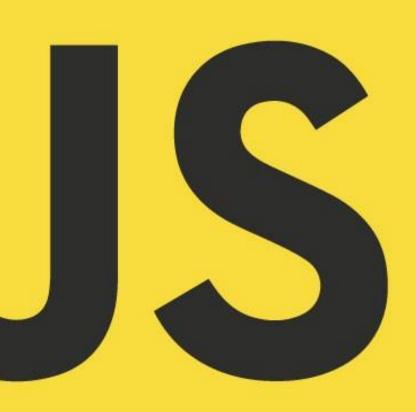
[Arrays]

UT 4_1 - Programación con colecciones, funciones y objetos definidos por el usuario: Arrays

Marina Navarro Pina





Colecciones

Las colecciones en JavaScript permiten almacenar y manipular grupos de datos.

Arrays, Maps y Sets son las principales colecciones utilizadas en JavaScript.

Arrays

En JavaScript son un tipo especial de objeto.

A diferencia de otros lenguajes de programación donde los Arrays tienen un tamaño fijo, en JavaScript *puedes añadir o eliminar elementos en cualquier momento*.

Los elementos pueden ser de cualquier tipo (number, string, boolean, etc.), incluyendo otros Arrays y objetos.

Arrays - Creación

Se recomienda crearlos usando notación JSON:

```
const a = []
const b = [2,4,6]
```

aunque también podemos crearlos como instancias del objeto Array (NO recomendado):

Arrays - Elementos

Sus elementos pueden ser **de cualquier tipo**: números, cadenas, objetos, e incluso otros Arrays. Esto permite una gran flexibilidad. Si no está definido un elemento su valor será *undefined*.

Ejemplo 1 - Creación y Acceso

Acceder a un elemento de un array que <u>no existe no provoca</u> un <u>error</u> (devuelve *undefined*), pero sí lo provoca acceder a un elemento de algo que no es un array. Con ES2020 (ES11) se incluyó el operador de encadenamiento (?.) que permite acceder a propiedades de un objeto de manera segura, evitando errores si el objeto es *null* o *undefined*:

```
console.log(alumnos?.[0])
// si alumnos es un array muestra el valor de su primer
// elemento y si no muestra undefined pero no lanza un error
```

Arrays - Elementos

Sus elementos pueden ser **de cualquier tipo**: números, cadenas, objetos, e incluso otros Arrays. Esto permite una gran flexibilidad. Si no está definido un elemento su valor será *undefined*.

Ejemplo 1

Acceder a un elemento de un array que <u>no existe no provoca</u> un <u>error</u> (devuelve *undefined*), pero sí lo provoca acceder a un elemento de algo que no es un array. Con ES2020 (ES11) se ha incluido el operador ?. para evitar tener que comprobar nosotros que sea un array:

Operaciones

Las operaciones con arrays en JavaScript son fundamentales para manipular colecciones de datos.

Más Información:

- https://www.w3schools.com/jsref/jsref_obj_array.asp
- https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/e/Global_Objects/Array

Operaciones - Métodos y Propiedades

Ejemplo 2

Principales métodos y propiedades de los arrays:

- 1. Propiedad **length**: Devuelve la **cantidad de elementos** en un array.
- 2. Añadir elementos: métodos **push** (al final) y **unshift** (al principio)
- 3. Eliminar elementos: métodos **pop** y **shift**
- 4. Método **splice**: eliminar o insertar elementos en cualquier posición
- Método slice: subarray con los elementos indicados sin modificar el array original

Operaciones - Arrays y Strings

Ejemplo 3

Cada objeto (y los arrays son un tipo de objeto) tienen definido el método .toString() que lo convierte en una cadena con los elementos separados por ','.

Para convertir los elementos de un array en una cadena con un separador específico, se utiliza el método .join()

Por otro lado, el método .split() permite convertir una cadena en un array.

Operaciones - Ordenación

Ejemplo 4

El método .sort() se utiliza para ordenar los elementos de un array. Este método modifica el array original y devuelve el array ordenado.

Para *personalizar el criterio de ordenación*, se puede pasar una **función** al método .sort(). Esta función debe devolver un valor negativo, positivo o cero, dependiendo de la comparación entre dos elementos.

Más Información:

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Array/sort

Operaciones - Otros métodos comúnes

.concat(): concatena arrays

```
let a = [2, 4, 6]
let b = ['a', 'b', 'c']
let c = a.concat(b)  // c = [2, 4, 6, 'a', 'b', 'c']
```

.reverse(): invierte el orden de los elementos del array

```
let a = [2, 4, 6]
let b = a.reverse()  // b = [6, 4, 2]
```

- indexOf(): devuelve la primera posición del elemento pasado como parámetro o -1 si no se encuentra en el array
- .lastIndexOf(): devuelve la última posición del elemento pasado como parámetro o -1 si no se encuentra en el array

Buenas prácticas

En progreso...