

String Methods

<code>str.charAt(index)</code>	Return character in string str at specified index
<code>str.toLowerCaseCase()</code>	Convert string str to lower / upper case
<code>str.toLocaleLowerCase()</code>	Convert chars in string str to lower / upper case while respecting current locale
<code>str.[last]indexOf(substr)</code>	Return first / last index within string str of substring substr
<code>str.split(separator)</code>	Split string str into an array of substrings separated by param separator
<code>str.trim[Start][End]()</code>	Trim whitespace from beginning and / or end of string str
<code>str.codePointAt(index)</code>	Return non-negative int from string str that is the UTF-16 encoded code point at given index
<code>str1.includes(str2)</code>	Return true if str2 is found in string str1
<code>str1.starts[endsWith]With(str2)</code>	Return true if string str1 starts / ends with string str2
<code>str.normalize()</code>	Return Unicode Normalization Form of string str
<code>str.repeat(int)</code>	Return string repeated int times of string str
<code>str[@@iterator]()</code>	Return a new Iterator that iterates over the code points of string str, returning each code point as String value
<code>str.charCodeAt(index)</code>	Return number indicating Unicode value of char at given index of string str
<code>str1.concat(str2)</code>	Combine text of strings str1 and str2 and return a new string
<code>str.slice(start, end)</code>	Extract a section of string str from start to end

String Methods (cont)

<code>str.substr(start, length)</code>	Deprecated. Return characters in string str from start having length
<code>str.substring(index1, index2)</code>	Return subset of string str between index1 and index2
<code>str.trimLeft[Right]()</code>	Trim whitespace from left / right side of string str
<code>str1.localeCompare(str2)</code>	Return -1, 0, or 1 indicating if string str1 is less than, equal to, or greater than str2
<code>str.match(regex)</code>	Match a regular expression regex against string str
<code>str1.replace(regex, str2)</code>	Replace matched regex elements in string str1 with string str2
<code>str.search(regex)</code>	Return position of search for a match between regex and string str
<code>str.length</code>	Return length of string str
<code>str.toWellFormed()</code>	Returns a string where all lone surrogates of this string are replaced with the Unicode replacement character U+FFFD.

Object methods

<code>Object.assign(target, ...sources)</code>	copies properties from one or more source objects to target object
<code>Object.create(proto, [propertiesObject])</code>	creates new object, using an existing object as the prototype
<code>Object.defineProperty[ies](obj, prop, descriptor)</code>	defines new or modifies existing property
<code>Object.entries(obj)</code>	returns array of object's [key, value] pairs
<code>Object.freeze(obj)</code>	freezes an object, which then can no longer be changed
<code>Object.fromEntries()</code>	transforms a list of key-value pairs into an object



Object methods (cont)

<code>Object.getOwnPropertyDescriptors(obj, prop)</code>	returns a property descriptor / all own property descriptors for an own property
<code>Object.getOwnPropertyNames(obj)</code>	returns array of all properties
<code>Object.getOwnPropertySymbols(obj)</code>	array of all symbol properties
<code>Object.getPrototypeOf(obj)</code>	returns the prototype
<code>Object.is(value1, value2)</code>	determines whether two values are the same value
<code>Object.isExtensible(obj)</code>	determines whether an object can have new properties added to it
<code>Object.isFrozen(obj)</code> <code>Object.isSealed(obj)</code>	determines if an object is frozen / sealed
<code>Object.keys(obj)</code>	returns array of object's enumerable property names
<code>Object.preventExtensions(obj)</code>	prevents new properties from being added to an object
<code>obj.hasOwnProperty(prop)</code>	returns boolean indicating whether object has the specified property
<code>prototypeObj.isPrototypeOf(object)</code>	checks if object exists in another object's prototype chain
<code>obj.propertyIsEnumerable(prop)</code>	checks whether the specified property is enumerable and is the object's own property
<code>obj.toString()</code>	returns a string representing the object
<code>Object.seal(obj)</code>	prevents new properties from being added and marks all existing properties as non-configurable
<code>Object.values(obj)</code>	returns array of object's own enumerable property values

Array methods

<code>a1.concat(a2)</code>	Return new array by joining arrays a1 and a2 together
<code>a1.join(separator)</code>	Join all elements of array a1 into a string separated by separator arg
<code>a1.slice(start, end)</code>	Extract a section from start to end of array a1 and return a new array
<code>a1.lastIndexOf(obj)</code>	Return first / last index of obj within array a1
<code>a1.forEach(fn)</code>	Calls function fn for each element in the array a1
<code>a1.every(fn)</code>	Return true if every element in array a1 satisfies provided testing function fn
<code>a1.some(fn)</code>	Return true if at least one element in array a1 satisfies provided testing function fn
<code>a1.filter(fn)</code>	Create a new array with all elements of array a1 which pass the filtering function fn
<code>a1.map(fn)</code>	Create a new array with results of calling function fn on every element in array a1
<code>a1.reduceRight(fn)</code>	Apply a function fn against an accumulator and each value (from left to right / right to left) of the array as to reduce it to a single value
<code>a1.pop()</code>	Remove and return last element from array a1
<code>a1.push(obj)</code>	Add obj to end of array a1 and return new length
<code>a1.reverse()</code> / <code>a1.toReversed()</code>	Reverse order of elements of array a1 in place / non-destructive



Array methods (cont)

<code>a1.sort()</code> / <code>a1.toSorted()</code>	Sort the elements of array a1 in place / non-destructive
<code>a1.splice(start, deleteCount, items)</code> / <code>str.toSpliced(...)</code>	Change content of array a1 by removing existing elements and/or adding new elements in place / non-destructive
<code>a1.unshift(obj)</code>	Add obj to start of array a1 and return new length
<code>a1.toString()</code>	Return a string representing array a1 and its elements (same as <code>a1.join(',')</code>)
<code>a1.toLocaleString()</code>	Return a localized string representing array a1 and its elements (similar to <code>a1.join(',')</code>)
<code>Array.isArray(var)</code>	Return true if var is an array a1.length
<code>a1.length</code>	Return length of a1
<code>a1.with()</code>	Non-destructive (returns new array) version of setting value with square brackets
<code>a1.find[Last](item)</code>	Returns the first / last found item
<code>a1.find[L-ast]Index(item)</code>	Returns the index of the first / last found item



By **flight**
cheatography.com/flight/

Published 17th November, 2022.
Last updated 8th January, 2025.
Page 3 of 3.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>