1. PLF in Webentwicklung und Mobile Computing

Klasse: 5AKIF
Datum: 4. November 2024

Aufgabe 1: Arrayfunktionen

In der Datei aufgabe1/driving_school.json befinden sich JSON Daten, die die Kurse einer Fahrschule beschreiben. Es gibt mehrere Kurse wie Basic Course (Grundkurs), Class A, Class B und Class AM. Die Kurse werden mehrmals angeboten. Daher hat jeder Kurs mehrere Blöcke. Ein Block ist z. B. der Semesterferienkurs. Im Block gibt es einen Schedule, der die einzelnen Kurstermine speichert. Pro Block gibt es Kunden und einen Fahrlehrer, der den Kurs hält.

In der Datei aufgabe1/array_functions.js befinden sich verschiedene Aufgaben in den Kommentaren, die Sie lösen sollen. Die Korrekte Ausgabe, wenn das Programm mit *node array_functions.js* gestartet wird, ist wie folgt:

```
AUFGABE 1
```

(index)	course_name	course_code	blocks
0	'Basic Course'	'BC101'	2
1	'Class A'	'CA101'	1
2	'Class B'	'CB101'	1
3	'Class AM'	'CAM101'	1

```
AUFGABE 2
{ course_name: 'Class A', course_code: 'CA101' }
AUFGABE 3
{ course_code: 'BC101', lowest_price: 200, highest_price: 220 }
AUFGABE 4
{ first_day: '2024-11-06', last_day: '2024-11-15' }
AUFGABE 5
```

(index)	block_id	price	customers
0	1	200	2
1	2	220	3

```
AUFGABE 6
```

(index)	course_name	customers	revenue
0	'Basic Course'	5	1060
1	'Class A'	2	500
2	'Class B'	2	600
3	'Class AM'	2	360

```
AUFGABE 7
```

(index)	id	name
0	1	'John Smith'
1	2	'Sarah Williams'
2	3	'Chris Adams'

Aufgabe 2: Typescript interfaces definieren

In aufgabe2/src/app.ts befindet sich eine vorkonfigurierte Typescript app. Sie liest das JSON File mit den Fahrschuldaten aus Aufgabe 1 ein. Es sollen Interfaces für die gespeicherten JSON Daten erstellt werden. Die Funktion *listCustomers* soll die Kunden für einen bestehenden Block eines Kurses zurückliefern. In den Kommentaren dieser Datei befinden sich weitere Anweisungen. Wenn das Programm mit *npm run start* gestartet wird, soll folgende Ausgabe auf der Konsole erscheinen.

```
> aufgabe2@1.0.0 start
> tsc && node src/app.js

4 courses loaded.
List of customers in BC101, Block 1: [ { id: 101, name: 'Alice Johnson' }, { id:
102, name: 'Bob Lee' } ]
```

Aufgabe 3: fetch und typeguards

Für diese Aufgabe benötigen Sie das Backend der Todo App, das auf *https://localhost:5443* läuft. Es befindet sich im Repo *https://github.com/Die-Spengergasse/course-wmc-5sem* im Ordner *30_TodoApp/TodoBackend* und kann mit der Datei *startServer.cmd* gestartet werden.

In aufgabe3/src/app.ts ist bereits Code enthalten, der die Daten von *https://localhost:5443/api/TodoTasks* einliest. Ihre Aufgabe ist es, die Serverantwort zu projizieren (wir brauchen nur 4 Properties) und ein Interface samt typeguard dafür zu erstellen. In den Kommentaren dieser Datei befinden sich weitere Anweisungen. Wenn das Programm mit *npm run start* gestartet wird, soll folgende Ausgabe auf der Konsole erscheinen.

```
> aufgabe3@1.0.0 start
> tsc && node src/app.js

TodoTask Amet aliquid laborum illo perspiciatis dolorum. is in TodoItem Itaque et
earum modi similique veniam explicabo quo.
TodoTask Qui ad nisi provident reiciendis ea et doloribus quia odio. is in
TodoItem Itaque et earum modi similique veniam explicabo quo.
TodoTask Hic reprehenderit ipsum odit earum qui illum maiores vel. is in TodoItem
Itaque et earum modi similique veniam explicabo quo.
```

```
TodoTask Voluptatem voluptatem beatae. is in TodoItem Itaque et earum modi
similique veniam explicabo quo.
TodoTask Ipsum earum deleniti nobis corrupti. is in TodoItem Itaque et earum modi
similique veniam explicabo quo.
TodoTask Tempora qui voluptatem deleniti in at ipsum. is in TodoItem Itaque et
earum modi similique veniam explicabo quo.
```

Bewertung und Abgabe

Erstellen Sie in Ihrem WMC Repo einen feature Branch mit *git checkout -B exercise/wmc_plf_20241104*. Committen Sie Ihre Lösung und führen Sie einen pull request in den main branch durch.

- Aufgabe 1 (14 Punkte)
  - 2 Punkte für jedes der 7 Beispiele.
- Aufgabe 2 (9 Punkte)
  - 6 Punkte für die Interfaces
  - 3 Punkte für die Funktion *listCustomers*
    - 1 Punkt für den korrekten Prototypen
    - 2 Punkte für die korrekte Logik
- Aufgabe 3 (6 Punkte)
  - 2 Punkte für das Interface
  - 2 Punkte für den typeguard
  - 2 Punkte für die Funktion *fetchTodoTasks*

22 - 20 Punkte: Sehr gut, 19 - 17 Punkte: Gut, 16 - 14 Punkte: Befriedigend, 13 - 11 Punkte: Genügend.