APRENDERAPROGRAMAR.COM

¿ARRAYS ASOCIATIVOS JAVASCRIPT? ¿MAPS? RECORRER PROPIEDADES DE OBJETOS CON FOR IN. EJEMPLOS EJERCICIOS RESUELTOS. (CU01146E)

Sección: Cursos
Categoría: Tutorial básico del programador web: JavaScript desde cero

Fecha revisión: 2029

Resumen: Entrega nº46 del Tutorial básico "JavaScript desde cero".

Autor: César Krall

¿ARRAYS ASOCIATIVOS JAVASCRIPT?

Con frecuencia se oye que los objetos JavaScript definen arrays asociativos. ¿Existen los arrays asociativos en JavaScript de la misma forma en que se conocen en otros lenguajes? También se alude muchas veces a los objetos JavaScript como maps. Vamos a tratar de aclarar esta terminología y su significado.

PROPIEDADES ¿DEFINEN ARRAYS ASOCIATIVOS?

Hemos visto que generalmente definimos las propiedades y luego accedemos a ellas con la notación de punto como nombreObjeto.nombrePropiedad. Pero hay otra forma alternativa de acceso: la basada en usar una notación similar a la de acceso a los elementos de un array.

La sintaxis de notación tipo array para acceder a las propiedades de un objeto es la siguiente:

nombreObjeto['nombrePropiedad']

Esta sintaxis equivale a escribir nombreObjeto.nombrePropiedad

Fijarse que en el primer caso nombrePropiedad está entre comillas y en el segundo caso no, aunque también sería válido nombreObjeto[nombreVariableTipoString].

Escribe el siguiente código en tu editor y comprueba cómo el acceso a propiedades con ambas sintaxis genera los mismos resultados.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><title>Ejemplo aprenderaprogramar.com</title><meta charset="utf-8">
<script type="text/javascript">
function TaxiRenault (tipoMotor) { this.tipoMotor = tipoMotor;
this.marca = 'Renault';
this.getCapacidad = function () { if (tipoMotor == 'Diesel') { return 40;} else {return 35;} }
}
function ejemploObjetos() {
var taxi1 = new TaxiRenault();
alert ('La marca de taxi 1 como taxi1.marca es ' + taxi1.marca);
alert ('La marca de taxi 1 como taxi1.[\'marca\'] es ' + taxi1['marca']);
}
</script>
</head>
<body><div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
<div style="color:blue;" id ="pulsador" onclick="ejemploObjetos()"> Probar </div>
</body>
</html>
```

Si lo deseamos podemos crear objetos carentes de métodos con el número de propiedades que deseemos y después rescatar esas propiedades invocando el elemento con la sintaxis tipo array. La sintaxis es:

```
var nombreDelObjeto = { propiedad1: valorPropiedad1, propiedad2: valorPropiedad2,
propiedadN: valorPropiedadN };
```

Con esta sintaxis algunos caracteres o palabras reservadas no pueden usarse como nombre de propiedad sin encerrarlos entre comillas. Por ejemplo no debemos definir como propiedad +: obtenerSuma (donde obtenerSuma es el nombre de una función asociada). Si queremos que + sea una propiedad escribiremos '+': obtenerSuma. En realidad si usamos esta sintaxis todos los nombres de propiedades pueden encerrarse entre comillas porque JavaScript los considera de tipo String.

Resulta válido tanto var pintor = {'edad':32, 'nombre':'jose'} como var pintor = {edad:32, nombre:'jose'}

Con frecuencia se dice que los objetos JavaScript funcionan como Maps o que hacen un mapeo entre Strings y valores. El concepto de Map es similar al de diccionario: en un Map tenemos muchos pares (key, value) o (llave, valor). La llave es un identificador breve al que está asociado algo más complejo (el valor). Por ejemplo podríamos crear un map donde la llave es el número de pasaporte y el valor un objeto que porta todos los datos de una persona (nombre, domicilio, lugar de nacimiento, etc.). En un diccionario las claves o llaves son los términos y los valores las definiciones del término. Para encontrar una definición, buscamos la llave y a continuación leemos la definición.

En otras ocasiones se dice que los objetos JavaScript definen arrays asociativos porque permiten el acceso a las propiedades a través de la **sintaxis propia de arrays** usando como índice el nombre de la propiedad. ¿Pero realmente se genera un array al crear un objeto?

Vamos a comprobarlo. Escribe el siguiente código y ejecútalo. A continuación comentaremos las conclusiones que podemos sacar de él.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><title>Ejemplo aprenderaprogramar.com</title><meta charset="utf-8">
<script type="text/javascript">
function ejemploObjetos() {
var ejemploArray = new Array();
ejemploArray[0] = 'verde'; ejemploArray[1] = 'Argentina'; ejemploArray[2] = 'mate';
alert ('La longitud de ejemploArray es ' + ejemploArray.length);
var ejemploPropiedades = new Array();
ejemploPropiedades['color'] = 'verde'; ejemploPropiedades['pais'] = 'Argentina'; ejemploPropiedades['bebida'] = 'mate';
alert ('La longitud de ejemploPropiedades es ' + ejemploPropiedades.length);
alert ('La propiedad color vale ' + ejemploPropiedades['color']);
alert ('El índice cero del array vale ' + ejemploPropiedades[0]);
for (var i=0; i<ejemploArray.length; i++) { alert ('Valor en ejemploArray: ' + ejemploArray[i]); }
for (var i=0; i<ejemploPropiedades.length; i++) { alert ('Valor en ejemploPropiedades ' + ejemploPropiedades[i]); }
}
</script>
</head>
<body><div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
<div style="color:blue;" id ="pulsador" onclick="ejemploObjetos()"> Probar </div>
</body></html>
```

Al ejecutar este código el resultado esperado es el siguiente:

La longitud de ejemploArray es 3 -- > Para el array creado con índices numéricos
La longitud de ejemploPropiedades es 0 -- > Para el array creado añadiendo propiedades nombreArray['x'] = 'valor';
La propiedad color vale verde -- > La propiedad se puede rescatar
El índice cero del array vale undefined -- > No existe elemento índice cero ¿entonces es esto un array?
Valor en ejemploArray: verde, Valor en ejemploArray: Argentina, Valor en ejemploArray: mate

Comentarios: este código no hace lo que podría suponerse que debería hacer. Vamos a plantear y responder preguntas relativa a lo que ocurre cuando se ejecuta este código.

¿Es ejemploPropiedades un array? Sí, puesto que lo hemos creado con new Array().

¿Qué contiene el elemento con índice cero del array? No contiene nada (contiene undefined), ya que no ha sido definido, ni este ni ningún otro elementos. El array tiene cero elementos (está vacío).

¿Entonces al definir ejemploPropiedades['color'] = 'verde'; no estamos definiendo un elemento del array? No, con esa sintaxis lo que estamos haciendo es añadir una propiedad al objeto. Una propiedad no es un elemento en la colección de elementos que constituye un array, sino algo propio del objeto (y por tanto inherente al objeto dentro del cual estaría la colección de elementos). La confusión puede venir porque en otros lenguajes no existe esta sintaxis para acceder a las propiedades de un objeto y porque en otros lenguajes con arrays asociativos se permite el uso indistinto de cadenas o números para acceder a los elementos del array. Tenemos que pensar que JavaScript es JavaScript y no sigue necesariamente las convenciones o pautas habituales en otros lenguajes.

¿Por qué ejemploPropiedades.length devuelve cero? Porque no existen (al menos por el momento) elementos en el array. Por tanto no podemos recorrer este array con un bucle for porque no existen elementos en el array.

¿Cómo podría conocer o recorrer las propiedades existentes en un objeto?

Con un for in. La sintaxis a emplear para recorrer las propiedades de un objeto es la siguiente:

```
    for (nombrePropiedad in nombreObjeto) {
    ... ejecución de sentencias ...
}
```

Escribe este código y comprueba el resultado:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><title>Ejemplo aprenderaprogramar.com</title><meta charset="utf-8">
<script type="text/javascript">
function ejemploObjetos() { var ejemploPropiedades = new Array();
ejemploPropiedades['color'] = 'verde'; ejemploPropiedades['pais'] = 'Argentina'; ejemploPropiedades['bebida'] = 'mate';
msg = '';
```

```
for(nombrePropiedad in ejemploPropiedades) {
msg = msg + 'Propiedad: ' + nombrePropiedad + '. Valor propiedad es: '+ejemploPropiedades[nombrePropiedad]+ ' \n';
}
alert (msg);
}
</script>
</head>
<body><div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
<div style="color:blue;" id ="pulsador" onclick="ejemploObjetos()"> Probar </div>
</body>
</html>
```

El resultado esperado es que se muestre por pantalla:

Propiedad: color. Valor propiedad es: verde
Propiedad: pais. Valor propiedad es: Argentina
Propiedad: bebida. Valor propiedad es: mate

¿Las propiedades y forma de acceso "asociativo" existen para cualquier objeto aunque no sea de tipo Array?

Sí, el acceso basado en la sintaxis nombreObjeto.['nombrePropiedad'] existen para cualquier objeto.

¿Es correcto o no hablar de arrays asociativos en JavaScript?

La especificación oficial JavaScript no hace uso del término array asociativo y nosotros preferimos no hacer uso del término array asociativo sino referirnos a sintaxis de acceso de tipo array. No obstante, encontrarás que mucha gente hace uso de este término (algunos por desconocimiento, pero otros porque lo consideran adecuado). Grandes expertos en JavaScript hacen uso del término y otros grandes expertos indican que no se debe hacer uso del término.

Si te fijas usando el for in puedes conseguir que un objeto se comporte de forma muy parecida a como lo haría un array. Ten clara la sintaxis y las posibilidades que existen, y respecto a si usar o no el término, haz lo que consideres más oportuno.

¿Para qué es útil el acceso a propiedades con la sintaxis tipo array?

Algunos programadores la usan porque les gusta, pero hay casos en que es la única forma de acceder a una propiedad. Supón que el nombre de una propiedad lo establece el usuario (o se crea dinámicamente de alguna manera) introduciendo un dato que almacenas en una variable denominada nombrePropiedad. En este caso no es posible acceder a la propiedad con la sintaxis nombreObjeto.nombrePropiedad, pero sí será posible acceder con la sintaxis nombreObjeto[nombrePropiedad].

EJERCICIO

El siguiente código hace uso de la notación tipo array para invocar propiedades. También crea objetos únicos (los objetos plus, minus, operaciones y calcular). Analiza el código y trata de comprenderlo.

Se pide realizar los siguientes cambios:

a) Reemplaza toda la notación basada en sintaxis tipo array para el acceso a propiedades por sintaxis basada en notación de punto. Ejecuta el código y comprueba su funcionamiento.

b) Sobre el código de la opción a), cambia la definición de objetos para que no sean objetos únicos, sino que plus, minus y calcular sean funciones simples, y operaciones un objeto instanciable (que tendrás que instanciar si es necesario). Ejecuta el código y comprueba su funcionamiento.

c) Sobre el código de la opción c), añade la posibilidad de hacer cálculos de multiplicación y división de la misma forma que se hacen cálculos de suma y resta. Muestra un mensaje por cada tipo de operación. Ejecuta el código y comprueba su funcionamiento.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><title>Ejemplo aprenderaprogramar.com</title><meta charset="utf-8">
<script type="text/javascript">
var plus = function(x,y){ return x + y };
var minus = function(x,y){ return x - y };
var operaciones = {
  '+': plus,
  '-': minus
};
var calcular = function(x, y, operacion){ return operaciones[operacion](x, y); }
function ejemploObjetos() {
alert ('Resultado de calcular(3, 15, \'+\') es '+ calcular(3,15, '+'));
}
</script>
</head>
<body><div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
<div style="color:blue;" id ="pulsador" onclick="ejemploObjetos()"> Probar </div>
</body>
</html>
```

Este código nos recuerda, entre otras cosas, que: un objeto se puede crear usando la palabra clave var, una función se puede crear usando la palabra clave var, y una propiedad de un objeto puede ser una función. Esta nomenclatura es un poco “complicada”, pero hay que irse acostumbrando poco a poco a ella.

Para comprobar si tus respuestas y código son correctos puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU01147E

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=78&Itemid=206