How to replace all occurrences of a string?

Asked 10 years, 2 months ago Active 4 days ago Viewed 3.1m times

3975

708

I have this string:

```
"Test abc test test abc test test test abc test test abc"
```

Doing:

```
str = str.replace('abc', '');
```

seems to only remove the first occurrence of `abc` in the string above.

How can I replace **all** occurrences of it?

javascript string replace

edited Jul 15 at 15:50
Alexander Abakumov
5,799 5 50 81

asked Jul 17 '09 at 17:53
Click Upvote
106k 232 531 703

When replacing all occurrences of `aba` in `ababa` with `ca` , which result do you expect? `caba` ? `abca` ? `cca` ? – reinierpost Aug 2 at 12:58

57 Answers

1 2 next

For the sake of completeness, I got to thinking about which method I should use to do this. There are basically two ways to do this as

Note: In general, extending the built-in prototypes in JavaScript is generally not recommended. I am providing as extensions on the String prototype simply for purposes of illustration, showing different implementations of a hypothetical standard method on the `String` built-in prototype.

Regular Expression Based Implementation

```
String.prototype.replaceAll = function(search, replacement) {
    var target = this;
    return target.replace(new RegExp(search, 'g'), replacement);
};
```

Split and Join (Functional) Implementation

```
String.prototype.replaceAll = function(search, replacement) {
    var target = this;
    return target.split(search).join(replacement);
};
```

Not knowing too much about how regular expressions work behind the scenes in terms of efficiency, I tended to lean toward the split and join implementation in the past without thinking about performance. When I did wonder which was more efficient, and by what margin, I used it as an excuse to find out.

On my Chrome Windows 8 machine, **the regular expression based implementation is the fastest**, with the **split and join implementation being 53% slower**. Meaning the regular expressions are twice as fast for the lorem ipsum input I used.

Check out this **benchmark** running these two implementations against each other.

As noted in the comment below by @ThomasLeduc and others, there could be an issue with the regular expression-based implementation if `search` contains certain characters which are reserved as special characters in regular expressions. The implementation assumes that the caller will escape the string beforehand or will only pass strings that are without the characters in the table in *Regular Expressions* (MDN).

MDN also provides an implementation to escape our strings. It would be nice if this was also standardized as `RegExp.escape(str)`, but alas, it does not exist:

```
function escapeRegExp(str) {
  return str.replace(/[.*+?^${}()|[\]\\]/g, "\\$&"); // $& means the whole matched
string
}
```

We could call `escapeRegExp` within our `String.prototype.replaceAll` implementation, however, I'm not sure how much this will affect the performance (potentially even for strings for which the escape is not needed, like all alphanumeric strings).

edited May 27 '17 at 21:55
Peter Mortensen
14.5k 19 89 118

answered Jul 12 '13 at 1:46
Cory Gross
31k 15 56 75

8 On Android 4.1 the regexp method is 15% faster, but you are not escaping the expression to search for, so this benchmark is incomplete. – andreszs Jan 19 '15 at 0:01

31 There is an issue with this solution, if you're search string appears to contains special characters of a regexp expression, they will be interpreted. I recommend the @Sandy Unitedwolf answer. – Thomas Leduc Feb 17 '16 at 16:09

2 The first one will do this: 'bla.bla'.replaceAll('.', '_'); "_______". The second one will do 'bla_bla', which is more generally what you want to do. – RobKohr Feb 21 '16 at 1:12

1 @ThomasLeduc it looks like the issue you mentioned can be worked around with lodash => lodash.com/docs/4.17.5#escapeRegExp – mattnificent Mar 15 '18 at 7:47

1 Leaving this here for future viewers who are also wondering why extending the native object is a bad idea: stackoverflow.com/questions/14034180/… – Ben Cooper Apr 6 '18 at 19:28

4078

```
str = str.replace(/abc/g, '');
```

In response to comment:

```
var find = 'abc';
var re = new RegExp(find, 'g');

str = str.replace(re, '');
```

```
function replaceAll(str, find, replace) {
    return str.replace(new RegExp(find, 'g'), replace);
}
```

Note: Regular expressions contain special (meta) characters, and as such it is dangerous to blindly pass an argument in the `find` function above without pre-processing it to escape those characters. This is covered in the Mozilla Developer Network's JavaScript Guide on Regular Expressions, where they present the following utility function:

```
function escapeRegExp(str) {
    return str.replace(/([.*+?^=!:${}()|\[\]\/\\])/g, "\\$1");
}
```

So in order to make the `replaceAll()` function above safer, it could be modified to the following if you also include `escapeRegExp`:

```
function replaceAll(str, find, replace) {
    return str.replace(new RegExp(escapeRegExp(find), 'g'), replace);
}
```

edited Jun 21 '17 at 22:17
MC Emperor
10.5k 12 56 93

answered Jul 17 '09 at 17:54
Unimportant
98.3k 15 122 133

3 Regular expressions are the only way to accomplish this "out of the box" without implementing your own 'replaceAll' method. I'm not suggesting their use just for the sake of using them. – Unimportant Jul 17 '09 at 19:47

10 This is my own function from this comment: `function replaceAll(find, replace,str) { var re = new RegExp(find, 'g'); str = str.replace(re, replace); return str; }` – Click Upvote Jul 17 '09 at 21:11

40 Major caveat of of the `replaceAll` implementation is that it won't work if `find` contains metacharacters. – Martin Ender Aug 27 '13 at 0:39

1 @SeanBright you're right. I used your javascript in a php code so I had to add an extra backslash to escape. On the fiddle your code is perfect. I should have checked. Apologies jsfiddle.net/7y19xyq8 – Onimusha Jul 2 '15 at 16:38

2 @Bennett it's bad practice to extend the prototype of JavaScript native types. – Emile Bergeron Sep 12 '17 at 15:40

Note: Don't use this in real code

1284

```
str = "Test abc test test abc test...".split("abc").join("");
```

The general pattern is

```
str.split(search).join(replacement)
```

This used to be faster in some cases than using `replaceAll` and a regular expression, but that doesn't seem to be the case anymore in modern browsers. So, this should really only be used as a quick hack to avoid needing to escape the regular expression, not in real code.

edited Jan 24 '17 at 14:38

answered Jul 17 '09 at 20:29
Matthew Crumley
82.7k 21 97 120

133 It surprised me, as I would expect this method to be allocating more objects, creating more garbage, and thus take longer, but when I actually tested in a browser this method was consistently about 20% faster than the accepted response. Results may vary, of course. – MgSam Sep 20 '13 at 20:22

38 I was curious myself, and set up this: jsperf.com/replace-all-vs-split-join . It seems that v8 is just crazy fast at splitting/joining arrays compared to other javascript engines. – fabi Nov 23 '13 at 21:02

7 Very nice - also saves you from the possibility of bad RegExps when passing in special characters. I'm adding in a generic flag for "find this and replace it" in an object property, but was concerned if I needed to replace "." or "}" and forgot I was using RegEx 3 months later! – tobriand Apr 3 '14 at 8:01

5 And as String.prototype: `String.prototype.replaceAll = function(f,r){return this.split(f).join(r);}` . Usage: `"My string".replaceAll('st', '');` produces "My ring" – MacroMan Sep 26 '14 at 11:50

5 Great answer! Out of interest, why do you advise not to use this in real code? – Steve Chambers Apr 21 '17 at 12:57

617

Using a regular expression with the `g` flag set will replace all:

```
someString = 'the cat looks like a cat';
anotherString = someString.replace(/cat/g, 'dog');
// anotherString now contains "the dog looks like a dog"
```

edited May 17 '16 at 12:15
R3tep
9,384 8 31 64

answered May 6 '09 at 23:18
Adam A
11.9k 5 25 33

13 Kinda silly I think, but the JS global regex is the only way to do multiple replaces. – Mike May 7 '09 at 2:43

3 well technically you can loop through `var sourceText` count the number of instances (`numOfInstances`) using `substring` or split and count the length (among other strategies) of `thatWhichYouWantToReplace` then do `for (var i = 0; i < numOfInstances; i++){ sourceText = sourceText.replace('thatWhichYouWantToReplace', ''); }` or even easier just use a while loop (`while sourceText.indexOf(thatWhichYouWantToReplace > -1){ sourceText = sourceText.replace(...)`) but I don't see why you would want to do it that way when using `/g` is so easy and probably more performant. – Zargold Mar 20 '16 at 22:20

98

Here's a string prototype function based on the accepted answer:

```
String.prototype.replaceAll = function (find, replace) {
    var str = this;
    return str.replace(new RegExp(find, 'g'), replace);
};
```

EDIT

If your `find` will contain special characters then you need to escape them:

```
String.prototype.replaceAll = function (find, replace) {
    var str = this;
    return str.replace(new RegExp(find.replace(/[-\/\\^$*+?.()|[\]{}]/g, '\\$&'), 'g'),
replace);
};
```

Fiddle: http://jsfiddle.net/cdbzL/

edited Oct 7 '13 at 16:40

answered Feb 11 '13 at 23:03
jesal
5,683 3 44 50

8 [Don't modify objects you don't own](#) – Aamir Afridi Oct 9 '14 at 9:59

jesal, this is a wonderful solution to a string replacement problem I've encountered, and it apparently overcomes the "immutability" of strings in JavaScript. Could you or someone else explain how this prototype function overrides the immutability of strings? This *does* work; I just want to understand this ancillary question it poses. – Tom Jan 12 '16 at 0:08

1 @Tom: It *doesn't* overcome the immutability at all: `var str = this` creates a second reference to the immutable string, to which the `replace` method is applied, which in turn returns a *new immutable string*. these prototype functions return a new immutable string, if not, you'd be able to write `someStrVar.replaceAll('o', '0');` and `someStrVar` would be changed. Instead, you have to write `someStrVar = someStrVar.replaceAll('o', '0');` <-- reassign to set the var to hold the *new immutable string*. there's no way around that. Try in console: `x = 'foobar'; x.replaceAll('o', '0'); x;` – Elias Van Ootegem Jun 15 '16 at 16:26

83

Update:

It's somewhat late for an update, but since I just stumbled on this question, and noticed that my previous answer is not one I'm happy with. Since the question involved replaceing a single word, it's incredible nobody thought of using word boundaries (`\b`)

```
'a cat is not a caterpillar'.replace(/\bcat\b/gi,'dog');
//"a dog is not a caterpillar"
```

This is a simple regex that avoids replacing parts of words in most cases. However, a dash `-` is still considered a word boundary. So conditionals can be used in this case to avoid replacing strings like `cool-cat` :

```
'a cat is not a cool-cat'.replace(/\bcat\b/gi,'dog');//wrong
//"a dog is not a cool-dog" -- nips
'a cat is not a cool-cat'.replace(/(?:\b([^-]))cat(?:\b([^-]))/gi,'$1dog$2');
//"a dog is not a cool-cat"
```

basically, this question is the same as the question here: [Javascript replace " ' " with " '' "](#)

@Mike, check the answer I gave there... regexp isn't the only way to replace multiple occurrences of a subsrting, far from it. Think flexible, think split!

```
var newText = "the cat looks like a cat".split('cat').join('dog');
var regText = "the cat looks like a cat".replace(/(?:(^|[^a-z]))(([^a-z]*)(?=cat)cat)(?!
[a-z])/gi,"$1dog");
```

The output is the same as the accepted answer, however, using the /cat/g expression on this string:

```
var oops = 'the cat looks like a cat, not a caterpillar or
coolcat'.replace(/cat/g,'dog');
//returns "the dog looks like a dog, not a dogerpillar or cooldog" ??
```

Oops indeed, this probably isn't what you want. What is, then? IMHO, a regex that only replaces 'cat' conditionally. (ie not part of a word), like so:

```
var caterpillar = 'the cat looks like a cat, not a caterpillar or coolcat'.replace(/(?:
(^|[^a-z]))(([^a-z]*)(?=cat)cat)(?![a-z])/gi,"$1dog");
//return "the dog looks like a dog, not a caterpillar or coolcat"
```

My guess is, this meets your needs. It's not fullproof, of course, but it should be enough to get you started. I'd recommend reading some more on these pages. This'll prove useful in perfecting this expression to meet your specific needs.

http://www.javascriptkit.com/jsref/regexp.shtml

http://www.regular-expressions.info

Final addition:

Given that this question still gets a lot of views, I thought I might add an example of `.replace` used with a callback function. In this case, it dramatically simplifies the expression *and* provides even more flexibility, like replacing with correct capitalisation or replacing both `cat` and `cats` in one go:

```
'Two cats are not 1 Cat! They\'re just cool-cats, you caterpillar'
   .replace(/(^|.\b)(cat)(s?\b.|$)/gi,function(all,char1,cat,char2)
    {
       //check 1st, capitalize if required
       var replacement = (cat.charAt(0) === 'C' ? 'D' : 'd') + 'og';
       if (char1 === ' ' && char2 === 's')
       {//replace plurals, too
           cat = replacement + 's';
```

```
            cat = char1 === '-' || char2 === '-' ? cat : replacement;
        }
        return char1 + cat + char2;//return replacement string
    });
//returns:
//Two dogs are not 1 Dog! They're just cool-cats, you caterpillar
```

edited May 23 '17 at 11:47
Community ♦
1 1

answered Mar 1 '12 at 10:02
Elias Van Ootegem
59.9k 9 86 126

I think the additional interesting part should placed at the bottom. ps.: I noticed just now that the half of the first line is out of the area, let me allow to fix that! – user669677 Jul 4 '13 at 16:31

59

Match against a global regular expression:

```
anotherString = someString.replace(/cat/g, 'dog');
```

answered May 6 '09 at 23:23
scronide
10.1k 3 24 31

49

These are the most common and readable methods.

```
var str = "Test abc test test abc test test test abc test test abc"
```

Method-01:

```
str = str.replace(/abc/g, "replaced text");
```

Method-02:

Method-03:

```
str = str.replace(new RegExp("abc", "g"), "replaced text");
```

Method-04:

```
while(str.includes("abc")){
    str = str.replace("abc", "replaced text");
}
```

Output:

```
console.log(str);
// Test replaced text test test replaced text test test test replaced text test test
replaced text
```

edited Apr 1 at 12:56

answered Mar 25 at 9:38
Adnan Toky
920 5 12

43

For replacing a single time use:

```
var res = str.replace('abc', "");
```

For replacing multiple times use:

```
var res = str.replace(/abc/g, "");
```

edited May 29 at 12:33
Mihai Chelaru
3,106 10 17 28

answered May 29 at 11:11
Indrajeet Singh
2,209 20 20

41

```
str = str.replace(/abc/g, '');
```

Or try the replaceAll function from here:

[What are useful JavaScript methods that extends built-in objects?]

```
str = str.replaceAll('abc', ''); OR

var search = 'abc';
str = str.replaceAll(search, '');
```

EDIT: Clarification about replaceAll availability

The 'replaceAll' method is added to String's prototype. This means it will be available for all string objects/literals.

E.g.

```
var output = "test this".replaceAll('this', 'that');  //output is 'test that'.
output = output.replaceAll('that', 'this'); //output is 'test this'
```

edited May 23 '17 at 12:34
Community ♦
1 1

answered Jul 17 '09 at 17:55
SolutionYogi
27.2k 10 64 77

2 Can you rewrite the replaceAll() function there for those not using prototype? – Click Upvote Jul 17 '09 at 17:57

@Click Upvote....you are using prototype, it's part of all JS objects. I think you are thinking of prototype.js the JS library. – seth Jul 17 '09 at 18:20

seth, a little correction. If you add method to a prototype, it is available to all objects of that type. The replceAll method was added to String prototype and it should work for all string objects. – SolutionYogi Jul 17 '09 at 18:23

@solutionyogi -- Yep, I've used prototype (correctly) before. I was just addressing the OP's comment about "not using prototype" which I assumed to mean Prototype.js (perhaps incorrectly?). I should have said "a prototype" as I was trying to say JavaScript objects have a prototype. Thus, the OP was already 'using prototype,' albeit in an "indirect" way. Indirect might be the wrong term to use here but I'm tired so mea culpa. – seth Jul 17 '09 at 23:41

37

Say you want to replace all the 'abc' with 'x':

```
let some_str = 'abc def def lom abc abc def'.split('abc').join('x')
console.log(some_str) //x def def lom x x def
```

I was trying to think about something more simple than modifying the string prototype.

edited May 27 '17 at 22:01
Peter Mortensen
14.5k 19 89 118

answered Sep 10 '16 at 14:59
Emilio Grisolía
746 1 7 13

1 Simple, easy, and probably the most performant option: nice answer. – machineghost Oct 25 '16 at 23:51

I don't know if it performance-wise, but.... it works. – Emilio Grisolía Jan 13 '17 at 18:18

Actually, while I haven't metric-ed it personally, split/join is usually a very performant solution. – machineghost Jan 13 '17 at 18:30

1 it's even in chrome, 100% faster on firefox, and 50% slower on IE...: jsperf.com/replace-regex-split-join – Olivier May 22 '18 at 14:39

33

Use a regular expression:

```
str.replace(/abc/g, '');
```

edited May 27 '17 at 21:56
Peter Mortensen
14.5k 19 89 118

answered Jul 17 '09 at 17:56
Donnie DeBoer
2,299 12 14

30

Replacing single quotes:

```
function JavaScriptEncode(text){
    text = text.replace(/'/g,'&apos;')
    // More encode here if required
```

edited Dec 17 '17 at 15:53
Termininja
4,696 12 32 39

answered Jun 19 '15 at 21:24
Chris Rosete
1,022 14 9

2 How can i change second line to replace strings? This doesn't work: text = text.replace('hey', 'hello'); Any idea? – Stefan Đorđević Dec 9 '16 at 22:37

2 Sure Stefan, here is the code... text = text.replace(/hey/g, 'hello'); – Chris Rosete Dec 12 '16 at 13:56

28

Using `RegExp` in **JavaScript** could do the job for you, just simply do something like below, don't forget the `/g` after which standout for **global**:

```
var str ="Test abc test test abc test test test abc test test abc";
str = str.replace(/abc/g, '');
```

If you think of reuse, create a function to do that for you, but it's not recommended as it's only one line function, but again if you heavily use this, you can write something like this:

```
String.prototype.replaceAll = String.prototype.replaceAll || function(string, replaced)
{
  return this.replace(new RegExp(string, 'g'), replaced);
};
```

and simply use it in your code over and over like below:

```
var str ="Test abc test test abc test test test abc test test abc";
str = str.replaceAll('abc', '');
```

But as I mention earlier, it won't make a huge difference in terms of lines to be written or performance, only caching the function may effect some faster performance on long strings and also a good practice of DRY code if you want to reuse.

edited Mar 12 at 14:48

answered Jun 6 '17 at 13:39
Alireza
62.5k 15 200 135

24

This is the **fastest** version that *doesn't use regular expressions*.

Revised jsperf

```
replaceAll = function(string, omit, place, prevstring) {
  if (prevstring && string === prevstring)
    return string;
  prevstring = string.replace(omit, place);
  return replaceAll(prevstring, omit, place, string)
}
```

It is almost **twice** as fast as the split and join method.

As pointed out in a comment here, this will not work if your `omit` variable contains `place`, as in: `replaceAll("string", "s", "ss")`, because it will always be able to replace another occurrence of the word.

There is another jsperf with variants on my recursive replace that go even faster (http://jsperf.com/replace-all-vs-split-join/12)!

- Update July 27th 2017: It looks like RegExp now has the fastest performance in the recently released Chrome 59.

edited Jul 27 '17 at 15:57

answered Apr 4 '14 at 18:49
Cole Lawrence
565 4 13

I loved your solution... I wish I could give you +10 but here's my +1. I think you can store index of the sub-string that was replaced and jump to the next if a match is found at a lower index to avoid that infinite loop problem. I can't comment about the performance because I didn't test it but that's just my 2 cents on this piece of excellence. – Fr0zenFyr Jul 10 '14 at 4:40

@fr0zenfyr if you want to check if omit is in place (to prevent infinite loop) you could do a conditional like `if(place.replace(omit) === omit) {` No match, so it's safe to use replace loop `} else {` Match so use different method like split and join `}` – Cole Lawrence Jul 10 '14 at 12:41

Hmm.. but whats the point combining two solutions? I'm anyway not a fan of split/join approach anyway.. thanks for the advise.. – Fr0zenFyr Jul 11 '14 at 4:14

@Fr0zenFyr I believe the purpose of combining the two solutions would be to fallback on a slower method if you can't use the faster one (when the loop would be infinite for example). So it would be a safe guard to ensure functionality with efficiency, but without possibility of failure. – Cole Lawrence Jul 11 '14 at 13:13

Flawed........... – momomo Jul 7 '17 at 9:52

23

//loop it until number occurrences comes to 0. OR simply copy/paste

```
function replaceAll(find, replace, str)
{
  while( str.indexOf(find) > -1)
  {
    str = str.replace(find, replace);
  }
  return str;
}
```

edited Sep 5 '13 at 14:01
Gonzalo Larralde
3,162 19 29

answered Jun 5 '13 at 4:57
Raseela
247 2 2

23 This method is dangerous, do not use it. If the replacement string contains the search keyword, then an infinite loop will occur. At the very least, store the result of `.indexOf` in a variable, and use this variable as the second parameter of `.indexOf` (minus length of keyword, plus length of replacement string). – Rob W Oct 29 '13 at 11:33

I'm thinking about first replacing the search pattern with a weired unicode char, of that we are sure it is not used in the input string. Like U+E000 in the private area. And then replace it back to the target. I've build that here.. I'm not sure if thats a good Idea. – Lux Jul 2 '14 at 13:42

23

```
str = str.replace(new RegExp("abc", 'g'), "");
```

worked better for me than the above answers. so `new RegExp("abc", 'g')` creates a RegExp what matches all occurence (`'g'` flag) of the text (`"abc"`). The second part is what gets replaced to, in your case empty string (`""`). `str` is the string, and we have to override it, as `replace(...)` just returns result, but not overrides. In some cases you might want to use that.

edited Dec 29 '17 at 17:52

answered Dec 28 '17 at 12:27
csomakk
4,157 1 20 31

While this code snippet may be the solution, including an explanation really helps to improve the quality of your post. Remember that you are answering the question for readers in the future, and those people might not know the reasons for your code suggestion. – yivi Dec 28 '17 at 15:46

20

If what you want to find is already in a string, and you don't have a regex escaper handy, you can use join/split:

```
function replaceMulti(haystack, needle, replacement)
{
    return haystack.split(needle).join(replacement);
}

someString = 'the cat looks like a cat';
console.log(replaceMulti(someString, 'cat', 'dog'));
```

Run code snippet   Expand snippet

edited Jul 23 '18 at 17:42
Pitu
40 6

answered Jun 19 '13 at 23:21
rakslice
6,305 3 42 49

thanks! This solution works perfectly for my problem :) – xero399 Aug 27 at 7:08

17

```
function replaceAll(str, find, replace) {
  var i = str.indexOf(find);
  if (i > -1){
    str = str.replace(find, replace);
    i = i + replace.length;
    var st2 = str.substring(i);
    if(st2.indexOf(find) > -1){
      str = str.substring(0,i) + replaceAll(st2, find, replace);
    }
  }
  return str;
}
```

I wonder how well that performs ... substring is native or walks over char array to create the new characters. – momomo Jul 7 '17 at 9:53

15

I like this method (it looks a little cleaner):

```
text = text.replace(new RegExp("cat","g"), "dog");
```

edited May 27 '17 at 21:57
Peter Mortensen
14.5k 19 89 118

answered May 15 '13 at 14:03
Owen
2,280 4 34 44

1 Okay, how do you escape the string to use it as a regex pattern? – rakslice Jun 19 '13 at 21:16

I dont, I just use it for plain text – Owen Mar 19 at 11:58

12

```
var str = "ff ff f f a de def";
str = str.replace(/f/g,'');
alert(str);
```

http://jsfiddle.net/ANHR9/

answered Sep 4 '13 at 10:01
pkdkk
1,874 7 32 60

what was the point of a fiddle that only contains the same javascript – JGallardo Feb 8 '17 at 0:42

```
while (str.indexOf('abc') !== -1)
```

answered Apr 29 '14 at 10:25
zdennis
147 1 2

12

If the string contain similar pattern like `abccc` , you can use this:

```
str.replace(/abc(\s|$)/g, "")
```

edited Apr 27 '18 at 8:49

community wiki
3 revs, 2 users 78%
mostafa elmadany

11

The previous answers are way too complicated. Just use the replace function like this:

```
str.replace(/your_regex_pattern/g, replacement_string);
```

Example:

```
var str = "Test abc test test abc test test test abc test test abc";

var res = str.replace(/[abc]+/g, "");

console.log(res);
```

Run code snippet   Expand snippet

edited Nov 18 '18 at 3:27
Peter Mortensen
14.5k 19 89 118

answered Oct 23 '18 at 9:34
Black
5,622 15 68 139

11

The simplest way to this without using any regex is split and join like code here :

```
var str="Test abc test test abc test test test abc test test abc";
str.split('abc').join('')
```

answered May 14 at 18:42

sajadre

393 1 4 17

9

If you are trying to ensure that the string you are looking for won't exist even after the replacement, you need to use a loop.

For example:

```
var str = 'test aabcbc';
str = str.replace(/abc/g, '');
```

When complete, you will still have 'test abc'!

The simplest loop to solve this would be:

```
var str = 'test aabcbc';
while (str != str.replace(/abc/g, '')){
   str.replace(/abc/g, '');
}
```

But that runs the replacement twice for each cycle. Perhaps (at risk of being voted down) that can be combined for a slightly more efficient but less readable form:

```
var str = 'test aabcbc';
while (str != (str = str.replace(/abc/g, ''))){}
// alert(str); alerts 'test '!
```

This can be particularly useful when looking for duplicate strings.
For example, if we have 'a,,,b' and we wish to remove all duplicate commas.

SamGoody
9,146 8 60 76

9

Although people have mentioned the use of regex but there's a better approach if you want to replace the text irrespective of the case of the text. Like uppercase or lowercase. Use below syntax

```
//Consider below example
originalString.replace(/stringToBeReplaced/gi, '');

//Output will be all the occurrences removed irrespective of casing.
```

You can refer the detailed example here.

edited Jun 8 '16 at 18:51
evandrix
5,031 3 24 31

answered May 3 '16 at 18:39
Cheezy Code
1,278 9 14

from the example site: "/toBeReplacedString/gi is the regex you need to use. Here g represents for global match and i represents case insensitive. By default regex is case sensitive" – alikuli Aug 3 '16 at 10:15

8

Just add `/g`

```
document.body.innerHTML = document.body.innerHTML.replace('hello', 'hi');
```

to

```
// Replace 'hello' string with /hello/g regular expression.
document.body.innerHTML = document.body.innerHTML.replace(/hello/g, 'hi');
```

`/g` means global

edited Oct 15 '15 at 12:13

answered Jun 5 '15 at 7:16

8

I have solved this problem by a simple line of code.

```
str.replace(/Current string/g, "Replaced string");
```

Check example on jsfiddle https://jsfiddle.net/pot6whnx/1/

answered Mar 7 at 4:33
Riajul Islam
593 6 11

7

You can simply use below method

```
/**
 * Replace all the occerencess of $find by $replace in $originalString
 * @param  {originalString} input - Raw string.
 * @param  {find} input - Target key word or regex that need to be replaced.
 * @param  {replace} input - Replacement key word
 * @return {String}       Output string
 */
function replaceAll(originalString, find, replace) {
  return originalString.replace(new RegExp(find, 'g'), replace);
};
```

edited Feb 23 '16 at 3:47

answered Feb 15 '16 at 12:05
tk_
9,175 4 57 73

1 2 next