**Using regular expressions in JavaScript**

Regular expressions are used with the RegExp methods test() and exec() and with the String methods match(), replace(), search(), and split(). These methods are explained in detail in the [JavaScript reference](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference).

| **Methods that use regular expressions** | |
| --- | --- |
| **Method** | **Description** |
| [exec()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/exec) | Executes a search for a match in a string. It returns an array of information  or null on a mismatch. |
| [test()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/test) | Tests for a match in a string. It returns true or false. |
| [match()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/match) | Returns an array containing all of the matches, including capturing groups,  or null if no match is found. |
| [matchAll()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/matchAll) | Returns an iterator containing all of the matches, including capturing groups. |
| [search()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/search) | Tests for a match in a string. It returns the index of the match, or -1 if the search fails. |
| [replace()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/replace) | Executes a search for a match in a string, and replaces the matched substring with  a replacement substring. |
| [split()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/split) | Uses a regular expression or a fixed string to break a string into an array of substrings. |

When you want to know whether a pattern is found in a string, use the test() or search() methods; for more information (but slower execution) use the exec() or match() methods. If you use exec() or match() and if the match succeeds, these methods return an array and update properties of the associated regular expression object and also of the predefined regular expression object, RegExp. If the match fails, the exec() method returns null (which coerces to false).

In the following example, the script uses the exec() method to find a match in a string.

var myRe = /d(b+)d/g;

var myArray = myRe.exec('cdbbdbsbz');

If you do not need to access the properties of the regular expression, an alternative way of creating myArray is with this script:

var myArray = /d(b+)d/g.exec('cdbbdbsbz');

// similar to "cdbbdbsbz".match(/d(b+)d/g); however,

// "cdbbdbsbz".match(/d(b+)d/g) outputs Array [ "dbbd" ], while

// /d(b+)d/g.exec('cdbbdbsbz') outputs Array [ 'dbbd', 'bb', index: 1, input: 'cdbbdbsbz' ].

(See [different behaviors](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions#g-different-behaviors) for further info about the different behaviors.)

If you want to construct the regular expression from a string, yet another alternative is this script:

var myRe = new RegExp('d(b+)d', 'g');

var myArray = myRe.exec('cdbbdbsbz');

With these scripts, the match succeeds and returns the array and updates the properties shown in the following table.

As shown in the second form of this example, you can use a regular expression created with an object initializer without assigning it to a variable. If you do, however, every occurrence is a new regular expression. For this reason, if you use this form without assigning it to a variable, you cannot subsequently access the properties of that regular expression. For example, assume you have this script:

var myRe = /d(b+)d/g;

var myArray = myRe.exec('cdbbdbsbz');

console.log('The value of lastIndex is ' + myRe.lastIndex);

// "The value of lastIndex is 5"

However, if you have this script:

var myArray = /d(b+)d/g.exec('cdbbdbsbz');

console.log('The value of lastIndex is ' + /d(b+)d/g.lastIndex);

// "The value of lastIndex is 0"

The occurrences of /d(b+)d/g in the two statements are different regular expression objects and hence have different values for their lastIndex property. If you need to access the properties of a regular expression created with an object initializer, you should first assign it to a variable.

| **Regular expression flags** | | |
| --- | --- | --- |
| **Flag** | **Description** | **Corresponding property** |
| g | Global search. | [RegExp.prototype.global](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/global) |
| i | Case-insensitive search. | [RegExp.prototype.ignoreCase](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/ignoreCase) |
| m | Multi-line search. | [RegExp.prototype.multiline](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/multiline) |
| s | Allows . to match newline characters. (Added in ES2018, not yet supported in Firefox). | [RegExp.prototype.dotAll](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/dotAll) |
| u | "unicode"; treat a pattern as a sequence of unicode code points. | [RegExp.prototype.unicode](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/unicode) |
| y | Perform a "sticky" search that matches starting at the current position in the target string. See [sticky](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/sticky). | [RegExp.prototype.sticky](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/sticky) |

To include a flag with the regular expression, use this syntax:

var re = /pattern/flags;

or

var re = new RegExp('pattern', 'flags');