# String Manipulation
# (Zeichenketten Beeinflussung)

Strings (Zeichenketten) können auf mehrere Arten manipuliert (beeinflusst) werden.

Alle Zeichen eines Strings haben einen Index, dieser beginnt bei 0 für das erste Zeichen.
Das zweite Zeichen hat dann den Index 1, das dritte Zeichen den Index 2 usw.

Auch ein Leerzeichen ist ein Zeichen in einem String.

Man kann einen String auch vom Ende ansteuern, dann haben ist der Index negativ, und beginnt bei -1 (es kann nicht bei -0 beginnen, da mathematisch -0=0 ist, somit hätten das erste Zeichen von vorne gezählt und das letzte Zeichen von hinten gezählt, den gleichen Index!)

Alle nachfolgenden Methoden ändern den ursprünglichen String nicht!

## .length

Die Anzahl der einzelnen Zeichen des Strings wird mit .length festgestellt. Dabei wird jedes einzelne Zeichen gezählt.

```
var emailString = "Max.Mustermann@Musterwelt.de";
var stringLaenge = emailString.length;
```

Hier wird die Variable emailString mit dem Wert “Max.Mustermann@Musterwelt.de“ initialisiert. Danach wird die Variable stringLaenge initialisiert und ihr der Wert der Länge von emailString zugewiesen.

```
console.log(stringLaenge);
// gibt auf der Console aus:
28
```

## .indexOf()

Mit .indexOf() wird der String nach einzelnen Zeichen oder Zeichenabfolgen durchsucht. Der Suchparameter wird in den Klammen als String angegeben. Das Ergebnis ist die Indexnummer im String, ab dem der Suchparameter als erstes vorkommt. Dabei wird case-sensitive gesucht, das heißt die Groß- und Kleinschreibung im String muss mit der Groß- und Kleinschreibung im Suchparameter übereinstimmen.

```
var emailString = "Max.Mustermann@Musterwelt.de";
var strFirstIndex = emailString.indexOf("Muster");
```

Hier wird emailString nach der Zeichenabfolge “Muster“ durchsucht und gibt die betreffende Indexzahl an, wo die Zeichenabfolge das erste Mal auftritt (hier also nach “Max.“)

```
console.log(strFirstIndex);

// gibt auf der Console aus:
4
```

Wenn nach dem Suchparameter noch eine Zahl für eine Indexnummer angeben wird, dann wird der String ab diesem Index durchsucht.

```
var emailString = "Max.Mustermann@Musterwelt.de";
var strAbIndex = emailString.indexOf("Muster", 10);
```

Hier wird  emailString ab der Indexnummer 10 nach der Zeichenabfolge “Muster“ durchsucht und gibt die betreffende Indexzahl an, wo die Zeichenabfolge das erste Mal ab Indexnummer 10 auftritt (hier also nach “@“)

```
console.log(strAbIndex);

// gibt auf der Console aus:
15
```

**.lastIndexOf()**

Mit .lastIndexOf() wird der String ebenso nach einzelnen Zeichen oder Zeichenabfolgen durchsucht. Der Suchparameter wird in den Klammen als String angegeben.
Das Ergebnis ist die Indexnummer im String, ab dem der Suchparameter als letztes gezeigt wird.

```
var emailString = "Max.Mustermann@Musterwelt.de";
var strLastIndex = emailString.lastIndexOf("Muster");
```

Hier wird emailString nach der Zeichenabfolge “Muster“ durchsucht und gibt die betreffende Indexzahl an, wo die Zeichenabfolge das letzte Mal auftritt (hier also nach “@“)

```
console.log(strLastIndex);

// gibt auf der Console aus:
15
```

**.search()**

Mit .search() wird der String ebenso wie mit .indexOf() nach einem Parameter durchsucht,
diese beiden Befehle zeigen deas gleiche Ergebnis.

```
var emailString = "Max.Mustermann@Musterwelt.de";
var strSearch = emailString.search("Muster");
```

Hier wird emailString nach der Zeichenabfolge “Muster“ durchsucht und gibt die betreffende Indexzahl an, wo die Zeichenabfolge das erste Mal auftritt (hier also nach “Max.“)

```
console.log(strSearch);
```

```
// gibt auf der Console aus:
4
```

Allerdings lässt sich .searchOf immer nur die erste-Indexzahl bestimmen, ein Ergänzung ab welchem Index gesucht werden soll, ist nicht möglich

**Nicht-Case-Sensitive Suche mit .search(/…./i)**

Ergänzt man .search() mit dem Suchparameter und danach mit „i“, dann wird case-insensitiv gesucht, d.h. Groß- und Kleinschreibung wird bei der Suche nicht berücksichtigt:
Dabei ändert sich der Syntax von .search(), der Suchparamter wird nicht in Anführungszeichen sondern nach einem Slash genannt und danach ein Slash und i

```
var caseSensitiveString = "Pfarrer BrAuN hat eine braune Aktentasche";
var strSearchCaseSensitive = caseSensitiveString.search(/braun/i);
```

Durchsucht caseSensitiveString unabhängig von Groß- und Kleinschreibung nach der Zeichenabfolge “braun“ und gibt die betreffende Indexzahl an, wo die Zeichenabfolge das erste Mal auftritt (hier also nach “Pfarrer „)

```
console.log(strSearchCaseSensitive);
```

```
// Gibt auf der Console aus:
8
```

**.charAt()**

Mit .charAt() kann ein Zeichen des Strings mit Hilfe der Indexnummer herausgefunden werden.

Allerdings funktioniert dies nur mit positiven Indexnummern!

```
var emailString = "Max.Mustermann@Musterwelt.de";
var zeichenAtVierzehn = emailString.charAt(14);
```

Ermittelt das Zeichen an Indexnummer 14

```
console.log(zeichenAtVierzehn);
```

// Gibt auf der Console aus:
@

**.slice()**

Mit .slice() wird der String indexbasiert kopiert.
Wird dabei nur ein Parameter (Indexnummer) angegeben, dann wird der String ab dieser Indexnummer kopiert.
Werden zwei Parameter (Indexnummer, Indexnummer) angeben, so wird ab der ersten Indexnummer kopiert bis die zweite Indexnummer erreicht wird, ab der zweiten Indexnummer findet kein Kopieren mehr statt ( d.h. das Zeichen bei der zweiten Indexzahl wird nicht mitkopiert).

```
var emailString = "Max.Mustermann@Musterwelt.de";
var sliceStringEINPara = emailString.slice(4);
```

Kopiert ab der Indexnummer 4

```
console.log(sliceStringEINPara);
```

// Gibt auf der Console aus:
Mustermann@Musterwelt.de

```
var emailString = "Max.Mustermann@Musterwelt.de";
var sliceStringZWEIPara = emailString.slice(4, 21);
```

Kopiert ab der Indexnummer 4 und stoppt das Kopieren bei Indexnummer 21

```
console.log(sliceStringZWEIPara);
```

// Gibt auf der Console aus:
Mustermann@Muster

Bei .slice() kann auch mit negativen Indexnummern kopiert werden:

Bei Angabe einer negativen Index-Nummer wird der String von hinten gezählt ab der Indexnummer bis zum Ende des String (nicht bis zum Anfang!!!) kopiert.

```
var emailString = "Max.Mustermann@Musterwelt.de";
var sliceStringEINParaNEG = emailString.slice(-3);
```

Kopiert ab der Indexnummer -3 bis zum Ende des Strings

```
console.log(sliceStringEINParaNEG);
```

// Gibt auf der Console aus:
.de

```
var emailString = "Max.Mustermann@Musterwelt.de";
var sliceStringZWEIParaNEG = emailString.slice(-13, -3);
```

Kopiert ab der Indexnummer -12 bis zum Ende des Strings

```
console.log(sliceStringZWEIParaNEG);
```

**.substring()**

Mit Substring wird aus einem String indexbasiert herausgeschnitten.
Das Ergebnis ist identisch mit .slice().

Bei einem Parameter wird ab dem diesem bis zum Ende des Strings herausgeschnitten.

```
var emailString = "Max.Mustermann@Musterwelt.de";
var substringStringEINPara = emailString.substr(4);
```

Schneidet ab Indexnummer 4 heraus (und kopiert somit).

```
console.log(substringStringEINPara);
```

// Gibt auf der Console aus:
Mustermann@Musterwelt.de

Bei der Angabe von 2 Parametern ist der erste Parameter der Startpunkt, der zweite Parameter die Länge des heraus zu schneidenen Teils.

```
var emailString = "Max.Mustermann@Musterwelt.de";
var substringStringZWEIPara = emailString.substr(4, 10);
```

Schneidet ab Indexnummer 4  10 Zeichen heraus (und kopiert sie somit).

```
console.log(substringStringZWEIPara);
```

```
// Gibt auf der Console aus:
Mustermann
```

Der erste Parameter (der Index ab dem kopiert wird) darf negativ sein, nicht jedoch die Länge!

.slice() vs .substring()

Beide Befehle beeinflussen den ursprünglichen String nicht.
Bei .substring() ist der zweite Parameter die Länge des heraus zu schneidenen String, bei .slice() ist der zweite Parameter der zweite Index, ab dem das Kopieren beendet wird.

**.replace()**

Mit .replace() kann im String nach einer Zeichenabfolge gesucht werden und diese ersetzt werden. Der erste Parameter ist die zu ersetzende Zeichenabfolge, der zweite Parameter die Ersatz-Zeichenfolge. Dabei wird nur das erste Erscheinen der Zeichenabfolge ersetzt.

```
var wrongText = "Meine Lieglingsfarbe ist braun, deshalb sind alle meine Kleider braun";
var replacedText = wrongText.replace("braun", "gelb");
```

Hier wird das erste braun im String durch gelb ersetzt.

```
console.log(replacedText);
```

```
// Gibt auf der Console aus:
Meine Lieglingsfarbe ist gelb, deshalb sind alle meine Kleider braun
```

Um im String alle Zeichenabfolgen die dem Suchparameter entsprechen, muss „global“ im String ersetzt werden, dazu ändert sich der Syntax von .replace

```
var wrongText = "Meine Lieglingsfarbe ist braun, deshalb sind alle meine Kleider braun";
var totallyReplacedText = wrongText.replace(/braun/g, "gelb");
```

Sucht im ganzen String nach braun und ersetzt es durch gelb

```
console.log(totallyReplacedText);
```

// Gibt auf der Console aus:
Meine Lieglingsfarbe ist gelb, deshalb sind alle meine Kleider gelb

**.toUpperCase()**

Wandelt alle Zeichen in Großbuchstaben um.

```
var kleineZeichen = "abcde";
var inGrossUmgewandelt = kleineZeichen.toUpperCase();

console.log(inGrossUmgewandelt);
```

// Gibt auf der Console aus:
ABCDE

**.toLowerCase()**

Wandelt alle Zeichen in Kleinbuchstaben um.

```
var grosseZeichen = "VWXYZ";
var inKleinUmgewandelt = grosseZeichen.toLowerCase();

console.log(inKleinUmgewandelt);
```

// Gibt auf der Console aus:
vwxyz

**.concat()**

Strings können mit .concat() zusammen geführt werden. Der zweite String ist hier eine Variable und wird als Parameter in der Methode .concat an den ersten String angefügt ohne Leerzeichen.

.

```
var stringEins = "Guten";
var stringZwei = "Morgen";
var concatedString = stringEins.concat(stringZwei);

console.log(concatedString);
```

// Gibt auf der Console aus:
GutenMorgen

Es können mehrere Parameter in der Methode .concat() übergeben werden.

```
var stringEins = "Guten";
var stringZwei = "Morgen";
var concatedStringVieleParam = stringEins.concat(" ", stringZwei, " meine Herrschaften.");

console.log(concatedStringVieleParam);
```

// Gibt auf der Console aus:
Guten Morgen meine Herrschaften.

**+ Operator**

Eine andere Möglichkeit, Strings zusammen zu führen ist die Addition mit dem + Operator. Dies nennt ebenso man „string concatenation“
Hierbei entsteht ein neuer String aus der Addition eines String mit einem String oder mit einer Variablen beliebigen Typs. Das Ergebnis ist immer ein String!

```
var string_1Var = "Dass Jens "
var numberVar = 8;
var string_2Var = " Paar Schuhe hat, ist ";
var booleanVar = true;
var neuerString = string_1Var + numberVar + string_2Var + booleanVar;

console.log(neuerString);
```

// Gibt auf der Console aus:
Dass Jens 8 Paar Schuhe hat, ist true

**.split()**

Mit .split() wird ein String zerlegt, die einzelnen Teile sind Strings welche wiederum Items eines Arrays sind.
Dabei muss als Parameter der Selector, anhand welcher gesplittet werden soll, angeben werden. Wird als Parameter (“”) angegeben, so wird nach jedem einzelnen Zeichen gesplittet.

```
var alphabet = "abcdefghijklmnopqrstuvwxyz";
var alphabetAlsArray = alphabet.split("");

console.log(alphabetAlsArray);

// Gibt auf der Console aus:
 (26) ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"]
```

```
var aBisKMitKomma = "a,b,c,d,e,f,g,h,i,j,k";
var aBisKAlsArray = aBisKMitKomma.split(",");
```

Hier wird der String an den Stellen des Kommas gesplittet und die einzelnen Teile werden als einzelne Strings als Item eines Arrays gespeichert.

```
console.log(aBisKAlsArray);

// Gibt auf der Console aus:

11) ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k"]
```

```
var aBisKMitKomma = "a,b,c,d,e,f,g,h,i,j,k";
var aBisKAlsArrayGekürzt = aBisKMitKomma.split(",", 5);
```

Hier wird der String an den Stellen des Kommas gesplittet und die einzelnen Teile werden als einzelne Strings als Item eines Arrays gespeichert, der Array wird auf 5 Items begrenzt.

```
console.log(aBisKAlsArrayGekürzt);

// Gibt auf der Console aus:
5) ["a", "b", "c", "d", "e"]
```

**Questions? You‘re welcome!**

holger.zerbe@web.de