4.6 Clase en vivo - Profesores

Status	Done

Unidad 4: Arrays y Objetos

Objetivos de clase

- Arrays y métodos
- Resolución microdesafío
- Objetos

Repaso Unidad 4

Duración: 5 minutos

Consigna: Repasar lo visto en Unidad 4

Ejemplo en vivo: Arrays

Duración: 10 minutos

Consigna: Mostrar en la pantalla los siguientes ejemplos:

Declaración del array

```
// Declaraciòn de array vacío
const arrayA = [];
// Declaracion de array con nùmeros
const arrayB = [1,2];
// Declaracion de array con strings
const arrayC = ["C1","C2","C3"];
// Declaracion de array con booleanos
const arrayD = [true,false,true,false];
```

```
// Declaracion de array mixto
const arrayE = [1,false,"C4"];
```

Acceso al array

Los elementos dentro de un array tienen un índice que determina su posición.

Así, es posible acceder a los elementos dentro de un array a través de su posición:

```
const  numeros = [1,2,3,4,5];
console.log( numeros[0] ) // 1;
console.log( numeros[3] ) // 4;
let resultado  = numeros[1] + numeros[2]
console.log( resultado  ) // 5;
```

Recorrido del array

Decimos que estamos recorriendo un Array cuando empleamos un **bucle** para acceder a cada elemento por separado.

Los Array en JavaScript son **objetos iterables**, lo que permite usar distintas estructuras para iterar sobre ellos.

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

for (let index = 0; index < 5; index++) {

alert(numeros[index]);

}
```

Ejemplo en vivo: Array: Métodos y propiedades

Duración: 35 minutos

Consigna: Mostrar en la pantalla los siguientes ejemplos:

Length

Al igual que en un String, la propiedad **length** nos sirve para obtener la longitud de un Array, es decir, para identificar cuántos elementos tiene.

```
const miArray = ["marca", 3 ,"palabra"];

console.log( miArray.length ); //imprime 3

const numeros = [1, 2, 3, 4, 5, 6, 7, 8]

for (let i= 0; i < numeros.length; i++) {

alert(numeros[i]);

}
```

Agregar elementos

Para sumar un elemento a un Array ya existente, se utiliza el **método push**, pasando como parámetro el valor (o variable) a agregar.

```
const miArray = ["marca", 3, "palabra"]

miArray.push('otro elemento')

console.log(miArray.length) // ⇒ 4

console.log(miArray)
```

```
//["marca", 3, "palabra", “otro elemento”]
```

Agregar elementos al inicio del array

Utilizamos el método **unshift()** de forma similar:

```
const miArray = ["marca", 3, "palabra"]

miArray.unshift('otro elemento')

console.log(miArray)

//[“otro elemento”, "marca", 3, "palabra"]
```

Pop y Shift

Mediante el método pop() quitamos el último elemento del array. Utilizando Shift() quitamos el primer elemento del array.

```
const nombres = ["Luis", "Ana", "Julia", “Juan”]

nombres.pop()

console.log(nombres) // ["Luis", "Ana", "Julia"]

nombres.shift()

console.log(nombres) // ["Ana", "Julia"]
```

Join

Mediante el método join() podemos generar un *string* con todos los elementos del array, separados por el valor que pasamos por parámetro:

```
const nombres = ["Luis", "Ana", "Julia", “Juan”]

console.log( nombres.join(“, ”) )

// Luis, Ana, Julia

console.log( nombres.join(“*”) )

// Luis*Ana*Julia
```

Indexof

El método indexOf() nos permite obtener el índice de un elemento en un array. Recibe por parámetro el elemento que queremos buscar en el array y, en caso de existir, nos retorna su índice. Si el elemento no existe nos retornará como valor: **-1**

```
const nombres = ['Rita', 'Pedro', 'Miguel', 'Ana', 'Vanesa'];

console.log( nombres.indexOf('Rita') ) // ⇒ 0

console.log( nombres.indexOf('Ana') ) // ⇒ 3

console.log( nombres.indexOf('Julieta') ) // ⇒ -1
```

Includes

El método includes() me permite saber si el elemento que recibe por parámetro, existe o no dentro de un array, retornando un valor booleano en caso afirmativo o negativo.

Ten presente que la búsqueda a realizar debe ser exacta, y no parcial.

```
const nombres = ['Rita', 'Pedro', 'Miguel', 'Ana', 'Vanesa']

console.log( nombres.includes('Rita') ) // ⇒ true

console.log( nombres.includes('Miguel') ) // ⇒ true

console.log( nombres.includes('Julieta') ) // ⇒ false
```

Sort

El método sort() ordena un array de forma ascendente. Cuando trabajamos con un array de elementos, podemos invocarlo de forma directa para que ordene el contenido.

```
const nombres = ['Rita', 'Pedro', 'Miguel', 'Ana', 'Vanesa']

nombres.sort()

console.log( nombres )

// ⇒ ['Ana', 'Miguel', 'Pedro', 'Rita', 'Vanesa']
```

Reverse

Como su nombre lo indica, el método reverse() invierte el orden de los elementos dentro de un array.

```
const nombres = ['Rita', 'Pedro', 'Miguel', 'Ana', 'Vanesa']

nombres.reverse()

console.log( nombres )
```

```
// ⇒ ['Vanesa', 'Ana', 'Miguel', 'Pedro', 'Rita']
```

Sort y Reverse

Si deseamos ordenar de forma descendente los elementos de un array, podemos concatenar el método sort seguido de reverse, para aplicar este tipo de ordenamiento.

```
const nombres = ['Rita', 'Pedro', 'Miguel', 'Ana', 'Vanesa']

nombres.sort().reverse()

console.log( nombres )

// ⇒ ['Vanesa', 'Rita', 'Pedro', 'Miguel', 'Ana']
```

Actividad: Exploración de Propiedades y Métodos

Duración: 20 minutos

Explicación y demostración: Proporciona ejemplos prácticos de cómo utilizar cada método y cuál es su efecto en el array.

Posible solución:

```
let numeros = [3, 1, 4, 1, 5, 9];
numeros.sort(); // Ordena el array
numeros.reverse(); // Invierte el array
console.log(numeros.join("-")); // Convierte el array en stri
ng separado por "-"
console.log(numeros.indexOf(4)); // Encuentra el índice del e
lemento 4
```

```
console.log(numeros.includes(9)); // Verifica si el 9 está in
cluido en el array
```

Ejemplo en vivo: Recuperamos Objetos Literales

Duración: 5 minutos

Consigna: Presentar el siguiente ejemplo en vivo:

Los objetos literales son un tipo de dato que almacena varias claves y valores en un formato textual.

```
const productos = [{ id: 1, producto: "Arroz" },

{ id: 2,  producto: "Fideo" },

{ id: 3,  producto: "Pan" }];
```

Ejemplo en vivo: Sentencia for of

Duración: 10 minutos

Consigna: Presentar el siguiente ejemplo en vivo de como usar la sentencia for of aplicado a objetos, producto y array:

1° Ejemplo

```
class Producto {

constructor(*nombre*, *precio*) {

*this*.nombre  = *nombre*.toUpperCase();

*this*.precio  = parseFloat(*precio*);

*this*.vendido = false;
```

```
}

sumaIva() {

*this*.precio = *this*.precio * 1.21;

}

}
```

//Tenemos nuestra clase JS para instanciarla por cada nuevo producto que se agregue en el Array.

2° Ejemplo

```
const productos = [];

productos.push(**new** Producto("arroz", "125"));

productos.push(**new** Producto("fideo", "70"));

productos.push(**new** Producto("pan", "50"));

//Iteramos el array con for...of para modificarlos a todos

for (const producto of productos)

producto.sumaIva();

// Todo ciclo for como también todo condicional que resuelva su
```

Actividad: Métodos de Arrays

Duración: 10 minutos

Consigna: Resolución

- Dado el siguiente array:const frutas = ['uva', 'pera', 'banana']; ¿Qué método agregaría 'mango' al final del array?

a) frutas.unshift('mango');

b)frutas.push('mango');

c) frutas.pop('mango');

- Dado el siguiente array:
  const numeros = [70, 55, 88, 63, 97];
  ¿Qué método se usaría para encontrar el índice del valor 8 en el array?

a) numeros.indexOf(88);

b) numeros.includes(88);

c) numeros.reverse(88);

- Dado el siguiente array:const palabras = ['tomate', 'zanahoria', 'manzana', 'banana', 'naranja'];Si se desea ordenar alfabéticamente y unir los elementos con comas, ¿Qué opción es correcta?

a) palabras.reverse().join('-');

b) palabras.sort().join(', ');

c) palabras.push(', ').sort();

Actividad: Manipulación de Arrays

Duración: 20 minutos.

Explicación y demostración: Muestra cómo usar cada uno de los métodos con un ejemplo práctico.

Posible solución:

```
let frutas = ["manzana", "banana"];
frutas.push("naranja"); // Añade al final
frutas.pop(); // Elimina el último
```

```
frutas.shift(); // Elimina el primero
frutas.unshift("kiwi"); // Añade al inicio
```

Preclase - 30 minutos previos

Repaso de Conceptos Clave (10 minutos)

- **Arrays en JavaScript:** Breve repaso sobre la creación y manipulación de arrays, incluyendo métodos como `push`, `pop`, `shift`, `unshift`, `sort`, y `reverse`.
- **Acceso y Recorrido de Arrays:** Explicación de cómo acceder a elementos específicos mediante índices y cómo recorrer arrays usando bucles como `for` y `for...of`.
- **Combinación de Arrays y Objetos:** Repaso sobre cómo arrays pueden contener objetos y cómo manipular estos datos estructurados.

Espacio para Preguntas y Respuestas (15 minutos)

- Permitir que los estudiantes formulen preguntas sobre cualquier aspecto de arrays que encuentren confuso o deseen explorar más a fondo.
- Proporcionar respuestas claras y ejemplos adicionales sobre cómo implementar y utilizar diferentes métodos de arrays.
- Si no hay preguntas, iniciar una discusión sobre escenarios comunes en los que se utilizan arrays en proyectos reales, como el manejo de listas de usuarios, productos, etc.

Demostraciones Prácticas (Opcional, según las preguntas)

- Demostrar en vivo cómo se pueden resolver problemas comunes usando arrays, como la búsqueda de elementos, ordenamiento de datos, y transformaciones de arrays.
- Mostrar cómo los métodos como `map`, `filter`, y `reduce` pueden ser utilizados para operaciones más avanzadas en arrays.