02: Person Picker

Schreibe ein *NodeJS* Programm, das mittels *synchroner Programmierung* folgenden Funktionsumfang abdeckt:

- Konvertierung eines *Arrays von Strings* (siehe Datei `students.json`) in ein *Array von Objekten*
- Zufälliges auswählen einer Person
- Sortierte Ausgabe aller Personen

Gehe dabei wie folgt vor:

1. Installiere dir die 3rd party Module
   - *yargs*
   - *lodash*
   - *chalk*
2. Erstelle zwei Files
   - `app.js`
   - `converter.js`
3. Baue dein Programm so auf wie unsere Notizzettel Applikation

   `app.js:` definiert den Einstiegspunkt in das Programm und übernimmt die Steuerung

   `converter.js:` übernimmt die Business Logic
4. Erlaube folgende Kommandos
   - `node app.js convert --input="students.json"`
   - `node app.js pick --input="db.json"`
   - `node app.js print --input="db.json" --sortby="firstname" --order="asc"`

Happy coding!

Die Funktionen (`converter.js`)

Implementiere die folgende Funktionen in `converter.js`. Überprüfe in jeder Funktion, die eine Datei übergeben bekommt, ob diese Datei tatsächlich existiert! Existiert die Datei nicht, beende die Funktion und liefere ein entsprechendes Fehler-Objekt zurück (siehe „*Rückgabewert*"bei der jeweiligen Funktion unten).

convert(input):

Konvertiert die übergebene Datei, die ein *Array von Strings* enthält, in ein *Array von Objekten* und schreibt die konvertierten Daten in die Datei `db.json`.

Die Funktion soll zur Konvertierung die Funktion `makeObject` verwenden (siehe unten).

- Parameter: der Pfad zu der Datei, die konvertiert werden soll
- Rückgabewert:

  Success: { `success: true` }

  Failure: { `success: false, message:` *`error-message`* }

makeObject(people):

`makeObject` ist eine interne Hilfsfunktion. Sie soll also nicht exportiert werden)

Die Methode bekommt das von `convert(input)` gelesene *Array von Strings* und konvertiert das Array mit Hilfe der Methoden

- `split` (*lodash*)
- `toUpper` (*lodash*)

in ein *Array von Objekten* mit dem Aufbau:

{ `lastname: "REITSAMER", firstname: "Wolf"` }

Der Nachname soll also in Großbuchstaben gespeichert werden!

pickRandom(input):

Wählt mit Hilfe der *lodash* Funktion `shuffle` eine zufällige Person aus der Datenbank aus und liefert diese zurück.

- Parameter: der Pfad zur bereits konvertierten Datenbank (also zu dem von `convert` erstellten File - im konkreten Fall *db.json*).
- Rückgabewert:

  Success: { `success: true, data:` *`person`* }

  Failure: { `success: false, message:` *`error-message`* }

Happy coding!

getSorted(input, sortBy, order):

Verwende die Funktion `orderBy` aus dem Modul `lodash`, um alle Personen deiner Datenbank nach der Eigenschaft *sortBy* aufsteigend (*asc*) oder absteigend (*desc*) zu sortieren.

- Parameter:

  input der Pfad zur bereits konvertierten Datenbank (also zu dem von `convert` erstellten File).

  sortBy der Name des keys nach dem sortiert werden soll (also 'firstname' oder 'lastname')

  order die Sortierreihenfolge - also *'asc'* oder *'desc'*

- Rückgabewert:

  Success: { `success: true, data:` *`sortiertes Array`* }

  Failure: { `success: false, message:` *`error-message`* }

printPerson(person):

`printPerson` bekommt ein Object der Form

{ `lastname: "REITSAMER", firstname: "Wolf"` }

und gibt die Person mit Hilfe des Moduls `chalk` in blau am Bildschirm folgendermaßen aus:

`REITSAMER, Wolf`

Die Funktion liefert nichts zurück.

Verwende diese Funktion in `app.js`, um die mittels `pickRandom` erhaltene Person bzw. die mittels `getSorted` erhaltenen Personen am Bildschirm auszugeben!

Allgemeines

Lies in der Dokumentation der verwendeten Module nach wie die hier zu verwendenden Funktionen funktionieren. Alle Funktionen haben gute Beispiele in der Dokumentation!

Implementiere in `converter.js` die im Punkt „*Die Funktionen*“aufgelisteten Funktionen.

Verwende das Modul `chalk`, um in deinem Programm

- Fehlermeldungen rot
- Successmeldungen grün
- den Output in `printPerson` (siehe unten) blau

auszugeben.

Jede Funktion, die den Pfad zu einer Datei erhält, soll gleich zu Beginn überprüfen, ob die Datei existiert. Existiert die Datei nicht, soll die Funktion ein entsprechendes Fehler-Objekt zurückliefern (siehe *Rückgabewert* der Funktionen)

Happy coding!