Übungsblatt 6 WT:V

Bis zum 13.07.2022, 23:59, sind Lösungen zu folgenden Aufgaben abzugeben: 1, 2, 3, 4, 5.

Aufgabe 1 : Multiple Choice (1+1+1 Punkte)

Kreuzen Sie Zutreffendes an:

(a) Welche der folgenden Aussagen bezüglich DOM und JavaScript sind korrekt?

- ☐ DOM wird ausschließlich von JavaScript implementiert.
- ☐ Es lässt sich ermitteln, welchen DOM-Level ein Browser implementiert.
- ☐ Neue Browser müssen DOM-Level 3 implementieren.

(b) Die JavaScript Object Notation (JSON) ...?

- ☐ gestattet das dynamische Nachladen von Funktionalität.
- ☐ wird mit Hilfe der Funktion `eval()` deserialisiert.
- ☐ wird von JavaScript nativ unterstützt.
- ☐ ist eine Teilmenge der JavaScript-Syntax für Objektliterale.

(c) Folgende Konzepte gelten für die Programmiersprache JavaScript:

- ☐ Der JavaScript-Code eines Dokumentes wird beim Einlesen durch einen Browser, der JavaScript aktiviert hat, sofort ausgeführt.
- ☐ Bei JavaScript handelt es sich um eine interpretierte Variante von Java.
- ☐ JavaScript realisiert Vererbung mittels Klassen.
- ☐ JavaScript realisiert Vererbung mittels Prototypen.

Beachten Sie, dass zu einer Frage mehrere Antworten zutreffen können. Eine Frage gilt als richtig beantwortet, falls alle zutreffenden und keine unzutreffende Antwort angekreuzt ist.

Aufgabe 2 : JavaScript: Statische/dynamische Semantik; Einbettung (1+2+1 Punkte)

(a) Was ist der Unterschied zwischen statischer und dynamischer Semantik einer Programmiersprache?

(b) Welche der folgenden Analysen sind bei JavaScript Teil der statischen und welche Teil der dynamischen Semantik? Weisen Sie diese zu und vergleichen Sie die Einordnung mit der Programmiersprache Java.

(b1) Überprüfung von Array-Indizes

(b2) Effekte, die durch die Ausführung von Sprachkonstrukten entstehen

(b3) Überprüfung der Existenz von Funktionsdefinitionen, die in Aufrufen vorkommen

(b4) Test auf Typkonformität zwischen Operanden und Operationen

(c) Nennen Sie mindestens zwei Möglichkeiten, JavaScript-Code in HTML einzubetten.

Aufgabe 3 : JavaScript: Stack (2+2+2 Punkte)

Verwenden Sie für diese Aufgabe die Datei `stack.html` aus dem Repository zur Übung. Erstellen Sie die zugehörige `stack.js` und implementieren Sie darin die unten geforderten Funktionalitäten und sowie eine Funktion `initialize()`, die nach abgeschlossenem Laden des Dokumentes aufgerufen wird (siehe `stack.html`). Ändern Sie für (a) und (b) die HTML-Datei *nicht*, sondern benutzen Sie ausschließlich die DOM-API und keine Bibliotheken Dritter!

(a) Jedes mal wenn der „Push"-Knopf gedrückt wird, soll der `ul`-Liste ein neues Listenelement mit dem Inhalt des Textfeldes `pushText` unten angefügt werden. Hinweise:

- Erzeugung von Elementen: `let` *Element* `= document.createElement("`*Tag-Name*`");`
- Einfügen von Elementen: *Elternelement*`.appendChild(`*Element*`);`
- Zugriff auf den Wert eines Input-Elements mittels *Element*`.value`

(b) Jedes mal wenn der „Pop"-Knopf gedrückt wird, soll das letzte Listenelement der `ul`-Liste entfernt werden, und der Inhalt des Textfeldes `popText` auf den Inhalt des entfernten Listenelementes gesetzt werden. Hinweise:

- Löschen von Elementen: *Elternelement*`.removeChild(`*Element*`);`
- Anzahl an Elementen in einer `HTMLCollection`: *Kollektion*`.length`

(c) Die Funktion `initialize()` wird aktuell von der `stack.html` aus per `onload`-Attribut im `body` aufgerufen. Eine modernere Alternative ist die Methode `addEventListener()`. Erklären Sie in jeweils 1-3 Sätzen zwei wichtige Vorteile der Nutzung von `addEventListener()`. Passen Sie das HTML-Dokument und Ihre JavaScript-Datei so an, dass statt dem `onload`-Attribut `addEventListener()` verwendet wird.

Empfohlene Quellen: MDN: addEventListener(), MDN: Event Liste

Aufgabe 4 : JavaScript: Cross-Site Scripting (XSS) (1+1+1+2 Punkte)
Informieren Sie sich über die Problematik des „Cross-Site Scripting", abgekürzt XSS.

(a) Was bedeutet der Begriff der „Code-Injektion"?

(b) Wie lässt sich Code-Injektion vorbeugen?

(c) Es werden im Allgemeinen drei Arten von XSS-Sicherheitslücken in Web-Seiten unterschieden, nämlich DOM-basierte, nicht-persistente und persistente. Beschreiben Sie das Prinzip hinter jeder der drei Sicherheitslücken und wie ein Angreifer sie ausnutzen kann, um Benutzer der Web-Seite auszuspionieren.

(d) Im Repository zur Übung finden Sie die Dateien `xss-search.html` und `xss-search-result.html`. Erstere ist die Einstiegsseite eines Suchskriptes, wie es häufig auf Web-Seiten zu finden ist. Nehmen Sie an, dass Sie letztere Datei als Ergebnis ihrer Suchanfrage erhalten, wenn das Suchwort nicht gefunden wurde. Sie wissen, dass der Server die Formulareingaben nicht verändert. Welche der vorgenannten XSS-Sicherheitslücken ist in diesem Beispiel enthalten? Begründen Sie Ihre Antwort und geben Sie an wie sie den JavaScript-Code `alert("XSS");` in die Seite injizieren können, so dass er beim Anzeigen der Seite ausgeführt wird.

Aufgabe 5 : Programmiermodul: JavaScript [P] (2+2+5+0 Punkte)

Erweitern Sie Ihre Notes-App mit JavaScript so, dass Einträge ohne Neuladen der Seite hinzugefügt und bearbeitet werden können. Wenn Sie Ihre eigene Implementierung weiterverwenden, kopieren Sie die für die Aufgaben vorgegebenen neuen Funktionen aus dem Repository zur Übung in Ihre Codebasis.

Schreiben Sie Ihr JavaScript in der Datei `static/script.js`, die im Fuß des HTML-Templates eingebunden wird.

(a) Fügen Sie dem Formular für das Erstellen eines neuen Eintrags per `addEventListener` einen Handler für das `submit`-Event hinzu, der das Absenden des Formulars verhindert. Stattdessen sollen die Formulardaten über die vorgegebene Funktion `sendPostRequest` an den Server gesendet werden. Übergeben Sie der Funktion dafür die Zieladresse, ein `FormData`-Objekt mit dem aktuellen Inhalt des Formulars und eine Callback-Funktion für die Serverantwort.

(b) Die Funktion `sendPostRequest` setzt den HTTP-Header `Accept: application/json`, der für HTTP-Content-Negotiation verwendet werden kann. Auf der Server-Seite können Sie die vorgegebene Funktion `client_wants_json` nutzen, um zu prüfen, ob dieser Header gesetzt wurde.

Passen Sie den Flask-Routen-Handler für `"/create"` so an, dass in diesem Falle (und nur dann) statt des Templates ein JSON-Reponse mit dem `Content-Type: application/json` zurückgegeben wird:

- Wurde der Eintrag hinzugefügt, geben Sie den neuen Eintrag als JSON zurück. Sie können für die Serialisierung der Datenklasse die vorgegebene Funktion `serialize_json` nutzen.
- Geben Sie im Fehlerfall die Fehlermeldung als JSON mit HTTP-Status 400 zurück (z.B.: `{"error": "Titel ist erforderlich!"}`).

Stellen Sie sicher, dass im Falle eines JSON-Responses keine HTTP-Weiterleitung stattfindet.

(c) Generieren Sie mit der JSON-Antwort vom Server neue DOM-Elemente, die Sie an passender Stelle mit allen notwendigen Event-Handlern in die Seite einfügen. Im Erfolgsfall soll so per JavaScript ein neuer Notiz- oder Todo-Eintrag eingefügt werden. Im Fehlerfall zeigen Sie die entsprechende Fehlermeldung an.

(d) **Bonus (2 Punkte):** Fügen Sie allen „Löschen“-Buttons `click`-Event-Handler hinzu und implementieren Sie so das Löschen von Einträgen mittels asynchroner JSON-Requests.

Hinweise zur Abgabe

- Erstellen Sie *eine* PDF-Datei, in der Texte und Grafiken zu den Aufgaben enthalten sind.
- Quellcode (Python, Java, JavaScript, PHP, HTML, CSS, XML, XSL, XSD, etc.) innerhalb des PDF-Dokuments wird nicht korrigiert. Quellcode jeder Aufgabe soll, wie im Tutorium vorgestellt, über GitLab verwaltet werden.
- Stellen Sie sicher, dass Sie keine in der `.gitignore`-Datei gelisteten Dateien und Verzeichnisse versehentlich mit committen.
- Referenzieren Sie Ihre Quellcode-Dateien in dem PDF-Dokument, so dass sie einer Aufgabe eindeutig zuzuordnen sind. Verlinken sie die entsprechenden Repositories.
- Gruppenabgaben mit bis zu drei Personen sind erlaubt; pro Gruppe genügt dann eine Abgabe, in der alle Gruppenmitglieder mit Namen, Matrikel, und E-Mail-Adresse verzeichnet sind.
- Abzugeben ist *eine* PDF-Datei über Moodle.