Praktikum zu Web-Technologien/Web-Engineering, Wintersemester 2023/2024

Praktikum 9 (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 18.01.24 (Ende der Vorlesungszeit)!

Hinweis zum entstehenden Code:

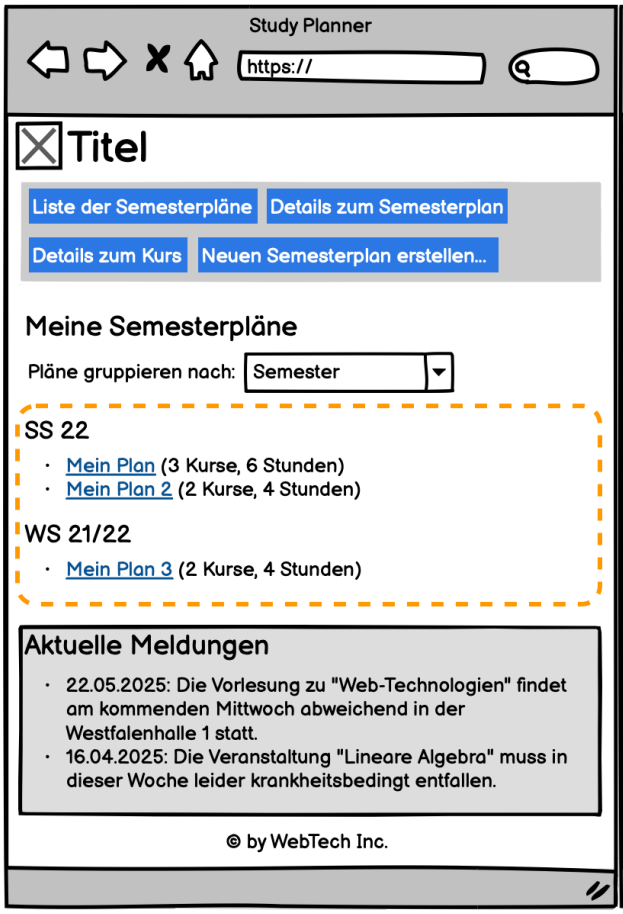
- Zur Verwaltung und zum kollaborativen Bearbeiten Ihres Quellcodes empfehlen wir Ihnen die Nutzung von Git (z.B. in Verbindung mit dem GitLab-Server des FB4, Login per FH-Account). Eine kurze Einführung in Git finden Sie hier.
- Falls Sie Git nicht verwenden, so legen Sie Ihren Quellcode pro Praktikumsaufgabe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

Aufgabe 1: Dynamisches Umschalten der Ansicht

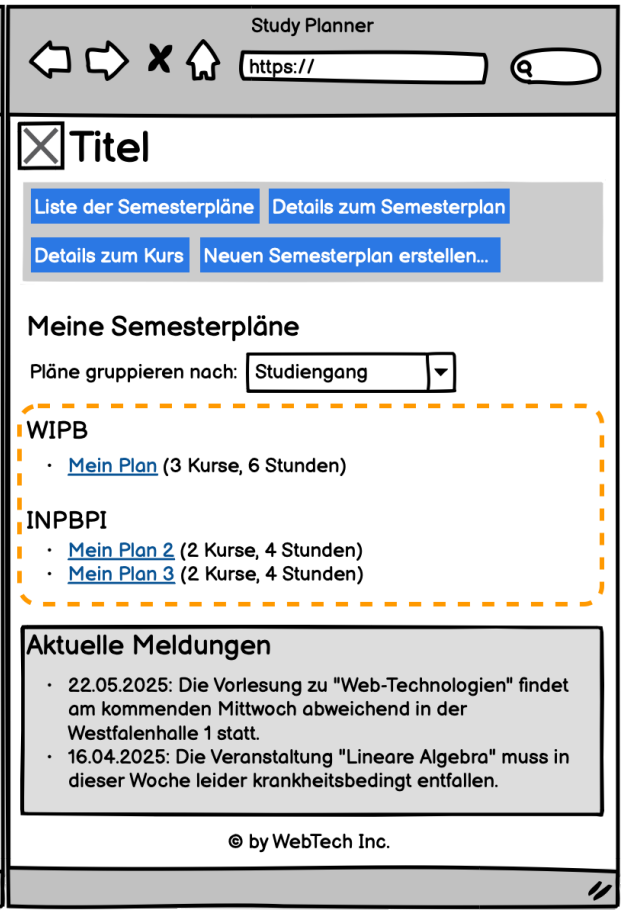
Bisher haben wir die Seite *Liste der Semesterpläne* (`index.html`) so strukturiert, dass alle Semesterpläne gruppiert nach Semestern angezeigt werden. Erweitern Sie die Seite nun so, dass die Semesterpläne *wahlweise gruppiert nach Semestern oder nach Studiengängen* angezeigt werden können.

Die Ansicht soll dabei mit Hilfe einer Dropdown-Box dynamisch umschaltbar sein, wie in folgenden Mockups dargestellt:

1. Gruppieren nach Semester:



2. Gruppieren nach Studiengang:



Nutzen Sie die Möglichkeiten der *DOM-Manipulation* und des *Event-Handlings* mit JavaScript zur Umsetzung. Gehen Sie dabei wie folgt vor:

1. Lagern Sie den Code zum Umschalten der Ansicht in einer separaten Datei `assets/js/index.js` und binden Sie diese geeignet ein.

2. Ergänzen Sie die Beispieldaten in `assets/js/script.js` (→ Praktikum 8, Aufgabe 2.1) um zwei weitere `Semesterplan`-Objekte (es sollten also insgesamt drei `Semesterplan`-Objekte existieren). Vergeben Sie jeweils sinnvolle Werte für die Eigenschaften `semester` und `studiengang`, damit Sie die Gruppierung gut testen können. Verwalten Sie die drei Objekte in einem Array.

3. Ergänzen Sie die Seite um eine Dropdown-Box mit den Auswahlmöglichkeiten "Semester" und "Studiengang". Bei Auswahl eines Wertes in der Dropdown-Box soll sich die Ansicht automatisch anpassen wie in den obigen Mockups darstellt (d.h.

gruppiert nach Semester oder nach Studiengang).

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.

4. Die auf der Seite angezeigten Daten zu den Semesterplänen sollen nun nicht mehr im HTML-Code fest vorgegeben sein. Stattdessen soll die Ansicht dynamisch aufgebaut werden und die `Semesterplan`-Objekte anzeigen, die Sie in `assets/js/script.js` erstellt haben. Die Namen der Semesterpläne sollen dabei weiterhin Hyperlinks sein.

   Wichtig: In den oben stehenden Mockups sind die entsprechenden Bereiche der Ansicht mit einem orange gestrichelten Kasten markiert. Nur diese Bereiche sollen dynamisch per JavaScript erzeugt werden. Alle weiteren Bereiche der Seite können wie bisher statisch im HTML-Code beschrieben werden.

5. Verwenden Sie die Funktionen `getAnzahlKurse` und `getAnzahlStunden` aus Praktikum 8, um auch die Anzahl der Kurse und Stunden für jeden Semesterplan dynamisch anzuzeigen.

6. Beim erstmaligen Öffnen der Seite sollen die Semesterpläne standardmäßig nach Semester gruppiert sein.

Hinweis zum Gruppieren von Werten in einem Array:

Nutzen Sie die folgende Funktion, um die Inhalte eines Arrays nach einer bestimmten Eigenschaft zu gruppieren:

```
const gruppiereNach = (array, eigenschaft) =>
  array.reduce((ergebnis, element) => {
    if (!ergebnis[element[eigenschaft]]) {
      ergebnis[element[eigenschaft]] = [];
    }
    ergebnis[element[eigenschaft]].push(element);
    return ergebnis;
  }, {});
```

Die Funktion können Sie z.B. auf folgende Weise verwenden:

```
// Gruppieren nach Studiengang:
// plaene ist ein Array, welches die Semesterplan-Objekte enthält
let gruppiertePlaene = gruppiereNach(plaene, "studiengang");
```

`gruppiertePlaene` ist dann ein Objekt mit folgender Struktur:

```
{
  "WIPB": [
    {
      "id": 0,
      "name": "Mein Plan",
      "semester": "SS 22",
      "studiengang": "WIPB",
      "kurse": [...]
    }
  ],
  "INPBPI": [
    {
      "id": 1,
      "name": "Mein Plan 2",
      "semester": "SS 22",
      "studiengang": "INPBPI",
      "kurse": [...]
    },
    {
      "id": 2,
      "name": "Mein Plan 3",
      "semester": "WS 21/22",
      "studiengang": "INPBPI",
      "kurse": [ ... ]
    }
  ]
}
```

Die Inhalte dieses Objektes erhalten/traversieren Sie z.B. so:

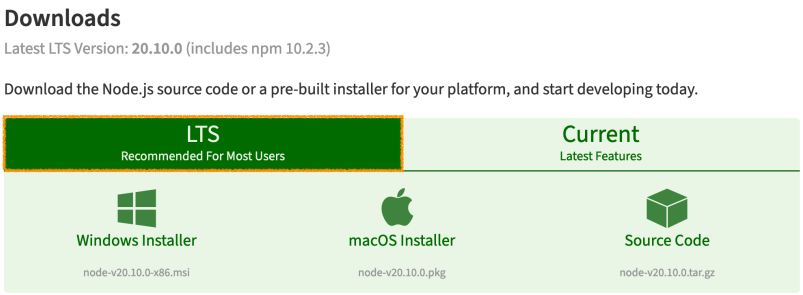
```
// Über alle Gruppen iterieren. ACHTUNG: "in" statt "of" in der Schleife
// verwenden, da wir über Eigenschaften eines Objektes iterieren.
for (let gruppe in gruppiertePlaene) {
    // [...]

    // Über alle Pläne einer Gruppe iterieren. Diese Pläne stecken in ei
    // Array, auf welches wir über die Indexnotation für Objekte zugreif
    // (eckige Klammern).
    for (let plan of gruppiertePlaene[gruppe]) {
      // [...]
    }
  }
```

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. Auf den Rechnern im Labor ist Node.js bereits vorinstalliert. Falls Sie also mit einem Laborrechner arbeiten, so können Sie direkt mit Schritt 4 fortfahren.
2. Falls Sie mit einem eigenen Rechner arbeiten, so laden Sie den für Ihr Betriebssystem passenden Node.js-Installer hier herunter: https://nodejs.org/en/download/

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

3. 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. "v20.10.0").

4. Speichern Sie das Programm aus Aufgabe 1 (JS CODING KATA) von Übungsblatt 7 in eine Datei namens `tictac.js` in Ihr Praktikumsverzeichnis (lösen Sie die Aufgabe ggf. zuerst, falls noch nicht geschehen).

5. Wechseln Sie nun auf der Konsole (siehe Schritt 3, Punkt 1) in Ihr Praktikumsverzeichnis (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.
- 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):