LITERATURLISTE • RX.JS

Asynchrone Programmierung
Asynchroner Code ist Code, der parallel ausgeführt wird. Das bedeutet, dass nicht gewartet wird, bis zum Beispiel eine Funktion abgearbeitet wurde, bevor die nächste begonnen wird. Asynchrone Funktionen werden aufgerufen, während andere Funktionen noch in der Ausführung sind. Sie können sich dadurch auch gegenseitig beeinflussen.
https://www.heise.de/developer/artikel/Einfuehrung-in-die-asynchrone-JavaScript-Programmierung-2752531.html?seite=all

Bedingende Operatoren
Operatoren können auf Observables angewendet werden. Bedingende Operatoren formulieren eine Bedingung. Erfüllt der eingehende Stream bzw. die eingehenden Streams die Bedingung, wird der so neu erzeugte Stream mit einem True-Ausdruck terminiert. Beispiele: .every, .defaultIfEmpty.
https://rxmarbles.com/#defaultIfEmpty

Callback Funktion
Bei Callback Funktionen handelt es sich um Funktionen, die innerhalb einer anderen Funktion als Parameter ausgeführt werden. Sie sind zwar kein Bestandteil der rx.js Bibliothek und auch nicht per se asynchron, werden aber häufig in der asynchronen Programmierung verwendet.
https://www.mediaevent.de/javascript/callback-function.html

Filternde Operatoren
Filternde Operatoren erzeugen ein neues Observable, indem sie verschiedene Filter auf den eingehenden Stream anwenden und nur bestimmte Events in den neu erzeugten Stream übernehmen. Beispiel: .filter.
https://rxmarbles.com/#filter

Initiierende Operatoren
Hierbei handelt es sich um Operatoren, die ein neues Observable erzeugen. Beispiele: .of, .interval.
https://rxmarbles.com/#from

Kombinierende Operatoren
Operatoren können auf Observables angewendet werden. Kombinierende Operatoren erzeugen ein neues Observable, indem sie zwei oder mehr Streams miteinander kombinieren. Beispiel: .combineLatest
https://rxmarbles.com/#zip

Marble Diagramm
Ein Marble Diagramm (Murmel-Diagramm) ist eine etablierte Schreibweise, bei der Observables als Zeitstrahlen dargestellt werden, auf denen sich Kreise (Marbles/Murmeln) befinden, die die emitierten Werte representieren.
https://rxmarbles.com/#zip
https://www.buschmais.de/2018/05/08/einfuehrung-in-rxjs/

Mathematische Operatoren
Diese Operatoren wenden mathematische Operationen auf die eingehenden Events an. Beispiel: .count.
https://rxmarbles.com/#count

Observable
Ein Observable macht einen Strom an Daten verfügbar, um diesen mit verschiedenen Befehlen zu verarbeiten. Die Operatoren .from, .interval, .of und .timer erzeugen Observables.
https://medium.com/@mpodlasin/promises-vs-observables-4c123c51fe13
https://medium.com/javascript-everyday/javascript-theory-promise-vs-observable-d3087bc1239a
https://www.buschmais.de/2018/05/08/einfuehrung-in-rxjs/
https://rxmarbles.com/#from

Promise
Ein Promise macht es möglich auf ein zukünftig eintretendes Ereignis zu reagieren, während parallel anderer Code weiterhin ausgeführt wird.
https://medium.com/@mpodlasin/promises-vs-observables-4c123c51fe13
https://medium.com/javascript-everyday/javascript-theory-promise-vs-observable-d3087bc1239a

Subscribe und Unsubscribe
Eine Subscription erstellt einen Zugriff auf die Daten eines Observables. Eine Subscription muss immer mittels unsubscribe beendet werden, damit sie nicht den Prozessor und Arbeitsspeicher unnötig belastet. Komfortabler ist es, Operatoren zu verwenden, da diese nicht unsubscribed werden müssen.
https://rxmarbles.com/#map

Transformierende Operatoren
Sie dienen dazu, die eingehenden Werte nach einem vorgegebenen Schema umzuwandeln. Beispiel: .map.
https://rxmarbles.com/#map