JavaScript Array.prototype.splice()

JavaScript's Array.prototype.splice() method removes one or more elements from any position in the array and returns the removed elements in a new array. It also allows you to add one or more elements to the middle of an array.

JavaScript's **Array.prototype.shift()** and **Array.prototype.pop()** methods allow you to remove elements from the *beginning* and *end* of a JavaScript array. These methods are simple to use and require no arguments because there is no potential for ambiguity: the concepts **"first element"** and **"last element"** require no further explanation. But when you want to remove one or more elements from the *middle* of a JavaScript array, there are details required. For example: *where* in the array do we want to start removing elements? Also, *how many* elements do we want to remove?

The **Array.prototype.splice()** method answers that question by removing one or more elements from any position in the array and returning the removed elements in a new array. Initially, this can throw you off because if you want to remove only one element, you would expect just that one element to be returned. But the **Array.prototype.splice()** method always returns an array. So, just keep in mind that if you plan to remove one element, you'll need to access the first element in the array that is returned.

The syntax for this is simple: you just pass a minimum of two numbers to the **splice()** method: the position in the array at which you want to start removing elements, and the number of elements to remove. In this case, you are only removing elements from the array. But you do have the option of adding as many additional parameters as you like. So, beginning with the 3rd parameter, you specify one or more elements to ***ADD*** to the array, starting at the position specified with the first parameter. For example: **myArray.splice(2, 3)** would remove three elements from **myArray**, starting at index # 2. But, **myArray.splice(2, 3, 'a', 'b', 'c')** would also add the strings '**a**', '**b**', '**c**' to the array starting at index # 2.

JavaScript's **Array.prototype.unshift()** and **Array.prototype.push()** methods allow you to remove elements from the *beginning* and end of a JavaScript array. But if you want to remove elements from the *middle* of an array, the **Array.prototype.splice()** method is the correct tool. In this case, you provide a zero as the 2nd argument, which means that you are saying: *"I do not want to remove any elements from the array"*. If you provide any arguments after the 2nd argument, however, then those will be added to the array starting at the position specified in the 1st argument. For example: **myArray.splice(2, 0, 'HELLO', 'GOODBYE').** Here, you'd be adding the strings '**HELLO**', '**GOODBYE**' to the array starting at position # 2. But keep in mind that in this case, the **Array.prototype.splice()** method will return an empty array, because that method always returns an array. But if you do not remove any elements from the original array, then an empty array is returned.

Try it yourself !

Edit in JSFiddle

- JavaScript
- Result

```
//--------------

var foo = ['a', 'b', 'c', 'd', 'e', 'f'];

console.warn('the 1st argument is the start position, and the 2nd
console.dir(foo.splice(0, 1)); // ['a']
console.info('..and the original array is changed:');
console.dir(foo); // [b', 'c', 'd', 'e', 'f']

//--------------

var foo = ['a', 'b', 'c', 'd', 'e', 'f'];

console.warn('start at the beginning, and remove two elements: foo
console.dir(foo.splice(0, 2)); // ['a', 'b']
console.info('..and the original array is changed:');
console.dir(foo); // ['c', 'd', 'e', 'f']

//--------------

var foo = ['a', 'b', 'c', 'd', 'e', 'f'];

console.warn('start on the third elemet, and remove three elements
console.dir(foo.splice(2, 3)); // ['c', 'd', 'e']
console.info('..and the original array is changed:');
console.dir(foo); // ["a","b","f"]

//--------------

var foo = ['a', 'b', 'c', 'd', 'e', 'f'];
console.warn('when passing no arguments, nothing is removed: foo.s
console.dir(foo.splice()); // []
console.info('..and the original array is changed:');
console.dir(foo); // ['a', 'b', 'c', 'd', 'e', 'f']

//--------------

var foo = ['a', 'b', 'c', 'd', 'e', 'f'];

console.warn('when the start position is a negative number, it rep
console.dir(foo.splice(-4, 3)); // ['c', 'd', 'e']
console.info('..and the original array is changed:');
console.dir(foo) // ["a", "b", "f"]
//--------------
```

In the above example, click the **JavaScript** tab. There we call the splice method on an array. In the first case, we take a very simple approach; the first argument is **0** and the second argument is **1: `foo.splice(0, 1)`**. This is similar to using the JavaScript **shift()** method, except that **shift()** returns the removed element, whereas the **splice()** method returns the removed element in an array. This is a very simple example, but the main takeaway is: the first argument is the *position to start at*, and the second argument is the *number of elements to remove.*

Later in the examples, we pass no arguments to the splice method. In this case, no elements are removed from the original array and an empty array is removed.

Click the **Result** tab to see the output for all of the **splice** method examples.

Starting from the end of the array

In the last example, we provide a *negative* number for the first argument. A negative number tells the splice method that we want to **"start at the end"**. For example: `foo.splice(-4, 3)` tells that **splice** method that we want to start at the **fourth-to-last** element in the array, and remove three elements.

Summary

Working with the beginning or the end of a JavaScript array is fairly straightforward, and to make matters even better, the Array.prototype's **push()**, **pop()**, **shift()** and **unshift()** methods simplify the process. It's when you want to remove or add elements to the middle of an array that things can get a bit more complex. Fortunately, though, the **Array.prototype.splice()** method provides a way to remove one or more elements from or add elements to the middle of a JavaScript array. But the key thing to remember is: this method always returns an array. So, if you are removing elements and you want to access any of the removed elements, you'll need to iterate the returned array. But if you are adding elements only, then an empty array will be returned.