Clase 15

JavaScript Parte 4

Temas: Arrays - IIterar arrays a través de sus distintos métodos: for Each, every, filter, some, map, reduce - For in - For of - Strings y sus métodos **Template Strings**

Array

¿Qué es un array?

Un array es una colección o agrupación de elementos en una misma variable, cada uno de ellos ubicado por la posición que ocupa en el array. En Javascript, se pueden definir de varias formas:

Constructor	Descripción
new Array(n)	Crea un array de n elementos .
new Array(e1, e2...)	Crea un array con ninguno o varios elementos.
[e1, e2...]	Simplemente, los elementos dentro de corchetes: []. Notación preferida.

```
// Forma tradicional
const array = new Array("a", "b", "c");

// Mediante literales (preferida)
const array = ["a", "b", "c"]; // Array con 3 elementos
const empty = []; // Array vacío (0 elementos)
const mixto = ["a", 5, true]; // Array mixto (string, number, boolean)
```

Array

Propiedad	Descripción
.length	Devuelve el número de elementos del array.
[pos]	Operador que devuelve el elemento número pos del array.

Ejemplo
const array = ["a", "b", "c"];

array[0]; // 'a'
array[2]; // 'c'
array[5]; // undefined

Las posiciones empiezan a contar desde 0 y si intentamos acceder a una posición que no existe (mayor del tamaño del array), nos devolverá un undefined. https://www.w3schools.com/js/js_arrays.asp

Array

Métodos de Array

Método	Descripción
.push(obj1, obj2...)	Añade uno o varios elementos al final del array. Devuelve tamaño del array.
.pop()	Elimina y devuelve el último elemento del array.
.unshift(obj1, obj2...)	Añade uno o varios elementos al inicio del array. Devuelve tamaño del array.
.shift()	Elimina y devuelve el primer elemento del array.
.concat(obj1, obj2...)	Concatena los elementos (o elementos de los arrays) pasados por parámetro.
.indexOf(obj, from)	Devuelve la posición de la primera aparición de obj desde from.
.lastIndexOf(obj, from)	Devuelve la posición de la última aparición de obj desde from.

https://www.w3schools.com/js/js_array_methods.asp

Array

Métodos de Orden

Método	Descripción
.reverse()	Invierte el orden de elementos del array.
.sort()	Ordena los elementos del array bajo un criterio de ordenación alfabética.
.sort(func)	Ordena los elementos del array bajo un criterio de ordenación func.

Función de comparación

Como hemos visto, la ordenación que realiza sort() por defecto es siempre una ordenación alfabética. Sin embargo, podemos pasarle por parámetro lo que se conoce con los nombres de función de ordenación o función de comparación. Dicha función, lo que hace es establecer otro criterio de ordenación, en lugar del que tiene por defecto:

```
const array = [1, 8, 2, 32, 9, 7, 4];

// Función de comparación para ordenación natural
const fc = function (a, b) {
  return a -b;
};
array.sort(fc)
```

https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_sort

Array

Método	Descripción
.forEach(cb, arg)	Realiza la operación definida en cb por cada elemento del array.
.every(cb, arg)	Comprueba si todos los elementos del array cumplen la condición de cb.
.some(cb, arg)	Comprueba si al menos un elem. del array cumple la condición de cb.
.map(cb, arg)	Construye un array con lo que devuelve cb por cada elemento del array.
.filter(cb, arg)	Construye un array con los elementos que cumplen el filtro de cb.
.findIndex(cb, arg)	Devuelve la posición del elemento que cumple la condición de cb.
.find(cb, arg)	Devuelve el elemento que cumple la condición de cb.
.reduce(cb, arg)	Ejecuta cb con cada elemento (de izq a der), acumulando el resultado.
.reduceRight(cb, arg)	Idem al anterior, pero en orden de derecha a izquierda.

Array

Ejemplos

```
const arr = ["a", "b", "c", "d"];

const f = function () {
  console.log("Un elemento.");
};
arr.forEach(f);  //Un elemento. /Un elemento./ Un elemento./Un elemento

arr.forEach((e) => console.log(e)); // Devuelve 'a' / 'b' / 'c' / 'd'
arr.forEach((e, i) => console.log(e, i)); // Devuelve 'a' 0 / 'b' 1 / 'c' 2 / 'd' 3
arr.forEach((e, i, a) => console.log(a[0])); // Devuelve 'a' / 'a' / 'a' / 'a'
arr.forEach((e, i, a) => console.log(a[i])); // Devuelve 'a' / 'b' / 'c' / 'd'

const arr = ["a", "b", "c", "d"];
arr.every((e) => e.length == 1); // true    ( todos tienen longitud 1, devuelve true)
```

Array

Ejemplos

```
const arr = ["a", "bb", "c", "d"];
arr.some((e) => e.length == 2); // true    ( si al menos un elemento tiene long 2 devuelve true)

const arr = ["Ana", "Pablo", "Pedro", "Pancracio", "Heriberto"];
const nuevoArr = arr.filter((e) => e[0] == "P");
nuevoArr; // Devuelve ['Pablo', 'Pedro', 'Pancracio']

const arr = ["Ana", "Pablo",  "Pancracio", "Heriberto"];
arr.find((e) => e.length == 5); // 'Pablo'
arr.findIndex((e) => e.length == 5); // 1
```

Array

Ejemplo: reduce (acumulador)

```
const arr = [95, 5, 25, 10, 25];
arr.reduce((p, e) => {
  console.log("P="+p+" e="+e);
  return p + e;
});

// P=95 e=5     (1ª iteración: elemento 1: 95 + elemento 2: 5) = 100
// P=100 e=25   (2ª iteración: 100 + elemento 3: 25) = 125
// P=125 e=10   (3ª iteración: 125 + elemento 4: 10) = 135
// P=135 e=25   (4ª iteración: 135 + elemento 5: 25) = 160

// Finalmente, devuelve 160
```

https://www.w3schools.com/js/js_array_iteration.asp

String

¿Qué es un string?

En programación, cuando hablamos de una variable que posee información de texto, decimos que su tipo de dato es String . En Javascript, es muy sencillo crear una variable de texto, hay dos formas de hacerlo:

Constructor	Descripción
new String(s)	Crea un objeto de texto a partir del texto s pasado por parámetro.
's'	Simplemente, el texto entre comillas. Notación preferida.

Los String son tipos de datos primitivos, y como tal, es más sencillo utilizar los literales que la notación con new. Para englobar los textos, se pueden utilizar comillas simples ', comillas dobles " o backticks ` (o comilla invertida o francesa).

```
// Literales
const texto1 = "¡Hola a todos!";
const texto2 = "Otro mensaje de texto";

// Objeto
const texto1 = new String("¡Hola a todos!");
const texto2 = new String("Otro mensaje de texto");
```

JavaScript llama String a una cadena de caracteres o a un solo caracter

String

Propiedades

Propiedad	Descripción
.length	Devuelve el número de caracteres de la variable de tipo string en cuestión

Métodos

Método	Descripción
.charAt(pos)	Devuelve el carácter en la posición pos de la variable.
.concat(str1, str2...)	Devuelve el texto de la variable unido a str1, a str2...
.indexOf(str)	Devuelve la primera posición del texto str.
.indexOf(str, from)	Idem al anterior, partiendo desde la posición from.
.lastIndexOf(str, from)	Idem al anterior, pero devuelve la última posición.

String

var cad=”hola como estas”;

posición **01**2**3**456789101112**13**14

cad.charAt(0) // devuelve ‘h’

cad[1] //devuelve o

cad[20] // indefinido

cad.indexOf("a") // 3

cad.indexOf("a",4) //1 3

cad.lastIndexOf("a") //1 3

cad.lastIndexOf("a",13) //13

String

Método	Descripción
.repeat(n)	Devuelve el texto de la variable repetido n veces.
.toLowerCase()	Devuelve el texto de la variable en minúsculas.
.toUpperCase()	Devuelve el texto de la variable en mayúsculas.
.trim()	Devuelve el texto sin espacios a la izquierda y derecha.
.replace(str|, newstr)	Reemplaza la primera aparición del texto str por newstr.
.substr(ini, len)	Devuelve el subtexto desde la posición ini hasta ini+len.
.substring(ini, end)	Devuelve el subtexto desde la posición ini hasta end.
.split(sep)	Devuelve un array por el substring sep como separador

https://www.w3schools.com/js/js_string_methods.asp

String

Plantilla de cadena de caracteres (Template String)

Resumen

Las plantillas de cadena de texto (template strings) son literales de texto que habilitan el uso de expresiones incrustadas. Es posible utilizar cadenas de texto de más de una línea, y funcionalidades de interpolación de cadenas de texto con ellas.

Sintaxis

`cadena de texto`

`línea 1 de la cadena de texto
 línea 2 de la cadena de texto`

`cadena de texto ${expresión} texto`

```
function suma(a,b){
 return a+b;
}
var a=Number(prompt("ingrese un numero a:"));
var b=Number(prompt("ingrese un numero b:"));
console.log("la suma entre " + a + " y " + b + " es: " + suma(a,b));  //  la suma entre 12 y 21 es: 33
console.log(`la suma entre ${a} y ${b} es: ${suma(a,b)}  `);          //  la suma entre 12 y 21 es: 33
```