**String Functions in JavaScript | Web Development Tutorials #50**

**String Functions In JavaScript**

In this tutorial, we are going to learn about different string functions that are mostly used in JavaScript. These functions are really helpful in extracting the strings either from DOM manipulation or from APIs or AJAX sources. Make a new file as *tut50.html* and add an instant boilerplate to get the HTML template. Then title it as **JavaScript String Functions** under the <title> tag.

We will begin by writing a very simple JavaScript code under the <script> tag as follows-

var str = "This is a string";

console.log(str);

Copy

Now let us see those functions of strings by which we can either modify, alter, slice, break, etc. the strings.

* Suppose we want to locate the position of any word in the string, then we can do as follows-

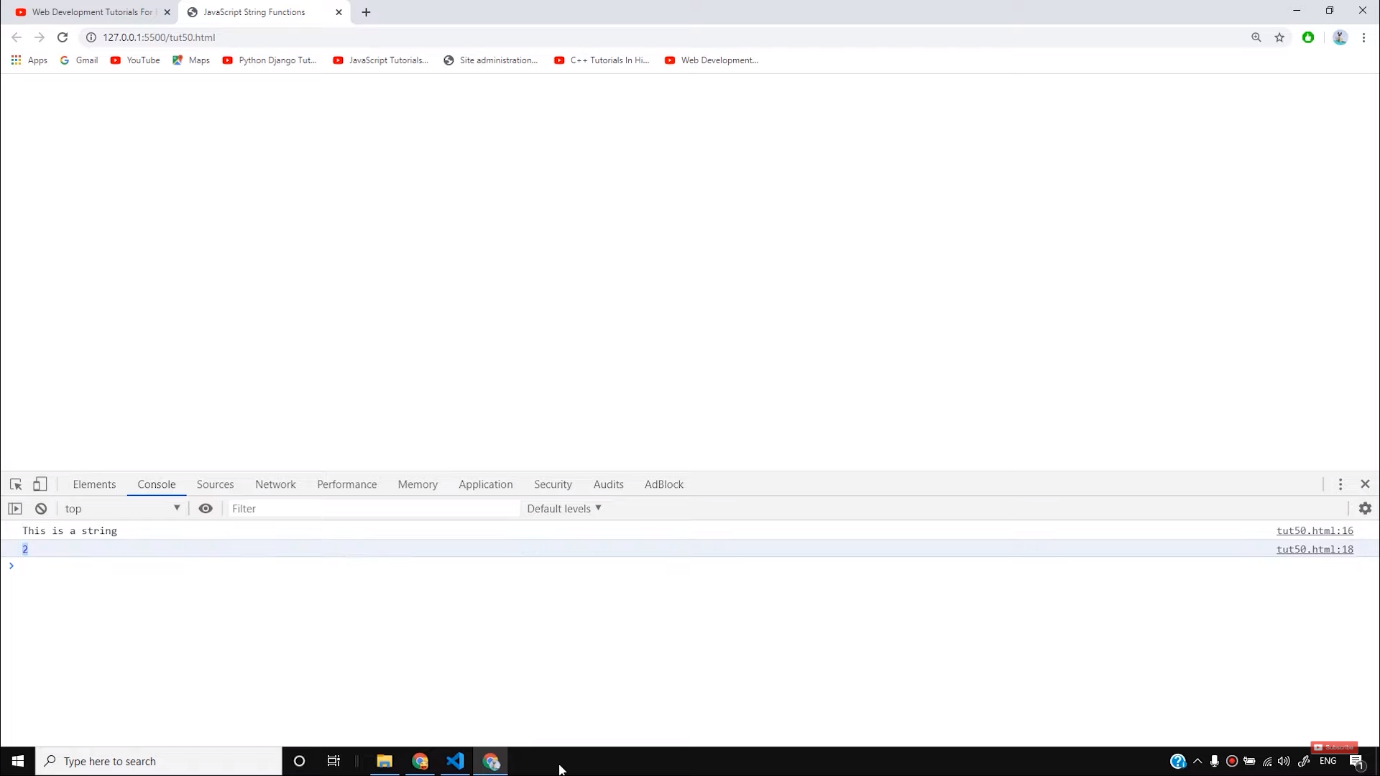
// First occurrence of a substring

var position = str.indexOf('is');

console.log(position)

Copy

**indexOf()**function is used here to locate the position of any string It gives the first occurrence of the substring. Here the position of “is” is 2, therefore the output will be as follows-



* To get the position of the last substring, we can use the function **lastIndexOf()** as follows-

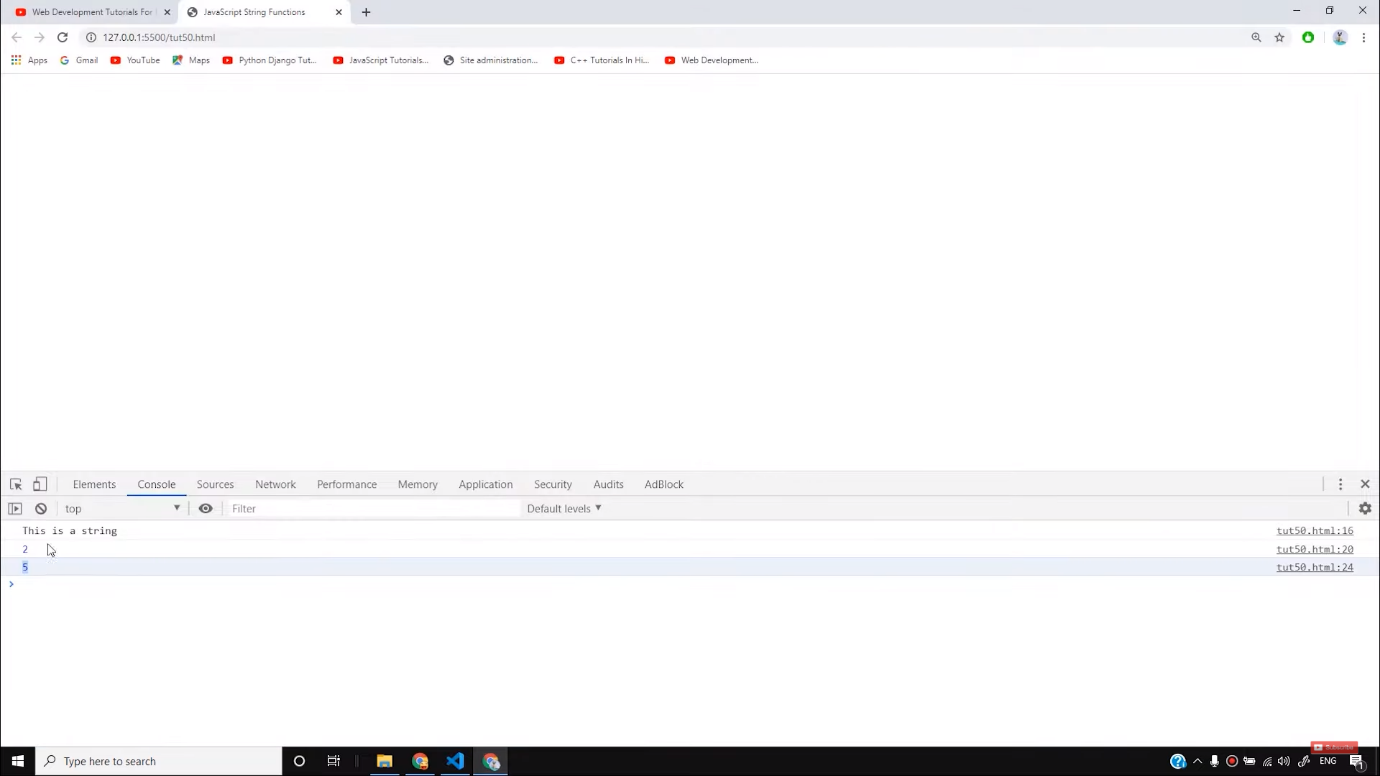
// Last occurence of a substring

position = str.lastIndexOf('is');

console.log(position)

Copy

Here the output will be 5 as follows-



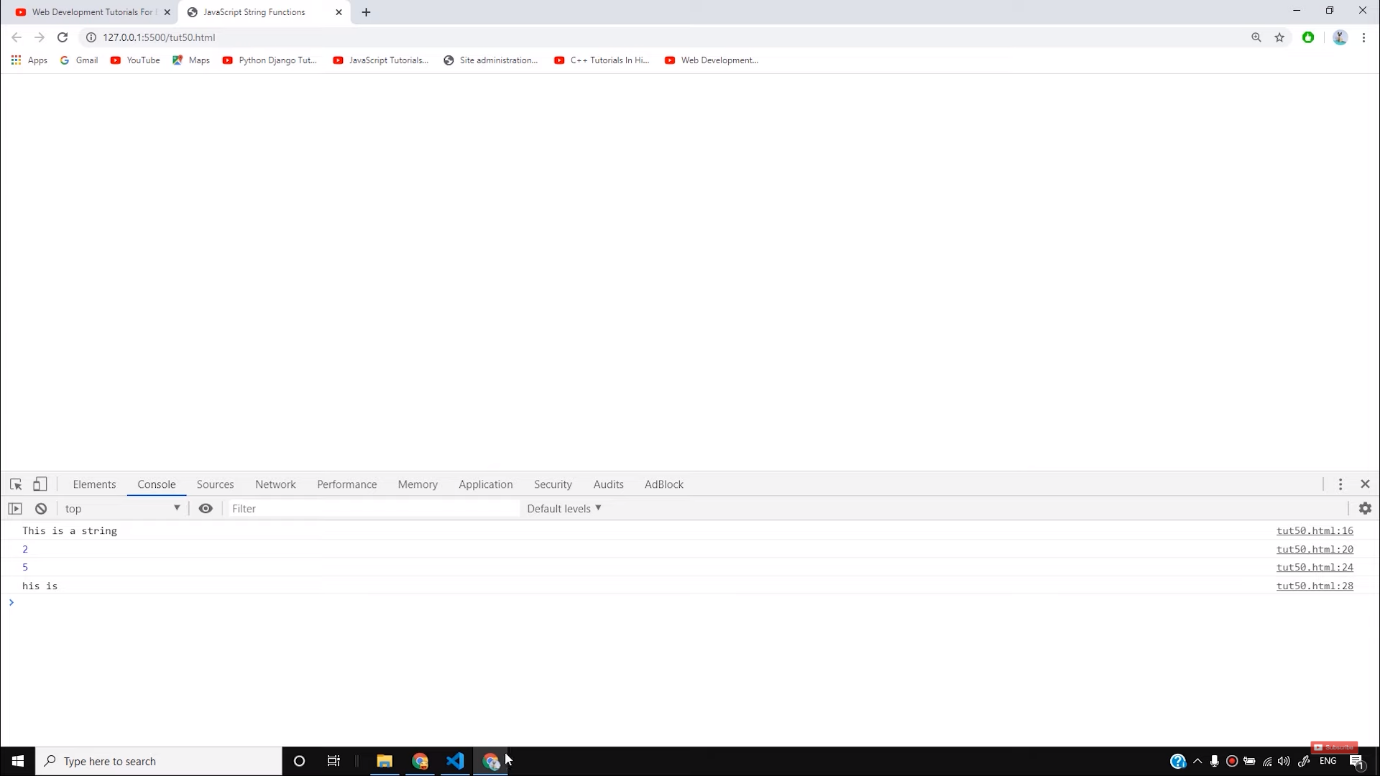
* To get a substring from a string we can use the **slice()** function as follows-

var substr = str.slice(1,7);

console.log(substr)

Copy

Here you will get the output of the substring from index numbers 1 to 7 as follows-



However, there are other functions that help to get the substrings like-

var substr = str.substring(1,7);

var substr1 = str.substr(1,3);

Copy

* We can also replace a particular string to another string with the help of **replace()**function as follows-

var replaced = str.replace('string', 'Harry');

console.log(str)

console.log(replaced)

Copy

In this example, the string “This is a string” has been now been changed to “This is a Harry” and is saved to a variable **replaced.**The original string is still saved in the variable **str.**

We can also convert the whole strong to an uppercase or lowercase letters as shown below-

console.log(str.toUpperCase());

console.log(str.toLowerCase());

Copy

* To concat any two strings, we can take the help of **concat()**function as follows-

var newString = str.concat('New String')

console.log(newString)

Copy

* To remove all the white spaces in the string, we can use **trim()**function as shown below-

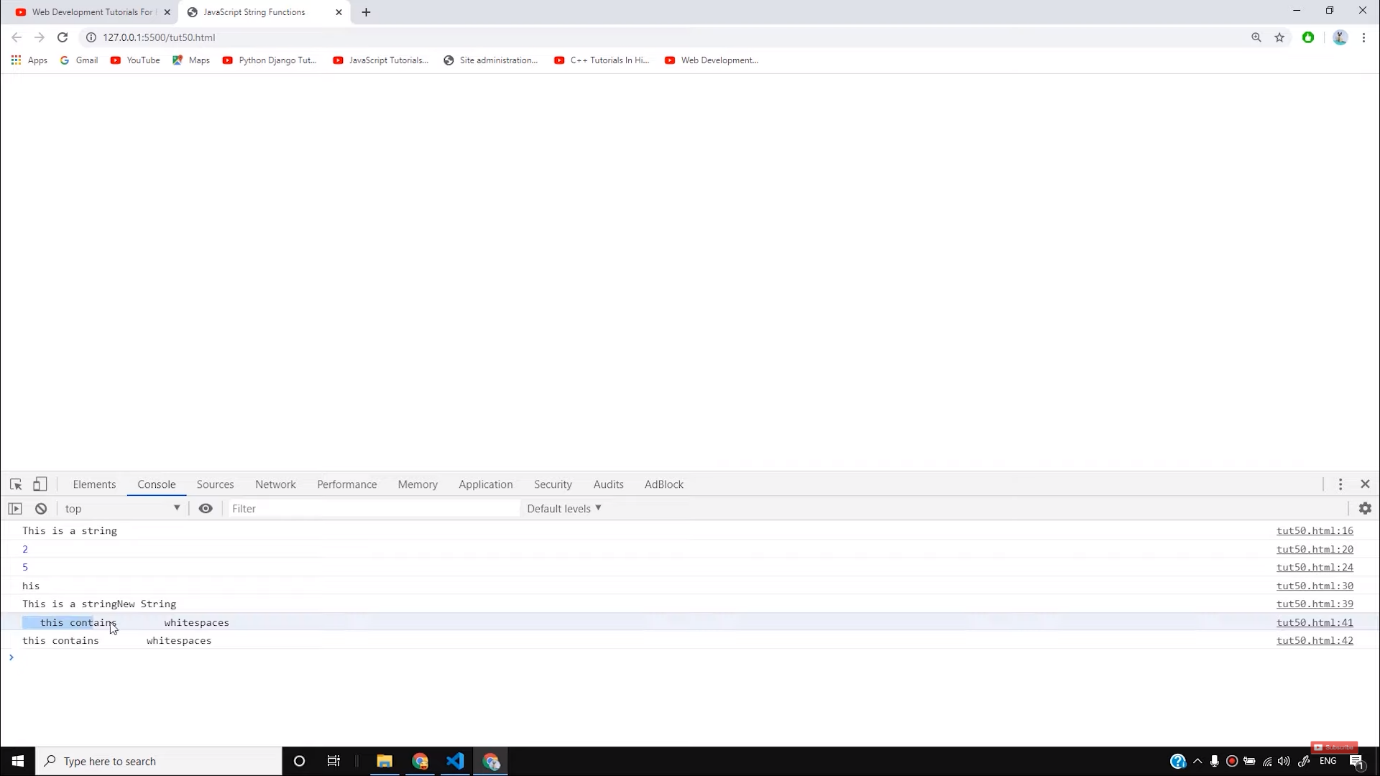
var strWithWhitespaces = " this contains whitespaces ";

console.log(strWithWhitespaces)

console.log(strWithWhitespaces.trim())

Copy

The result will be as follows-



* To extract any character from a string, we can use **charAt()**function as follows-

var char2 = str.charAt(2);

console.log(char2)

Copy

It will return ‘i’ as the output.

So I believe, you must have understood how these various JavaScript functions work on a website. In the future, we will make very good websites equipped with these functions. Till then, stay tuned with the tutorials, and keep practicing.

**Code as described/written in the video**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript String Functions</title>

<style>

</style>

</head>

<body>

<script>

var str = "This is a string";

console.log(str);

// First occurence of a substring

var position = str.indexOf('is');

console.log(position)

// Last occurence of a substring

position = str.lastIndexOf('is');

console.log(position)

// Substring from a string

// var substr = str.slice(1,7);

// var substr = str.substring(1,7);

var substr1 = str.substr(1,3);

console.log(substr1)

// var replaced = str.replace('string', 'Harry');

// console.log(str)

// console.log(replaced)

// console.log(str.toUpperCase());

// console.log(str.toLowerCase());

// var newString = str.concat('New String')

// console.log(newString)

// var strWithWhitespaces = " this contains whitespaces ";

// console.log(strWithWhitespaces)

// console.log(strWithWhitespaces.trim())

// var char2 = str.charAt(2);

// var char2 = str.charCodeAt(2); // Not very important

// console.log(char2)

console.log(str[3])

</script>

</body>

</html>

# JavaScript String Methods

[❮ Previous](https://www.w3schools.com/js/js_strings.asp)[Next ❯](https://www.w3schools.com/js/js_string_search.asp)

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.

## **JavaScript String Length**

The length property returns the length of a string:

### **Example**

let txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
let length = txt.length;

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_length)

## **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: the start position, and the end position (end not included).

### **Example**

Slice out a portion of a string from position 7 to position 13 (13 not included):

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

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_slice)

## **Note**

JavaScript counts positions from zero.

First position is 0.

Second position is 1.

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**

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

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_slice_negative)

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

### **Example**

let part = str.slice(7);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_slice_rest)

or, counting from the end:

### **Example**

let part = str.slice(-12);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_slice_rest_negative)

ADVERTISEMENT

## **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 »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_substring)

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 »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_substr)

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 »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_substr1)

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 »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_substr2)

## **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 »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_replace)

## **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 »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_replace_first)

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 »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_replace_case)

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 »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_replace_insensitive)

## **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 »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_replace_global)

## **Note**

You will learn a lot more about regular expressions in the chapter [JavaScript Regular Expressions](https://www.w3schools.com/js/js_regexp.asp).

## **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 »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_toupper)

## **JavaScript String toLowerCase()**

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_tolower)

## **JavaScript String concat()**

concat() joins two or more strings:

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_concat)

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 »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_trim)

## **JavaScript String Padding**

ECMAScript 2017 added two String methods: 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 with another string:

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_padding4)

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_padding1)

## **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 »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_padding3)

## **Browser Support**

padStart() is an ECMAScript 2017 feature.

It is supported in all modern browsers:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Chrome | Edge | Firefox | Safari | Opera |
| Yes | Yes | Yes | Yes | Yes |

padStart() is not supported in Internet Explorer.

## **JavaScript String padEnd()**

The padEnd() method pads a string with another string:

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_padding6)

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_padding2)

## **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 »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_padding5)

## **Browser Support**

padEnd() is an ECMAScript 2017 feature.

It is supported in all modern browsers:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Chrome | Edge | Firefox | Safari | Opera |
| Yes | Yes | Yes | Yes | Yes |

padEnd() is not supported in Internet Explorer.

## **Extracting String Characters**

There are 3 methods for extracting string characters:

* charAt(position)
* charCodeAt(position)
* Property access [ ]

## **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 »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_charat)

## **JavaScript String charCodeAt()**

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**

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

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_charcodeat)

## **Property Access**

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

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_prop)

## **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";    // Gives no error, but does not work

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_prop2)

## **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(",")    // Split on commas  
text.split(" ")    // Split on spaces  
text.split("|")    // Split on pipe

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_split)

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 »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_split_char)

# JavaScript String Search

[❮ Previous](https://www.w3schools.com/js/js_string_methods.asp)[Next ❯](https://www.w3schools.com/js/js_string_templates.asp)

## **JavaScript Search Methods**

* String indexOf()
* String lastIndexOf()
* String startsWith()
* String endsWith()

## **JavaScript String indexOf()**

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

### **Example**

let str = "Please locate where 'locate' occurs!";  
str.indexOf("locate");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_indexof)

## **Note**

JavaScript counts positions from zero.

0 is the first position in a string, 1 is the second, 2 is the third, ...

## **JavaScript String lastIndexOf()**

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

### **Example**

let str = "Please locate where 'locate' occurs!";  
str.lastIndexOf("locate");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_lastindexof)

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

### **Example**

let str = "Please locate where 'locate' occurs!";  
str.lastIndexOf("John");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_indexof_1)

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

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_indexof_2)

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**

let str = "Please locate where 'locate' occurs!";  
str.lastIndexOf("locate", 15);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_lastindexof_2)

## **JavaScript String search()**

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

### **Example**

let str = "Please locate where 'locate' occurs!";  
str.search("locate");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_search_locate)

## **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:

* 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.

ADVERTISEMENT

## **JavaScript String match()**

The match() method searches a string for a match against a regular expression, and returns the matches, as an Array object.

### **Example 1**

Search a string for "ain":

let text = "The rain in SPAIN stays mainly in the plain";  
text.match(/ain/g);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_match_regexp)

Read more about regular expressions in the chapter [JS RegExp](https://www.w3schools.com/js/js_regexp.asp).

## **Note**

If a regular expression does not include the g modifier (to perform a global search), the match() method will return only the first match in the string.

## **Syntax**

*string*.match(regexp)

|  |  |
| --- | --- |
| regexp | Required. The value to search for, as a regular expression. |
| Returns: | An Array, containing the matches, one item for each match, or null if no match is found |

### **Example 2**

Perform a global, case-insensitive search for "ain":

let text = "The rain in SPAIN stays mainly in the plain";  
text.match(/ain/gi);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_match_regexp2)

## **JavaScript String includes()**

The includes() method returns true if a string contains a specified value.

### **Example**

let text = "Hello world, welcome to the universe.";  
text.includes("world");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_includes)

## **Syntax**

*string*.includes(searchvalue, start)

|  |  |
| --- | --- |
| searchvalue | Required. The string to search for |
| start | Optional. Default 0. Position to start the search |
| Returns: | Returns true if the string contains the value, otherwise false |
| JS Version: | ES6 (2015) |

Check if a string includes "world", starting the search at position 12:

let text = "Hello world, welcome to the universe.";  
text.includes("world", 12);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_includes2)

## **Browser Support**

includes() is an ES6 feature (JavaScript 2015).

It is supported in all modern browsers:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Chrome | Edge | Firefox | Safari | Opera |
| Yes | Yes | Yes | Yes | Yes |

includes() is not supported in Internet Explorer.

## **JavaScript String startsWith()**

The startsWith() method returns true if a string begins with a specified value, otherwise false:

### **Example**

let text = "Hello world, welcome to the universe.";  
  
text.startsWith("Hello");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_startswith)

## **Syntax**

*string*.startsWith(searchvalue, start)

## **Parameter Values**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| searchvalue | Required. The value to search for. |
| start | Optional. Default 0. The position to start the search. |

### **Examples**

let text = "Hello world, welcome to the universe.";  
  
text.startsWith("world")    // Returns false

let text = "Hello world, welcome to the universe.";  
  
text.startsWith("world", 5)    // Returns false

let text = "Hello world, welcome to the universe.";  
  
text.startsWith("world", 6)    // Returns true

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_startswith2)

## **Note**

The startsWith() method is case sensitive.

## **Browser Support**

startsWith() is an ES6 feature (JavaScript 2015).

It is supported in all modern browsers:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Chrome | Edge | Firefox | Safari | Opera |
| Yes | Yes | Yes | Yes | Yes |

startsWith() is not supported in Internet Explorer.

## **JavaScript String endsWith()**

The endsWith() method returns true if a string ends with a specified value, otherwise false:

### **Example**

Check if a string ends with "Doe":

let text = "John Doe";  
text.endsWith("Doe");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_endswith)

## **Syntax**

*string*.endsWith(searchvalue, length)

## **Parameter Values**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| searchvalue | Required. The value to search for. |
| length | Optional. The length to search. |

Check if the 11 first characters of a string ends with "world":

let text = "Hello world, welcome to the universe.";  
text.endsWith("world", 11);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_endswith2)

## **Note**

The endsWith() method is case sensitive.

## **Browser Support**

endsWith() is an ES6 feature (JavaScript 2015).

It is supported in all modern browsers:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Chrome | Edge | Firefox | Safari | Opera |
| Yes | Yes | Yes | Yes | Yes |

endsWith() is not supported in Internet Explorer.

## **Complete String Reference**

For a complete String reference, go to our:

[Complete JavaScript String Reference](https://www.w3schools.com/jsref/jsref_obj_string.asp).

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

# JavaScript Template Literals

[❮ Previous](https://www.w3schools.com/js/js_string_search.asp)[Next ❯](https://www.w3schools.com/js/js_numbers.asp)

Synonyms:

* Template Literals
* Template Strings
* String Templates
* Back-Tics Syntax

## **Back-Tics Syntax**

**Template Literals** use back-ticks (``) rather than the quotes ("") to define a string:

### **Example**

let text = `Hello World!`;

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_templates)

## **Quotes Inside Strings**

With **template literals**, you can use both single and double quotes inside a string:

### **Example**

let text = `He's often called "Johnny"`;

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_templates_quotes)

## **Multiline Strings**

**Template literals** allows multiline strings:

### **Example**

let text =  
`The quick  
brown fox  
jumps over  
the lazy dog`;

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_templates_multiline)

## **Interpolation**

**Template literals** provide an easy way to interpolate variables and expressions into strings.

The method is called string interpolation.

The syntax is:

${...}

ADVERTISEMENT

## **Variable Substitutions**

**Template literals** allow variables in strings:

### **Example**

let firstName = "John";  
let lastName = "Doe";  
  
let text = `Welcome ${firstName}, ${lastName}!`;

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_templates_variables)

Automatic replacing of variables with real values is called **string interpolation**.

## **Expression Substitution**

**Template literals** allow expressions in strings:

### **Example**

let price = 10;  
let VAT = 0.25;  
  
let total = `Total: ${(price \* (1 + VAT)).toFixed(2)}`;

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_templates_expressions)

Automatic replacing of expressions with real values is called **string interpolation**.

## **HTML Templates**

### **Example**

let header = "Templates Literals";  
let tags = ["template literals", "javascript", "es6"];  
  
let html = `<h2>${header}</h2><ul>`;  
for (const x of tags) {  
  html += `<li>${x}</li>`;  
}  
  
html += `</ul>`;

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_templates_html)