Web-Anwendungen, **SS17 - Fragentypen**

Hinweis: Dieses Dokument ist keine Klausur, sondern eine lose (und nicht notwendigerweise vollständige) Sammlung an Fragen wie sie auch in einer Klausur vorkommen könnten.

- Punktzahlen werden in Kästen notiert und einmal als Summe je Aufgabe und dann einzeln für jede Teilaufgabe angegeben. Im Rahmen dieser Nicht-Klausur haben die Punktzahlen allerdings keine tiefere Bedeutung.
- Diese Nicht-Klausur besteht, inklusive dem Deckblatt aus **11 Seiten** and **10 Fragen**. Es können maximal **57** **Punkte** erreicht werden.
- Alle Anhänge zur Nicht-Klausur werden separat ausgeteilt. Die Heftung der Anhänge darf gelöst werden, Notizen auf dem Anhang werden jedoch keinesfalls bewertet. Sie sollte zu keinem Zeitpunkt zwischen mehreren Seiten blättern müssen, um sich vorgegebene `HTML`- oder `CSS`-Dokumente anzusehen.
- Die folgenden Themengebiete sind explizit ausgegrenzt:
  - Semantische Auszeichnungen mit `schema.org` oder Open Graph. Die allgemein semantisch sinnvollen `HTML`-Elemente sind hingegen relevant.
  - Zeitabhängige Animationen mit CSS
  - 3D-Transformationen mit CSS
  - Angular 2, nicht aber Prinzipien und Probleme von Single Page Applications

1) HTML

$\boxed{\sum 6}$

(a) Das in XYZ) zu sehende HTML-Dokument enthält vier syntaktische und zwei semantische Fehler. Geben Sie zwei der syntaktischen und einen semantischen Fehler an und erläutern Sie *stichwortartig* den Fehler. 3

(b) Das in XYZ) zu sehende HTML-Dokument wurde ausschließlich mit semantisch bedeutungslosen `<div>` und `<span>`-Elementen ausgezeichnet. Geben Sie für vier `<div>`-Elemente und zwei `<span>`-Elemente semantisch passende Elemente an. 3

2) Templating mit Liquid

$\boxed{\sum 3}$

(a) Zeichnen Sie den `DOM`-Baum für das sich aus den Daten ABC und dem Template XYZ ergebenden `HTML`-Dokument. $\boxed{3}$

> **Hinweis:** Sie können davon ausgehen, dass der Liquid-Code selbst syntaktisch korrekt sein wird und keinen Gebrauch von komplizierten Schleifen oder Berechnungen machen wird.

3) Layout und Box-Modell

$\sum 6$

(a) Wie breit ist ein Element mit `width: 10px;`, `border-left: 15px;` und `margin: 5px;`? [3]

(b) Skizzieren Sie das Layout für das `HTML`-Dokument XYZ. Machen Sie in Ihrer Skizze deutlich welche Elemente über die volle Seitenbreite gehen (Block-Elemente), sich an der rechten oder linken Seite orientieren (`float`) oder intelligent umgebrochen werden (`float`, flex). [3]

> **Hinweis:** Zur Lösung ist kein Zentimetermaß nötig, alle Breiten werden Vielfache von 20%, 25%, oder `calc(1/3)` sein.

4) CSS-Selektoren und DOM-Bäume $\boxed{\sum 9}$

(a) Berechnen Sie die Spezifitäten für die in XYZ angegebenen CSS-Selektoren. Geben Sie die einzelnen Summanden in einer Tabelle an. $\boxed{3}$

(b) Schreiben Sie einen CSS-Selektor, der exakt die in Abbildung XYZ und ABC markierten Knoten markiert. $\boxed{3}$

(c) Markieren Sie alle Knoten in Abbildung XYZ, die dem folgenden CSS-Selektor entsprechen. $\boxed{3}$

5) JavaScript allgemein

$\boxed{\sum 3}$

(a) Welche der folgenden Aussagen treffen auf das Programm zu? $\boxed{3}$

```
1  const f1 = (p1, p2) => {
2    for (v of p1) {
3      console.log(p2(v));
4    }
5  };
6
7  // Aufruf #1
8  f1([1,2,3], (v) => v + 1);
9  // Aufruf #2
10 f1("123", (v) => +v * 2);
11 // Aufruf #3
12 f1([1, "2", '3'], (v) => v + 1);
```

- ☐ Der Parameter `p1` der Funktion `f1` muss eine Callback-Funktion sein.
- ☐ Der Parameter `p2` der Funktion `f1` muss eine Callback-Funktion sein.
- ☐ Der Parameter `p1` der Funktion `f1` muss ein Array sein.
- ☐ Der Parameter `p2` der Funktion `f1` muss ein Array sein.
- ☐ Das Programm macht zu keinem Zeitpunkt eine Ausgabe.
- ☐ Das Ergebnis von Aufruf #1 ist 6.
- ☐ Das Ergebnis von Aufruf #1 ist 12.
- ☐ Die Laufvariable `v` nimmt in Aufruf #1 die Werte `1`, `2` und `3` an.
- ☐ Die Laufvariable `v` nimmt in Aufruf #2 die Werte `1`, `2` und `3` an.
- ☐ Die Laufvariable `v` nimmt in Aufruf #3 die Werte `1`, `2` und `3` an.
- ☐ Der Aufruf `f1([], undefined, 2)` ließe das Programm abstürzen.
- ☐ Der Aufruf `f1({}, (v) => v)` ließe das Programm abstürzen.

6) JavaScript allgemein $\boxed{\sum 6}$

(a) Welche der folgenden Datentypen von `p1` führen **nicht** zu einem Absturz? $\boxed{3}$

```
1  const f1 = (p1) => {
2    let l1 = 0;
3    for (v of p1) {
4      if (v !== +v) {
5        l1++;
6      }
7    }
8
9    return (l1);
10 };
11
12 const f2 = function*() {
13   yield (1);
14   yield ('2');
15   yield ({});
16 }
```

- ☐ Der Parameter `p1` der Funktion `f1` darf ein Funktions-Objekt sein.
- ☐ Der Parameter `p1` der Funktion `f1` darf Iterator-Objekt sein.
- ☐ Der Parameter `p1` der Funktion `f1` darf ein Daten-Objekt sein.
- ☐ Der Parameter `p1` der Funktion `f1` darf eine Zahl sein.
- ☐ Der Parameter `p1` der Funktion `f1` darf eine Liste sein.
- ☐ Der Parameter `p1` der Funktion `f1` darf eine String sein.

(b) Schreiben Sie drei Aufrufe von `f1`, deren Ergebnis jeweils 2 ist. Nutzen Sie dafür Jeden der von Ihnen angekreuzten Typen der vorigen Aufgabe exakt ein mal. $\boxed{3}$

7) JavaScript allgemein $\boxed{\sum 3}$

(a) Das folgende Programm ist syntaktisch korrekt, wird zur Laufzeit aber abstürzen. $\boxed{3}$

```
1  const MyType = function(name) {
2    this.name = name;
3
4    MyType.count++;
5  }
6
7  MyType.prototype.greet = function() {
8    console.log(`Hello ${this.name}`);
9  }
10
11 MyType.count = 0;
12
13 let p1 = new MyType('Hans');
14 let p2 = new MyType('Franz');
15 let p3 = MyType('Dampf');
16
17 p1.greet();
18 p2.greet();
19 p3.greet();
20
21 console.log(`There are ${MyType.count} instances`);
```

- ☐ Das Programm stürzt in Zeile 2 ab, weil `this.name` undefiniert ist.
- ☐ Das Programm stürzt in Zeile 4 ab, weil `MyType.count` undefiniert ist.
- ☐ Das Programm stürzt in Zeile 8 ab, weil `this.name` undefiniert ist.
- ☐ Das Programm stürzt in Zeile 11 ab, weil `MyType.count` undefiniert ist.
- ☐ Das Programm stürzt in Zeile 15 ab, weil `MyType` ohne `new` aufgerufen wird.
- ☐ Das Programm stürzt in Zeile 17 ab, weil `p1.greet` undefiniert ist.
- ☐ Das Programm stürzt in Zeile 18 ab, weil `p2.greet` undefiniert ist.
- ☐ Das Programm stürzt in Zeile 19 ab, weil `p3.greet` undefiniert ist.
- ☐ Das Programm stürzt in Zeile 21 ab, weil `MyType.count` undefiniert ist.
- ☐ Der Zähler `MyType.count` steht zum Zeitpunkt des Absturzes auf 0.
- ☐ Der Zähler `MyType.count` steht zum Zeitpunkt des Absturzes auf 1.
- ☐ Der Zähler `MyType.count` steht zum Zeitpunkt des Absturzes auf 2.
- ☐ Der Zähler `MyType.count` steht zum Zeitpunkt des Absturzes auf 3.
- ☐ Der Zähler `MyType.count` steht zum Zeitpunkt des Absturzes auf 4.

8) JavaScript auf dem Server $\boxed{\sum 6}$

(a) Welche Middleware-Funktionen aus JavaScript-Programm XYZ werden für eine Anfrage an den Pfad `/person/1/diary` aufgerufen? [3]

(b) Welche der folgenden Aussagen treffen auf den Programm-Ausschnitt zu? [3]

```
1  const express = require('express');
2  const app = express();
3  let data = 0;
4
5  // Middleware #1
6  app.use((req, res, next) => {
7    console.log(`Requested: ${req.url}`);
8    next();
9  });
10
11 // Middleware #2
12 app.get('/', (req, res) => {
13   res.send(200, data);
14 });
15
16 // Middleware #3
17 app.put('/', (req, res) => {
18   data += 1
19   res.send(200, data);
20 });
21
22 // Middleware #4
23 app.delete('/', (req, res) => {
24   data = 0;
25 });
```

☐ Es gibt Anfragen keine Anfrage, die nicht durch Middleware #1 läuft.

☐ Jede Anfrage wird mit Statuscode `200` beantwortet.

☐ Anfrage #2 kann nur über ein Formular ausgelöst werden.

☐ Der Zustand des Programms wird nach einem Neustart des Servers beibehalten.

9) XML

$\boxed{\sum 6}$

(a) Welche der folgenden XML-Dokumente werden durch das gegebene XML-Schema validiert? 3

(b) Schreiben Sie ein XML-Schema, welches die gegebenen XML-Dokumente validiert. 3

10) Ankreuzfragen $\boxed{\sum 9}$

Diese Fragen beziehen sich nicht durchgehend auf ein spezielles Themengebiet. Das grundlegende Schema ist immer identisch: Sie müssen zunächst eine klar definierte Anzahl von Kreuzen setzen und Ihre Antwort dann häufig kurz begründen.

(a) i. Welche der folgenden Aussagen über die Beziehung zwischen `JSON` und JavaScript ist wahr (1 Kreuz)? $\boxed{1}$

- ☐ Jedes valide JavaScript-Programm ist ein valides `JSON`-Dokument.
- ☐ Jedes in validem JavaScript notierte `JSON`-Objekt ist ein valides `JSON`-Dokument.
- ☐ Jedes valide `JSON`-Dokument ist ein valides Javascript-Programm

ii. Begründen Sie ihre Antwort mit kurzen Gegenbeispielen zu den beiden falschen Aussagen. $\boxed{2}$

(b) i. Welche der folgenden Aussagen über die Beziehung zwischen `HTML` und dem `DOM`-Baum ist wahr (1 Kreuz)? $\boxed{1}$

- ☐ Der sich aus einem `HTML`-Dokument ergebende `DOM`-Baum ist eindeutig.
- ☐ Das sich aus einem `DOM`-Baum ergebende `HTML`-Dokument ist eindeutig.
- ☐ Das `DOM`-Baum kann nach dem Laden nicht mehr modifiziert werden.

ii. Begründen Sie ihre Antwort mit kurzen Erläuterungen zu den falschen Aussagen. $\boxed{2}$

(c) i. Gegeben ist die folgende `URL`: `http://web.fh-wedel.de/mitarbeiter?name=harms`. Welche der folgenden Aussagen sind wahr (4 Kreuze)? $\boxed{2}$

- ☐ Es handelt sich um eine absolute `URL`.
- ☐ Es handelt sich um eine relative `URL`.
- ☐ Die `URL` enthält eine Port-Angabe.
- ☐ Die `URL` enthält einen Benutzernamen.
- ☐ Die `URL` enthält ein Passwort.
- ☐ Die `URL` enthält eine Pfad-Angabe.
- ☐ Die `URL` enthält Abfrage-Parameter.
- ☐ Die `URL` enthält ein Fragment.
- ☐ Die `URL` enthält lediglich erlaubte Zeichen und Bedarf keiner speziellen Kodierung.

ii. Nennen Sie ein beliebiges Zeichen, dass als Bestandteile des `name`-Parameters speziell kodiert werden müsste. $\boxed{1}$