Creating multiline strings in JavaScript

Ask Question

2163

368

I have the following code in Ruby. I want to convert this code into JavaScript. what's the equivalent code in JS?

```
text = <<"HERE"
This
Is
A
Multiline
String
HERE
```

javascript string multiline heredoc

edited May 8 '13 at 15:37
Potherca
7,429 3 48 66

asked Apr 30 '09 at 2:11
Newy
13k 7 34 52

36 Answers

1 2 next

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

Teams
Q&A for work

Learn More

2650

Update:

ECMAScript 6 (ES6) introduces a new type of literal, namely **template literals**. They have many features, variable interpolation among others, but most importantly for this question, they can be multiline.

A template literal is delimited by *backticks*:

```
var html = `
  <div>
    <span>Some HTML here</span>
  </div>
`;
```

(Note: I'm not advocating to use HTML in strings)

Browser support is OK, but you can use transpilers to be more compatible.

Original ES5 answer:

Javascript doesn't have a here-document syntax. You can escape the literal newline, however, which comes close:

```
"foo \
bar"
```

edited Feb 14 '18 at 16:47
Andy Mercer
4,069 3 29 61

answered Apr 30 '09 at 2:15
Anonymous

31k 1 20 19

206 Be warned: some browsers will insert newlines at the continuance, some will not. – staticsan Apr 30 '09 at 2:22

31 Visual Studio 2010 seems to be confused by this syntax as well. – jcollum Apr 17 '11 at 21:58

47 @Nate It is specified in ECMA-262 5th Edition section 7.8.4 and called *LineContinuation* : "A line terminator character cannot appear in a string literal, except as part of a *LineContinuation* to produce the empty character sequence. The correct way to cause a line terminator character to be part of the String value of a string literal is to use an escape sequence such as \n or \u000A." – some Sep 25 '12 at 2:28

7 So in summary it seems this is the most straightforward approach, compatible with all browsers at least back to IE6 (and probably earlier), as long as you don't care whether or not extra newlines might be added at the end of each line. Does anyone know which browsers/versions add newlines and which don't? – Matt Browne Feb 26 '13 at 18:35

14 I don't see why you'd do this when browsers treat it inconsistently. "line1\n" + "line2" across multiple lines is readable enough and you're guaranteed consistent behavior. – SamStephens Mar 20 '13 at 20:14

1193

ES6 Update:

As the first answer mentions, with ES6/Babel, you can now create multi-line strings simply by using backticks:

```
const htmlString = `Say hello to
multi-line
strings!`;
```

Interpolating variables is a popular new feature that comes with back-tick delimited strings:

```
const htmlString = `${user.name} liked your post about strings`;
```

This just transpiles down to concatenation:

```
user.name + ' liked your post about strings'
```

Original ES5 answer:

> Google's JavaScript style guide recommends to use string concatenation instead of escaping newlines:
>
> **Do not do this:**
>
> ```
> var myString = 'A rather long string of English text, an error me
>                 actually that just keeps going and going -- an er
>                 message to make the Energizer bunny blush (right
>                 those Schwarzenegger shades)! Where was I? Oh yes
>                 you\'ve got an error and all the extraneous white
>                 just gravy.  Have a nice day.';
> ```
>
> The whitespace at the beginning of each line can't be safely stripped at compile time; whitespace after the slash will result in tricky errors; and while most script engines support this, it is not part of ECMAScript.
>
> **Use string concatenation instead:**
>
> ```
> var myString = 'A rather long string of English text, an error me
>                'actually that just keeps going and going -- an er
>                'message to make the Energizer bunny blush (right
>                'those Schwarzenegger shades)! Where was I? Oh yes
>                'you\'ve got an error and all the extraneous white
>                'just gravy.  Have a nice day.';
> ```

edited Oct 2 '18 at 19:26

answered Jun 6 '11 at 2:30

Devin G Rhode
14.6k 5 27 43

18 I don't understand Google's recommendation. All browsers except extremely old ones support the backslash followed by newline approach, and will continue to do so in the future for backward compatibility. The only time you'd need to avoid it is if you needed to be sure that one and only one newline (or no newline) was added at the end of each line (see also my comment on the accepted answer). – Matt Browne Feb 26 '13 at 18:40

4 Note that template strings aren't supported in IE11, Firefox 31, Chrome 35, or Safari 7. See kangax.github.io/compat-table/es6 – Dwayne May 24 '14 at 2:41

24 @MattBrowne Google's recommendation is already documented by them, in order of importance of reasons: (1) The whitespace at the beginning of each line [in the example, you don't want that whitespace in your string but it looks nicer in the code] (2) whitespace after the slash will result in tricky errors [if you end a line with `\ ` instead of `\` it's hard to notice] and (3) while most script engines support this, it is not part of ECMAScript [i.e. why use nonstandard features?] Remember it's a style guide, which is about making code easy to read+maintain+debug: not just "it works" correct. – ShreevatsaR Jul 31 '16 at 20:29

1 amazing that after all these years string concatenation is still the best/safest/most compliant way to go with this. template literals (above answer) don't work in IE and escaping lines is just a mess that you're soon going to regret – Tiago Duarte Nov 11 '16 at 12:31

654

the pattern `text = <<"HERE" This Is A Multiline String HERE` is not available in js (I remember using it much in my good old Perl days).

To keep oversight with complex or long multiline strings I sometimes use an array pattern:

```
var myString =
   ['<div id="someId">',
    'some content<br />',
    '<a href="#someRef">someRefTxt</a>',
    '</div>'
   ].join('\n');
```

or the pattern anonymous already showed (escape newline), which can be an ugly block in your code:

```
    var myString =
       '<div id="someId"> \
some content<br /> \
<a href="#someRef">someRefTxt</a> \
</div>';
```

Here's another weird but working 'trick'[1]:

```
var myString = (function () {/*
   <div id="someId">
     some content<br />
     <a href="#someRef">someRefTxt</a>
    </div>
*/}).toString().match(/[^]*\/\*([^]*)\*\/\}$/)[1];
```

external edit: jsfiddle

ES20xx supports spanning strings over multiple lines using template strings:

```
let str = `This is a text
    with multiple lines.
    Escapes are interpreted,
    \n is a newline.`;
let str = String.raw`This is a text
    with multiple lines.
    Escapes are not interpreted,
    \n is not a newline.`;
```

[1] Note: this will be lost after minifying/obfuscating your code

edited Aug 24 '18 at 6:58

answered Apr 30 '09 at 7:22

KooiInc
81.3k 22 111 144

30 Please don't use the array pattern. It will be slower than plain-old string concatenation in most cases. – BMiner Jul 17 '11 at 12:39

144 Really? jsperf.com/join-concat/24 – KooiInc Jul 17 '11 at 13:21

68 The array pattern is more readable and the performance loss for an application is often negligible. As that perf test shows, even IE7 can do tens of thousands of operations per second. – Benjamin Atkin Aug 20 '11 at 8:16

18 +1 for an elegant alternative that not only works the same way in all browsers, but is also future-proof. – Pavel May 21 '12 at 6:06

24 @KooiInc Your tests start with the array already created, that skews the results. If you add the initialization of the array, straight concatenation is faster jsperf.com/string-concat-without-sringbuilder/7 See stackoverflow.com/questions/51185/… As a trick for newlines, it may be OK, but it's definitely doing more work than it should – Juan Mendes Aug 4 '13 at 8:02

336

You *can* have multiline strings in pure JavaScript.

This method is based on the serialization of functions, which is defined to be implementation-dependent. It does work in the most browsers (see below), but there's no guarantee that it will still work in the future, so do not rely on it.

Using the following function:

```
function hereDoc(f) {
  return f.toString().
      replace(/^[^\/]+\/\*!?/, '').
      replace(/\*\/[^\/]+$/, '');
}
```

You can have here-documents like this:

```
var tennysonQuote = hereDoc(function() {/*!
  Theirs not to make reply,
  Theirs not to reason why,
  Theirs but to do and die
*/});
```

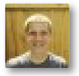
The method has successfully been tested in the following browsers (not mentioned = not tested):

- IE 4 - 10
- Opera 9.50 - 12 (not in 9-)
- Safari 4 - 6 (not in 3-)
- Chrome 1 - 45
- Firefox 17 - 21 (not in 16-)
- Rekonq 0.7.0 - 0.8.0
- *Not supported in Konqueror 4.7.4*

Be careful with your minifier, though. It tends to remove comments. For the YUI compressor, a comment starting with `/*!` (like the one I used) will be preserved.

I think a *real* solution would be to use CoffeeScript.

edited Oct 9 '15 at 19:00



Web_Designer
35.4k 74 182 240

answered Apr 6 '11 at 18:16

Jordão
46k 11 93 123

23 +1+1 Clever! (works in Node.JS) – Bryan Field Jun 4 '11 at 21:57

213 What!? creating and decompiling a Function to hack a multiline comment into being a multiline string? Now *that's* ugly. – fforw Jun 17 '11 at 15:49

7 It's actually *beyond* ugly... Although, there's no *decompiling* involved... – Jordão Jun 17 '11 at 18:39

3 jsfiddle.net/fqpwf works in Chrome 13 and IE8/9, but not FF6. I hate to say it, but I like it, and if it could be an intentional feature of each browser (so that it wouldn't disappear), I'd use it. – Jason Kleban Sep 9 '11 at 21:36

2 Extremely handy. I'm using it for (Jasmine) unit tests, but avoiding it for production code. – Jason Jul 13 '12 at 5:23

189

You can do this...

```
var string = 'This is\n' +
'a multiline\n' +
'string';
```

edited Jan 19 '14 at 4:14

community wiki
5 revs, 2 users 97%
alex

132

I came up with this very jimmy rigged method of a multi lined string. Since converting a function into a string also returns any comments inside the function you can use the comments as your string using a multilined comment /**/. You just have to trim off the ends and you have your string.

```
var myString = function(){/*
    This is some
    awesome multi-lined
    string using a comment
    inside a function
    returned as a string.
    Enjoy the jimmy rigged code.
*/}.toString().slice(14,-3)

alert(myString)
```

answered Mar 21 '13 at 21:05

Luke
1,369 1 8 3

34 This is absolutely terrifying. I love it (although you may need to do a regex match because I'm not sure how precise the whitespace for `toString()` is. – Kevin Cox Apr 7 '13 at 21:53

7 stackoverflow.com/a/5571069/499214 – John Dvorak Jun 5 '13 at 11:53

41 Also beware of minifiers that strip comments... :D – jondavidjohn Oct 22 '13 at 19:07

2 This is why we can't have nice things. – Danilo M. Oliveira Oct 15 '18 at 18:39

1 You can do some weird stuff in javascript land. Though in all honesty, you should never use this. – Luke Oct 25 '18 at 23:25

I'm surprised I didn't see this, because it works everywhere I've

86

tested it and is very useful for e.g. templates:

```
<script type="bogus" id="multi">
    My
    multiline
    string
</script>
<script>
    alert($('#multi').html());
</script>
```

Does anybody know of an environment where there is HTML but it doesn't work?

answered Jan 3 '12 at 19:51

Peter V. Mørch
4,984 2 32 51

23 Anywhere you don't want to put your strings into seperate and distant script elements. – Lodewijk Jan 9 '12 at 1:12

9 A valid objection! It isn't perfect. But for templates, that separation is not only ok, but perhaps even encouraged. – Peter V. Mørch Feb 3 '12 at 9:03

1 I prefer splitting everything over 80/120 characters into multiline, I'm afraid that's more than just templates. I now prefer 'line1 ' + 'line2' syntax. It's also the fastest (although this might rival it for really large texts). It's a nice trick though. – Lodewijk Feb 3 '12 at 22:51

27 actually, this is HTML not Javascript :-/ – CpILL May 22 '12 at 8:54

5 however, the task of obtaining a multiline string in javascript can be done this way – Davi Fiamenghi Jul 30 '13 at 21:41

I solved this by outputting a div, making it hidden, and calling the div id by jQuery when I needed it.

48

e.g.

```
<div id="UniqueID" style="display:none;">
     Strings
     On
     Multiple
     Lines
     Here
</div>
```

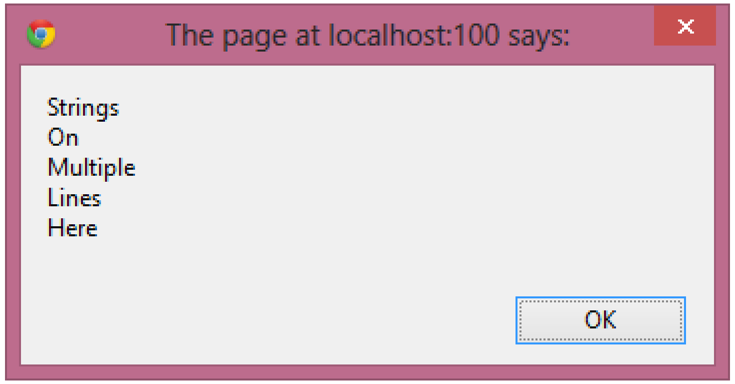
Then when I need to get the string, I just use the following jQuery:

```
$('#UniqueID').html();
```

Which returns my text on multiple lines. If I call

```
alert($('#UniqueID').html());
```

I get:



The page at localhost:100 says:

```
Strings
On
Multiple
Lines
Here
```

edited Dec 16 '16 at 14:20

XXN

60 8

answered Aug 17 '12 at 14:25

Tom Beech

1,312 13 20

2 Thanks for this! It's the only answer I've found that solves my problem, which involves unknown strings that may contain any combination of single and double quotes being directly inserted into the code with no opportunity for pre-encoding. (it's coming from a templating language that creates the JS -- still from a trusted source and not a form submission, so it's not TOTALLY demented). – octern Jun 23 '13 at 17:19

4 What if the string is HTML? – Dan Dascalescu Jan 24 '14 at 8:39

4 $('#UniqueID').content() – mplungjan Jan 24 '14 at 9:28

1 @Pacerier Everything I've read, from Google as well as other sites, says that nowadays Google does index `display:none` content, most likely due to the popularity of JavaScript-styled front-ends. (For example, an FAQ page with hide/show functionality.) You need to be careful though, because Google says they can punish you if the hidden content appears to be designed to artificially inflate your SEO rankings. – Gavin Aug 8 '17 at 13:12

27

There are multiple ways to achieve this

1. Slash concatenation

```
var MultiLine=  '1\
  2\
  3\
  4\
  5\
  6\
  7\
  8\
  9';
```

2. regular concatenation

```
var MultiLine = '1'
+'2'
+'3'
+'4'
+'5';
```

3. Array Join concatenation

```
var MultiLine = [
'1',
'2',
'3',
'4',
'5'
].join('');
```

Performance wise, **Slash concatenation** (first one) is the fastest.

Refer this test case for more details regarding the performance

Update:

With the **ES2015**, we can take advantage of its Template strings feature. With it, we just need to use back-ticks for creating multi line strings

Example:

```
 `<h1>{{title}}</h1>
  <h2>{{hero.name}} details!</h2>
  <div><label>id: </label>{{hero.id}}</div>
  <div><label>name: </label>{{hero.name}}</div>
  `
```

edited Nov 8 '16 at 8:33

answered May 26 '14 at 9:34

Vignesh Subramanian
3,796 6 41 107

10 I think it's that you've just regurgitated what has already on the page for five years, but in a cleaner way. – RandomInsano Aug 2 '14 at 18:22

27

Using script tags:

- add a `<script>...</script>` block containing your multiline text into `head` tag;
- get your multiline text as is... (watch out for text encoding: UTF-8, ASCII)

```
<script>

    // pure javascript
    var text = document.getElementById("mySoapMessage").innerHTM

    // using JQuery's document ready for safety
    $(document).ready(function() {

        var text = $("#mySoapMessage").html();

    });

</script>

<script id="mySoapMessage" type="text/plain">

    <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/
xmlns:typ="...">
       <soapenv:Header/>
       <soapenv:Body>
          <typ:getConvocadosElement>
             ...
          </typ:getConvocadosElement>
       </soapenv:Body>
    </soapenv:Envelope>
```

```
        <!-- this comment will be present on your string -->
        //uh-oh, javascript comments...  SOAP request will fail


    </script>
```

edited Jan 3 '17 at 13:58

answered Aug 23 '12 at 18:30

jpfreire
1,088 13 21

24

I like this syntax and indendation:

```
string = 'my long string...\n'
       + 'continue here\n'
       + 'and here.';
```

(but actually can't be considered as multiline string)

answered Dec 13 '11 at 20:09

semente
4,823 2 26 31

3 I use this, except I put the '+' at the end of the preceding line, to make it clear the statement is continued on the next line. Your way does line up the indents more evenly though. – Sean Oct 4 '12 at 8:54

7 putting the + at the beginning allows one to comment out that line without having to edit other lines when its the first/last line of the sequence. – moliad Dec 12 '13 at 15:38

3 I prefer the + at the front too as visually I do not need to scan to the end of the line to know the next one is a continuation. – Daniel Sokolowski May 7

'14 at 15:40

17

There's this library that makes it beautiful:

https://github.com/sindresorhus/multiline

Before

```
var str = '' +
'<!doctype html>' +
'<html>' +
'    <body>' +
'        <h1>❤ unicorns</h1>' +
'    </body>' +
'</html>' +
'';
```

After

```
var str = multiline(function(){/*
<!doctype html>
<html>
    <body>
        <h1>❤ unicorns</h1>
    </body>
</html>
*/});
```

answered Apr 25 '14 at 11:34

mightyiam
1,846 16 33

1 This support in `nodejs` , using in browser must becareful. – Huei Tan May 5 '14 at 8:52

3 @HueiTan Docs state it also works in the browser. Which makes sense -

it's just `Function.prototype.String()` . – mikemaccana Jul 13 '14 at 19:14

1 Best answer for me because it at least achieves multiline without all the rubbish in the middle(The rubbish at the beginning and ends I can deal with). – Damien Golding Aug 27 '14 at 6:25

13

Downvoters: This code is supplied for information only.

This has been tested in Fx 19 and Chrome 24 on Mac

DEMO

```
var new_comment; /*<<<EOF
    <li class="photobooth-comment">
       <span class="username">
          <a href="#">You</a>
       </span>
       <span class="comment-text">
          $text
       </span>
       <span class="comment-time">
          2d
       </span>
    </li>
EOF*/
// note the script tag here is hardcoded as the FIRST tag
new_comment=document.currentScript.innerHTML.split("EOF")[1];
alert(new_comment.replace('$text','Here goes some text'));
```

edited Jan 17 '16 at 15:44

answered Feb 17 '13 at 9:56

mplungjan
89.4k 22 127 184

12 That's horrific. +1. And you can use document.currentScript instead of getElement... – Orwellophile May 27 '15 at 10:00

1 Undefined "you" in chrome for osx – mplungjan May 27 '15 at 16:46

1 jsfiddle-fixed - I must have had "you" defined globally in my console. Works now (chrome/osx). The nice thing about adding the comment to a var is that you're not in a function context, jsfiddle-function-heredoc although the function thing would be cool for class methods. might be better to pass it a replace { this: that } object anyways. fun to push something crazy to the limit anyway :) – Orwellophile Jun 1 '15 at 16:44

1 Forget the haters. This is the only correct answer bar ES6. All the other answers require concatenation, computation of some sort, or escaping. This is actually pretty cool and I'm going to use it as a way to add documentation to a game I'm working on as a hobby. As long as this trick isn't used for anything that could invoke a bug (I can see how someone would go "Semicolon, derp. Lets put the comment on the next line." and then it breaks your code.) But, is that really a big deal in my hobby game? No, and I can use the cool trick for something useful. Great answer. – Thomas Dignan Jul 27 '15 at 21:10

2 I've never been brave enough to use this technique in production code, but where I DO use it a lot is in unit testing, where often it's easiest to dump the value of some structure as a (quite long) string and compare it to what it 'should' be. – Ben McIntyre Feb 3 '16 at 0:00

10

to sum up, I have tried 2 approaches listed here in user javascript programming (Opera 11.01):

- this one didn't work: Creating multiline strings in JavaScript
- this worked fairly well, I have also figured out how to make it look good in Notepad++ source view: Creating multiline strings in JavaScript

So I recommend the working approach for Opera user JS users. Unlike what the author was saying:

> It doesn't work on firefox or opera; only on IE, chrome and safari.

It DOES work in Opera 11. At least in user JS scripts. Too bad I can't comment on individual answers or upvote the answer, I'd do it immediately. If possible, someone with higher privileges please do it for me.

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

answered May 20 '11 at 13:10
Tyler
332 3 11

9

Updated for 2015: it's six years later now: most people use a module loader, and the main module systems each have ways of loading templates. It's not inline, but the most common type of multiline string are templates, and **templates should generally be kept out of JS anyway**.

require.js: 'require text'.

Using require.js 'text' plugin, with a multiline template in **template.html**

```
var template = require('text!template.html')
```

NPM/browserify: the 'brfs' module

Browserify uses a 'brfs' module to load text files. This will actually build your template into your bundled HTML.

```
var fs = require("fs");
var template = fs.readFileSync(template.html', 'utf8');
```

Easy.

edited Dec 31 '14 at 17:48

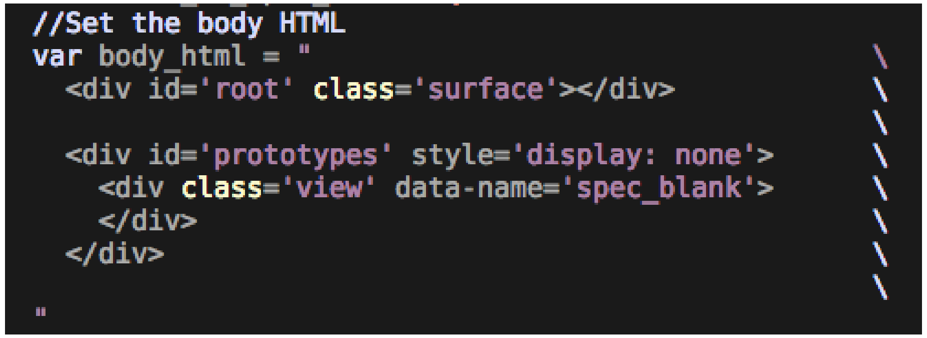
answered Feb 10 '14 at 11:13

mikemaccana

43.5k 46 235 300

9

If you're willing to use the escaped newlines, they can be used *nicely*. **It looks like a document with a page border**.



```
//Set the body HTML
var body_html = "                                                     \
  <div id='root' class='surface'></div>                               \
                                                                      \
  <div id='prototypes' style='display: none'>                         \
    <div class='view' data-name='spec_blank'>                         \
    </div>                                                            \
  </div>                                                              \
                                                                      \
"
```

edited Sep 30 '15 at 14:14

answered Apr 28 '15 at 18:31

seo

1,598 18 17

1 Wouldn't this add extraneous blank spaces? – tomByrer Dec 6 '15 at 12:29

8

This works in IE, Safari, Chrome and Firefox:

```
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.j
<div class="crazy_idea" thorn_in_my_side='<table  border="0">
                        <tr>
                            <td ><span class="mlayouttablecellsdynam
$65.00</span></td>
                        </tr>
                    </table>'></div>
<script type="text/javascript">
    alert($(".crazy_idea").attr("thorn_in_my_side"));
</script>
```

answered Nov 23 '10 at 16:46

stillatmycomputer
145 1 2

6 Just think about it. Do you think it's valid? Don't you think it can cause display problems? – Sk8erPeter Feb 24 '12 at 1:55

5 Why the downvotes? This is a creative answer, if not very practical! – dotancohen Feb 29 '12 at 2:32

2 no, it's not. One should rather use templates: $.tmpl() (api.jquery.com/tmpl), or EJS (embeddedjs.com/getting_started.html), etc. One reason for downvotes is that it's really far from a valid code and using this can cause huge display problems. – Sk8erPeter Mar 24 '12 at 0:07

My extension to https://stackoverflow.com/a/15558082/80404. It expects comment in a form `/*! any multiline comment */` where

8

symbol ! is used to prevent removing by minification (at least for YUI compressor)

```
Function.prototype.extractComment = function() {
    var startComment = "/*!";
    var endComment = "*/";
    var str = this.toString();

    var start = str.indexOf(startComment);
    var end = str.lastIndexOf(endComment);

    return str.slice(start + startComment.length, -(str.length - end
};
```

Example:

```
var tmpl = function() { /*!
 <div class="navbar-collapse collapse">
    <ul class="nav navbar-nav">
    </ul>
 </div>
*/}.extractComment();
```

edited May 23 '17 at 10:31
Community ♦
1 1

answered Dec 11 '13 at 17:30
pocheptsov
1,445 16 19

The equivalent in javascript is:

8

```
var text = `
This
Is
```

```
A
Multiline
String
`;
```

Here's the specification. See browser support at the bottom of this page. Here are some examples too.

edited Nov 4 '15 at 14:06

answered Nov 4 '15 at 13:59

Lonnie Best
2,791 5 31 45

7

You can use TypeScript (JavaScript SuperSet), it supports multiline strings, and transpiles back down to pure JavaScript without overhead:

```
var templates = {
    myString: `this is
a multiline
string`
}

alert(templates.myString);
```

If you'd want to accomplish the same with plain JavaScript:

```
var templates =
{
 myString: function(){/*
    This is some
    awesome multi-lined
    string using a comment
    inside a function
    returned as a string.
    Enjoy the jimmy rigged code.
*/}.toString().slice(14,-3)

}
alert(templates.myString)
```

Note that the iPad/Safari does not support `'functionName.toString()'`

If you have a lot of legacy code, you can also use the plain JavaScript variant in TypeScript (for cleanup purposes):

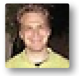
```
interface externTemplates
{
    myString:string;
}

declare var templates:externTemplates;

alert(templates.myString)
```

and you can use the multiline-string object from the plain JavaScript variant, where you put the templates into another file (which you can merge in the bundle).

You can try TypeScript at
http://www.typescriptlang.org/Playground

edited Oct 7 '15 at 13:12

answered Sep 21 '15 at 15:23


Stefan Steiger
45.7k 56 271 360

ES6 allows you to use a backtick to specify a string on multiple lines.

5

It's called a Template Literal. Like this:

```
var multilineString = `One line of text
    second line of text
    third line of text
    fourth line of text`;
```

Using the backtick works in NodeJS, and it's supported by Chrome, Firefox, Edge, Safari, and Opera.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals

answered Mar 28 '16 at 18:43

earl3s
1,940 1 18 21

4

Easiest way to make multiline strings in Javascrips is with the use of backticks (``). This allows you to create multiline strings in which you can insert variables with `${variableName}` .

Example:

```
let name = 'Willem';
let age = 26;

let multilineString = `
my name is: ${name}

my age is: ${age}
`;

console.log(multilineString);
```

Run code snippet Expand snippet

compatibility :

- It was introduces in `ES6` // `es2015`
- It is now natively supported by all major browser vendors (except internet explorer)

Check exact compatibility in Mozilla docs here

edited Oct 1 '18 at 21:27

answered Sep 9 '18 at 11:00

Willem van der Veen
5,230 3 25 36

3

My version of array-based join for string concat:

```
var c = []; //c stands for content
c.push("<div id='thisDiv' style='left:10px'></div>");
c.push("<div onclick='showDo(\'something\');'></div>");
$(body).append(c.join('\n'));
```

This has worked well for me, especially as I often insert values into the html constructed this way. But it has lots of limitations. Indentation would be nice. Not having to deal with nested quotation marks would be really nice, and just the bulkyness of it bothers me.

Is the .push() to add to the array taking up a lot of time? See this related answer:

(Is there a reason JavaScript developers don't use Array.push()?)

After looking at these (opposing) test runs, it looks like .push() is fine for string arrays which will not likely grow over 100 items - I will avoid it in favor of indexed adds for larger arrays.

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

answered Oct 14 '13 at 0:58
KTys
155 9

3

You can use `+=` to concatenate your string, seems like no one answered that, which will be readable, and also neat... something like this

```
var hello = 'hello' +
            'world' +
            'blah';
```

can be also written as

```
var hello = 'hello';
    hello += ' world';
    hello += ' blah';

console.log(hello);
```

answered Jan 18 '14 at 13:18
Mr. Alien
119k 26 226 232

3

Also do note that, when extending string over multiple lines using forward backslash at end of each line, any extra characters (mostly spaces, tabs and comments added by mistake) after forward backslash will cause unexpected character error, which i took an hour to find out

```
var string = "line1\  // comment, space or tabs here raise error
line2";
```

answered Jul 13 '16 at 19:25
Prakash GPz
1,000 2 10 22

3

Please for the love of the internet use string concatenation and opt not to use ES6 solutions for this. ES6 is NOT supported all across the board, much like CSS3 and certain browsers being slow to adapt to the CSS3 movement. Use plain ol' JavaScript, your end users will thank you.

Example:

```
var str = "This world is neither flat nor round. "+
          "Once was lost will be found";
```

answered Oct 11 '17 at 23:28
Pragmatiq
47 4

1 while i agree with your point, i wouldn't call javascript "good" ol – user151496 Mar 5 '18 at 0:18

2

You have to use the concatenation operator '+'.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <p id="demo"></p>
    <script>
        var str = "This "
                + "\n<br>is "
                + "\n<br>multiline "
                + "\n<br>string.";
        document.getElementById("demo").innerHTML = str;
     </script>
</body>
</html>
```

By using `\n` your source code will look like -

```
This
 <br>is
 <br>multiline
 <br>string.
```

By using `<br>` your browser output will look like -

```
This
is
multiline
string.
```

answered Aug 13 '17 at 22:57

Sonevol
221 2 12

1

I think this workaround should work in IE, Chrome, Firefox, Safari, Opera -

Using jQuery :

```
<xmp id="unique_id" style="display:none;">
  Some plain text
  Both type of quotes :  " ' " And  ' " '
  JS Code : alert("Hello World");
  HTML Code : <div class="some_class"></div>
</xmp>
<script>
   alert($('#unique_id').html());
</script>
```

Using Pure Javascript :

```
<xmp id="unique_id" style="display:none;">
  Some plain text
  Both type of quotes :  " ' " And  ' " '
  JS Code : alert("Hello World");
  HTML Code : <div class="some_class"></div>
</xmp>
<script>
   alert(document.getElementById('unique_id').innerHTML);
</script>
```

Cheers!!

answered Jan 28 '13 at 12:20

Aditya Hajare
184 1 2 17

Just tried the Anonymous answer and found there's a little trick here, it doesn't work if there's a space after backslash `\`

0

So the following solution doesn't work -

```
var x = { test:'<?xml version="1.0"?>\ <-- One space here
            <?mso-application progid="Excel.Sheet"?>'
};
```

But when space is removed it works -

```
var x = { test:'<?xml version="1.0"?>\<-- No space here now
          <?mso-application progid="Excel.Sheet"?>'
};

alert(x.test);
```

Hope it helps !!

answered Nov 23 '12 at 13:10

Anmol Saraf
9,590 7 41 52

7 well obviously if you have a space after a backslash, backslash escapes the space. It is supposed to escape linebreak, not space. – Sejanus Dec 14 '12 at 8:47

0

If you happen to be running in Node only, you could use the fs module to read in the multi-line string from a file:

```
var diagram;
var fs = require('fs');
fs.readFile( __dirname + '/diagram.txt', function (err, data) {
  if (err) {
    throw err;
  }
  diagram = data.toString();
});
```

answered Sep 9 '14 at 0:02

Charles Brandt
393 3 10

1 2 next

protected by Travis J Dec 12 '13 at 3:03

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?