Praktikum zu Web-Technologien/Web-Engineering, Wintersemester 2022/2023

Praktikum 10 (DOM, Events, Node.js, CTF)

Im Rahmen des Praktikums entwickeln wir eine Web-Anwendung, welche wir Schritt für Schritt mit weiteren Anforderungen, Funktionen und Technologien erweitern.

Hinweis zu Bonuspunkten: Zur Vergabe der Bonuspunkte werden wir den von Ihnen produzierten Quellcode begutachten, sobald Sie bestimmte *Meilensteine* erreicht haben. Sobald ein Praktikum einen solchen Meilenstein markiert, finden Sie einen entsprechenden Hinweis auf dem jeweiligen Praktikumsblatt. Dieser Hinweis beinhaltet auch, wieviele Bonuspunkte für den jeweiligen Meilenstein erzielbar sind. Da die Praktika aufeinander aufbauen, enthält ein Meilenstein jeweils alle vorhergehenden Praktikumsaufgaben - zur Erlangung aller Bonuspunkte sind also alle Aufgaben zu bearbeiten. Um die Bonuspunkte zu erhalten, sprechen Sie uns einfach im Praktikum an, sobald Sie einen Meilenstein erreicht haben. Wir schauen uns dann gemeinsam Ihren Stand in einer kleinen Abnahme an. Dabei muss Ihre Gruppe, in welcher Sie die Aufgabe gelöst haben, vollständig anwesend sein. Insgesamt gibt es drei Meilensteine, und es sind maximal 10 Bonuspunkte über das Praktikum erreichbar. *Wichtig:* Abnahmen erfolgen nur bis maximal 20.01.23 (Ende der Vorlesungszeit)!

Hinweis zum entstehenden Code:

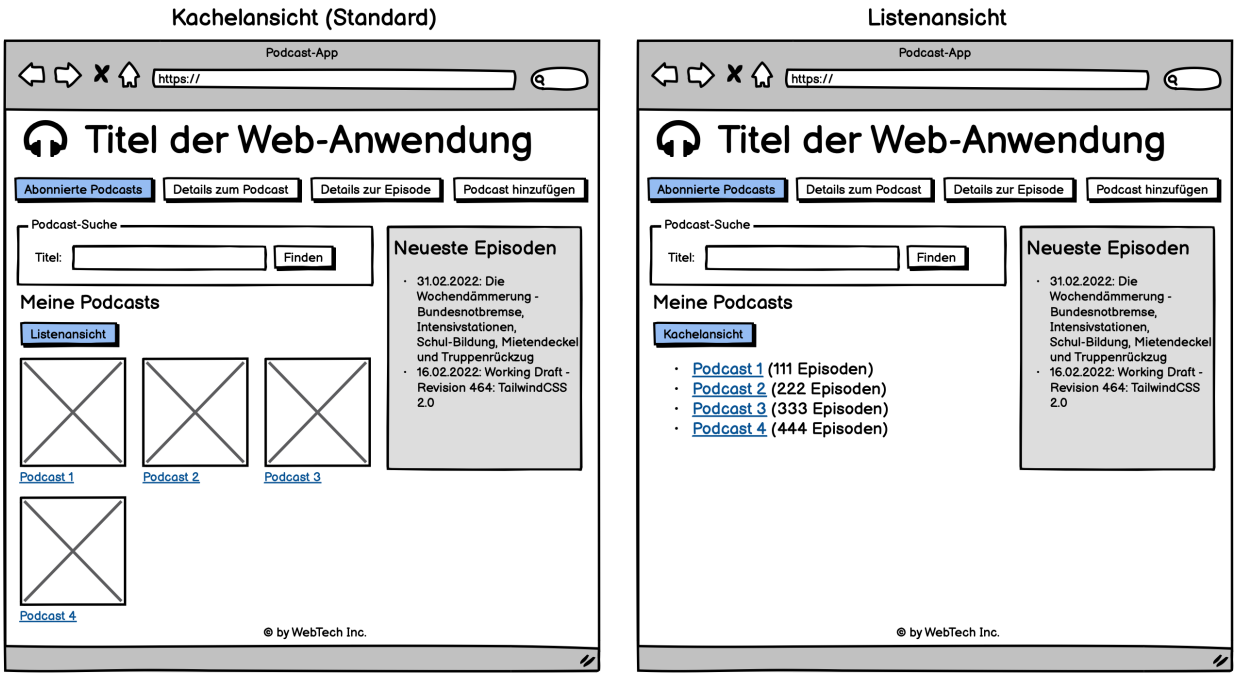
- Zur Verwaltung und zum kollaborativen Bearbeiten Ihres Quellcodes empfehlen wir Ihnen die Nutzung von Git (z.B. in Verbindung mit dem GitLab-Server des FB4, Login per FH-Account). Eine kurze Einführung in Git finden Sie hier.
- Falls Sie Git nicht verwenden, so legen Sie Ihren Quellcode pro Praktikumsaufgabe in separaten Verzeichnissen (z.B. „praktikum2“, „praktikum3“) ab, damit die einzelnen Entwicklungsschritte bei den Abnahmen ersichtlich sind.

Aufgabe 1: Dynamisches Umschalten der Ansicht

In Praktikum 7 (Aufgabe 1) haben wir eine alternative Version der *Liste der abonnierten Podcasts* erstellt, welche statt einer ungeordneten Aufzählung eine Kachelansicht anzeigt. Diese beiden Ansichten (`index.html` und `index2.html`) führen wir nun zusammen, so dass am Ende dieser Aufgabe wieder nur die Datei `index.html` existiert.

Nutzen Sie die Möglichkeiten der *DOM-Manipulation* und des *Event-Handlings* mit

JavaScript, um die Ansicht dynamisch umschaltbar zu machen, wie in folgendem Mockup dargestellt:



Gehen Sie dabei wie folgt vor:

1. Ergänzen Sie die Seite um einen Button, über welchen die Ansicht gewechselt werden kann. Befindet sich die Seite in der Listenansicht (d.h. die Podcasts werden als ungeordnete Liste mit der Anzahl der Episoden angezeigt), so soll die Beschriftung des Buttons "Kachelansicht" lauten. Befindet sich die Ansicht in der Kachelansicht, so soll der Button entsprechend mit "Listenansicht" beschriftet sein.

2. Sorgen Sie dafür, dass die Ansicht bei Klick auf den Button in die jeweils andere Ansicht (Liste oder Kacheln) wechselt. **Wichtig:** Dabei soll kein Seitenwechsel stattfinden (also keine Verlinkung auf eine andere HTML-Seite). Stattdessen soll der Inhalt der Seite *dynamisch per JavaScript* umgebaut werden. Lagern Sie den benötigten JavaScript-Code in einem neuen externen Skript `index.js` und binden Sie dieses geeignet ein. **Lösen Sie die Aufgabe ohne Verwendung von `innerHTML`.**

3. Beim erstmaligen Öffnen der Seite soll standardmäßig die Kachelansicht aktiviert sein.

4. Behalten Sie ansonsten alle Stylings und Layouts aus den vorherigen Praktika bei. Lagern Sie insbesondere die Regeln für die Kachelansicht weiterhin im externen Stylesheet `index.css` und binden Sie dieses in `index.html` geeignet ein.

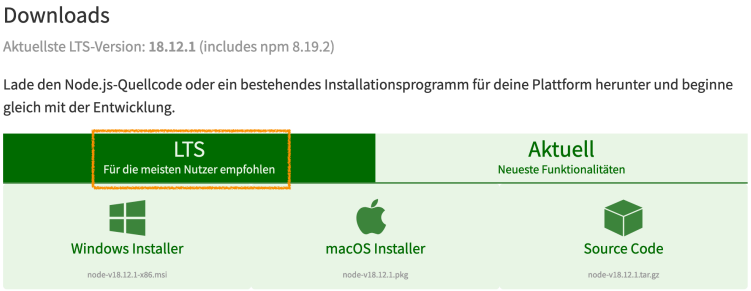
5. Löschen Sie nach erfolgter Umstellung die `index2.html` und entfernen Sie die entsprechende Verlinkung aus der Navigation.

Aufgabe 2: Node.js einrichten und ausprobieren

In dieser Aufgabe bereiten wir die folgenden Praktika vor. Ziel der Aufgabe ist es, eine lauffähige Node.js-Umgebung zu haben.

1. Laden Sie den für Ihr Betriebssystem passenden Node.js-Installer hier herunter: https://nodejs.org/de/download/

   Hinweis: Bitte wählen Sie unbedingt die LTS-Variante statt der aktuellsten (current) Version, um Stabilitätsprobleme zu vermeiden:

2. Testen Sie Ihre Node.js-Installation:

   1. Öffnen Sie eine Konsole, z.B.:
      - Eingabeaufforderung auf Windows 10: Tastenkombination *Windows + R* drücken, im sich öffnenden Fenster "cmd" eingeben und Taste *Enter* drücken
      - Terminal auf MacOS: Tastenkombination *⌘ + Leertaste* drücken, im sich öffnenden Fenster "terminal" eingeben und Taste *Enter* drücken

   2. Geben Sie in der Konsole den Befehl `node -v` ein. Falls Node.js korrekt installiert ist, sollte nun die Node-Version angezeigt werden (z.B. "v18.12.1").

3. Speichern Sie das Programm aus Aufgabe 1 (JS CODING KATA) von Übungsblatt 8 in eine Datei namens `tictac.js` in Ihr "praktikum09"-Verzeichnis (lösen Sie die Aufgabe ggf. zuerst, falls noch nicht geschehen).

4. Wechseln Sie nun auf der Konsole (siehe Schritt 2, Punkt 1) in Ihr "praktikum10"-Verzeichnis (Befehl `cd <Pfad zum Verzeichnis>`) und führen Sie dort den folgenden Befehl aus: `node tictac.js`. Unser Programm sollte ausgeführt werden und eine Ausgabe wie diese produzieren:

```
sven@tsu:~/Documents/lehre/webtechnologien/material/veranstaltung/praktikum/praktikum11 (sose19) $ node tictac.js
1 2 Tic 4 Tac Tic 7 8 Tic Tac 11 Tic 13 14 TicTac 16 17 Tic 19 Tac Tic 22 23 Tic Tac 26 Tic 28 29 TicTac 31 32 Tic 34 Tac T
ic 37 38 Tic Tac 41 Tic 43 44 TicTac 46 47 Tic 49 Tac Tic 52 53 Tic Tac 56 Tic 58 59 TicTac 61 62 Tic 64 Tac Tic 67 68 Tic
Tac 71 Tic 73 74 TicTac 76 77 Tic 79 Tac Tic 82 83 Tic Tac 86 Tic 88 89 TicTac 91 92 Tic 94 Tac Tic 97 98 Tic Tac
```

Tipp: In Visual Studio Code können Sie über einen Rechtsklick auf die Datei und Auswahl des Menüpunktes *Open in Terminal* eine integrierte Konsole öffnen.

Allgemeine Hinweise:

- Validieren Sie Ihren HTML-Code mit dem W3C Markup Validator. Der entstehende HTML-Code soll beim Check keine Fehler und Warnungen produzieren.
- Ausnahme: Warnungen bzgl. `date` - und `time` -Eingabefeldern können Sie ignorieren.
- Validieren Sie Ihren CSS-Code mit dem W3C CSS Validation Service. Der entstehende CSS-Code soll beim Check keine Fehler und Warnungen produzieren.

Zusatzaufgabe 3: Capture the Flag (CTF)

Die folgende Aufgabe ist freiwillig – es gibt für diese Aufgabe keine Bonuspunkte (nur Ruhm, Ehre, Spaß, etc. pp.)!

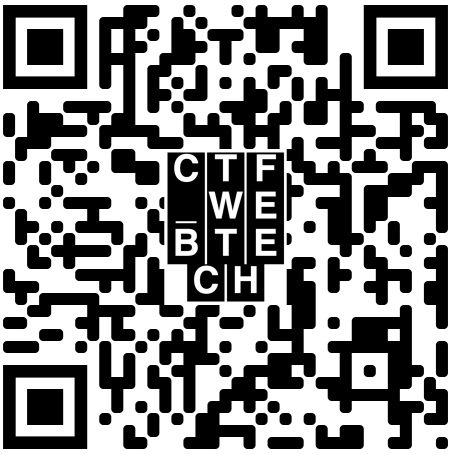
In dieser freiwilligen Zusatzaufgabe gibt es eine neue Challenge vom Typ *Capture the Flag* (CTF, siehe Praktikum 3 für die grundlegende Beschreibung des Prinzips). Auch dieses Mal ist das Flag wieder eine Zeichenkette, die mit `ctf_` beginnt (z.B. `ctf_This_i5_the_Flag_forma7`).

Finden Sie das Flag, das hier versteckt ist:

Challenge 3: "Catch me...if you can" (Schwierigkeitsgrad: leicht-mittel)
https://labs.inf.fh-dortmund.de/ctfd-challenge-1/catchme/catch-me.html

Wenn Sie das Flag gefunden haben, schicken Sie es per E-Mail an sven.joerges@fh-dortmund.de (gerne inklusive Lösungsweg, den Sie gewählt haben). Und nichts verraten! :-)

Dies ist die letzte im Rahmen der Veranstaltung veröffentlichte CTF-Aufgabe. Falls Sie Lust auf mehr (und schwierigere!) Challenges haben, dann schauen Sie doch mal hier vorbei (CTF-Server des FB4):


CTF
WE
BTE
CH