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

Example

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

Try it Yourself »

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

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

Example

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

Try it Yourself »

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

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

Try it Yourself »

The `lastIndexOf()` methods searches backwards (from the end to the beginning), meaning: if the second parameter is `15`, the search starts at position 15, and searches to the beginning of the string.

Example

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

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

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

- The `search()` method cannot take a second start position argument.
- The `indexOf()` method cannot take powerful search values (regular expressions).

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 start position, and the end position (end not included).

This example slices out a portion of a string from position 7 to position 12 (13-1):

Example

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

The result of res will be:

```
Banana
```

Try it Yourself »

Remember: JavaScript counts positions from zero. First position is 0.

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 »

If you omit the second parameter, the method will slice out the rest of the string:

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.

Example

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

The result of res will be:

Try it Yourself »

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

Example

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

The result of res will be:

```
Banana, Kiwi
```

Try it Yourself »

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

Example

```
var str = "Apple, Banana, Kiwi";
var res = str.substr(-4);
```

The result of res will be:

```
Kiwi
```

Try it Yourself »

Replacing String Content

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

```
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()` method 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()` method is case sensitive. Writing MICROSOFT (with upper-case) will not work:

Example

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

Try it Yourself »

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

Example

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 »

Example

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

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.

String.trim()

The `trim()` method removes whitespace from both sides of a string:

Example

```
var str = "       Hello World!        ";
alert(str.trim());
```

Try it Yourself »

The `trim()` method is not supported in Internet Explorer 8 or lower.

If you need to support IE 8, you can use `replace()` with a regular expression instead:

Example

```
var str = "       Hello World!        ";
alert(str.replace(/^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g, ''));
```

Try it Yourself »

You can also use the replace solution above to add a trim function to the JavaScript `String.prototype`:

Example

```
if (!String.prototype.trim) {
  String.prototype.trim = function () {
    return this.replace(/^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g, '');
  };
}
```

Try it Yourself »

Extracting String Characters

There are 3 methods for extracting string characters:

- `charAt(position)`
- `charCodeAt(position)`
- Property access [ ]

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
```

Try it Yourself »

The charCodeAt() Method

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

The method returns a UTF-16 code (an integer between 0 and 65535).

Example

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

Try it Yourself »

Property Access

ECMAScript 5 (2009) allows property access [ ] on strings:

Example

```
var str = "HELLO WORLD";
str[0];                     // returns H
```

Try it Yourself »

Property access might be a little **unpredictable:**

- It does not work in Internet Explorer 7 or earlier
- It makes strings look like arrays (but they are not)
- If no character is found, [ ] returns undefined, while charAt() returns an empty string.
- It is read only. str[0] = "A" gives no error (but does not work!)

Example

```
var str = "HELLO WORLD";
str[0] = "A";               // Gives no error, but does not work
str[0];                     // returns H
```

Try it Yourself »

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 omitted, the returned array will contain the whole string in index [0].

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:

Find the position of the character `h` in the string `txt`.

```
var txt = "abcdefghijklm";
var pos = txt.          ;
```

Submit Answer »

Start the Exercise

‹ Previous Next ›

COLOR PICKER

HOW TO

Tabs
Dropdowns
Accordions
Side Navigation
Top Navigation
Modal Boxes
Progress Bars

HTML Includes
Google Maps
Range Sliders
Tooltips
Slideshow
Filter List
Sort List

SHARE

CERTIFICATES

HTML
CSS
JavaScript
SQL
Python
PHP
jQuery
Bootstrap
XML

Read More »

REPORT ERROR
PRINT PAGE
FORUM
ABOUT

Top References

Top Examples

Web Certificates

W3Schools is optimized for learning, testing, and training. Examples might be simplified to improve reading and basic understanding. Tutorials, references, and examples are constantly reviewed to avoid errors, but we cannot warrant full correctness of all content. While using this site, you agree to have read and accepted our terms of use, cookie and privacy policy. Copyright 1999-2020 by Refsnes Data. All Rights Reserved.
Powered by W3.CSS.