Merge Arrays with JavaScript

OSCON, Austin, TX • May 8-11 • Save 20% PC20DWALSH

By David Walsh on July 8, 2015 36

Merging arrays is a fairly common occurrence. I remember when I worked a lot with PHP I would use `array_merge()` all the time. I found myself merging arrays often when handling form submission.

{Track:js}

30 Day Free Trial

JavaScript has a simple, native function for merging arrays (`concat`) but it produces a new array. Since JavaScript vars are passed by reference, `concat` may mess up a reference. If you want to merge a second array into an existing first array, you can use this trick:

```
var array1 = [1, 2, 3];
var array2 = [4, 5, 6];
Array.prototype.push.apply(array1, array2);

console.log(array1); // is: [1, 2, 3, 4, 5, 6]
```

Using an `Array.prototype` method, `push` in this case, allows you to merge the second array into the first. The alternative is iterating through the second array and using `push` on the first array. This shortcut is niiiiiiiiice!

IMAGECON 2017

March 1st, 2017 | Four Seasons San Francisco

Join us as we discuss the image challenges facing developers today.

LEARN MORE

Recent Features

CSS @supports

Feature detection via JavaScript is a client side best practice and for all the right reasons, but unfortunately that same functionality hasn't been available within CSS. What we end up doing is repeating the same properties multiple times with each browser prefix. Yuck. Another thing we...

Creating Scrolling Parallax Effects with CSS

Introduction For quite a long time now websites with the so called "parallax" effect have been really popular. In case you have not heard of this effect, it basically includes different layers of images that are moving in different directions or with different speed. This leads to a...

Incredible Demos

Image Reflections with CSS

Image reflection is a great way to subtly spice up an image. The first method of creating these reflections was baking them right into the images themselves. Within the past few years, we've introduced JavaScript strategies and CANVAS alternatives to achieve image reflections without...

MooTools Window Object Dumping

Ever want to see all of the information stored within the window property of your browser? Here's your chance. The XHTML We need a wrapper DIV that we'll consider a console. The CSS I like making this look like a command-line console. The MooTools JavaScript Depending on what you have loaded...

Discussion

Ilya

Why not https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/Array/concat ?

David Walsh

`concat` returns a new array, doesn't merge into the first one.

Ilya

Yes, it returns new array, but you didn't say that in post

Valtteri

With ES6 spread operator:

```
var array1 = [0, 1, 2];
var array2 = [3, 4, 5];
array1.push(...array2);
```

Source: [MDN]

Nikhil

Why is

```
Array.prototype.push.apply(array1, array2);
```

different from

```
Array.push.apply(array1, array2);
```

, which returns

```
[1,2,3]
```

Ilya

there is no `Array.push`

Florent

It's only in Firefox and not standard (proposed here but no news since: http://wiki.ecmascript.org/doku.php?id=strawman:array_statics).

Anyway, you can use `

Florent

… you can use

```
[].push.apply(array1, array2)
```

Vivek Kumar Bansal

There's a `concat` method for this purpose.

David Walsh

Again, `concat` creates a new array. My method in the post does not.

Ryan Murphy

I wish you posted this 2 days ago :) I went back and refactored.
Thanks!

Logo Ping

Same here. I have refactored it as well. I used to use this tool:
https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/Array/concat

But it created new arrays instead of merging them. Thanks to this article, I won't have to use this tool again.

Nice work David.

Preston

Yechezkel

Why not

```
array2.push.apply(array1, array2);
```

?

Val

I guess that's just a matter of taste. I personally wouldn't like to to reference the same variable twice in the same line.

Ilya

how do you prepend a string to variable? :)
I mean how do you write the following:

```
str = 'foo' + str;
```

Robert

This can be shortened just a little:

```
array1.push.apply(array1, array2);
```

Although as mentioned above the ES6 spread operator is great for concatenation.

chris-l

If you miss for some reason `array_merge`, then you can do:

```
var array_merge = Function.prototype.apply.bind(Array.prototype.push);
```

Then you can use like this:

```
array_merge(array1, array2);
```

(IMHO, using concat is better. Returning a new array instead of changing one of the arguments is more "functional". But of course, I'm pretty sure there are cases where changing an existing array may be required)

Berkana

You could save yourself a bunch of typing by doing

```
[].push.apply(array1, array2);
```

since the empty array will look up the prototype chain to find the `push` method, then `apply` it.

IlyaVF

Although the result is the same, but in your variate there is are two extra steps of creating a new array and looking through the prototype chain.

Jay Doubleyou

Thanks for sharing!

Nick

A new array via `concat` is preferable to a mutative merge!

Ray

I tend to discourage this type of thing as opposed to properly using `concat`. As it can be difficult, at first glance for other devs to figure out what the intent is.

Guy

I think that a lot of people are missing the point. `concat` and `[].push.apply(array1, array2)` would be used in different situations.

`concat` is used if you *want* a new array – immutability is an example.

`[].push.apply(array1, array2)` – is used if you need to maintain a reference to the primary array. e.g. it's a property of an object that you're iterating through and is referenced elsewhere.

Romain

Thanks for sharing this trick!

FGRibreau

Beware of apply. A too large array2 will result in a Uncaught RangeError: Maximum call stack size exceeded.

E.g. this will crash in google chrome (with lodash):

```
var arr = [1,2];
arr.push.apply(arr, _.range(0, 500000));
```

IlyaVF

This is a nice trick. Just wanted to add some explanation why this is working.

The `push` method expects comma separated arguments to be used as a new elements for the array to push to.

The `apply` method expects two arguments: the first one to be used as a context (`this`) for the method, and the second one as an array of arguments to be applied to the method.

So the trick is that when you use _apply_ the _array1_ is used a context for `push` method and `array2` in the original example will be used as an array of arguments for the push method.

And it does not matter if you use `[].push` or `array1.push` or `array2.push` or `Array.prototype.push` . The `apply` method will replace the context (`this`) for the method according to its first argument. Only `Array.prototype.push` is the fastest (there is no prototype chain lookup).

Ctibor

Thats really nice shortcut.. I love it :)

Ajay

```
array1 = array1.concat(array2);
```

This serves the same output. So what is the difference between using this method and `Array.prototype.push.apply(array1, array2)` ?

Aaron

“Since JavaScript vars are passed by reference, concat may mess up a reference.”

What does this mean? Mutating a reference IS messing up the reference. This is a performance optimization that could lead to subtle bugs. In other words, you should probably stick to concat unless you have a good reason for mutation.

IlyaVF

I think by this statement the author meant something like this:

```
var a = [1,2,3];
var b = a;
a = a.concat([4,5]);
a
// -> [1,2,3,4,5];
```

```
b
// -> [1,2,3]
```

Whereas using push method _b_ would point to the same array as _a_.

Adam

This was covered previously on DWB: http://davidwalsh.name/combining-js-arrays

noproblemo

concat useful to flattening array , example :

```
var arr=[[0],[1],[2],[3],[4],[5]];
  var rslt=[].concat.apply([],arr);
  console.log(rslt)//[0,1,2,3,4,5]
```

is there a better way ???

Dissertationmall.co.uk

`push` and `pop` add/remove elements from the end of the array `unshift` / `shift` — add/remove elements from the beginning of the array `splice` — add/remove elements from the specified location of the array.

joe hoeller

How do you make an array of arrays?

```
$scope.result = [ ];

//do stuff here to push the arrays into the above array so the result look like this:

$scope.result = [  [1,2], [3,4], [5,6]  ];
```

Brian Peacock

Why not return `Array.concat()` out of a function?

```
function mergeArrays() {
      return [].concat.apply([], arguments);
    }
    var arrA = [1,2,3,4,5];
    var arrB = [6,7,8,9,10];
    var arrC = ['eleven','twelve','thirteen'];

    arrA = mergeArrays(arrA, arrB, arrC);
    //-> arrA: [1,2,3,4,5,6,7,8,9,10,'eleven','twelve','thirteen']
```

Just a thought.

Mark

Brian Peacock, Thanks alot for your code!!! It worked like charm!!!

Name	Email	Website

Wrap your code in `<pre class="{language}"></pre>` *tags, link to a GitHub gist, JSFiddle fiddle, or CodePen pen to embed!*

☐ **Continue this conversation via email**

Post Comment! | Use Code Editor