MYTODO

Autor: Yannis Biasutti
Datum: 05.06.2020

Github Repository: https://github.com/biasutti/bfh-js-web-project

Kurzbeschrieb:
Aufgrund der Corona Krise wurde entschieden das Modul BTI7540q in ein Modul ohne Prüfung umzuwandeln. Anstelle dieser Prüfung soll eine Todo Applikation mittels eines zur freien Wahl stehenden JavaScript Framework umgesetzt werden. Dazu soll eine kleine Dokumentation geschrieben werden, welche die gröbsten Unterschiede zu dem im Unterricht erarbeiteten VanillaJS App aufzeigen.

AUFGABENSTELLUNG

Während ca. 6 Wochen wurde im (online) Präsents Unterricht eine VanillaJS Applikation erarbeitet, mit welcher man Todos verwalten kann. Ein Login wurde implementiert und die Grundaktionen wie Erstellen und Bearbeiten von Todos wurde umgesetzt. Die Backendumgebung wurde von Herr Locher zur Verfügung gestellt.

REST Service: http://distsys.ch:1450/

Als Projektauftrag gab es die Aufforderung die gleiche Applikation in einem zur freien Wahl stehenden JavaScript Framework nachzubauen. Als solches Framework wurde Angular gewählt.

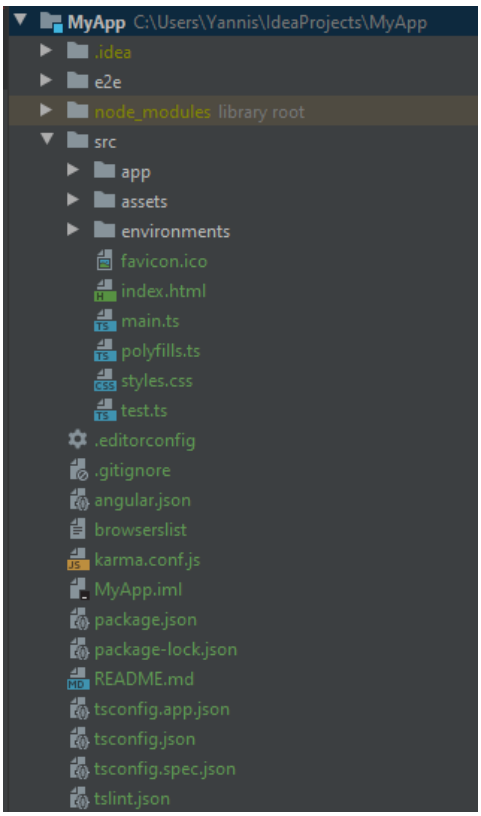
VORGEHEN

Im Unterricht wurde nicht auf solche Frameworks eingegangen also man musste sich in einem ersten Schritt, bevor man überhaupt an die Umsetzung der Applikation denken konnte, eine Einführung in das gewählte Framework durcharbeiten. In Angular gibt es die relativ bekannte Tutorial Tour of Heroes. Dies war auch mein Startpunkt und ich habe begonnen die AngularCLI besser kennenzulernen. Das Tutorial besteht aus sechs Kapitel, in welchen man Schrittweise alle benötigten Komponenten einer Applikation kennenlernt. Dies habe ich auch immer direkt versucht auf die eigene Todo Applikation umzumünzen.

VERGLEICH VANILLA JS / ANGULAR

1. PROJEKTSETUP

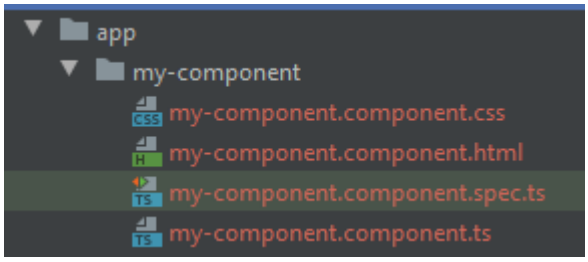
Zum erstellen eines Projektes in VanillaJS hat man keinerlei Hilfsmittel, man startet mit einem leeren Repository in welchem man sich eine freie Struktur erstellen kann. Dies hat Vor- so wie auch Nachteile. Als Junior in der JavaScript Welt hat man durch das strukturierte Abarbeiten der Unterrichtsaufgaben ein Verständnis entwickelt welche Komponenten es mindestens benötigt, um eine kleine Applikation zu entwickeln. Ich habe aber keine Erfahrungswerte wie sich so ein Projektsetup, welches sehr rudimentär ist, konkret die Templates im HTML File, auf ein mittleres bis grosses Projekt auswirkt und ob es überhaupt tragbar wäre. In Angular sieht dies schon anders aus. Zur Initialisierung des Projektes hat man ein Tool zur Verfügung welches einem während dem ganzen Entwicklungsprozess begleitet. Dies ist die AngularCLI. Mittels der folgenden Commands kann ein komplettes Projektsetup erstellt werden mit den Files und Verzeichnissen wie Sie von Angular auch zur Skalierung von grösseren Projekten benötigt werden.




```
ng new MyApp
```

2. KOMPONENTEN

Komponenten entsprechen in VanillaJS so wie auch in Angular meistens einer entsprechenden Webpage in der Applikation. In VanillaJS haben wir uns eine IIFE zu Hilfe genommen, um unsere Objekte von anderen Teilen der Applikation abzukapseln und nur kontrolliert Zugang zu Variablen zu gewähren. Angular wird grundsätzlich nur in TypeScript entwickeln und wird in modernen Umgebungen zu ECMA6 transpiliert. ECMA6 bringt ein überarbeitetes Modulsystem mit von welchem hier gebraucht gemacht wird. Das Grundgerüst einer Komponent kann in Angular auch mittels der AngularCLI generiert werden, es entstehen Files für die Logik, Frontendansicht so wie auch Unittests.




```
ng generate component MyComponent
```

Ein weiterer Unterschied von VanillaJS Komponenten zu Angular Komponenten ist das verwenden von Services. Alle AJAX Calls welche aus der VanillaJS Komponente gemacht werden sind grundsätzlich auch in dieser definiert und müssen in jeder Komponenten auch neu definiert werden. Angular behilft sich hier einer weiteren speziellen Art einer Komponente, Services. Services entsprechen einem oder auch mehreren externen Endpoints und behandeln alle Verbindungen zu diesen. Die Komponente welchen Zugriff auf die Daten eines Endpoints brauchen binden nun nur noch diesen Service ein und die Konfiguration ist an einem zentralen Punkt geregelt.

3. ROUTER

Ein Router in einer JavaScript Applikation, unabhängig von Framework, hat die Aufgabe eine URL zu einer Komponente zu mappen. Als Entrypoint kann eine Default Homepage definiert werden und Logik, welche jede Seite betrifft, kann hier bereits implementiert werden. In unserer VanillaJS Applikation ist der Router nicht anderes als eine Komponente. Man muss sich aber aktiv darum kümmern einen solchen Router zu erstellen und die benötigten Default Settings zu konfigurieren. Im Vergleich zu Angular wird dieser Router wieder mit der AngularCLI erstellt.

```
ng generate module app-routing
```

Sobald man ein Routermodule generiert und die benötigen Routen registriert hat kann man diese über einen Routerlink aufrufen und zwischen Seite navigieren, ohne einen Pagereload zu triggern.

Beispiele für Logik welche alle Seiten betrifft ist das schützen von Seiten welche einen Eingeloggten Zustand benötigen. Dies kann in Angular mittels eines AuthGuards erzielt werden welches einfach auf jede Route angewendet werden kann. Konkret bin ich mit Angular hier an meine Grenzen gestossen und kann nicht weiter über die Funktionalität eines AuthGuards berichten.

SCHWIERIGKEITEN

Durch die wöchentliche Besprechung zu Unterrichtszeiten konnten beim erarbeiten der VanillaJS keine grossen Schwierigkeiten auftreten und alles war sehr klar dokumentiert. Bei Angular war ich positiv überrascht wie klar die Anweisungen, selbstverständlich mit dem Vorwissen aus dem Unterricht, nun waren. Die grössten Schwierigkeiten waren dann Teile, welche nicht konkret von diesem Tutorial behandelt wurde und man sich selbständig durch Doku und Forenposts eine Lösung erarbeiten musste. Es war schwierig zu entscheiden ob dieser Lösungsansatz nun der «State oft the Art» Ansatz war oder ob es bereits wieder neue oder bessere bzw. einfacheren Wege gab dies zu lösen. Dies habe ich konkret beim Umsetzen des Logins erfahren. Mir war nicht klar wo ich unsere in VanillaJS erstellte Store Komponente umsetzen soll und habe diverse Lösungsansätze verfolgt, welche aber immer verwirrender wurden und ich habe mich dann entschieden eine VanillaJS nahe Lösung zu implementieren.

FAZIT

Die Aussage von Herr Locher das man für kleinere Applikationen problemlose, bzw. sogar am besten mit VanillaJS arbeitet, kann ich voll und ganz unterzeichnen. Ich sehe die Vorteile von Angular für ein grösseres und professionelles Umfeld, jedoch braucht es auch sehr viel mehr Einarbeitung in ein solches Framework und dessen Umfeld. Die Freiheit ein Framework zu wählen welches seinem eigenen Need entspricht, fand ich in diesem Kontext interessant jedoch denke ich, es macht in Zukunft durchaus Sinn den Unterricht wie in den letzten Jahren durchzuführen und ein Framework auch im Unterricht zu behandeln. Gerade mit dem Aspekt das es meistens nicht eine Lösung gibt, finde ich es Wertvoll auch von Bestpractises der Dozenten zu profitiere.