Learn about Collection of Record in JavaScript

In the previous article, we introduced you to a couple of 2-dimensional array features - 2D Array in JavaScript, in many cases applied when we need to keep information about 1 or more lists of numbers of strings certain data, Array object will be the most commonly used and most used tool ...

TipsMake.com - In the previous article, we introduced you to a couple of 2-dimensional array features - 2D Array in JavaScript , in many cases applied when we need to keep information about 1 or more lists The number of certain data strings and Array objects will be the most commonly used and most used tools.

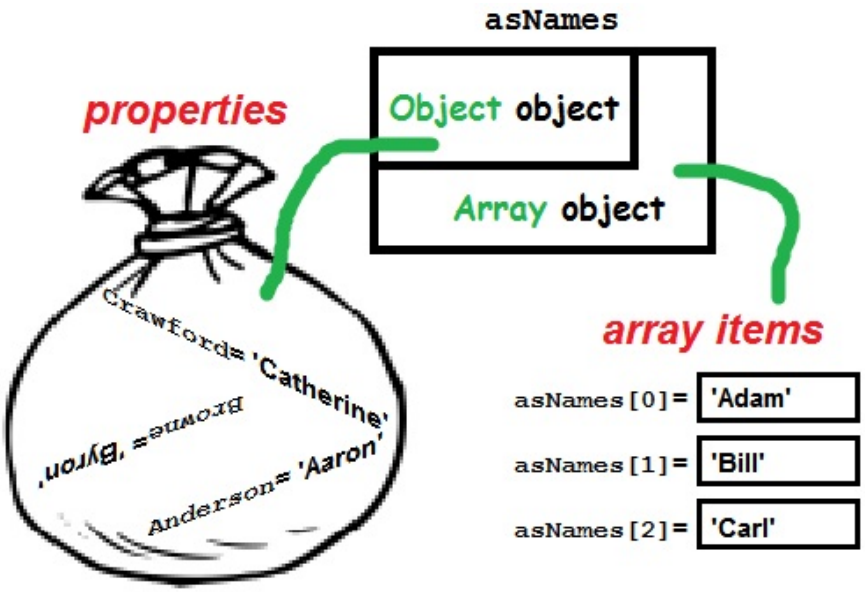
Examples are as follows:

var aNames = ["Adam", "Bill", "Carl"]; // create an array

alert (aNames [0]); // Adam - look it up by its index number

// iterate through them
cho (var j = 0; j
alert (aNames [j]); //
}

But in fact, **JavaScript** also supports **Associative Arrays** (also known as **Maps** or **Dictionaries**), providing users with another way to access the list of objects that are not located, but are interlocks. link:



However, this support feature does not work in the same way as we did before:

```
// this seems reasonable .

var aNames = new Array ();
aNames ["Anderson"] = "Adam";
aNames ["Browne"] = "Bill";
aNames ["Crawford"] = "Carl";

alert (aNames ["Anderson"]); // shows Adam (as expected)

// . but có th? không có h?p l?:

alert (aNames.length); // shows 0 !!! (why not 3?)
```

The code above has the function of creating an Array object, then inserting nothing data into any component in this array. And the next process will proceed later: all JavaScript variables (including Array) will be treated as objects with the basic properties of a typical Object. These properties are built upon the class blocks that support the ability to process and monitor the value of the link array. Examples are as follows:

```
var mapNames = new Object (); // or: var mapNames = {};
mapNames ["Anderson"] = "Adam";
mapNames ["Browne"] = "Bill";
mapNames.Crawford = "Carl"; // alternate syntax
mapNames [2] = "Two";

alert (mapNames.length); // undefined (nó không ph?i là m?t array)

for (var j in mapNames) {// show the keys
alert (j); // Anderson, Browne, Crawford, 2
}
cho (var j trong mapNames) {// hi?n th? d? li?u t??ng ?ng v?i m?i key
alert (mapNames [j]); // Adam, Bill, Carl, Two
}
```

The pair of key / value values ??is frequently called in the properties - **Properties** of an object. And the Object object supports the option to allow users to set, initialize key and value at the same time:

```
var mapNames = {
"Anderson": "Adam", // syntax is key: value
"Browne": "Bill",
"Crawford": "Carl"
};
cho (var n trong mapNames) {// show the keys and values
alert (n + "=" + mapNames [n]); // Anderson = Adam, Browne = Bill, etc .
}
```

Key value on the left and value on the right. This value is not necessarily a string or any corresponding object, just need to have the key / value attribute available. Some information or reference examples, you can read more

here.

Assign objects to function parameters:

This versatile function proved to be very useful in many cases, one of which is the process of transmitting data through a function call structure, similar to how parameters are passed through variable names. Examples are as follows:

.
DoThat ({color: "Red", font: "Arial"}); // unnamed object
.
function DoThat (o) {
alert (o.color); // shows: Red
alert (o.font); // shows: Arial
}

The above invocation procedure will create anonymous objects - Anonymous (combination of key: value values), and the called command syntaxes can access the value in the object via the name. Note that if the key is in the form of a string, you can omit the quotation marks when initializing the attribute value.

Create the combined Record array:

In fact, we can easily create map objects quickly and add more associated properties. However, this approach has one drawback, for example:

var o = {}; // or: var o = new Object ();
o.lastName = "Anderson";
o.age = 17;

. later that day .

o.LastName = "Smith"; // oops, uppercase L

If you miss the name of any key value, then you must assign a new attribute to this object. Although JavaScript can be combined with map objects, there is no need to declare it directly, but doing it this way will help us reduce a lot of errors.

In most practical cases, we can do this by initializing a predefined record - with the accompanying structure containing all the information and data in that object. Then, the initialized pieces of information will be assigned to the object, through which we can easily classify and manage the entire system of related attributes.

At the same time, you will probably need more ways to initialize the original record value, set the default value, fixed way . with **JavaScript** syntax as follows:

function PersonRec () {
this.sNameLast = ''; // the "this." c?n thi?t part
this.sNameFirst = '';

```
this.nAge = 0;
this.fIsParent = false;
this.asChildren = new Array ();
}
```

For example, we have completed the initialization of the object with certain properties. And when you have reached this step, you can think of considering any object as a record with many different data fields. For those who already have a C ++-based knowledge base, it can be difficult to use the function keyword to initialize any construction structure, but can be confusing if in the code section again contains the function or constructor variable. The example code above has no parameters, but the code below will take care of initializing the number of parameter variables, then assigning them to a separate data field:

```
function PersonRec (p1, p2, p3, p4, p5) {
this.sNameLast = (p1! = undefined)? p1: '';
this.sNameFirst = (p2! = undefined)? p2: '';
this.nAge = (p3! = undefined)? p3: -1;
this.fIsParent = (p4! = undefined)? p4: false;
this.asChildren = (p5! = undefined)? p5: new Array ();
}
```

If this is the case, missing characters or variables will be detected, and the corresponding fields will automatically be assigned the default data. To create a record, use the following syntax:

```
var rPer1 = new PersonRec (); // populated with defaults

var rPer2 = new PersonRec ("Anderson", "Adam", 35, true, ["Andy", "Al"]); // all data

var rPer3 = new PersonRec ("Browne", "Bill"); // v?i m?t d? li?u và Some defaults

rPer3.nAge = 43; // update the record
rPer3.sNameFirst = "William"; // update the record
```

The built-in function can do more than simply assign data, such as calculating the data of a school based on a number of other relevant fields, reading or forwarding information from the site, choosing data from the database . or any operation the user wants to perform whenever a Record object is created. Besides, these record distortions do not need to be stored in the usual way. As with any other object-oriented programming language, users can initiate methods, functions that are applied separately to each different data type. For example:

```
function PersonRec (p1, p2) {
this.sNameLast = p1;
this.sNameFirst = p2;
.
// ------------------------------- add some methods; ie, member functions
this.GetFullName = function () {
return (this.sNameFirst + "" + this.sNameLast);
};

this.toString = function () {
var s = "Name:" + this.GetFullName () + "n";
```

```
s + = "Age:" + this.nAge + "n";
if (this.fIsParent) {
s + = "Children:" + this.asChildren;
}
return (s);
};
}
```

We can easily see that the Object.toString () component is already available, but only displaying the output data in the [Object object] form, is not really useful in many cases, this way only matches integration in to write override functions as above, through which we can easily display data during the debug process.

Use the Record Record array:

A sample object we initialized here is often used as a component in the data array, similar to a record, and this is a very basic concept of collection. Suppose that we have a data string displayed as a drop-down menu, and when a user selects a component, the system will proceed with the process of determining the corresponding data field in the form.

In essence, **JavaScript** will perform the process of identifying data via **AJAX** , or analyzing information flow through the web, or simply from a single file. More simply, we can perform this process first in the source code section:

```
var arPersons = [
new PersonRec ("Anderson", "Adam", 35, true, ["Andy", "Alice"]),
new PersonRec ("Browne", "Bill", 25, false),
new PersonRec ("Crawford", "Carl", 45, true, ["Caroline"])
];
```

After completing the initialization of the array object of the corresponding records, we can easily 'string' the whole string with JavaScript syntax as follows:

```
var oCombo = document.getElementById ('selectName'); // a element ▼

oCombo.options.length = 0; // clear out the box
oCombo.options.length = arPersons.length; // prepare to repopulate

cho (var j = 0; j  oCombo.options [j] .text = arPersons [j] .sNameLast;
}

. later, see what's been selected .

var i = oCombo.selectedIndex;
alert (arPersons [i] .sNameFirst + "is" + arPersons [i] .nAge);
```

The display results of the data sheet will look like this:

Last Name	First Name	Age	# of Kids
Anderson	Adam	35	2
Browne	Bill	25	0
Crawford	Carl	45	1

Create the operation cycle of the elements in the array, then display the properties of the **PersonRec** object as follows:

```
function BuildOutputTable () {
sOut var = "";
sOut + = "";
sOut + = "Last Name First NameAge # of Kids"
sOut + = "";
cho (var j = 0; j  var rP = arPersons [j]; // a PersonRec object
sOut + = "";
sOut + = "" + rP.sNameLast + "";
sOut + = "" + rP.sNameFirst + "";
sOut + = "" + rP.nAge + "";
sOut + = "" + rP.asChildren.length + "";
sOut + = "";
}
sOut + = "";
return (sOut);
}
```

Some other articles related to our topic today:

- Objects as associative arrays

- Mastering Javascript Arrays

- Mastering JSON (JavaScript Object Notation)

Good luck!