WEB-TECHNOLOGIEN

Übung 12: DOM-Manipulation, Node.js

Aufgabe 1: DOM-Manipulation

Gegeben sei unten stehender HTML- und JavaScript-Code. Ergänzen Sie `script.js`, so dass der Inhalt des Arrays `daten` gemäß der Zieldarstellung (Sortierung beachten!) ausgegeben wird.

`seite.html`:

```html
<!DOCTYPE html>
<html>
<head>
  <title>DOM-Manipulation</title>
  <meta charset="utf-8">
</head>
<body>
  <h2>Relevante Programmiersprachen</h2>
  <table>
    <thead>
      <tr>
        <th>Rang</th>
        <th>Programmiersprache</th>
        <th>Bewertung (%)</th>
      </tr>
    </thead>
    <tbody></tbody>
  </table>
  <script src="script.js"></script>
</body>
</html>
```

`script.js`:

```js
let daten = [
  { lang: "Python", rating: 8.294 },
  { lang: "JavaScript", rating: 3.302 },
  { lang: "C", rating: 13.337 },
  { lang: "Java", rating: 16.904 },
  { lang: "Visual Basic .NET", rating: 6.459 },
  { lang: "C++", rating: 8.158 } ];

// Erstellt ein td-Element mit dem gegebenen
// Textinhalt
let createTd = function(content) {
  let td = document.createElement("td");
  td.textContent = content;
  return td;
};
```

Zieldarstellung im Browser:

Relevante Programmiersprachen

Rang	Programmiersprache	Bewertung (%)
1	Java	16.904
2	C	13.337
3	Python	8.294
4	C++	8.158
5	Visual Basic .NET	6.459
6	JavaScript	3.302

Aufgabe 2: Synchrone & Asynchrone IO

Ergänzen Sie das unten stehende Node.js-Programm `file.js` folgendermaßen:

1. Das Programm soll die Zeichenkette „Hallo WEB1“ in eine Datei `test.txt` schreiben.
2. Danach soll das Programm die Datei wieder einlesen und den gelesenen Inhalt auf der Konsole ausgeben.
3. Die Ausgabe des Programmes soll folgendermaßen aussehen (Reihenfolge der Ausgaben beachten!):

```
Datei schreiben und lesen:
Gelesener Inhalt: Hallo WEB1
```

4. Sorgen Sie dafür, dass die Dateisystemoperationen den Programmfluss nicht blockieren.

`file.js`:

```
const fs = require("fs");

// Hier den Code einfügen!

console.log("Datei schreiben und lesen:");
```

API-Hilfe:

Datei schreiben:

- **`writeFileSync(fileName, data)`**
- **`writeFile(fileName, data, callback)`**, Parameter für **`callback`**: **`err`**

Datei lesen:

- **`readFileSync(fileName)`**
- **`readFile(fileName, callback)`**, Parameter für **`callback`**: **`err, data`**

Aufgabe 3: Web-Server

Realisieren Sie mit Hilfe des “http“-Moduls von Node.js einen Web-Server mit folgenden Eigenschaften:

1. Der Web-Server soll über die URL http://localhost:8080 erreichbar sein.
2. Bei Aufruf der URL soll der Web-Server stets den Inhalt der HTML-Seite `welcome.html` liefern. Diese liegt auf dem Web-Server als Datei vor.

Hinweis: Verwenden Sie das „fs“-Modul zum Einlesen der HTML-Datei.