Get column from a two dimensional array

Asked 7 years, 11 months ago Active 8 months ago Viewed 100k times

51

16

How can I retrieve a column from a **2-dimensional array** and not a single entry? I'm doing this because I want to search for a string in one of the columns only so if there is another way to accomplish this please tell me.

I'm using the array defined this way:

```
var array=[];
```

At the end the size of this array is 20(col)x3(rows) and I need to read the first row and check the existence of some phrase in it.

javascript

edited Jan 25 at 11:42
Thor84no
4,751 1 23 48

asked Oct 21 '11 at 10:16
Ameen
902 1 16 29

7 Answers

21

You have to loop through each element in the 2d-array, and get the *n*th column.

```
function getCol(matrix, col){
   var column = [];
   for(var i=0; i<matrix.length; i++){
      column.push(matrix[i][col]);
   }
   return column;
}

var array = [new Array(20), new Array(20), new Array(20)]; //..your 3x20 array
```

edited Aug 2 '12 at 18:43
Reisepass
83 2 5

answered Oct 21 '11 at 10:22
Rob W
284k 55 672 598

5 I knew that i can do it this way, but i thought there was a predefined function or method that retrieves this data. Anyways, thanks for the answer appreciate it. – Ameen Oct 21 '11 at 14:19

109

Taking a column is easy with the map function.

```
// a two-dimensional array
var two_d = [[1,2,3],[4,5,6],[7,8,9]];

// take the third column
var col3 = two_d.map(function(value,index) { return value[2]; });
```

Why bother with the slice at all? Just filter the matrix to find the rows of interest.

```
var interesting = two_d.filter(function(value,index) {return value[1]==5;});
// interesting is now [[4,5,6]]
```

Sadly, filter and map are not natively available on IE9 and lower. The MDN documentation provides implementations for browsers without native support.

answered Oct 20 '12 at 6:40
Leif Carlsen
2,689 3 13 10

31

Use `Array.prototype.map()` with an arrow function:

```
const arrayColumn = (arr, n) => arr.map(x => x[n]);

const twoDimensionalArray = [
```

```
];

console.log(arrayColumn(twoDimensionalArray, 0));
```

Run code snippet Expand snippet

Note: `Array.prototype.map()` and arrow functions are part of ECMAScript 6 and not supported everywhere, see ECMAScript 6 compatibility table.

edited Dec 1 '16 at 19:28

answered Jan 24 '16 at 17:49

Michał Perłakowski
50.8k 16 117 133

2 Is it just me, or is this unnecessarily complicated in js? At any rate, that worked, thanks @Michal. – James Feb 25 '18 at 4:08

4

You can use the following array methods to obtain a column from a 2D array:

Array.prototype.map()

```
const array_column = (array, column) => array.map(e => e[column]);
```

Array.prototype.reduce()

```
const array_column = (array, column) => array.reduce((a, c) => {
  a.push(c[column]);
  return a;
}, []);
```

Array.prototype.forEach()

```
const array_column = (array, column) => {
  const result = [];
```

```
  });

  return result;
};
```

If your 2D array is a square (the same number of columns for each row), you can use the following method:

Array.prototype.flat() / .filter()

```
const array_column = (array, column) => array.flat().filter((e, i) => i % array.length
=== column);
```

answered Nov 2 '18 at 1:52

Grant Miller

9,007 13 49 77

2

This function works to arrays and objects. obs: it works like array_column php function. It means that an optional third parameter can be passed to define what column will correspond to the indices of return.

```
function array_column(list, column, indice){
    var result;

    if(typeof indice != "undefined"){
        result = {};

        for(key in list)
            result[list[key][indice]] = list[key][column];
    }else{
        result = [];

        for(key in list)
            result.push( list[key][column] );
    }

    return result;
}
```

```
function array_column_conditional(list, column, indice){
    var result;

    if(typeof indice != "undefined"){
        result = {};

        for(key in list)
            if(typeof list[key][column] !== 'undefined' && typeof list[key][indice] !==
'undefined')
                result[list[key][indice]] = list[key][column];
    }else{
        result = [];

        for(key in list)
            if(typeof list[key][column] !== 'undefined')
                result.push( list[key][column] );
    }

    return result;
}
```

usability:

```
var lista = [
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]
];

var obj_list = [
  {a: 1, b: 2, c: 3},
  {a: 4, b: 5, c: 6},
  {a: 8, c: 9}
];

var objeto = {
  d: {a: 1, b: 3},
  e: {a: 4, b: 5, c: 6},
  f: {a: 7, b: 8, c: 9}
};

var list_obj = {
  d: [1, 2, 3],
```

```
console.log( "column list: ", array_column(lista, 1) );
console.log( "column obj_list: ", array_column(obj_list, 'b', 'c') );
console.log( "column objeto: ", array_column(objeto, 'c') );
console.log( "column list_obj: ", array_column(list_obj, 0, 0) );

console.log( "column list conditional: ", array_column_conditional(lista, 1) );
console.log( "column obj_list conditional: ", array_column_conditional(obj_list, 'b',
'c') );
console.log( "column objeto conditional: ", array_column_conditional(objeto, 'c') );
console.log( "column list_obj conditional: ", array_column_conditional(list_obj, 0, 0)
);
```

Output:

```
/*
column list:  Array [ 2, 5, 8 ]
column obj_list:  Object { 3: 2, 6: 5, 9: undefined }
column objeto:  Array [ undefined, 6, 9 ]
column list_obj:  Object { 1: 1, 4: 4, 7: 7 }

column list conditional:  Array [ 2, 5, 8 ]
column obj_list conditional:  Object { 3: 2, 6: 5 }
column objeto conditional:  Array [ 6, 9 ]
column list_obj conditional:  Object { 1: 1, 4: 4, 7: 7 }
*/
```

edited Jul 14 '16 at 16:23

answered Jan 8 '15 at 14:39

Doglas

325 4 17

1

```
function arrayColumn(arr, n) {
  return arr.map(x=> x[n]);
}

var twoDimensionalArray = [
  [1, 2, 3],
  [4, 5, 6]
```

```
console.log(arrayColumn(twoDimensionalArray, 1));
```

Run code snippet Expand snippet

answered Aug 16 '16 at 18:12

MEHNAZ HUSSAIN
31 7

This is a very clean method that leverages a built in JavaScript method for arrays. – Tim Holt Dec 1 '16 at 19:23

1

I have created a library matrix-slicer to manipulate with matrix items. So your problem could be solved like this:

```
var m = new Matrix([
    [1, 2],
    [3, 4],
]);

m.getColumn(1); // => [2, 4]
```

Possible it will be useful for somebody. ;-)

answered Jul 27 '18 at 13:26

Alexandr
36 6