async-await.md

# Javascript Async Await

## Wieso gibt es das?

- JavaScript Async Await wurde entwickelt, um asynchrone Operationen effizienter und lesbarer zu behandeln. Es ermöglicht die Ausführung von Code, während asynchrone Operationen im Hintergrund ablaufen, ohne die Ausführung zu blockieren.

## Was ist der Unterschied zu Synchron?

### Synchroner Code:

- Synchroner Code führt Aufgaben sequenziell aus und blockiert die Ausführung, bis eine Aufgabe abgeschlossen ist.
- Beispiel:

```
function synchronerCode() {
  const result1 = operation1(); // Blockiert die Ausführung
  const result2 = operation2(); // Wird erst gestartet, wenn operation1 abgeschlossen ist
  console.log(result1, result2);
}
```

### Async Await Code:

- Async Await ermöglicht die asynchrone Ausführung von Code und blockiert nicht die Ausführung anderer Aufgaben.
- Beispiel:

```
async function asyncCode() {
  const result1 = await asyncOperation1(); // Läuft im Hintergrund, während der Code fortgesetzt wird
  const result2 = await asyncOperation2(); // Wird gestartet, sobald asyncOperation1 abgeschlossen ist
  console.log(result1, result2);
}
```

## In welchen Fällen benutzt man Async Await?

- Async Await wird in den folgenden Fällen verwendet:
  - Wenn der Code auf das Ergebnis einer asynchronen Operation warten muss, bevor er fortgesetzt werden kann.
  - Wenn der Code mehrere asynchrone Operationen sequenziell ausführen soll.
  - Wenn der Code auf das Ergebnis einer asynchronen Operation basierend auf einer Bedingung warten muss.
  - Um mehrere Operation zu parallelisiert werden sollen

## Wie genau funktioniert es

- *Beispiel 1: Warten auf das Ergebnis einer asynchronen Operation ohne async await:*

```
function syncFunktion() {
    const result = asyncOperation().then(function(ergebnis) { console.log(ergebnis) });
}
```

- *Beispiel 2: Warten auf das Ergebnis einer asynchronen Operation:*

```
async function asyncFunktion() {
    const result = await asyncOperation();
    console.log(result);
}
```

- *Beispiel 3: Ausführen mehrerer asynchroner Operationen sequenziell:*

```
async function sequenziellerCode() {
    const result1 = await asyncOperation1();
    const result2 = await asyncOperation2();
    console.log(result1, result2);
}
```

- *Beispiel 4: Parallele Ausführung mehrerer asynchroner Operationen:*

```
async function parallelerCode() {
    const [result1, result2] = await Promise.all([asyncOperation1(), asyncOperation2()]);
    console.log(result1, result2);
}
```

- *Beispiel 5: Fetch mit und ohne async

```
fetch ("https://api.com/values")
        .then (response => response.json())
        .then (json => console.log (json));

(async () => {
        const response = await fetch ("https://api.com/values");
        let json =  await response.json();
        console.log (json);
})
```