MAK Async JavaScript

Webprogrammierung und Mobile Computing 4

Name:

2023-10-23

Allgemeines

Öffnen Sie ein Terminalfenster und führen Sie folgendes Kommando aus:

```
sudo assignment pull wmc4-x3q1r
```

Folgen Sie den Anweisungen und authentifizieren Sie sich mit Ihrem Microsoft-Account. Die Vorlage für Ihre MAK wird automatisch in das Verzeichnis

`/media/usb/Work/wmc4-x3q1r`

heruntergeladen und vorbereitet. Sie können das Verzeichnis direkt in Visual Studio Code öffnen und mit der Bearbeitung der Aufgabenstellung beginnen.

Am Ende der Arbeit können Sie Ihre Lösung mit folgendem Kommando abgeben:

```
sudo assignment submit wmc4-x3q1r
```

1 Asynchrone Berechnungen mit Promises (40 %)

Gegeben ist die folgende synchrone Implementierung von zwei Funktionen, die sehr triviale Berechnungen durchführen.

```
function add(a, b) {
    return a + b;
}

function div(a, b) {
    if (b === 0)
        throw new Error('div by zero');
    return a / b;
}
```

Weiters finden Sie in der Datei `calculations.js` drei Berechnungen, die diese Funktionen verwenden.

- Schreiben Sie die Funktionen dahingehend um, dass diese unter Verwendung von `setTimeout` bzw. `setImmediate` asynchron ausgeführt werden. Verwenden Sie ein Promise, um das Ergebnis bzw. den Fehler der asynchronen Operation zurückzuliefern.
- Ändern Sie auch den Aufruf der Funktionen für die drei Berechnungen indem sie eine *Promise Chain* verwenden und sorgen Sie für ein korrektes Fehlerhandling.

2 Kombinieren von JSON Files (60 %)

2.1 Allgemeines

- Machen Sie sich mit der Implementierung in der Datei `index.js` vertraut und testen Sie das Service mit den vorgegebenen Requests in der Datei `requests.rest`.
- Testen Sie auch die Fehlerbehandlung, indem Sie bewusst die Datei `data/klassen.json` umbennenen bzw. achten Sie auch das falsche Format in den Datei `missingKey.json` und `invalidJSON.json`.

Sie werden sehen, dass die Implementierung sehr naiv ist und synchrone File-I/O verwendet.

2.2 Implementierung readJSONAsync

Schreiben Sie eine Funktion `readJSONAsync`, welche dieselbe Funktionalität wie die bestehende `readJSONSync` Funktion beinhaltet. Greifen Sie dabei auf die Funktion `readFile` zurück. Verwenden Sie ein Callback und die Error-First-Strategie, um das Ergebnis bzw. etwaige Fehler zurückzuliefern

2.3 Implementierung readSchuelerAsync und readKlassenAsync

Implementieren Sie analog zu den bestehenden Funktionen `readSchuelerSync` und `readKlassenSync` eine asynchrone Variante, die auf die Funktion `readJSONAsync` zurückgreift.

2.4 Anpassung Request-Handler

Passen Sie den Request-Handler dahingehend an, dass die asynchronen Funktionen verwendet werden. Achten Sie dabei auch auf ein korrektes Errorhandling in der entstehenden *Callback-Hell*.

Testen Sie die neue Implementierung ebenfalls mit den Requests aus `requests.rest`.

2.5 Zusatzaufgabe (bei Lösung ein zusätzliches Plus)

Nachdem das Einlesen und Parsen der beiden Dateien unabhängig voneinander ist, kann dies parallel ausgeführt werden. Schreiben Sie als Zusatzaufgabe eine Version des Requesthandlers mit der relativen URL `/api/schuelerP`, welche die beiden Dateien parallel ausliest und dann zusammenführt und filtert.