

# JS Variablen (variables) und Zahlen (numbers)

---

## Lernziele

- den Unterschied zwischen `var`, `let` und `const` kennen
  - die verschiedenen Datentypen verstehen
  - grundlegende mathematische Operationen verwenden
- 

## Variablendeklarationen

Variablen sind eine `reference` oder ein `alias` für Daten, die im Speicher gespeichert sind. Du kannst auf diese Daten zugreifen, indem du diese Variable verwendest. Es gibt drei verschiedene Schlüsselwörter, um eine Variable zu deklarieren:

- `const` - deklariert eine Konstante, der Wert kann nicht geändert werden. `const` ist der Standardweg, um Variablen zu deklarieren. (Streng genommen, da sich ihr Wert nicht *ändern* kann, sind sie eigentlich keine *Variablen*, sondern eher **Konstanten**!). Der Geltungsbereich der Konstante ist auf den aktuellen Codeblock beschränkt.
- `let` - deklariert eine Variable, der Wert kann geändert werden. Sollte nur verwendet werden, wenn du später einen neuen Wert zuweisen musst. Der Geltungsbereich der Variable ist auf den aktuellen Codeblock beschränkt.
- `var` - deklariert eine Variable, der Wert kann geändert werden. Sollte nur verwendet werden, wenn du später einen neuen Wert zuweisen musst. Der Geltungsbereich der Variablen ist auf die aktuelle Funktion beschränkt.

Wir empfehlen die Verwendung von `const` und `let`. Die Verwendung von `var` ist weniger üblich und sollte generell zugunsten von `const` und `let` vermieden werden. In unseren Beispielen und Übungen verwenden wir in Zukunft nur `const` und `let`.

```
const aNewConstant = 1234;
```

Die Schlüsselwörter `let` (und `var` - siehe Hinweis oben) werden verwendet, wenn du dieser Variablen später einen anderen Wert zuweisen musst, beispielsweise um einen Zähler zu erhöhen.

```
let counter = 0;  
counter = counter + 1; // den Wert des Zählers neu zuweisen
```

Das `=` Zeichen in der Programmierung funktioniert nicht ganz so wie das mathematische Gleichheitszeichen, das du (vielleicht) aus der Schule kennst. Es bedeutet: "Der Wert des Elements rechts vom Gleichheitszeichen wird im Element links davon gespeichert". Was das Element auf der rechten Seite tatsächlich darstellt, wird zuerst berechnet und danach gespeichert.

## Primitive Datentypen

JavaScript ist eine dynamisch typisierte Sprache, das bedeutet, dass du nicht angeben musst, welche Art von Wert du speichern möchtest. JavaScript erkennt dies automatisch.

Es gibt 7 primitive Datentypen:

type	represents
string	eine Zeichenkette: "abcd"
number	eine Zahl: 1234
boolean	eine binäre Aussage, kann true oder false sein
null	repräsentiert "nichts", wird typischerweise von Entwicklern gesetzt
undefined	repräsentiert den Zustand des "Nichtvorhandenseins". Alles, was in JavaScript nicht angegeben oder nicht gefunden wird, hat standardmäßig den Wert undefined
BigInt	selten verwendet, wird für ganze Zahlen größer als 9007199254740991 verwendet
Symbol	selten verwendet, wird zum Erstellen einzigartiger Elemente verwendet

## Variablenbenennung

Ausdrucksstarke Variablennamen sind sehr wichtig für die Lesbarkeit des Codes. Der Code wird einfacher zu verstehen und benötigt weniger Kommentare. Es gibt einige wichtige Richtlinien, die du bei der Benennung einer Variablen beachten solltest:

- Verwende Camel Case: socialFeedEntry anstelle von socialfeedentry
- Schreibe alle Wörter aus: error statt e, followerButton statt flBtn
- Sei sehr spezifisch, längere Namen sind besser als kürzere: updatedFollowerCounter statt counter.

## Mathematik & Operatoren

Als Programmierer musst du manchmal mathematische Operationen verwenden, um bestimmte Breiten oder Positionen von Elementen zu berechnen. Operatoren berechnen Werte basierend auf einem oder zwei Ausdrücken.

operator	effect
+	Addiert zwei Zahlen.
-	Subtrahiert zwei Zahlen
*	Multipliziert zwei Zahlen
/	Dividiert zwei Zahlen
**	Potenziert zwei Zahlen: 2 ** 4 → 16
%	Der Rest oder Modulus. Gibt den Rest nach einer Ganzzahldivision zurück: 8 % 3 → 2.

Der Rest ist ein sehr nützlicher Operator, kann aber am Anfang schwer zu verstehen sein. Ein Beispiel aus dem echten Leben wäre die Zeit auf einer Uhr. Nach 12 Uhr erreichst du nicht 13 Uhr, sondern fängst um 1

Uhr nachmittags wieder an. 3 Stunden nach Mitternacht hast du nicht 15 Uhr (oder 27h im 24-Stunden-Format), sondern 3 Uhr morgens. Es ist die Stunde, die wir haben mod 12:

```
5 % 12; // → 5
12 % 12; // → 0
13 % 12; // → 1
15 % 12; // → 3
27 % 12; // → 3
```

Du kannst diesen Operator auch verwenden, um zu bestimmen, ob eine Zahl gerade oder ungerade ist:

```
6 % 2; // → 0
```

Das ist **0** für alle **geraden (even)** Zahlen, weil nach der Division einer geraden Zahl durch **2** nichts übrig bleibt.


```
5 % 2; // → 1
```

Das ist **1** für alle **ungeraden (odd)** Zahlen, weil nach dieser Division immer **1** übrig bleibt.

## Operatorenpriorität

In der Mathematik haben einige Operatoren eine höhere Priorität als andere. Das bedeutet, dass sie vor Operatoren mit niedrigerer Priorität ausgeführt werden. Zum Beispiel kommt die Multiplikation vor der Addition.

 Du kannst mehr über die [Operatorenpriorität im MDN](#) lesen.

 Wenn du unsicher bist, verwende Klammern um Berechnungen, um die Priorität manuell festzulegen. Prettier entfernt automatisch unnötige Klammern aus deinem Ausdruck.

## Zuweisungsoperatoren

Du kennst bereits den Standardzuweisungsoperator `=`. Dieser Operator weist einfach den Wert auf der rechten Seite dem Element auf der linken Seite zu. Es gibt weitere Zuweisungsoperatoren für sehr häufige Aktionen wie das Erhöhen einer Variablen um einen festen Wert.

operator	effect
<code>+=</code>	Erhöht den Wert der Variablen auf der linken Seite um den Wert auf der rechten Seite: <code>count += 6</code> → count wird um 6 erhöht
<code>-=</code>	Verringert den Wert der Variablen auf der linken Seite um den Wert auf der rechten Seite
<code>*=</code>	Multipliziert die Variable auf der linken Seite mit dem Wert auf der rechten Seite

operator	effect
/=	Dividiert die Variable auf der linken Seite durch den Wert auf der rechten Seite
++	Erhöht den Wert einer Variablen um eins: <code>count++</code> → <code>count</code> wird um eins erhöht
--	Verringert den Wert einer Variablen um eins: <code>count--</code> → <code>count</code> wird um eins verringert

## Typumwandlung

Wenn du einen Operator mit einer Variablen verwendest, die einen ungeeigneten Typ hat, wird JavaScript diese Variable automatisch in einen geeigneten Typ umwandeln. Zum Beispiel:

```
4 / "2"; // → 4 / 2
```

Es gibt keinen `/` Operator für Strings, also wandelt JavaScript den String, wenn möglich, in eine Zahl um. Das gilt auch für boolesche Operatoren, die wir in einer späteren Sitzung behandeln werden.

! Es gibt einen weiteren `+` Operator in JavaScript, der zwei Strings miteinander verknüpft: `"a" + "b" → "ab"`. Wenn du eine Zahl und einen String "addierst", wird die Zahl in einen String umgewandelt: `"a" + 6 → "a6"`. Stelle sicher, dass beide Variablen Zahlen sind, wenn du sie addieren möchtest.

📖 Lies mehr über [Typumwandlung](#) im MDN.

## Zahlensysteme

Beim Arbeiten mit Computern ist es manchmal nützlich, mit einem anderen Zahlensystem als dem Standard-Zehnersystem zu arbeiten, da ein Computer nur **binäre** Zahlen versteht, die nur aus 0 und 1 bestehen. Du musst diese Systeme nicht auswendig lernen, aber es ist gut, wenn du schon einmal davon gehört hast.

- **Dezimalsystem**: die Standardzahlen, hat 10 Symbole "0" bis "9".
- **Binärsystem**: hat nur 2 Symbole "0" und "1". Wenn du eine
- **Hexadezimalsystem**: hat 16 Symbole "0" bis "9" und "a" bis "f". Wenn du eine Zahl größer als

---

## Ressourcen

- [Operatorenpriorität](#) im MDN
- [Typumwandlung](#) im MDN