Efficient Javascript String Replacement

Asked 16 years ago Modified 1 year, 10 months ago Viewed 65k times

40

Hey there, I've got a block of HTML that I'm going to be using repeatedly (at various times during a users visit, not at once). I think that the best way to accomplish this is to create an HTML div, hide it, and when needed take its innerHTML and do a replace() on several keywords. As an example HTML block...

```
<div id='sample'>
  <h4>%TITLE%</h4>
  <p>Text text %KEYWORD% text</p>
  <p>%CONTENT%</p>
  <img src="images/%ID%/1.jpg" />
</div>
```

Would the best way to replace those keywords with dynamic data be to go...

```
template = document.getElementById('sample');
template = template.replace(/%TITLE%/, some_var_with_title);
template = template.replace(/%KEYWORD%/, some_var_with_keyword);
template = template.replace(/%CONTENT%/, some_var_with_content);
template = template.replace(/%ID%/, some_var_with_id);
```

It just feels like I've chosen a stupid way to do this. Does anyone have any suggestions on how to do this faster, smarter, or better in any way? This code will be executed fairly often during a users visit, sometimes as often as once every 3-4 seconds.

Thanks in advance.

javascript string performance replace

Share
Improve this question
Follow

edited Dec 18, 2008 at 14:24

asked Dec 18, 2008 at 14:16
Josh

12 Answers

Sorted by: Highest score (default)

It looks like you want to use a template.

95

```
//Updated 28 October 2011: Now allows 0, NaN, false, null and undefined in
output.
function template( templateid, data ){
    return document.getElementById( templateid ).innerHTML
      .replace(
        /%(\w*)%/g, // or /{(\w*)}/g for "{this} instead of %this%"
        function( m, key ){
          return data.hasOwnProperty( key ) ? data[ key ] : "";
        }
      );
}
```

Explanation of the code:

- Expects `templateid` to be the id of an existing element.
- Expects `data` to be an object with the data.
- Uses two parameters to replace to do the substitution:
- The first is a regexp that searches for all `%keys%` (or `{keys}` if you use the alternate version). The key can be a combination of A-Z, a-z, 0-9 and underscore _.
- The second is a anonymous function that gets called for every match.
- The anonymous function searches the data object for the key that the regexp found. If the key is found in the data, then the value of the key is returned and that value will be replacing the key in the final output. If the key isn't found, an empty string is returned.

Example of template:

```
<div id="mytemplate">
  <p>%test%</p>
  <p>%word%</p>
</div>
```

Example of call:

```
document.getElementById("my").innerHTML=template("mytemplate",
{test:"MYTEST",word:"MYWORD"});
```

Share

Improve this answer

Follow

edited Jan 24, 2017 at 11:50

answered Dec 18, 2008 at 14:30

some
49.4k ● 14 ● 81 ● 95

1 Thanks, this rocks. I was just about ready to include a plugin like "jQuery printf" in my app, but this is all I really need :-) – rescdsk Oct 19, 2011 at 13:26

2 Except! That it is incapable of inserting the number zero! The replace function should really check for the value being null / undefined, rather than for truth. – rescdsk Oct 19, 2011 at 13:49

rescdsk: You are right, it was incapable of inserting any falsy values like 0, NaN, false, null and undefined. I have updated the code to use `hasOwnProptery` on the object. If the property exists it will be included (even undefined). If the property doesn't exists then it will be empty space. You can change it to whatever you want by inserting text between the last `""`. – some Oct 28, 2011 at 13:40

This is even a great oneliner: `document.getElementById('templateid').innerHTML.replace(/%(\w*)%/g, (m, key) => data.hasOwnProperty(key) ? data[key] : "")` – Fabian von Ellerts May 28, 2019 at 10:52

1 Thanks for this wee script. Saved me some headache. – Merovex Dec 30, 2021 at 11:58

36

You could probably adapt this code to do what you want:

```
let user = {
    "firstName": "John",
    "login": "john_doe",
    "password": "test",
};

let template = `Hey {firstName},

    You recently requested your password.
    login: {login}
    password: {password}

    If you did not request your password, please disregard this message.
    `;

template = template.replace(/{([^{}]+)}/g, function(keyExpr, key) {
    return user[key] || "";
});
```

You might also want to look into JavaScriptTemplates

Share

Improve this answer

Follow

edited Apr 16, 2021 at 16:29
Sergey Ponomarev
3,161 ● 1 ● 37 ● 48

answered Dec 18, 2008 at 14:30
Kristof Neirynck
4,132 ● 2 ● 36 ● 50

5 To avoid the additional replace call inside the handler function, just group the regex match: textbody.replace(/{([^{}]+)}/g, function(textMatched, key) { .... – Diego May 3, 2013 at 20:37

wow! regex king! – minigeek Feb 11, 2022 at 6:46

To resolve deeper objects, e.g. {user.details.name}, see: stackoverflow.com/a/22129960 – fooquency Jan 11, 2023 at 23:52

Template Replacement

25

A quick and easy solution will be to use the String.prototype.replace method.
It takes a second parameter that can be either a value or a function:

```
function replaceMe(template, data) {
  const pattern = /{\s*(\w+?)\s*}/g; // {property}
  return template.replace(pattern, (_, token) => data[token] || '');
}
```

###*Example*:

```
const html = `
  <div>
    <h4>{title}</h4>
    <p>My name is {name}</p>
    <img src="{url}" />
  </div>
`;

const data = {
  title: 'My Profile',
  name: 'John Smith',
  url: 'http://images/john.jpeg'
};
```

And call it like so:

```
replaceMe(html, data);
```

Share edited Apr 15, 2021 at 1:18 answered May 26, 2018 at 18:09

Improve this answer

Follow

Lior Elrom
20.8k ● 16 ● 82 ● 94

1 This is the most correct and efficient way to solve this problem. Two notes: [1] change the regex to `/\{\s*(\w+?)\\s*}/g`, as you would probably like to accept only variable-like keys and ignore any spaces in the brackets. [2] You must add a fallback to `data[token]` into empty string (`data[token]||''`), as there may be a case where the data object doesn't

include a found key, in this case JS will output the string `undefined`. I will make the changes to your answer accordingly. – Slavik Meltser Jul 9, 2019 at 10:51

@SlavikMeltser Is this really the most correct and efficient way to solve this problem? Have you looked at stackoverflow.com/a/378001/36866 that was written here in this thread more than 10 years ago, that use the same principle but don't have the bug with the fallback? If data[token] is the number zero, it will be an empty string with your suggestion. – some Aug 22, 2019 at 14:38

@SlavikMeltser, never said it's the "most correct and efficient way" but only offered a "quick and easy solution" for this challenge. Our solutions are indeed very similar (didn't notice it initially) however, I offered a robust option that can be used in different scenarios. Hope it makes sense. – Lior Elrom Aug 22, 2019 at 14:50

1 @some Of cores, assuming that the data is provided in strings only same way as the assumption of `data` is an object. In most of the cases this will do. This is because, the main purpose of this solution is to use it within templates mechanisms. Which means that `'0'` as a string is still positive. But, you are right, if you want to make it even more robust, then there are a lot more features to add beyond just `hasOwnProperty`, like checking that `template` is even a string, or `data` is an object, etc. That's the beauty of it, you are always have more space to improve. – Slavik Meltser Aug 24, 2019 at 9:07

@SlavikMeltser you are correct. This is just a simple string replacement and never intended to be a full-featured template engine like Mustache, Handlebars or EJS. – Lior Elrom Nov 18, 2020 at 15:39

14

I doubt there will be anything more efficient. The alternative would be splitting it into parts and then concatenating, but I don't think that would be much efficient. Perhaps even less, considering that every concatenation results in a new string which has the same size as its operands.

Added: This is probably the most elegant way to write this. Besides - what are you worried about? Memory usage? It's abundant and Javascript has a decent memory manager. Execution speed? Then you must have some gigantic string. IMHO this is good.

Share Improve this answer Follow

answered Dec 18, 2008 at 14:21

Vilx-
107k ● 90 ● 288 ● 430

Thanks for the reply. In actuality this is a much bigger block with many more replaces, so before I started I wanted to make sure there wasn't something I was missing. Thanks again. – Josh Dec 18, 2008 at 14:28

1 And there are better ways to implement it. – some Dec 18, 2008 at 14:47

2

Your method is a standard way to implement a poor-man's templating system, so it's fine.

It could be worth your while to check out some JavaScript templating libraries, such as JST.

Share Improve this answer Follow

answered Dec 18, 2008 at 19:39

orip
75.3k ● 21 ● 119 ● 149

1

You can make it more efficient by chaining the replaces instead of making all these interim assignments.

i.e.

```
with(document.getElementById('sample'))
{
  innerHTML = innerHTML.replace(a, A).replace(b, B).replace(c, C); //etc
}
```

Share Improve this answer Follow

answered Dec 18, 2008 at 14:22

annakata
75.7k ● 18 ● 115 ● 180

Maybe, but doesn't this make readability worse? Although you might stack these calls vertically... – Vilx- Dec 18, 2008 at 14:23

putting this in a with block also will break if you're replacing a keyword with a variable name that is also an object property, like "id", for instance. – Kenan Banks Dec 18, 2008 at 19:50

sigh - look performance is *not* the same because chaining you create the object but do not assign it. For a chain N long you save N-1 assignments. Putting this in a with block certainly breaks if you have properties declared in scope of with, but I'm assuming as per the OP he's *not doing that* – annakata Dec 18, 2008 at 20:00

@annakata, my benchmarks show no difference, do yours show one?. Since in JS assignment is just creating a reference, why should its time be non-neglibel? – orip Dec 21, 2008 at 6:59

1 Finally someone used the with() in js , I hear its not good to use with () because " Use of the with statement is not recommended, as it may be the source of confusing bugs and compatibility issues. See the "Ambiguity Contra" paragraph in the "Description" section below for details. at " developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/…
– user3548161 Aug 26, 2018 at 9:19

1

If you're willing to use the Prototype library, they have nice built in templating functionality.

That would look like:

```
element.innerHTML = (new Template(element.innerHTML)).evaluate({
    title: 'a title',
    keyword: 'some keyword',
    content: 'A bunch of content',
    id: 'id here'
})
```

This would be especially nice if you were running your code in a loop due to the ease of creating JSON objects/Javascript object literals.

Still, I wouldn't expect any speed increase.

Also, you would need to change your delimiter style to `#{keyword}` rather than `%keyword%`

Share Improve this answer Follow

answered Dec 18, 2008 at 18:38

Kenan Banks

212k ● 36 ● 159 ● 175

1

Mustachejs is great for really elegant templating:

```
<div id='sample'>
  <h4>{{TITLE}}</h4>
  <p>Text text {{KEYWORD}} text</p>
  <p>{{CONTENT}}</p>
  <img src="images/{{ID}}/1.jpg" />
</div>
```

You can then use the template something like this:

```
var template = document.getElementById(templateid).innerHTML;
var newHtml = Mustache.render(template, {
    TITLE: some_var_with_title,
    KEYWORD: some_var_with_keyword,
    CONTENT: some_var_with_content,
    ID: some_var_with_id
});
document.getElementById('sample').innerHTML = newHtml;
```

This especially works nicely if you are getting JSON back from an Ajax call - you can just pass it straight in to the `Mustache.render()` call.

Slight variations allow for running the same template on each the browser or the server. See https://github.com/janl/mustache.js for more details.

Share Improve this answer Follow

answered Feb 13, 2014 at 8:54

Philip Callender
1,485 ● 1 ● 14 ● 21

1

This approach generates function templates that can be cached:

```
function compileMessage (message) {

  return new Function('obj', 'with(obj){ return \'' +
    message.replace(/\n/g, '\\n').split(/{{([^{}]+)}}/g).map(function
(expression, i) {
      return i%2 ? ( '\'+(' + expression.trim() + ')+\'' ) : expression;
    }).join('') +
  '\'; }');

}

var renderMessage = compileMessage('Hi {{ recipient.first_name }},\n\n' +

'Lorem ipsum dolor sit amet...\n\n' +

'Best Regarts,\n\n' +

'{{ sender.first_name }}');


renderMessage({
  recipient: {
    first_name: 'John'
  },
  sender: {
    first_name: 'William'
  }
});
```

returns:

```
"Hi John,

Lorem ipsum dolor sit amet...

Best Regarts,

William"
```

Share
Improve this answer

edited Jan 20, 2017 at 0:16

answered Jan 20, 2017 at 0:08

Jesús Germade
9 ● 2

Follow

0

Try this: http://json2html.com/

It supports complex JSON objects as well.

Share Improve this answer Follow

answered Aug 4, 2016 at 11:19

sura2k
7,497 ● 13 ● 64 ● 82

0

In 2023 all modern browsers have JavaScript ES6 (ECMAScript 2015) with built-in **template literals** with replacement functionality. There is no longer a need to use external libraries or regex unless you need to support IE: https://caniuse.com/?search=template%20literals

Basic usage:

- Template literals start and end with backticks `
- Replacement values enclosed like so `${value}` where `value` is a JS expression.
- Added benefits: no need to escape single or double quote characters and newlines are preserved.

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

```
// Generate HTML markup for XYZ page
// @param {object} obj - parameter object with replacement values
function getMyHtml(obj) {
  if (!obj) return "";
  return `<div id='sample'>
  <h4>${obj.title}</h4>
  <p>Text text ${obj.keyword} text</p>
  <p>${obj.content}</p>
  <img
src="https://live.staticflickr.com/${obj.id}/52313050555_c70c17a288_m.jpg" />
</div>`
}

// example usage:
document.getElementById('container').innerHTML = getMyHtml({
    title: 'Cellular Base Stations in Orbit',
    keyword: 'SpaceX',
    content: 'Tonight’s unveil: Connecting directly...',
    id: 65535
  });
```

```
<div id='container' style='border: 1px solid blue;'>

</div>
```

Run code snippet   Expand snippet

(example links to a flickr photo by Steve Jurvetson)

Share Improve this answer Follow

answered Feb 7, 2023 at 15:28

Martin Lisowski
666 ● 1 ● 5 ● 15

-2

```
var template = "<div id='sample'><h4>%VAR%</h4><p>Text text %VAR% text</p>
<p>%VAR%</p><img src="images/%VAR%/1.jpg" /></div>";

var replace = function(temp,replace){
temp = temp.split('%VAR%');
for(var i in replace){
          if(typeof temp[i] != 'undefined'){
            temp[i] = temp[i] + replace[i];
          }
        }
   return temp.join('');
}

replace(template,['title','keyword','content','id'])
```

Share Improve this answer Follow

answered Oct 15, 2015 at 3:14

garyfunghc
1

3 Please add some explanation to your answer – Alex Oct 15, 2015 at 3:19

1 Your quotes are broken. – PM 2Ring Feb 15, 2016 at 11:25