PROGRAMACIÓN WEB EN EL ENTORNO CLIENTE

MF0491_3

Gobierno de Canarias
Servicio Canario de Empleo

FUERTEVENTURA 2000
Centro de Formación Profesional

PIEC
PLAN INTEGRAL DE EMPLEO DE CANARIAS

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL
MINISTERIO DE TRABAJO Y ECONOMÍA SOCIAL
SERVICIO PÚBLICO DE EMPLEO ESTATAL
SEPE

Realizado por Manuel Macías

Arrays

Los objetos te permiten almacenar colecciones de datos a través de nombres.

Pero a menudo necesitamos una colección ordenada, donde tenemos un 1ro, un 2do, un 3er elemento y así sucesivamente. Por ejemplo, necesitamos almacenar una lista de algo: usuarios, bienes, elementos HTML, etc.

No es conveniente usar objetos aquí, porque no proveen métodos para manejar el orden de los elementos. No podemos insertar una nueva propiedad “entre” los existentes. Los objetos no están hechos para eso.

Existe una estructura llamada Array (llamada en español array o matriz/vector) para almacenar colecciones ordenadas.

Declaración

Hay dos sintaxis para crear un array vacío:

```
let arr = new Array();
let arr = [];
```

Casi siempre se usa la segunda. Podemos suministrar elementos iniciales entre los corchetes:

```
let fruits = ["Apple", "Orange", "Plum"];
```

Los elementos del array están numerados comenzando desde cero.

Podemos obtener un elemento por su número entre corchetes:

```
let fruits = ["Apple", "Orange", "Plum"];

alert( fruits[0] ); // Apple
alert( fruits[1] ); // Orange
alert( fruits[2] ); // Plum
```

Podemos reemplazar un elemento:

fruits[2] = 'Pear'; // ahora ["Apple", "Orange", "Pear"]

…o agregar uno nuevo al array:

fruits[3] = 'Lemon'; // ahora ["Apple", "Orange", "Pear", "Lemon"]

La cuenta total de elementos en el array es su longitud length:

```
let fruits = ["Apple", "Orange", "Plum"];

alert( fruits.length ); // 3
```

También podemos usar alert para mostrar el array completo.

```
let fruits = ["Apple", "Orange", "Plum"];

alert( fruits ); // Apple,Orange,Plum
```

Un array puede almacenar elementos de cualquier tipo.

Por ejemplo:

```
// mezcla de valores
let arr = [ 'Apple', { name: 'John' }, true, function() { alert('hello'); } ];

// obtener el objeto del índice 1 y mostrar su nombre
alert( arr[1].name ); // John

// obtener la función del índice 3 y ejecutarla
arr[3](); // hello
```

Coma residual

Un array, al igual que un objeto, puede tener una coma final:

```
let fruits = [
  "Apple",
  "Orange",
  "Plum",
];
```

La “coma final” hace más simple insertar y remover items, porque todas la líneas se vuelven similares.

Obtener los últimos elementos con “at”

Digamos que queremos el último elemento de un array.

Podemos calcular explícitamente el último índice y luego acceder al elemento:

```
let fruits = ["Apple", "Orange", "Plum"];

alert( fruits[fruits.length-1] ); // Plum
```

--------- ********** ----------

```
let fruits = ["Apple", "Orange", "Plum"];

// es lo mismo que fruits[fruits.length-1]
alert( fruits.at(-1) ); // Plum
```

Métodos pop/push, shift/unshift

Una **cola** es uno de los usos más comunes de un array. En ciencias de la computación, significa una colección ordenada de elementos que soportan dos operaciones:

unshift: inserta un elemento al principio.
shift: obtiene el elemento del principio, avanzando la cola, y así el segundo elemento se vuelve primero.

Métodos pop/push.

Hay otro caso de uso para los arrays, la estructura de datos llamada **pila**. Esta soporta dos operaciones:

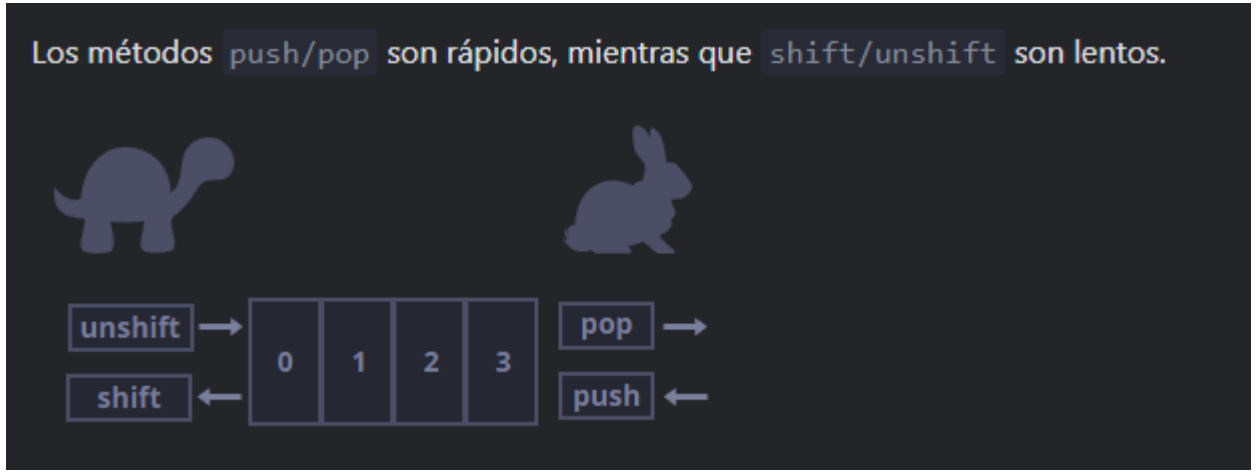
push: agrega un elemento al final.
pop: toma un elemento desde el final.

Entonces los elementos nuevos son agregados o tomados siempre desde el “final”.

Métodos pop/push

Para las pilas, la última introducida es la primera en ser recibida, en inglés esto es llamado principio LIFO (Last-In-First-Out, última en entrar primera en salir).
Para las colas, tenemos FIFO (First-In-First-Out primera en entrar, primera en salir).

Los arrays en JavaScript pueden trabajar como colas o pilas. Ellos permiten agregar o quitar elementos al o del principio o al o del final.



La operación **shift** debe hacer 3 cosas:

1. Remover el elemento con índice 0.
2. Mover todos lo elementos hacia la izquierda y renumerarlos: desde el índice 1 a 0, de 2 a 1 y así sucesivamente.
3. Actualizar la longitud: la propiedad length.

push/pop: no necesitan mover nada. Para extraer un elemento del final, el método pop limpia el índice y acorta length.

Métodos shift/unshift

Unshift

Agrega el elemento al principio del array:

```
let fruits = ["Orange", "Pear"];

fruits.unshift('Apple');

alert( fruits ); // Apple, Orange, Pear
```

Métodos shift/unshift

Unshift

Agrega el elemento al principio del array:

```
let fruits = ["Orange", "Pear"];

fruits.unshift('Apple');

alert( fruits ); // Apple, Orange, Pear
```

El método split()

El método split() divide (fragmenta) un string en dos o más sub cadenas usando un separador (divisor). El separador puede ser un solo carácter, otra cadena, o una expresión regular.

Una vez que se ha dividido la cadena en múltiples sub cadenas, el método split() sitúa las subdivisiones resultantes en una cadena que posteriormente es retornada como respuesta. Lo anterior es realizado sin modificar la cadena original.

El método split()

```
let mensaje = 'Soy un tipo feliz y afortunado';
// Dividiendo la cadena "mensaje" usando el carácter espacio
let arr = mensaje.split(' ');

// El array
console.log(arr); // ["Soy", "un", "tipo", "feliz", "y", "afortunado"]

// Acceso a cada elemento del array resultante
console.log(arr[0]); // "Soy"
console.log(arr[1]); // "un"
console.log(arr[2]); // "tipo"
console.log(arr[3]); // "feliz"
console.log(arr[4]); // "y",
console.log(arr[5]); // "afortunado"
```

El método join()

El método join () se utiliza para convertir todos los elementos de un array en una cadena.

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

var energy = fruits.join(" y ");
```

El resultado de la salida de energy:

Plátano y naranja y manzana y mango

Arrays multidimensionales

Los arrays multidimensionales son un estructuras de datos que almacenan los valores en más de una dimensión. Los arrays que hemos visto hasta ahora almacenan valores en una dimensión, por eso para acceder a las posiciones utilizamos tan solo un índice. Los arrays de 2 dimensiones guardan sus valores, por decirlo de alguna manera, en filas y columnas y por ello necesitaremos dos índices para acceder a cada una de sus posiciones.

Arrays multidimensionales

```
var temperaturas_medias_ciudad0 = new Array(3)
temperaturas_medias_ciudad0[0] = 12
temperaturas_medias_ciudad0[1] = 10
temperaturas_medias_ciudad0[2] = 11

var temperaturas_medias_ciudad1 = new Array (3)
temperaturas_medias_ciudad1[0] = 5
temperaturas_medias_ciudad1[1] = 0
temperaturas_medias_ciudad1[2] = 2

var temperaturas_medias_ciudad2 = new Array (3)
temperaturas_medias_ciudad2[0] = 10
temperaturas_medias_ciudad2[1] = 8
temperaturas_medias_ciudad2[2] = 10
```

Arrays multidimensionales

Crear el nuevo array con los tres anteriores:

```
var temperaturas_cuidades = new Array (3)
temperaturas_cuidades[0] = temperaturas_medias_ciudad0
temperaturas_cuidades[1] = temperaturas_medias_ciudad1
temperaturas_cuidades[2] = temperaturas_medias_ciudad2

var temperaturas_cuidades = new Array(new Array (12,10,11), new Array(5,0,2),new Array(10,8,10))
```

Arrays multidimensionales

Recorrer el array con for anidados

```
document.write("<table width=200 border=1 cellpadding=1 cellspacing=1>");
for (i=0;i<temperaturas_cuidades.length;i++){
	document.write("<tr>")
	document.write("<td><b>Ciudad " + i + "</b></td>")
	for (j=0;j<temperaturas_cuidades[i].length;j++){
	document.write("<td>" + temperaturas_cuidades[i][j] + "</td>")
	}
	document.write("</tr>")
}
document.write("</table>")
```

Arrays multidimensionales

Otro ejemplo:

```
var arrayMuchasDimensiones = [1, ["hola", "que", "tal", ["estas", "estamos", "estoy"], ["bien", "mal"], "acabo"], 2, 5];

alert (arrayMuchasDimensiones[0])
alert (arrayMuchasDimensiones[1][2])
alert (arrayMuchasDimensiones[1][3][1])
```

El método forEach

El método forEach de JavaScript es una de las varias formas de recorrer un array.

```
var numeros = [1, 2, 3, 4, 5];
```

Utilizando el tradicional "bucle for" para recorrer el array sería así:

```
for (i = 0; i < numeros.length; i++) {
    console.log(numeros[i]);
}
```

El método forEach

El método **forEach** también se utiliza para recorrer un array, pero utiliza una función diferente a la del clásico "bucle for".

El método **forEach** pasa una función callback para cada elemento del array junto con los siguientes parámetros:

- **Valor actual** (requerido) - El valor del elemento actual del array
- **Index** (opcional) - El número de índice del elemento actual
- **array** (opcional) - El objeto del array al que pertenece el elemento actual

El método forEach

En primer lugar, para recorrer un array utilizando el método **forEach**, se necesita una función callback (o función anónima):

```
numeros.forEach(function() {
    // código
});
```

La función se ejecutará para cada elemento del array. Debe tomar al menos un parámetro que represente los elementos del array:

```
numeros.forEach(function(numero) {
    console.log(numero);
});
```

El método forEach

Como alternativa, puede utilizar la representación de la función de flecha de ES6 para simplificar el código:

```
numeros.forEach(numero => console.log(numero));
```

El método forEach

Parámetros opcionales

Index

El primero es el parámetro "index", que representa el número de índice de cada elemento.

Básicamente, podemos ver el número de índice de un elemento si lo incluimos como segundo parámetro:

```
numeros.forEach((numero, index) => {
console.log('Indice: ' + index + ' Valor: ' + numero);
});
```

El método forEach

Parámetros opcionales

Array

El parámetro del array es el propio array. También es opcional y se puede utilizar si es necesario en varias operaciones. En caso contrario, si lo llamamos, simplemente se imprimirá tantas veces como el número de elementos del array:

```
numeros.forEach((numero, index, array) => {
    console.log(array);
});});
```

Programación web en el entorno cliente

GRACIAS

MANUEL MACÍAS

TUTORIAS@MANUELMACIAS.ES

PROGRAMACIÓN JAVASCRIPT