DATA TYPES IN JAVASCRIPT - ARRAYS

Objetivos

- ✓ Comprender qué son los ***arrays*** y por qué son útiles
- ✓ Acceder a un elemento utilizando el índice de un array
- ✓ Agregar elementos utilizando los métodos ***unshift*** y ***push***
- ✓ Eliminar elementos utilizando los métodos ***splice***, ***shift***, ***pop***
- ✓ Iterar sobre un ***array*** con un bucle ***for***
- ✓ Iterar sobre un ***array*** con el método ***forEach***

Arrays

Son estructuras de datos que nos permiten agrupar una colección de elementos en una variable.

Syntax:
var array_name = [item1, item2, ...];

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

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

Declaración

Puede declararse vacío:

```
const arrayNames = [];
```

O con algunos elementos:

```
const arrayNames = ["Pedro", "Jake", "Joan", "Mike"];
```

Los elementos del array no tienen que ser todos del mismo tipo; podemos mezclar cadenas, números o cualquier otro tipo de dato.

```
const arrayNames = ["Pedro", 2, true];
```

Acceso a elementos

Podemos acceder a elementos individuales en el array por su posición. La posición se denomina *index* (índice).

El índice del primer elemento es 0.

```
const arrayNames = ["Pedro", "Jake", "Joan", "Mike"];
```

Entonces para imprimir los elementos del array en la consola:

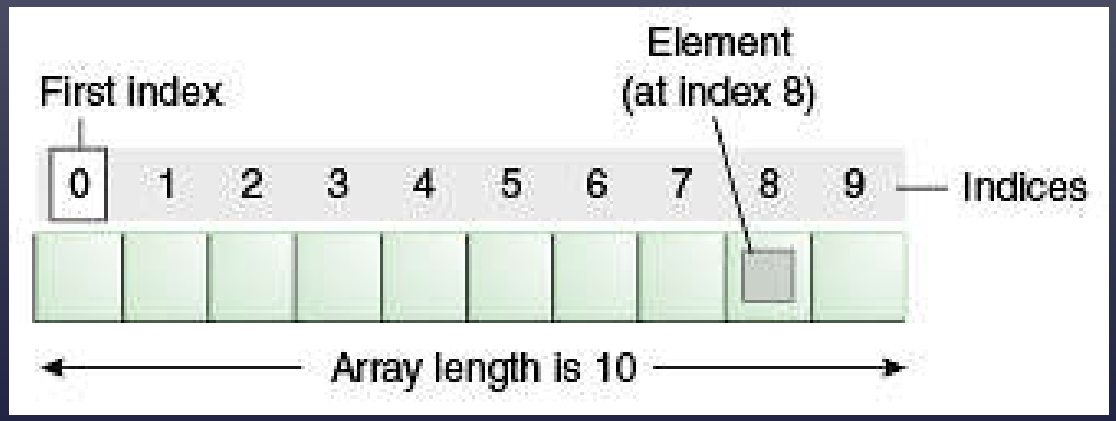
```
const arrayNames = ["Pedro", "Jake", "Joan", "Mike"];

console.log(arrayNames[0]);      // <== Pedro
console.log(arrayNames[1]); // <== Jake
console.log(arrayNames[2]); // <== Joan
console.log(arrayNames[3]); // <== Mike
console.log(arrayNames[99]); // <== undefined
```

*Si intentamos acceder a un índice que no existe, retornará **undefined***

Length of array

La longitud de un array es el número de elementos que el array está almacenando.
Entonces, si un array tiene 10 elementos, el primer índice será 0 y el último 9



Agregar elementos

Utilizamos el método **.push()**. El nuevo valor se almacenará al final del array, no al principio o en una posición aleatoria.

```
let arrayNames = ["Pedro", "Jake", "Joan"];

arrayNames.push("Rachel");

console.log(arrayNames);

// [ 'Pedro', 'Jake', 'Joan', 'Rachel' ]
```

Agregar elementos

Si deseas agregar un elemento al comienzo del array, use **unshift** en lugar de push

```
let arrayNames = ["Pedro", "Jake", "Joan"];

arrayNames.unshift("Rachel");

console.log(arrayNames);

// [ 'Rachel', 'Pedro', 'Jake', 'Joan' ]
```

Eliminar elementos de un array

Utilizamos el método ***splice***

array.splice(start, deleteCount)

Start: índice en el que comenza a cambiar el array
deleteCount: número de elementos para eliminar

```
var arrayNames = ["Pedro", "Jake", "Joan"];
arrayNames.splice(0,1);
console.log(arrayNames);
//[ 'Jake', 'Joan' ]
```

Eliminar elementos de un array

Pero hay otras formas de eliminar elementos. Por ejemplo, el método ***pop*** nos permite eliminar el último valor, mientras que ***shift*** elimina el primero.

```
var arrayNames = ["Pedro", "Jake", "Joan"];
arrayNames.pop();
console.log(arrayNames);
// [ 'Pedro', 'Jake' ]

var arrayNames2 = ["Jensen", "Mike", "Tom"];
arrayNames2.shift();
console.log(arrayNames2);
// [ 'Mike', 'Tom' ]
```

Método	Acción
Push	Agrega un elemento al final
Unshift	Agrega un elemento al principio
Shift	Elimina el primer elemento
Pop	Elimina el último elemento
Splice	Elimina elementos de cualquier parte del array

Iterando sobre cada elemento en el array

for

Supongamos que queremos sumar todos los números en un array, o queremos saludar a cada uno de los nombres en un array. Para eso iteramos.

```
const arrayNames = ["Pedro", "Jake", "Joan"];

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

Ahora no importa cuántos elementos hay en un array, este loop los imprimirá a todos.

Iterando sobre cada elemento en el array

forEach method
para iterar sobre arrays. Es similar a *for* pero más ordenado.

```
var arrayNames = ["Pedro", "Jake", "Joan"];

arrayNames.forEach(function(name){
    console.log(name);
})
```

Iterando sobre cada elemento - forEach

- Sin parámetros: solo llamará a la función tantas veces como elementos estén en el array

```
// ES5:
['a','b'].forEach(function(){
    console.log('hi!');
});

// ES6:
['a','b'].forEach(() => console.log('hi!'));

// => hi!
// => hi!
```

Iterando sobre cada elemento – forEach

- Primer parámetro: element

Si pasamos un parámetro, será igual a cada elemento en cada iteración.

```
// ES5:
[1,2,3,4].forEach(function(element){
    console.log(element*2);
});

// ES6:
[1,2,3,4].forEach(element => console.log(element*2));

// console:
// => 2
// => 4
// => 6
// => 8
```

Iterando sobre cada elemento – forEach

- Segundo parámetro: index

```
const raceResults = ['Helen', 'John', 'Peter', 'Merry'];
raceResults.forEach(function(elem, index){
    console.log(elem + ' finished the race in ' + (index+1) + ' position!');
});

// => Helen finished the race in 1 position!
// => John finished the race in 2 position!
// => Peter finished the race in 3 position!
// => Merry finished the race in 4 position!
```

Practice time

Escriba la función a continuación utilizando la sintaxis de arrow function.

¿Puedes adivinar el output?

```
function printStars(howMany){
    console.log("*".repeat(howMany));
}

[1,2,3,4,5].forEach(function(num){
    printStars(num);
})
```

.split() method

nos permite separar un string en partes y devolverá todas las partes como elementos de un nuevo array

Resumiendo

- ✓ Los arrays son estructuras de datos que nos permiten almacenar una colección de elementos, no importa el tipo. Podemos manipular los arrays, cambiar, agregar o eliminar sus elementos.

- ✓ También aprendimos forEach para iterar sobre arrays