Vergleiche und logische Operatoren

Vergleiche: Werte gegenüberstellen

Mit Vergleichsoperatoren prüfst du, wie zwei Werte zueinander stehen. Das Ergebnis eines Vergleichs ist immer ein Boolean:

- `true` bedeutet: Die Aussage stimmt.
- `false` bedeutet: Die Aussage stimmt nicht.

```
console.log(5 > 3);    // true
console.log(5 < 3);    // false
console.log(5 === 5);  // true
```

Vergleiche sind wichtig, weil ein Programm damit später Entscheidungen treffen kann, etwa: „Ist die Person alt genug?“ oder „Ist ein Eingabefeld leer?“

Die wichtigsten Vergleichsoperatoren

Operator	Bedeutung	Beispiel	Ergebnis
`===`	gleich	`5 === 5`	`true`
`!==`	nicht gleich	`5 !== 3`	`true`
`>`	größer als	`10 > 7`	`true`
`<`	kleiner als	`4 < 2`	`false`
`>=`	größer oder gleich	`18 >= 18`	`true`

Operator	Bedeutung	Beispiel	Ergebnis
`<=`	kleiner oder gleich	`12 <= 10`	`false`

Schreibe die Ergebnisse testweise in die Konsole:

```
const alter = 16;


console.log(alter >= 18); // false
console.log(alter < 18);  // true
console.log(alter === 16); // true
```

> **Merke:** Ein einzelnes Gleichheitszeichen `=` weist einer Variablen einen Wert zu. Drei Gleichheitszeichen `===` vergleichen zwei Werte.

```
const punkte = 10; // Wert zuweisen


console.log(punkte === 10); // Werte vergleichen
```

Strenger Vergleich mit `===`

In JavaScript gibt es zwei Arten, Gleichheit zu vergleichen:

```
console.log(5 == "5");  // true
console.log(5 === "5"); // false
```

Der Unterschied:

- `==` vergleicht Werte recht locker. JavaScript versucht dabei, Datentypen umzuwandeln.
- `===` vergleicht **Wert und Datentyp**. Zahl und Text sind also nicht gleich.

Im ersten Beispiel wird der String `"5"` automatisch in die Zahl `5` umgewandelt. Beim strengen Vergleich mit `===` passiert das nicht.

```
const alterAlsZahl = 18;
const alterAlsText = "18";


console.log(alterAlsZahl === alterAlsText); // false
```

Für Anfänger und in modernem JavaScript gilt eine klare Empfehlung:

> Verwende fast immer `===` und `!==` statt `==` und `!=`.

Das macht deinen Code zuverlässiger und leichter verständlich.

Zahlen vergleichen

Zahlen lassen sich mit den Größenvergleichen prüfen.

```
const temperatur = 22;

console.log(temperatur > 20);  // true
console.log(temperatur < 30);  // true
console.log(temperatur >= 22); // true
console.log(temperatur <= 21); // false
```

Auch Rechnungen dürfen Teil eines Vergleichs sein:

```
const preis = 40;
const rabatt = 10;

console.log(preis - rabatt < 35); // true
```

JavaScript berechnet zunächst `preis - rabatt`. Danach wird geprüft, ob das Ergebnis kleiner als `35` ist.

Strings vergleichen

Strings kannst du mit `===` auf genau gleiche Inhalte prüfen.

```
const farbe = "blau";

console.log(farbe === "blau"); // true
console.log(farbe === "rot");  // false
```

Bei Strings zählen Groß- und Kleinschreibung:

```
console.log("Hallo" === "hallo"); // false
console.log("JavaScript" === "JavaScript"); // true
```

Auch Leerzeichen sind Teil des Textes:

```
console.log("Anna" === "Anna "); // false
```

Der zweite String enthält am Ende ein zusätzliches Leerzeichen. Solche kleinen Unterschiede können später bei Eingaben von Nutzern wichtig sein.

Booleans vergleichen

Booleans haben bereits einen Wahrheitswert. Du kannst sie trotzdem vergleichen:

```
const istAngemeldet = true;

console.log(istAngemeldet === true);  // true
console.log(istAngemeldet === false); // false
```

Oft ist der direkte Wert aber besser lesbar:

```
const istAngemeldet = true;

console.log(istAngemeldet); // true
```

Sobald du Bedingungen mit `if` verwendest, kannst du Booleans besonders praktisch einsetzen.

Logische Operatoren: Aussagen verbinden

Logische Operatoren verbinden mehrere Vergleiche zu einer größeren Aussage. Die drei wichtigsten Operatoren sind:

Operator	Bedeutung	Gesprochen als
`&&`	und	„und“
`||`	oder	„oder“

Operator	Bedeutung	Gesprochen als
`!`	nicht	„nicht“

`&&`: Beide Aussagen müssen stimmen

Der Operator `&&` bedeutet „und“. Das Gesamtergebnis ist nur dann `true`, wenn **beide** Aussagen `true` sind.

```
const alter = 20;
const hatTicket = true;

console.log(alter >= 18 && hatTicket === true); // true
```

Hier werden zwei Fragen geprüft:

1. Ist die Person mindestens 18 Jahre alt?
2. Hat sie ein Ticket?

Nur wenn beide Antworten `true` sind, ergibt auch der gesamte Ausdruck `true`.

```
console.log(true && true);   // true
console.log(true && false);  // false
console.log(false && true);  // false
console.log(false && false); // false
```

Ein weiteres Beispiel:

```
const benutzername = "Mira";
const passwortLaenge = 10;

const eingabeIstGueltig = benutzername !== "" && passwortLaenge >= 8;

console.log(eingabeIstGueltig); // true
```

Der Benutzername darf nicht leer sein **und** das Passwort muss mindestens acht Zeichen lang sein.

`||`: Mindestens eine Aussage muss stimmen

Der Operator `||` bedeutet „oder“. Das Ergebnis ist `true`, wenn mindestens eine der Aussagen stimmt.

```
const istWochenende = false;
const hatUrlaub = true;

console.log(istWochenende || hatUrlaub); // true
```

Es reicht hier, wenn Wochenende **oder** Urlaub ist.

```
console.log(true || true);   // true
console.log(true || false);  // true
console.log(false || true);  // true
console.log(false || false); // false
```

Ein praktisches Beispiel:

```
const rolle = "admin";

const darfBearbeiten = rolle === "admin" || rolle === "redaktion";

console.log(darfBearbeiten); // true
```

Eine Person darf Inhalte bearbeiten, wenn sie die Rolle `"admin"` oder `"redaktion"` besitzt.

`!`: Einen Wahrheitswert umkehren

Das Ausrufezeichen `!` bedeutet „nicht“. Es dreht einen Boolean um:

```
console.log(!true);  // false
console.log(!false); // true
```

Beispiel mit einer Variablen:

```
const istLeer = false;

console.log(!istLeer); // true
```

Du kannst damit auch prüfen, ob etwas *nicht* gleich ist. Meist ist dafür aber `!==` besser lesbar:

```
const status = "offen";

console.log(!(status === "erledigt")); // true
console.log(status !== "erledigt");    // true
```

Beide Zeilen liefern dasselbe Ergebnis. Die zweite Variante ist meist klarer.

Mehrere Operatoren kombinieren

Du darfst Vergleiche und logische Operatoren verbinden:

```
const alter = 17;
const hatBegleitung = true;

const darfTeilnehmen = alter >= 18 || hatBegleitung === true;

console.log(darfTeilnehmen); // true
```

Die Person ist noch nicht volljährig, hat aber eine Begleitung. Weil eine der beiden Aussagen stimmt, ist das Ergebnis `true`.

Bei komplexeren Ausdrücken helfen Klammern. Sie machen deutlich, welche Teile zusammengehören:

```
const alter = 20;
const hatTicket = true;
const istMitglied = false;

const darfEintritt =
  (alter >= 18 && hatTicket === true) || istMitglied === true;

console.log(darfEintritt); // true
```

Die Klammern bedeuten hier:

- Volljährig **und** Ticket vorhanden
  oder
- Mitgliedschaft vorhanden

> **Tipp:** Setze Klammern, wenn ein Ausdruck nicht sofort eindeutig lesbar ist. Das verhindert Fehler und erleichtert anderen Personen das Verstehen deines Codes.

Reihenfolge der Auswertung

JavaScript beachtet bei logischen Operatoren eine feste Reihenfolge:

1. `!` wird zuerst ausgewertet.
2. `&&` wird danach ausgewertet.
3. `||` wird zuletzt ausgewertet.

Dieses Beispiel:

```
const ergebnis = true || false && false;


console.log(ergebnis); // true
```

wird so verstanden:

```
const ergebnis = true || (false && false);


console.log(ergebnis); // true
```

Für Einsteiger ist es sinnvoll, bei gemischten Operatoren Klammern zu verwenden:

```
const ergebnis = true || (false && false);


console.log(ergebnis); // true
```

Kurzschlussauswertung

JavaScript wertet logische Ausdrücke häufig nicht vollständig aus, wenn das Ergebnis schon feststeht. Das nennt man *Kurzschlussauswertung*.

Bei `&&` reicht ein `false`, damit der gesamte Ausdruck `false` wird:

```
const istEingeloggt = false;
const hatNachrichten = true;

console.log(istEingeloggt && hatNachrichten); // false
```

Bei `||` reicht ein `true`, damit der gesamte Ausdruck `true` wird:

```
const istAdmin = true;
const istRedaktion = false;

console.log(istAdmin || istRedaktion); // true
```

Dieses Verhalten wird später nützlich, um Prüfungen sicher und effizient zu formulieren.

Typische Fehler

`=` statt `===` verwenden

```
const name = "Lea";

console.log(name = "Tom");
```

Das ist kein Vergleich. Der Wert von `name` wird hier auf `"Tom"` gesetzt.

Richtig:

```
const name = "Lea";

console.log(name === "Tom"); // false
```

String und Zahl verwechseln

```
const punkte = "10";

console.log(punkte === 10); // false
```

`punkte` ist ein String, weil der Wert in Anführungszeichen steht. Wenn du eine Zahl speichern möchtest, lasse die Anführungszeichen weg:

```
const punkte = 10;


console.log(punkte === 10); // true
```

Groß- und Kleinschreibung übersehen

```
const land = "Deutschland";


console.log(land === "deutschland"); // false
```

JavaScript unterscheidet genau zwischen `"Deutschland"` und `"deutschland"`.

Unnötig komplizierte Boolean-Vergleiche

Dieser Code funktioniert:

```
const hatBezahlt = true;


console.log(hatBezahlt === true);
```

Meist genügt jedoch:

```
const hatBezahlt = true;


console.log(hatBezahlt);
```

Für den umgekehrten Fall:

```
const hatBezahlt = false;


console.log(!hatBezahlt);
```

Praxisbeispiel: Teilnahme prüfen

Stell dir vor, eine Veranstaltung ist für Personen ab 16 Jahren. Jüngere Personen dürfen nur mit einer Begleitperson teilnehmen.

```
const alter = 14;
const hatBegleitung = true;

const darfTeilnehmen = alter >= 16 || hatBegleitung;

console.log(darfTeilnehmen); // true
```

Teste verschiedene Werte:

```
const alter = 15;
const hatBegleitung = false;

const darfTeilnehmen = alter >= 16 || hatBegleitung;

console.log(darfTeilnehmen); // false
```

Ändere `alter` und `hatBegleitung` und beobachte die Ergebnisse in der Browser-Konsole.

Übung

Lege die folgenden Variablen an:

```
const temperatur = 28;
const regnet = false;
const istWochenende = true;
```

Erstelle anschließend Variablen für diese Aussagen:

1. Die Temperatur ist mindestens 25.
2. Es regnet nicht.
3. Es ist warm und es regnet nicht.
4. Es ist Wochenende oder die Temperatur liegt über 30.
5. Es ist nicht Wochenende.

Gib jede Variable mit `console.log()` aus.

Mögliche Lösung

```
const temperatur = 28;
const regnet = false;
const istWochenende = true;

const istWarm = temperatur >= 25;
const istTrocken = !regnet;
const istGutesWetter = temperatur >= 25 && !regnet;
const istAusflugstag = istWochenende || temperatur > 30;
const istKeinWochenende = !istWochenende;

console.log(istWarm);           // true
console.log(istTrocken);        // true
console.log(istGutesWetter);    // true
console.log(istAusflugstag);    // true
console.log(istKeinWochenende); // false
```

Zusammenfassung

- Vergleichsoperatoren liefern immer `true` oder `false`.
- Verwende `===` für einen sicheren Vergleich von Wert **und** Datentyp.
- Mit `!==` prüfst du, ob zwei Werte verschieden sind.
- `&&` verlangt, dass beide Aussagen wahr sind.
- `||` verlangt, dass mindestens eine Aussage wahr ist.
- `!` kehrt einen Wahrheitswert um.
- Klammern machen komplexe logische Ausdrücke verständlicher.

Mit diesen Grundlagen kannst du im nächsten Kapitel Bedingungen formulieren: Dein Programm wird dann abhängig von Vergleichen unterschiedliche Aktionen ausführen.

Revision #1
Created 2026-07-24 13:44:17 UTC by art10m
Updated 2026-07-24 13:44:17 UTC by art10m