JavaScript: Callback-Funktionen - Mitschrift

Was ist eine Callback-Funktion?

Eine Callback-Funktion ist eine Funktion, die als Argument an eine andere Funktion übergeben wird. Sie wird ausgeführt, sobald eine bestimmte Aufgabe abgeschlossen ist. Das ist besonders hilfreich bei asynchronen Abläufen, wie bei Serveranfragen oder Event-Handling.

Beispiel: Einfache Callback-Funktion

Hier ein einfaches Beispiel für eine Callback-Funktion:

Code:

```
function greet(name) {
console.log(`Hallo, ${name}!`);
}
function processUserInput(callback) {
const name = prompt('Bitte deinen Namen eingeben.');
callback(name);
}
processUserInput(greet);
```

Erklärung:

1. `greet` gibt eine Begrüßung aus.
2. `processUserInput` fordert den Benutzer auf, seinen Namen einzugeben und ruft dann eine Callback-Funktion auf.
3. Dadurch bleibt der Code flexibel – eine andere Funktion kann übergeben werden, ohne `processUserInput` zu ändern.

Warum sind Callback-Funktionen nützlich?

- Asynchrone Operationen: Ideal für das Warten auf Daten von einem Server oder für Timer.
- Event-Handling: Ermöglicht das Reagieren auf Klicks, Tastendrücke usw.
- Modularität und Wiederverwendbarkeit: Funktionen bleiben flexibel und unabhängig.

Achtung: Die Callback-Hölle!

Zu viele verschachtelte Callbacks machen den Code unübersichtlich. Ein schlechtes Beispiel:

Code:

```
setTimeout(() => {
console.log('Schritt 1');
setTimeout(() => {
console.log('Schritt 2');
setTimeout(() => {
console.log('Schritt 3');
}, 1000);
}, 1000);
}, 1000);
```

Besser:

Stattdessen sollte man Promises oder async/await verwenden.

Fazit

Callback-Funktionen sind eine der Grundlagen von JavaScript. Sie helfen, Code modular und asynchron zu gestalten. Ein gutes Verständnis ist wichtig, um mit modernen Konzepten wie Promises und async/await arbeiten zu können.