Modulprüfung „Web-Technologien“: Aufgabenstellung

Rahmenbedingungen und Ablauf

a. Bearbeitung, Hilfsmittel, Bewertung

- Die Bearbeitung der Aufgabenstellung erfolgt in Einzelarbeit am Chromebook. Zur Bearbeitung wird die vorinstallierte Cloudumgebung Coder mit VS Code und dem Chrome-Browser verwendet.
- Folgende **Hilfsmittel** dürfen verwendet werden:
    1. Bei der Bearbeitung der Prüfung darf ein **beidseitig handbeschriebenes DIN A4-Blatt im Original** als Hilfsmittel verwendet werden.
    2. Eingebaute Funktionen von VS Code wie z.B. Code-Vervollständigung oder Formatierung sowie von die Entwicklertools von Chrome dürfen verwendet werden.
    3. Zudem ist zur Recherche der Zugriff auf die MDN Web Docs (https://developer.mozilla.org) freigeschaltet.
- Zum Bestehen der Prüfung sind 50 Punkte erforderlich. Insgesamt sind 113 Punkte (100 Punkte + 13 Bonuspunkte) erreichbar.
- Die Bearbeitungszeit beträgt 120 Minuten.

b. Abgabe

- Speichern Sie Ihre Lösungen zur Abgabe im vorbereiteten VS Code-Projekt ab. **Tipp:** Speichern Sie auch während der Bearbeitung regelmäßig, um Datenverlust zu vermeiden.
- Füllen Sie unbedingt das Formular „Ihre Anmeldedaten zur Prüfung“ aus und geben Sie das Formular und diese Aufgabenstellung vor Verlassen des Prüfungsraumes ab!

c. Täuschungsversuch, Plagiat

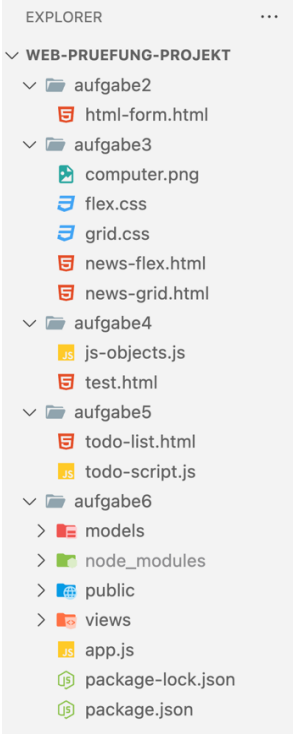
- **Belehrung zu Täuschungsversuchen:** Bei einem Täuschungsversuch wird die betreffende Prüfungsleistung mit „nicht ausreichend“ (5,0) bewertet. Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling von der Erbringung weiterer Prüfungs- und Studienleistungen ausgeschlossen und zudem exmatrikuliert werden (§ 11 Abs. 3 RahmenPO, § 63 Abs. 5 HG).
- **Plagiate** werten wir grundsätzlich als Täuschungsversuch (siehe „Belehrung zu Täuschungsversuchen“). Dies umfasst Plagiate von anderen Studierenden ebenso wie Plagiate durch Kopieren externer Quellen (z.B. kopierter Quellcode von Web-Quellen wie StackOverflow, ChatGPT etc.). Die Prüfer behalten sich vor, Abgaben mit Hilfe von Software-Werkzeugen automatisiert auf solche Plagiate zu überprüfen.

Allgemeines: Wissensaufgabe und Programmieraufgaben

Bei Aufgabe 1 handelt es sich um eine *Wissensaufgabe*, die Sie mit Stift und Papier lösen müssen.

Alle weiteren Aufgaben (2-6) sind *Programmieraufgaben*, welche Sie in einem vorbereiteten VS Code-Projekt am Rechner bearbeiten. Führen Sie zum Öffnen des Projektes die Schritte aus, welche auf dem Zettel „Ihre Anmeldedaten zur Prüfung“ beschrieben sind.

Nach Durchführen dieser Schritte sollte das Projekt wie in der untenstehenden Abbildung aussehen. Für jede der folgenden Aufgaben finden Sie im Projekt ein separates Unterverzeichnis, d.h. Aufgabe 2 → Verzeichnis „aufgabe2“, Aufgabe 3 → Verzeichnis „aufgabe3“ etc.



Allgemeines: Hilfreiche Tastenkürzel

Tool	**Aktion**	**Tastenkürzel**
VS Code	Code formatieren	Strg + Shift + I
VS Code	Neues Terminal öffnen	Strg + Shift + C
VS Code	Command Palette	Strg + Shift + P
VS Code	Laufende Server anzeigen (View „Ports“)	Burger-Menü links oben → View → Open View… → „Ports“ eingeben und mit Return bestätigen
Chrome	Devtools öffnen	Strg + Shift + I

Aufgabe 1: Wissensfragen [10 Punkte]

Kreuzen Sie in den folgenden Tabellen an, welche Aussagen richtig und welche falsch sind. Für jede korrekt markierte Aussage erhalten Sie 1 Punkt.

Thema: Internet und WWW	**Richtig**	**Falsch**
HTML ist im TCP/IP-Modell ein Protokoll der Anwendungsschicht.		
127.0.0.1 ist eine gültige IPv4-Adresse.		

Thema: HTML	**Richtig**	**Falsch**
`<input>` ist ein Beispiel für ein *standalone tag*.		
Block-Elemente nehmen nur so viel horizontalen Platz ein, wie sie benötigen.		
`<a href="../seite.html">Link</a>` ist ein Beispiel für einen relativen Verweis.		

Thema: CSS	**Richtig**	**Falsch**
Über `!important` ist es möglich, einen Inline-Style zu überschreiben.		
`position: static` entspricht dem normalen Elementfluss.		
Breakpoints sind ein Mittel, um die Eigenschaft „fluide“ eines responsiven Layouts zu realisieren.		

Thema: JavaScript	**Richtig**	**Falsch**
`Boolean(42/"Pommes")` ergibt `true`.		
Im Browser landen mit `var` deklarierte globale Variablen im `window`-Objekt.		

Aufgabe 2: HTML-Formulare [17 Punkte]

Im Verzeichnis „aufgabe2“ finden Sie die HTML-Datei `html-form.html`. Ergänzen Sie diese Datei um HTML-Code, der das in Abbildung 1 dargestellte Formular erzeugt. Das Formular soll folgende Eigenschaften besitzen:

1. Im Eingabefeld „Nickname“ sollen nur Eingaben der folgenden Form erlaubt sein: Genau fünf Kleinbuchstaben, gefolgt von mindestens einer bis maximal drei Ziffern (Beispiele: „tydur001“, „brack3“, „schlo23“).
2. Beim Eingabefeld „Passwort“ soll es sich um ein Passwortfeld handeln, welches nur Eingaben mit einer Mindestlänge von 10 Zeichen erlaubt.
3. Der Klick auf den Button „Registrierung“ soll zum Verschicken einer HTTP-Anfrage der folgenden Form führen:

   Header:
   ```
   POST /registration HTTP /1.1
   ```

   Body (mit Beispieleingaben):
   ```
   name=tydur001&passwort=geheim1234&datenschutz=zugestimmt
   ```

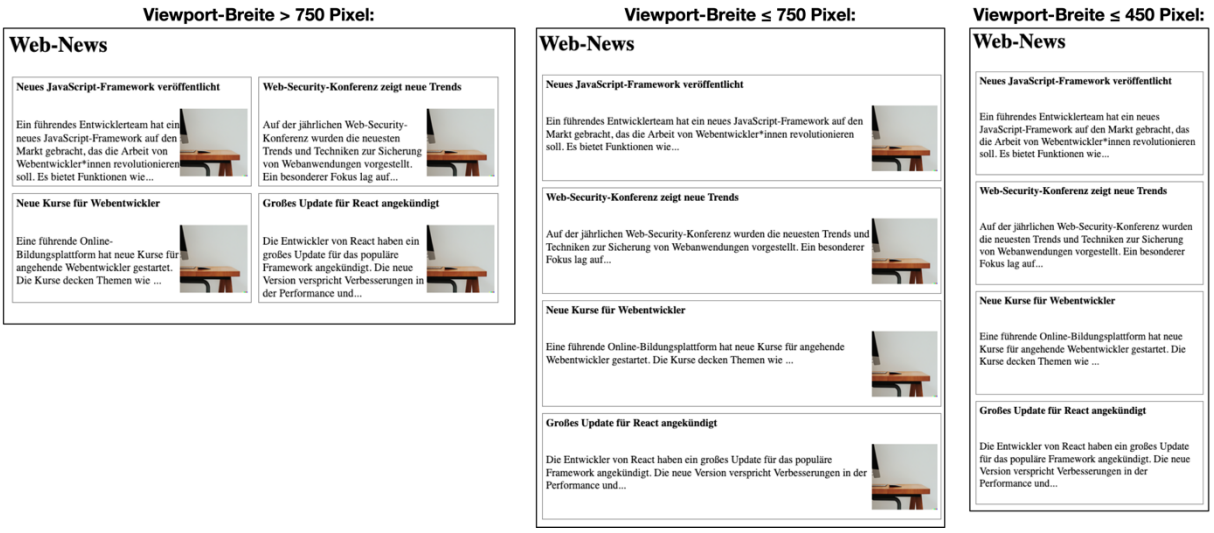
4. Verwenden Sie `label` und `fieldset` zur Strukturierung des Formulars.

Abbildung 1: Darstellung des HTML-Formulars im Browser

Aufgabe 3: Layouting [13 Punkte]

Im Verzeichnis „aufgabe3“ finden Sie die HTML-Datei `news-flex.html` sowie das externe Stylesheet `flex.css`. Die bestehende Implementierung nutzt *Flexbox* und *Media Queries*, um das folgende responsive Layout zu erzeugen (Abbildung 2):

Abbildung 2: Layout



Bauen Sie das bestehende Layout so um, dass statt *Flexbox* nur noch *Grid* verwendet wird. Dafür finden Sie im Verzeichnis „aufgabe3“ die vorbereiteten Dateien `news-grid.html` und `grid.css`. Ergänzen Sie zur Lösung der Aufgabe die Datei `grid.css`.

Das Grid-basierte Layout soll so aussehen und funktionieren wie die Flexbox-basierte Version (siehe Abbildung 2).

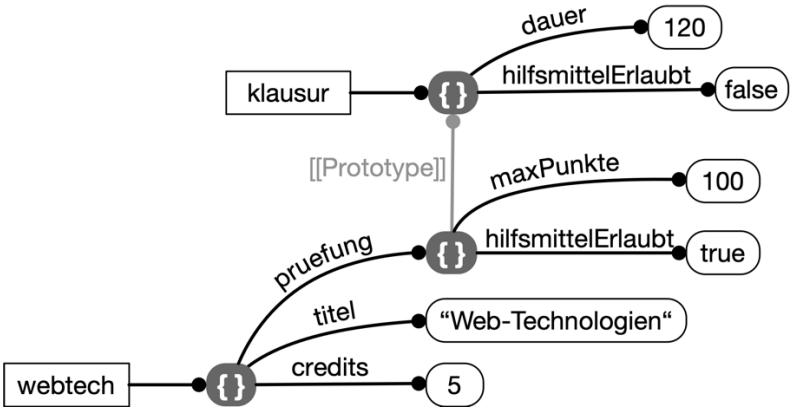
Bitte beachten:

- Verwenden Sie *ausschließlich* Grid zur Lösung der Aufgabe.
- Lösen Sie die Aufgabe *ohne* Verwendung von `!important`.
- Die Dateien `news-grid.html`, `news-flex.html` und `flex.css` dürfen nicht verändert werden.

Aufgabe 4: Objekte in JavaScript [10 Punkte]

Abbildung 3 zeigt Variablen und Objekte in der aus der Vorlesung bekannten "Kabeldarstellung". Ergänzen Sie in der Datei `js-objects.js` (Verzeichnis „aufgabe4“) mit Hilfe der *Objektliteralnotation* und weiteren Funktionen von JavaScript den Code, der diese Variablen und Objekte erzeugt.

Abbildung 3: JavaScript-Objekte in Kabelnotation



Tipp: Die Datei `test.html` im Verzeichnis „aufgabe4“ können Sie bei Bedarf nutzen, um sich die Objekte zur Kontrolle in der Browser-Konsole ausgeben zu lassen.

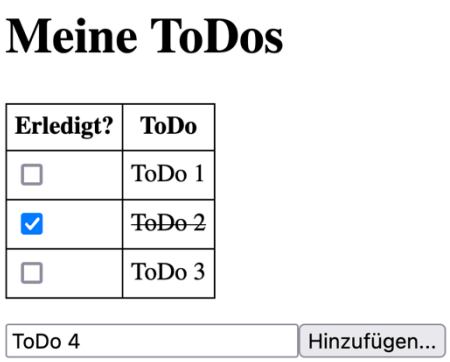
Aufgabe 5: DOM-Manipulation [25 Punkte]

Im Verzeichnis „aufgabe5“ finden Sie die Dateien `todo-list.html` und `todo-script.js`, welche eine einfache ToDo-Liste implementieren sollen.

1. Ergänzen Sie die Datei `todo-script.js`, so dass bei einem Klick auf den Button mit der Aufschrift „Hinzufügen…“ die Eingabe im `input`-Element gelesen und als Eintrag in der Tabelle hinzugefügt wird. Eine Tabellenzeile besteht dabei aus einer Checkbox (Spalte „Erledigt?“) und dem Text des ToDos (Spalte „ToDo“), wie in Abbildung 4 dargestellt.

2. Realisieren Sie die Checkboxen in der Spalte „Erledigt?“, so dass man damit einzelne ToDos als „erledigt“ markieren kann. Dazu soll bei einem Klick auf eine Checkbox der Text des ToDos durchgestrichen angezeigt werden (siehe „ToDo 2“ in Abbildung 4). Eine solche Darstellung erreichen Sie, indem Sie die CSS-Eigenschaft `text-decoration` auf den Wert `line-through` setzen. Bei erneutem Klick auf die Checkbox soll der Text wieder normal angezeigt werden (`text-decoration: none`).

Abbildung 4: Zielzustand (Beispiel)



Bitte beachten:

- Verwenden Sie lediglich die native DOM-API des Browsers zur Lösung der Aufgabe, d.h. keine Bibliotheken wie z.B. JQuery.
- **Lösen Sie die Aufgabe *ohne* Verwendung von `innerHTML` bzw. `insertAdjacentHTML`.**
- Der in den Dateien `todo-list.html` und `todo-script.js` bereits vorhandene Code darf nicht verändert werden!

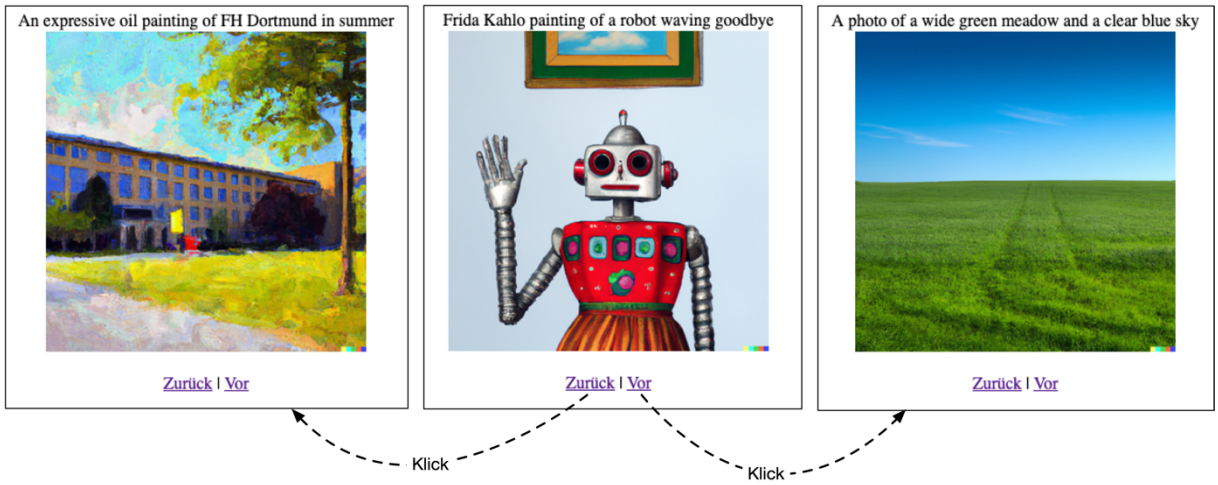
Aufgabe 6: Express [25 Punkte]

Realisieren Sie mit Hilfe von Express eine Web-Anwendung zur Anzeige von Bildern. Diese Anwendung soll wie folgt aussehen:

Abbildung 5: Aussehen der Web-Anwendung

Das aktuelle Bild wird als *Abbildung* angezeigt. Über dem Bild befindet sich eine *Bildbeschriftung*. Unter dem Bild befinden sich zwei *Hyperlinks*, mit denen das vorherige bzw. das nächste Bild angezeigt werden kann, wie in folgender Abbildung dargestellt:

Abbildung 6: Funktionsweise der Hyperlinks



Im Verzeichnis „aufgabe6“ ist eine geeignete Modularisierung der Web-Anwendung schon in Form verschiedener Dateien vorbereitet. Ergänzen Sie diese Dateien nun folgendermaßen:

1. `aufgabe6/app.js`:
   - Die Web-Anwendung soll über die Datei `app.js` gestartet werden können.

- Realisieren Sie in der Datei passende *Routen*, um die in Abbildung 6 dargestellte Funktionsweise der Hyperlinks zu unterstützen.

2. `aufgabe6/models/bilder.js`:
   - Bei dieser Datei handelt es sich um ein *Modul*, welches die Datenhaltung für die Anwendung simuliert. In dieser Datei ist das Array `bilder` vorbereitet, welches die Daten für die Anwendung enthält. Das Array definiert die bekannten Bilder und in welcher Reihenfolge sie angezeigt werden.
   - Realisieren Sie in dem Modul folgende Funktionen:
     1. `holeStartBild()`: Gibt das erste Bild aus dem Array `bilder` zurück (Index 0).
     2. `vor()`: Gibt das nächste Bild aus dem Array `bilder` zurück (aktueller Index + 1). Ist das Ende des Arrays erreicht, so soll wieder bei Index 0 angefangen werden.
     3. `zurueck()`: Gibt das vorherige Bild aus dem Array `bilder` zurück (aktueller Index - 1). Ist der Anfang des Arrays erreicht, so soll zum Index `bilder.length-1` gesprungen werden.
   - Definieren Sie die Schnittstelle des Moduls in einer Form, so dass andere Module auf die Funktionen `holeStartBild()`, `vor()` und `zurueck()` zugreifen können. Andere Module sollen nicht direkt auf das Array `bilder` zugreifen können.

3. `aufgabe6/views/home.ejs`:
   - Ergänzen Sie das EJS-Template, so dass die HTML-Seite gemäß Abbildung 5 dargestellt wird.
   - Die Hyperlinks sollen funktionieren wie in Abbildung 6 dargestellt.

Bitte beachten:

- Die Bilder zur Anzeige finden Sie im Verzeichnis „aufgabe6/public/img“.
- **Auf keinen Fall das Verzeichnis `node_modules` leeren oder löschen!** Da Sie nicht mit dem Internet verbunden sind, können Sie sich die Bibliotheken nicht erneut herunterladen.