# Variablen mit `let` und `const`

## Wozu dienen Variablen?

Variablen sind **benannte Speicherplätze** für Werte. Stell dir eine Variable wie eine beschriftete Box vor: Du legst einen Wert hinein und kannst ihn später über ihren Namen wieder verwenden.

```
let name = "Mira";
let alter = 16;

console.log(name);
console.log(alter);
```

Die Konsole zeigt:

```
Mira
16
```

Statt denselben Wert immer wieder direkt in den Code zu schreiben, gibst du ihm also einen aussagekräftigen Namen.

```
console.log("Mira");
console.log("Mira");
console.log("Mira");
```

Praktischer ist:

```
const name = "Mira";

console.log(name);
console.log(name);
console.log(name);
```

Wenn sich der Name ändern soll, musst du ihn dann nur an **einer Stelle** anpassen.

---

# Eine Variable mit `let` erstellen

Mit `let` erstellst du eine Variable, deren Wert sich später ändern darf.

```
let punkte = 0;

console.log(punkte);
```

Zunächst enthält `punkte` den Wert `0`. Später kannst du einen neuen Wert zuweisen:

```
let punkte = 0;

punkte = 10;

console.log(punkte);
```

Die Ausgabe lautet:

```
10
```

Wichtig: Beim ersten Erstellen steht `let` vor dem Variablennamen. Wenn du den Wert später nur änderst, schreibst du **kein** `let` mehr.

```
let punkte = 0;

// richtig
punkte = 10;
```

Dieser Code wäre falsch:

```
let punkte = 0;

let punkte = 10;
```

JavaScript meldet hier einen Fehler, weil die Variable `punkte` im gleichen Bereich bereits erstellt wurde.

# Wert schrittweise verändern

Eine Variable kann auch mit ihrem bisherigen Wert weitergerechnet werden.

```
let punkte = 5;

punkte = punkte + 3;

console.log(punkte);
```

Die Variable enthält danach den Wert `8`.

Das kommt häufig vor, etwa bei Punkteständen, Warenkörben oder Zählern.

```
let besucher = 0;

besucher = besucher + 1;
besucher = besucher + 1;

console.log(besucher);
```

Ausgabe:

```
2
```

---

# Eine Konstante mit `const` erstellen

Mit `const` erstellst du eine **Konstante**. Ihr Wert darf nach dem Erstellen nicht durch einen neuen Wert ersetzt werden.

```
const geburtsjahr = 2008;

console.log(geburtsjahr);
```

Dieser Code funktioniert nicht:

```
const geburtsjahr = 2008;
```

```
geburtsjahr = 2009;
```

Der Browser meldet einen Fehler, weil eine mit `const` erstellte Variable nicht neu zugewiesen werden darf.

Verwende `const`, wenn ein Wert in deinem Programm gleich bleiben soll:

```
const seitentitel = "Mein Profil";
const maximaleTeilnehmerzahl = 20;
const istEingeloggt = true;
```

Ein besonders häufiger Fall sind Texte, die sich nicht ändern sollen:

```
const begruessung = "Willkommen auf meiner Webseite!";

console.log(begruessung);
```

---

# `let` oder `const`?

Die wichtigste Frage lautet:

> **Soll dieser Wert später ersetzt werden?**

- Verwende `const`, wenn der Wert gleich bleibt.
- Verwende `let`, wenn sich der Wert ändern kann.

```
const spielName = "Zahlenraten";
let punktestand = 0;

punktestand = 1;

console.log(spielName);
console.log(punktestand);
```

In diesem Beispiel bleibt der Spielname immer gleich. Der Punktestand kann sich dagegen während des Spiels ändern.

Eine gute Gewohnheit ist:

>

> Verwende zuerst `const`. Wechsle nur zu `let`, wenn du den Wert später wirklich verändern musst.

Dadurch wird dein Code leichter verständlich. Andere Personen sehen sofort, welche Werte feststehen und welche sich im Ablauf ändern können.

---

# Variablen ohne Anfangswert

Du kannst eine Variable mit `let` auch zunächst ohne Wert erstellen:

```
let lieblingsfarbe;

console.log(lieblingsfarbe);
```

Die Ausgabe ist:

```
undefined
```

`undefined` bedeutet: Es wurde noch kein Wert gespeichert.

Später kannst du einen Wert zuweisen:

```
let lieblingsfarbe;

lieblingsfarbe = "Blau";

console.log(lieblingsfarbe);
```

Bei `const` ist das nicht erlaubt. Eine Konstante braucht **sofort** einen Wert.

```
// falsch
const lieblingsfarbe;
```

Richtig wäre:

```
const lieblingsfarbe = "Blau";
```

---

# Gute Namen für Variablen

Der Name einer Variable sollte erklären, **welchen Wert** sie enthält.

Weniger verständlich:

```
let x = 18;
let a = "Lea";
```

Besser:

```
let alter = 18;
const vorname = "Lea";
```

Bei längeren Namen wird in JavaScript üblicherweise die **camelCase-Schreibweise** verwendet: Das erste Wort beginnt klein, jedes weitere Wort beginnt mit einem Großbuchstaben.

```
let anzahlDerBesucher = 42;
const aktuellerBenutzername = "LeaM";
let istMenueGeoeffnet = false;
```

Diese Schreibweise erinnert mit den „Höckern“ an einen Kamelrücken – daher der Name *camelCase*.

# Regeln für Variablennamen

Ein Variablenname darf:

- Buchstaben enthalten,
- Zahlen enthalten, aber **nicht am Anfang**,
- `_` und `$` enthalten.

```
let spieler1 = "Noah";
let _intern = "Test";
let $preis = 9;
```

Für gut lesbaren Code solltest du `$` und `_` jedoch nur verwenden, wenn es einen klaren Grund gibt.

Ein Variablenname darf nicht mit einer Zahl beginnen:

```
// falsch
let 1spieler = "Noah";
```

Außerdem sind bestimmte Wörter bereits von JavaScript reserviert. Diese darfst du nicht als Variablennamen verwenden:

```
// falsch
let const = 5;
let if = true;
```

---

# Groß- und Kleinschreibung beachten

JavaScript unterscheidet zwischen Groß- und Kleinschreibung. `name`, `Name` und `NAME` sind drei unterschiedliche Bezeichnungen.

```
const name = "Mira";
const Name = "Jonas";

console.log(name);
console.log(Name);
```

Die Ausgabe lautet:

```
Mira
Jonas
```

Achte deshalb genau darauf, einen Variablennamen überall gleich zu schreiben.

```
const lieblingsEssen = "Pizza";

console.log(lieblingsessen);
```

Dieser Code erzeugt einen Fehler: `lieblingsessen` ist nicht dasselbe wie `lieblingsEssen`.

---

# Ein vollständiges Beispiel

Im folgenden Beispiel werden feste Angaben mit `const` gespeichert. Der Punktestand darf sich ändern und wird deshalb mit `let` erstellt.

```
const spielername = "Aylin";
const zielPunkte = 10;

let punktestand = 0;

punktestand = punktestand + 4;

console.log("Spielerin: " + spielername);
console.log("Punktestand: " + punktestand);
console.log("Ziel: " + zielPunkte);
```

Die Konsole zeigt:

```
Spielerin: Aylin
Punktestand: 4
Ziel: 10
```

Der Teil mit `+` verbindet hier Texte und Werte zu einer gemeinsamen Ausgabe. Wie du mit Texten und Zahlen noch genauer arbeitest, lernst du in den nächsten Abschnitten.

---

# Typische Fehler

## `const` neu zuweisen

```
const anzahl = 3;
anzahl = 4;
```

**Problem:** Eine Konstante darf nicht neu zugewiesen werden.

**Lösung:** Wenn sich der Wert ändern soll, verwende `let`.

```
let anzahl = 3;
anzahl = 4;
```

# Variable versehentlich erneut erstellen

```
let nachricht = "Hallo";
let nachricht = "Guten Morgen";
```

**Problem:** `nachricht` wurde im gleichen Bereich bereits erstellt.

**Lösung:** Beim Ändern nur den Namen schreiben.

```
let nachricht = "Hallo";
nachricht = "Guten Morgen";
```

# Namen unterschiedlich schreiben

```
const benutzerName = "Sam";
console.log(benutzername);
```

**Problem:** `benutzerName` und `benutzername` sind unterschiedliche Namen.

**Lösung:** Schreibe den Namen exakt gleich.

```
const benutzerName = "Sam";
console.log(benutzerName);
```

---

# Merke dir

- Variablen speichern Werte unter einem Namen.
- Mit `let` erstellst du Werte, die später verändert werden können.
- Mit `const` erstellst du Werte, die nicht neu zugewiesen werden dürfen.
- Verwende verständliche Namen wie `punktestand` oder `istEingeloggt`.
- JavaScript unterscheidet Groß- und Kleinschreibung.
- Erstelle eine Variable nur einmal mit `let` oder `const`. Danach weist du neue Werte ohne diese Wörter zu.

---

# Übung: Mein Profil

Erstelle Variablen für ein kleines Profil:

1. Speichere einen Namen, ein Alter und eine Lieblingsfarbe.
2. Entscheide für jeden Wert, ob `let` oder `const` besser passt.
3. Gib alle Werte mit `console.log()` aus.
4. Ändere anschließend das Alter um eins.

Eine mögliche Lösung:

```
const name = "Emil";
let alter = 14;
const lieblingsfarbe = "Grün";

console.log(name);
console.log(alter);
console.log(lieblingsfarbe);

alter = alter + 1;

console.log(alter);
```

**Zusatzaufgabe:** Erstelle einen Punktestand mit dem Anfangswert `0`. Erhöhe ihn dreimal und gib nach jeder Änderung den neuen Wert in der Konsole aus.

---

Revision #1
Created 2026-07-24 13:44:16 UTC by art10m
Updated 2026-07-24 13:44:16 UTC by art10m