How can I create a two dimensional array in JavaScript?

1034

233

I have been reading online and some places say it isn't possible, some say it is and then give an example and others refute the example, etc.

1. How do I declare a 2 dimensional array in JavaScript? (assuming it's possible)
2. How would I access its members? (`myArray[0][1]` or `myArray[0,1]` ?)

javascript arrays multidimensional-array

edited Apr 20 '15 at 2:52

royhowie
9,307 13 38 61

asked Jun 8 '09 at 18:24

Diego
6,700 7 28 45

16 Assuming a somewhat pedantic definition, it is technically impossible to create a 2d array in javascript. But you can create an array of arrays, which is tantamount to the same. – I. J. Kennedy Jul 29 '14 at 5:05

Duplicate of - stackoverflow.com/q/6495187/104380 – vsync Mar 26 '16 at 16:55

6 FYI... when you fill an array with more arrays using `var arr2D = new Array(5).fill(new Array(3));` , each element of Array(5) will point to the same Array(3). So it's best to use a for loop to dynamically populate sub arrays. – Josh Stribling May 23 '16 at 8:51

34 `a = Array(5).fill(0).map(x => Array(10).fill(0))` – Longfei Wu Mar 25 '17 at 14:21

1 In other words, `fill` doesn't call `new Array(3)` for each index of the array being filled, since it's not a lambda expression or anything, such as Longfei Wu's comment above, which initially fills the array with 0's, then uses the map function with a lambda to fill each element with a new array. The fill function simply fills the array with exactly what you tell it to. Does that make sense? For more info on the `map` function, see: developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/… – Josh Stribling Sep 16 '17 at 5:38

42 Answers

1167

```
var items = [
  [1, 2],
  [3, 4],
  [5, 6]
];
console.log(items[0][0]); // 1
console.log(items);
```

Run code snippet Expand snippet

edited Sep 10 '16 at 15:20
Ruslan López
3,129 1 16 28

answered Jun 8 '09 at 18:27
Ballsacian1
14.5k 2 20 25

27 It would be difficult to initialize a large multidimensional array this way. However, this function can be used to create an empty multidimensional, with the dimensions specified as parameters. – Anderson Green Apr 6 '13 at 16:49

2 @AndersonGreen It's a good thing you mentioned a link for those interested in multi-D array solution, but the question and Ballsacian1's answer are about "2D" array, not "multi-D" array – evilReiko Jun 14 '14 at 9:56

You should go through the whole thing... e.g. alert(items[0][1]); // 2 etc. – Dois May 28 '15 at 8:11

1 @SashikaXP, this does not work for first indices other than 0. – Michael Franzl Dec 30 '15 at 17:55

The question is how to declare a two dimensional array. Which is what I was looking for and found this and following answers which fail to discern the difference between declare and initialize. There's also declaration with known length or unbounded, neither of which is discussed. – chris May 30 '16 at 1:20

You simply make each item within the array an array.

404

```
var x = new Array(10);
```

Run code snippet Expand snippet

edited Aug 29 '18 at 8:56

vsync
52.4k 38 171 236

answered Jun 8 '09 at 18:28

Sufian
6,644 4 17 21

5 Can they use things like strings for their keys and values? myArray['Book']['item1'] ? – Diego Jun 8 '09 at 19:54

38 @Diego, yes, but that's not what arrays are intended for. It's better to use an object when your keys are strings. – Matthew Crumley Jun 8 '09 at 20:05

8 I like this example better than the accepted answer because this can be implemented for dynamically sized arrays, e.g. `new Array(size)` where `size` is a variable. – Variadicism Sep 12 '15 at 22:44

doesn't work - error on assignment. How has this answer got 280 likes if it is useless? – Gargo Sep 15 '16 at 18:51

1 This is working, thanks. You can see the example Gargo jsfiddle.net/matasoy/oetw73sj – matasoy Sep 23 '16 at 7:23

179

Similar to activa's answer, here's a function to create an n-dimensional array:

```
function createArray(length) {
    var arr = new Array(length || 0),
        i = length;

    if (arguments.length > 1) {
        var args = Array.prototype.slice.call(arguments, 1);
        while(i--) arr[length-1 - i] = createArray.apply(this, args);
    }

    return arr;
}

createArray();     // [] or new Array()

createArray(2);    // new Array(2)
```

yckart
21.8k 5 98 113

Matthew Crumley
81.8k 21 95 119

1 Can this create a 4 dimensional array? – trusktr May 19 '11 at 2:18

3 @trusktr: Yes, you could create as many dimensions as you want (within your memory constraints). Just pass in the length of the four dimensions. For example, `var array = createArray(2, 3, 4, 5);` . – Matthew Crumley May 19 '11 at 4:21

Nice! I actually asked about this here: stackoverflow.com/questions/6053332/javascript-4d-arrays and a variety of interesting answers. – trusktr May 19 '11 at 5:50

3 Best answer ! However, I would not recommend to use it with 0 or 1 parameters (useless) – Apolo May 15 '14 at 14:11

n-dimensional you say? Can this create a 5 dimensional array? – BritishDeveloper Jun 19 '15 at 22:19

79

Javascript only has 1-dimensional arrays, but you can build arrays of arrays, as others pointed out.

The following function can be used to construct a 2-d array of fixed dimensions:

```
function Create2DArray(rows) {
  var arr = [];

  for (var i=0;i<rows;i++) {
     arr[i] = [];
  }

  return arr;
}
```

The number of columns is not really important, because it is not required to specify the size of an array before using it.

Then you can just call:

```
var arr = Create2DArray(100);

arr[50][2] = 5;
```

Philippe Leybaert
136k 26 188 211

i want to make a 2-dim array that would represent a deck of cards. Which would be a 2-dim array that holds the card value and then in then the suit. What would be the easiest way to do that. – Doug Hauf Mar 3 '14 at 17:58

1 function Create2DArray(rows) { var arr = []; for (var i=0;i<rows;i++) { arr[i] = []; } return arr; } function print(deck) { for(t=1;t<=4;t++) { for (i=1;i<=13;i++) { document.writeln(deck[t][i]); } } } fucntion fillDeck(d) { for(t=1;t<=4;t++) { myCardDeck[t][1] = t; for (i=1;i<=13;i++) { myCardDeck[t][i] = i; } } } function loadCardDeck() { var myCardDeck = Create2DArray(13); fillDeck(myCardDeck); print(myCardDeck); } – Doug Hauf Mar 3 '14 at 17:58

2 @Doug: You actually want a one-dimensional array of objects with 2 attributes. var deck= []; deck[0]= { face:1, suit:'H'}; – TeasingDart Sep 18 '15 at 23:21

@DougHauf that's a minified 2D-array ?? :P :D – Mahi Nov 9 '16 at 12:50

70

The easiest way:

```
var myArray = [[]];
```

answered Jan 22 '13 at 18:47

Fred
749 5 2

28 which is a 2-dimension array – Maurizio In denmark Jul 2 '13 at 13:07

12 Yeah, careful with that. Assigning myArray[0][whatever] is fine, but try and set myArray[1][whatever] and it complains that myArray[1] is undefined. – Philip Apr 17 '14 at 16:24

19 @Philip you have to set `myArray[1]=[];` before assigning `myArray[1][0]=5;` – 182764125216 Sep 19 '14 at 20:14

4 Be aware, this does *not* "create an empty 1x1 array" as @AndersonGreen wrote. It creates a "1x0" array (i.e. 1 row containing an array with 0 columns). `myArray.length == 1` and `myArray[0].length == 0` . Which then gives the wrong result if you then copy a "genuinely empty" "0x0" array into it. – JonBrave Nov 17 '16 at 9:20

1 @182764125216 that was *knowledge of the day* for me. Thanks :) – th3pirat3 Feb 7 '18 at 23:20

44 one dimensional array of objects, each of those objects would be a one dimensional array consisting of two elements.

So, that's the cause of the conflicting view points.

answered Jun 8 '09 at 18:33

James Conigliaro
3,635 15 21

37 No, it's not. In some languages, you can have multidimensional arrays like `string[3,5] = "foo";` . It's a better approach for some scenarios, because the Y axis is not actually a child of the X axis. – Rafael Soares Aug 4 '11 at 15:29

2 Once it gets to the underlying machine code, all tensors of dimension > 1 are arrays of arrays, whichever language we are talking about. It is worthwhile keeping this in mind for reasons of cache optimisation. Any decent language that caters seriously for numerical computing will allow you to align your multidimensional structure in memory such that your most-used dimension is stored contiguously. Python's Numpy, Fortran, and C, come to mind. Indeed there are cases when it is worthwhile to reduce dimensionality into multiple structures for this reason. – Thomas Browne Oct 27 '14 at 18:18

Computers have no notion of dimensions. There is only 1 dimension, the memory address. Everything else is notational decoration for the benefit of the programmer. – TeasingDart Sep 18 '15 at 23:23

3 @ThomasBrowne Not exactly. "Arrays of arrays" require some storage for the sizes of inner arrays (they may differ) and another pointer dereferencing to find the place where an inner array is stored. In any "decent" language multidimentional arrays differ from jagged arrays, because they're different data structures per se. (And the confusing part is that C arrays are multidimentional, even though they're indexed with [a][b] syntax.) – polkovnikov.ph Dec 18 '15 at 23:20

How to create an empty two dimensional array (one-line)

35

```
Array.from(Array(2), () => new Array(4))
```

2 and 4 being first and second dimensions respectively.

We are making use of `Array.from` , which can take an array-like param and an optional mapping for each of the elements.

> Array.from(arrayLike[, mapFn[, thisArg]])

Run code snippet Expand snippet

The same trick can be used to Create a JavaScript array containing 1...N

Alternatively (but more inefficient 12% with `n = 10,000`)

```
Array(2).fill(null).map(() => Array(4))
```

The performance decrease comes with the fact that we have to have the first dimension values initialized to run `.map`. Remember that `Array` will not allocate the positions until you order it to through `.fill` or direct value assignment.

```
var arr = Array(2).fill(null).map(() => Array(4));
arr[0][0] = 'foo';
console.info(arr);
```

Run code snippet Expand snippet

Follow up

Here's a method that appears correct, **but has issues.**

```
Array(2).fill(Array(4)); // BAD! Rows are copied by reference
```

While it does return the apparently desired two dimensional array (`[ [ <4 empty items> ], [ <4 empty items> ] ]`), there a catch: first dimension arrays have been copied by reference. That means a `arr[0][0] = 'foo'` would actually change two rows instead of one.

Run code snippet Expand snippet

edited May 12 at 23:13

Trent
2,511 4 18 36

answered Mar 9 '18 at 19:53

zurfyx
14.2k 10 77 111

I suggest this: `Array.from({length:5}, () => [])` – vsync Aug 29 '18 at 9:11

29

Few people show the use of push:
To bring something new, I will show you how to initialize the matrix with some value, example: 0 or an empty string "".
Reminding that if you have a 10 elements array, in javascript the last index will be 9!

```
function matrix( rows, cols, defaultValue){

  var arr = [];

  // Creates all lines:
  for(var i=0; i < rows; i++){

      // Creates an empty line
      arr.push([]);

      // Adds cols to the empty line:
      arr[i].push( new Array(cols));

      for(var j=0; j < cols; j++){
        // Initializes:
        arr[i][j] = defaultValue;
      }
  }

return arr;
}
```

answered Aug 8 '13 at 2:22

Sergio Abreu
1,757 18 13

26

Two-liner:

```
var a = [];
while(a.push([]) < 10);
```

It will generate an array a of the length 10, filled with arrays. (Push adds an element to an array and returns the new length)

answered May 26 '14 at 10:00

domenukk
789 13 24

12 One-liner: `for (var a=[]; a.push([])<10;);` ? – Bergi Jul 7 '14 at 22:07

@Bergi will the a variable still be defined in the next line..? – StinkyCat Apr 11 '16 at 10:48

1 @StinkyCat: Yes, that's how `var` works. It's always function-scoped. – Bergi Apr 11 '16 at 10:51

I know, therefore your one-liner is useless in this case: you cannot "access its members" (check question) – StinkyCat Apr 11 '16 at 11:05

1 domenukk and @Bergi, you're both correct. I tried it out and I can access a after the for. I apologize! and thank you, may this be a lesson to me ;) – StinkyCat Apr 13 '16 at 14:06

24

The sanest answer seems to be

```
var nrows = ~~(Math.random() * 10);
var ncols = ~~(Math.random() * 10);
console.log(`rows:${nrows}`);
```

Run code snippet Expand snippet

Note we can't **directly** fill with the rows since fill uses shallow copy constructor, therefore all rows would share the *same* memory...here is example which demonstrates how each row would be shared (taken from other answers):

```
// DON'T do this: each row in arr, is shared
var arr = Array(2).fill(Array(4));
arr[0][0] = 'foo'; // also modifies arr[1][0]
console.info(arr);
```

edited Jul 28 '18 at 11:00
giorgi moniava
14.3k 3 24 61

answered Feb 15 '16 at 4:19
Francesco Dondi
849 6 15

It works. jsfiddle.net/trdnhy9q – VladSavitsky Mar 15 '16 at 15:26

This should be at the very top. I did something similar using `Array.apply(null, Array(nrows))` but this is much more elegant. – dimiguel Mar 25 '16 at 6:05

This regard my last comment... Internet Explorer and Opera don't have support for `fill` . This won't work on a majority of browsers. – dimiguel Mar 25 '16 at 20:50

@dimgl Fill can be emulated in this instance with a constant map: `Array(nrows).map(() => 0)` , or, `Array(nrows).map(function(){ return 0; });` – Conor O'Brien Jan 17 '17 at 18:56

18

The easiest way:

```
var arr  = [];

var arr1 = ['00','01'];
var arr2 = ['10','11'];
var arr3 = ['20','21'];
```

```
alert(arr[1][1]); // '11'
alert(arr[2][0]); // '20'
```

edited Dec 27 '13 at 13:32

answered Dec 27 '13 at 8:59

Chicharito

892 5 25 47

18

This is what i achieved :

```
var appVar = [[]];
appVar[0][4] = "bineesh";
appVar[0][5] = "kumar";
console.log(appVar[0][4] + appVar[0][5]);
console.log(appVar);
```

Run code snippet Expand snippet

This spelled me bineeshkumar

edited Sep 10 '16 at 6:35

Ruslan López

3,129 1 16 28

answered May 18 '16 at 15:16

Bineesh

388 1 3 17

1 Notice how you can only access the 0 index of the parent array. This isn't as useful as something which allows you to set, for example, appVar[5][9] = 10; ... you would get 'Unable to set property "9" of undefined' with this. – RaisinBranCrunch Jul 30 '17 at 16:38

But `appVar[1][4] = "bineesh";` is wrong, how to solve it? – Gank May 20 '18 at 13:50

Two dimensional arrays are created the same way single dimensional arrays are. And you access them like `array[0][1]` .

answered Jun 8 '09 at 18:27

tj111
18.6k 6 54 75

13

I'm not sure if anyone has answered this but I found this worked for me pretty well -

```
var array = [[,],[,]]
```

eg:

```
var a = [[1,2],[3,4]]
```

For a 2 dimensional array, for instance.

edited Jul 1 '12 at 6:50

Nikson Kanti Paul
2,902 1 25 45

answered Jul 1 '12 at 3:28

Uchenna
139 1 2

How can I do this dynamically? I want the inner arrays with different sizes. – alap Jan 19 '14 at 16:48

3 You don't need extra commas `var array = [[],[]]` is adequate. – Kaya Toast Jan 31 '15 at 7:29

13

To create a 2D array in javaScript we can create an Array first and then add Arrays as it's elements. This method will return a 2D array with the given number of rows and columns.

```
function Create2DArray(rows,columns) {
   var x = new Array(rows);
   for (var i = 0; i < rows; i++) {
       x[i] = new Array(columns);
   }
```

```
var array = Create2DArray(10,20);
```

edited Jun 26 '14 at 6:56

answered Jun 25 '14 at 11:38

prime

6,077 6 51 85

2 Please would you add some explanatory information to your ansdwer showing how it works, and why it solves the problem. This will help others who find this page in the future – Our Man in Bananas Jun 25 '14 at 12:16

When would you need an Array that is preinitialized with a certain number of colums in Javascript? You can access the n-th element of a [] array as well. – domenukk Jul 8 '14 at 14:49

I noticed the function starts with capital C, which (by certain conventions) suggest it would be a Function constructor and you would use it with the new keyword. A very minor and somewhat opinionated maybe, but I would still suggest un-capitalized word. – Hachi Aug 24 '14 at 5:53

12

To create a non-sparse "2D" array (x,y) with all indices addressable and values set to null:

```
let 2Darray = new Array(x).fill(null).map(item =>(new Array(y).fill(null)))
```

bonus "3D" Array (x,y,z)

```
let 3Darray = new Array(x).fill(null).map(item=>(new
Array(y).fill(null)).map(item=>Array(z).fill(null)))
```

Variations and corrections on this have been mentioned in comments and at various points in response to this question but not as an actual answer so I am adding it here.

It should be noted that (similar to most other answers) this has O(x*y) time complexity so it probably not suitable for very large arrays.

answered Sep 8 '18 at 18:31

Justin Ohms

Use Array Comprehensions

10

In JavaScript 1.7 and higher you can use **array comprehensions** to create two dimensional arrays. You can also filter and/or manipulate the entries while filling the array and don't have to use loops.

```
var rows = [1, 2, 3];
var cols = ["a", "b", "c", "d"];

var grid = [ for (r of rows) [ for (c of cols) r+c ] ];

/*
         grid = [
            ["1a","1b","1c","1d"],
            ["2a","2b","2c","2d"],
            ["3a","3b","3c","3d"]
         ]
*/
```

You can create any `n x m` array you want and fill it with a default value by calling

```
var default = 0;  // your 2d array will be filled with this value
var n_dim = 2;
var m_dim = 7;

var arr = [ for (n of Array(n_dim)) [ for (m of Array(m_dim) default ]]
/*
         arr = [
            [0, 0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0],
         ]
*/
```

More examples and documentation can be found here.

A quick google check here... yup... the `for` statement is still a loop... – Pimp Trizkit Mar 10 '18 at 15:25

8

For one liner lovers Array.from()

```
// creates 8x8 array filed with "0"
const arr2d = Array.from({ length: 8 }, () => Array.from({ length: 8 }, () => "0"));
```

Another one (from comment by dmitry_romanov) use Array().fill()

```
// creates 8x8 array filed with "0"
const arr2d = Array(8).fill(0).map(() => Array(8).fill("0"));
```

answered Jul 1 '17 at 20:00

my-
127 1 9

2 we can remove `0` in the first `fill()` function: `const arr2d = Array(8).fill().map(() => Array(8).fill("0"));` – Jinsong Li Nov 21 '17 at 14:38

8

My approach is very similar to @Bineesh answer but with a more general approach.

You can declare the double array as follows:

```
var myDoubleArray = [[]];
```

And the storing and accessing the contents in the following manner:

```
var testArray1 = [9,8]
```

```
myDoubleArray[index++] = testArray3;

console.log(myDoubleArray[0],myDoubleArray[1][3], myDoubleArray[2]['test'],)
```

This will print the expected output

```
[ 9, 8 ] 9 123
```

edited Jan 14 '18 at 0:25

answered Dec 4 '16 at 4:36

Alexander

358 4 13

6

I found below is the simplest way:

```
var array1 = [[]];
array1[0][100] = 5;

alert(array1[0][100]);
alert(array1.length);
alert(array1[0].length);
```

answered Apr 15 '14 at 9:14

GMsoF

3,764 10 54 102

5

I had to make a flexible array function to add "records" to it as i needed and to be able to update them and do whatever calculations e needed before i sent it to a database for further processing. Here's the code, hope it helps :).

```
function Add2List(clmn1, clmn2, clmn3) {
    aColumns.push(clmn1,clmn2,clmn3); // Creates array with "record"
    aLine.splice(aPos, 0,aColumns);  // Inserts new "record" at position aPos in main
```

Feel free to optimize and / or point out any bugs :)

answered Oct 12 '12 at 10:59

CJ Mendes
189 4 10

How about just `aLine.push([clmn1, clmn2, clmn3]);` ? – Pimp Trizkit Mar 10 '18 at 15:55

5

Javascript does not support two dimensional arrays, instead we store an array inside another array and fetch the data from that array depending on what position of that array you want to access. Remember array numeration starts at **ZERO**.

Code Example:

```
/* Two dimensional array that's 5 x 5

       C0 C1 C2 C3 C4
    R0[1][1][1][1][1]
    R1[1][1][1][1][1]
    R2[1][1][1][1][1]
    R3[1][1][1][1][1]
    R4[1][1][1][1][1]
*/

var row0 = [1,1,1,1,1],
    row1 = [1,1,1,1,1],
    row2 = [1,1,1,1,1],
    row3 = [1,1,1,1,1],
    row4 = [1,1,1,1,1];

var table = [row0,row1,row2,row3,row4];
console.log(table[0][0]); // Get the first item in the array
```

answered Jul 30 '15 at 23:53

Rick
6,591 2 24 35

5

```
function createArray(x, y) {
    return Array.apply(null, Array(x)).map(e => Array(y));
}
```

You can easily turn this function into an ES5 function as well.

```
function createArray(x, y) {
    return Array.apply(null, Array(x)).map(function(e) {
        return Array(y);
    });
}
```

Why this works: the `new Array(n)` constructor creates an object with a prototype of `Array.prototype` and then assigns the object's `length`, resulting in an unpopulated array. Due to its lack of actual members we can't run the `Array.prototype.map` function on it.

However, when you provide more than one argument to the constructor, such as when you do `Array(1, 2, 3, 4)`, the constructor will use the `arguments` object to instantiate and populate an `Array` object correctly.

For this reason, we can use `Array.apply(null, Array(x))`, because the `apply` function will spread the arguments into the constructor. For clarification, doing `Array.apply(null, Array(3))` is equivalent to doing `Array(null, null, null)`.

Now that we've created an actual populated array, all we need to do is call `map` and create the second layer (`y`).

answered Mar 25 '16 at 5:56

dimiguel
785 9 24

5

Below one, creates a 5x5 matrix and fill them with `null`

```
var md = [];
for(var i=0; i<5; i++) {
    md.push(new Array(5).fill(null));
}
```

edited Aug 29 '18 at 6:53

Javascript Lover - SKT
1,469 1 8 28

answered Oct 23 '16 at 5:33

zeah
145 1 5

3 This answer is wrong. It will create an array with same array filling in its slots. `md[1][0] = 3` and all the rest of elements are updated too – Qiang Nov 15 '16 at 6:33

1 @Qiang - yes you are right. Edited my post. – zeah Nov 21 '16 at 5:42

4

```
var playList = [
  ['I Did It My Way', 'Frank Sinatra'],
  ['Respect', 'Aretha Franklin'],
  ['Imagine', 'John Lennon'],
  ['Born to Run', 'Bruce Springsteen'],
  ['Louie Louie', 'The Kingsmen'],
  ['Maybellene', 'Chuck Berry']
];

function print(message) {
  document.write(message);
}

function printSongs( songs ) {
  var listHTML;
  listHTML = '<ol>';
  for ( var i = 0; i < songs.length; i += 1) {
    listHTML += '<li>' + songs[i][0] + ' by ' + songs[i][1] + '</li>';
  }
  listHTML += '</ol>';
  print(listHTML);
}

printSongs(playList);
```

antelove
815 6 7

3

You could allocate an array of rows, where each row is an array of the same length. Or you could allocate a one-dimensional array with rows*columns elements and define methods to map row/column coordinates to element indices.

Whichever implementation you pick, if you wrap it in an object you can define the accessor methods in a prototype to make the API easy to use.

answered Jun 8 '09 at 18:32

Nat
8,940 3 27 32

3

I found that this code works for me:

```
var map = [
    []
];

mapWidth = 50;
mapHeight = 50;
fillEmptyMap(map, mapWidth, mapHeight);
```

...

```
function fillEmptyMap(array, width, height) {
    for (var x = 0; x < width; x++) {
        array[x] = [];
        for (var y = 0; y < height; y++) {

            array[x][y] = [0];
        }
    }
```


53 7

A simplified example:

3

```
var blocks = [];

blocks[0] = [];

blocks[0][0] = 7;
```

edited Sep 21 '15 at 7:05

Anders
5,882 6 34 57

answered Sep 21 '15 at 5:17

rickatech
461 4 8

One liner to create a m*n 2 dimensional array filled with 0.

3

```
new Array(m).fill(new Array(n).fill(0));
```

answered Jan 3 '17 at 1:07

geniuscarrier
3,749 2 16 15

3 Actually, this will create only two arrays. Second dimensions is going to be the same array in every index. – Pijusn Mar 7 '17 at 19:07

6 Yes, I confirm the gotcha. Quick fix: `a = Array(m).fill(0).map(() => Array(n).fill(0))` ? `map` will untie reference and create unique array per slot. – dmitry_romanov Apr 15 '17 at 4:28

I've made a modification of Matthew Crumley's answer for creating a multidimensional array function. I've added the dimensions of the

```
*
*   @param array dimensions
*   @param any type value
*
*   @return array array
 */
function createArray(dimensions, value) {
    // Create new array
    var array = new Array(dimensions[0] || 0);
    var i = dimensions[0];

    // If dimensions array's length is bigger than 1
    // we start creating arrays in the array elements with recursions
    // to achieve multidimensional array
    if (dimensions.length > 1) {
        // Remove the first value from the array
        var args = Array.prototype.slice.call(dimensions, 1);
        // For each index in the created array create a new array with recursion
        while(i--) {
            array[dimensions[0]-1 - i] = createArray(args, value);
        }
    // If there is only one element left in the dimensions array
    // assign value to each of the new array's elements if value is set as param
    } else {
        if (typeof value !== 'undefined') {
            while(i--) {
                array[dimensions[0]-1 - i] = value;
            }
        }
    }

    return array;
}

createArray([]);              // [] or new Array()

createArray([2], 'empty');    // ['empty', 'empty']

createArray([3, 2], 0);       // [[0, 0],
                              //  [0, 0],
                              //  [0, 0]]
```

1 2 next

protected by Mysticial Jul 24 '14 at 5:56

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?