How to check whether a string contains a substring in JavaScript?

Asked 9 years, 8 months ago Active 24 days ago Viewed 5.6m times

7433

Usually I would expect a `String.contains()` method, but there doesn't seem to be one.

What is a reasonable way to check for this?

javascript string substring string-matching

1399

edited Apr 3 at 13:17

community wiki
29 revs, 19 users 18%
Peter O.

locked by Samuel Liew ♦ Apr 5 '18 at 6:46

This question's answers are a collaborative effort. If you see something that can be improved, just edit the answer to improve it! *No additional answers can be added here.*

Read more about locked posts here.

3 Answers

12914

String#includes()

ES6 introduced `String.prototype.includes` :

```
var string = "foo",
  substring = "oo";

console.log(string.includes(substring))
```

Run code snippet Expand snippet

`includes` doesn't have IE support, though.

String#indexOf()

In an ES5 or older environments, `String.prototype.indexOf` returns the index of a substring (or -1 if not found):

```
var string = "foo",
  substring = "oo";

console.log(string.indexOf(substring) !== -1)
```

Run code snippet Expand snippet

RegExp#test()

More advanced users may prefer RegExp#test, which allows for testing for against regular expressions:

```
var string = "foo",
  regex = /oo/;

console.log(regex.test(string));
```

Run code snippet Expand snippet

edited Jul 6 at 4:01

community wiki
35 revs, 28 users 14%
Ry-

26 Not supporting IE is a feature though – kgbph May 21 at 9:15

1 I don't like IE either, but if you have two functions that are largely identical, and one is better supported than the other one, I think you should pick the better supported one? So `indexOf()` it is... – rob74 May 24 at 11:18

@rob74 If you are concerned about performance, use `test` – Abandoned Cart Jun 26 at 2:42

1 Is it possible to do a case insensitive search? – Eric McWinNEr Jun 26 at 16:32

1 @EricMcWinNEr `/regexpattern/i.test(str)` --> i flag stands for case insensitivity – Code Maniac Jul 6 at 4:23

462

There is a `String.prototype.includes` in ES6:

```
"potato".includes("to");
> true
```

Note that this does not work in Internet Explorer or some other old browsers with no or incomplete ES6 support. To make it work in old browsers, you may wish to use a transpiler like Babel, a shim library like es6-shim, or this polyfill from MDN:

```
if (!String.prototype.includes) {
  String.prototype.includes = function(search, start) {
    'use strict';
    if (typeof start !== 'number') {
      start = 0;
    }

    if (start + search.length > this.length) {
      return false;
    } else {
      return this.indexOf(search, start) !== -1;
    }
  };
}
```

edited Apr 13 at 15:29

Mark Amery
70.5k 33 273 319

answered Jan 7 '13 at 10:23

eliocs
13.2k 6 35 47

Just do `"potato".includes("to");` and run it through Babel. – Derk Jan Speelman Jun 3 at 14:40

Another alternative is KMP.

9

The KMP algorithm searches for a length-*m* substring in a length-*n* string in worst-case O(*n*+*m*) time, compared to a worst case of O(*n*·*m*) for the naive algorithm, so using KMP may be reasonable if you care about worst-case time complexity.

Here's a JavaScript implementation by Project Nayuki, taken from https://www.nayuki.io/res/knuth-morris-pratt-string-matching/kmp-string-matcher.js:

```
// Searches for the given pattern string in the given text string using the Knuth-
Morris-Pratt string matching algorithm.
// If the pattern is found, this returns the index of the start of the earliest match in
'text'. Otherwise -1 is returned.
function kmpSearch(pattern, text) {
    if (pattern.length == 0)
        return 0;  // Immediate match

    // Compute longest suffix-prefix table
    var lsp = [0];  // Base case
    for (var i = 1; i < pattern.length; i++) {
        var j = lsp[i - 1];  // Start by assuming we're extending the previous LSP
        while (j > 0 && pattern.charAt(i) != pattern.charAt(j))
            j = lsp[j - 1];
        if (pattern.charAt(i) == pattern.charAt(j))
            j++;
        lsp.push(j);
    }

    // Walk through text string
    var j = 0;  // Number of chars matched in pattern
    for (var i = 0; i < text.length; i++) {
        while (j > 0 && text.charAt(i) != pattern.charAt(j))
            j = lsp[j - 1];  // Fall back in the pattern
        if (text.charAt(i) == pattern.charAt(j)) {
            j++;  // Next char matched, increment position
            if (j == pattern.length)
                return i - (j - 1);
        }
    }
    return -1;  // Not found
}
```

Example usage:

```
kmpSearch('ays', 'haystack') != -1 // true
kmpSearch('asdf', 'haystack') != -1 // false
```

edited May 19 at 18:44

community wiki
6 revs, 2 users 83%
Mark Amery