TypeScript - String

String is a primitive data type that is used to store text data.

String values are surrounded by single quotation marks or double quotation marks.

```JavaScript
let employeeName:string = 'John Smith';

//OR

let employeeName:string = "John Smith";
```

Template String

Since TypeScript version 1.4, TypeScript has included support for ES6 Template strings. Template strings are used to embed expressions into strings.

```JavaScript
let employeeName:string = "Harry";

let employeeDept:string = "Admin";

// Pre-ES6

let employeeDesc1: string = employeeName + " works in the " + employeeDept + " department.";

// Post-ES6

let employeeDesc2: string = `${employeeName} works in the ${employeeDept} department.`;

console.log(employeeDesc1);//Harry works in the Admin department.

console.log(employeeDesc2);//Harry works in the Admin department.
```

Here, instead of writing a string that is a combination of text and variables with concatenations, we can use a single statement with back-ticks `. The variable values are written as ${}.

Using template strings, it is easier to embed expressions and also less tedious to write long text-based strings.

String methods

Method	Description
charAt()	Returns the character at the given index
concat()	Returns a combination of the two or more specified strings
indexOf()	Returns an index of first occurrence of the specified substring from a string (-1 if not found)
replace()	Replaces the matched substring with a new substring
split()	Splits the string into substrings and returns an array
toUpperCase()	Converts all the characters of the string into upper case

toLowerCase()	Converts all the characters of the string into lower case
charCodeAt()	Returns a number that is the UTF-16 code unit value at the given index
codePointAt()	Returns a nonnegative integer Number that is the code point value of the UTF-16 encoded code point starting at the specified index
includes()	Checks whether a string includes another string
endsWith()	Checks whether a string ends with another string
LastIndexOf()	Returns the index of last occurrence of value in the string
localeCompare()	Checks whether a string comes before, after or is the same as the given string
match()	Matches a regular expression against the given string

normalize()	Returns the Unicode Normalization Form of the given string.
padEnd()	Pads the end of the current string with the given string
padStart()	Pads the beginning of the current string with given string
repeat()	Returns a string consisting of the elements of the object repeated in the given times.
search()	Searches for a match between a regular expression and a string
slice()	Returns a section of a string
startsWith()	Checks whether a string starts with another string
substr()	Returns a string beginning at the specified location and of the given characters

substring()	Returns a string between the two given indexes
toLocaleLowerCase()	Returns a lower case string while respecting current locale
toLocaleUpperCase()	Returns an upper case string while respecting current locale
trim()	Trims the white space from beginning and end of string
trimLeft()	Trims the white space from left side of the string
trimRight()	Trims the white space from right side of the string

charAt()

The charAt() method returns a character at the specified index from a string.

Signature:
character = str.charAt(index)

This method takes one number argument index and returns the character at the given index in the string.

Example: charAt()

```JavaScript
let str: string = 'Hello TypeScript';
str.charAt(0); // returns 'H'
str.charAt(2); // returns 'l'
"Hello World".charAt(2); returns 'l'
```

concat()

The concat() method concatenates two or more specified strings.

Signature:
str.concat(string2[, string3, ..., stringN])

This method takes two or more arguments of strings and returns a concatenation of the given strings.

Example: concat()

```
Unset
let str1: string = 'Hello';
let str2: string = 'TypeScript';
str1.concat(str2); // returns 'HelloTypeScript'
str1.concat(' ', str2); // returns 'Hello TypeScript'
str1.concat(' Mr. ', 'Bond'); // returns 'Hello Mr. Bond'
```

indexOf()

The indexOf() method returns an index of first occurrence of the specified sub string from a string. The index starts from 0. It returns -1 if not found. the indexOf() method search is case-sensitive, so 't' and 'T' are different.

Optionally, you can specify an index as a second parameter to define where the searching should start from.

Signature:
str.indexOf(searchValue[, fromIndex])

This method takes two arguments, the search string and an optional index number denoting the location the searching should start.

Example: indexOf()

```
Unset
let str: string = 'TypeScript';

str.indexOf('T'); // returns 0
str.indexOf('p'); // returns 2
str.indexOf('e'); // returns 3
```

```
str.indexOf('T', 1); // returns -1
str.indexOf('t', 1); // returns 9
```

replace()

The replace() method replaces the matched substring with the specified string. The regular expression can also be used for searching.

Signature:
str.replace(regexp|substr, newSubstr|function)

This method takes two arguments: a regex of string to be found, and the new string that will replace the existing substring.

Example: replace()

```
Unset
let str1: string = 'Hello Javascript';
let str2: string = 'TypeScript';

str1.replace('Java', 'Type'); // returns 'Hello TypeScript'
str1.replace('JavaScript', str2); // returns 'Hello TypeScript'
str1.replace(/Hello/gi, 'Hi'); // returns 'Hi TypeScript'
```

split()

The split() method splits a string into substrings based on a specified separator character and returns an array of substrings.

Signature:
str.split([separator[, limit]])

This method takes two arguments: a separator string and an optional limit specifying the number of entries to be found.

Example: split()

```
Unset
let str1: string = 'Apple, Banana, Orange';
let str2: string = ',';

str1.split(str2) // returns [ 'Apple', ' Banana', ' Orange' ]
str1.split(',') // returns [ 'Apple', ' Banana', ' Orange' ]
str1.split(',', 2) // returns [ 'Apple', ' Banana' ]
str1.split(',', 1) // returns [ 'Apple']
```

toUpperCase()

The toUpperCase() method returns an upper case representation of the string it is called on.

Example: toUpperCase()

```
Unset
let str: string = 'Hello Typescript';
str.toUpperCase(); // returns 'HELLO TYPESCRIPT'
'hello typescript'.toUpperCase(); // returns 'HELLO TYPESCRIPT'
```

toLowerCase()

The toLowerCase() method returns a lower case representation of the string it is called on.

Example: toLowerCase()

```
Unset
let str: string = 'Hello Typescript';
str.toLowerCase(); // returns hello typescript
'HELLO TYPESCRIPT'.toLowerCase(); // returns hello typescript
```