Objects and Arrays (JavaScript)

01/18/2017 3 minutes to read Contributors

In this article

Expando Properties and Methods
Arrays as Objects
Multi-Dimensional Arrays

JavaScript objects are collections of properties and methods. A method is a function that is a member of an object. A property is a value or set of values (in the form of an array or object) that is a member of an object. JavaScript supports four kinds of objects:

- Intrinsic objects, such as `Array` and `String`.
- Objects you create.
- Host objects, such as `window` and `document`.
- ActiveX objects.

Expando Properties and Methods

All objects in JavaScript support expando properties and methods, which can be added and removed at run time. These properties and methods can have any name and can be identified by numbers. If the name of the property or method is a simple identifier, it can be written after the object name with a period, such as `myObj.name`, `myObj.age`, and `myObj.getAge` in the following code:

JavaScript Copy

```javascript
var myObj = new Object();
myObj.name = "Fred";
myObj.age = 42;

myObj.getAge =
    function () {
        return this.age;
    };

document.write(myObj.name);
document.write("<br/>");
document.write(myObj.age);
document.write("<br/>");
document.write(myObj.getAge());

// Output:
// Fred
// 42
// 42
```

If the name of the property or method is not a simple identifier or is unknown at the time you write the script, you can use an expression inside square brackets to index the property. The names of all expando properties in JavaScript are converted to strings before being added to the object.

JavaScript Copy

```
var myObj = new Object();

// Add two expando properties that cannot be written in the
// object.property syntax.
// The first contains invalid characters (spaces), so must be
// written inside square brackets.
myObj["not a valid identifier"] = "This is the property value";

// The second expando name is a number, so it also must
// be placed inside square brackets
myObj[100] = "100";
```

For information about creating an object from a definition, see Creating Objects.

Arrays as Objects

In JavaScript, objects and arrays are handled almost identically, because arrays are merely a special kind of object. Both objects and arrays can have properties and methods.

Arrays have a `length` property but objects do not. When you assign a value to an element of an array whose index is greater than its length (for example, `myArray[100] = "hello"`), the `length` property is automatically increased to the new length. Similarly, if you make the `length` property smaller, any element whose index is outside the length of the array is deleted.

JavaScript Copy

```
// An array with three elements
var myArray = new Array(3);

// Add some data
```

```
myArray[0] = "Hello";
myArray[1] = 42;
myArray[2] = new Date(2000, 1, 1);

document.write("original length is: " + myArray.length);
document.write("<br/>");
// Add some expando properties
myArray.expando = "JavaScript!";
myArray["another Expando"] = "Windows";

// This will still display 3, since the two expando properties
// don't affect the length.
document.write("new length is : " + myArray.length);

// Output:
// original length is: 3
// new length is : 3
```

Arrays provide methods to iterate over and manipulate members. The following example shows how to obtain the properties of objects stored in an array.

JavaScript Copy

```
var myArray = new Array(3);

// Add some data
for(var i = 1; i <= 3; i++) {
    myArray[i] = new Date(2000 + i, 1, 1);
}

myArray.forEach(function (item) {
    document.write(item.getFullYear());
});

// Output:
// 2001
// 2002
// 2003
```

Multi-Dimensional Arrays

JavaScript does not directly support multi-dimensional arrays, but you can get the behavior of multi-dimensional arrays by storing arrays within the elements of another array. (You can store any sort of data inside array elements, including other arrays.) For example, the following code builds a multiplication table for the numbers up to 5.

JavaScript Copy

```
// The size of the table.
var iMaxNum = 5;
// Loop counters.
var i, j;

// Set the length of the array to iMaxNum + 1.
// The first array index is zero, not 1.
var MultiplicationTable = new Array(iMaxNum + 1);

// Loop for each major number (each row in the table)
for (i = 1; i <= iMaxNum; i++)
{
    // Create the columns in the table
    MultiplicationTable[i] = new Array(iMaxNum + 1);

    // Fill the row with the results of the multiplication
    for (j = 1; j <= iMaxNum; j++)
    {
        MultiplicationTable[i][j] = i * j;
    }
}

document.write(MultiplicationTable[3][4]);
document.write("<br/>");
document.write(MultiplicationTable[5][2]);
document.write("<br/>");
document.write(MultiplicationTable[1][4]);

// Output:
// 12
// 10
// 4
```