Modul 294 - Aufgabenblatt 05

von Lukas Meier

Unterricht vom 13.01.2026

1 Vorbereitung

Erstellen Sie einen neuen Ordner und speichern Sie die Datei `index.html` aus dem Netzwerkordner darin. Öffnen Sie die Datei im Browser sowie in einem Code-Editor Ihrer Wahl und stellen Sie sicher, dass ein `script`-Tag vorhanden ist.

2 Synchroner und asynchroner Code

Schreiben Sie zwei `console.log`-Ausgaben direkt untereinander. Erstellen Sie anschliessend einen `setTimeout`, welcher nach 2 Sekunden eine weitere Meldung in der Konsole ausgibt.

Ein Timeout kann in JavaScript mit der folgenden Syntax realisiert werden: `setTimeout(function, millisekunden)`. Damit wird die funktion im ersten Parameter nach der anzahl Millisekunden des zweiten Parameters ausgeführt.

Beobachten und beschreiben Sie die Reihenfolge der Ausgaben.

3 Erstes Promise

Erstellen Sie ein Promise, welches nach 1 Sekunde erfolgreich aufgelöst wird und den Text „Promise erfüllt“ zurückgibt.

Geben Sie das Resultat mithilfe von `then` in der Konsole aus.

4 Promise mit Fehler

Erweitern Sie das Promise aus der vorherigen Aufgabe so, dass es alternativ abgelehnt wird.

Fangen Sie den Fehler mit `catch` ab und geben Sie eine sinnvolle Fehlermeldung in der Konsole aus.

5 Asynchrone Funktion

Schreiben Sie eine Funktion `ladeDaten()`, welche mit dem Keyword `async` definiert ist und einen Text zurückgibt.

Rufen Sie die Funktion auf und geben Sie den Rückgabewert in der Konsole aus. Beobachten Sie, welchen Datentyp der Rückgabewert hat.

6 await verwenden

Erstellen Sie innerhalb einer `async`-Funktion ein Promise, welches nach 2 Sekunden den Wert `42` zurückgibt.

Nutzen Sie `await`, um den Wert in einer Variable zu speichern, und geben Sie diese anschliessend aus.

7 Fehlerbehandlung mit try / catch

Erweitern Sie die Funktion aus der vorherigen Aufgabe so, dass das Promise auch fehlschlagen kann.

Behandeln Sie diesen Fall mit einem `try / catch`-Block und geben Sie eine Fehlermeldung in der Konsole aus.

8 Fetch – erste Anfrage

Nutzen Sie die `fetch`-Methode, um Daten von folgendem Endpoint zu laden:

`https://jsonplaceholder.typicode.com/todos/`

Geben Sie die geladenen Daten vollständig in der Konsole aus. Verwenden Sie hierfür `then`.

(Denken Sie daran, dass Sie auf der Response von `fetch` noch die Methode `json` aufrufen müssen, welche wiederum ein Promise zurückgibt.)

9 Fetch mit async / await

Schreiben Sie die Fetch-Anfrage aus der vorherigen Aufgabe um, sodass sie mithilfe von `async / await` umgesetzt ist.

Lagern Sie den Code in eine Funktion `loadTodos()` aus und rufen Sie diese beim Laden der Seite auf.

10 ToDos im DOM darstellen

Erstellen Sie im HTML ein leeres `ul`-Element.

Stellen Sie nun alle geladenen ToDos als `li`-Elemente dar. Jedes ToDo soll mindestens folgende Informationen anzeigen:

- Titel
- Status (completed / not completed)

Kennzeichnen Sie erledigte ToDos visuell (z. B. durch Text oder CSS-Klasse).

11 User zu ToDos laden

Laden Sie zusätzlich die Benutzerdaten vom folgenden Endpoint:

`https://jsonplaceholder.typicode.com/users/`

Die ToDos enthalten eine `userId`, welche einem Benutzer entspricht.

Ordnen Sie jedem ToDo den passenden Benutzernamen zu und zeigen Sie diesen ebenfalls im `li`-Element an.

12 Kombinierte Darstellung

Erweitern Sie die Darstellung der ToDos so, dass pro Eintrag folgende Informationen sichtbar sind:

- Titel des ToDos
- Name des Benutzers

- Status des ToDos

Achten Sie darauf, dass die Daten sauber strukturiert und übersichtlich dargestellt sind.

13 Optional: Filter

Erweitern Sie die Anwendung optional um eine Filterfunktion:

- Ein Button zeigt nur erledigte ToDos
- Ein weiterer Button zeigt nur nicht erledigte ToDos

Nutzen Sie hierfür bereits geladene Daten und führen Sie keine erneuten Fetch-Anfragen aus.