|  |  |
| --- | --- |
| **Properties** | **Description** |
| length | A read/write property indicating the current number of elements within the array. You may set this property to dynamically expand an array's length. |
| prototype | Use this property to attach additional properties and/or methods that get reflected in all instances of the array. |

**http://www.javascriptkit.com/jsref/bullet.gifMethods**

**Note: "[]" surrounding a parameter below means the parameter is optional.**

|  |  |
| --- | --- |
| **Methods** | **Description** |
| concat(value1, ...) | Concatenates either plain values or another array with the existing array, and returns the new array. Does **NOT** alter the original array. [Example.](http://www.javascriptkit.com/jsref/arrays.shtml#e2)  Note: If the values to concat are strings or numbers, their actual values are added to the returned array. If the values to concat are object references, the same object reference will be added, and not the object itself. This means that both the old and new array will now contain references to those object(s), with changes to the referenced object affecting both arrays. |
| every(testfunction[thisobj])  *This is a* ***JavaScript1.6*** *feature, supported in Firefox1.5+ but NOT IE7 or below.* | Traverses an array and only returns true if all elements within it satisfies the condition set forth by testfunction(). Use it for example to test if all values within an array is greater than 0. Compliments the array method some()  testfunction() is the function reference containing the desired test. Its syntax must conform to the below:  ***testfunction(elementValue, elementIndex, targetArray){ }***  The three arguments are passed implicitly into the function, containing the current element value, index, plus the array being manipulated.  **Example:** Here's an example that quickly scans an array to make sure all of its elements' value is greater than 0.  var numbersarray=[2, 4, 5, -1, 34]  function isZeroAbove(element, index, array) {  return (element > 0) }  if (numbersarray.every(isZeroAbove)) //evaluates to false  alert("All elements inside array is above 0 in value!")  The testfunction() accepts an optional parameter "thisobj", which you can pass in a different object to be used as the "this" reference within the function. |
| filter(testfunction[thisobj])  *This is a* ***JavaScript1.6*** *feature, supported in Firefox1.5+ but NOT IE7 or below.* | Returns a new array containing all elements of the existing array that pass the condition set forth by testfunction(). Original array is not changed. Use it to filter down an array based on the desired condition.  testfunction() is the function reference containing the desired code to execute:  var numbersarray=[-3, 5, 34, 19]  function greaterThanFive(element, index, array) {  return (element > 5) }  var FiveplusArray=numbersarray.filter(greaterThanFive) //new array contains [34, 19] |
| foreach(testfunction[thisobj])  *This is a* ***JavaScript1.6*** *feature, supported in Firefox1.5+ but NOT IE7 or below.* | Iterates through the array and executes function testfunction() on each of its element. The processing function **cannot** return a value, so the result must be handled immediately. Use it for example to print out the value of all of the array elements.  testfunction() is the function reference containing the desired code to execute:  var numbersarray=[-3, 5, 34, 19]  function outputarray(element, index, array) {  document.write("Element "+index+" contains the value "+element+"<br />")  }  numbersarray.forEach(outputarray)  //Output: //Element 0 contains the value -3 //Element 1 contains the value 5 //Element 2 contains the value 34 //Element 3 contains the value 19 |
| indexOf(targetElement,  [startIndex])  *This is a* ***JavaScript1.6*** *feature, supported in Firefox1.5+ but NOT IE7 or below.* | Returns the first index in which targetElment (value) is found within an array, or -1 if nothing is found. An optional [startIndex] lets you specify the position in which to begin the search (default is 0, or search entire array):  var fruits=["Apple", "Oranges", "Pork", "Chicken"]  alert(fruits.indexOf("Pork")) //alerts 2 |
| join([separator]) | Converts each element within the array to a string, and joins them into one large string. Pass in an optional separator as argument to be used to separate each array element. If none is passed, the default comma (') is used:  var fruits=["Apple", "Oranges"] var result1=fruits.join() //creates the String "Apple,Oranges" var result2=fruits.join("\*") //creates the String "Apple\*Oranges" |
| lastIndexOf(targetElement,  [startIndex]  *This is a* ***JavaScript1.6*** *feature, supported in Firefox1.5+ but NOT IE7 or below.* | Returns the first index in which targetElment (value) is found within an array starting from the last element and backwards, or -1 if nothing is found. An optional [startIndex] lets you specify the position in which to begin the search (default is array.length-1, or search entire array). |
| map(testfunction[thisobj])  *This is a* ***JavaScript1.6*** *feature, supported in Firefox1.5+ but NOT IE7 or below.* | Returns a new array based on the return value of testfunction() on each of the array elements. Original array is not changed. Use it to transform the values of all elements within an array using some logic and derive the results as a new array.  testfunction() is the function reference containing the desired code to execute:  var numbersarray=[-3, 5, 34, 19]  function doubleIt(element, index, array) {  return (element\*2) }  var doubledarray=numbersarray.map(doubleIt) returns [-6, 10, 68, 38] |
| pop() | Deletes the last element within array and returns the deleted element. Original array is modified. |
| push(value1, ...) | Adds the argument values to the end of the array, and modifies the original array with the new additions. Returns the new length of the array. |
| reverse() | Reverses the order of all elements within the array. Original array is modified. |
| shift() | Deletes and returns the first element within the array. Original array is modified to account for the missing element (so 2nd element now becomes the first etc).  See also unshift(value) below, which does the opposite by adding a new value to the beginning of an array (so formerly 1st element becomes 2nd, 2nd becomes 3rd etc). |
| slice(start, [end]) | Returns a "slice" of the original array based on the start and end arguments. Original array is not changed. The slice includes the new array referenced by the start index and up to but NOT including the end index itself. If "end" is not specified, the end of the array is assumed. |
| splice(startIndex, [how\_many], [value1, ...]) | Deletes how\_many array elements starting from startIndex, and optionally replaces them with value1, value2 etc. Original array is modified. You can use splice() to delete an element from an array. [Example(s).](http://www.javascriptkit.com/jsref/arrays.shtml#e3)  This method returns the elements deleted from array. |
| some(testfunction[thisobj])  *This is a* ***JavaScript1.6*** *feature, supported in Firefox1.5+ but NOT IE7 or below.* | Traverses an array and returns true if any one of its elements satisfies the condition set forth by testfunction(). Use it for example to test if at least one element value within the array is above 0. Compliments the array method every() [Example(s).](http://www.javascriptkit.com/jsref/arrays.shtml#e2)  testfunction() is the function reference containing the desired code to execute:  var numbersarray=[-3, -34, 1, 32, -100]  function isZeroAbove(element, index, array) { return (element > 0) }  if (numbersarray.some(isZeroAbove)) //evaluates to true alert("One or more elements inside array is above 0 in value!")  The testfunction() accepts an optional parameter "thisobj", which you can pass in a different object to be used as the "this" reference within the function. |
| sort([SortFunction]) | By default sorts an array alphabetically and ascending. By passing in an optional SortFunction, you can sort numerically and by other criteria as well.  If SortFunction is defined, the array elements are sorted based on the relationship between each pair of elements within the array, "a" and "b", and your function's return value. The three possible return numbers are: <0 (less than 0), 0, or >0 (greater than 0):   * **Less than 0**: Sort "a" to be a lower index than "b" * **Zero**: "a" and "b" should be considered equal, and no sorting performed. * **Greater than 0**: Sort "b" to be a lower index than "a".   Take a look at the following 3 distinct examples:  **//Sort Alphabetically and ascending:** var myarray=["Bob","Bully","Amy"] myarray.sort() //Array now becomes ["Amy", "Bob", "Bully"]  **//Sort Alphabetically and descending:** var myarray=["Bob","Bully","Amy"] myarray.sort() myarray.reverse() //Array now becomes ["Bully", "Bob", "Amy"]  **//Sort numerically and ascending:** var myarray2=[25, 8, 7, 41] myarray2.sort(function(a,b){return a - b}) //Array now becomes [7, 8, 25, 41]  **//Sort numerically and descending:** var myarray2=[25, 8, 7, 41] myarray2.sort(function(a,b){return b - a}) //Array now becomes [41, 25, 8, 71]  **//Randomize the order of the array:** var myarray3=[25, 8, "George", "John"] myarray3.sort(function() {return 0.5 - Math.random()}) //Array elements now scrambled  To sort numerically, you need to compare the relationship between "a" to "b", with a return value of <0 indicating to sort ascending, and >0 to sort descending instead. In the case of the statement "return a - b", since whenever "a" is less than "b", a negative value is returned, it results in the array being sorted ascending (from small to large).  To sort randomly, you need to return a random number that can randomly be <0, 0, or >0, irrespective to the relationship between "a" and "b". |
| toSource() | Returns an array literal representing the specified array. |
| toString() | Returns a string representing the array and its elements. |
| unshift(value1, ...) | Adds the argument values to the beginning of the array, pushing existing arrays back. Returns the new length of the array. Original array is modified:  var fruits=["Apple", "Oranges"] fruits.unshift("Grapes") //fruits becomes ["Grapes", "Apples", "Oranges"] |
| valueOf() | Returns the primitive value of the array. |

**http://www.javascriptkit.com/jsref/bullet.gifExamples**

**http://www.javascriptkit.com/Red_CurlyC035.gifconcat(value1,...)**

var fruits=["Apple", "Oranges"]  
var meat=["Pork", "Chicken"]  
  
var dinner=fruits.concat(meat) //creates ["Apple", "Oranges", "Pork", "Chicken"]. fruits and meat arrays not changed.  
  
var snack=fruits.concat("Grapes", ["Cookies", "Milk"]) //creates ["Apple", "Oranges", "Grapes", "Cookies", "Milk"] fruits array not changed.

**http://www.javascriptkit.com/Red_CurlyC035.gifsplice(startIndex, [how\_many], [value1, ...])**

Splice() is one of those array functions that can use some explanation. Lets look at its parameters in detail:

1) **startIndex**- the array element to begin the insertion or deletion of the array.  
2) **how\_many**- the number of elements, beginning with the element specified in startIndex, to delete from the array. Optional. Not specifying this parameter causes all elements starting from startIndex to be deleted.  
3) **value1, value2 etc**- Optional values to be inserted into the array, starting at startIndex.

var myarray=[13, 36, 25, 52, 83]  
myarray.splice(2, 2) //myarray is now [13, 36 , 83]. The 3rd and 4th element is removed.

var myarray=[13, 36, 25, 52, 83] //reset array for 2nd example  
myarray.splice(2, 3, 42, 15) //myarray is now [13, 36 , 42, 15]. The 3rd, 4th and 5th element is removed, replaced with 42 and 15.