JavaScript String Methods

❮ Previous Next ❯

Basic String Methods

Javascript strings are primitive and immutable: All string methods produce a new string without altering the original string.

String length
String charAt()
String charCodeAt()
String at()
String [ ]
String slice()
String substring()
String substr()

String toUpperCase()
String toLowerCase()
String concat()
String trim()
String trimStart()
String trimEnd()
String padStart()
String padEnd()
String repeat()
String replace()
String replaceAll()
String split()

See Also:

String Search Methods
String Templates

JavaScript String Length

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

Example

```
let text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
let length = text.length;
```

Try it Yourself »

Extracting String Characters

There are 4 methods for extracting string characters:

- The `at(position)` Method
- The `charAt(position)` Method
- The `charCodeAt(position)` Method
- Using property access [] like in arrays

JavaScript String charAt()

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

Example

```
let text = "HELLO WORLD";
let char = text.charAt(0);
```

Try it Yourself »

JavaScript String charCodeAt()

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

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

Example

```
let text = "HELLO WORLD";
let char = text.charCodeAt(0);
```

Try it Yourself »

JavaScript String at()

ES2022 introduced the string method `at()` :

Examples

Get the third letter of name:

```
const name = "W3Schools";
let letter = name.at(2);
```

Try it Yourself »

Get the third letter of name:

```
const name = "W3Schools";
let letter = name[2];
```

Try it Yourself »

The `at()` method returns the character at a specified index (position) in a string.

The `at()` method is supported in all modern browsers since March

2022:

Note

The `at()` method is a new addition to JavaScript.

It allows the use of negative indexes while `charAt()` do not.

Now you can use `myString.at(-2)` instead of `charAt(myString.length-2)`.

Browser Support

`at()` is an ES2022 feature.

JavaScript 2022 (ES2022) is supported in all modern browsers since March 2023:

Chrome 94	Edge 94	Firefox 93	Safari 16.4	Opera 79
Sep 2021	Sep 2021	Oct 2021	Mar 2023	Oct 2021

Property Access [ ]

Example

```
let text = "HELLO WORLD";
let char = text[0];
```

Try it Yourself »

Note

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

- 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

```
let text = "HELLO WORLD";
text[0] = "A";
```

Try it Yourself »

Extracting String Parts

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

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

JavaScript String slice()

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

The method takes 2 parameters: start position, and end position (end not included).

Example

Slice out a portion of a string from position 7 to position 13:

```
let text = "Apple, Banana, Kiwi";
let part = text.slice(7, 13);
```

Try it Yourself »

Note

JavaScript counts positions from zero.

First position is 0.

Second position is 1.

Examples

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

```
let text = "Apple, Banana, Kiwi";
let part = text.slice(7);
```

Try it Yourself »

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

```
let text = "Apple, Banana, Kiwi";
let part = text.slice(-12);
```

Try it Yourself »

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

```
let text = "Apple, Banana, Kiwi";
let part = text.slice(-12, -6);
```

Try it Yourself »

JavaScript String substring()

`substring()` is similar to `slice()`.

The difference is that start and end values less than 0 are treated as 0 in `substring()`.

Example

```
let str = "Apple, Banana, Kiwi";
let part = str.substring(7, 13);
```

Try it Yourself »

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

JavaScript String substr()

`substr()` is similar to `slice()`.

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

Example

```
let str = "Apple, Banana, Kiwi";
let part = str.substr(7, 6);
```

Try it Yourself »

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

Example

```
let str = "Apple, Banana, Kiwi";
let part = str.substr(7);
```

Try it Yourself »

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

Example

```
let str = "Apple, Banana, Kiwi";
let part = str.substr(-4);
```

Try it Yourself »

Converting to Upper and Lower Case

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

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

JavaScript String toUpperCase()

Example

```
let text1 = "Hello World!";
let text2 = text1.toUpperCase();
```

Try it Yourself »

JavaScript String toLowerCase()

Example

```
let text1 = "Hello World!";       let text2 = text1.toLowerCase();
```

Try it Yourself »

JavaScript String concat()

`concat()` joins two or more strings:

Example

```
let text1 = "Hello";
let text2 = "World";
let 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

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

Note

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.

JavaScript String trim()

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

Example

```
let text1 = "      Hello World!      ";
let text2 = text1.trim();
```

Try it Yourself »

JavaScript String trimStart()

ECMAScript 2019 added the String method `trimStart()` to JavaScript.

The `trimStart()` method works like `trim()`, but removes whitespace only from the start of a string.

Example

```
let text1 = "     Hello World!     ";
let text2 = text1.trimStart();
```

Try it Yourself »

JavaScript String `trimStart()` is supported in all modern browsers since January 2020:

Chrome 66	Edge 79	Firefox 61	Safari 12	Opera 50
Apr 2018	Jan 2020	Jun 2018	Sep 2018	May 2018

JavaScript String trimEnd()

ECMAScript 2019 added the string method `trimEnd()` to JavaScript.

The `trimEnd()` method works like `trim()`, but removes whitespace only from the end of a string.

Example

```
let text1 = "     Hello World!     ";
let text2 = text1.trimEnd();
```

Try it Yourself »

JavaScript String `trimEnd()` is supported in all modern browsers since January 2020:

Chrome 66	Edge 79	Firefox 61	Safari 12	Opera 50
Apr 2018	Jan 2020	Jun 2018	Sep 2018	May 2018

JavaScript String Padding

ECMAScript 2017 added two new string methods to JavaScript: `padStart()` and `padEnd()` to support padding at the beginning and at the end of a string.

JavaScript String padStart()

The `padStart()` method pads a string from the start.

It pads a string with another string (multiple times) until it reaches a given length.

Examples

Pad a string with "0" until it reaches the length 4:

```
let text = "5";
let padded = text.padStart(4,"0");
```

Try it Yourself »

Pad a string with "x" until it reaches the length 4:

```
let text = "5";
let padded = text.padStart(4,"x");
```

Try it Yourself »

Note

The `padStart()` method is a string method.

To pad a number, convert the number to a string first.

See the example below.

Example

```
let numb = 5;
let text = numb.toString();
let padded = text.padStart(4,"0");
```

Try it Yourself »

Browser Support

`padStart()` is an ECMAScript 2017 feature.

ES2017 is supported in all modern browsers since September 2017:

Chrome 58	Edge 15	Firefox 52	Safari 11	Opera 45
Apr 2017	Apr 2017	Mar 2017	Sep 2017	May 2017

`padStart()` is not supported in Internet Explorer.

JavaScript String padEnd()

The `padEnd()` method pads a string from the end.

It pads a string with another string (multiple times) until it reaches a given length.

Examples

```
let text = "5";
let padded = text.padEnd(4,"0");
```

Try it Yourself »

```
let text = "5";
let padded = text.padEnd(4,"x");
```

Try it Yourself »

Note

The `padEnd()` method is a string method.

To pad a number, convert the number to a string first.

See the example below.

Example

```
let numb = 5;
let text = numb.toString();
let padded = text.padEnd(4,"0");
```

Try it Yourself »

Browser Support

`padEnd()` is an ECMAScript 2017 feature.

ES2017 is supported in all modern browsers since September 2017:

Chrome 58	Edge 15	Firefox 52	Safari 11	Opera 45
Apr 2017	Apr 2017	Mar 2017	Sep 2017	May 2017

`padEnd()` is not supported in Internet Explorer.

JavaScript String repeat()

The `repeat()` method returns a string with a number of copies of a string.

The `repeat()` method returns a new string.

The `repeat()` method does not change the original string.

Examples

Create copies of a text:

```
let text = "Hello world!";
let result = text.repeat(2);
```

Try it Yourself »

```
let text = "Hello world!";
let result = text.repeat(4);
```

Try it Yourself »

Syntax

```
string.repeat(count)
```

Parameters

Parameter	Description
count	Required. The number of copies wanted.

Return Value

Type	Description
String	A new string containing the copies.

Browser Support

`repeat()` is an ES6 feature (JavaScript 2015).

ES6 is fully supported in all modern browsers since June 2017:

Chrome 51	Edge 15	Firefox 54	Safari 10	Opera 38
May 2016	Apr 2017	Jun 2017	Sep 2016	Jun 2016

`repeat()` is not supported in Internet Explorer.

Replacing String Content

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

Example

```
let text = "Please visit Microsoft!";
let newText = text.replace("Microsoft", "W3Schools");
```

Try it Yourself »

Note

The `replace()` method does not change the string it is called on.

The `replace()` method returns a new string.

The `replace()` method replaces **only the first** match

If you want to replace all matches, use a regular expression with the /g flag set. See examples below.

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

Example

```
let text = "Please visit Microsoft and Microsoft!";
let newText = text.replace("Microsoft", "W3Schools");
```

Try it Yourself »

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

Example

```
let text = "Please visit Microsoft!";
let newText = text.replace("MICROSOFT", "W3Schools");
```

Try it Yourself »

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

Example

```
let text = "Please visit Microsoft!";
let newText = text.replace(/MICROSOFT/i, "W3Schools");
```

Try it Yourself »

Note

Regular expressions are written without quotes.

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

Example

```
let text = "Please visit Microsoft and Microsoft!";
let newText = text.replace(/Microsoft/g, "W3Schools");
```

Try it Yourself »

Note

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

JavaScript String ReplaceAll()

In 2021, JavaScript introduced the string method `replaceAll()`:

Example

```
text = text.replaceAll("Cats","Dogs");
text = text.replaceAll("cats","dogs");
```

Try it Yourself »

The `replaceAll()` method allows you to specify a regular expression instead of a string to be replaced.

If the parameter is a regular expression, the global flag (g) must be set, otherwise a TypeError is thrown.

Example

```
text = text.replaceAll(/Cats/g,"Dogs");
text = text.replaceAll(/cats/g,"dogs");
```

Try it Yourself »

Note

`replaceAll()` is an ES2021 feature.

`replaceAll()` does not work in Internet Explorer.

Converting a String to an Array

If you want to work with a string as an array, you can convert it to an array.

JavaScript String split()

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

Example

```
text.split(",")    text.split(" ")    text.split("|")
```

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

```
text.split("")
```

Try it Yourself »

Complete String Reference

For a complete String reference, go to our:

Complete JavaScript String Reference.

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

Exercise?

Consider the following string:

```
let x = 'Having fun?';
```

Which one of the following statements returns 'fun'?

- ○ `x.slice(7, 10)`
- ○ `x.substring(7, 9)`
- ○ `x.substr(7, 10)`

Submit Answer »

❮ Previous | Next ❯

COLOR PICKER