Gedächtnisprotokoll (2. Termin)

Webtechnologien WS 24/25

Keine Garantie auf Vollständigkeit oder Korrektheit

1. HTML (15 Punkte)

1.1. Basics (4 Punkte)

1.1.1. Head (2 Punkte)

Betrachten Sie den folgenden (möglicherweise inkorrekten) HTML-Code und beantworten Sie die untenstehenden Fragen:

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <link rel="stylesheet" href="style.css">
  <script src="script.js">
</head>
```

wahr	falsch	
☐	☐	Es darf im `<head>` nur ein `<meta>`-Element geben
☐	☐	Inhalte von `<meta>` werden nicht gerendert, aber können von Maschinen gelesen werden
☐	☐	`<link>` ist falsch, weil an der Stelle kein klickbarer Hyperlink erwünscht ist. Stattdessen wäre `<style>` richtig
☐	☐	`<script>` braucht einen schließenden Tag und kann als Inhalt weiteren JavaScript Code haben

1.1.2. Body (2 Punkte)

Betrachten Sie den folgenden (möglicherweise inkorrekten) HTML-Code und beantworten Sie die untenstehenden Fragen:

```
<body>
  <title>Lorem Ipsum</title>
  <h1>Lorem Ipsum
  <p>Lorem ipsum<br><p>dolor sit amet</p>
</body>
```

wahr	falsch	
☐	☐	`<title>` gehört in den `<head>`, nicht in den `<body>`
☐	☐	` ` ist ein Void Tag und darf keinen schließenden Tag haben
☐	☐	`<h1>` braucht einen schließenden Tag, sonst kann der Browser die Seite nicht darstellen
☐	☐	Das erste `<p>` braucht einen schließenden Tag, sonst kann der Browser die Seite nicht darstellen

1.2. Struktur

$(2 + 2 + 2 = 6$ Punkte)

Geben Sie zwei HTML-Elemente an, die zur **Seitenstrukturierung** verwendet werden und beschreiben Sie diese kurz.

Geben Sie zwei HTML-Elemente an, die zur **Textstrukturierung** verwendet werden und beschreiben Sie diese kurz.

Geben Sie zwei HTML-Elemente an, die zur **Textformatierung** verwendet werden und beschreiben Sie diese kurz.

1.3. Formulare (5 Punkte)

Ergänzen Sie den folgenden HTML-Code. Dabei sollen die Eingabefelder für E-Mail und Passwort automatisch vom Browser jeweils entsprechend validiert werden können und das Passwort soll bei der Eingabe maskiert sein. Beim Abschicken des Formulars sollen die beiden Werte im Body einer HTTP-Anfrage an `http://example.com` gesendet werden.

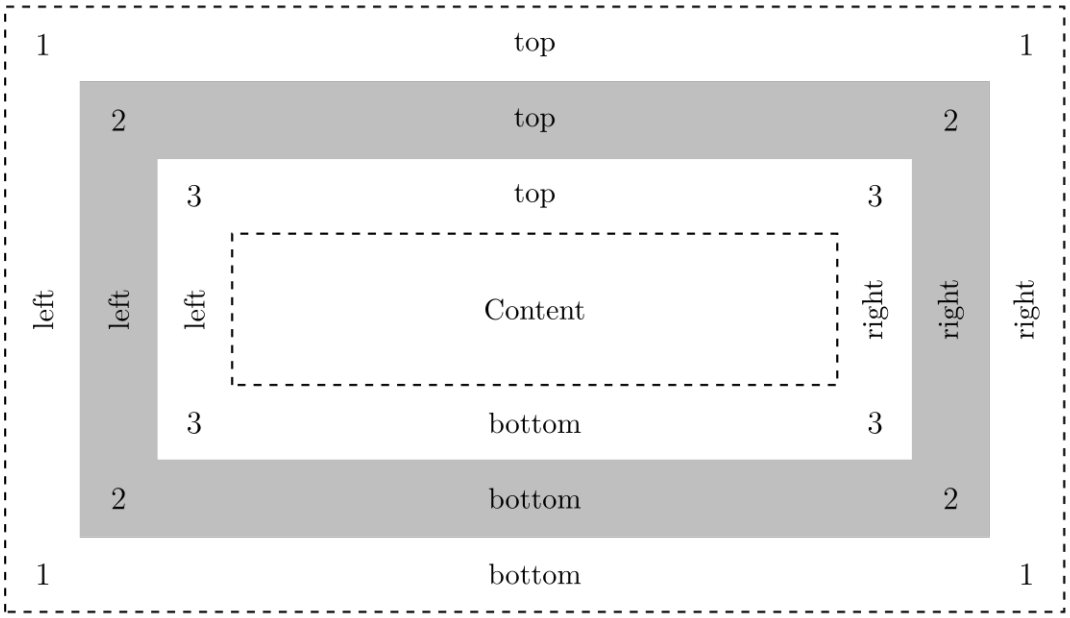
```
<form method="            " action="                        ">

  <!-- E-Mail -->

  <input type="               " name="mail">

  <!-- Passwort -->

  <input type="               " name="pw">

  <button type="               ">

</form>
```

2. CSS (18 Punkte)

2.1. Basics (6 Punkte)

2.1.1. CSS Box Model (3 Punkte)

Benennen Sie die mit 1, 2 und 3 beschrifteten Bestandteile im CSS Box Model:



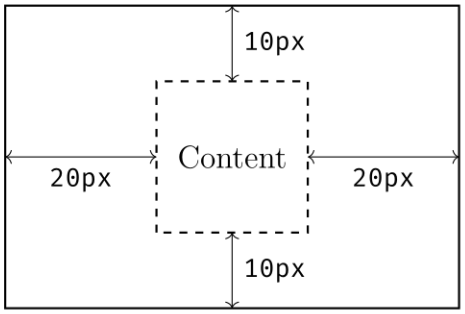
1: _____

2: _____

3: _____

2.1.2. Kurzschreibweisen (3 Punkte)

Ergänzen Sie die `padding`-Kurzschreibweise, um das dargestellte Ergebnis zu erzielen:




```
* {
    padding: _____ px  _____ px;
}
```

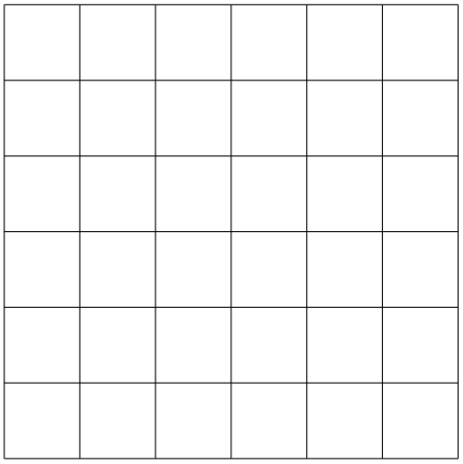
2.2. Positionierung (8 Punkte)

Betrachten Sie den folgenden HTML-Ausschnitt, der vom darunter gezeigten Stylesheet `style.css` gestyled wird. Zeichnen Sie rechts im Raster ein, wie die einzelnen `<div>`-Elemente platziert werden. Gehen Sie davon aus, dass jede Zelle im Grid 50×50 Pixel groß ist und der Viewport eine Größe von 300×300 Pixel hat.

```html
<body>
    <div class="box">1</div>
    <p></p>
    <div class="box">2</div>
    <section>
        <div class="box">3</div>
        <div class="box">4</div>
    </section>
</body>
```

style.css

```css
.box {
    width: 50px;
    height: 50px;
    position: absolute;
}
div {
    left: 100px;
    bottom: 150px;
}
section div ~ div {
    left: 150px;
    bottom: 100px;
}
section > div {
    left: 0px;
    bottom: 50px;
}
p + div {
    left: 200px;
}
```



2.3. Bootstrap (4 Punkte)

Es geht um die reaktive Darstellung von Bootstrap-Karten-Elementen (`.card`). Ergänzen Sie den Code mithilfe von Bootstrap-Breakpoints derart, dass folgende Anforderungen erfüllt werden:

- Auf Bildschirmen die kleiner als 768 px sind, stehen alle drei Karten untereinander
- Auf Bildschirmen zwischen 768 px und 992 px, stehen die ersten beiden Karten nebeneinander in einer Zeile, und die dritte in einer eigenen Zeile darunter
- Auf Bildschirmen breiter als 992 px stehen alle drei Karten nebeneinander
- Zwischen den einzelnen Karten-Containern steht jeweils ein weiteres `<div>`, das immer genau eine Spalte breit ist und von Ihnen nicht geändert werden soll. Es steht jeweils in der gleichen Zeile wie das vorherige `<div>`

Alle notwendigen Bootstrap-Breakpoints mit den dazugehörigen Größen und CSS-Klassen finden Sie in der Tabelle:

xs	sm	md	lg	xl
`col-*`	`col-sm-*`	`col-md-*`	`col-lg-*`	`col-xl-*`
< 576 px	≥ 576 px	≥ 768 px	≥ 992 px	≥ 1200 px

```
<body>

    <div class="                                                    ">

        <div class="card">
            <div class="card-body">1</div>
        </div>
    </div>
    <div class="col-1"></div>

    <div class="                                                    ">

        <div class="card">
            <div class="card-body">2</div>
        </div>
    </div>
    <div class="col-1"></div>

    <div class="                                                    ">

        <div class="card">
            <div class="card-body">3</div>
        </div>
    </div>

</body>
```

3. JavaScript und Vue.js (32 Punkte)

3.1. JS Basics (4 Punkte)

3.1.1. Arrays (2 Punkte)

```
let list = [1, 2, 3, 4];
let result = list.filter(n => n % 2 === 0).map(n => n * 2);
console.log(result.pop());
```

wahr	falsch	
☐	☐	`result` hat am Ende den Wert `[2,4]`
☐	☐	Die `filter` Funktion erstellt ein neues Array, in dem nur noch gerade Zahlen vorkommen
☐	☐	Am Ende der Ausführung enthält `result` genau ein Element
☐	☐	Auf der Konsole wird `4` ausgegeben

3.1.2. Objekte (2 Punkte)

```
function Lecture(title, capacity) {
    this.title = title;
    this.capacity = capacity;
    this[0] = 42;
}

let lecture = new Lecture('Webtech', 200);
console.log(lecture[1]);
```

wahr	falsch	
☐	☐	`lecture.title` hat den Wert `42`
☐	☐	`lecture` hat genau zwei Eigenschaften, `title` und `capacity`
☐	☐	Der Code würde so nicht funktionieren, weil die Funktion `Lecture` kein Objekt zurückgibt
☐	☐	Auf der Konsole wird `undefined` ausgegeben, weil `lecture` nicht die Eigenschaft `1` hat

3.2. Listenfunktionale (10 Punkte)

Betrachten Sie die folgende Bücherliste in JavaScript. Sie können davon ausgehen, dass alle weiteren Bücher in der Liste ebenfalls dieselbe Struktur haben.

```
let books = [
    { id: 1, title: 'Of Mice And Men', rating: 8.3, year: 1937 },
    { id: 2, title: '1984', rating: 8.9, year: 1949 }
    // ...
];
```

Schreiben Sie für die folgenden Szenarien jeweils einen Ausdruck mit Listenfunktionalen, der das gewünschte Ergebnis liefert. Sie dürfen keine Variablen verwenden; die gesamte Logik muss in dem `return`-Statement implementiert werden.

a) Alle Bücher, die mit 4.0 oder besser bewertet wurden

```
return books
```

b) Der Titel des Buches mit der schlechtesten Bewertung

```
return books
```

c) Die Titel aller Bücher, die vor 1950 veröffentlicht wurden

```
return books
```

d) Die Länge der Titel (Anzahl der Zeichen, inklusive Leerzeichen) aller Bücher, deren Bewertung besser als 4 ist und die vor 1950 veröffentlicht wurden

```
return books
```

3.3. JavaScript im Browser (7 Punkte)

Implementieren Sie die Funktion `changeColor`, die beim Aufruf die Klassen `bg-green` und `bg-red` tauscht.

```
<body>
    <div class="bg-green">1</div>
    <div class="bg-red">2</div>
    <button onclick="changeColor()">Farben tauschen</button>
    <script>









    </script>
</body>
```

Implementieren Sie eine Funktion, die beim Klicken auf den Button ausgeführt wird und den Text „Message“ zu „done!“ ändert

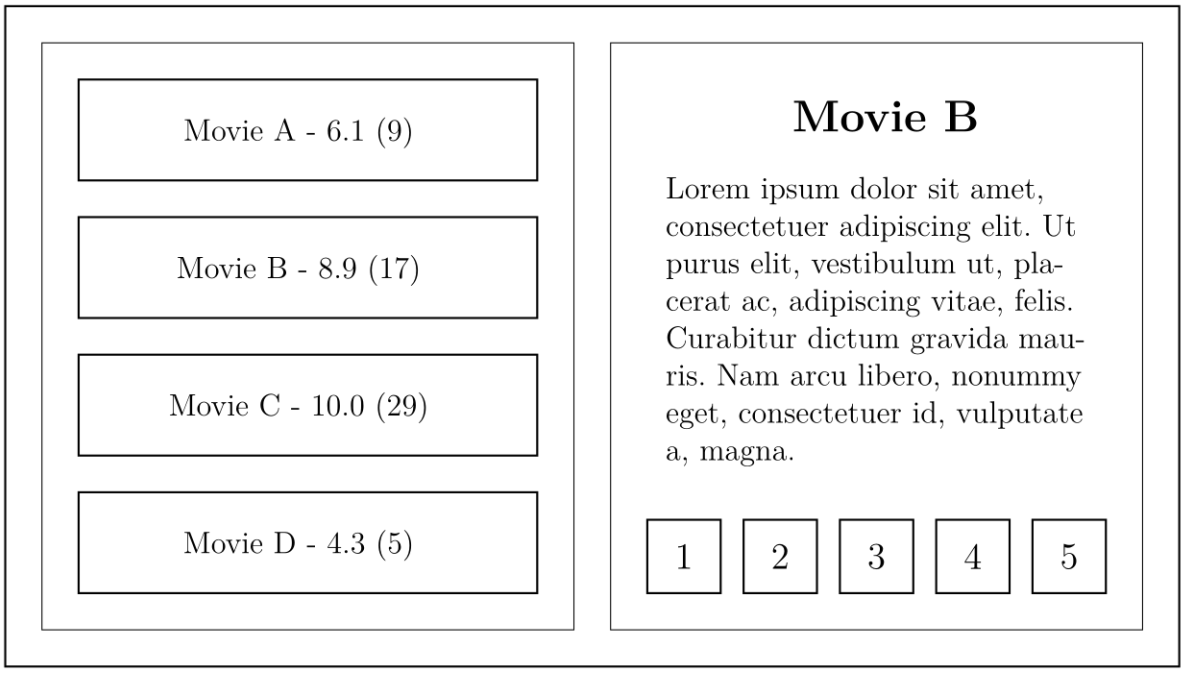
```
<body>
    <p id="text">Message</p>
    <button id="changeTextBtn">Text ändern</button>
    <script>









    </script>
</body>
```

3.4. Vue (11 Punkte)

Sie sollen eine Filmliste mit Vue.js implementieren. Die Komponente `App.vue` ist schon vorgegeben und darf nicht geändert werden. Implementieren Sie `MovieList.vue` und `MovieDetails.vue` wie folgt:

Die Filme in `MovieList.vue` sollen der durchschnittlichen Bewertung nach absteigend sortiert sein (der beste Film steht an erster Stelle). Insbesondere ist darauf zu achten, dass sich die Reihenfolge automatisch anpasst, wenn sich die Bewertungen der Filme ändern. Dabei sollen die einzelnen Filme klar voneinander abgegrenzt werden, z.B. durch Verwendung von `<div>`-Elementen oder `<ul>` und `<li>`. Für jeden Film wird jeweils der Titel, die durchschnittliche Bewertung sowie die Anzahl vorhandener Bewertungen dargestellt. Orientieren Sie sich an der Skizze unten. Wenn ein Film angeklickt wird, soll dieser ausgewählt werden, indem ein entsprechendes `selectMovie` Event ausgelöst wird.

Die Komponente `MovieDetails.vue` soll den aktuell ausgewählten Film genauer darstellen. Der Titel (`title`) des Films soll als Überschrift in einem `<h2>` Element dargestellt werden; darunter die detaillierte Beschreibung (`description`) als `<p>`. Schließlich soll es fünf Buttons (1-5) geben, um dem Film eine Bewertung hinzuzufügen. Orientieren Sie sich auch hier wieder an der unten dargestellten Skizze. Wenn dem Film mit einem der Buttons eine neue Bewertung gegeben wird, sollen die Daten des Films aktualisiert werden, indem ein entsprechendes `updateMovie` Event ausgelöst wird.



App.vue

```
<script setup>
import { ref } from 'vue';

const movies = ref([
    { id: 0, title: 'The Shining', description: '...', ratings: [3, 5, 4] },
    { id: 1, title: 'Interstellar', description: '...', ratings: [5, 5, 4, 5] }
    // ...
]);

const selectedMovie = ref(null);

function selectMovie(movie) {
    selectedMovie.value = movie;
}

function updateMovie(movie) {
    let index = movies.value.findIndex(m => m.id === movie.id);
    if (index !== -1) {
        movies[index] = movie;
    }
}
</script>

<template>
    <div class="col-6">
        <MovieList :movies="movies"
            @selectMovie="selectMovie"/>
    </div>
    <div class="col-6">
        <MovieDetails v-if="selectedMovie"
            :movie="selectedMovie"
            @updateMovie="updateMovie"/>
    </div>
</template>
```

MovieList.vue

```
<script setup>









</script>

<template>









</template>
```

MovieDetails.vue

```
<script setup>









</script>

<template>









</template>
```

4. Kommunikation und Sicherheit (15 Punkte)

4.1. Freitext (8 Punkte)

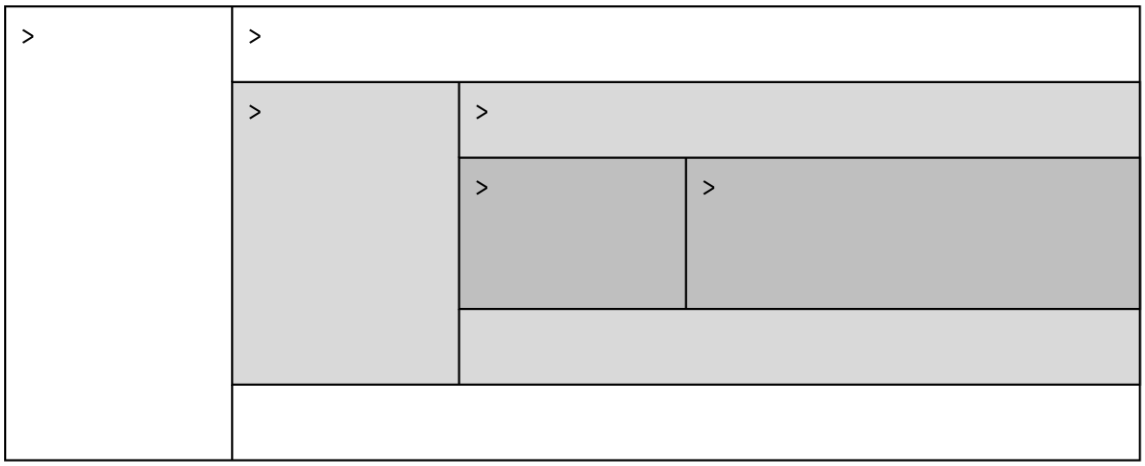
Können TCP-Segmente in HTTP-Paketen gekapselt sein? Begründen Sie Ihre Antwort.

Beschreiben sie einen Cookie-Lebenszyklus, ab dem ersten Aufruf einer Website.

Erklären Sie den Unterschied zwischen den HTTP-Methoden POST und PUT.

4.2. Protokolle (3 Punkte)

Benennen Sie in der folgenden Abbildung die Bestandteile von ineinander gekapselten HTTP-, IP- und TCP-Paketen.



4.3. Digitale Signatur (4 Punkte)

Wozu dienen digitale Signaturen? Erklären Sie auch, wie das Erstellen einer digitalen Signatur beim Sender abläuft und wie der Empfänger diese im Anschluss verwendet.

5. Server (20 Punkte)

5.1. Freitext ($2 + 2 + 3 = 7$ Punkte)

Was ist die Funktion der Controller-Komponente im MVC-Pattern?

Beschreiben Sie kurz, wofür das REST-Paradigma steht.

Was ist der Unterschied zwischen First-Party Cookies und Third-Party Cookies? Wer setzt sie jeweils und warum?

5.2. Routing (5 Punkte)

nginx.conf

```
server {
    listen 80;
    location / {
        proxy_pass http://localhost:3000;
    }
    location /kuepper {
        proxy_pass http://localhost:3001;
    }
}
```

server1.js

```
let express = require('express')();
express.get('/courses', (req, res) => {
    if (req.query.module) {
        for (module of ["Webtech", "E-Commerce", "Geschäftsprozesse"]) {
            if (module === req.query.module) {
                res.status(200).json({ module: module });
                return;
            }
        }
        res.status(404).send();
    } else {
        res.status(400).send();
    }
});
express.listen(3000);
```

server2.js

```
let http = require('http');
http.createServer((req,res) => {
    if (req.method === 'OPTIONS') {
        res.writeHead(204, { 'Allow': 'OPTIONS, GET' });
        res.end();
    } else if (req.method === 'GET') {
        res.writeHead(200, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify({ chair: 'SNET' }));
    } else {
        res.writeHead(405);
        res.end();
    }
}).listen(3001);
```

Welche URL muss aufgerufen werden, um die jeweilige Antwort im Body zu erhalten?

a) `{ "module": "Webtech" }`

b) `{ "chair": "SNET" }`

Mit welchem Statuscode antwortet der Server jeweils auf folgende Anfragen?

a) `OPTIONS http://123.1.1.123/kuepper`

b) `GET http://123.1.1.123/courses?module=MoSys`

c) `POST http://123.1.1.123/kuepper?name=Axel`

5.3. Node.js (8 Punkte)

Implementieren Sie einen Node.js-Server basierend auf dem `http`-Modul. Der Server soll nur GET-Anfragen bearbeiten und auf dem Port 8080 laufen. Bei einer Anfrage soll der Query Parameter `id` ausgelesen werden. Falls die `id` in der Anfrage nicht gesetzt ist oder einen ungültigen Wert hat, soll der Server mit dem Statuscode `404` antworten. Andernfalls die Antwort den Code `200` haben und im Body die entsprechende U-Bahn-Linie als JSON zurückgeben. Verwenden Sie `findLine` zum Finden der U-Bahn-Linie.

```
let http = require('http');
let url = require('url');

let lines = [
    { id: 8, start: 'Hermannstraße', end: 'Wittenau' },
    { id: 9, start: 'Rathaus Steglitz', end: 'Osloer Straße' }
    // ...
];

function findLine(id) {
    return lines.find(line => line.id === id);
}
```