promises.md

JavaScript Promises

Was sind Promises?

- Promises sind ein asynchrones Programmiermuster in JavaScript, das es ermöglicht, mit asynchronen Operationen zu arbeiten und den Code lesbarer und besser handhabbar zu machen.

> *Das Konzept von JavaScript sieht keinerlei Nebenläufigkeit vor. Es gibt nur einen Hauptverarbeitungsstrang (Thread), und auf diesem Strang müssen alle anfallenden Aufgaben erledigt werden. Es gibt aber auch Aufgaben, die Zeit benötigen, wie z. B. ein fetch-Request oder auch länger laufende interne Operationen. Würde ein JavaScript Programm auf die Fertigstellung dieser Aufgaben warten, dann würden bis dahin keine weiteren Aufgaben mehr erledigt werden. Das betrifft vor allem die Verarbeitung von GUI-Ereignissen wie click oder input. Für den Anwender sieht es so aus, als wäre das Programm träge oder würde sogar hängenbleiben.* selfHTML

Wieso gibt es Promises?

- Promises wurden entwickelt, um das Problem der sogenannten "Callback-Hölle" zu lösen, bei dem verschachtelte Callback-Funktionen zu unleserlichem und schwer wartbarem Code führen können.

Wie funktionieren Promises?

- Promises repräsentieren den eventuellen Abschluss einer asynchronen Operation und haben drei Zustände: pending (ausstehend), fulfilled (erfüllt) und rejected (abgelehnt).
- Ein Promise wird mit einer Funktion erstellt, die zwei Parameter (resolve und reject) akzeptiert. Diese Parameter werden verwendet, um den Erfolg oder Misserfolg des Promises zu steuern.
- Der Code, der auf das Ergebnis des Promises warten soll, kann mit Hilfe von `.then()` und `.catch()` an die Promise-Instanz angehängt werden.

Beispiel für die Verwendung von Promises:

```
const promise = new Promise((resolve, reject) => {
    setTimeout(() => {
        const randomNumber = Math.random();
        if (randomNumber > 0.5) {
            resolve(randomNumber);
        } else {
            reject(new Error('Zahl kleiner als 0.5'));
        }
    }, 1000);
});

promise
    .then(result => {
        console.log('Erfolgreich:', result);
    })
    .catch(error => {
        console.log('Fehler:', error.message);
    });
```

Weitere Funktionen von Promises:

- **.then()** wird verwendet, um Code auszuführen, wenn das Promise erfolgreich erfüllt wurde.

- **.catch()** wird verwendet, um Code auszuführen, wenn das Promise abgelehnt wurde.
- **.finally()** wird verwendet, um Code auszuführen, unabhängig davon, ob das Promise erfolgreich erfüllt oder abgelehnt wurde.
- **.all()** wird verwendet, um eine Liste von Promises zu übergeben und zu warten, bis alle Promises erfolgreich erfüllt wurden.
- **.race()** wird verwendet, um eine Liste von Promises zu übergeben und zu warten, bis eines der Promises erfüllt oder abgelehnt wurde.