w3schools.com

THE WORLD'S LARGEST WEB DEVELOPER SITE

JavaScript String Methods

❮ Previous Next ❯

String methods help you to work with strings.

String Methods and Properties

Primitive values, like "John Doe", cannot have properties or methods (because they are not objects).

But with JavaScript, methods and properties are also available to primitive values, because JavaScript treats primitive values as objects when executing methods and properties.

String Length

The **length** property returns the length of a string:

Example

```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var sln = txt.length;
```

Try it Yourself »

Finding a String in a String

The **indexOf()** method returns the index of (the position of) the **first** occurrence of a specified text in a string:

Example

```
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate");
```

The **lastIndexOf()** method returns the index of the **last** occurrence of a specified text in a string:

Example

```
var str = "Please locate where 'locate' occurs!";
var pos = str.lastIndexOf("locate");
```

Try it Yourself »

Both the indexOf(), and the lastIndexOf() methods return -1 if the text is not found.

JavaScript counts positions from zero.
0 is the first position in a string, 1 is the second, 2 is the third ...

Both methods accept a second parameter as the starting position for the search:

Example

```
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate",15);
```

Try it Yourself »

Searching for a String in a String

The **search()** method searches a string for a specified value and returns the position of the match:

Example

```
var str = "Please locate where 'locate' occurs!";
var pos = str.search("locate");
```

Try it Yourself »

Did You Notice?

The two methods, indexOf() and search(), are **equal?**

They accept the same arguments (parameters), and return the same value?

The two methods are **NOT** equal. These are the differences:

You will learn more about regular expressions in a later chapter.

Extracting String Parts

There are 3 methods for extracting a part of a string:

- slice(start, end)
- substring(start, end)
- substr(start, length)

The slice() Method

slice() extracts a part of a string and returns the extracted part in a new string.

The method takes 2 parameters: the starting index (position), and the ending index (position).

This example slices out a portion of a string from position 7 to position 13:

Example

```
var str = "Apple, Banana, Kiwi";
var res = str.slice(7, 13);
```

The result of res will be:

```
Banana
```

Try it Yourself »

If a parameter is negative, the position is counted from the end of the string.

This example slices out a portion of a string from position -12 to position -6:

Example

```
var str = "Apple, Banana, Kiwi";
var res = str.slice(-12, -6);
```

The result of res will be:

```
Banana
```

Try it Yourself »

Example

```
var res = str.slice(7);
```

Try it Yourself »

or, counting from the end:

Example

```
var res = str.slice(-12);
```

Try it Yourself »

Negative positions do not work in Internet Explorer 8 and earlier.

The substring() Method

substring() is similar to slice().

The difference is that substring() cannot accept negative indexes.

Example

```
var str = "Apple, Banana, Kiwi";
var res = str.substring(7, 13);
```

The result of *res* will be:

```
Banana
```

Try it Yourself »

If you omit the second parameter, substring() will slice out the rest of the string.

The substr() Method

substr() is similar to slice().

The difference is that the second parameter specifies the **length** of the extracted part.

```
var str = "Apple, Banana, Kiwi";
var res = str.substr(7, 6);
```

The result of res will be:

```
Banana
```

Try it Yourself »

If the first parameter is negative, the position counts from the end of the string.

The second parameter can not be negative, because it defines the length.

If you omit the second parameter, substr() will slice out the rest of the string.

Replacing String Content

The **replace()** method replaces a specified value with another value in a string:

Example

```
str = "Please visit Microsoft!";
var n = str.replace("Microsoft", "W3Schools");
```

Try it Yourself »

The replace() method does not change the string it is called on. It returns a new string.

By default, the replace() function replaces **only the first** match:

Example

```
str = "Please visit Microsoft and Microsoft!";
var n = str.replace("Microsoft", "W3Schools");
```

Try it Yourself »

By default, the replace() function is case sensitive. Writing MICROSOFT (with upper-case) will not work:

Example

```
str = "Please visit Microsoft!";
```

Try it Yourself »

To replace case insensitive, use a **regular expression** with an **/i** flag (insensitive):

Example

```
str = "Please visit Microsoft!";
var n = str.replace(/MICROSOFT/i, "W3Schools");
```

Try it Yourself »

Note that regular expressions are written without quotes.

To replace all matches, use a **regular expression** with a **/g** flag (global match):

Example

```
str = "Please visit Microsoft and Microsoft!";
var n = str.replace(/Microsoft/g, "W3Schools");
```

Try it Yourself »

You will learn a lot more about regular expressions in the chapter JavaScript Regular Expressions.

Converting to Upper and Lower Case

A string is converted to upper case with **toUpperCase()**:

Example

```
var text1 = "Hello World!";       // String
var text2 = text1.toUpperCase();  // text2 is text1 converted to upper
```

Try it Yourself »

A string is converted to lower case with **toLowerCase()**:

Example

```
var text1 = "Hello World!";       // String
```

Try it Yourself »

The concat() Method

concat() joins two or more strings:

Example

```
var text1 = "Hello";
var text2 = "World";
var text3 = text1.concat(" ", text2);
```

Try it Yourself »

The **concat()** method can be used instead of the plus operator. These two lines do the same:

Example

```
var text = "Hello" + " " + "World!";
var text = "Hello".concat(" ", "World!");
```

All string methods return a new string. They don't modify the original string.
Formally said: Strings are immutable: Strings cannot be changed, only replaced.

Extracting String Characters

There are 2 **safe** methods for extracting string characters:

- charAt(position)
- charCodeAt(position)

The charAt() Method

The **charAt()** method returns the character at a specified index (position) in a string:

Example

```
var str = "HELLO WORLD";
str.charAt(0);            // returns H
```

The charCodeAt() Method

The **charCodeAt()** method returns the unicode of the character at a specified index in a string:

Example

```
var str = "HELLO WORLD";

str.charCodeAt(0);         // returns 72
```

Try it Yourself »

Accessing a String as an Array is Unsafe

You might have seen code like this, accessing a string as an array:

```
var str = "HELLO WORLD";

str[0];                   // returns H
```

This is **unsafe** and **unpredictable:**

- It does not work in all browsers (not in IE5, IE6, IE7)
- It makes strings look like arrays (but they are not)
- str[0] = "H" does not give an error (but does not work)

If you want to read a string as an array, convert it to an array first.

Converting a String to an Array

A string can be converted to an array with the **split()** method:

Example

```
var txt = "a,b,c,d,e";   // String
txt.split(",");          // Split on commas
txt.split(" ");          // Split on spaces
txt.split("|");          // Split on pipe
```

Try it Yourself »

If the separator is "", the returned array will be an array of single characters:

Example

```
var txt = "Hello";       // String
txt.split("");           // Split in characters
```

Try it Yourself »

Complete String Reference

For a complete reference, go to our Complete JavaScript String Reference.

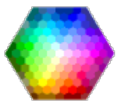
The reference contains descriptions and examples of all string properties and methods.

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 » Exercise 5 » Exercise 6 »

❮ Previous Next ❯

COLOR PICKER



HOW TO

Tabs
Dropdowns
Accordions
Convert Weights
Animated Buttons
Side Navigation
Top Navigation
Modal Boxes
Progress Bars
Parallax
Login Form
HTML Includes