Schleifen mit `for`

Lernziele

Nach dieser Lektion kannst du:

- eine `for`-Schleife schreiben und lesen,
- Code eine bestimmte Anzahl von Malen wiederholen,
- einen Zähler mit `let` verwenden,
- typische Fehler wie eine Endlosschleife oder einen falschen Startwert erkennen.

Was ist eine `for`-Schleife?

Eine Schleife wiederholt einen Codeblock. Eine `for`-Schleife ist besonders praktisch, wenn du schon weißt, *wie oft* etwas passieren soll.

Stell dir vor, du möchtest fünfmal „Hallo!“ in der Konsole ausgeben. Ohne Schleife müsstest du denselben Befehl fünfmal schreiben:

```
console.log("Hallo!");
console.log("Hallo!");
console.log("Hallo!");
console.log("Hallo!");
console.log("Hallo!");
```

Mit einer `for`-Schleife geht das kürzer und übersichtlicher:

```
for (let i = 1; i <= 5; i++) {
  console.log("Hallo!");
}
```

Die Konsole zeigt:

```
Hallo!
Hallo!
Hallo!
Hallo!
Hallo!
```

Der Aufbau einer `for`-Schleife

Eine `for`-Schleife besteht aus drei Teilen:

```
for (Startwert; Bedingung; Veränderung) {
  // Code, der wiederholt wird
}
```

Ein konkretes Beispiel:

```
for (let i = 1; i <= 5; i++) {
  console.log(i);
}
```

Die drei Teile bedeuten:

Teil	Beispiel	Bedeutung
Startwert	`let i = 1`	Die Zählvariable `i` beginnt bei `1`.
Bedingung	`i <= 5`	Die Schleife läuft, solange `i` kleiner oder gleich `5` ist.
Veränderung	`i++`	Nach jedem Durchlauf wird `i` um `1` erhöht.

`i` ist ein üblicher Name für einen Zähler. Du darfst aber auch einen verständlicheren Namen wählen:

```
for (let nummer = 1; nummer <= 5; nummer++) {
  console.log(nummer);
}
```

Schritt für Schritt: Was passiert in der Schleife?

Sieh dir diese Schleife an:

```
for (let i = 1; i <= 3; i++) {
  console.log("Durchlauf: " + i);
}
```

JavaScript arbeitet sie so ab:

1. `let i = 1` erstellt den Zähler mit dem Wert `1`.
2. JavaScript prüft: Ist `i <= 3`?
3. Ja. Der Code im Block wird ausgeführt.
4. `i++` erhöht `i` auf `2`.
5. Die Bedingung wird erneut geprüft.
6. Sobald `i` den Wert `4` hat, ist `i <= 3` falsch. Die Schleife endet.

Die Ausgabe lautet:

```
Durchlauf: 1
Durchlauf: 2
Durchlauf: 3
```

> **Merke:** Der Code innerhalb der geschweiften Klammern wird bei jedem Durchlauf ausgeführt.

Zählen von 1 bis 10

Ein häufiger Anwendungsfall ist das Hochzählen:

```
for (let i = 1; i <= 10; i++) {
  console.log(i);
}
```

Ausgabe:

```
1
2
3
4
5
6
7
8
9
10
```

Achte auf den Vergleichsoperator `<=`:

- `i <= 10` bedeutet: Die `10` wird noch ausgegeben.
- `i < 10` bedeutet: Die Schleife endet bereits bei `9`.

```
for (let i = 1; i < 10; i++) {
  console.log(i);
}
```

Diese Schleife gibt nur die Zahlen von `1` bis `9` aus.

Bei 0 beginnen

In JavaScript beginnt man häufig bei `0` zu zählen. Das ist später besonders wichtig bei Arrays.

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

Ausgabe:

```
0
1
2
3
4
```

Die Schleife läuft insgesamt fünfmal:

- beim Wert `0`,
- beim Wert `1`,
- beim Wert `2`,
- beim Wert `3`,
- beim Wert `4`.

Obwohl die letzte Zahl `4` ist, gibt es fünf Durchläufe.

Rückwärts zählen

Du kannst einen Zähler auch verkleinern. Dafür verwendest du `--`.

```
for (let i = 5; i >= 1; i--) {
  console.log(i);
}
```

Ausgabe:

```
5
4
3
2
1
```

Hier gilt:

- Start bei `5`
- Wiederholung, solange `i` größer oder gleich `1` ist
- Nach jedem Durchlauf wird `i` um `1` kleiner

Das Gegenstück zu `i++` ist `i--`:

```
i++; // erhöht um 1
i--; // verringert um 1
```

In größeren Schritten zählen

Die Veränderung muss nicht immer genau `1` sein. Du kannst beispielsweise in Zweierschritten zählen:

```
for (let i = 0; i <= 10; i += 2) {
  console.log(i);
}
```

Ausgabe:

```
0
2
4
6
8
10
```

`i += 2` bedeutet: Addiere bei jedem Durchlauf `2` zu `i`.

Auch andere Schrittgrößen sind möglich:

```
for (let i = 10; i >= 0; i -= 2) {
  console.log(i);
}
```

Ausgabe:

```
10
8
6
4
2
0
```

Texte mit einem Zähler ausgeben

Der Zähler kann innerhalb der Schleife verwendet werden. So kannst du beispielsweise nummerierte Begrüßungen erzeugen:

```
for (let i = 1; i <= 3; i++) {
  console.log("Begrüßung Nummer " + i);
}
```

Ausgabe:

```
Begrüßung Nummer 1
Begrüßung Nummer 2
Begrüßung Nummer 3
```

Mit einem Template Literal ist die Schreibweise oft gut lesbar:

```
for (let i = 1; i <= 3; i++) {
  console.log(`Begrüßung Nummer ${i}`);
}
```

Berechnungen in einer Schleife

Schleifen sind nützlich, um wiederholt Berechnungen durchzuführen.

Dieses Beispiel gibt die Einmaleins-Reihe für die Zahl `4` aus:

```
for (let i = 1; i <= 10; i++) {
  let ergebnis = 4 * i;
  console.log(`4 mal ${i} ist ${ergebnis}`);
}
```

Ausgabe:

```
4 mal 1 ist 4
4 mal 2 ist 8
4 mal 3 ist 12
...
4 mal 10 ist 40
```

Die Variable `ergebnis` wird bei jedem Durchlauf neu berechnet.

Einen Wert aufsummieren

Eine Schleife kann auch Werte sammeln. Zum Beispiel möchtest du alle Zahlen von `1` bis `5` addieren.

```
let summe = 0;

for (let i = 1; i <= 5; i++) {
  summe = summe + i;
}

console.log(summe);
```

Ausgabe:

```
15
```

So verändert sich `summe`:

Durchlauf	`i`	Neue Summe
Start	–	`0`
1	`1`	`1`
2	`2`	`3`
3	`3`	`6`
4	`4`	`10`
5	`5`	`15`

Die Zeile

```
summe = summe + i;
```

kann auch kürzer geschrieben werden:

```
summe += i;
```

Beide Varianten haben dieselbe Bedeutung.

Der Gültigkeitsbereich des Zählers

Wird der Zähler mit `let` in der `for`-Schleife erstellt, existiert er nur innerhalb der Schleife:

```
for (let i = 0; i < 3; i++) {
  console.log(i);
}

console.log(i);
```

Die letzte Zeile führt zu einem Fehler, weil `i` außerhalb der Schleife nicht bekannt ist.

Das ist sinnvoll: Der Zähler wird meist nur für die Schleife benötigt. Verwende deshalb in der Regel `let` direkt im Schleifenkopf.

Häufige Fehler

Die Bedingung endet zu früh

```
for (let i = 1; i < 5; i++) {
  console.log(i);
}
```

Diese Schleife gibt `1` bis `4` aus, nicht bis `5`.

Wenn die `5` dazugehören soll, brauchst du:

```
for (let i = 1; i <= 5; i++) {
  console.log(i);
}
```

Der Zähler wird in die falsche Richtung verändert

```
for (let i = 1; i <= 5; i--) {
  console.log(i);
}
```

Hier wird `i` immer kleiner: `1`, `0`, `-1` und so weiter. Gleichzeitig bleibt die Bedingung `i <= 5` wahr. Das kann zu einer **Endlosschleife** führen.

Korrekt ist:

```
for (let i = 1; i <= 5; i++) {
  console.log(i);
}
```

> **Achtung:** Wenn eine Webseite oder die Konsole scheinbar nicht mehr reagiert, könnte eine Endlosschleife die Ursache sein. Prüfe dann besonders die Bedingung und die Veränderung des Zählers.

Ein Semikolon vergessen

Im Kopf einer `for`-Schleife stehen zwei Semikolons:

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

Falsch wäre:

```
for (let i = 0 i < 5 i++) {
  console.log(i);
}
```

JavaScript kann die drei Bereiche dann nicht mehr voneinander unterscheiden.

Geschweifte Klammern weglassen

Bei nur einer Anweisung kann JavaScript zwar ohne Klammern arbeiten:

```
for (let i = 1; i <= 3; i++)
  console.log(i);
```

Für Einsteiger ist es aber besser, die Klammern immer zu schreiben:

```
for (let i = 1; i <= 3; i++) {
  console.log(i);
}
```

So ist klar erkennbar, welcher Code zur Schleife gehört. Außerdem kannst du später leichter weitere Anweisungen ergänzen.

Praktisches Beispiel: Countdown

Ein Countdown lässt sich mit einer rückwärts laufenden Schleife umsetzen:

```
for (let sekunden = 5; sekunden >= 1; sekunden--) {
  console.log(`${sekunden} ...`);
}

console.log("Los!");
```

Ausgabe:

```
5 ...
4 ...
3 ...
2 ...
1 ...
Los!
```

Wichtig: Die Ausgaben erscheinen hier sehr schnell hintereinander. Ein echter Countdown mit einer Sekunde Pause benötigt später zusätzliche Funktionen wie `setInterval()`.

Übungen

Übung 1: Zahlen ausgeben

Schreibe eine `for`-Schleife, die alle Zahlen von `1` bis `20` in der Konsole ausgibt.

Lösung ansehen

```
for (let i = 1; i <= 20; i++) {
  console.log(i);
}
```

Übung 2: Gerade Zahlen

Gib alle geraden Zahlen von `2` bis `20` aus.

Lösung ansehen

```
for (let i = 2; i <= 20; i += 2) {
  console.log(i);
}
```

Übung 3: Rückwärts zählen

Erstelle einen Countdown von `10` bis `0`.

Lösung ansehen

```
for (let i = 10; i >= 0; i--) {
  console.log(i);
}
```

Übung 4: Kleine Rechenreihe

Gib die Ergebnisse der 7er-Reihe von `7 mal 1` bis `7 mal 10` aus.

Lösung ansehen

```
for (let i = 1; i <= 10; i++) {
  let ergebnis = 7 * i;
  console.log(`7 mal ${i} ist ${ergebnis}`);
}
```

Zusammenfassung

Eine `for`-Schleife wiederholt Code, solange ihre Bedingung wahr ist:

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

Dabei gilt:

- Der **Startwert** legt fest, wo gezählt wird.
- Die **Bedingung** entscheidet, ob ein weiterer Durchlauf erfolgt.
- Die **Veränderung** passt den Zähler nach jedem Durchlauf an.
- `i++` erhöht einen Wert um `1`, `i--` verringert ihn um `1`.
- Achte darauf, dass Bedingung und Veränderung zusammenpassen, damit keine Endlosschleife entsteht.

Im nächsten Schritt lernst du die `while`-Schleife kennen. Sie eignet sich besonders für Wiederholungen, bei denen die Anzahl der Durchläufe vorher noch nicht feststeht.

Revision #1
Created 2026-07-24 13:44:19 UTC by art10m
Updated 2026-07-24 13:44:19 UTC by art10m