Aufgabe 2

Aufgabe: Inhalte in eine Datei schreiben oder hinzufügen

Option 1: Wähle, ob du mit einer synchronen oder asynchronen Methode arbeiten möchtest. Einfacher ist es mit der synchronen Methode.

Option 2: Implementiere Lösungen für beide Ansätze, falls du dich damit wohlfühlst.

Beschreibung

Schreib ein Node.js-Skript, das:

1. Den Benutzer nach einem Dateinamen fragt.
2. Den Benutzer nach einer Nachricht fragt, die in die Datei geschrieben oder an die Datei angehängt werden soll.
3. Die Nachricht in die Datei schreibt. Wenn die Datei bereits existiert, soll die Nachricht an die Datei angehängt werden.

Hinweise und Erklärungen zur Umsetzung

1. **Dateioperationen in Node.js**:
- Mach dich mit dem `fs`-Modul vertraut, das in Node.js verwendet wird, um mit dem Dateisystem zu interagieren.
- Es gibt sowohl synchrone als auch asynchrone Methoden, um Dateien zu lesen und zu schreiben. Du solltest wissen, wann du welche Methode verwenden möchtest.
2. **Benutzereingaben**:
- Verwende die `readline`-Bibliothek für asynchrone Eingaben oder `readline-sync` für synchronen Input. Dabei benötigst du immer ein Interface für die Benutzereingaben.

```JS
const readline = require('readline');

// Erstelle ein Interface für die Benutzereingabe

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});
```

Asynchrone Eingaben mit `readline`:

```JS
// Beispiel für die Benutzerabfrage

rl.question('Gib den Namen der Datei ein: ', (fileName) => {

console.log(`Dateiname: ${fileName}`);

rl.close(); // Schnittstelle schließen

});
```

Synchrone Eingaben mit `readline-sync`:

```JS
const readlineSync = require('readline-sync');

// Beispiel für die Benutzerabfrage

const fileName = readlineSync.question('Gib den Namen der Datei ein:
');

console.log(`Dateiname: ${fileName}`);
```

3. **Nachricht in die Datei schreiben**:

- Um den Inhalt in eine Datei zu schreiben oder anzuhängen, kannst du die Methoden `fs.appendFile` oder `fs.appendFileSync` verwenden. Hier sind Beispiele:

Asynchrones Anhängen mit `fs.promises.appendFile`:

```JS
const fs = require('fs').promises;

async function appendToFile() {
    try {
        await fs.appendFile('beispielTextdatei.txt', 'Neue
Nachricht\n');
        console.log('Nachricht erfolgreich hinzugefügt.');
    } catch (err) {
        console.error('Fehler beim Schreiben in die Datei:', err);
    }
}

// Funktion aufrufen
appendToFile();
```

Synchrones Anhängen mit `fs.appendFileSync`:

```JS
const fs = require('fs');

try {

fs.appendFileSync('beispielTextdatei.txt', 'Neue Nachricht\n');

console.log('Nachricht erfolgreich hinzugefügt.');

} catch (err) {

console.error('Fehler beim Schreiben in die Datei:', err);

}
```

- Denk daran, dass beim Anhängen einer Nachricht ein Zeilenumbruch (`\n`) hilfreich sein kann, um die Lesbarkeit zu erhöhen.

4. **Fehlerbehandlung**:

- Stelle sicher, dass du Fehler beim Schreiben in die Datei behandelst. Verwende `try-catch`-Blöcke für die synchrone Version und Fehler-Callbacks oder Promises für die asynchrone Version.
- Überlege, was passiert, wenn die Datei nicht existiert, und wie du das Programm so gestalten kannst, dass es in diesem Fall die Datei erstellt.

5. **Unterscheidung zwischen synchronen und asynchronen Ansätzen**:

- **Synchron**: Alle Operationen blockieren den weiteren Code, bis sie abgeschlossen sind. Dies ist einfach zu implementieren, kann aber bei langen Dateioperationen die Benutzeroberfläche einfrieren.
- **Asynchron**: Der Code kann andere Aufgaben ausführen, während auf die Dateioperation gewartet wird. Das erfordert ein Verständnis für Promises oder `async/await`.

Hinweise zur Ausführung

- Stelle sicher, dass du die benötigten Module installierst, insbesondere `readline-sync` für die synchrone Lösung, wenn du diese verwendest:

```shell
npm install readline-sync
```

- Für die asynchrone Lösung sind keine zusätzlichen Installationen notwendig, da die `readline`-Bibliothek und das `fs`-Modul bereits in Node.js enthalten sind.

Du kannst die Skripte in einer `.js`-Datei speichern und mit Node.js ausführen.