Klausur Grundlagen der Webentwicklung

PI/KI Bachelor

Prof. Dr. Kretschmer, WS 2020/2021

Rahmenbedingungen: 90 Minuten Bearbeitungszeit, keine Hilfsmittel. Schreiben Sie Ihre Lösungen in den dafür vorgesehenen Platz. Sollte dieser nicht ausreichen, können Sie auch die Rückseite verwenden. Vermerken Sie das dann bitte auf der Vorderseite. Insgesamt sind 78 Punkte erreichbar. Dies entspricht 100%. Mit 31 Punkten haben Sie bestanden.

Name	Vorname	Matrikelnr.	Studiengang

1	2	3	4	5	6	Summe	Note

Aufgabe 1

(32 Punkte) Beantworten Sie **kurz** folgende Fragen. Das Themengebiet steht jeweils in Klammern.

1.	Was ist ein Codeelement? (Unicode)
2.	Welche möglichen Längen (in Bytes) kann die Codierung eines Zeichens in UTF-8 haben? (Unicode)
3.	Wie lautet die Dokumenttyp-Deklaration für HTML5? (HTML)
4.	Was versteht man unter einem globalen Attribut? (HTML)
5.	Welche drei Elemente werden bei Definitionslisten verwendet? (HTML)
6.	Welches Attribut enthält beim a-Element die URL, auf die verwiesen wird? (HTML)
7.	Welche drei Begriffe kürzt HSL ab? (CSS)
8.	Was besagt die Längenangabe 50vh ? (CSS)
9.	Wozu dient ein Kombinator? (CSS)
10.	Aus welchen vier ineinander geschachtelten Boxen besteht eine Box? (CSS)
11.	Wofür steht die Abkürzung DOM und was ist das? (DOM)

12.	Was liefert die Methode `element.querySelectorAll(selectors)`? (DOM)
13.	Welchen Vorteil hat die Verwendung von `addEventListener` gegenüber der Zuweisung an eine Eigenschaft wie z.B. `onclick`? (DOM)
14.	Nennen Sie zwei der sechs Forderungen an eine REST-API, aber nicht „einheitliche Schnittstelle“ mit seinen Unterpunkten. (REST)
15.	Warum entspricht ein Endpunkt wie `/login` nicht den REST-Anforderungen? (REST)
16.	Was ist eine full-stack language? (allgemein)

Aufgabe 2

(8 Punkte) Das XML-Element `aufgabe` hat die optionalen Kindelemente `notiz` und `wiederholung`. Die Reihenfolge dieser beiden Elemente ist beliebig. Dahinter folgt kein, ein oder mehrere Elemente `zusatzkategorie`. Das obligatorische Attribut `id` enthält die eindeutige ID der Aufgabe. Das obligatorische Attribut `name` enthält den Namen der Aufgabe. Das Attribut `erledigt` enthält entweder `ja` oder `nein`. Dieses Attribut darf fehlen. Dann ist der Wert `nein`. Das optionale Attribut `vorher` enthält eine oder mehrere IDs von Aufgaben, die erledigt sein müssen, bevor diese Aufgabe erledigt werden kann. Die IDs sind durch Leerzeichen getrennt.

Geben Sie eine möglichst gut zutreffende (Teil-)DTD für das XML-Element `aufgabe` und seine Attribute an. Die Kindelemente müssen **nicht** definiert werden.

Aufgabe 3

(10 Punkte)

a) Geben Sie den Wert der vier Variablen `r1,r2,r3,r4` an.

```
let f = (g, x) => g(g(x))
let h = x => 2 * x
let r1 = f(h, 3)

let cities = ['Berlin', 'Hamburg', 'Trier', 'Saarlouis', 'Bamberg']

let r2 = cities.slice(2, 4).map(v => v.length)
let r3 = cities.filter(v => /i[d-r]/.test(v)).reduce((r, v) => r + v.length, 0)
cities.shift()
cities.pop()
let r4 = cities.map(v => v[0]).join('+')
```

r1=	r2=	r3=	r4=

b) Gegeben sei eine Variable `word`, die einen String enthält. Geben Sie einen einzeiligen JavaScript-Ausdruck an, dessen Wert die Anzahl des Kleinbuchstabens `e` in `word` ist.

Aufgabe 4

(8 Punkte) Gegeben sei das linksstehende `body`-Element einer HTML-Datei.

a) Geben Sie unter den zwei Selektoren auf der rechten Seite die `id`'s der durch die Selektoren selektierten Elemente an.

b) Geben Sie je einen Selektor an, der nur das Element mit der `id` 9 bzw. 14 selektiert und nichts anderes. Dabei dürfen Sie nicht auf das Attribut `id` zugreifen.

```
<body id="1">
  <p id="2">Brettspiele</p>
  <ul id="3">
    <li id="4">
      <p id="5">Schach</p>
      <ul id="6">
        <li id="7">Schnellschach</li>
        <li id="8">
          <p id="9">Blitzschach</p>
          <ul id="10">
            <li id="11">3 Minuten</li>
            <li id="12">5 Minuten</li>
          </ul>
        </li>
        <li id="13">Blindschach</li>
      </ul>
    </li>
    <li id="14">Go</li>
    <li id="15">Gobang</li>
    <li id="16">
      <p id="17">Dame</p>
      <ol id="18">
        <li id="19">normale Dame</li>
        <li id="20">Laskerdame</li>
      </ol>
    </li>
  </ul>
</body>
```

a)

`body * p`

`ul>li:nth-child(3n)`

b)

Selektor für 9:

Selektor für 14:

Aufgabe 5

(10 Punkte)

Gegeben sei ein Feld mit Informationen zu Links, dessen Aufbau z.B. wie folgt aussieht:

```
const links = [
  { text: 'Google', url: 'https://www.google.de/'},
  { text: 'htw saar', url: 'https://www.htwsaar.de/'}
]
```

Schreiben Sie eine Funktion `createLinkList`, die ein solches Feld als Argument erhält, daraus eine ungeordnete Liste von Links (a-Elementen) erzeugt und diese Liste als Kindelement des body-Elementes anhängt. Jeder Link verweist auf die gegebene URL und hat als Beschriftung den Text. Im Beispiel ergibt sich:

- Google
- htw saar

Ergänzen Sie den JavaScript-Code, um die geforderte Funktionalität zu erreichen. Verwenden Sie weder `innerHTML` noch `insertAdjacentHTML`.

```
function createLinkList(links) {
  const body = document.body









}
```

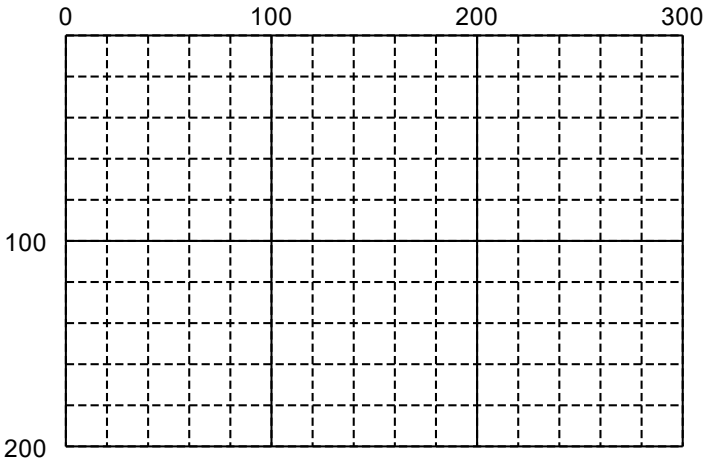
Aufgabe 6

(10 Punkte) Skizzieren Sie die folgenden zwei SVG-Graphiken auf dem dafür vorgesehenen Gitter. Folgende Eigenschaften sind per CSS festgelegt:

```
circle,line,rect,path, polygon {
    stroke: black;stroke-width: 2;fill: transparent;
}
```

a)

```
<svg width="300" height="200" version="1.1">
   <path d="M 200,0 L 100,120 H 200 v -60 l 60,-60" />
</svg>
```



b)

```
<svg width="300" height="200" version="1.1">
  <path d="M 300,0 L 260,80" />
  <g transform="translate(40,60)">
      <rect width="200" height="100" x="20" y="20" />
      <g transform="scale(0.5)">
          <rect width="200" height="100" x="160" y="80" />
      </g>
  </g>
</svg>
```