Skip to main content

Using advanced JavaScript array methods

Andy Walpole / Category: General (/category/general) / Feb 22, 2012 / 864 words

Advanced is an adjective used subjectively here because you may be already using the methods below on a regular basis, but in by using the term advanced I am referring to JavaScript from versions 1.6 to 1.8 that are not available in Internet Explorer versions 8 and below.

IE 8 was launched with the proud boast of being CSS 2.1 and ECMAScript 3 compliant, which for Microsoft was quite an achievement in 2009. But this technology is looking particularly old hat now. IE9 saw a massive jump in JavaScript support and I would expect that there will be an equally big jump between IE10 and IE9. For ECMAScript 5 browser compatibility see kangax's pretty table here. (http://kangax.github.com/es5-compat-table/)

240px-JavaScript-logo.png

However IE8 is still one of the most popular browsers but being held back by only using ECMAScript 3 is no longer a necessity because of the underscore.js library (http://documentcloud.github.com/underscore/). Here you will find a full range of EMCA5 array methods which cannot be used in the legacy browsers. There may well be other JavaScript libraries that offer the same goodies but that isn't what this post is concerned with. Underscore.js nicely compliments jQuery so let them both flow some code love into your next project.

indexOf

JavaScript array methods are split into three camps: mutator, accessor and iteration.

You'll probably be already familiar with many JavaScript mutator methods, which are pop, push, reverse, shift, sort, splice and unshift; but JavaScript 1.6 introduced a new but extremely useful accessor method, indexOf.

This is already widely used as a string object method, but has now been extended to the array object and offers the equivalent to the PHP in_array function.

When Jonathan Snook wrote Testing for a Value in a JavaScript Array (http://snook.ca/archives/javascript/testing_for_a_v) in 2006 he fell back to using the in operator in the for loop to achieve the same result, but now the in operator can be thrown out and indexOf let loose:

```
var anArray = ['john','mary','george','freddy','jane']

console.log(anArray.indexOf("mary"));

// answer 1
```

forEach

The foreach loop is a feature of all the main programming languages (http://en.wikipedia.org/wiki/Foreach_loop) and now, thanks to ECMAScript 5, we can use it in JavaScript. It is one of many new iteration methods incorporated into all modern browsers.

forEach is useful for looping through array values:

```
var anArray = ['john', 'mary', 'george', 'freddy', 'jane'];
var aString;

if (typeof anArray !== "sring") {

    console.log("I am an array object");

}

// result: I am an array object

anArray.forEach(function (result) {

    aString = result;
```

```
    if (typeof result === "string") {
        aString += " is a string";
    }

    console.log(aString);

});

// Result:
// john is a string
// mary is a string
// george is a string
// freddy is a string
// jane is a string
```

filter

This allows the coder to take out values from an array like so:

```
var anArray = ['john', 'mary', 'peter', 'susan', "", 'mark', 'sarah', undefined];

var newArray = anArray.filter(function (value) {

    return !(value === 'john' || value === 'peter' || value === 'mark' || value === "" || value === undefined);

});

console.log(newArray);

// return ["mary", "susan", "sarah"]
```

Above removes the boys from the list plus empty or undefined values. Admittedly, the repeated use of OR in the example above is a little clumsy but it demonstrates the purpose of filter clearly. This is an array iteration method that would buddy up well with regex.

every

This allows you to loop through an array to find all elements that meet the declared criteria and returns true or false. This new method is particularly useful for dealing with numbers like so:

```
var anArray = [10, 21, 56, 124, 450];

anArray.every(function (value) {

    return (value > 50);

});
// returns false

anArray.every(function (value) {

    return (value > 5);

});
// returns true
```

map

In the above methods .forEach() returns nothing (it iterates through array and spits out strings), .every() returns a boolean value, while .map() returns a new array of objects. Let me explain with an example:

```
var anArray = [2, 67, 789, 223454];

var newArray = anArray.map(function (result) {

    return Math.sqrt(result) + Math.PI - Math.LOG10E * Math.LN10;

});

console.log(newArray)

// result: [3.555806215962888, 10.326945425462243, 30.23073646396607, 474.85080559533054]
```

Above used an array of digits, did some funky maths manipulation and then returned the results into a new array.

Some end notes

Hopefully this short post has opened up a number of possibilities for array iteration to you other than using the for loop.

The code in this post uses anonymous functions as callbacks but they can also be named. Also, I referenced underscore.js in the introduction but you don't even have to go as far using this script. The Mozilla developers have provided compatibility code for all the array methods which extend the Array.Prototype. As an example, cut the paste the code below and place it at the beginning of your script if you want to start using the filter method in all browsers. For other code visit their global objects page (https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Array).

```
if (!Array.prototype.filter) {

    Array.prototype.filter = function (fun /*, thisp */ ) {

        "use strict";



        if (this == null) throw new TypeError();



        var t = Object(this);

        var len = t.length >>> 0;

        if (typeof fun != "function") throw new TypeError();



        var res = [];

        var thisp = arguments[1];

        for (var i = 0; i < len; i++) {

            if (i in t) {

                var val = t[i]; // in case fun mutates this

                if (fun.call(thisp, val, i, t)) res.push(val);

            }

        }



        return res;

    };

}
```

For a further more fuller intellectual appraisement read Dmitry Soshnikov's (http://dmitrysoshnikov.com/) article, JavaScript array “extras” in detail (http://dev.opera.com/articles/view/javascript-array-extras-in-detail/), on the Opera Developers website

Scroll to top

Blog comments are not open for this article

for the article Using advanced JavaScript array methods / blog unblock (http://www.jonathanmurray.com/bulk.php?entry_id=47624)