JavaScript Arrays

Los arrays JavaScript se utilizan para almacenar varios valores en una sola variable.

Ejemplo

```
var cars = ["Saab", "Volvo", "BMW"];
```

Inténtalo tú mismo "

¿Qué es un Array?

Una matriz es una variable especial, que puede contener más de un valor a la vez.

Si tiene una lista de elementos (una lista de nombres de automóviles, por ejemplo), el almacenamiento de los coches en variables individuales podría tener el siguiente aspecto:

```
var car1 = "Saab";
var car2 = "Volvo";
var car3 = "BMW";
```

Sin embargo, ¿qué pasa si desea recorrer los coches y encontrar uno específico? ¿Y si no tuvieras 3 coches, sino 300?

La solución es una matriz!

Una matriz puede contener muchos valores bajo un solo nombre y puede acceder a los valores haciendo referencia a un número de índice.

Creación de una matriz

El uso de una matriz literal es la forma más fácil de crear una matriz de JavaScript.

Sintaxis:

```
var array_name = [item1, item2, ...];
```

Ejemplo

```
var cars = ["Saab", "Volvo", "BMW"];
```

Inténtalo tú mismo "

Los espacios y los saltos de línea no son importantes. Una declaración puede abarcar varias líneas:

Ejemplo

```
var cars = [
  "Saab",
  "Volvo",
  "BMW"
];
```

Inténtalo tú mismo "

Poner una coma después del último elemento (como "BMW",) es inconsistente entre los navegadores.

IE 8 y anteriores fallarán.

Uso de la palabra clave JavaScript new

El siguiente ejemplo también crea una matriz y le asigna valores:

Ejemplo

```
var   cars = new Array("Saab", "Volvo", "BMW");
```

Inténtalo tú mismo "

Los dos ejemplos anteriores hacen exactamente lo mismo. No es necesario usar una nueva matriz ().
Para mayor simplicidad, legibilidad y velocidad de ejecución, use el primero (el método literal de la matriz).

Acceda a los elementos de una matriz

Usted se refiere a un elemento de matriz haciendo referencia al **número de índice** .

Esta declaración accede al valor del primer elemento en los automóviles:

```
var   name = cars[0];
```

Esta declaración modifica el primer elemento en los automóviles:

cars[0]= “Opel”;

Ejemplo

```
var   cars = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML = cars[0];
```

Inténtalo tú mismo "

[0] es el primer elemento en una matriz. [1] es el segundo. Los índices de matriz comienzan con 0.

Accede a la matriz completa

Con JavaScript, se puede acceder a la matriz completa consultando el nombre de la matriz:

Ejemplo

```
var cars = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML = cars;
```

Inténtalo tú mismo "

Las matrices son objetos

Las matrices son un tipo especial de objetos. El operador typeof en JavaScript devuelve el "objeto" para las matrices.

Pero, las matrices JavaScript se describen mejor como matrices.

Las matrices usan números para acceder a sus "elementos". En este ejemplo, la persona [0] devuelve a John:

Formación:

```
var person = ["John", "Doe", 46];
```

Inténtalo tú mismo "

Los objetos usan nombres para acceder a sus "miembros". En este ejemplo, person.firstName devuelve a John:

Objeto:

```
var person = {firstName:"John", lastName:"Doe", age:46};
```

Inténtalo tú mismo "

Los elementos de matriz pueden ser objetos

Las variables de JavaScript pueden ser objetos. Las matrices son tipos especiales de objetos.

Debido a esto, puede tener variables de diferentes tipos en la misma matriz.

Puede tener objetos en una matriz. Puede tener funciones en una matriz. Puede tener matrices en una matriz:

```
myArray[0] = Date.now;
myArray[1] = myFunction;
myArray[2] = myCars;
```

Adición de elementos a una matriz

La forma más fácil de agregar un nuevo elemento a una matriz es usar el método push:

Ejemplo

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.push("Lemon");                 // adds a new element (Lemon) to fruits
```

Inténtalo tú mismo "

También se puede agregar un elemento nuevo a una matriz utilizando la propiedad length:

Ejemplo

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[fruits.length] = "Lemon";     // adds a new element (Lemon) to fruits
```

Inténtalo tú mismo "

ADVERTENCIA !

Agregar elementos con índices altos puede crear "agujeros" indefinidos en una matriz:

Ejemplo

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[6] = "Lemon";                  // adds a new element (Lemon) to fruits
```

Inténtalo tú mismo "

Arrays asociativos

Muchos lenguajes de programación admiten arrays con índices nombrados.

Las matrices con índices nombrados se denominan matrices asociativas (o hashes).

JavaScript no admite matrices con índices con nombre.

En JavaScript, las matrices siempre usan índices numerados .

Ejemplo

```
var   person = [];
person[0] = "John";
person[1] = "Doe";
person[2] = 46;
var x = person.length;         // person.length will return 3
var y = person[0];             // person[0] will return "John"
```

Inténtalo tú mismo "

ADVERTENCIA !!
Si usa índices con nombre, JavaScript redefinirá la matriz a un objeto estándar.
Después de eso, algunos métodos y propiedades de array
producirán resultados incorrectos .

Ejemplo:

```
var   person = [];
person["firstName"] = "John";
person["lastName"] = "Doe";
person["age"] = 46;
var x = person.length;         // person.length will return 0
var y = person[0];             // person[0] will return undefined
```

Inténtalo tú mismo "

Evitar new Array ()

No hay necesidad de usar el constructor de matriz incorporado de JavaScript nuevo Array ().

Use [] en su lugar.

Estas dos declaraciones diferentes crean una nueva matriz vacía llamada puntos:

```
var  points = new Array();         // Bad
var points = [];                   // Good
```

Estas dos declaraciones diferentes crean una nueva matriz que contiene 6 números:

```
var  points = new Array(40, 100, 1, 5, 25, 10); // Bad
var points = [40, 100, 1, 5, 25, 10];          // Good
```

Inténtalo tú mismo "

La nueva palabra clave solo complica el código. También puede producir algunos resultados inesperados:

```
var  points = new Array(40, 100);  // Creates an array with two elements
(40 and 100)
```

¿Qué pasa si elimino uno de los elementos?

```
var  points = new Array(40);       // Creates an array with 40 undefined
elements !!!!!
```

Inténtalo tú mismo "

Cómo reconocer una matriz

Una pregunta común es: ¿Cómo puedo saber si una variable es una matriz?

El problema es que el operador JavaScript typeof devuelve "objeto":

```
var   fruits = ["Banana", "Orange", "Apple", "Mango"];

typeof fruits;             // returns object
```

Inténtalo tú mismo "

El operador typeof devuelve el objeto porque una matriz de JavaScript es un objeto.

Solución 1:

Para resolver este problema, ECMAScript 5 define un nuevo método Array.isArray () :

```
Array.isArray(fruits);      // returns true
```

Inténtalo tú mismo "

¡Pruébelo con ejercicios!

Ejercicio 1 » Ejercicio 2» Ejercicio 3 » Ejercicio 4» Ejercicio 5 »

JavaScript Array Methods

La fuerza de las matrices JavaScript reside en los métodos de matrices.

Conversión de matrices a cadenas

El método JavaScript toString () convierte una matriz a una cadena de valores de matriz (separados por comas).

Ejemplo

```
var   fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.toString();
```

Resultado

```
Banana,Orange,Apple,Mango
```

Inténtalo tú mismo "

El método join () también une todos los elementos de la matriz en una cadena.

Se comporta como toString (), pero además puede especificar el separador:

Ejemplo

```
var   fruits = ["Banana", "Orange","Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.join(" * ");
```

Resultado

```
Banana * Orange * Apple * Mango
```

Inténtalo tú mismo "

Popping and Pushing

Cuando trabaja con matrices, es fácil eliminar elementos y agregar nuevos elementos.

Esto es lo que hace popping y pushing:

Sacar elementos de una matriz, o empujar objetos en una matriz.

Apareciendo

El método pop () elimina el último elemento de una matriz:

Ejemplo

```
var   fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.pop();              // Removes the last element ("Mango") from fruits
```

Inténtalo tú mismo "

El método pop () devuelve el valor que se "extrajo":

Ejemplo

```
var   fruits = ["Banana", "Orange", "Apple", "Mango"];
var x = fruits.pop();      // the value of x is "Mango"
```

Inténtalo tú mismo "

Pushing

El método push () agrega un nuevo elemento a una matriz (al final):

Ejemplo

```
var   fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.push("Kiwi");       //  Adds a new element ("Kiwi") to fruits
```

Inténtalo tú mismo "

El método push () devuelve la nueva longitud de la matriz:

Ejemplo

```
var   fruits = ["Banana", "Orange", "Apple", "Mango"];
var x = fruits.push("Kiwi");   //  the value of x is 5
```

Inténtalo tú mismo "

Shifting Elements

shift es equivalente a pop, trabajando en el primer elemento en lugar del último.

El método **shift ()** elimina el primer elemento de matriz y "desplaza" todos los demás elementos a un índice más bajo.

Ejemplo

```
var   fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.shift();          // Removes the first element "Banana" from fruits
```

Inténtalo tú mismo "

El método shift () devuelve la cadena que se "desplazó":

Ejemplo

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

Inténtalo tú mismo "

El método **unshift ()** agrega un nuevo elemento a una matriz (al principio) y "desempaqueta" elementos anteriores:

Ejemplo

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.unshift("Lemon");    // Adds a new element "Lemon" to fruits
```

Inténtalo tú mismo "

El método unshift () devuelve la nueva longitud de la matriz.

Ejemplo

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.unshift("Lemon");    // Returns 5
```

Inténtalo tú mismo "

Cambiando elementos

Se accede a los elementos de matriz utilizando su número de índice :

Los índices de matriz comienzan con 0. [0] es el primer elemento de matriz, [1] es el segundo, [2] es el tercero ...

Ejemplo

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[0] = "Kiwi";       // Changes the first element of fruits to "Kiwi"
```

Inténtalo tú mismo "

La propiedad de longitud proporciona una manera fácil de agregar un nuevo elemento a una matriz:

Ejemplo

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[fruits.length] = "Kiwi";         // Appends "Kiwi" to fruit
```

Inténtalo tú mismo "

Eliminando Elementos

Dado que las matrices de JavaScript son objetos, los elementos se pueden eliminar utilizando el operador de JavaScript eliminar :

Ejemplo

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
delete fruits[0];          // Changes the first element in fruits to undefined
```

Inténtalo tú mismo "

El uso de eliminar puede dejar agujeros no definidos en la matriz. Use pop () o shift () en su lugar.

Agregando una matriz

El método splice () se puede usar para agregar elementos nuevos a una matriz:

Ejemplo

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(2, 0, "Lemon", "Kiwi");
```

Inténtalo tú mismo "

El primer parámetro (2) define la posición donde deben agregarse nuevos elementos (empalmados).

El segundo parámetro (0) define cuántos elementos deben eliminarse .

El resto de los parámetros ("Lemon", "Kiwi") definen los nuevos elementos que se agregarán .

Uso de splice () para eliminar elementos

Con la configuración inteligente de parámetros, puede usar splice () para eliminar elementos sin dejar "agujeros" en la matriz:

Ejemplo

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(0, 1);        // Removes the first element of fruits
```

Inténtalo tú mismo "

El primer parámetro (0) define la posición donde deben agregarse nuevos elementos (empalmados).

El segundo parámetro (1) define cuántos elementos deben eliminarse .

El resto de los parámetros se omiten. No se agregarán nuevos elementos.

Conjuntos de combinación (concatenación)

El método concat () crea una nueva matriz mediante la combinación (concatenación) de matrices existentes:

Ejemplo (fusión de dos matrices)

```
var myGirls = ["Cecilie", "Lone"];
var myBoys = ["Emil", "Tobias","Linus"];
var myChildren = myGirls.concat(myBoys);     // Concatenates (joins)
myGirls and myBoys
```

Inténtalo tú mismo "

El método concat () no cambia las matrices existentes. Siempre devuelve una nueva matriz.

El método concat () puede tomar cualquier cantidad de argumentos de matriz:

Ejemplo (combinación de tres arreglos)

```
var arr1 = ["Cecilie", "Lone"];
var arr2 = ["Emil", "Tobias","Linus"];
```

```
var   arr3 = ["Robin", "Morgan"];
var myChildren = arr1.concat(arr2, arr3);     // Concatenates arr1 with arr2
and arr3
```

Inténtalo tú mismo "

El método concat () también puede tomar valores como argumentos:

Ejemplo (Combinación de una matriz con valores)

```
var   arr1 = ["Cecilie", "Lone"];
var myChildren = arr1.concat(["Emil", "Tobias","Linus"]);
```

Inténtalo tú mismo "

Rebanar una matriz

El método slice () corta un fragmento de una matriz en una nueva matriz.

Este ejemplo corta una parte de una matriz a partir del elemento de matriz 1 ("Naranja"):

Ejemplo

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

Inténtalo tú mismo "

El método slice () crea una nueva matriz. No elimina ningún elemento de la matriz de origen.

Este ejemplo corta una parte de una matriz a partir del elemento de matriz 3 ("Apple"):

Ejemplo

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

Inténtalo tú mismo "

El método slice () puede tomar dos argumentos como slice (1, 3).

Luego, el método selecciona elementos del argumento de inicio y hasta (pero no incluye) el argumento final.

Ejemplo

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

Inténtalo tú mismo "

Si se omite el argumento final, como en los primeros ejemplos, el método slice () corta el resto de la matriz.

Ejemplo

```
var   fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
var citrus = fruits.slice(2);
```

Inténtalo tú mismo "

ToString automático ()

JavaScript convierte automáticamente una matriz en una cadena separada por comas cuando se espera un valor primitivo.

Este es siempre el caso cuando intenta generar una matriz.

Estos dos ejemplos producirán el mismo resultado:

Ejemplo

```
var   fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.toString();
```

Inténtalo tú mismo "

Ejemplo

```
var   fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits;
```

Inténtalo tú mismo "

Todos los objetos JavaScript tienen un método toString ().

¡Pruébelo con ejercicios!

Ejercicio 1 » Ejercicio 2» Ejercicio 3 » Ejercicio 4»