



HTML

CSS

MORE ▾



JavaScript Array Methods



Converting Arrays to Strings

The JavaScript method `toString()` converts an array to a string of (comma separated) array values.

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
document.getElementById("demo").innerHTML = fruits.toString();
```

Result:

Banana,Orange,Apple,Mango

The `join()` method also joins all array elements into a string.

It behaves just like `toString()`, but in addition you can specify the separator:

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
document.getElementById("demo").innerHTML = fruits.join(" * ");
```

Result:

Banana * Orange * Apple * Mango

Popping and Pushing

When you work with arrays, it is easy to remove elements and add new elements.

This is what popping and pushing is:

Popping items **out** of an array, or pushing items **into** an array.

Popping

The `pop()` method removes the last element from an array:

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.pop();           // Removes the last element ("Mango") from fruits
```

The `pop()` method returns the value that was "popped out":

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
var x = fruits.pop();      // the value of x is "Mango"
```

Pushing

The `push()` method adds a new element to an array (at the end):

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.push("Kiwi");      // Adds a new element ("Kiwi") to fruits
```

The `push()` method returns the new array length:

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
var x = fruits.push("Kiwi");  // the value of x is 5
```

Shifting Elements

Shifting is equivalent to popping, working on the first element instead of the last.

The `shift()` method removes the first array element and "shifts" all other elements to a lower index.

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.shift();           // Removes the first element "Banana" from fruits
```

The `shift()` method returns the string that was "shifted out":

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var x = fruits.shift();    // the value of x is "Banana"
```

The `unshift()` method adds a new element to an array (at the beginning), and "unshifts" older elements:

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
```

```
fruits.unshift("Lemon");    // Adds a new element "Lemon" to fruits
```

The `unshift()` method returns the new array length.

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.unshift("Lemon");    // Returns 5
```

Changing Elements

Array elements are accessed using their **index number**:

Array **indexes** start with 0. [0] is the first array element, [1] is the second, [2] is the third ...

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits[0] = "Kiwi";        // Changes the first element of fruits to "Kiwi"
```

The `length` property provides an easy way to append a new element to an array:

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[fruits.length] = "Kiwi";           // Appends "Kiwi" to fruits
```

Deleting Elements

Since JavaScript arrays are objects, elements can be deleted by using the JavaScript operator **delete** :

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
delete fruits[0];           // Changes the first element in fruits to undefined
```

Using **delete** may leave undefined holes in the array. Use `pop()` or `shift()` instead.

Splicing an Array

The **splice()** method can be used to add new items to an array:

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
```

```
fruits.splice(2, 0, "Lemon", "Kiwi");
```

The first parameter (2) defines the position **where** new elements should be **added** (spliced in).

The second parameter (0) defines **how many** elements should be **removed**.

The rest of the parameters ("Lemon" , "Kiwi") define the new elements to be **added**.

The `splice()` method returns an array with the deleted items:

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.splice(2, 2, "Lemon", "Kiwi");
```

Using splice() to Remove Elements

With clever parameter setting, you can use `splice()` to remove elements without leaving "holes" in the array:

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.splice(0, 1);           // Removes the first element of fruits
```

The first parameter (0) defines the position where new elements should be **added** (spliced

in).

The second parameter (1) defines **how many** elements should be **removed**.

The rest of the parameters are omitted. No new elements will be added.

Merging (Concatenating) Arrays

The `concat()` method creates a new array by merging (concatenating) existing arrays:

Example (Merging Two Arrays)

```
var myGirls = ["Cecilie", "Lone"];
var myBoys = ["Emil", "Tobias", "Linus"];
var myChildren = myGirls.concat(myBoys); // Concatenates (joins) myGirls and
myBoys
```

The `concat()` method does not change the existing arrays. It always returns a new array.

The `concat()` method can take any number of array arguments:

Example (Merging Three Arrays)

```
var arr1 = ["Cecilie", "Lone"];
var arr2 = ["Emil", "Tobias", "Linus"];
var arr3 = ["Robin", "Morgan"];
var myChildren = arr1.concat(arr2, arr3); // Concatenates arr1 with arr2 and
arr3
```


The `concat()` method can also take values as arguments:

Example (Merging an Array with Values)

```
var arr1 = ["Cecilie", "Lone"];  
var myChildren = arr1.concat("Emil", "Tobias", "Linus");
```

Slicing an Array

The `slice()` method slices out a piece of an array into a new array.

This example slices out a part of an array starting from array element 1 ("Orange"):

Example

```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];  
var citrus = fruits.slice(1);
```

The `slice()` method creates a new array. It does not remove any elements from the source array.

This example slices out a part of an array starting from array element 3 ("Apple"):

Example

```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
```

```
var citrus = fruits.slice(3);
```

The `slice()` method can take two arguments like `slice(1, 3)`.

The method then selects elements from the start argument, and up to (but not including) the end argument.

Example

```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];  
var citrus = fruits.slice(1, 3);
```

If the end argument is omitted, like in the first examples, the `slice()` method slices out the rest of the array.

Example

```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];  
var citrus = fruits.slice(2);
```

Automatic toString()

JavaScript automatically converts an array to a comma separated string when a primitive value is expected.

This is always the case when you try to output an array.

These two examples will produce the same result:

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
document.getElementById("demo").innerHTML = fruits.toString();
```

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
document.getElementById("demo").innerHTML = fruits;
```

All JavaScript objects have a `toString()` method.

Finding Max and Min Values in an Array

There are no built-in functions for finding the highest or lowest value in a JavaScript array.

You will learn how you solve this problem in the next chapter of this tutorial.

Sorting Arrays

Sorting arrays are covered in the next chapter of this tutorial.

Complete Array Reference

For a complete reference, go to our [Complete JavaScript Array Reference](#).

The reference contains descriptions and examples of all Array properties and methods.

Exercise:

Use the correct Array method to remove the **last item** of the `fruits` array.

```
var fruits = ["Banana", "Orange", "Apple"];  
          ;
```

Submit Answer »

[Start the Exercise](#)
