Übungsblatt 7: Promises, HTTP und Fetch

Vorstellung in den Tutorien am 13. - 15. Dezember 2022

7.1 JavaScript: Promises

Promises (in anderen Programmiersprachen auch *Futures* genannt) sind ein sehr nützliches Werkzeug für die asynchrone Programmierung mit JavaScript. Sie repräsentieren Platzhalter für Ergebnisse, die erst in der Zukunft eintreffen. Indem man den Promises *Callback-Funktionen* übergibt, kann man Funktionalität definieren, die asynchron ausgeführt wird, also erst dann, wenn das Ergebnis vorliegt.

In der Vorlage finden Sie zwei Dateien: `index.html` und `index.js`. Ergänzen Sie in der `index.js` einen Event-Listener für den `roulette`-Button, der folgendes auslöst:

- Klickt man auf den Button, wird ein Promise erzeugt, welches nach einer Sekunde ein Ergebnis liefert.
- Sofort nach dem Klicken wird das Element `loadingSpinner` angezeigt und dem Element `loadingText` der Text *"Loading..."* eingefügt. Außerdem werden beide Buttons auf der Seite deaktiviert.
- Das Promise hat eine 20%-ige Chance, einen Fehlerfall auszugeben. In desem Fall wird der Nutzer mit einem *Alert* darüber informiert, dass er verloren hat.
- Egal, ob der Nutzer verloren hat oder nicht, werden nach Auflösung des Promises folgende Aktionen ausgeführt:
  - Das Element `loadingSpinner` wird versteckt und der Text des `loadingText`-Elements wird entfernt.
  - Beide Buttons werden wieder aktiviert.
  - Zuletzt wird dem Nutzer mit einem *Alert* für die Teilnahme am Spiel gedankt.

7.2 HTTP

Das Hypertext Transfer Protocol (HTTP) bildet die Grundlage des WWW. Beantworten Sie folgende Fragen:

1. Wie läuft die Kommunikation über HTTP ab? Welches Grundprinzip gilt dabei?
2. Welche HTTP-Methoden gibt es? Wofür werden Sie jeweils verwendet? Welche Methoden sind idempotent bzw. safe?
3. Welche HTTP-Statuscodes kennen Sie?

Postman (`https://www.getpostman.com/`) ist ein Tool, mit dem Sie manuell HTTP Anfragen erstellen und abschicken können. Nutzen Sie es, um eine GET-Anfrage an die URL `https://randomuser.me/api/` zu senden. Betrachten Sie dabei die Bestandteile der Anfrage und Antwort. Was liefert Ihnen diese API zurück?

7.3 JavaScript: Die fetch-API

Die *fetch*-API erleichtert es Webentwicklern, Inhalte asynchron mit HTTP-Anfragen zu laden. Promises spielen hierbei eine zentrale Rolle, da mittels des fetch-Befehls für eine HTTP-Anfrage ein Promise erzeugt wird, welches aufgelöst wird, wenn die Antwort auf die Anfrage eintrifft.

Ergänzen Sie die index.js um Event-Listener für den `meet`-Button, die folgendes auslösen:

- Das Element `card-body` wird angezeigt. Dies soll nur beim ersten Klicken geschehen, der Event-Listener hierfür muss also anschließend entfernt werden.
- Es wird eine GET-Anfrage an die in `picUrl` definierte API gesendet. Die API liefert daraufhin die URL für ein Hundebild zurück, welche als src für das Bildelement `profilePic` gesetzt wird.
- Außerdem wird eine GET-Anfrage an die in `dataUrl` definiert API gesendet. In der vorherigen Aufgabe haben Sie bereits manuell eine Anfrage an diesen Endpoint gesendet und wissen daher, dass die API ein zufällig generiertes Nutzerprofil als JSON zurückgibt.

  Zeigen Sie die Profildaten wie folgt auf der Seite an:

  - Das Element `name` soll den Vornamen (`name.first`) enthalten,
  - `age` das Alter (`dob.age`),
  - `city` die Stadt (`location.city`),
  - und `country` das Land (`location.country`).