Reactive Extensions (rxjs)

In diesem kleinen Tutotial werden wir andere nützliche Wege erkunden, wie man Observables einsetzt. Zum Ende dieses Tutorials werdet ihr lernen, wie man

* ein Observable aus DOM-Events erstellt (The Angular Way)
* ein Observable aus einem Array erstellt
* ein Observable aus einer Liste von Argumenten erstellt
* einen Timer implementiert
* parallele Operationen ausführt
* Fehler behandelt
* sich benachrichtigen lässt, wenn ein Observable abgeschlossen ist

Im Rahmen dieses Tutorial sollet ihr alle Code-Schnipsel in unserer Applikation testen und euch die **Ergebnisse gründlich ansehen**. Dadurch werdet ihr für das kommende Quiz in diesem Abschnitt und natürlich auf die weitere Implemnetierung unserer Applikation vorbereitet.

Ich bitte euch den gesamten Code von Hand zu schreiben, auch wenn ihr dieses Dokument als digitale Version erhaltet.

**Hinweis:**

Um vielleicht etwas nachzulesen, findet ihr hier die API eines Observables aus RxJs:

<https://github.com/Reactive-Extensions/RxJS/blob/master/doc/api/core/observable.md>

Ansonsten hilft vielleicht die abstrakte Dokumentation hier:

<http://reactivex.io/documentation/observable.html>

Erstellen von Observables aus DOM Events

Ihr habt bereits gelernt, wie man Observables aus DOM Events erstellen kann.



Dieser Code ist eng mit dem DOM gekoppelt und schwer (Unit)- testbar.Client-seitiger Code, der stark an den DOM gekoppelt ist, ist genauso schlecht wie Server-seitiger Code, der direkt in die Datenbank schreibt. Wir brauchen mehr Abstraktion. In Angular sollten wir keine Bibliotheken wie jQuery verwenden oder direkt mit dem "document" Objekt arbeiten, denn das zerstört das Angular Prinzip, der DOM-Entkoppelung von Components. Stattdessen benutzen wir Property-Binding und lassen Angular die Arbeit machen. Der Grund, warum ich hier jQuery für diesen Code verwendet habe, ist, um die Komplexität gering zu halten und den Focus auf die Vorteile von Observables zu richten.

The Angular Way

Im Abschnitt über Forms, betrachteten wir die Control-Klasse - sie repräsentiert ein Eingabefeld einer Form. Diese Klasse hat eine Methode namens valueChanges, die eine Observable zurück gibt. Wir können auf dieses Observable subscriben und werden benachrichtigt, sobald sich der Wert des Eingabefelds ändert. Auf diese Weise arbeiten wir nicht direkt direkt mit einem DOM-Element.

1. Verwendet die folgende Vorlage in AppComponent:



1. Erstellt das Formular-Objekt in AppComponent:



1. Erhaltet eine Referenz auf die Abstrakte "Search" Control (bzw. FormControl) und subscribe(d) auf dessen valueChanges Methode (Im Konstruktor):



1. Führt die Anwendung aus und gebt etwas in das Eingabefeld ein. Beachtet, dass mit jedem Tastendruck, den ihr tätigt, das Observable den aktuellen Wert des Eingabefelds pusht.
2. Wendet die folgenden Operatoren vor dem subscribe auf valueChanges an. Was seht ihr in der Konsole bei Eingabe eines oder mehrerer Leerzeichen?



**Beachte**: <string> str castet die Variable str (Typ: any) in eine Zeichenkette.

Erstellen eines Observable aus einem Array

Das Erstellen von Observables ist nicht nur auf asynchrone Datenströme beschränkt. Wir können auch ein Observable aus einem Array erstellen. Isn't that nice? ;)



Diese Zeile Code gibt uns ein Observable mit drei Elementen zurück: 1, 2 und 3. Das bedeutet, dass wir bei einem "subscribe" unserer callback Methode 3x aufgerufen wird und wir jeweils einen Wert erhalten.

Was ist der Vorteil, ein Array zu einem Observable Stream umzuwandeln? Wir können diesen Observable Stream mit anderen Observables verbinden. Zum Beispiel aus DOM-Events, AJAX-Calls, Web Sockets, etc. Denkt beispielsweise an eine Reise-Suchmaschine, wie Kayak. Viele dieser Suchmaschinen lassen den Benutzer nach Flügen an genau einem oder auch in einem 2-Tages-Fenster suchen.

Nehmen wir an, der Benutzer hat das 2-Tages-Fenster gewählt. Dann erstellen wir ein Array der Reisedaten mit dem 2-Tage-Fenster,



konvertieren das Array zu einem Observable, und für jedes Datenelement , verwenden wir die map Operation um Angebote für das angegebene Datum vom Server ab zu rufen.



Was sehen wir in der Konsole? Warum bekommen wir "undefined" nach jedem Server-Aufruf? Weil unsere map Operation keine Werte zurück gibt - sondern nur ein console.log() macht.

Wenn wir den map Operator aber folgendermaßen modifizieren:



Welche anderen Möglichkeiten haben wir um ein Observable zu erstellen?

Wir können die Observable.of() Methode nutzen um von eine ganzen Liste von Argumenten ein Observable zu erstellen.

