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

Zeichenketten (Strings)

Eine Zeichenkette (String) ist eine Folge von Zeichen, die in JavaScript in einfache oder doppelte Anführungszeichen eingeschlossen ist. Somit gibt es zwei gleichwertige Formen von Anführungszeichen. Beachten Sie, dass gleiche Arten von Anführungszeichen verwendet werden. Beginnen Sie eine Zeichenkette mit einem doppelten Anführungszeichen ", müssen Sie diese auch mit einem doppelten Anführungszeichen beenden.

Möchten Sie den Text: `Und ich fragte ihn: "Hallo, wie geht es dir?"` auf dem Bildschirm ausgeben, müssen Sie die Zeichenkette mit einem einfachen Anführungszeichen ' beginnen und beenden, da sich die doppelten Anführungszeichen bereits in der auszugebenden Zeichenkette befinden.

Angabe in JavaScript	Ausgabe
"Dies ist eine Zeichenkette."	Dies ist eine Zeichenkette.
'Das ist auch eine Zeichenkette.'	Das ist auch eine Zeichenkette.
'Die Zahl "123" ist auch eine Zeichenkette.'	Die Zahl "123" ist auch eine Zeichenkette.
"'So funktioniert es auch', sagte er."	'So funktioniert es auch', sagte er.

Der wichtigste Unterschied zwischen Zeichenketten und numerischen Werten liegt in der **Funktion der Operatoren**. Die Elemente einer Zeichenkette können nicht arithmetisch miteinander verknüpft werden. Bei ganzen Zahlen und Gleitkommazahlen entspricht das Zeichen + der mathematischen Addition. Werden zwei Zeichenketten mit dem Zeichen + verknüpft, werden sie zu einer Zeichenkette zusammengefasst. In allen anderen mathematischen Operationen werden die Zeichenketten als Zahlenwerte interpretiert. Können Zeichenketten nicht in einen Zahlenwert umgewandelt werden, liefert die Berechnung den Wert `NaN` (not a number = keine Zahl) zurück.

```
var Zahl = 1 ;
var Zeichen = '1';
var Ergebnis  = Zahl + Zahl;          // ergibt 2;
Ergebnis = Zeichen + Zeichen;         // ergibt '11';
Ergebnis = Zeichen - Zeichen;         // ergibt 0;
Ergebnis = '1a' * Zahl;               // ergibt NaN
```

Steuerzeichen in Zeichenketten

Innerhalb einer Zeichenkette können Sie Sonderzeichen angeben, um beispielsweise einen Zeilenumbruch durchzuführen oder einen Tabulator einzufügen. Diese Sonderzeichen werden mit einem Backslash \ eingeleitet und als Escape-Sequenzen oder **Steuerzeichen** bezeichnet.

Die Wirkung der Steuerzeichen ist nur innerhalb eines ausgegebenen Textes zu erkennen, z. B. innerhalb einer Meldung über die Funktion `alert()`. Bei der Ausgabe von Text im Browser über die Funktion `document.write()` müssen Sie stattdessen HTML-Tags verwenden, da der Browser Tabulatoren und Zeilenumbrüche im HTML-Code als Leerzeichen interpretiert.

Steuerzeichen	Bedeutung
`\n`	new line: Zeilenumbruch (neue Zeile)
`\r`	return: "Wagenrücklauf": Der Cursor steht in der nächsten Zeile wieder an Position 1. In JavaScript hat dies keine weitere Bedeutung.
`\t`	Tabulator
`\f`	form feed: Seitenvorschub. In JavaScript hat dies keine weitere Bedeutung.
`\b`	backspace: ein Zeichen zurück
`\"`	doppeltes Anführungszeichen, auch innerhalb von doppelten Anführungszeichen, z. B. `alert("\"");`
`\'`	einfaches Anführungszeichen, auch innerhalb von einfachen Anführungszeichen, z. B. `alert('\'');`
`\\`	Backslash

Boolesche Werte (Wahrheitswerte)

Die booleschen Werte sind `true` (wahr) und `false` (falsch). Wahrheitswerte werden eingesetzt, wenn ein Wert nur zwei Zustände annehmen kann, z. B. Licht an oder Licht aus. Ausdrücke geben häufig einen booleschen Wert zurück, z. B. beim Vergleichen von Zahlen. Der Vergleich `2 > 3` liefert das Ergebnis `false`, weil die Zahl 2 nicht größer ist als 3.

Boolescher Wert	Bedeutung
`true`	"Wahr", die Bedingung ist erfüllt, z. B. 2 < 3.
`false`	"Falsch", die Bedingung ist nicht erfüllt, z. B. 2 > 3.

Beachten Sie, dass auch Zahlenwerte in JavaScript für boolesche Ausdrücke stehen. Das Literal `0` steht für `false`, und alle numerischen Werte ungleich `0` entsprechen dem Fall, dass Sie dort `true` notieren.

Ganze Zahlen

Ganze Zahlen besitzen keine Nachkommastellen und können negative sowie positive Werte und den Wert null annehmen. In JavaScript können ganze Zahlen in einer von drei möglichen Darstellungen eingesetzt werden:

- ✔ **Dezimaldarstellung**: Dies ist die gebräuchlichste Darstellung von Zahlen. Die Zahl wird im Dezimalsystem (Basis 10) dargestellt, d. h., dass die Ziffern 0..9 verwendet werden.
- ✔ **Hexadezimaldarstellung**: Die Zahlen werden im Hexadezimalsystem (Basis 16) dargestellt. Die verwendbaren Ziffern sind 0..9, A..F (A=10, B=11 ... F=15). Hexadezimale Zahlen werden durch ein einleitendes 0x (bzw. 0X) gekennzeichnet.
- ✔ Die **Oktaldarstellung** (zur Basis 8) ist seit JavaScript 1.5 nicht mehr Bestandteil von JavaScript und wird nur noch aus Gründen der Rückwärtskompatibilität unterstützt. Oktalzahlen werden durch eine vorangestellte 0 (Null) gekennzeichnet.

Dezimal	Hexadezimal
0	0x0
1	0x1
8	0x8
15	0xF
16	0x10
42	0x2A

Gleitkommazahlen

Gleitkommazahlen können Nachkommastellen besitzen. Das Kennzeichen ist entweder ein Dezimalpunkt, ein Exponent oder beides. Bei einer Fixkommazahl ist die Position des Kommas vorgegeben, bei einer Gleitkommazahl kann das Komma "gleiten".

In JavaScript wird die amerikanische Notation, also ein Punkt , verwendet, um die Dezimalstellen einer Zahl kenntlich zu machen.

Die größte und kleinste speicherbare Zahl können Sie z. B. durch die folgenden Anweisungen ausgeben:

Fixkommazahl	Gleitkommazahl
1000.0	1.0E3
2000.4	20004.0E-1
-4000.0	-4e3
-0.004	-4e-3

Beispiel

```
document.write("Max: ", Number.MAX_VALUE);
document.write("Min: ", Number.MIN_VALUE);
```

Beachten Sie, dass beim Überschreiten des Wertebereichs der ganzen Zahlen automatisch eine Konvertierung in Gleitkommazahlen erfolgt. Gleitkommazahlen unterliegen immer Rundungsfehlern. Beachten Sie dies bei Berechnungen mit großen Zahlen.

```
k = 12345678901234567890;
document.write(k);  // liefert 12345678901234567000, da die Zahl zuerst in eine
                    // Gleitkommazahl konvertiert und danach gerundet wird !
```