Arbeit mit Strings

length

length - Länge des Strings

Suchen und Finden mit indexOf()

indexOf() - Position eines Strings innerhalb größeren Strings herausfinden

erwartet zu suchenden String als Argument und gibt Position zurück

Zählung der Position beginnt bei 0

wenn indexOf() String nicht findet – Rückgabe = -1

indexOf() gibt Position des ersten Fundes aus

indexOf() beginnt am Anfang des Strings mit Suche

lastIndexOf() findet letztes Vorkommen des gesuchten Strings

Leerzeichen entfernen

trim() entfernt Leerzeichen am Anfang und Ende eines Strings, aber nicht in Mitte

Auswahl mit charAt()

charAt() erwartet als Parameter Position des zu extrahierenden Zeichens

toUpperCase() und toLowerCase()

toLowerCase() wandelt Strings in Kleinbuchstaben

toUpperCase() wandelt Strings in Großbuchstaben

Teilstring extrahieren mit slice() und substring()

slice() und substring() (dt.: Teilzeichenkette) extrahiert Teilstring aus größerem String
extrahiert Zeichen von indexA bis exclusive indexB

benötigte Argumente:

- Position des ersten Zeichens
- Position des letzten Zeichens
  (aber Anzahl von Zeichen, die zu extrahieren – bei veraltetem substr())

Unterschiede slice() und substring()

Syntax: string.slice(indexStart, indexEnd);
Syntax: string.substring(indexStart, indexEnd);

Gemeinsamkeiten:

- wenn indexStart = indexEnd: gibt leere Zeichenfolge zurück
- wenn indexEnd weggelassen: Extrahiert Zeichen bis zum Ende der Zeichenfolge
- wenn eines der Argumente > Länge der Zeichenfolge, wird Länge der Zeichenfolge verwendet

Unterschiede:

- substring() tauscht Argumente aus, wenn indexStart > indexEnd
  - also weiterhin wird Zeichenfolge zurückgegeben.
  slice() gibt in diesem Fall leere Zeichenfolge zurück
- wenn eines oder beide Argumente negativ oder NaN, werden sie von substring()-Methode so behandelt, als ob sie 0 wären
  slice() behandelt NaN-Argumente ebenfalls als 0,
  zählt aber bei negativen Werten vom Ende des Strings rückwärts, um Indizes zu finden

Property Access

ECMAScript 5 (2009) ermöglicht den Zugriff auf Propertys [] für Zeichenfolgen:

```
let text = "HALLO WELT";
console.log(text[0]);          // => H
```

Hinweise und Probleme:

- funktioniert nicht im IE 7 oder früher
- lässt Strings wie Arrays aussehen (sind es aber nicht)
- wird kein Zeichen gefunden, Rückgabe undefined, während charAt() leere Zeichenfolge zurückgibt
- ist schreibgeschützt: str[0] = "A" gibt keinen Fehler aus (funktioniert aber nicht!)

Konvertieren zu Datentyp string

String()

globale Methode String() konvertiert Zahlen in Strings

kann auf jede Art von Zahlen, Literale, Variablen oder Ausdrücke verwendet werden:

String(x)

Methode toString()

x.toString()

Referenz

Funktion	Zweck	Beispielaufruf für let wort = "abcde"	Rückgabewert
charAt()	Ermittelt das Zeichen an der angegebenen Stelle	wort.charAt(1)	b
includes()	Prüft, ob der String den angegebenen Teilstring enthält	wort.includes("bc")	true
startsWith()	Prüft, ob der String mit dem angegebenen Teilstring beginnt	wort.startsWith("ab")	true
endsWith()	Prüft, ob der String mit dem angegebenen Teilstring endet	wort.endsWith("de")	true
indexOf()	Ermittelt die Position des angegebenen Teilstrings innerhalb des Strings	wort.indexOf("cd")	2
lastIndexOf()	Ermittelt die Position des angegebenen Teilstrings innerhalb des Strings (Suche beginnt von hinten)	wort.lastIndexOf("de")	3
repeat()	Wiederholt den String so oft wie angegeben	wort.repeat(2)	abcdeabcde
slice()	Extrahiert einen Teilstring an der angegebenen Position bis zu einer weiteren Position	wort.slice (1, 3)	bc
substring()	Extrahiert einen Teilstring an der angegebenen Position bis zu einer weiteren Position	wort.substring(1, 3)	bc
toLowerCase()	Gibt den String in Kleinschreibung zurück	"AbCdE".toLowerCase()	abcde
toUpperCase()	Gibt den String in Großschreibung zurück	"AbCdE".toUpperCase()	ABCDE
trim()	Entfernt Leerzeichen am Anfang und am Ende	" abc ".trim()	abc

Komplette string-Referenz:
https://www.w3schools.com/jsref/jsref_obj_string.asp

Switch-Anweisung

Syntax:

```
switch (Ausdruck) {
  case Wert1:
    //Programmblock
    break;
  //usw.
  default:
    //Programmblock
}
```

alles dreht sich um Ausdruck - normalerweise Variable

hat Ausdruck Wert1, wird erster Programmblock ausgeführt, bei Wert2 zweiter Programmblock usw.

default-Abschnitt wird ausgeführt, wenn keiner der vorherigen Werte zutrifft und ist optional

Programmblock sollte mit Kommando break abgeschlossen werden

sonst führt JS-Interpreter alle Anweisungen bis zum nächsten break oder Ende des switch-Blocks aus

case prüft auf identisch (===) – d.h. Datentyp beachten!

For-Schleifen

Schleife führt Anweisung eine bestimmte Anzahl von Malen aus

Syntax:

```
for (Initialisierung; Bedingung; Befehlsfolge) {
  //Anweisungen
}
```

for-Schleife hat drei Parameter:

- Initialisierung:
  oft läuft Zählvariable mit, die Anzahl der Wiederholungen zählt - kann hier initialisiert werden
  bei Initialisierung mehrerer Variablen, Anweisungen mit Komma voneinander getrennt
- Bedingung:
  for-Schleife wird so lange ausgeführt, bis Bedingung nicht mehr erfüllt
- Befehlsfolge:
  nach jedem Durchlauf wird Befehlsfolge ausgeführt (meist ein Befehl, wenn mehrere dann Trennung durch Komma)
  um Schleife zu beenden, hier Befehle ausführen, die nach bestimmter Anzahl von Durchläufen Bedingung (den zweiten Parameter) nicht mehr erfüllbar machen

geschweifte Klammern um Anweisungsblock dann zwingend, wenn Block aus mehreren Befehlen besteht

wenn nur ein Befehl, können geschweiften Klammern entfallen
dann bitte Code einrücken damit er übersichtlich und lesbar

generell empfehlenswert, immer geschweifte Klammern zu verwenden

oft wird Zählvariable direkt in Schleife verwendet

kann auch mehrere Zählvariablen verwenden, die durch Kommata voneinander getrennt
wird nur selten genutzt

Do-while-Schleife

Einsatz, wenn unklar, wie oft Anweisungsblock hintereinander ausgeführt werden soll
darum Block so lange ausführen, bis Bedingung nicht mehr erfüllt

Syntax:

```
do {
  //Anweisungsblock
} while (Bedingung);
```

Anweisungsblock wird ausgeführt, dann wird Bedingung überprüft

ist Bedingung erfüllt, wird Block erneut ausgeführt und dann erneut geprüft
sonst wird Schleife verlassen

Achtung: Schleife wird auf jeden Fall einmal ausgeführt!

While-Schleife

weitere Form der Schleifen - while-Schleifen (ohne do)

```
while (Bedingung) {
  //Anweisungsblock
}
```

Bedingung wird vor Durchlauf des Anweisungsblocks überprüft
sehr nützlich, da Anweisungsblock nicht ausgeführt, wenn Bedingung von Anfang an nicht erfüllt:

Schleifensteuerung

Schleife muss nicht so oft durchlaufen werden, wie vorgesehen
break, veranlasst sofortiges Verlassen der aktuellen Schleife oder switch-Anweisung
sorgt dafür, dass Programm mit Anweisung weiterläuft, die auf beendete Anweisung folgt
kann innerhalb von switch, while und for-Schleifen verwendet werden

continue bei Schleifen - vor allem bei for-Schleifen - zum nächsten Schleifendurchlauf springen
(beispielsweise, wenn man weiß, dass Durchlauf nicht gewünschtes Ergebnis bringt und Rest aus Effizienzgründen nicht ausführen lassen will)
beendet Ausführung von Anweisungen im aktuellen Durchlauf der aktuellen oder benannten Schleife und setzt Schleife mit nächster Iteration fort

- In while Schleife wird zur Bedingung gesprungen
- In for Schleife wird zur Schlussanweisung gesprungen

Funktionen

haben schon vordefinierte Funktionen verwendet – prompt, Number
vordefinierte Funktion bestehen aus vielen Einzelanweisungen – sind meist nicht in JS geschrieben

Funktionen sind Bestandteil der JS-Engine oder der Umgebung (z. B. eines Browsers) und deswegen oft in C, C++ oder Java implementiert - native (»einheimisch«) Funktionen oder built-in (eingebaut)

Funktion = Programmblock, der nicht sofort ausgeführt wird, explizit auf- bzw. abgerufen werden kann

Funktion führt Befehle aus (Prozedur), oder gibt Wert zurück

Funktionsdeklaration

Funktionsdeklaration (Funktionsdefinition, Funktionsanweisung) besteht aus Schlüsselwort function gefolgt von:

- Name der Funktion
- Liste von Parametern - in runden Klammern und durch Kommas getrennt
- JavaScript Anweisungen, die durch Funktion definiert werden und in geschweiften Klammern stehen
- Rückgabewert return

```
function Funktionsname() {
  //Programmblock
  return Wert;
}

Funktionsname();
```

der Funktion wird Wert übergeben
Werte, die an Funktion übergeben und innerhalb der Funktion geändert werden, ändert Wert nur innerhalb der Funktion - nicht global oder in aufrufender Funktion
müssen daher bei Bedarf zurückgegeben werden