JS Schleifen

Lernziele

- Verständnis des Konzepts von Schleifen
- Verständnis klassischer `for`-Schleifen
- Verständnis moderner `for...in`- und `for...of`-Schleifen
- Verständnis von `while`-Schleifen

Was ist eine Schleife

Eine Schleife führt einen bestimmten Codeblock immer wieder aus, bis ein Abbruchkriterium erreicht ist. In JavaScript existieren zwei grundlegende Arten von Schleifen:

- `while`-Schleifen: Werden verwendet, wenn eine Aufgabe so lange ausgeführt werden soll, bis ein bestimmtes Kriterium erfüllt ist.
- `for`-Schleifen: Werden häufig verwendet, wenn eine Aufgabe x Mal ausgeführt werden soll oder für jedes Element in einem Objekt/Array.

`while`

Die `while`-Schleife ist die grundlegendste Art von Schleife. Sie wiederholt einen Codeblock, solange das angegebene Kriterium `true` ist.

```
let string = "a";

while (string.length <= 8) {
  console.log(string);
  string = string + string;
}

// 'a'
// 'aa'
// 'aaaa'
// 'aaaaaaaa'
```

In diesem Beispiel wiederholt sich die while-Schleife 4 Mal, bis die Zeichenkette zu lang wird und das Schleifenkriterium zu `false` wird.

`for`

`for`-Schleifen sind dazu gedacht, eine Aufgabe so lange zu wiederholen, wie eine bestimmte Bedingung erfüllt ist. Sie bestehen aus vier internen Teilen:

- Der Initialisierungsausdruck: Der Ausdruck (falls vorhanden) wird ausgeführt. In der Regel wird ein oder mehrere Schleifenzähler initialisiert, aber es kann auch jeder andere Ausdruck ausgeführt werden, sogar Variablendeklarationen.
- Der Bedingungsausdruck: Solange die Bedingung zu `true` evaluiert wird, wird die Schleifenanweisung ausgeführt, andernfalls wird die Schleife beendet. Wenn kein Bedingungsausdruck angegeben ist, wird die Bedingung standardmäßig als `true` angenommen.
- Die Schleifenanweisung: Wird solange ausgeführt, wie der Wert der Bedingung `true` ist. Um mehrere Anweisungen auszuführen, verwende einen Blocksatz ({}).
- Der Nachdenk-Ausdruck: Falls vorhanden, wird der Nachdenk-Ausdruck nach der Schleifenanweisung ausgeführt.

```
for (initialization; condition; afterthought) statement;
```

oder

```
for (initialization; condition; afterthought) {
  statement;
  statement;
}
```

Auch wenn es nur eine einzelne Anweisung gibt, die ausgeführt werden soll, wird empfohlen, immer Blockanweisungen zu verwenden, um die Lesbarkeit zu verbessern.

```
for (let counter = 0; counter < 4; counter++) {
  console.log(counter);
}
// 0
// 1
// 2
// 3
```

Der Rumpf der for-Schleife enthält den Code, der in jeder Iteration ausgeführt wird. Im obigen Beispiel ist es ein `console.log`, das den Wert des Zählers bei jeder Iteration ausgibt, bis der Wert von `counter` 4 erreicht und die Schleife beendet wird.

```
for (let arr = [2, 4, 6]; arr.length > 0; arr.shift()) {
  console.log(arr[0]);
}
// 2
// 4
// 6
```

Die Verwendung von for-Schleifen ist nicht auf die Verwaltung einer Zählervariable beschränkt, wie im obigen Beispiel demonstriert.

`for...in`

Die `for...in`-Schleife ist eine Kurznotation, um durch alle Schlüssel eines Objekts zu iterieren:

```
const user = {
  name: "Alex",
  age: 28,
  email: "alex@mail.com",
};

for (const key in user) {
  console.log(user[key]);
}

// 'Alex'
// 28
// 'alex@mail.com'
```

Die Schleife hat eine Iterationsvariable, in diesem Fall `key`, die in jeder Iteration den jeweiligen Schlüsselwert zugewiesen bekommt (zuerst 'name', dann 'age' und schließlich 'email').

`for...of`

Ähnlich wie `for...in` ist die `for...of`-Schleife eine Kurznotation, allerdings für das Durchlaufen aller Elemente eines Arrays.

```
const fruits = ["apple", "banana", "melon"];

for (const fruit of fruits) {
  console.log(fruit);
}

// 'apple'
// 'banana'
// 'melon'
```

Dieses Mal wird die Iterationsvariable `fruit` in jeder Iteration dem jeweiligen Array-Element zugewiesen.

Ressourcen

- MDN article about loops and iterations