Tag 4 - JavaScript: Loops

Einleitung

Während `if`-Anweisungen dem Programm erlauben, **eine Entscheidung zu treffen**, erlauben **Schleifen**, einen bestimmten Codeblock **mehrfach auszuführen** – solange eine bestimmte Bedingung erfüllt ist oder bis ein bestimmter Zustand erreicht wurde. Schleifen sind essenziell für alle Formen von Automatisierung, Iteration über Daten und komplexe Programmstrukturen.

In JavaScript gibt es mehrere Schleifentypen:

- `while`
- `do...while`
- `for`
- `for...of`
- `for...in`
- sowie Kontrollbefehle wie `break` und `continue`

---

1. Was sind Schleifen?

Schleifen gehören – neben Bedingungen – zu den wichtigsten Kontrollstrukturen in jeder Programmiersprache. Sie erlauben es, bestimmte Anweisungen **wiederholt auszuführen**:

- **solange** eine Bedingung erfüllt ist (`while`, `do...while`)
- oder **eine bestimmte Anzahl von Malen** (`for`)

Statt also z. B. 10 Mal `console.log("Hallo")` zu schreiben, lassen wir eine Schleife das für uns übernehmen – das ist **effizient, flexibel und übersichtlich**.

---

2. Die `while`-Schleife

Aufbau:

```
while (Bedingung) {
  // Codeblock, der so lange wiederholt wird,
  // bis die Bedingung false ergibt
}
```

Die Bedingung wird vor jeder Ausführung geprüft. Wenn sie true ergibt, wird der Block ausgeführt – dann geht es wieder zurück zum Anfang.

Bestandteile:

- `while`: Schlüsselwort zur Einleitung der Schleife
- `(Bedingung)`: Ausdruck, der vor jeder Iteration auf `true` oder `false` geprüft wird
- `{ ... }`: Codeblock, der wiederholt ausgeführt wird, solange die Bedingung `true` ergibt

Zahlen von 0 bis 90 in Zehnerschritten:

```
let n = 0;
while (n < 91) {
  console.log(n);
  n += 10;
}
```

- Initialisierung: `let n = 0` → Startwert
- Bedingung: `n < 91` → Schleife läuft, solange `n` kleiner als 91 ist

- Inkrement: `n += 10` → Erhöhung nach jeder Runde

Typischer Anfängerfehler:

```
let x = 1;
while (x < 5) {
  console.log(x);
} // Endlosschleife! x wird nie verändert
```

Interaktive Variante:

```
let counter = 1;
while (!confirm(`[${counter}] Weiter?`)) {
  counter++;
}
```

Hier wird die Schleife nur beendet, wenn der Nutzer auf „Abbrechen“ klickt. Dabei nutzen wir confirm() und !-Negation clever aus.

---

3. Die `do...while`-Schleife

Diese Schleife funktioniert wie while, aber mit einem wichtigen Unterschied:

- Der Block wird mindestens einmal ausgeführt, da die Bedingung am Ende geprüft wird.

Aufbau:

```
do {
  // mindestens einmal ausgeführter Code
} while (Bedingung);
```

Bestandteile:

- `do`: Beginn des auszuführenden Blocks
- `{ ... }`: auszuführender Block (immer mindestens 1×)
- `while`: Nachgestellte Bedingung – Schleife läuft, solange Bedingung `true` ergibt

Beispiel:

```
let isOver;
let counter = 1;
do {
  isOver = !confirm(`[${counter++}] Weiter?`);
} while (!isOver);
```

- Schleife startet sofort
- `confirm()` liefert `true` oder `false`
- `!` negiert die Rückgabe, um Schleife zu steuern

Vergleich

```
let cond = false;
while (cond) {
  console.log("while-Schleife"); // wird nie ausgeführt
}
```

```
do {
  console.log("do...while-Schleife"); // wird einmal ausgeführt
} while (cond);
```

---

4. Die `for`-Schleife

Die for-Schleife ist ideal, wenn wir wissen, wie oft etwas wiederholt werden soll. Sie kombiniert Initialisierung, Bedingung und Inkrementierung in einer Zeile.

Aufbau:

```
for (Initialisierung; Bedingung; Inkrement) {
  // wiederholter Codeblock
}
```

Bestandteile:

- **Initialisierung**: einmaliger Startausdruck (z. B. `let i = 0`)
- **Bedingung**: wird vor jedem Schleifendurchlauf geprüft (z. B. `i < 10`)
- **Inkrement**: wird nach jedem Durchlauf ausgeführt (z. B. `i++`)
- **Block**: wiederholte Anweisungen

Beispiel:

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

Beispiel: Zahlen von 0 bis 9 ausgeben

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

Vergleich mit while:

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

Anwendung mit Arrays:

```
let werte = [10, 30, 50, 100];
let summe = 0;
for (let i = 0; i < werte.length; i++) {
  summe += werte[i];
}
console.log(summe); // → 190
```

---

5. Schleifen & Arrays – Varianten

Rückwärtslauf:

```
for (let i = array.length - 1; i >= 0; i--) {
  console.log(array[i]);
}
```

Überspringen (z. B. jedes zweite Element):

```
for (let i = 0; i < array.length; i += 2) {
  console.log(array[i]);
}
```

---

6. `for...of` – Iteration über Arrays

Aufbau:

```
for (let element of array) {
  // Zugriff auf jedes Element direkt
}
```

Bestandteile:

- `for`: Schleifenstart
- `let element`: neue Variable für jedes Element
- `of`: Schlüsselwort zur Iteration über Werte
- `array`: Datenquelle, z. B. Array, Set, String

Beispiel:

```
let werte = [10, 30, 50, 100];
let summe = 0;
for (let wert of werte) {
  summe += wert;
}
console.log(summe);
```

Beispiel mit Objekten in Arrays:

```
let städte = [
  { name: "Tokyo", einwohner: 37.26e6 },
  { name: "Delhi", einwohner: 25.87e6 },
  { name: "São Paulo", einwohner: 20.88e6 }
];

for (let stadt of städte) {
  if (stadt.einwohner > 20e6) {
    console.log(`${stadt.name} (${stadt.einwohner})`);
  }
}
```

7. `for...in` – Iteration über Objekt-Schlüssel

Aufbau:

```
for (let key in objekt) {
  // Zugriff auf Schlüssel und über key auf Werte
}
```

Bestandteile:

- `key`: enthält den Namen des jeweiligen Felds (z. B. "name")
- `objekt[key]`: Zugriff auf den Wert **(nicht mit Punktnotation nutzbar)**

Beispiel:

```
let benutzer = {
  name: "Alice",
  alter: 30,
  email: "alice@example.com"
};

for (let key in benutzer) {
  console.log(`${key} → ${benutzer[key]}`);
}
```

---

8. `break` & `continue`

`break`:

Beendet die **gesamte Schleife** sofort

```
while (true) {
  let eingabe = prompt("Zahl eingeben oder Abbrechen klicken");
  if (eingabe === null) {
    break;
  }
  console.log("Du hast eingegeben: ", eingabe);
}
```

`continue`:

Springt zur **nächsten Iteration**, überspringt aktuellen Rest

```
for (let i = 0; i < 10; i++) {
  if (i === 3) continue; // überspringt 3
  console.log(i);
}
```

9. Zusammenfassendes Beispiel: Dynamische Eingabe mit Arrays

Bestandteile:

- leeres Array: `[]`
- Schleife: `while (!fertig)`
- Eingabe: `prompt()` → gibt `string` oder `null`
- Abbruch: wenn Eingabe `null` ist → `fertig = true`
- Hinzufügen: `.push()`

```
let namen = [];
let fertig = false;
while (!fertig) {
  let eingabe = prompt("Name eingeben oder Abbrechen klicken");
  if (eingabe !== null) {
    namen.push(eingabe);
  } else {
    fertig = true;
  }
}
for (let name of namen) {
  console.log(name);
}
```

10. Übungen zu Schleifen

Aufgabe 1: `for` mit vollständigem Aufbau

```
for (let i = 1; i <= 100; i++) {
  console.log(i); // zählt von 1 bis 100
}
```

Aufgabe 2: Summierung über Schleife

- Initialisierung: `summe = 0`
- Iteration über Array
- Zugriff per Index: `werte[i]`

Aufgabe 3: Benutzereingabe mit `while`

- `prompt()` → Eingabe abfragen
- `null` → Abbrechen
- `.push()` → Array füllen

Aufgabe 4: Objektdaten mit `for...in`

- `key` → Feldname
- `objekt[key]` → Zugriff auf Wert

Aufgabe 5: Kontrollanweisungen testen

- `continue`: 5 überspringen
- `break`: bei 7 abbrechen

Aufgabe 6: `for...of` mit Filterbedingung

```
let teilnehmer = [
  { name: "Mia", punkte: 34 },
  { name: "Luca", punkte: 55 }
];
for (let person of teilnehmer) {
  if (person.punkte > 50) {
    console.log(person.name);
  }
}
```