2.4 Falsy- und Truthy-Werte

Dieser »Verrückter Hutmacher«-Abschnitt beschreibt einen verwirrenden Aspekt von JavaScript ausführlicher. Wenn Sie dem Ratschlag aus dem vorherigen Abschnitt folgen und in Bedingungen nur boolesche Werte verwenden, können Sie ihn getrost überspringen.

In JavaScript müssen Bedingungen (also etwa die in einer `if`-Anweisung) nicht unbedingt boolesche Werte sein. Auch die »falschen« Werte `0`, `NaN`, `null`, `undefined` und der leere String lassen eine Bedingung fehlschlagen. Andere Werte werden dagegen als »wahr« gedeutet, sodass die Bedingung als erfüllt gilt. Gewöhnlich spricht man hierbei von Falsy- bzw. Truthy-Werten. Allerdings sind dies keine offiziellen Begriffe der Sprachspezifikation.

Hinweis

Falsy- und Truthy-Werte können auch bei Schleifenbedingungen, den Operanden der booleschen Operatoren `&&`, `||` und `!` sowie dem ersten Operanden von `? :` vorkommen. Alle diese Konstrukte werden noch weiter hinten in diesem Kapitel behandelt.

Auf den ersten Blick erscheinen die Konvertierungsregeln für boolesche Werte vernünftig. Nehmen wir an, Sie wollen lediglich sicherstellen, dass die Variable `performance` nicht `undefined` ist. Also schreiben Sie Folgendes:

```
if (performance) ... // Gefährlich!
```

Der Test schlägt zwar wie erwartet fehl, wenn die Variable `performance` den Wert `undefined` hat (und als Bonus auch, wenn sie `null` ist). Was aber, wenn `performance` ein leerer String oder die Zahl `0` ist? Soll der Test diese Fälle genauso behandeln wie das Fehlen eines Wertes? Manchmal kann das sinnvoll sein, manchmal aber nicht. Besser ist es, wenn der Code deutlich macht, was Sie wirklich erreichen wollen:

```
if (performance !== undefined) ...
```

2.5 Vergleichs- und Gleichheitsoperatoren

JavaScript verfügt über die üblichen Vergleichsoperatoren:

- `<` kleiner als
- `<=` kleiner oder gleich
- `>` größer als
- `>=` größer oder gleich

Beim Vergleich von Zahlen funktionieren diese Operatoren wie erwartet:

```
3 < 4  // true
3 >= 4 // false
```

Jeder Vergleich mit `NaN` ergibt `false`:

```
NaN < 4    // false
NaN >= 4   // false
NaN <= NaN // false
```

Mit denselben Operatoren lassen sich auch Strings vergleichen, wobei die alphabetische Reihenfolge beachtet wird:

```
'Hello' < 'Goodbye' // false: H steht hinter G
'Hello' < 'Hi'      // true: e steht vor i
```

Achten Sie beim Vergleich von Werten mit `<`, `<=`, `>` und `>=` darauf, dass entweder beide Operanden Zahlen oder beide Operanden Strings sind. Wandeln Sie die Operanden ggf. ausdrücklich um. Anderenfalls konvertiert JavaScript die Operanden. Das führt manchmal zu unerwünschten Ergebnissen, wie Sie im nächsten Abschnitt sehen werden.

Zum Test auf Gleichheit verwenden Sie die folgenden Operatoren:

`===` strikt gleich
`!==` nichtstrikt gleich

Diese strikten Gleichheitsoperatoren sind unproblematisch. Operanden unterschiedlicher Typen sind niemals exakt gleich. Die Werte `undefined` und `null` sind nur zu sich selbst strikt gleich. Zahlen, boolesche Werte und Strings sind nur dann strikt gleich, wenn ihre Werte gleich sind.

```
'42' === 42        // false: unterschiedliche Typen
undefined === null // false
'42' === '4' + 2   // tru: derselbe String-Wert, nämlich '42'
```

Daneben gibt es die schwachen Gleichheitsoperatoren `==` und `!=`, die auch Werte unterschiedlicher Typen vergleichen. Das ist im Allgemeinen nicht nützlich. Genaueres dazu erfahren Sie im nächsten Abschnitt.

Vorsicht

Es ist nicht möglich, `x === NaN` zu verwenden, um zu prüfen, ob `x` gleich `NaN` ist. Keine zwei `NaN`-Werte werden als gleich angesehen. Rufen Sie stattdessen `Number.isNaN(x)` auf.

Hinweis

`Object.is(x, y)` ist fast identisch mit `x === y`. Die einzigen Ausnahmen bestehen darin, dass `Object.is(+0, -0)` zu `false` ausgewertet wird und `Object.is(NaN, NaN)` zu `true`.

Wie in Java und Python bedeutet die Gleichheit von Objekten (einschließlich Arrays), dass die beiden Operanden auf dasselbe Objekt verweisen. Verweise auf verschiedene Objekte sind niemals gleich, selbst wenn die beiden Objekte den gleichen Inhalt haben.

```
let harry = { name: 'Harry Smith', age: 42 }
let harry2 = harry
harry === harry2 // true: zwei Verweise auf dasselbe Objekt
let harry3 = { name: 'Harry Smith', age: 42 }
harry === harry3 // false: verschiedene Objekte
```

2.6 Vergleiche unterschiedlicher Typen

Dies ist ein weiterer »Verrückter Hutmacher«-Abschnitt, der einen verwirrenden Aspekt von JavaScript ausführlicher beschreibt. Sie können ihn getrost überspringen, wenn Sie die goldene Regel Nr. 3 befolgen und Vergleiche unterschiedlicher Typen sowie vor allem die schwachen Gleichheitsoperatoren `==` und `!=` vermeiden.

Als Erstes sehen wir uns hier Vergleiche unterschiedlicher Typen mit den Operatoren `<`, `<=`, `>` und `>=` an.

Wenn ein Operand eine Zahl ist, wird der andere ebenfalls in eine Zahl umgewandelt. Nehmen wir an, der zweite Operator ist ein String. In diesem Fall wird er in den zugehörigen numerischen Wert umgewandelt, wenn er die String-Darstellung einer Zahl ist, in `0`, wenn der String leer ist, und in allen anderen Fällen in `NaN`. Jeder Vergleich mit `NaN` ergibt `false`, selbst `NaN <= NaN`.

```
'42' < 5     // false: '42' wird in die Zahl 42 umgewandelt
'' < 5       // true: '' wird in die Zahl 0 umgewandelt
'Hello' <= 5 // false: 'Hello' wird in NaN umgewandelt
5 <= 'Hello' // false: 'Hello' wird in NaN umgewandelt
```