JavaScript: Async/Await - Mitschrift

Was ist Async/Await?

Async/Await ist eine moderne Methode, um asynchronen Code in JavaScript zu schreiben. Es basiert auf Promises, macht den Code aber lesbarer und vermeidet verschachtelte Callbacks.

Wie wird eine async-Funktion definiert?

Eine Funktion wird mit `async` als asynchron gekennzeichnet. Sie gibt immer eine Promise zurück.

Code:

```
async function meineFunktion() {
return 'Hallo Welt';
}
meineFunktion().then(console.log); // Ausgabe: Hallo Welt
```

Wofür wird await verwendet?

`await` kann innerhalb einer `async`-Funktion genutzt werden, um auf das Ergebnis einer Promise zu warten.

Code:

```
async function warteAufErgebnis() {
const promise = new Promise((resolve) => {
setTimeout(() => resolve('Ergebnis ist da!'), 2000);
});
const ergebnis = await promise;
console.log(ergebnis); // Ausgabe nach 2 Sekunden: Ergebnis ist da!
}
warteAufErgebnis();
```

Fehlerbehandlung mit try...catch

Async-Funktionen können Fehler mit einem `try...catch`-Block abfangen, ähnlich wie synchroner Code.

Code:

```
async function ladeDaten() {
try {
const response = await fetch('https://api.beispiel.com/daten');
if (!response.ok) {
throw new Error(`HTTP-Fehler! Status: ${response.status}`);
}
const daten = await response.json();
console.log(daten);
} catch (error) {
console.error('Fehler beim Laden der Daten:', error);
}
}
ladeDaten();
```

Parallele Ausführung mit Promise.all

Um mehrere asynchrone Operationen gleichzeitig auszuführen, kann `Promise.all` verwendet werden.

Code:

```
async function ladeMehrereDaten() {
const urls = [
'https://api.beispiel.com/daten1',
'https://api.beispiel.com/daten2',
'https://api.beispiel.com/daten3',
];
try {
const fetchPromises = urls.map((url) => fetch(url));
const responses = await Promise.all(fetchPromises);
const datenPromises = responses.map((response) => response.json());
const alleDaten = await Promise.all(datenPromises);
console.log(alleDaten);
} catch (error) {
console.error('Fehler beim Laden der Daten:', error);
}
}
ladeMehrereDaten();
```

Fazit

Async/Await macht den Umgang mit asynchronem Code einfacher und übersichtlicher. Es verbessert die Lesbarkeit und hilft dabei, Fehler leichter zu behandeln. Ein gutes Verständnis von Async/Await ist essenziell für moderne JavaScript-Entwicklung.