JavaScript Array Management with Push(), Pop(), Shift() and Unshi

When you need to work with the first or last elements in a JavaScript array, the push(), pop(), shift() and unshift() methods are usually the best way to go.

Programmers who are new to JavaScript or come to it via languages such as PHP may find arrays a bit limiting. The main issue is usually the fact that in JavaScript, there are no associative arrays. While that may seem frustrating, associative array-like functionality can be achieved by leveraging the power and simplicity of objects. Although JavaScript arrays are restricted to numeric-based property names, they are otherwise quite flexible.

Since JavaScript array elements must have numeric property names, they are essentially anonymous. From a programmatic standpoint, the only array elements that you absolutely know anything about are the first and last elements in the array. And when an array has only one element, the first and last elements are one in the same.

The JavaScript **Array** object provides four very useful methods: **push()**, **pop()**, **shift()** and **unshift()**. To be precise, these methods belong to the Array object's **prototype** property. We know this because every array you create shares a copy of these four methods (that is because every array you create is an instance of the Array constructor).

These four methods allow us to programmatically work with the only two elements of an array that we can be sure about: the *beginning* and the *end*. Every element between the first and last is a mystery to us because, from a purely programmatic standpoint, we have no idea how many there are or what they are. We only know that as long as an array has at least one element, it has a first and last element.

The JavaScript Array.push() Method

The JavaScript **Array.push()** method adds a new element to the **end** of the array. When doing so, the array's length property increases by one. After adding the new element to the end of the array, **this method returns the new length of the array**.

Example # 1

```
var days = ['mon','tues','wed'];

console.dir(days);

console.log( days.push('thurs') ); //4

console.dir(days);
```

In Example # 1, our array starts out with three elements. The first **console.dir()** call allows you to inspect it in your console and see that it contains only: '**mon**','**tues**','**wed**'. We then use the Array object's **push()** method to add a new element to the *end* of the array ("thurs"). Notice that when we do this, we wrap the call in a **console.log()** call, allowing us to see that the **push()** method has returned the new length of the array: 4. Then, another **console.dir()** call allows us to inspect the array, demonstrating that "thurs" was, in fact, added to the end of the array.

Here is the JsFiddle.net link for Example # 1: http://jsfiddle.net/4EHkp/

The JavaScript Array.pop() Method

The JavaScript **Array.pop()** method removes the *last* element from the end of the array. When doing so, the array's length property decreases by one. After removing the last element from the end of the array, **this method returns the array element that was removed**.

Example # 2

```
var days = ['mon','tues','wed'];

console.dir(days);

console.log( days.pop() ); //wed

console.dir(days);
```

In Example # 2, our array starts out with three elements. We then use the Array object's **pop()** method to remove the last element from the end of the array ("wed"). Notice that when we do this, we wrap the call in a **console.log()** call, allowing us to see that the **pop()** method has returned the element that was removed from the end, which in this case is "wed". Then, another **console.dir()** call allows us to inspect the array, demonstrating that "wed" was in-fact removed from the end of the array.

Here is the JsFiddle.net link for Example # 2: http://jsfiddle.net/VpJmu/

The JavaScript Array.shift() Method

The JavaScript **Array.shift()** method removes the *first* element from the beginning of the array. When doing so, the array's length property decreases by one. After removing the first element from the beginning of the array, **this method returns the array element that was removed.**

Example # 3

```
var days = ['mon','tues','wed'];

console.dir(days);

console.log( days.shift() ); //mon

console.dir(days);
```

In Example # 3, our array starts out with three elements. We then use the Array object's **shift()** method to remove the *first* element from the beginning of the array ("mon"). Notice that when we do this, we wrap the call in a **console.log()** call, allowing us to see that the **shift()** method has returned the element that was removed from the beginning, which in this case is "mon". Then, another **console.dir()** call allows us to inspect the array, demonstrating that "mon" was, in fact, removed from the beginning of the array.

The JavaScript Array.unshift() Method

The JavaScript **Array.unshift()** method adds a new element to the *beginning* of the array. When doing so, the array's length property increases by one. After adding the new element to the beginning of the array, **this method returns the new length of the array.**

Example # 4

```
var days = ['mon','tues','wed'];

console.dir(days);

console.log( days.unshift('sun') ); //4

console.dir(days);
```

In Example # 4, our array starts out with three elements. The first **console.dir()** call allows you to inspect it in your console and see that it contains only: 'mon','tues','wed'. We then use the Array object's **unshift()** method to add a new element to the beginning of the array ("sun"). Notice that when we do this, we wrap the call in a console.log() call, allowing us to see that the unshift() method has returned the new length of the array: 4. Then, another **console.dir()** call allows us to inspect the array, demonstrating that "sun" was, in fact, added to the beginning of the array.

Summary

In this article, we learned about the JavaScript Array's **push()**, **pop()**, **shift()** and **unshift()** methods. We also learned that all JavaScript arrays can leverage these methods to work programmatically with the beginning and end of the array, regardless of how many elements are in between. Additionally, we learned that in two cases, the new length of the array is returned, and in the other two cases, the element that was removed is returned.

Helpful Links for the JavaScript Array's push(), pop(), shift() and unshift() methods

push()

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/push

http://msdn.microsoft.com/en-us/library/ie/6d0cbb1w(v=vs.94).aspx

pop()

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/pop

http://msdn.microsoft.com/en-us/library/ie/hx9fbx10(v=vs.94).aspx

shift()

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/shift

http://msdn.microsoft.com/en-us/library/ie/9e7b4w20(v=vs.94).aspx

unshift()

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/unshift

http://msdn.microsoft.com/en-us/library/ie/ezk94dwt(v=vs.94).aspx

2 Comments

JavaScript Interview Questions: Arrays | Kevin Chisholm - Blog

[…] JavaScript Array Management with Push(), Pop(), Shift() and Unshift()/a> […]

December 9, 2017 at 1:34 pm

Why is a JavaScript array's length property always one higher than the value of the last element's index? | Kevin Chisholm - Blog

[...] JavaScript Array Management with Push(), Pop(), Shift() and Unshift() [...]

January 17, 2018 at 11:20 am

Comments are closed.