Resumen JavaScript

Ginna Solano

¿Qué es JavaScript?

Es un lenguaje de programación orientado a objetos, es utilizado para crear eventos en páginas web.

Diferencia entre java y javaScript:

JavaScript:

Esta orientada a objetos. No se distingue entre tipos de objetos. La herencia es a través del mecanismo de prototipo, y las propiedades y métodos se puede añadir a cualquier objeto de forma dinámica. Los tipos de variables no se declaran.

Java:

Java es un lenguaje de programación basada en clases. Los objetos se dividen en clases e instancias con toda herencia a través de la jerarquía de clases. Las clases y los casos no pueden tener propiedades o métodos añadidos dinámicamente. Los tipos de variables deben ser declaradas.

Que es scripting language?

Es un lenguaje de programación que soporta scripts, es un lenguaje de control operando en alto nivel interpreta el código en vez de compilar así automáticamente ejecuta las tareas.

Que es V8?

Es un motor de javaScript utilizado en chrome.

En programación orientada a objetos:

Definiciones:

Objeto: Instancia de una clase, contiene propiedades o atributos con comportamientos los cuales reaccionan a eventos.

Clase: Propiedades y comportamientos de un objeto.

Herencia: (Por ejemplo, herencia de la clase C a la clase D) es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C.

Método: Es lo que el objeto puede hacer, puede producir cambios en las propiedades del objeto.

Atributos: Características de una clase.

Constructor: Es una función cuya misión es inicializar un objeto de una clase.

Evento: Es un suceso, la reacción que genera un objeto.

Objetos

En JavaScript, arrays son objetos, las funciones son objetos,

expresiones regulares son objetos, y, por supuesto, los objetos son objetos.

Un objeto es un contenedor de propiedades, donde una propiedad tiene un nombre y un valor. El nombre de la propiedad puede ser cualquier cadena, incluyendo la cadena vacía.

Los objetos son útiles para la recopilación y organización

datos. Los objetos pueden contener otros objetos, por lo que pueden representar fácilmente árbol o gráfica

estructuras.

JavaScript incluye una función prototipo vinculación que permite que un objeto hereda el

propiedades de otro.

Un objeto literal

Un objeto literal es un par de llaves que rodea cero o más nombre / valor

pares. Un literal objeto puede aparecer en cualquier lugar puede parecer una expresión:

```
empty_object var = {};
var títere = {
"Nombre de pila": "Jerónimo",
"Last-name": "Howard"
};
```

Retrieval (Recuperación):

Los valores se pueden recuperar de un objeto envolviendo una expresión de cadena en un [] sufijo.

```
stooge["first-name"] // "Joe"
```

Actualización

Un valor de un objeto puede ser actualizado por asignación. Si el nombre de la propiedad ya

existe en el objeto, el valor de la propiedad se sustituye:

```
stooge['first-name'] = 'Jerome';
```

Prototype

Cada objeto está vinculado a un objeto prototipo de la que puede heredar propiedades. Todos

objetos creados a partir de objetos literales están vinculados a Object.prototype

El enlace prototipo se utiliza sólo en la recuperación. Si tratamos de recuperar un valor de propiedad de

un objeto

Delete

El operador de eliminación se puede utilizar para eliminar una propiedad de un objeto. Quitará

una propiedad del objeto, si lo tiene. No va a tocar cualquiera de los objetos en el prototipo

Funciones

Una función encierra un conjunto de sentencias. Las funciones son la unidad modular fundamental

de JavaScript. Se utilizan para la reutilización de código, ocultación de información, y la composición. Las funciones se utilizan para especificar el comportamiento de los objetos

Cuando creamos una función detrás de ella están prototype,this y arguments.

Prototype: Cada objeto posee un prototype.
Puede añadir propiedades o métodos a una clase existente mediante la adición de ellos al prototipo asociado con la función constructora para esa clase.

This: Su valor hace referencia al propietario de la función que la está invocando o en su defecto, al objeto donde dicha función es un método.

Arguments: Un objeto similar a un arreglo que se corresponde con los parámetros pasados a la función.

Función literal:

Es la normal a la que se le da un nombre en la práctica esta forma de escribir nos permite hacer desarrollos separados con funciones que alteren las mismas variables dentro de un mismo objeto. básicamente, para escribir en notación literal hay que conocer dos formas de declarar los hijos y entender cómo igualar cualquier función a una variable.

Ejemplo:

```
miObjeto = {
    propiedad1 : "valor de la propiedad",
    propiedad2 : 45,
    metodoCualquiera : function (variable) {
      alert(variable);
   }
 };
 miObjeto.metodoCualquiera( miObjeto.propiedad1 );
// provocará un alert que diga "valor de la propiedad"
```

Función constructora:

Para definir un tipo de objeto, cree una función para el tipo de objeto que especifique su nombre, propiedades y métodos. Toda función de tipo constructora su nombre va capitalizada(su inicial va en mayúscula).

1. Defina el tipo de objeto escribiendo una función constructora.
2. Cree una instancia del objeto con la sentencia new.

Ejemplo:

```
function carro(fabricante, modelo, año) {
  this.fabricante = fabricante;
  this.modelo = modelo;
  this.año = año;
}
```

Nótese el uso de this para asignar valores a las propiedades del objeto.

Ahora puede crear un objeto llamado micarro como sigue:

```
micarro = new Carro("Eagle", "Talon TSi", 1993);
```

Scope

Alcance en un lenguaje de programación controla la visibilidad y tiempos de vida de las variables y parámetros. Este es un servicio importante para el programador, ya que reduce de nomenclatura

colisiones y proporciona gestión automática de memoria:

```
var foo = function () {
var a = 3, b = 5;
var bar = function () {
var b = 7, c = 11;
// En este punto, a es 3, b es 7, y c es 11
a + b + c =;
// En este punto, a es 21, b es 7, y c es 11
};
// En este punto, a es 3, b es 5, y c no se define
barra ();
// En este punto, a es 21, b es 5
```

Closure:

Los closures son funciones que manejan variables independientes. En otras palabras, la función definida en el closure "recuerda" el entorno en el que se ha creado.

Definiciones de instrucciones

Alert: Es una función que se muestra en forma de ventana al usuario para dar un aviso.

Prompt: Al igual que el alert se muestra en forma de ventana, pero permite al usuario ingresar información.

Eval(): Es utilizada para evaluar una cadena de código.

ToUppercase: Este método convierte una cadena de string a mayúsculas.

ToLowercase: Este método se utiliza para convertir una cadena de string a mayúsculas.

<noscript>: Esta etiqueta solo se mostrara al usuario cuando este no tiene soporte javaScript.

Window.onunload: El evento onunload ocurre una vez al perfil ha descargado (o la ventana del navegador se ha cerrado). Se produce cuando el usuario se desplaza fuera de la página (haciendo clic en un enlace, envía un formulario, cierre la ventana del navegador, etc.).

Undefined: Es cuando se declara la variable pero sin valor.

Null: El ***valor nulo*** es un valor simple que representa la referencia nula, vacía o inexistente.

Palabras reservadas:

Break: Esta sentencia termina una acción.

Function: Declaración de una función.

Return: Finaliza una función devolviendo un valor.

Var: Declaración de variable.

Case: Nos permite tener varias opciones o simplificar un poco el código.

If: Ejecuta una sentencia si una condición especificada es evaluada como verdadera.

Delete: Elimina una propiedad de un objeto.

Ejemplo:

```
var person = {firstName:"John", lastName:"Doe", age:50,
    eyeColor:"blue"};
    delete person.age;   // or delete person["age"];
```

Switch: Permite tener varias opciones o simplificar un poco el código.

Void: Se utiliza para tomar decisiones en función de distintos estados o valores de una variable.

Catch: Permite manejar el error. Emitiendo código.

This: This hace referencia al objeto global. No se puede establecer por la asignación durante la ejecución, y puede ser diferente cada vez que la función se llama.

While: Ejecuta una sentencia, hasta que la condición sea evaluada como verdadera.

In: El operador in retorna true si la propiedad especificada está en el objeto especificado.

Debugger: Es como colocar un punto de interrupción en el código.

For: Se utiliza para repetir una secuencia de instrucciones sobre todo cuando se conoce la cantidad exacta de veces que se quiere ejecutar cierta instrucción.

New: Constructor de un objeto.

Typeoff: Devuelve una cadena que identifica el tipo de datos de una expresión. ej: booleano,number,string.

Futuras palabras reservadas:

Class	Implements
Enum	Interface
Export	Package
Extends	Private
Import	Protected
Super	Public
Static	Yield

Lista de eventos

Nombre con prefijo on:

Relacionados con el ratón:

Onclick: Click sobre un elemento.

Ondblclick: Doble click sobre un elemento.

Onmousedown: Se pulsa un botón del ratón sobre un elemento.

Onmouseenter: El puntero del ratón entra en el área de un elemento.

Onmousemove: El puntero del ratón se está moviendo sobre el área de un elemento.

Onmouseup: Un botón del ratón se libera estando sobre un elemento

Contextmenu: Se pulsa el botón derecho del ratón (antes de que aparezca el menú contextual)

Onmouseleave: El puntero del ratón sale del área de un elemento.

Onmouseout: El puntero del ratón sale fuera del área del elemento o fuera de uno de sus hijos

Relacionados con el teclado

Onkeydown: El usuario tiene pulsada una tecla (para elementos de formulario y body)

Onkeypress: El usuario pulsa una tecla (momento justo en que la pulsa) (para elementos de formulario y body)

Diferencia entre: Window.onload y document ready:

Document ready se produce después de que el documento HTML se ha cargado, mientras que el onload evento se produce más tarde, cuando también se ha cargado todo el contenido (por ejemplo, imágenes).

El onload evento es un evento normal en el DOM, mientras que la ready de eventos es específica para jQuery.

El propósito de la ready de eventos es que debe ocurrir tan pronto como sea posible después de que el documento se ha cargado, por lo que el código que añade funcionalidad a los elementos de la página no tiene que esperar a que todo el contenido se cargue.

Otra diferencia entre los dos es que podemos tener más de un document.ready() función en una página web, pero sólo un onload función.

Guardar funciones en una variable.

Es posible porque en javaScript las funciones son de primer clase esto quiere decir que permite devolver funciones dentro de otra función y asignar funciones a variables. Esta variable seria de tipo función.

Objeto Date:

El objeto Date se utiliza para trabajar con fechas y horas.

Los objetos de la clase Date no tienen propiedades pero si un montón de métodos:

getDate() Devuelve el día del mes.

getDay() Devuelve el día de la semana.

getHours() Retorna la hora.

getMinutes() Devuelve los minutos.

getMonth() Devuelve el mes (atención al mes que empieza por 0).

getSeconds() Devuelve los segundos.

getTime() : Devuelve los milisegundos transcurridos entre el día 1 de enero de 1970 y la fecha correspondiente al objeto al que se le pasa el mensaje.

getYear() : Retorna el año, al que se le ha restado 1900. Por ejemplo, para el 1995 retorna 95, para el 2005 retorna 105. Este método está obsoleto en Netscape a partir de la versión 1.3 de Javascript y ahora se utiliza

getFullYear() : Retorna el año con todos los dígitos. Usar este método para estar seguros de que funcionará todo bien en fechas posteriores al año 2000.

Hay 4 formas de crear instancias de una fecha:

Ejemplo:

var d = new Date();
var d = new Date(milliseconds);
var d = new Date(dateString);
var d= new Date(year , month , day , hours , minutes , seconds , milliseconds)

Métodos

hasOwnProperty() ;

Método devuelve un booleano que indica si el objeto tiene la propiedad especificada.

```
o = new Object();
o.prop = 'exists';

function changeO() {
o.newprop = o.prop;
delete o.prop;
}
o.hasOwnProperty('prop');  // returns true
changeO();
o.hasOwnProperty('prop');  // returns false
```

toFixed:

Mantiene un número con los decimales que pongamos entre los paréntesis a su vez los redondea.

Ejemplo:

```
var num = 35.678;
num.toFixed(2)
alert(num);
```

Esto nos mostraría solo 2 decimales:

3.68

toPrecision()

Muestra un número en una longitud especificada.

Ejemplo:

```
var num = 13.3714;
num.toPrecision(5);
alert(num);
```

Esto nos mostraría:
13.371

El charCodeAt()

Método devuelve el valor numérico Unicode del carácter en el índice dado (a excepción de los puntos de código Unicode> 0x10000).

charCodeAt() devuelve NaN si el índice dado no es mayor que 0 o es mayor que la longitud de la cadena.

toString();

Convierte un número en una cadena.

```
var num = 15;
var n = num.toString();
```

El resultado de n sera:

15

Number();

Esta función convierte un objeto a un número que representa el valor del objeto.

Ejemplo:

```
var x1 = true;
var x2 = false;
var x3 = new Date();
var x4 = "999";
var x5 = "999 888";
 var n =
Number(x1) + "<br>" +
Number(x2) + "<br>" +
Number(x3) + "<br>" +
Number(x4) + "<br>" +
Number(x5);
```

El resultado será:

1,0,1382704503079,999,NaN

Number.toExponential(fractionDigits**);**

FractionDigits parámetro opcional controla el número de decimales.

Convierte un número en una notación exponencial.

```
var num = 5.56789;
var n = num.toExponential()
```

El resultado de *n* será:

5.56789e+0

Number.toFixed(fractionDigits);

El método toFixed convierte este número en una cadena en forma decimal. El opcional **fractionDigits** parámetro controla el número de decimales. Debe estar entre 0 y 20. El valor predeterminado es 0:

```
document.writeln (Math.PI.toFixed (0));
document.writeln (Math.PI.toFixed (2));
document.writeln (Math.PI.toFixed (7));
document.writeln (Math.PI.toFixed (16));
document.writeln (Math.PI.toFixed ());
// Produce:
3
3.14
3.1415927
3,1415926535897930
```

Number.toString (base);

El método toString convierte este número en una cadena. parámetro opcional radix controla radix, o base. Debe estar entre 2 y 36. Los radix defecto es base 10. El parámetro radix se utiliza más comúnmente con números enteros, pero se puede utilizar en cualquier número.
El caso más común, Number.toString (), se puede escribir más simplemente como String (número):

Convert a number to a string, using different bases

```
var num = 15;

var a = num.toString();

var b = num.toString(2);
var c = num.toString(8);
var d = num.toString(16);
```

The result of *a*,*b*,*c*, and *d* will be: 15,1111,17

Number.toPrecision (precisión):

El método toPrecision convierte este número en una cadena en forma decimal. El opcional parámetro de precisión controla el número de dígitos de precisión. Debe estar entre 1 y 21:

```
document.writeln (Math.PI.toPrecision (2));
document.writeln (Math.PI.toPrecision (7));
document.writeln (Math.PI.toPrecision (16));

// Produce
3.1
3.141593
3.141592653589793
```

Manejo de errores:

try : Código a intentar ejecutar .
catch: Bloque de código que se ejecuta si falla lo que esta dentro de try.
Finally: Se ejecuta sin importar si hay errores o no.

```
try {
  // sentencias a ejecutar que pueden generar excepciones
}
catch (e) {
  // De generarse alguna excepcion este es capturado
  // y puede ser accederse a traves de la variable e.
}
finally {
  // codigo que se ejecuta independientemente de si se
  // ejecuta el try o el catch o ambos.
}
```

throw: Lanza una excepción definida por el usuario.

Contiene una serie de características, sintaxis:

```
throw {
    name: Nombre del error o excepción.
    message: Descripción del error
  }
```

Métodos relacionados con:

Arrays

Arrays.push

El método push () añade nuevos elementos al final de una matriz y devuelve la nueva longitud.

Ejemplo:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.push("Kiwi");
```

El resultado de *las frutas* será: Banana,Orange,Apple, Mango,Kiwi

Array.pop

Elimina y devuelve el último elemento de este vector. Si la matriz está vacía, devuelve indefinido.

Ejemplo:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.pop();
```

El resultado de *las frutas* será:

Banana,Orange,Apple

Array.shift ():

Elimina el primer elemento de una matriz y devuelve dicho elemento .Si la matriz es vacía, devuelve undefined. desplazamiento suele ser mucho más lento que el pop:

Ejemplo:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.shift();
```

El resultado de *las frutas* será:

Orange,Apple,Mango

Array.reverse ():

Invertir el orden de los elementos de una matriz:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.reverse();
```

El resultado de *las frutas* será:

Mango,Apple,Orange,Banana

Array.slice(start,end):

El método slice () devuelve los elementos seleccionados de una matriz, como un nuevo objeto de matriz, selecciona los elementos a partir de la discusión *inicial* dado, y end determinado, *pero no incluye,* el argumento *final* dado.

Si inicio es mayor que o igual a la matriz. longitud, el resultado será una nueva matriz vacía.

Ejemplo:

```
var fruits = ["Banana", "Orange", "Lemon", "Apple",
"Mango"];
var citrus = fruits.slice(1, 3);
```

El resultado de *cítricos* será: Orange,Lemon

Array.sort():

Este método ordena los elementos de una matriz.

El orden puede ser alfabético o numérico, y, o bien ascendente (hacia arriba) o descendente (hacia abajo).

Por defecto, el método sort () ordena los valores como cadenas en orden alfabético y ascendente.

Esto funciona bien para cuerdas ("Apple" viene antes de "Banana"). Sin embargo, si los números están ordenados como cadenas, "25" es más grande que "100", porque "2" es mayor que "1".

Debido a esto, el método sort () producirá un resultado incorrecto al ordenar números.

Array.splice();

El método splice () añade / quita elementos a / de una matriz y devuelve el elemento eliminado (s).Nota: Este método cambia la matriz original.

Ejemplo

En la posición 2, añadir los nuevos elementos, y quite 1 artículo:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(2, 1, "Lemon", "Kiwi");
```

El resultado de *las frutas* será:

Banana,Orange,Lemon,Kiwi,Mango

Array unshift() ;

El método unshift () añade nuevos elementos al comienzo de una matriz y devuelve la nueva longitud.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.unshift("Lemon","Pineapple");
```

El resultado de *las frutas* será:

Lemon,Pineapple,Banana,Orange,Apple,Mango

Métodos relacionados con:

String:

regexp.exec(string);

El **exec()** método ejecuta una búsqueda de una coincidencia en una cadena especificada.Devuelve un array de resultados, o null .

string.charAt(pos):

El método charAt () devuelve el carácter en el índice especificado en una cadena.

El índice del primer carácter es 0, el segundo carácter es 1, y así sucesivamente.

Ejemplo:

Devuelva el primer carácter de una cadena:

```
var str = "HELLO WORLD";
var res = str.charAt(0)
```

El resultado de *res* será:

H

string.concat(string…):

El método concat () se utiliza para unir dos o más cadenas.

Este método no cambia las cadenas existentes, pero devuelve una nueva cadena que contiene el texto de las cadenas unidas.

Ejemplo

Únete dos cadenas:

```
var str1 = "Hello ";
var str2 = "world!";
var res = str1.concat(str2);
```

El resultado de *res* será:

Hello world!

string.lastIndexOf(searchString, position);

Ejemplos:

El método indexOf busca una searchstring dentro de una cadena. Si se encuentra, devuelve

la posición del primer carácter emparejado; de lo contrario, devuelve -1. La posición opcional

parámetro hace que la búsqueda para comenzar en alguna posición especificada en la cadena:

Buscar una cadena de "bienvenida":

```
var str = "Hello world, welcome to the universe.";
var n = str.indexOf("welcome");
```

El resultado de *n* será:

string.lastIndexOf (searchstring, posición)

El método lastIndexOf es como el método indexOf, excepto que busca desde el final de

La cadena en lugar de la parte delantera:

```
String var = 'Mississippi';
var p = text.lastIndexOf ('ss'); // P es 5
p = text.lastIndexOf ('ss', 3); // P es 2
p = text.lastIndexOf ('ss', 6); // P es 5
```

string.localeCompare(that):

El método localeCompare () compara dos cadenas en la localización actual. La configuración regional se basa en la configuración de idioma del navegador.

Devuelve un número que indica si la *cadena* viene antes, después o es igual que el *CompareString* en orden de clasificación.

. Si esta cadena es menor que la cadena, el resultado es negativo. Si

son iguales, el resultado es cero.

Ejemplo: Compara dos cadenas en la localización actual:

```
var str1 = "ab";
var str2 = "cd";
var n = str1.localeCompare(str2);
```

El resultado de *n* será:

-1 // str1 is sorted before str2

string.match(regexp):

El método match():

Busca en una cadena para un partido contra una expresión regular y devuelve los partidos, como un objeto Array.

El método partido coincide con una cadena y una expresión regular. Cómo lo hace esto depende la bandera g. Si no hay ningún indicador g, entonces el resultado de la llamada String. match (regexp) es el mismo que regexp.exec llamando al (cadena).

string.replace(searchValue, replaceValue):

El método replace () busca en una cadena para un valor especificado, o una *expresión regular* y devuelve una nueva cadena en la que se sustituyen los valores especificados.

Ejemplo

Devuelve una cadena en la que se sustituye "Microsoft" con "W3Schools":

```
var str = "Visit Microsoft!";
var res = str.replace("Microsoft", "W3Schools");
```

El resultado de *res* será:

Visit W3Schools!

string.search (regexp):

El método de búsqueda () busca en una cadena para un valor especificado, y devuelve la posición del partido.

El valor de búsqueda puede ser de cadena o una expresión regular.

Este método devuelve -1 si no se encuentra ninguna coincidencia.

Ejemplo

Busca "W3Schools":

```
var str = "Visit W3Schools!";
var n = str.search("W3Schools");
```

El resultado de *n* será:

6

string.slice(start,end):

El método slice () extrae partes de una cadena y devuelve las partes extraídas en una nueva cadena.

Utilice los parámetros de inicio y finalización para especificar la parte de la cadena que desea extraer.

El primer carácter tiene la posición 0, el segundo tiene la posición 1, y así sucesivamente.

Ejemplo: Extraer partes de una cadena:

```
var str = "Hello world!";
var res = str.slice(1,5);
```

El resultado de *res* será:

ello

string.split(separator, limit):

El método split () se utiliza para dividir una cadena en una matriz de subcadenas, y devuelve la nueva matriz.

Consejo: Si una cadena vacía ("") se utiliza como separador, la cadena se divide entre cada personaje.

Nota: El método split () no modifica la cadena original.

Ejemplo

Divide una cadena en una matriz de subcadenas:

```
var str = "How are you doing today?";
var res = str.split(" ");
```

El resultado de *res* será un array con los valores:

How,are,you,doing,today?

string.substring(start,end):

El método substring () extrae los caracteres de una cadena, entre dos índices especificados, y devuelve la nueva cadena de sub. Este método extrae los caracteres de una cadena entre "start" y "end", sin incluir el "fin" en sí.

Si "start" es mayor que "fin", este método se intercambiarán los dos argumentos, lo que significa str.substring (1,4) == str.substring (4,1).

Si bien en "Inicio" o "parada" es inferior a 0, se trata como si se tratara de 0.

Nota: El método substring () no modifica la cadena original.

Ejemplo: Extrae caracteres de una cadena:

```
var str = "Hello world!";
var res = str.substring(1, 4);
```

El resultado de *res* será:ell

String.fromCharCode(char…):

El método fromCharCode () convierte valores Unicode en caracteres.

Nota: Este es un método estático del objeto String, y la sintaxis es siempre String.fromCharCode ().

Ejemplo:

Convertir un número Unicode en un personaje:

var res = String.fromCharCode(65);

El resultado de *res* será:

A

JSON

JSON tiene seis tipos de valores : objetos, matrices , cadenas , números, booleanos (verdadero y falso), y el nulo valor especial . Espacios en blanco (espacios , tabuladores , retornos de carro y caracteres de nueva línea) pueden insertarse antes o después de cualquier valor . Esto puede hacer que los textos JSON más fácil para las personas a leer. El espacio en blanco se puede omitir para reducir los costos de transmisión o de almacenamiento.

Un objeto JSON es un contenedor desordenado de pares nombre / valor. Un nombre puede ser cualquier cadena . Un valor puede ser cualquier valor JSON , incluyendo arrays y objetos. Objetos JSON se pueden anidar a cualquier profundidad , pero en general es más eficaz para mantenerlos relativamente plana. La mayoría de los idiomas tienen una característica que los mapas fácilmente a los objetos JSON , como un objeto, estructura , ficha , diccionario, tabla hash , lista de propiedades, o matriz asociativa

Ejemplo de sintaxis

```
[
  {
"first": "Jerome",
"middle": "Lester",
"last": "Howard",
"nick-name": "Curly",
"born": 1903,
"died": 1952,
"quote": "nyuk-nyuk-nyuk!"
}
]
```