Web-Anwendungen, SS17 - Probeklausur

- Gestellt von: Marcus Riemer (**mri**)
- Erlaubte Hilfsmittel: Zeichengeräte, Taschenrechner.
- Dauer: 60 Minuten.
- Diese Klausur besteht inklusive dieses Deckblattes und den Anhängen aus **5 Fragen** auf **12 Seiten**. Es können maximal **40** **Punkte** erreicht werden. Punktzahlen werden in Kästen notiert und einmal als Summe (Σ) je Aufgabe und dann einzeln für jede Teilaufgabe angegeben.
- Sie sollten zu keinem Zeitpunkt zwischen mehreren Seiten blättern müssen um sich vorgegebene Quelltexte anzusehen. Daher werden alle **Quelltexte als Anhänge** zur Klausur separat ausgeteilt. Die Heftung der Anhänge darf **nicht** gelöst werden, Notizen auf dem Anhang werden keinesfalls bewertet.
- Verwenden Sie zur **Lösung** die **freigelassenen Räume auf den Aufgabenblättern** (diese können, müssen aber nicht komplett gefüllt werden). Sollte der Platz nicht ausreichen, verwenden Sie die Rückseite der Aufgabenblätter.
- Der Inhalt der **Rückseiten** wird nur **gewertet**, wenn er eindeutig als Lösung gekennzeichnet ist (Stichwort "Lösung" mit Angabe der Aufgabennummer) und vom vorgesehenen Lösungsfeld auf die zu wertende Rückseite verwiesen wird.

Hinweise zum "Probe"-Charakter

- Diese Klausur ist zufällig aus einer Sammlung an möglichen Klausurfragen erzeugt worden, die echten Klausuren werden grundsätzlich aus dem gleichen Pool erzeugt. Jede der vorliegenden Fragen hätte theoretisch also auch in einer "richtigen" Klausur verwendet werden können.
- Diese Probeklausur enthält keine letzte Aufgabe mit Multiple-Choice-Fragen für ungefähr 10 Punkte. In einer "echten" Klausur können Sie dennoch davon ausgehen, dass Ihnen Multiple-Choice-Fragen gestellt werden. Da Ihnen diese Art von Fragen aber schon vertraut ist (und sich diese Aufgaben nicht wiederholen werden) fehlen Sie im Rahmen dieser Probeklausur.
- Eine Musterlösung wird nicht veröffentlicht. Sie können allerdings in der Klausursprechstunde Webanwendungen am Mo. 21.08.2017 um 16:00 in HS5 gerne Fragen zur Klausur stellen.

1) Templating mit Liquid und HTML

$\sum 12$

(a) Zeichnen Sie den DOM-Baum, welcher sich mit den vorliegenden Daten (Listing 1a) aus dem Template (Listing 1b) ergibt. Die Texte der Elemente können Sie in der Baumdarstellung ignorieren. [3]

(b) Dieser DOM-Baum steht alleine für sich genommen nicht für ein valides HTML-Dokument, wieso? [1]

(c) Das Template enthält (mindestens) drei Fehler, durch die inkorrektes `HTML` erzeugt werden könnte. Allerdings ist das Auftreten dieser Fehler abhängig von bestimmten Konstellationen in den zugrundeliegenden Daten. Nennen Sie zwei Datensätze durch die syntaktisch oder strukturell ungültiges `HTML` generiert wird, verweisen Sie auf die Ursache im Liquid-Template durch Zeilennummer(n) und beschreiben sie stichwortartig den Fehler. 8

2) CSS

$\sum 6$

Listing 2a zeigt einen Ausschnitt aus einem HTML-Dokument für welchen die CSS-Regeln aus Listing 2b zum Einsatz kommen.

(a) Nennen Sie die konkret zum Einsatz kommenden Deklarationen (also die `name: wert;`-Zuweisungen unterschiedlicher CSS-Eigenschaften) für die Elemente in den folgenden Zeilen:

i. Zeile 2 [1]

ii. Zeile 5 [1]

iii. Zeile 10 während der Mauszeiger über dem dortigen `<li>`-Element ruht [1]

iv. Zeile 11 während der Mauszeiger über dem dortigen `<li>`-Element ruht [1]

v. Zeile 16 [1]

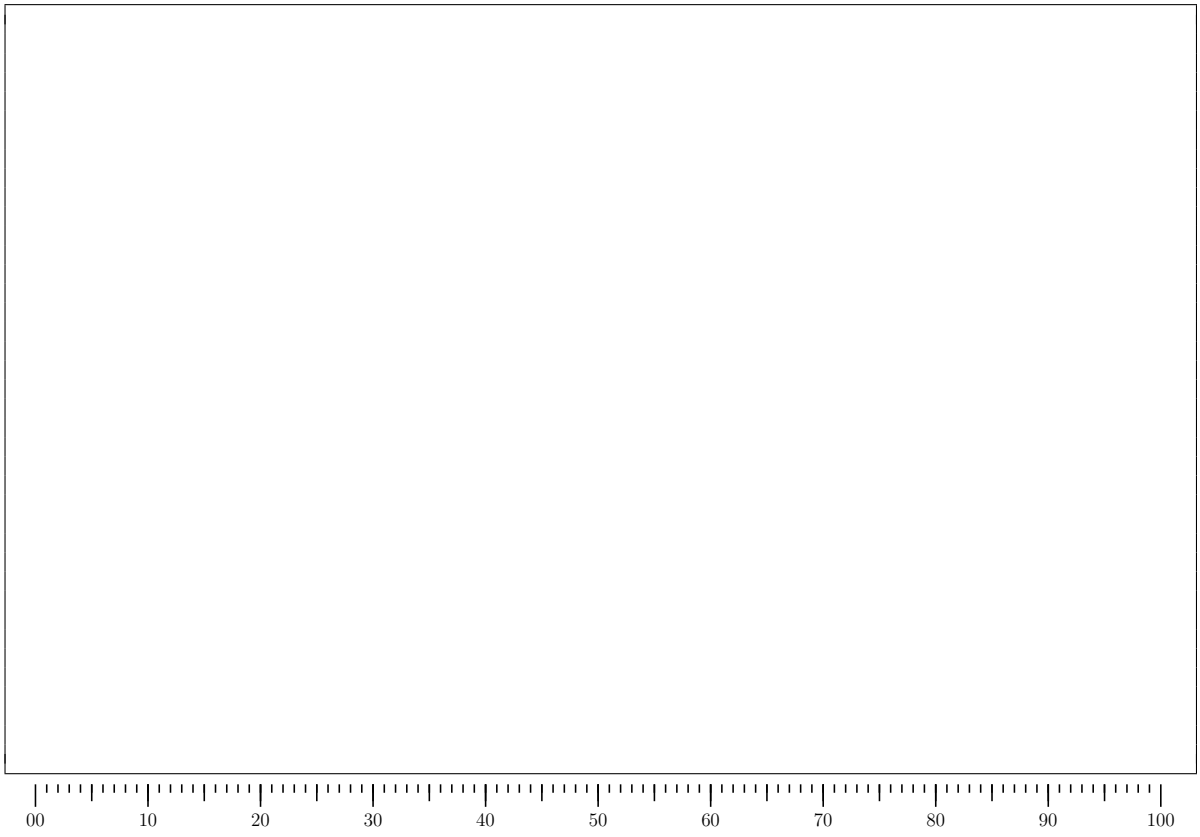
vi. Zeile 17 [1]

3) CSS

$\sum 9$

In Listing 3a sehen Sie ein HTML-Fragment welches mit dem CSS-Dokument aus Listing 3b in einem 100 Pixel breiten Viewport dargestellt wird.

(a) Skizzieren Sie die Darstellung in diesem Viewport. Auf etwaige Veränderungen der Größen oder Überläufe durch die Textelemente innerhalb der Elemente müssen Sie **keine** Rücksicht nehmen. Zur groben Orientierung sehen Sie unter dem entsprechenden Raum für die Lösung ein Lineal mit Pixelmaß. [7]



Vorsicht: Diese Skala verwendet **keine** Zentimeter!

(b) Nennen Sie einen CSS-Selektor der exakt das -Element in Zeile 3 von Listing 3b selektiert **ohne** dabei den ID-Selektor zu verwenden. [2]

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

In Listing 4a werden einige grundlegende Funktionsobjekte definiert. Alle im Folgenden zu sehenden Quelltexte setzen exakt dieses Listing (und keines der anderen) jeweils implizit fort.

(a) Was gibt das folgende Programm aus? $\boxed{2}$

```
1 const miezie = new Cat("Miezie");
2 const maunzi = new Cat("Maunzi", "rgba(0,0,0,1)");
3
4 miezie.greet();
5 maunzi.greet();
```

(b) Was gibt das folgende Programm aus? $\boxed{2}$

```
1 Cat.prototype.greet = function() {
2   console.log(`Maunzt: ${this.name}`)
3 }
4
5 const miezie = new Cat("Miezie", 'braun');
6 miezie.greet();
```

(c) Eine der folgenden Programmzeilen wird zu einem Absturz führen. Welche Zeile ist das und warum? $\boxed{2}$

```
1 const wuffi = new Dog(null, undefined);
2 wuffi.bark();
3 wuffi.greet();
```

5) XML

$\sum 7$

Die XML-Dokumente für diese Aufgabe (zu finden in Listing 5b, 5c und 5d) sind syntaktisch einwandfrei. Allerdings ist nur eines dieser XML-Dokumente valide im Sinne des Schemas in Listing 5a.

(a) Welches dieser Dokumente ist im Sinne der Schema-Definition in Listing 5a valide? [1]

☐ Dokument 1 ☐ Dokument 2 ☐ Dokument 3

(b) Benennen Sie je einen Fehler in den übrigen Dokumenten. Nehmen Sie dabei Bezug auf die Zeile in welcher der Fehler auftritt, die Zeile(n) im XML in denen die verletzte Restriktion genannt wird und begründen Sie stichwortartig die Art der Verletzung.

i. Verletzung in ☐ Dokument 1 ☐ Dokument 2 ☐ Dokument 3 [3]

ii. Verletzung in ☐ Dokument 1 ☐ Dokument 2 ☐ Dokument 3 [3]

Anhang: Aufgabe 1

```
{
  "page": {
    "ordered": true,
    "unordered": false,
    "items": [
      "Bratpfanne", "Schoepfkelle", "Pfannenwender",
      "Pfefferspray", "Kettenhandschuhe", "Langschwert"
    ]
  }
}
```

(a) Daten für Template

```
<div>
  <header>Willkommen</header>
  <h1>
    {{ page.name }}
  </h1>
  <article>
    {% if page.ordered %}
      <ol>
    {% endif %}
    {% if page.unordered %}
      <ul>
    {% endif %}
    {% for item in page.items %}
      <li>{{ item }}</li>
    {% endfor %}
    {% if page.ordered %}
      </ol>
    {% endif %}
    {% if page.unordered %}
      <ul>
    {% endif %}
  </article>
</div>
```

(b) HTML mit Liquid

Anhang: Aufgabe 2

```html
<h1>Hallo</h1>
<p>Lorem Ipsum</p>
<article>
  <h2>Welt</h2>
  <p>Lorem Ipsum</p>
</article>
<ul>
  <li>
    <ul>
      <li id="thefoo">Foo</li>
      <li id="thebar">Bar</li>
    </ul>
  </li>
  <li>
    <article>
      <h1 style="color: brown;"></h1>
      <p style="color: orange;"></p>
    </article>
  </li>
</ul>
```

(a) HTML-Fragment

```css
h1 p           { color: red; }
p              { color: blue; }
article *      { color: yellow; }
ul li:hover    { color: fuchsia; }
ul > article * { background-color: gray; }
#thefoo        { color: green; }
```

(b) CSS für Fragment

Anhang: Aufgabe 3

```html
<body>
  <h1>Head</h1>
  <div id="r1">1</div>
  <div id="l1">2</div>
  <div id="r2">3</div>
</body>
```

(a) HTML-Fragment

```css
h1, div {
    height: 20px;
    border-width: pen; /* Dicke ihres Zeichengerätes */
    border-color: pen; /* Farbe ihres Zeichengerätes */
}

#r1 {
    float: right;
    width: 90px;
}

#l1 {
    float: left;
    width: calc(25% - 10px);
}

#r2 {
    float: right;
    width: calc(50%);
}
```

(b) CSS für Fragment

Anhang: Aufgabe 4

```
1  const Animal = function(name) {
2    this.name = name;
3  }
4
5  Animal.prototype.greet = function() {
6    console.log(`Grüßt: ${this.name}`)
7  }
8
9  const Cat = function(name, colour) {
10   Animal.call(this, name);
11   this.colour = colour;
12 }
13
14 Cat.prototype = Object.create(Animal.prototype);
15
16 const Dog = function(name, colour) {
17   Animal.call(this, name);
18   this.colour = colour;
19 }
20
21 Dog.prototype.bark = function() {
22   console.log(`Bellt: ${this.name}`);
23 }
```

(a) JavaScript

Anhang: Aufgabe 5

```xml
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="party">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="char" minOccurs="1" maxOccurs="4">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="name" type="xs:string"/>
            </xs:sequence>
            <xs:attribute name="level" type="xs:integer"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

(a) XML-Schema

```xml
<party/>
```

(b) Dokument 1

```xml
<party>
  <char level="-1">
    <name></name>
  </char>
</party>
```

(c) Dokument 2

```xml
<party>
  <char level="1">
    <name>Femshep</name>
  </char>
  <char>
    <name>Wrex</name>
    <level>2</level>
  </char>
</party>
```

(d) Dokument 3