventos) facilmente através da linguagem de programação utilizada”

Wikipedia

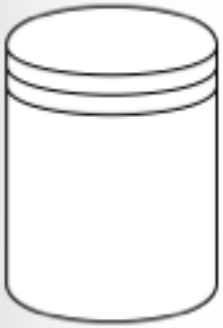
Entendeu?

O que é?

- Mais uma *buzzword*?
- Um paradigma de programação que permite operar sobre um fluxo assíncrono de dados
- Assim como array é uma abstração utilizada para representar um número finito de elementos, um Observable é uma abstração para representar um número possivelmente infinito de elementos.
- Utiliza *Observables, Operators e Observers* para expressar e “reagir” a ocorrência de eventos.
- Permite desacoplar a fonte dos eventos, das operações feitas sobre ele dos efeitos que eles causam

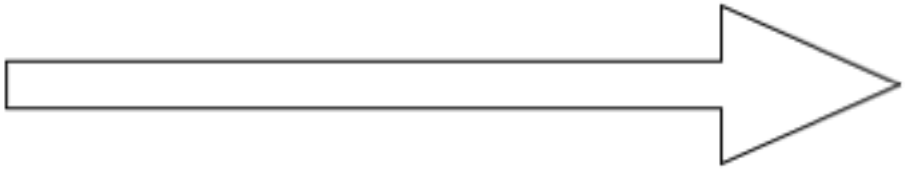

STARBUCKS
COFFEE
©purplishblack

Observable



Fonte dos dados

Operator



Transformções feitas sobre os dados

Observer

Efeitos

Onde pode ser utilizado?

Onde pode ser utilizado?

- Sensores de IoT
- Inputs de um usuário em uma GUI (cliques em um botão, movimento de mouse e etc.)
- Número grande (cursor) de elementos de um banco de dados
- Array
- Na verdade, qualquer fonte/fluxo de dados

Como usar?

Bibliotecas

Reactive Extensions

Reactive extensions (Rx)?

- Criado e aberto (2012) pela Microsoft
- Uma API com implementação para diversas linguagens
- As implementações entre as linguagens podem variar um pouco, ainda assim, a maioria dos operadores e nomenclatura é a mesma, então após aprender Rx em uma linguagem é fácil conseguir aplicar Rx em outras linguagens
- RxJava, RxJS, Rx .Net, RxScala, RxClojure, RxSwift, RxPHP, RxCpp, RxGo, RXPy…
- http://reactivex.io/languages.html

Observable/Operator/Observer

Observable

- Representa a fonte de eventos
- Permite propagar de eventos recebidos
- É como um “array para um número infinito de dados”
- Avisa os Observers toda vez que um novo dado é recebido
- Reutilizável
- Propaga três tipos de eventos “next”, “error” e “complete”

Operators

- Representa as transformações feitas sobre os eventos do Observable
- Permite “alterar” os dados propagados por um Observable
- Permite combinar Observables
- Não altera os Observables, cria novos.

Observer

- Representa a ação tomada quando um novo evento acontece
- Se inscreve nos Observables para ouvir eventos (“next”, “error” e “complete”)
- É como um “listener” de algumas linguagem

Vantagens

Vantagens

- Facilita o trabalho com um fluxo de dados assíncronos e potencialmente infinito
- Desacoplar a fonte de dados, das transformações e efeitos causados por esses dados
- Facilita o reuso de fonte de dados
- Uso constante de memória e processamento
- Como as operações sobre os dados são desacopladas da fonte dos dados, facilita testes unitários

Backpressure

Backpressure

- Acontece quando recebemos dados mais rápido do que somos capazes de processar
- Lossy strategies (throttle, etc…)
- Loss-less strategies (buffer, etc…)

Principais operadores

Principais operadores

- map
- flatMap
- buffer
- throttle
- take
- merge

Referências

Referências

- Mastering Reactive Javascript
- introtorx.com
- http://reactivex.io/
- http://reactivex.io/tutorials.html

Obrigado