Technische Hochschule Ingolstadt
Fakultät für Elektrotechnik und Informatik

Aufgabenblatt 2: Webclient-Programmierung mit JavaScript

Ziel dieses Aufgabenblattes ist es, sich mit JavaScript sowie der Webseiten-Manipulation mithilfe von JavaScript zu beschäftigten. Hierfür soll der HTML/CSS-Prototyp zur Adressverwaltung zu einer funktionierenden browserbasierten Anwendung ausgebaut werden. Die resultierende Anwendung ist selbstverständlich nur für den *single user*-Betrieb als *standalone*-Anwendung gedacht! Würde man die Funktionalität mit einer Server-Anbindung umsetzen, könnte man die hier vorgesehene Lösung jedoch für einen Offline-Betrieb zusätzlich einbauen und bei Bestehen einer Serververbindung entsprechend mit dem Server synchronisieren.

Im Laufe der Bearbeitung des Aufgabenblattes werden insbesondere folgende JavaScript[1]-Features vertieft:

- Objekte und anonyme Funktionen: Das Fachobjekt „Adresse“ wird in Form eines „Data Transfer Objects“ (Klasse „AdresseDTO“) definiert, um überall, wo eine Adresse benötigt wird, mit Objekten anstelle von Einzelwerten arbeiten zu können. Im Rahmen der Definition von Objekten werden anonyme Funktionen (also Funktionen ohne Namen) benötigt.
- Nutzung des HTML5-Features „LocalStorage“: Die Adressen werden in *LocalStorage* dauerhaft gespeichert („persistiert“), um bei späteren Aufrufen der Seiten auf bereits eingepflegte Adressen zugreifen zu können. Zur Modularisierung der Datenzugriffe wird ein sogenanntes „Data Access Object“ als Instanz der Klasse „AdressenDAO“ definiert, welches sämtliche Zugriffe auf eine persistente Liste von Adressen ermöglicht.
- Sortieren von Objekt-Arrays mit JavaScript: Analog zu den Möglichkeiten, die Java bietet, können auch in JavaScript Arrays sortiert werden. Hierfür bietet jedes Array-Objekt die Methode „sort()“ an. „sort()“ kann mit einer Funktion parametrisiert werden, die zwei Parameter für den paarweisen Vergleich zweier Objekte (hier AdresseDTOs) verarbeitet. Die Funktion muss ein zur Java-Methode Comparable.compareTo() analoges Ergebnis zurückliefern. Die Sortierfunktionalität wird für die Umsetzung der Suchfunktion benötigt.

Ein weiterer wichtiger Aspekt ist die direkte Manipulation des Document Object Models (DOM) der im Browser dargestellten Seite. Hierfür gibt es ein von JavaScript aus nutzbares API (s. http://krook.org/jsdom/), welches hier für folgende Aufgaben genutzt werden soll:

- Elemente (z.B. Eingabefelder) anhand ihrer „id“ finden, vgl. „document.getElementById()“
- Befüllen der Personentabelle aus der persistenten Personenliste: Hier wird die Tabelle zeilenweise neu aufgebaut (mit Ausnahme der Header-Zeile).
- Tabellenzeilen löschen: Der Benutzer kann Personenbeschreibungen löschen, woraufhin die entsprechende Zeile aus der Personentabelle gelöscht werden

[1] Siehe http://www.w3schools.com/js/default.asp, dort gibt es ein JavaScript-Tutorial.

muss. Zudem werden beim Befüllen der Personentabelle nach Änderungen (z.B. Anlegen einer Person) zuerst sämtliche Zeilen der Tabelle mit Ausnahme der Kopfzeile gelöscht, bevor für jede Personenbeschreibung eine neue Tabellenzeile eingefügt wird.
- Buttons und Images erzeugen für die Aktion-Spalte der Personentabelle

Aufgabenstellung

Machen Sie sich nach dem Download und dem Entpacken des Archivs „Aufgabe2-Lösungsfragment.zip“ mit dem darin enthaltenen Code-Fragment vertraut. Leider ist es nicht ausführbar, da der JavaScript-Code Lücken – gekennzeichnet durch Zeilen der Art `// *** (1) ***` – enthält.

Bevor Sie die Lücken im JavaScript-Code schließen, macht es Sinn, die in den HTML-Dateien fehlenden ID-Attribute der UI-Controls zu ergänzen. Welche benötigt werden, erkennen Sie im JavaScript-Code anhand der Verwendung der Funktion „getElementById()“. Finden Sie heraus, was diese Funktion leistet bzw. wofür sie verwendet wird und ergänzen Sie die benötigten IDs.

Ihre Aufgabe besteht nun darin, die Lücken (1) bis (6) durch geeigneten Programmcode zu schließen, so dass das Programm korrekt ausführbar ist. Im Folgenden werden die Lücken kurz erläutert:

Lücke (1): Funktion AdresseDTO.pruefe()

Ergänzen Sie den Funktionskörper geeignet, so dass die Funktion gemäß der Funktionsbeschreibung (s. Quelltext) arbeitet. Ein korrektes Adressobjekt liegt dann vor, wenn folgende Bedingungen erfüllt sind:
- Sämtliche Attribute müssen belegt sein
- Die E-Mail muss ein korrektes Format besitzen. Hinweis: zur Prüfung auf Korrektheit kann die Methode validateEmail() genutzt werden.
- Die Postleitzahl muss eine positiven numerischen Wert besitzen

Lücke (2) und (3): Hilfsfunktionen für die Funktion AdressenDAO.findeZuFilterUndSortiere()

Die Funktion findeZuFilterUndSortiere() wird benötigt, wenn der „suchen“-Button gedrückt wird. Sie liefert ein Array von Adressen, die die im Filter spezifizierten Kriterien erfüllen. Das Array ist gemäß dem gewählten Sortierkriterium sortiert. Zur Realisierung der Funktion werden die beiden Hilfsfunktionen „filter()“ bzw. „sortiereAdressenListe()“ verwendet, die gemäß ihrer Beschreibung (s. Quellcode) umzusetzen sind.

Lücke (4): Funktion loescheAdresse()

Die Funktion AdressenDAO.loeschePerson() löscht das AdresseDTO-Objekt mit der übergebenen „id“ aus dem Adressen-Array „adressenArray“. Zu beachten ist, dass das Array nach dem „Entfernen“ des zu löschenden Objekts zu persistieren ist! Vergleichen Sie hierzu die Funktion „aktualisiereAdresse()“. Beachten Sie weiterhin,

dass nur logisch gelöscht wird, d.h. das zu löschende Objekt erhält eine ID von -1. Es wird daraufhin bei der Suche ignoriert, beim Neueintragen einer Adresse wird der Array-Eintrag für das neue Objekt wiederverwendet.

Lücke (5) und (6): Funktion belegeZeile()

Aufgabe der Funktion „belegeZeile()“ ist es, die übergebene Adresse vom Typ AdresseDTO in eine neue Zeile der übergebenen Tabelle „table“ zu überführen. Während die Zelle für das Attribut „name“ korrekt erzeugt und befüllt wird, fehlen die weiteren Zellen (Lücke (5)). Analog verhält es sich mit den Buttons: der Edit-Button wird korrekt erzeugt und initialisiert, der Code für den Delete-Button ist zu ergänzen (Lücke (6)).

<u>Wichtige Hinweise:</u>

- Die Abnahme besteht aus 2 Teilen, erstens aus der Vorführung des Prototypen und zweitens aus Fragen zu dessen Funktionsweise (auf Quelltextebene!) sowie zu den vorgenommenen Änderungen.
- Als Webbrowser sei Google Chrome empfohlen. Dieser enthält Entwicklertools (unter Tools > Entwicklertools) mit folgenden Funktionalitäten (vgl. Abbildung 1):
    - Darstellung des HTML-Dokumentbaums (Registerkarte „Elements“)
    - Darstellung und Manipulation der Ressourcen einer Seite. Hier benötigen Sie vor allem den Bereich „Local Storage“ (unter Registerkarte „Application“), um die gespeicherten Objekte einsehen bzw. löschen zu können.
    - JavaScript-Debugging (unter Registerkarte „Sources“)

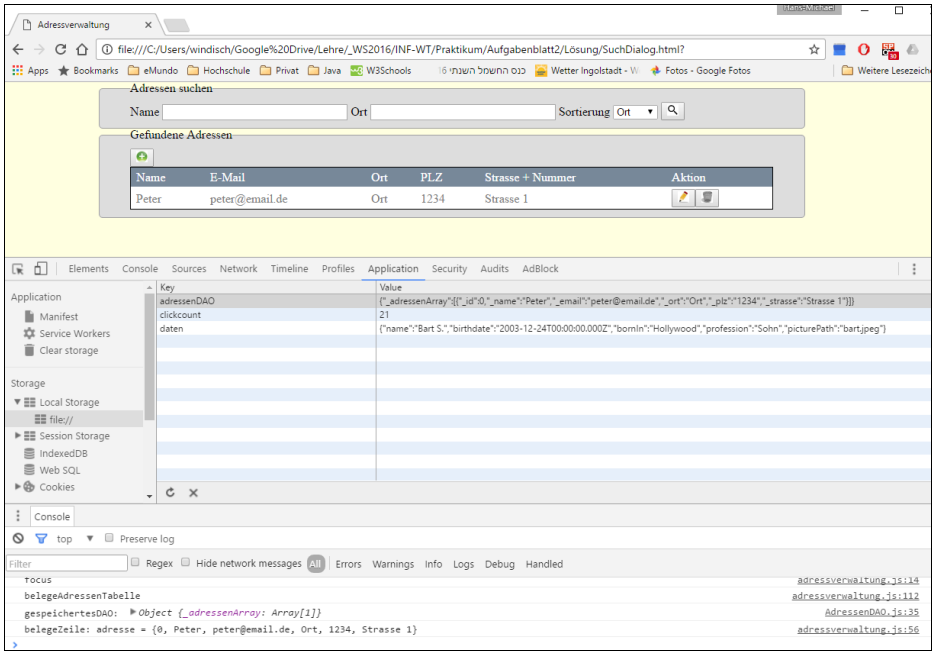

Abbildung 1: Google Chrome Entwicklertools