Übung: Entwicklung einer Todo-App

Speicherung der Daten im JSON-Format

In diesem Übungsblatt erweitern wir die Todo-App um eine Speicherfunktion für erstellte Aufgaben. Wir beginnen mit einer lokalen Datenspeicherung im Browser des Benutzers. Später wechseln wir zu einer zentralen Datenbank und beleuchten die Unterschiede sowie deren Vor- und Nachteile. Für die Speicherung verwenden wir das weit verbreitete JSON-Format.

Vorbereitung

Für diese Aufgabe müsst ihr die vorherige Aufgabe *"Die GUI mit HTML, CSS und JavaScript"* abgeschlossen haben. Die Aufgaben-App ist der Startpunkt für die folgenden Schritte.

Aufgabe 1: Aufgaben als JSON modellieren

Bervor wir uns um die Speicherung kümmern, überlegen wir, wie wir eine Aufgabe als Informationsobjekt modellieren und in einem geeigneten Format abspeichern können.

- Informiert euch über das JSON-Format, ein weit verbreitetes Datenformat im Internet. Überlegt euch anschließend, wie eine Aufgabe aus euerer Aufgaben-App in diesem Format aussehen könnte! Schreibt dafür ein paar Beispiele auf!
- Wie können wir mehr als eine Aufgabe im JSON-Format abbilden? Erstellt ein Beispiel mit 3 Aufgaben!
- Erstellt ein JSON-Objekt in eurer `script.js`, das mindestens 2 beispielhafte Aufgaben enthält. Diese Aufgaben sollen beim Aufruf der Seite anstelle der bisher im HTML-Code befindlichen Aufgaben angezeigt werden!

Aufgabe 2: Aufgaben in eurem Browser speichern

Die Web Storage API bietet die Möglichkeit, Daten lokal im Browser zu speichern. Dabei gibt es zwei unterschiedliche Varianten: Der `sessionStorage` speichert die Daten nur für eine Sitzung des Browsers und löscht sie beim Schließen der Webseite. Der `localStorage` dagegen speichert die Daten über die Sitzung hinaus. Wir verwenden deshalb letztere für die Sicherung der Aufgaben.

- Testet den `localStorage` und speichert das oben erstellte JSON-Objekt mit den Beispielaufgaben unter dem Key `todos`! Überprüft das Ergebnis in der Developer-Ansicht im Browser! ACHTUNG: der `localStorage` kann nur sogenannte Strings, also Zeichenketten, speichern! Schaut euch deshalb die Funktion `JSON.stringify()` an!
- Stellt nun sicher, dass die Todos aus dem `localStorage` gelesen werden und beim Aufruf der Webanwendung angezeigt werden! Löscht das JSON-Objekt mit den *hart codierten* Aufgaben aus eurer `script.js` heraus!
- Ändert die Funktion `addTodo()` so, dass neue Aufgaben den im `localStorage` befindlichen Aufgaben hinzugefügt werden! Behandelt auch den Fall, wenn ein Benutzer die Anwendung zum ersten Mal nutzt und somit noch keine Daten im `localStorage` vorhanden sind!
- Reflektiert die Statusänderung einer Aufgabe (offen/erledigt) ebenfalls im `localStorage`!
- Der Benutzer soll Aufgaben als wichtig kennzeichnen können! Erstellt dafür ein zusätzliches Feld und überlegt euch eine Möglichkeit, wie der Benutzer das über die GUI umsetzen kann! Wichtige Aufgaben sollen hervorgehoben werden und gleichzeitig am Anfang der Liste erscheinen!
- Löscht alle Aufgaben aus dem `localStorage`, wenn der Benutzer den Button `Liste leeren` anklickt!

Damit sind wir mit dem zweiten Teil der Aufgabe fertig. Unsere Aufgaben werden zumindest lokal in unserem Browser gespeichert. Später werden wir uns darum kümmern, unsere Daten auch über beliebig viele Geräte hinweg synchronisieren zu können. Zunächst geht es im dritten Teil aber darum, dass Aussehen unserer Anwendung zu verbessern.