JavaScript Strings

JavaScript strings are for storing and manipulating text.

A JavaScript string is zero or more characters written inside quotes.

### Example

let text = "John Doe";

You can use single or double quotes:

### Example

let carName1 = "Volvo XC60";  // Double quotes  
let carName2 = 'Volvo XC60';  // Single quotes

You can use quotes inside a string, as long as they don't match the quotes surrounding the string:

### Example

let answer1 = "It's alright";  
let answer2 = "He is called 'Johnny'";  
let answer3 = 'He is called "Johnny"';

## String Length

To find the length of a string, use the built-in length property:

### Example

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

//length = 26

## Escape Character

Because strings must be written within quotes, JavaScript will misunderstand this string:

let text = "We are the so-called "Vikings" from the north.";

The string will be chopped to "We are the so-called ".

The solution to avoid this problem, is to use the **backslash escape character**.

The backslash (\) escape character turns special characters into string characters:

|  |  |  |
| --- | --- | --- |
| **Code** | **Result** | **Description** |
| \' | ' | Single quote |
| \" | " | Double quote |
| \\ | \ | Backslash |

The sequence \"  inserts a double quote in a string:

### Example

let text = "We are the so-called \"Vikings\" from the north.";

The sequence \'  inserts a single quote in a string:

### Example

let text= 'It\'s alright.';

The sequence \\  inserts a backslash in a string:

### Example

let text = "The character \\ is called backslash.";

Six other escape sequences are valid in JavaScript:

|  |  |
| --- | --- |
| **Code** | **Result** |
| \b | Backspace |
| \f | Form Feed |
| \n | New Line |
| \r | Carriage Return |
| \t | Horizontal Tabulator |
| \v | Vertical Tabulator |

The 6 escape characters above were originally designed to control typewriters, teletypes, and fax machines. They do not make any sense in HTML.

## Breaking Long Code Lines

For best readability, programmers often like to avoid code lines longer than 80 characters.

If a JavaScript statement does not fit on one line, the best place to break it is after an operator:

### Example

document.getElementById("demo").innerHTML =  
"Hello Dolly!";

You can also break up a code line **within a text string** with a single backslash:

### Example

document.getElementById("demo").innerHTML = "Hello \  
Dolly!";

The \ method is not the preferred method. It might not have universal support.  
Some browsers do not allow spaces behind the \ character.

A safer way to break up a string, is to use string addition:

### Example

document.getElementById("demo").innerHTML = "Hello " +  
"Dolly!";

You cannot break up a code line with a backslash:

### Example

document.getElementById("demo").innerHTML = \  
"Hello Dolly!";

## JavaScript Strings as Objects

Normally, JavaScript strings are primitive values, created from literals:

let x = "John";

But strings can also be defined as objects with the keyword new:

let y = new String("John");

### Example

let x = "John";  
let y = new String("John");

Do not create Strings objects.

The new keyword complicates the code and slows down execution speed.

String objects can produce unexpected results:

When using the == operator, x and y are **equal**:

let x = "John";  
let y = new String("John");

When using the === operator, x and y are **not equal**:

let x = "John";  
let y = new String("John");

Note the difference between (x==y) and (x===y).

(x == y) true or false?

let x = new String("John");  
let y = new String("John");

(x === y) true or false?

let x = new String("John");  
let y = new String("John");

Comparing two JavaScript objects **always** returns **false**.

## 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 String Methods

|  |  |
| --- | --- |
| String length String slice() String substring() String substr() String replace() String replaceAll() String toUpperCase() String toLowerCase() String concat() | String trim() String trimStart() String trimEnd() String padStart() String padEnd() String charAt() String charCodeAt() String split() |

String search methods are covered in the next chapter.

## JavaScript String Length

The length property returns the length of a string:

### Example

let text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
let length = text.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: 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);

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Strings</h1>

<h2>The slice() Method</h2>

<p>The sliced (extracted) part of the string is:</p>

<p id="demo"></p>

<script>

let text = "Apple, Banana, Kiwi";

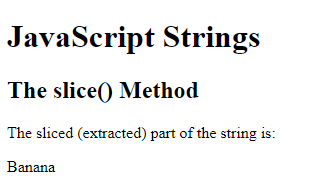
let part = text.slice(7,13);

document.getElementById("demo").innerHTML = part;

</script>

</body>

</html>



## Note

JavaScript counts positions from zero.

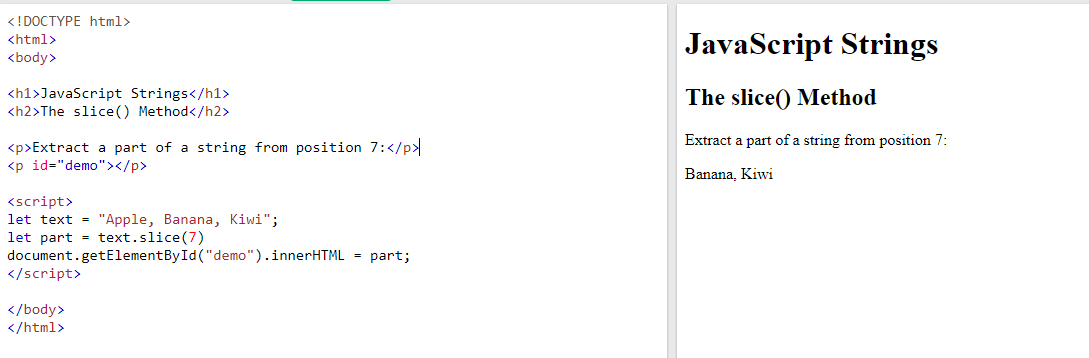
First position is 0.

Second position is 1.

### Examples

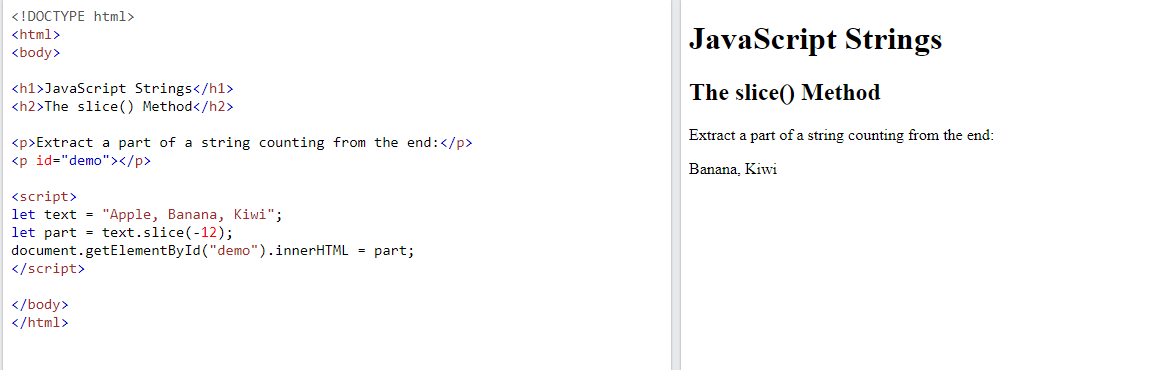
If you omit the second parameter, the method will slice out the rest of the string:

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



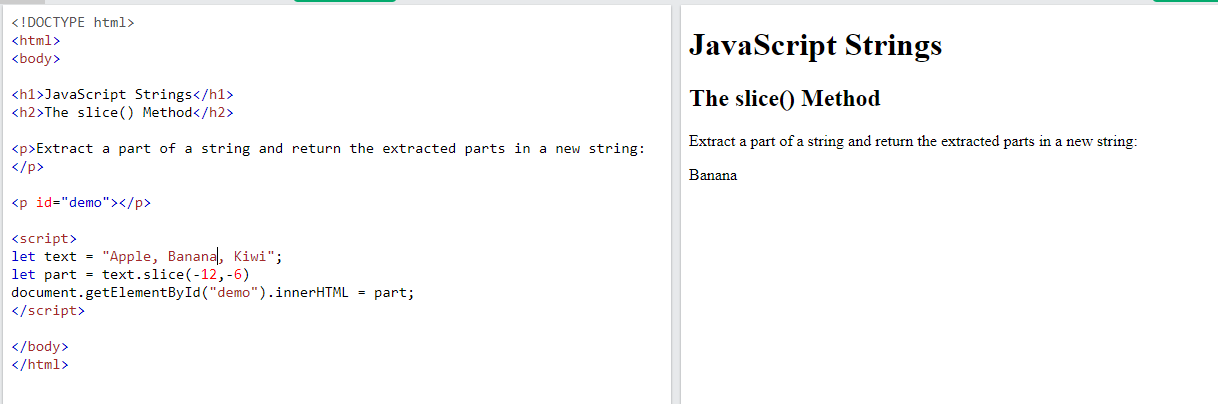
If a parameter is negative, the position is counted from the end of the string:

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



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);



## JavaScript String substring()

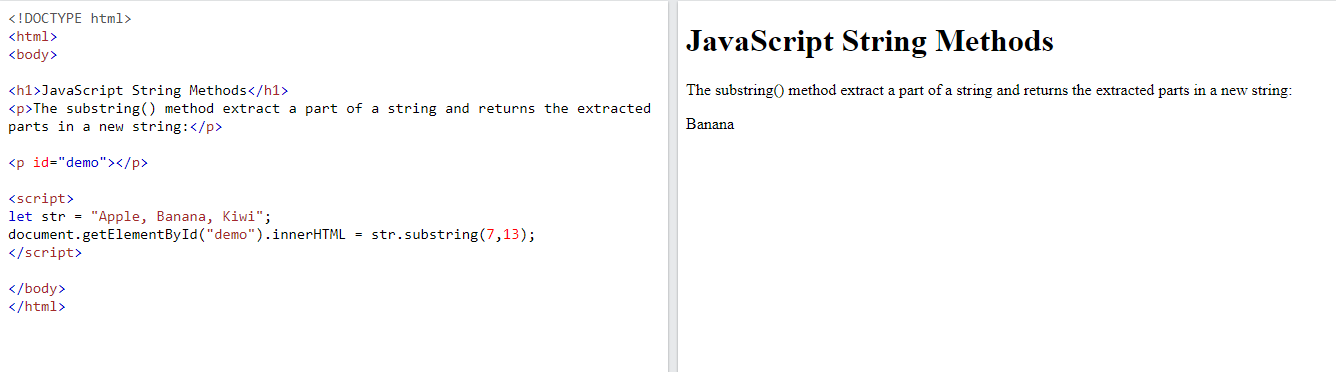
substring() is similar to slice().

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);

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



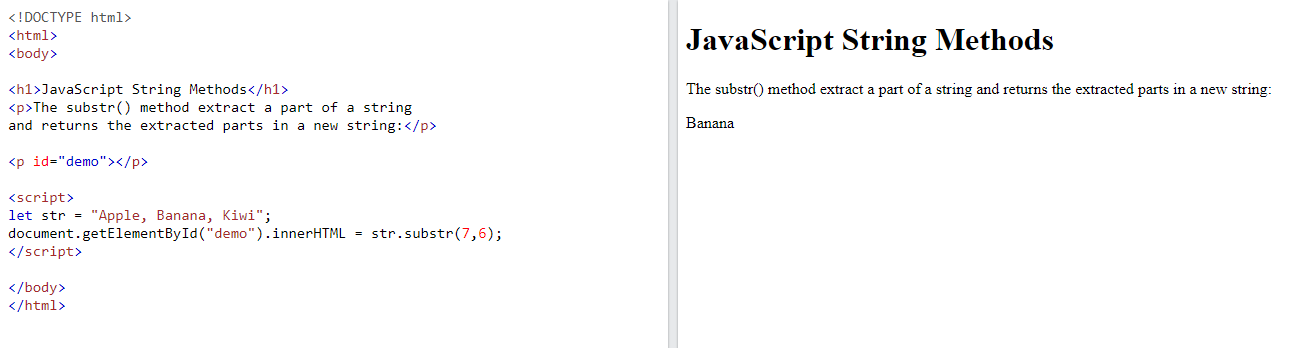
## JavaScript String substr()

substr() is similar to slice().

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

### Example

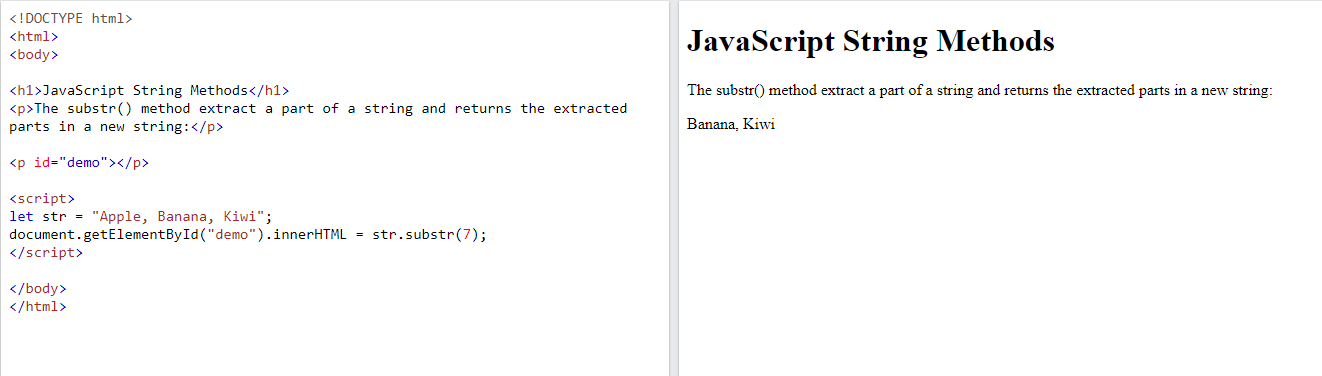
let str = "Apple, Banana, Kiwi";  
let part = str.substr(7, 6);



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

### Example

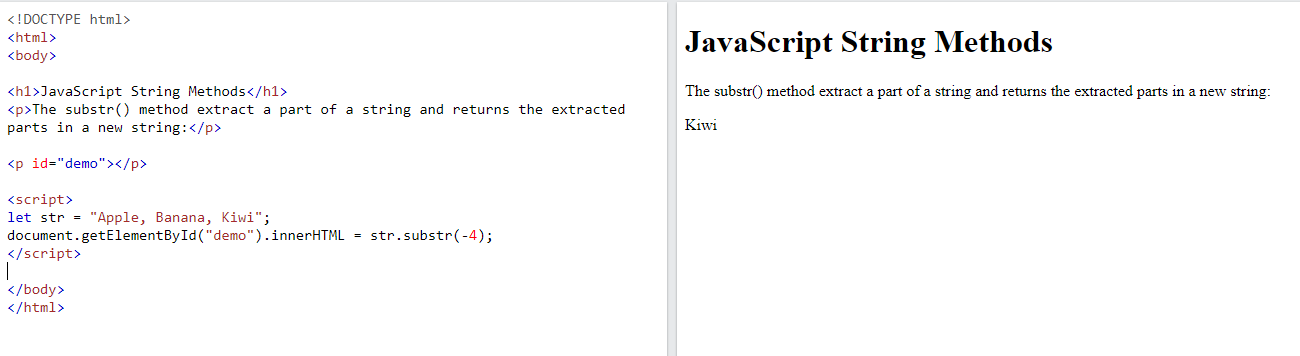
let str = "Apple, Banana, Kiwi";  
let part = str.substr(7);



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);

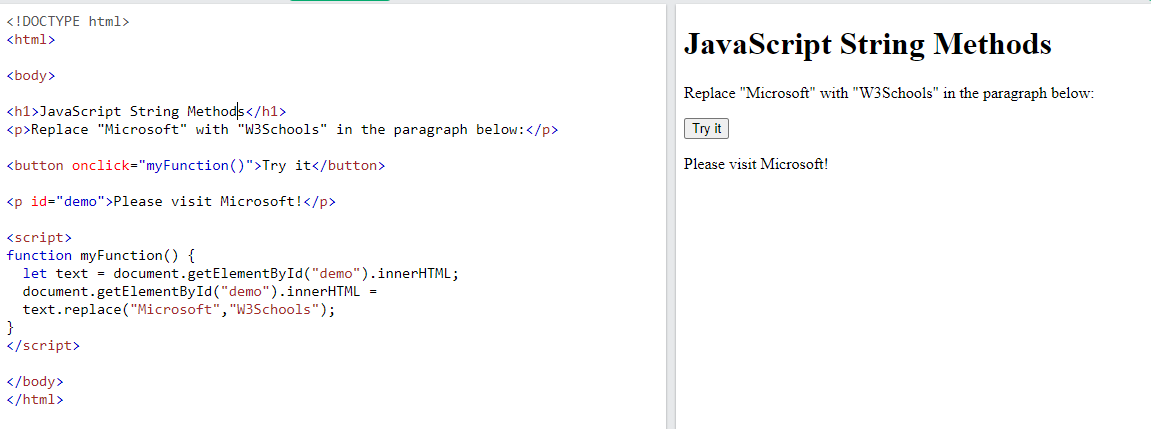


## Replacing String Content

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");





## 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");

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");

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");

## 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");

## Note

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

## JavaScript String ReplaceAll()

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

### Example

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

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");



## Note

replaceAll() is an [ES2021](https://www.w3schools.com/js/js_2021.asp) feature.

replaceAll() does not work in Internet Explorer.

## 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();

## JavaScript String toLowerCase()

### Example

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

## JavaScript String concat()

concat() joins two or more strings:

### Example

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



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:

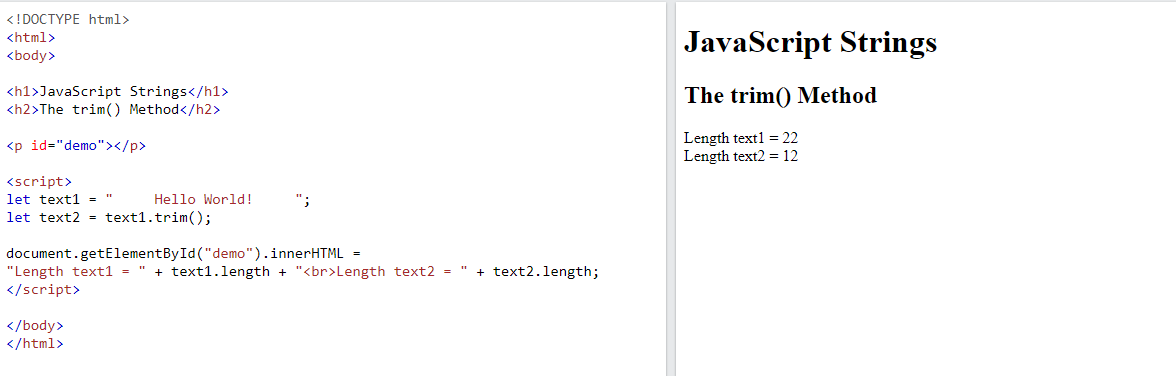
Strings are immutable: Strings cannot be changed, only replaced.

## JavaScript String trim()

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

### Example

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



## JavaScript String trimStart()

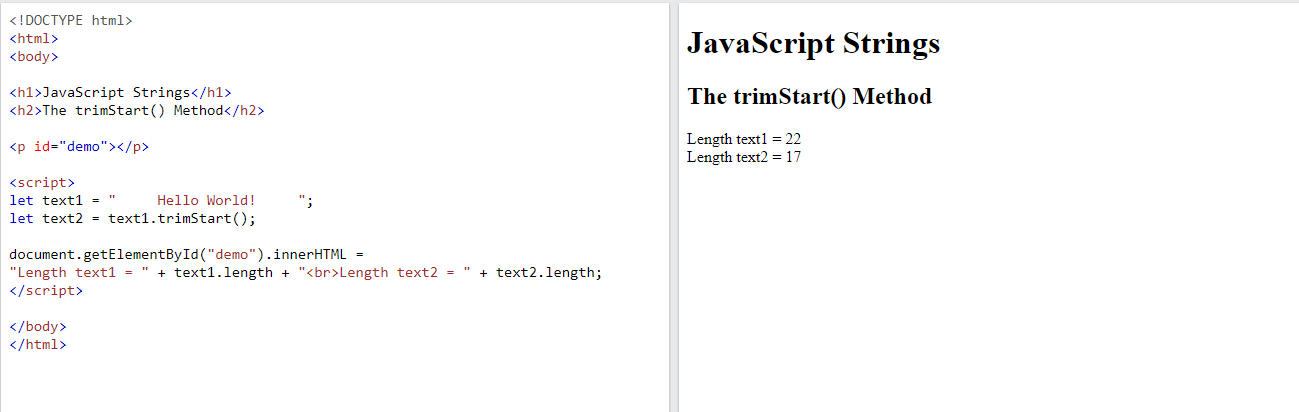
ECMAScript 2019 added the String method trimStart() to JavaScript.

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();

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



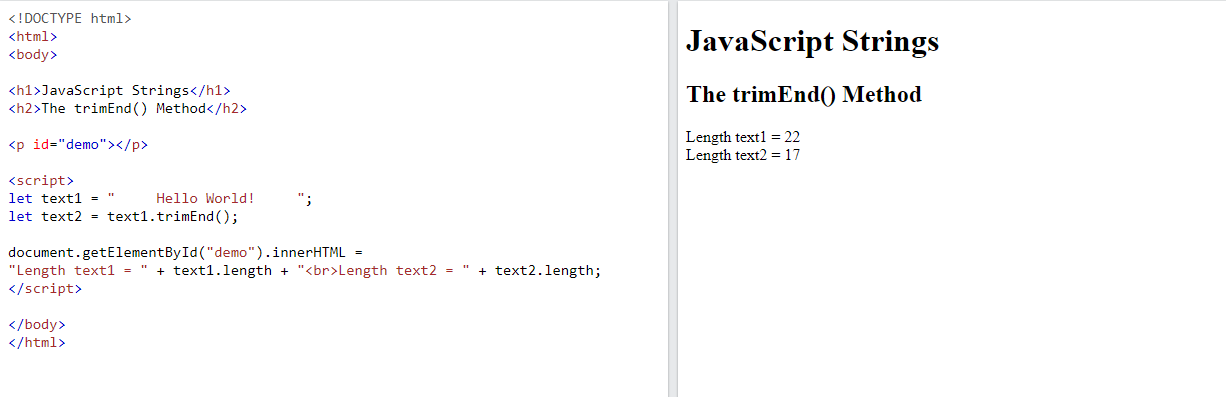
## JavaScript String trimEnd()

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();



avaScript String trimEnd() is supported in all browsers since January 2020:

## JavaScript String Padding

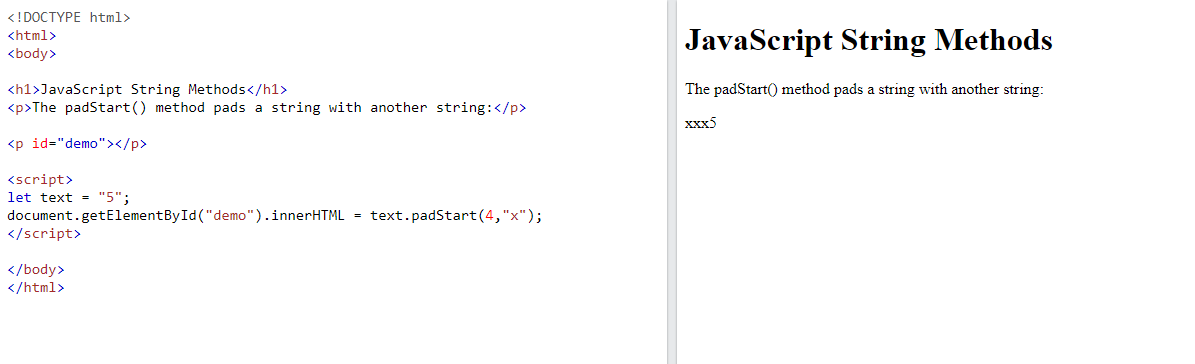
ECMAScript 2017 added two String methods: padStart() and padEnd() to support padding at the beginning and at the end of a string.

## JavaScript String padStart()

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

### Example

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



## 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");

## Browser Support

padStart() is an ECMAScript 2017 feature.

It is supported in all modern browsers:

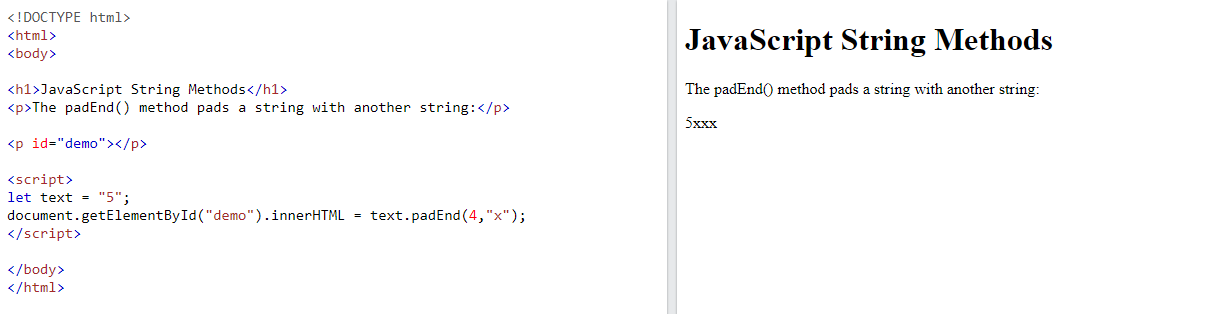
padStart() is not supported in Internet Explorer.

## JavaScript String padEnd()

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

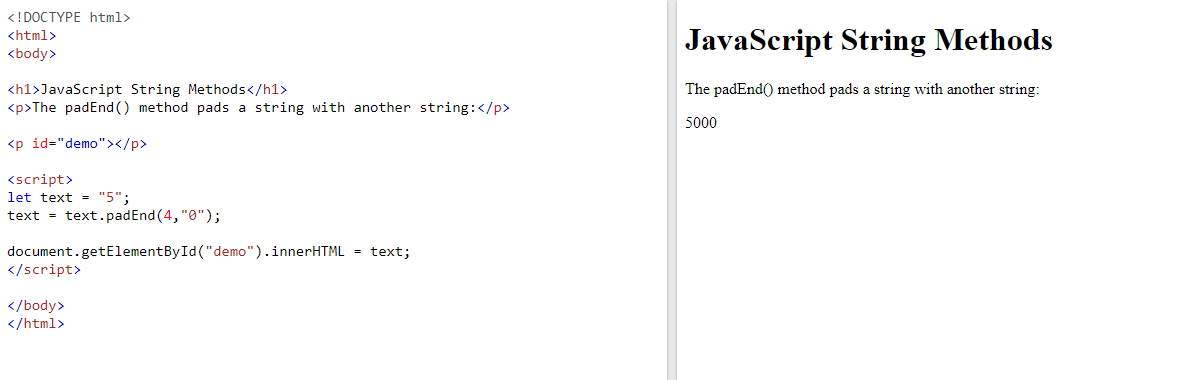
### Example

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



### Example

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



## 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");

## Browser Support

padEnd() is an ECMAScript 2017 feature.

It is supported in all modern browsers:

## Extracting String Characters

There are 3 methods for extracting string characters:

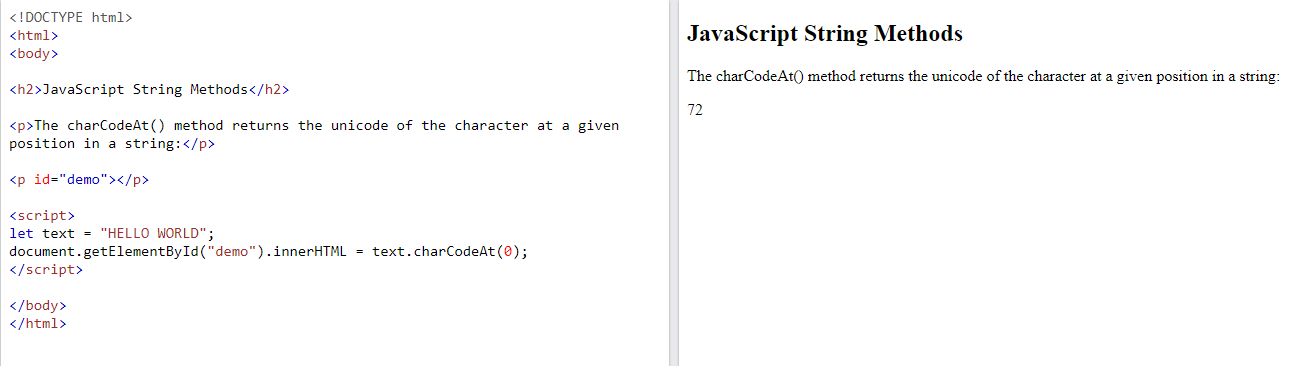
* charAt(position)
* charCodeAt(position)
* Property access [ ]

## 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);



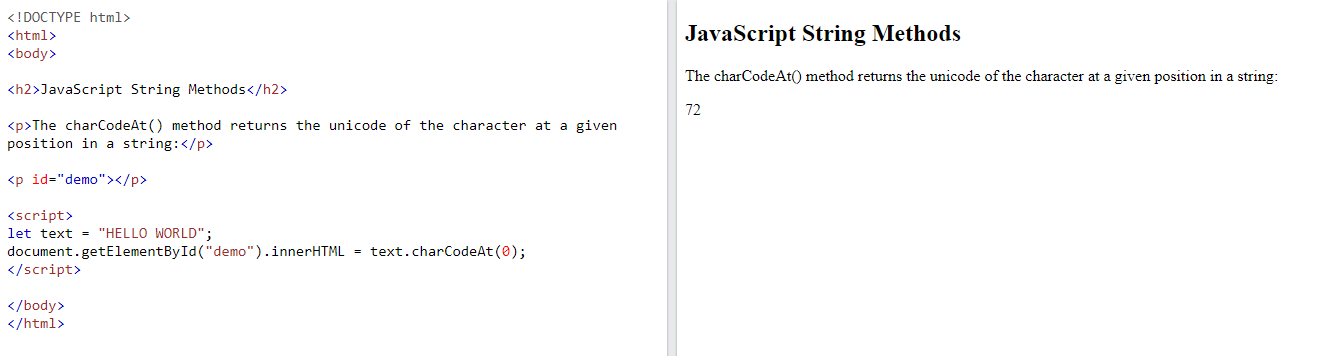
## JavaScript String charCodeAt()

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);

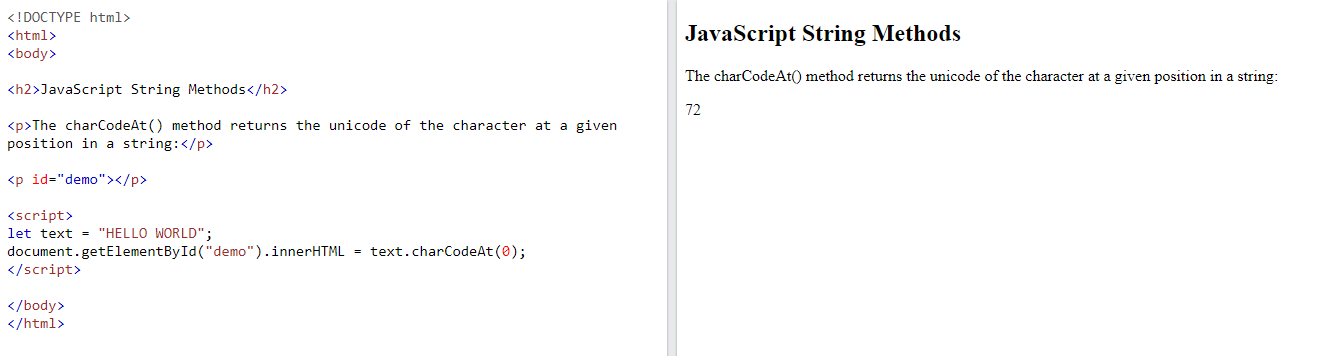


## Property Access

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

### Example

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



## Converting a String to an Array

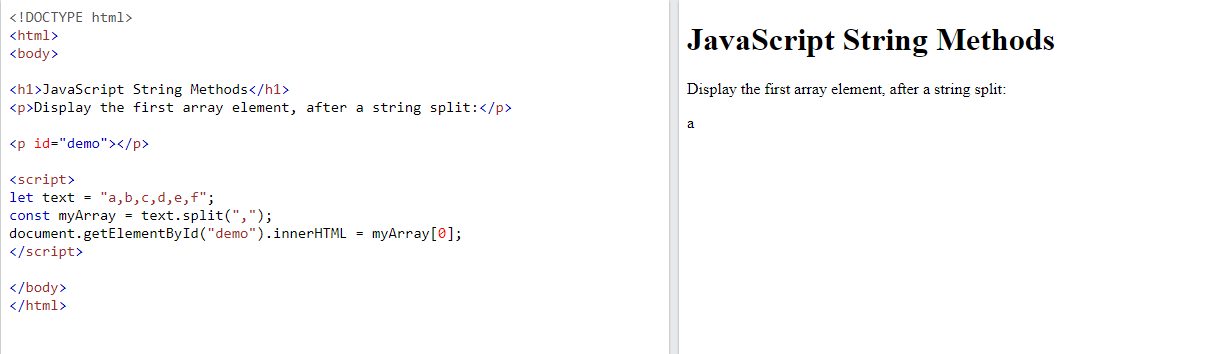
If you want to work with a string as an array, you can convert it to an array.

## JavaScript String split()

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

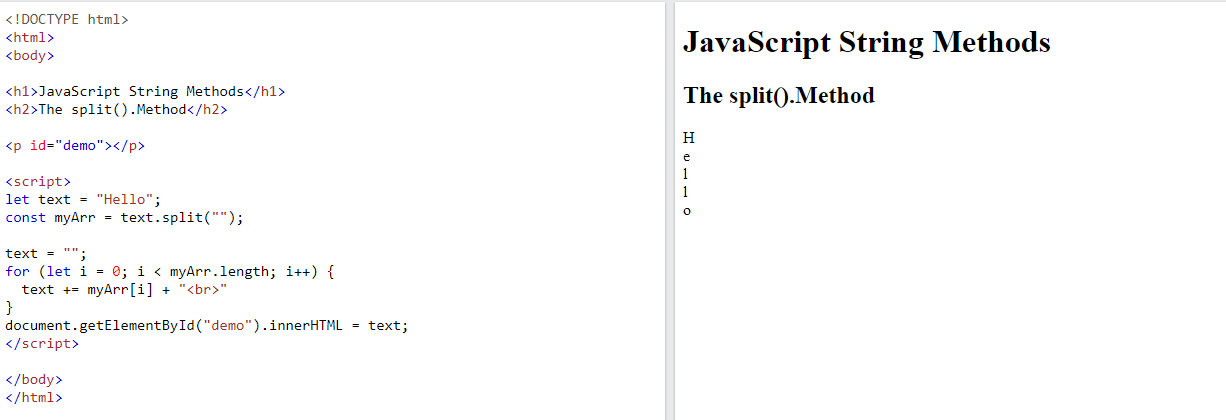
### Example

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



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:



# JavaScript String Search

## JavaScript Search Methods

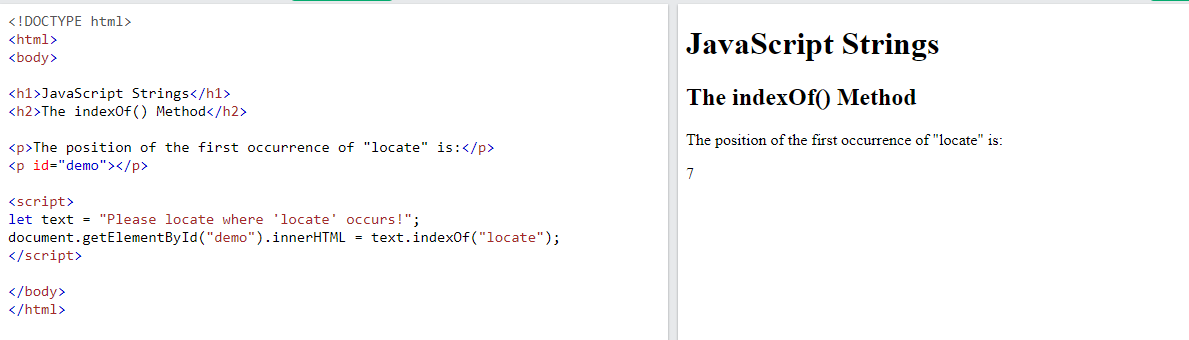
* String indexOf()
* String lastIndexOf()
* String search()
* String match()
* String matchAll()
* String includes()
* String startsWith()
* String endsWith()

## JavaScript String indexOf()

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

### Example

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



## 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

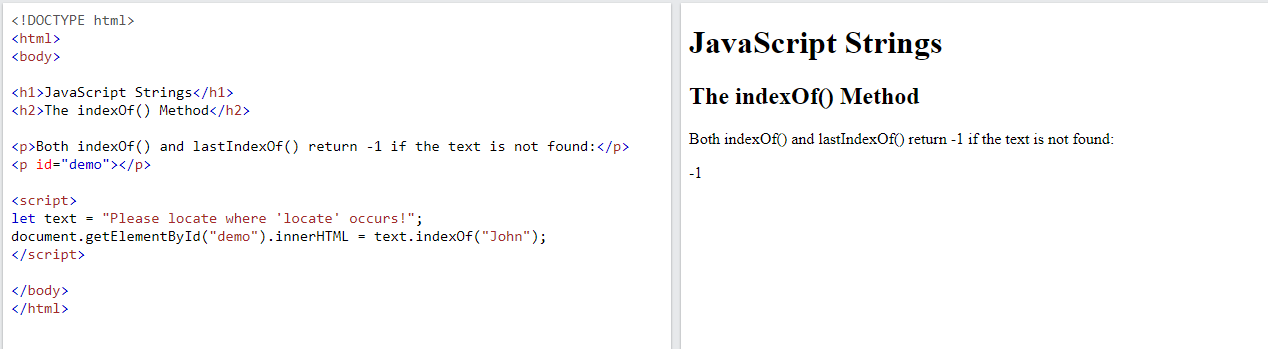
let text = "Please locate where 'locate' occurs!";  
text.lastIndexOf("locate");



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

### Example

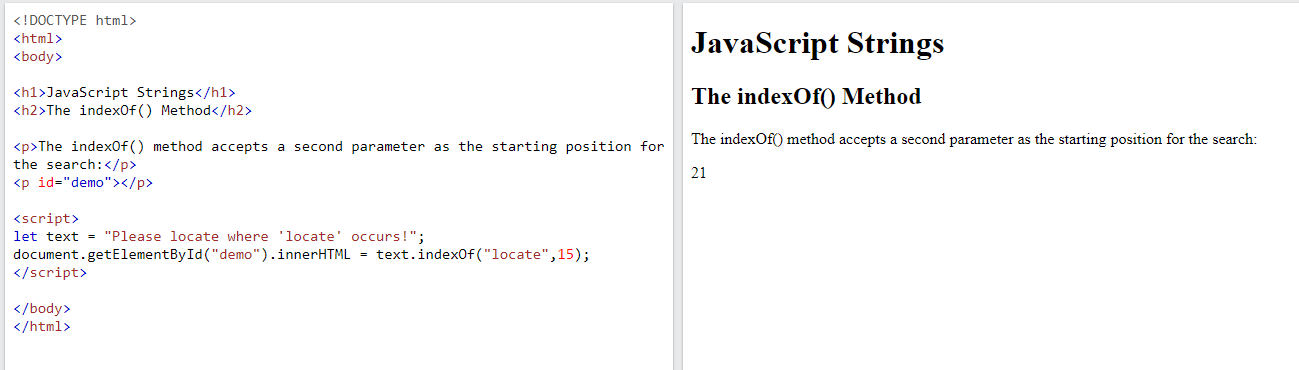
let text = "Please locate where 'locate' occurs!";  
text.lastIndexOf("John");



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

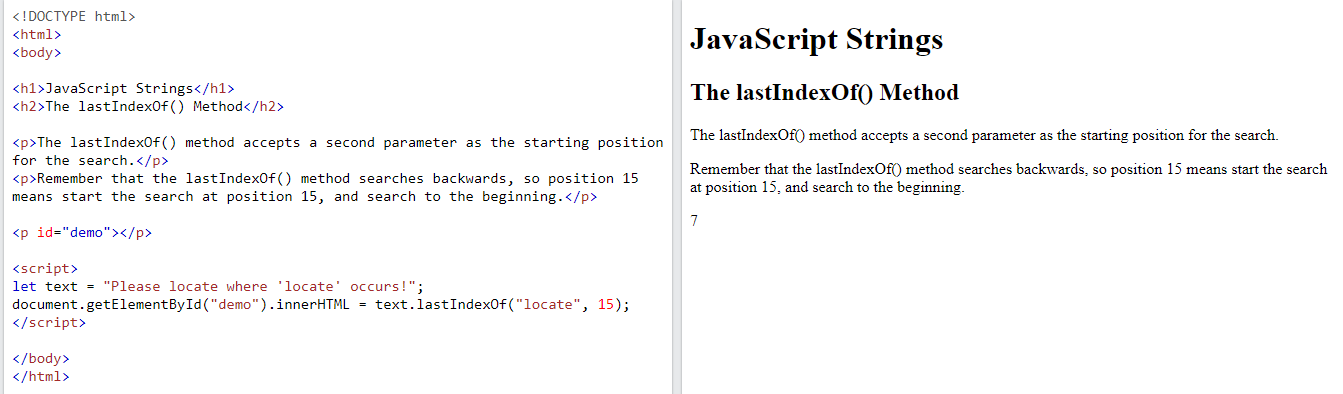
### Example

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



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.

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

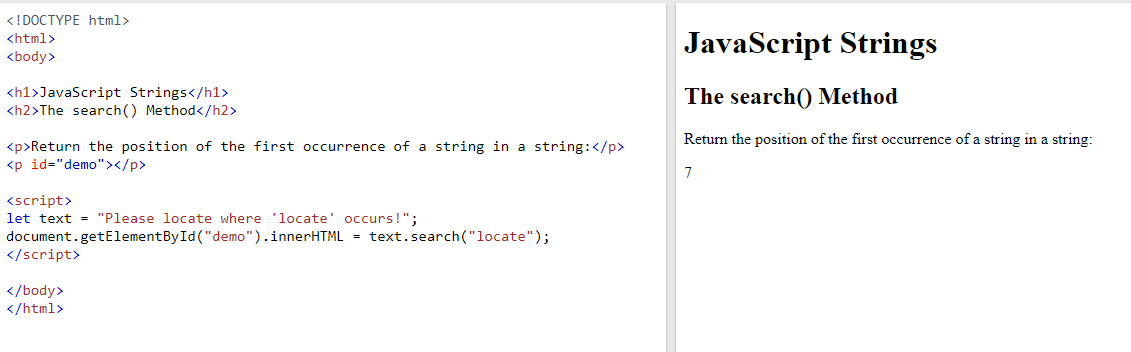


## JavaScript String search()

The search() method searches a string for a string (or a regular expression) and returns the position of the match:

### Examples

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



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

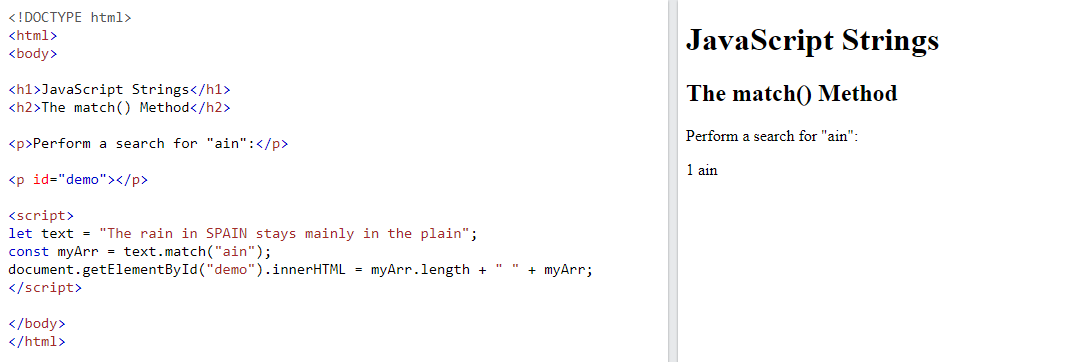
## JavaScript String match()

The match() method returns an array containing the results of matching a string against a string (or a regular expression).

### Examples

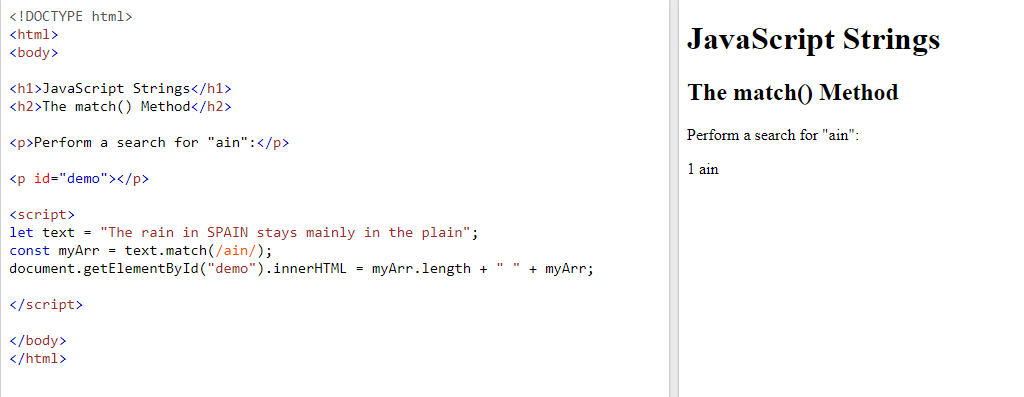
Perform a search for "ain":

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



Perform a search for "ain":

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



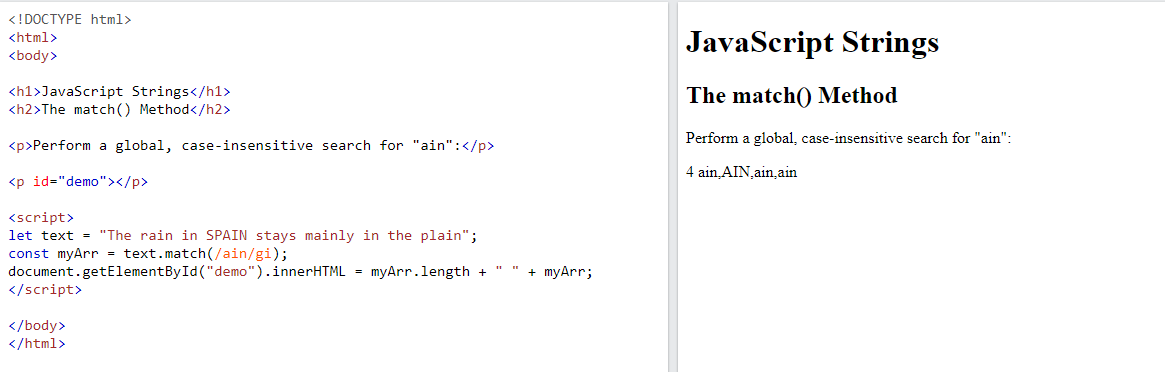
Perform a global search for "ain":

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



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

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



## Note

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

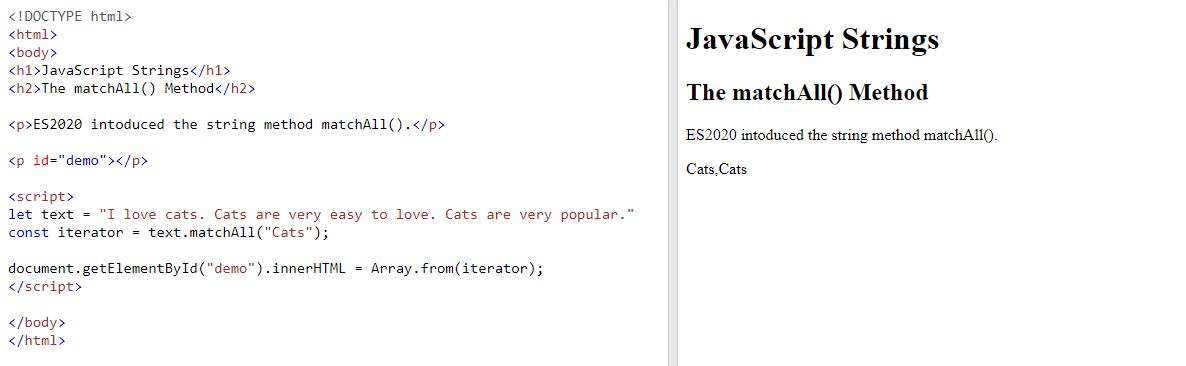
Read more about regular expressions in the chapter [JS RegExp](https://www.w3schools.com/js/js_regexp.asp).

## JavaScript String matchAll()

The matchAll() method returns an iterator containing the results of matching a string against a string (or a regular expression).

### Example

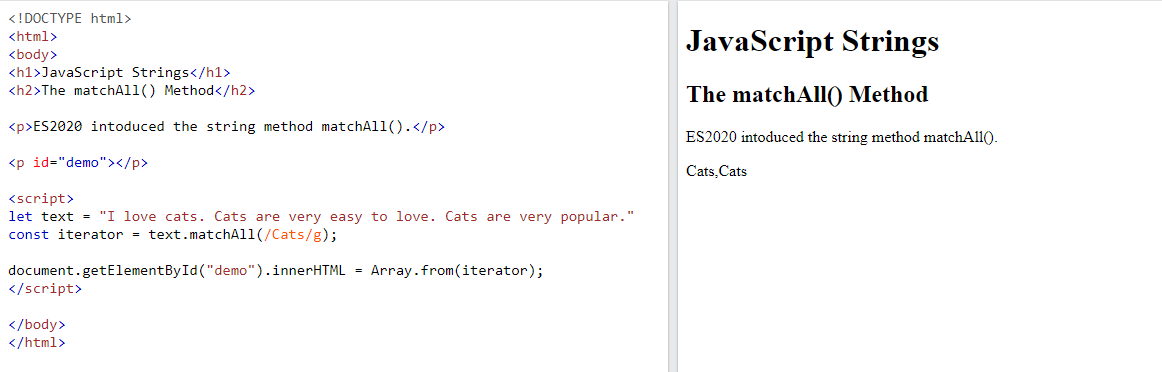
const iterator = text.matchAll("Cats");



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

### Example

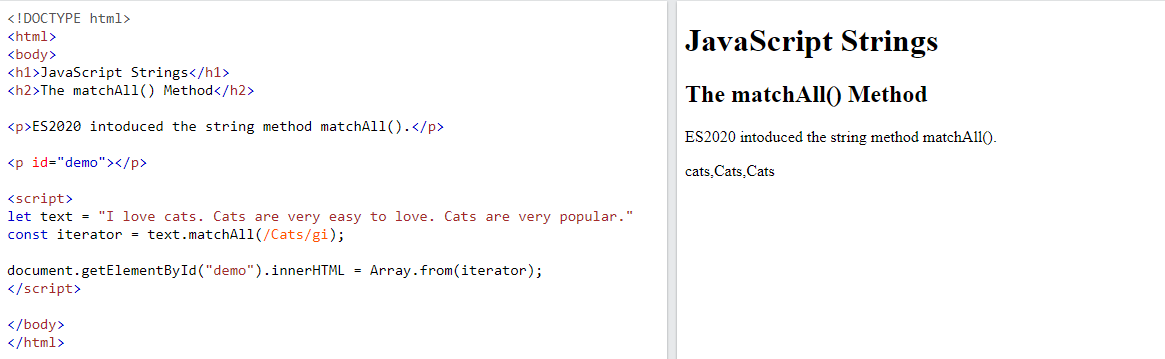
const iterator = text.matchAll(/Cats/g);



If you want to search case insensitive, the insensitive flag (i) must be set:

### Example

const iterator = text.matchAll(/Cats/gi);



## Notes

matchAll() is an [ES2020](https://www.w3schools.com/js/js_2020.asp) feature.

matchAll() does not work in Internet Explorer.

## JavaScript String includes()

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

Otherwise it returns false.

### Examples

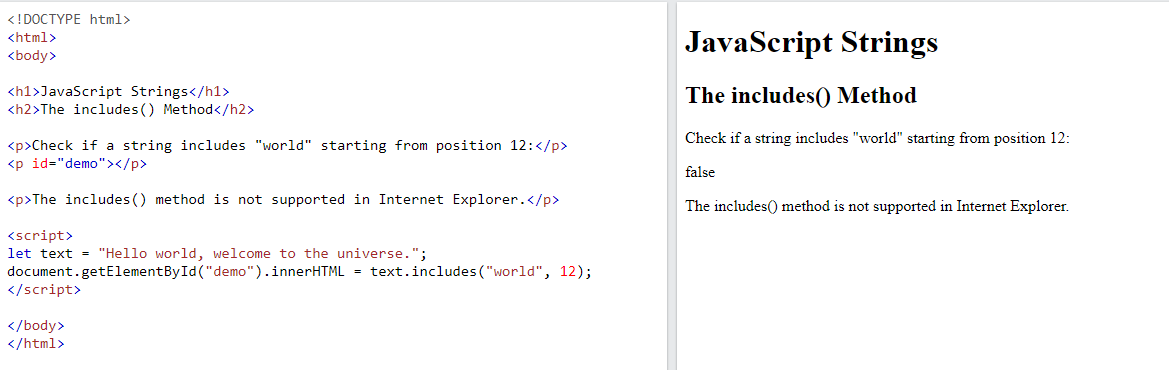
Check if a string includes "world":

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



Check if a string includes "world". Start at position 12:

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



## Notes

includes() is case sensitive.

includes() is an [ES6 feature](https://www.w3schools.com/js/js_es6.asp).

includes() is not supported in Internet Explorer.

## JavaScript String startsWith()

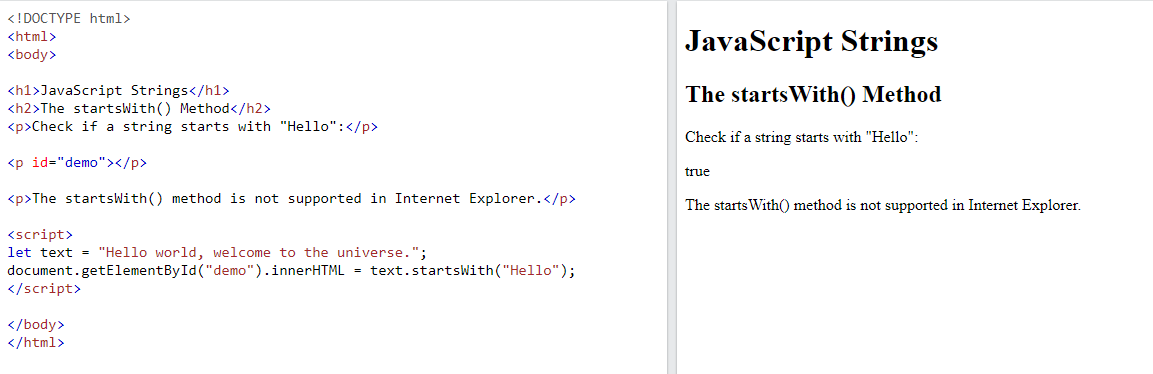
The startsWith() method returns true if a string begins with a specified value.

Otherwise it returns false:

### Examples

Returns true:

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



Returns false:

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



A start position for the search can be specified:

Returns false:

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



Returns true:

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



## Notes

startsWith() is case sensitive.

startsWith() is an [ES6 feature](https://www.w3schools.com/js/js_es6.asp).

startsWith() is not supported in Internet Explorer.

## JavaScript String endsWith()

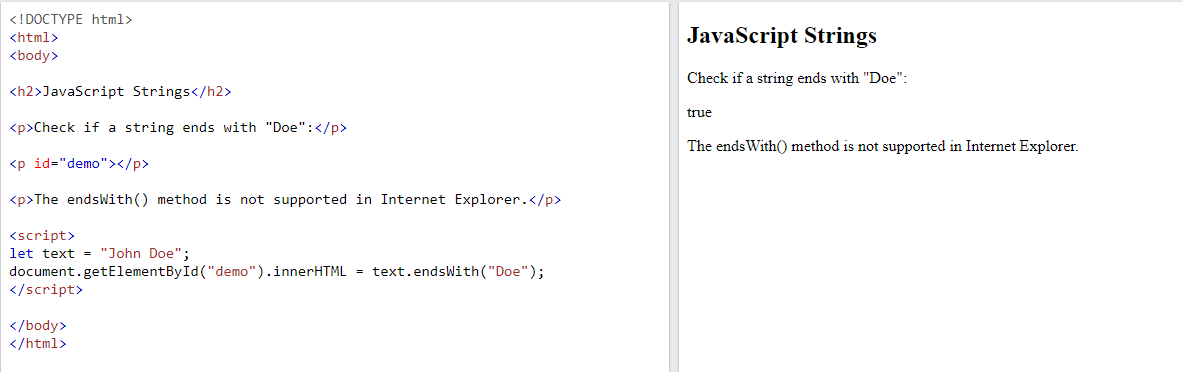
The endsWith() method returns true if a string ends with a specified value.

Otherwise it returns false:

### Examples

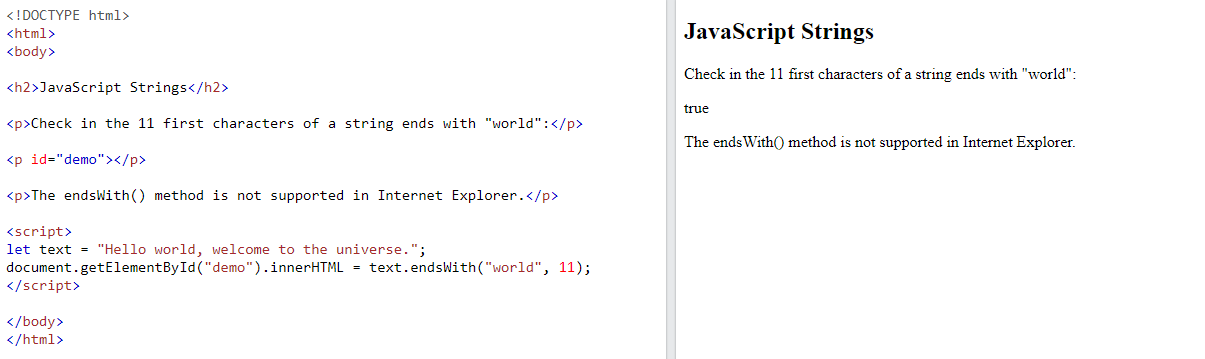
Check if a string ends with "Doe":

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



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

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



## Notes

endsWith() is case sensitive.

endsWith() is an [ES6 feature](https://www.w3schools.com/js/js_es6.asp).

endsWith() is not supported in Internet Explorer.