aus: H. Schröder, R. Steyer: JavaScript 1.8, Herdt 2014

3.8 Operatoren

Operatoren sind Zeichen, die verwendet werden, um Berechnungen durchzuführen oder Verknüpfungen und Vergleiche zwischen Variablen oder Literalen herzustellen. Man nennt das dann auch Operationen zwischen Operanden.

Eine Operation arbeitet in der Regel mit einem oder zwei Operanden. Entsprechend wird von unären oder binären Operatoren gesprochen. Es gibt auch einen Operator mit drei Operanden, der triadischer oder ternärer Operator genannt wird.

Ein Operator erzeugt immer einen **Ergebniswert**. Manche Operatoren können nur in Verbindung mit Variablen eines bestimmten Datentyps (sinnvoll) eingesetzt werden.

Operator	Datentyp
Arithmetischer Operator	Zahlen
Vergleichsoperator	Zahlen, Strings (Zeichenketten), boolesche Werte
Verknüpfungsoperator, Konkatenationsoperator	Alle Datentypen – das Ergebnis ist aber immer ein String.
Logischer Operator	Boolesche Werte
Bit-Operator	Zahlen, boolesche Werte
Zuweisungsoperator	Alle

Arithmetische Operatoren

Arithmetische Operatoren dienen zur Berechnung eines Wertes. Dazu wird eine Operation auf einen oder mehrere Operanden angewendet.

Name	Operator	Syntax	Beispiel	Wert
Addition	+	`Summand1 + Summand2`	`7 + 4`	11
Subtraktion	-	`Minuend - Subtrahend`	`7 - 4`	3
Multiplikation	*	`Faktor1 * Faktor2`	`7 * 4`	28

Name	Operator	Syntax	Beispiel	Wert
Division	/	`Dividend / Divisor`	`7 / 4`	1.75
Modulo	%	`Dividend % Divisor`	`7 % 4`	3
Negation	-	`-Operand`	`-(2 + 5)`	-7
Inkrement	++	`++Variable;` `Variable++`	`x = 10; ++x;` `y = 135; y++`	11 136
Dekrement	--	`--Variable;` `Variable--`	`x = 10; --x;` `y = 135; y--`	9 134

Die Grundrechenarten

Für die vier Grundrechenarten Addition, Subtraktion, Multiplikation und Division verwendet JavaScript die Zeichen +, -, * und /. Bei den zugehörigen Operatoren handelt es sich um binäre Operatoren, die eine Zahl als Ergebniswert liefern.

Modulo

Der Modulo-Operator berechnet den Rest einer ganzzahligen Division. Da JavaScript keinen Operator für die ganzzahlige Division besitzt, müssen Sie diesen mithilfe des Modulo-Operators selbst nachbilden.

Beispiel

Bei der ganzzahligen Division der Zahl 23 mit der Zahl 5 bleibt ein Rest von 3 (`23 = 4 * 5 + 3`), d. h., die Zahl 5 ist viermal in der Zahl 23 enthalten. Der Rest von 3 ist nicht mehr ganzzahlig durch 5 teilbar.

```
23 % 5                  // liefert den Wert 3
(23 - (23 % 5)) / 5     // liefert den Wert 4 = ganzzahlige Division
```

Modulo ist in JavaScript auch auf Gleitkommazahlen anwendbar. Die folgenden Anweisungen sind also gültig:

```
7.1 % 5
7.1 % 3.3
```

Sie sollten allerdings niemals Modulo mit Gleitkommazahlen ausführen, da es zu Rundungsproblemen kommen kann, die an der internen Darstellung der Gleitkommazahlen hängen und nicht zu verhindern sind.

Inkrement und Dekrement

Der Inkrementoperator sowie der Dekrementoperator sind unäre Operatoren, d. h., sie werden nur auf einen Operanden angewendet. Das Inkrement ++ bewirkt, dass der Wert des Operanden um 1 erhöht wird. Das Dekrement -- bewirkt, dass der Wert um 1 reduziert wird. Die Operatoren können dabei vor oder nach dem Operanden gesetzt werden. Bei letzterer Variante besitzt der Operand während der Operation den bisherigen Wert und wird erst danach erhöht, in komplexen Ausdrücken kann sich dies jedoch auf das zurückgegebene Ergebnis auswirken.

Die Angabe vor dem Operanden wird als Präfix-, die nach dem Operanden als Postfix-Notation bezeichnet. Im ersten Fall wird die Operation vor jeder weiteren Berechnung ausgeführt, in der Postfix-Notation erst nach der Berechnung des Ausdrucks.

Beispiel: *kap03/inkdek.html*

In dem Beispiel werden Variablen mit denselben Werten unterschiedlich inkrementiert und dekrementiert. Beachten Sie die daraus resultierenden unterschiedlichen Ergebnisse.

```
<script type="text/javascript">
①   var x = 5; var y = 1.5;
    document.write("<br>x = " + x + "; y = " + y);
②   document.write("<br>x++ + y ergibt: " + (x++ + y));
    x = 5; y = 1.5;
    document.write("<p>x = " + x + "; y = " + y);
③   document.write("<br>x + ++y ergibt: " + (x + ++y));
    x = 5; y = 1.5;
    document.write("<p>x = " + x + "; y = " + y);
④   document.write("<br>x-- + y ergibt: " + (x-- + y));
    x = 5; y = 1.5;
    document.write("<p>x = " + x + "; y = " + y);
⑤   document.write("<br>x + --y ergibt: " + (x + --y));
</script>
```

① Die Variablen x und y werden mit den Werten 5 und 1.5 initialisiert.

② Zuerst werden die Werte von x und y addiert und ausgegeben, erst danach wird x automatisch um den Wert 1 erhöht.

③ Die Variable y wird schon vor der Addition um 1 erhöht und dann erst mit dem Wert von x addiert.

④ Die Werte von x und y werden addiert und ausgegeben. Dann erst wird x um 1 verringert.

⑤ Erst nachdem y um 1 verringert wurde, werden beide Variablen addiert.

x = 5; y = 1.5
x++ + y ergibt: 6.5

x = 5; y = 1.5
x + ++y ergibt: 7.5

x = 5; y = 1.5
x-- + y ergibt: 6.5

x = 5; y = 1.5
x + --y ergibt: 5.5

Verschiedene Inkrement- und Dekrementberechnungen