|  |
| --- |
|  |
|  |  |
|  | **ARRAYS IN JAVASCRIPT**  **1. Sorting Number Array** |
|  |  |
|  | var array2=[1,2,34,8,0,6,3,12,56,23]; |
|  | var array1=[1,2,34,8,0,6,3,12,56,23]; |
|  |  |
|  | var result=array1.sort(); |
|  | //console.log(result); //o/p: [ 0, 1, 12, 2, 23, 3, 34, 56, 6, 8 ] |
|  |  |
|  | function myarrayresult(){ |
|  | array2.sort(function (a,b) { |
|  | return a-b; //For assending order |
|  | return b-a; //For descendening Order // Correct |
|  | }); |
|  | } |
|  | myarrayresult(); |
|  | //console.log(array2); //O/P: [ 0, 1, 2, 3, 6, 8, 12, 23, 34, 56 ] |
|  |  |
|  | **2. Sorting String Array** |
|  |  |
|  | var citys=["Hyderabad","Chennai","Banglore","Mumbai","Vizag","Delhi"]; |
|  | var cityresult=citys.sort(); |
|  | var cityresultrev=citys.sort().reverse(); |
|  | // console.log(cityresultrev); //o/p: [ 'Vizag', 'Mumbai', 'Hyderabad', 'Delhi', 'Chennai', 'Banglore' ] |
|  | //console.log(cityresult); //o/p: [ 'Banglore', 'Chennai', 'Delhi', 'Hyderabad', 'Mumbai', 'Vizag' ] |
|  | **3. Sort without sorting keyword** |
|  |  |
|  | var numArray = [45,67,89,34,23,1,7,46,33,98,6,2]; |
|  | for (var i = 0; i < numArray.length - 1; i++) { |
|  | var min = i; |
|  | for (var j = i + 1; j < numArray.length; j++) { |
|  | if (numArray[j] < numArray[min]) { |
|  | min = j; |
|  | } |
|  | } |
|  | if (min != i) { |
|  | var target = numArray[i]; |
|  | numArray[i] = numArray[min]; |
|  | numArray[min] = target; |
|  | } |
|  | } |
|  | //console.log(numArray); // O/P: [ 1, 2, 6, 7, 23, 33, 34, 45, 46, 67, 89, 98 ] |
|  | // Real Time EXAMPLE |
|  |  |
|  | var homes = [ |
|  | { "h\_id": "3", "city": "Dallas","state": "TX","zip": "75201","price": "300" }, |
|  | { "h\_id": "3", "city": "Bevery Hills","state": "CA","zip": "90210","price": "200" }, |
|  | { "h\_id": "5","city": "New York","state": "NY","zip": "00010","price": "100" } |
|  | ]; |
|  |  |
|  | homes.sort(function(a, b) { |
|  | return parseFloat(a.price) - parseFloat(b.price); |
|  | }); |
|  |  |
|  | //console.log(homes); |
|  | /\* |
|  | O/P: [ { h\_id: '5',city: 'New York',state: 'NY',zip: '00010',price: '100' }, |
|  | { h\_id: '4',city: 'Bevery Hills',state: 'CA',zip: '90210',price: '200'}, |
|  | { h\_id: '3',city: 'Dallas',state: 'TX',zip: '75201',price: '300' } ] |
|  | \*/ |
|  |  |
|  | /\* ----------------------------------------------------------------------------- \*/ |
|  |  |
|  | **Q).How do I remove a particular element from an array in JavaScript?** |
|  |  |
|  | var array5=[1,3,6,7,8] |
|  | var removedele=array5.indexOf(6); // find the position of element in an array We use indexOf() |
|  | if(removedele !=-1){ |
|  | array5.splice(removedele,1); // correct method |
|  | } |
|  | //delete array5[array5.indexOf(6)]; Another way (it creates Empty item) o/p:[ 1, 3, <1 empty item>, 7, 8 ] |
|  | //console.log(array5); //o/p: [ 1, 3, 7, 8 ] |
|  |  |
|  | //2. |
|  | function mufunc(){ |
|  | this.name="kumar"; //public |
|  | var name="kumar"; // private |
|  | } |
|  | var obj=new mufunc(); |
|  | //console.log(obj.name); |
|  |  |
|  | **3.a).Difference between Slice VS Slice** |
|  |  |
|  | /\* The splice() method returns the removed item(s) in an array and slice() method returns the selected element(s) in an array, as a new array object. |
|  |  |
|  | The splice() method changes the original array and slice() method doesn’t change the original array. |
|  |  |
|  | The splice() method can take n number of arguments and slice() method takes 2 arguments. \*/ |
|  |  |
|  | var array6=[0,1,2,3,4,5,6,7,8,9]; |
|  |  |
|  | var sliceArray=array6.slice(3) // o/p: [ 3, 4, 5, 6, 7, 8, 9 ] , Result: 0,1,2,3,4,5,6,7,8,9 (Does not Changes the Original Array) |
|  | //console.log(sliceArray); |
|  |  |
|  | var spliceArray=array6.splice(3) // o/p: [ 3, 4, 5, 6, 7, 8, 9 ] , Result: 0,1,2 (Changes the Original Array) |
|  | //console.log(spliceArray); |
|  |  |
|  |  |
|  | var spliceArray2=array6.splice(1,2,"Hello") // o/p: [ 3, 4, 5, 6 ] , Result: 0,1,2,Hello,World,7,8,9 |
|  | //console.log(spliceArray2); |
|  |  |
|  | //console.log("Result:" + array6) |
|  |  |
|  | **4. ADD Key to Array Dynamically** |
|  |  |
|  | var data =[ |
|  | { "id": "3", "city": "Hyderabad", "price": "1200" }, |
|  | { "id": "2", "city": "chennai","price": "500" }, |
|  | { "id": "4", "city": "Banglore","price": "1000" }, |
|  | { "id": "5", "city": "pune","price": "1400" } |
|  | ]; |
|  |  |
|  | var objlength=Object.keys(data).length; |
|  | //(or) |
|  | var arraylength=data.length; |
|  | //console.log(objlength); //o/p:4 |
|  | //console.log(arraylength); //o/p:4 |
|  |  |
|  | var res=[]; |
|  |  |
|  | for(var i=0; i<data.length;i++) |
|  | { |
|  | data[i]["country"]="india"; |
|  | res.push(data[i]); |
|  | } |
|  | //console.log(data); |
|  | /\* o/p: [ { id: '3', city: 'Hyderabad', price: '1200', country: 'india' }, |
|  | { id: '2', city: 'chennai', price: '500', country: 'india' }, |
|  | { id: '4', city: 'Banglore', price: '1000', country: 'india' }, |
|  | { id: '5', city: 'pune', price: '1400', country: 'india' } ] \*/ |
|  |  |
|  | //console.log(res[2].city); |
|  |  |
|  | **5. To find Length of an Object** |
|  |  |
|  | var myobj={ "id": "6", "city": "Mumbai", "price": "1500" } |
|  | var objlength=Object.entries(myobj).length; //o/p: 3 [ [ 'id', '6' ], [ 'city', 'Mumbai' ], [ 'price', '1500' ] ] |
|  | var objlength=Object.values(myobj).length; //o/p: 3 [ '6', 'Mumbai', '1500' ] |
|  | var objlength=Object.keys(myobj).length; //o/p: 3 [ 'id', 'city', 'price' ] |
|  | var objlength=Object.getOwnPropertyNames(myobj).length //o/p: 3 [ 'id', 'city', 'price' ] |
|  | //console.log(objlength); |
|  | //o/p: 3 |
|  |  |
|  | **6. Array of Different Data types** |
|  |  |
|  | var diffarray=["kumar",22, 44,true,"Rajesh"]; |
|  | var stringArray=[]; |
|  |  |
|  | for(var i=0;i<diffarray.length;i++) |
|  | { |
|  | //console.log(typeof diffarray[i]); |
|  | if(typeof diffarray[i] == 'boolean'){ |
|  | //console.log("Boolean: "+diffarray[i]); |
|  | } |
|  | else if(typeof diffarray[i] == 'number'){ |
|  | //console.log("Numbers:"+ diffarray[i]); |
|  | } |
|  | else if(typeof diffarray[i] == 'string'){ |
|  | //console.log("String:"+ diffarray[i]); |
|  | stringArray.push(diffarray[i]); |
|  | } |
|  | } |
|  | //console.log(stringArray); //O/P: [ 'kumar', 'Rajesh' ] |
|  | //console.log(diffarray.length); // O/P : 5 |
|  |  |
|  | **(7). Array Methods** |
|  |  |
|  | var MethodArray=["java","php","html","sql","angular"]; |
|  | **1. tostring** |
|  | var ToString=MethodArray.toString(); // O/P: java,php,html,sql,angular |
|  | //console.log(ToString); |
|  | **2. revere array** |
|  | var ReverseArray=MethodArray.reverse(); // O/P: [ 'angular', 'sql', 'html', 'php', 'java' ] |
|  | //console.log(ReverseArray); |
|  |  |
|  | **3. Concat method** |
|  |  |
|  | var Arraylist1=["red","blue","yellow","orange"]; |
|  | var Arraylist2=["grey","black","green","white"]; |
|  | var Arraylist3=["gold","sky"]; |
|  |  |
|  | var Arrayconcat2=Arraylist1.concat(Arraylist2); |
|  | var Arrayconcat3=Arraylist1.concat(Arraylist2,Arraylist3); |
|  | // console.log(Arrayconcat2); // O/p: [ 'red','blue','yellow','orange','grey','black','green','white' ] |
|  | //console.log(Arrayconcat3); // o/p: [ 'red','blue','yellow','orange','grey','black', 'green', 'white', 'gold', 'sky' ] |
|  |  |
|  | **4.join** |
|  | var ArraySource=["A","C","K","L"]; |
|  | var tosourceArray=ArraySource.join('\*'); |
|  | console.log(tosourceArray) //O/P: A\*C\*K\*L |
|  |  |
|  | **5. every method** |
|  |  |
|  | //The every() method tests whether all elements in the array pass the test implemented by the provided function. |
|  |  |
|  | function isBelowThreshold(currentValue) { |
|  | return currentValue < 40; |
|  | } |
|  |  |
|  | var arrayevery = [1, 30, 39, 29, 10, 13]; |
|  |  |
|  | //console.log(arrayevery.every(isBelowThreshold)); //output: true |
|  |  |
|  | **6. some method** |
|  | // The some() method tests whether at least one element in the array passes the test implemented by the provided function. |
|  | var arraysome = [1, 2, 3, 4, 5]; |
|  | var even = function(element) { |
|  | // checks whether an element is even |
|  | return element % 2 === 0; |
|  | }; |
|  | console.log(arraysome.some(even)); // output: true |
|  | **7. map method** |
|  |  |
|  | //The map() method creates a new array with the results of calling a provided function on every element in the calling array. |
|  |  |
|  | var arraymap = [1, 4, 9, 16]; |
|  | // pass a function to map |
|  | const map1 = arraymap.map(x => x \* 2); |
|  | console.log(map1); // output: Array [2, 8, 18, 32] |
|  |  |
|  | **8. filter method** |
|  |  |
|  | //https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/Array/filter |
|  |  |
|  | //The filter() method creates a new array with all elements that pass the test implemented by the provided function. |
|  |  |
|  | var wordsfilter = ['spray', 'limit', 'elite', 'exuberant', 'destruction', 'present']; |
|  |  |
|  | const resultfilter = wordsfilter.filter(word => word.length > 6); |
|  |  |
|  | console.log(resultfilter); // output: ["exuberant", "destruction", "present"] |
|  |  |
|  | **9.includes method** |
|  |  |
|  | /\* The includes() method determines whether an array includes a certain element, returning true or false as appropriate. |
|  | It uses the sameValueZero algorithm to determine whether the given element is found. \*/ |
|  | var arrayincludes = [1, 2, 3]; |
|  | console.log(arrayincludes.includes(2)); // output: true |
|  | var pets = ['cat', 'dog', 'bat']; |
|  | console.log(pets.includes('cat')); // output: true |
|  | console.log(pets.includes('at')); // output: false |
|  |  |
|  | **10.isArray() method** |
|  | //The Array.isArray() method determines whether the passed value is an Array. |
|  | syntax: Array.isArray(obj) |
|  | Array.isArray([1, 2, 3]); // true |
|  | Array.isArray({foo: 123}); // false |
|  | Array.isArray('foobar'); // false |
|  | Array.isArray(undefined); // false |
|  | **11. isArray VS instanceof Array** |
|  |  |
|  | **Q) How do you check if a variable is an array in JavaScript?** |
|  | /\* |
|  | A) 1. variable.constructor === Array |
|  | 2. variable.prop && variable.prop.constructor === Array |
|  | 3. variable instanceof Array |
|  | 4. Array.isArray(variable) |
|  | 5. Object.prototype.toString.call(variable) === '[object Array]'; \*/ |
|  |  |
|  | var value={"A":'a',"C":'c',"V":'v'}; |
|  | var data=["A","F","M"]; |
|  | if (data instanceof Array) { |
|  | console.log('value is Array!'); // for data variable |
|  | } else { |
|  | console.log('Not an array'); // for value variable |
|  | } |
|  |  |
|  | console.log( Array.isArray(data)); // o/p: true for data or false for value |
|  | console.log( data.constructor === Array) // true |
|  | var myarray= Object.prototype.toString.call(value) === '[object Object]'; // true for value ,false for data |
|  | var myarray= Object.prototype.toString.call(data) === '[object Array]'; // true for data ,false for value |
|  | console.log(myarray); // true |
|  | console.log(data instanceof Array) // true |
|  |  |
|  | //conclision: Array.IsArray would be better to use. Array.IsArray is faster |
|  |  |
|  | **11.Adding And Deleting methods:** |
|  |  |
|  |  |
|  | 1.The JavaScript method toString() converts an array to a string of (comma separated) array values. |
|  | 2.The pop() method removes the last element from an array: |
|  | 3.The push() method adds a new element to an array (at the end): |
|  | 4.The shift() method removes the first array element and "shifts" all other elements to a lower index. |
|  | 5.The unshift() method adds a new element to an array (at the beginning), and "unshifts" older elements: |
|  |  |
|  |  |
|  | var Arraycurd=["nokia","samsung","redmi","Apple","micromax"]; |
|  |  |
|  | //var Arraycurdpush=Arraycurd.push('moto'); //o/p: [ 'nokia', 'samsung', 'redmi', 'Apple', 'micromax', 'moto' ] |
|  | //var Arraycurdpop=Arraycurd.pop(); //o/p: [ 'nokia', 'samsung', 'redmi', 'Apple' ] |
|  | //var Arraycurdshift=Arraycurd.shift(); //o/p: [ 'samsung', 'redmi', 'Apple', 'micromax' ] |
|  | //var Arraycurdunshift=Arraycurd.unshift('moto'); //o/p: [ 'moto', 'nokia', 'samsung', 'redmi', 'Apple', 'micromax' ] |
|  | var Arraycurdtosring=Arraycurd.toString(); |
|  | console.log(Arraycurd); |
|  | console.log(Arraycurdtosring); // o/p : nokia,samsung,redmi,Apple,micromax |
|  |  |
|  |  |
|  | **12. How to iterate an array in javascript?** |
|  | /\* |
|  | Iterating over arrays: |
|  | Options: |
|  | 1.Simple for loop. |
|  | 2.One of the iteration methods.(Array.prototype.forEach(),Array.prototype.every(),Array.prototype.some()) |
|  | 3.Never use for...in or foreach...in. \*/ |
|  |  |
|  | //ex: |
|  | var arr = ["apple", "pear", "orange"]; |
|  | arr.forEach(function(elem) { |
|  | console.log(elem); //o/p:apple pear orange |
|  | }); |
|  |  |
|  | //2. |
|  | const myStringArray = ["Hello", "World"]; |
|  | for (const s of myStringArray) { |
|  | console.log(s); //o/p: Hello World |
|  | // ... do something with s ... |
|  | } |
|  |  |
|  | /\* |
|  | Iterating over objects |
|  | Options: |
|  | Combine for...in with hasOwnProperty(), in the manner described above. |
|  | Combine Object.keys() or Object.getOwnPropertyNames() with forEach() array iteration. |
|  | \*/ |
|  |  |
|  | //1. |
|  |  |
|  | var data={ |
|  | "messages": [ |
|  | {"msgFrom": "13223821242", "msgBody": "Hi there"}, |
|  | {"msgFrom": "Bill", "msgBody": "Hello!"} |
|  | ] |
|  | } |
|  | for (var key in data.messages) { |
|  | var obj = data.messages[key]; |
|  | console.log(obj) // o/p: { msgFrom: '13223821242', msgBody: 'Hi there' } { msgFrom: 'Bill', msgBody: 'Hello!' } |
|  | } |
|  |  |
|  | //2. |
|  | var obj = { first: "John", last: "Doe" }; |
|  | // Visit non-inherited enumerable keys |
|  | Object.keys(obj).forEach(function(key) { |
|  | console.log(key); //o.p: first last |
|  | }); |
|  |  |
|  | **13.Associative Array** |
|  | /\* 1.An associative array is declared or dynamically created |
|  | 2.An associative array is also an object \*/ |
|  | var arr = { "one": 1, "two": 2, "three": 3 }; |
|  | **14. removeDuplicatesArray** |
|  | var a=["a","b","a"]; |
|  | function removeDuplicatesArray(a) { |
|  | return a.filter(function(item, pos) { |
|  | return a.indexOf(item) == pos; |
|  | }) |
|  | } |
|  | removeDuplicatesArray(a); |