Klausur Web-Applikationen

Name:

Matrikelnummer:

Aufgabe	**-1-**	**-2-**	**-3-**	**-4-**	**-5-**	**-6-**	**-7-**
Punkte	25	7	6	22	7	9	14
erreicht							

Summe (max. 90 Punkte):

Note:

Zeit: 90 Minuten

Erlaubte Hilfsmittel: ein DIN-A4-Blatt mit eigenen Notizen (beidseitig)

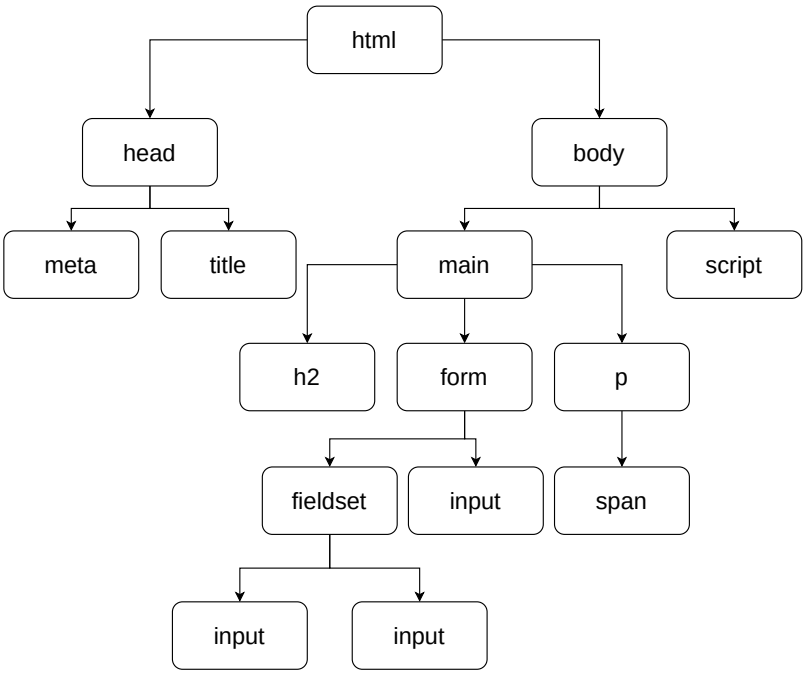
Viel Erfolg!

Aufgabe 1 (HTML & JavaScript) (25 Punkte)

Skizzieren Sie den zum HTML-Code zugehörigen DOM-Tree (ohne Text-Knoten). (4 Punkte)

```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Addierer</title>
  </head>
  <body>
    <main>
      <h2>Addierer</h2>
      <form>
        <fieldset>
          <input type="number" name="first" id="first" />
          <input type="number" name="second" id="second" />
        </fieldset>
        <input type="submit" id="calc" value="" />
      </form>
      <p>
        Die Summe lautet:
        <span id="result"></span>
      </p>
    </main>
    <script>
      // add first and second
      const add = () => {};
    </script>
  </body>
</html>
```

DOM-Tree:



Im `<script>`-Tag befindet sich eine unvollständige JavaScript Funktion `add()`. Implementieren Sie diese, sodass `add()`

- die Zahlen in den Eingabefeldern des Formulars addiert (Hinweis: auch bei `type="number"` sind Eingaben zunächst vom Datentyp `string`)
- die Summe auf zwei Nachkommastellen mit Hilfe der builtin-Funktion `.toFixed(2)` kürzt (z.B. `5.759.toFixed(2)` liefert `"5.76"`)
- das Ergebnis als Rückgabewert ausgibt (Datentyp `number` oder `string`)

(8 Punkte)

```
const add = () => {

const firstValue = document.getElementById("first").value;
const secondValue = document.getElementById("second").value;
const sum = parseFloat(firstValue) + parseFloat(secondValue);
return sum.toFixed(2);

}
```

Der nachfolgende Code setzt einen Klick-Listener auf den Submit-Button. Im Callback soll die nun mit `add()` korrekt berechnete Summe unter dem Formular angezeigt werden. Warum wird dies nicht wie gewünscht funktionieren? Was muss an welcher Stelle dem Code hinzugefügt werden, damit es funktioniert? (3 Punkte)

```
const calc = document.getElementById("calc");
calc.addEventListener(
  "click",
  function (e) {
    const sum = add();
    document.getElementById("result").innerHTML = sum;
  },
  false
);
```

Die Standardaktion beim Klick auf einen Button vom Typ `submit` ist das Absenden des Formulars. In diesem Zuge lädt der Browser die Seite neu, wodurch der zum `span`-Element hinzugefügte Text wieder verschwindet. Das Standardverhalten kann unterdrückt werden, indem man `e.preventDefault()` zur callback Funktion hinzufügt.

Erläutern Sie den Begriff "Barrierefreiheit" in Bezug auf das World Wide Web in 2-3 Sätzen. Beschreiben Sie zwei Wege, das HTML-Dokument zu Beginn dieser Aufgabe diesbezüglich zu verbessern. (6 Punkte)

Menschen haben nicht das gleiche Seh- und Hörvermögen, unterscheiden sich in kognitiven Fähigkeiten, Bewegungsmöglichkeiten, aber auch im Zugang zu Hard- und Software. Barrierefreiheit bedeutet Webseiten so zu gestalten, dass sie trotz aller Unterschiede von so vielen Menschen wie möglich genutzt werden können.

Verbesserungsmöglichkeiten: (2 genügen)

- Attribut `lang` auf `lang=de` umstellen, z.B. für korrekte Aussprache der deutschen Texte bei Screenreadern
- `value` des Submit-Buttons nicht leer lassen, um Aktion zu kennzeichnen
- ein `<legend>`-Tag zum `<fieldset>` hinzufügen, um Informationen zur Gruppierung der Eingabefelder bereit zu stellen
- ein `<label>`-Tag für die ersten beiden Eingabefelder hinzufügen, um den Zweck der Formularelemente zu beschreiben
- input `type="reset"` hinzufügen für einfacheres Zurücksetzen
- Titel mit `<title>` hinzufügen, der die Inhalte der Seite zusammenfasst
- <h2> in <h1> ändern: ein <h2>-Tag ohne ein <h1>-Tag kann bei der Navigation der Seite verwirrend sein
- Texte in andere Sprachen übersetzen
- CSS media queries verwenden, um verschiedene Displaygrößen zu unterstützen

Wordle ist ein Spiel, bei dem ein Wort mit fünf Buchstaben erraten wird. Sie wollen einen Prototyp davon implementieren, für den Sie ein HTML-Formular verwenden. Schreiben Sie den für den unten stehenden Screenshot benötigten HTML-Code innerhalb von `<form>`. Das Eingabefeld ist ein Pflichtfeld. (4 Punkte)


```
<form>

<label for="word">Bitte Wort eingeben: *</label>
<input type="text" name="word" id="word" maxlength="5" minlength="5" required />
<input type="submit" value="Absenden" />

</form>
```

(min/maxlength optional)

Aufgabe 2 (HTTP & WWW) (7 Punkte)

Vervollständigen Sie die Tabelle mit den numerischen HTTP-Statuscodes der gegebenen Nachrichten.
(3 Punkte)

Nachricht	Statuscode
OK	200
Internal Server Error	500
Unauthorized	401

`Authorization` und `Cookie` sind zwei Beispiele für HTTP-Request-Felder. Nennen Sie zwei weitere.
(2 Punkte)

Host, User-Agent, Content-Type, Accept, Date,...

Die folgende URL verweist auf die Episodenliste der Serie "Breaking Bad" auf Wikipedia. Wofür steht "#Episodenliste" und was muss im HTML-Dokument existieren, damit die URL wie gewünscht funktioniert.
(2 Punkte)

https://de.wikipedia.org/wiki/Breaking_Bad#Episodenliste

"#Episodenliste" ist Fragment/Anker und stellt eine Referenz auf eine bestimmte Stelle in der Resource dar. Im HTML-Dokument muss die entsprechende Sprungmarke existieren, z.B. `<div id="Episodenliste">` oder `<a name="Episodenliste">`

Aufgabe 3 (JSON) (6 Punkte)

Das folgende JSON ist nicht valide. Markieren Sie die vier fehlerhaften Stellen und beschreiben Sie stichwortartig, welche Änderung jeweils getätigt werden muss. (6 Punkte)

- äußere Klammern durch "{}" ersetzen, da Objekt
- Funktionsaufruf "parseInt" nicht möglich -> Wert ändern zu: 36037
- "responsibilities" bei erstem Mitarbeiter als Array []
- Komma nach zweitem Mitarbeiter entfernen

```
[
  "name": "Hochschule Fulda",
  "street": "Leipziger Straße",
  "houseNumber": "123",
  "postalCode": parseInt("36037"),
  "city": "Fulda",
  "employees": [
    {
      "firstName": "D.",
      "lastName": "Biezā",
      "number": 3050,
      "responsibilities": {"AI-Support", "Betreuung Software-Labore"}
    },
    {
      "firstName": "C.",
      "lastName": "Pape",
      "number": 379,
      "responsibilities": ["Betreuung WI-Labor", "Virtualisierung"]
    },
  ],
  "hasLibrary": true,
  "students": 9300
]
```

Aufgabe 4 (CSS) (22 Punkte)

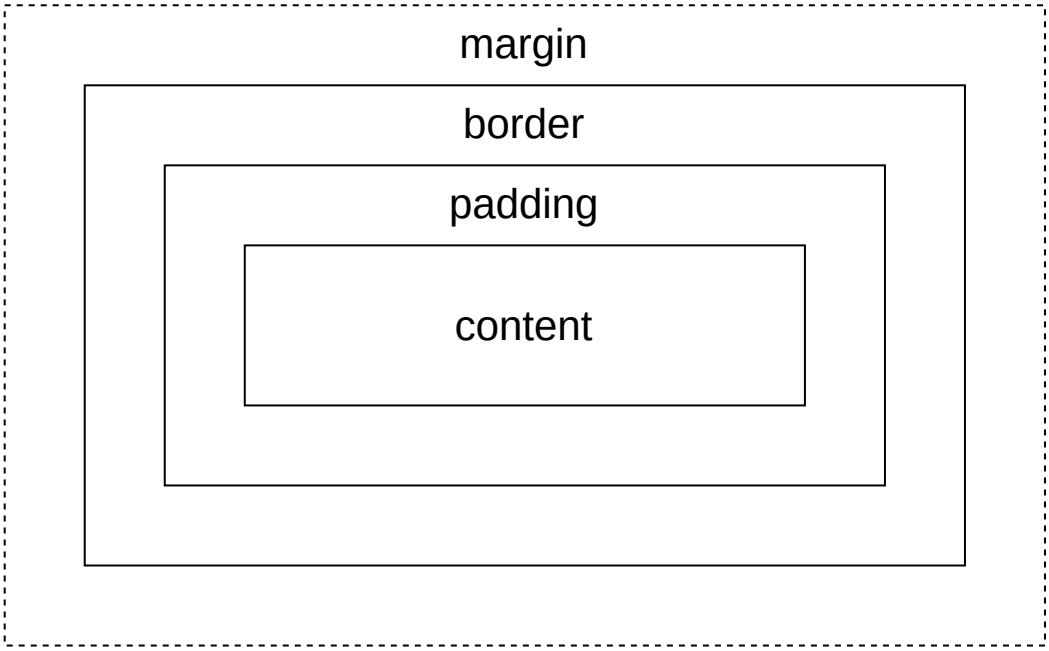
Nennen und beschreiben Sie 3 CSS Eigenschaften (CSS properties). (4 Punkte)

- background-color: setzt die Hintergrundfarbe eines Elements, z.B. durch einen Hexcode #f4aa12
- margin-top: spezifiziert den oberen, äußeren Abstand eines Elements, z.B. durch einen px-Wert
- font-size: legt die Schriftgröße fest, z.B. durch einen em-Wert oder ein Schlüsselwort wie large

Nennen Sie eine CSS-Eigenschaft, die nicht an ein Kindelement vererbt wird. (1 Punkt)

background-color, border, width, height, padding, margin, position, float, transition, display, grid-row, content, box-shadow, border-radius, left, align-self,...

Skizzieren Sie das CSS-Boxmodell. Benennen und markieren Sie dabei die vier Komponenten. (5 Punkte)



Diese Code-Ausschnitte sind für die nächsten zwei Teilaufgaben.

```
<body>
  <main>
    <h1>Orte</h1>
    <div class="landscape">
      <p id="meadow" class="green">Wiese</p>
      <p id="forest" class="green">Wald</p>
      <p id="river" class="blue">Fluss</p>
    </div>
    <div class="city">
      <p id="house" class="yellow">Haus</p>
      <p id="street" class="grey">Straße</p>
    </div>
  </main>
</body>
```

CSS:

```
main {
  color: pink;
}

.landscape {
  color: yellow;
}

.green {
  color: blue;
}

p:first-child {
  color: red;
}

.house {
  color: black;
}

#street {
  padding: 0;
}

p {
  color: orange;
}
```

Betrachten Sie den `<body>` eines HTML-Dokuments auf der vorherigen Seite. In welcher Farbe (Englisch) erscheinen die Wörter auf der Webseite, wenn das Stylesheet angewandt wird?
(9 Punkte)

Wort	Farbe
Orte	pink
Wiese	red
Wald	blue
Fluss	orange
Haus	red
Straße	orange

Nennen Sie drei CSS-Selektoren, um in dem HTML-Dokument **nur** das Element mit dem Text "Wald" zu selektieren. (3 Punkte)

```
#forest
p#forest
.green#forest
#forest.green
#meadow + p
.green + #forest
.green ~ #forest
.green ~ .green
p + #forest
p ~ #forest
.landscape > #forest
div > #forest
.green:nth-child(2)
p.green:nth-child(2)
.landscape > p:nth-child(2)
p[id='forest']
```

...

Aufgabe 5 (Reguläre Ausdrücke) (7 Punkte)

Nennen Sie 3 Strings mit einem Match zu dem regulären Ausdruck (JavaScript):
`^[Mm]ah?l(en|)$`
(3 Punkte)

- malen
- mahlen
- Malen
- Mahlen
- Mal
- Mahl

Zusatzstoffe in Lebensmitteln werden in Europa mit sogenannten "E-Nummern" gekennzeichnet. Das Format ist der Großbuchstabe `E`, gefolgt von einem Leerzeichen, gefolgt von einer Ziffernfolge.

Farbstoffe haben die E-Nummern `E 100` bis `E 199`. Erstellen Sie einen regulären Ausdruck (JavaScript), um Farbstoffe in einer Zutatenliste zu finden.

Bei Strings wie `E 1111` oder `BE 111` soll der reguläre Ausdruck nicht matchen. (4 Punkte)

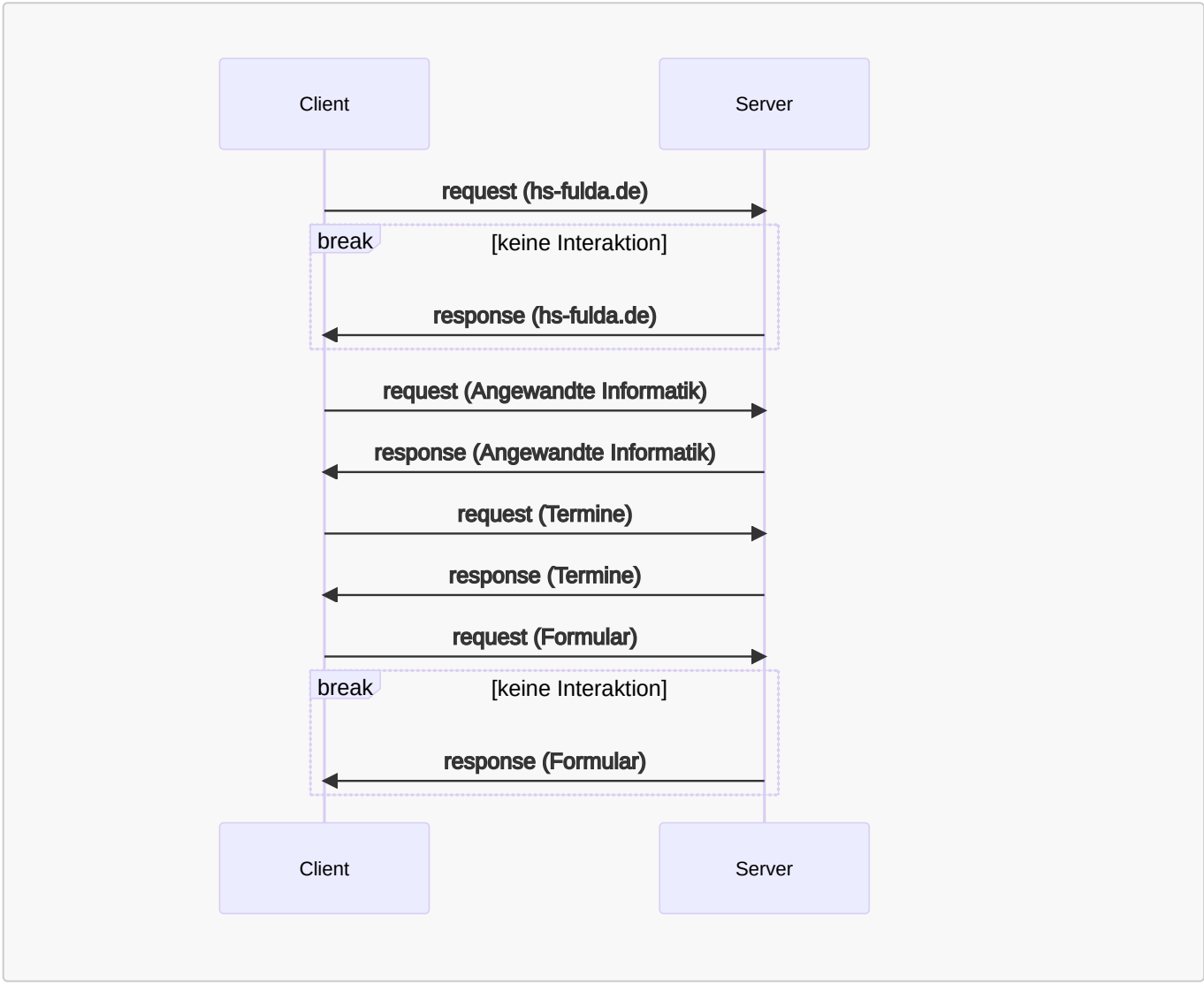
- `^E 1[0-9][0-9]$`
- `^E 1[0-9]{2}$`
- `^E[ ]1\d{2}$`
- `^(E 1)[0-9]{2}$`

Aufgabe 6 (Ajax) (9 Punkte)

Zeichnen Sie ein Diagramm, welches die folgenden Interaktionen im Zeitverlauf in einem Client-Server-Modell veranschaulicht. Kennzeichnen Sie dabei die Zeiträume, in denen der Client nicht mit der Webseite interagieren kann.

- Eingabe von "https://hs-fulda.de" in den Browser
- Klick auf "Angewandte Informatik" lädt per Ajax Teile der Seite neu
- Klick auf "Termine" lädt per Ajax Teile der Seite neu
- Klick auf "Absenden" sendet ein Formular ab und lädt eine neue Seite

(5 Punkte)



Beschreiben Sie zwei Vorteile eines asynchronen Datenflusses gegenüber eines synchronen Datenflusses? (4 Punkte)

- mehrere Anfragen können gleichzeitig versendet werden
- einzelne Schritte können in unterschiedlicher Reihenfolge erfolgen
- unabhängige Programmteile müssen nicht auf eine Antwort warten (das UI bleibt responsive)

Aufgabe 7 (Objektorientierung) (14 Punkte)

Der folgende Code-Ausschnitt ist für die nächsten zwei Teilaufgaben.

```
const Room = function (name, squareMeters) {
  this.name = name;
  this.squareMeters = squareMeters;
};
Room.prototype.openWindow = function () {
  console.log(`opening window in ${this.name}`);
};
Room.prototype.isLarge = function () {
  return this.squareMeters > 20;
};

class LivingRoom extends Room {
  isTvOn = true;

  constructor(name, squareMeters, pillows) {
    super(name, squareMeters);
    this.pillows = pillows;
  }
}

class BathRoom extends Room {
  constructor(name, squareMeters, hasBathtub) {
    super(name, squareMeters);
    this.hasBathtub = hasBathtub;
  }

  needABath() {
    console.log(this.hasBathtub ? "open the faucet" : "go swimming");
  }
}

const bathRoom = new BathRoom("bath", 15, true);
const livingRoom = new LivingRoom("living room", 21, 7);

console.log(livingRoom.squareMeters);
bathRoom.hasBathtub = false;
bathRoom.needABath();
bathRoom.openWindow();
console.log(livingRoom.isLarge());
console.log(!livingRoom.isTvOn);
console.log(bathRoom.pillows);
```

Betrachten Sie den objektorientieren Entwurf für Räume in einem Haus. Schreiben Sie die 6 Zeilen auf, die bei Ausführung des Codes auf die Konsole geloggt werden. (6 Punkte)

```
21

go swimming

opening window in bath

true

false

undefined
```

Entwerfen Sie eine weitere Klasse `Basement` (Keller), die von `Room` erbt. Ein Keller hat die Eigenschaften `name` und `squareMeters`, außerdem noch `windows` für die Anzahl an Fenstern. Wenn der Keller mindestens ein Fenster hat, soll sich `openWindow()` wie bereits implementiert verhalten. Sollte ein Keller keine Fenster haben, soll `openWindow()` stattdessen `no window available` ausloggen.

Sie können für diese Aufgabe ES5 oder ES6 Schreibweise verwenden. (8 Punkte)

ES6-Syntax:

```
class Basement extends Room {
  constructor(name, squareMeters, windows) {
    super(name, squareMeters);
    this.windows = windows;
  }

  openWindow() {
    if (this.windows < 1) {
      console.log("no window available");
    } else {
      super.openWindow();
    }
  }
}
```

ES5-Syntax:

```
const Basement = function (name, squareMeters, windows) {
  Room.call(this, name, squareMeters);
    this.windows = windows;
  };

Basement.prototype = Object.create(Room.prototype);
Basement.prototype.constructor = Basement;
Basement.prototype.openWindow = function () {
  if (this.windows < 1) {
    console.log("no window available");
  } else {
    Room.prototype.openWindow.call(this);
  }
};
```

Beispielhafter Aufruf:

```
const basement = new Basement("basement", 40, 1);
basement.openWindow();
```