Schleifen mit `while`

Wiederholen, solange eine Bedingung wahr ist

Eine `while`-Schleife wiederholt Code, **solange** eine Bedingung `true` ergibt. Sie ist besonders nützlich, wenn du vorher nicht genau weißt, wie oft sich etwas wiederholen muss.

Zum Beispiel: Eine Zahl soll so lange erhöht werden, bis sie 5 erreicht hat.

```
let zahl = 1;

while (zahl <= 5) {
  console.log(zahl);
  zahl = zahl + 1;
}
```

Die Konsole zeigt:

```
1
2
3
4
5
```

Der Aufbau einer `while`-Schleife

Eine `while`-Schleife besteht aus drei wichtigen Teilen:

```
while (bedingung) {
  // Code, der wiederholt wird
}
```

- `while` bedeutet auf Englisch „solange“.
- In den runden Klammern steht eine Bedingung.
- Der Code im Block `{ ... }` wird ausgeführt, solange die Bedingung wahr ist.

Schau dir dieses Beispiel Schritt für Schritt an:

```
let punktzahl = 0;

while (punktzahl < 3) {
  console.log("Noch nicht genug Punkte.");
  punktzahl = punktzahl + 1;
}

console.log("Ziel erreicht!");
```

So läuft das Programm ab:

1. `punktzahl` startet bei `0`.
2. Die Bedingung `punktzahl < 3` ist wahr.
3. Der Text wird ausgegeben.
4. `punktzahl` wird um `1` erhöht.
5. Die Bedingung wird erneut geprüft.
6. Sobald `punktzahl` den Wert `3` hat, ist die Bedingung falsch.
7. Die Schleife endet.

Die Ausgabe lautet:

```
Noch nicht genug Punkte.
Noch nicht genug Punkte.
Noch nicht genug Punkte.
Ziel erreicht!
```

> **Merke:** Die Bedingung wird *vor jedem Durchlauf* geprüft. Ist sie schon am Anfang `false`, wird der Schleifenblock kein einziges Mal ausgeführt.

Einen Zähler verändern

Damit eine `while`-Schleife irgendwann endet, muss sich meist eine Variable innerhalb der Schleife verändern. Diese Variable nennt man oft **Zähler**.

```
let zaehler = 1;

while (zaehler <= 4) {
  console.log("Durchlauf Nummer: " + zaehler);
  zaehler++;
}
```

`zaehler++` ist eine Kurzschreibweise für:

```
zaehler = zaehler + 1;
```

Die Schleife gibt Folgendes aus:

```
Durchlauf Nummer: 1
Durchlauf Nummer: 2
Durchlauf Nummer: 3
Durchlauf Nummer: 4
```

Du kannst einen Zähler auch rückwärts laufen lassen:

```
let countdown = 5;

while (countdown > 0) {
  console.log(countdown);
  countdown--;
}

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

`countdown--` bedeutet:

```
countdown = countdown - 1;
```

Ausgabe:

```
5
4
3
2
1
```

```
Los!
```

Achtung vor Endlosschleifen

Wenn die Bedingung immer wahr bleibt, endet eine `while`-Schleife nie. Das nennt man **Endlosschleife**. Sie kann dazu führen, dass der Browser nicht mehr reagiert.

Dieses Beispiel ist problematisch:

```
let zahl = 1;

while (zahl <= 5) {
  console.log(zahl);
}
```

Die Variable `zahl` wird nie verändert. Sie bleibt immer `1`, deshalb ist `zahl <= 5` immer wahr.

Die richtige Version erhöht `zahl` innerhalb der Schleife:

```
let zahl = 1;

while (zahl <= 5) {
  console.log(zahl);
  zahl++;
}
```

> **Prüffrage:** Wird mindestens eine Variable verändert, sodass die Bedingung irgendwann `false` werden kann?

Falls du versehentlich eine Endlosschleife startest, kannst du das Laden der Seite im Browser abbrechen oder den Tab schließen. Speichere deinen Code am besten regelmäßig.

Ein praktisches Beispiel: Guthaben abbauen

Stell dir vor, ein Spielzug kostet 2 Punkte. Solange genügend Punkte vorhanden sind, darf weitergespielt werden.

```
let guthaben = 10;

while (guthaben >= 2) {
  console.log("Eine Runde wird gespielt.");
  guthaben = guthaben - 2;
  console.log("Verbleibendes Guthaben: " + guthaben);
}

console.log("Nicht genug Guthaben für eine weitere Runde.");
```

Die Bedingung lautet `guthaben >= 2`. Eine Runde startet also nur, wenn noch mindestens 2 Punkte verfügbar sind.

Werte in einem Array mit `while` ausgeben

Auch Arrays kannst du mit einer `while`-Schleife durchgehen. Dafür benötigst du einen Index, der bei `0` beginnt.

```
const farben = ["Rot", "Blau", "Grün"];
let index = 0;

while (index < farben.length) {
  console.log(farben[index]);
  index++;
}
```

Ausgabe:

```
Rot
Blau
Grün
```

Dabei bedeutet:

- `farben.length`: Anzahl der Elemente im Array

- `farben[index]`: Element an der aktuellen Stelle
- `index++`: Zum nächsten Element wechseln

Die Bedingung `index < farben.length` ist wichtig. Der größte gültige Index liegt immer **eins unter** der Länge des Arrays.

Bei drei Elementen sind die gültigen Indizes `0`, `1` und `2`.

`while` oder `for`?

Viele Aufgaben lassen sich sowohl mit `for` als auch mit `while` lösen.

Eine `for`-Schleife passt gut, wenn die Anzahl der Wiederholungen von Anfang an klar ist:

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

Eine `while`-Schleife passt gut, wenn eine Bedingung entscheidet, wann Schluss ist:

```
let temperatur = 18;

while (temperatur < 22) {
  temperatur++;
  console.log("Temperatur: " + temperatur);
}
```

Hier ist nicht die Anzahl der Durchläufe entscheidend, sondern das Ziel: Die Temperatur soll 22 erreichen.

Verwende …	Wenn …
`for`	die Anzahl der Durchläufe klar feststeht
`while`	eine Bedingung bestimmt, wann die Wiederholung endet

Häufige Fehler

Die Zählvariable wird nicht verändert

```
let alter = 10;

while (alter < 18) {
  console.log("Noch nicht volljährig.");
}
```

Diese Schleife endet nie. Die Lösung:

```
let alter = 10;

while (alter < 18) {
  console.log("Noch nicht volljährig.");
  alter++;
}
```

Die Bedingung ist von Beginn an falsch

```
let versuche = 5;

while (versuche < 3) {
  console.log("Neuer Versuch");
  versuche++;
}
```

Hier erscheint keine Ausgabe, denn `5 < 3` ist direkt am Anfang falsch.

Falscher Vergleichsoperator

```
let nummer = 1;

while (nummer < 5) {
  console.log(nummer);
  nummer++;
}
```

Diese Schleife gibt nur `1` bis `4` aus. Soll auch die `5` erscheinen, brauchst du `<=`:

```
let nummer = 1;

while (nummer <= 5) {
  console.log(nummer);
  nummer++;
}
```

Übung: Gerade Zahlen ausgeben

Gib alle geraden Zahlen von 2 bis 10 mit einer `while`-Schleife aus.

Erwartete Ausgabe:

```
2
4
6
8
10
```

```
let zahl = 2;

while (zahl <= 10) {
  console.log(zahl);
  zahl = zahl + 2;
}
```

Übung: Begrüßungen zählen

Schreibe eine Schleife, die dreimal „Hallo!“ in der Konsole ausgibt. Verwende dafür eine Variable namens `anzahl`.

```
let anzahl = 1;

while (anzahl <= 3) {
  console.log("Hallo!");
  anzahl++;
}
```

Das Wichtigste auf einen Blick

- Eine `while`-Schleife wiederholt Code, solange ihre Bedingung `true` ist.
- Die Bedingung wird vor jedem Durchlauf geprüft.
- Eine Zählvariable muss sich meistens innerhalb der Schleife ändern.
- Ohne Veränderung kann eine Endlosschleife entstehen.
- Verwende `while`, wenn nicht die feste Anzahl der Wiederholungen, sondern eine Bedingung im Mittelpunkt steht.

Im nächsten Schritt kannst du Bedingungen und Schleifen zusammen einsetzen, um Programme flexibel auf unterschiedliche Situationen reagieren zu lassen.

Revision #1
Created 2026-07-24 13:44:20 UTC by art10m
Updated 2026-07-24 13:44:20 UTC by art10m