Arrays JS

En JavaScript, los arrays son un tipo de objeto que se utilizan para almacenar una colección de valores. Los arrays pueden almacenar cualquier tipo de valores, incluyendo números, cadenas de texto, objetos, funciones, etc.

Para crear un array, se puede utilizar la función `Array()` o las llaves `[]`. La sintaxis para crear un array con la función `Array()` es la siguiente:

```
let array = new Array();
```

La sintaxis para crear un array con llaves `[]` es la siguiente:

```
let array = [];
```

Trabajando con Arrays

La manera de representar un listado de elementos en programación, se denomina Array. Un array está compuesto por varios ítems, entre corchetes **[ ]**.

```
let listYears = [1972, 1984, 1988, 1992, 1997, 2021, 2022];
let listNames = ["Captain America", "Spiderman", "IronMan"];
let listSuperHeroes = [
  { id: 1, name: 'Tony Stark' },
  { id: 4, name: 'Peter Parker' },
];
```

Declarar y acceder a los elementos de un Array

Pensemos en el array como una lista de elementos, cada uno con su posición **(empezando por la 0)**. Para acceder a la primera posición del array, basta con poner entre corchetes el número 0.

```
let avengers = ["Hulk", "SpiderMan", "AntMan"];

let avenger = avengers[0]; // Probad a cambiar este numero ;)
console.log(avenger) // Devuelve "Hulk"
```

Métodos que modifican el array

Vamos a ver varios métodos que nos da el lenguaje y como vamos a trabajar con ellos. Primero veremos los métodos que modifican nuestro array inicial.

Vamos a declarar un array de una manera diferente, no es lo tradicional pero así vemos el funcionamiento de las posiciones.

```
let moderatIII = new Array();
            moderatIII[0] = "Eating hooks";
            moderatIII[1] = "Running";
            moderatIII[2] = "Finder";
            moderatIII[3] = "Ghostmother";
            moderatIII[4] = "Reminder";
            moderatIII[5] = "Intruder";
            moderatIII[6] = "Animal trails";

console.log(moderatIII);
```

pop() no admite parámetros. Elimina el último elemento del array.

```
moderatIII.pop();

console.log(moderatIII);
```

push() entre los paréntesis se pueden añadir tantos valores como se quiera añadir a el array, separados por comas (**,**). Añade un elemento al array en última posición.

```
moderatIII.push('Ethereal');

console.log(moderatIII);
```

reverse() tampoco admite parámetros. Invierte el orden del array.

```
moderatIII.reverse();

console.log(moderatIII);
```

shift() funciona como **pop()**, pero extrayendo del array el primer elemento. Todos los índices de la matriz se actualizan.

```
moderatIII.shift();
```

sort() ordena los elementos del array, tal y como ordena los nombres de archivo un ordenador, es decir:

- Los valores como los de esta matriz se ordenan por orden alfabético.

- Si la matriz consiste en una serie de números, estos se ordenan según las cifras más a la izquierda, de menor a mayor. Así, por ejemplo, una matriz que consista en **200,35,40,1** no se ordenará según el valor de las cifras —**1,35,40,200**—, sino que **sort()** devolverá **1,200,35,40**.

```
moderatIII.sort();

console.log(moderatIII);
```

splice() es un método especial, cuyo comportamiento depende de los parámetros que se le asignen:

- El **primer parámetro** es un **número** que representa el **índice** desde el que tiene que **empezar a modificar la matriz**.

- El **segundo** es otro **número** que indica el número de **elementos** que debe **eliminar** o **sustituir**, contando el inicial (por ello un valor de **0** no modifica nada en absoluto).

- Tras estos dos parámetros, se puede incluir una **lista de elementos** separados por comas:

- Si **no** se incluye ningún **elemento**, **splice()** simplemente **elimina** tantos elementos como se han indicado en el segundo parámetro, tomando como índice inicial el primer parámetro.

- Si se **incluyen menos elementos** que el valor del **segundo parámetro**, **rellena los «huecos»** disponibles hasta quedarse sin valores, **y elimina los que faltan**.

- Si **se incluyen más elementos** que el valor del **segundo parámetr**o, primero **elimina los indicados**, y luego **inserta todos los elementos** proporcionados entre el anterior y el posterior a la sección eliminada. Sin embargo, en este caso **splice() no devuelve la parte del array seccionada** como en los anteriores, **sino la nueva longitud**.

```
moderatIII.splice(2,2);

console.log(moderatIII);

moderatIII.splice(2,2,'Ethereal');

console.log(moderatIII);

moderatIII.splice(2,2,'Ethereal','Ethereal Remix vol.1','Ethere

console.log(moderatIII);
```

Como parámetros se puede proporcionar a **unshift()** una serie de elementos separados por comas, que se **añaden al principio de la matriz**. Los índices se actualizan en consecuencia, y el método devuelve la nueva longitud.

```
moderatIII.unshift('Reminder Remix vol.1','Reminder Remix vol.2
```

Métodos que no modifican la matriz

Estos métodos devuelven una representación del array, pero no lo modifican. Vamos a emplear los dos arrays siguientes.

```
let moderatII = new Array();
    moderatII[0] = "The Mark";
    moderatII[1] = "Bad Kingdom";
    moderatII[2] = "Versions";
    moderatII[3] = "Milk";
    moderatII[4] = "Gita";
    moderatII[5] = "Damage Done";

let moderat = new Array();
    moderat[0] = "Rusty nails";
    moderat[1] = "Seanmonkey";
    moderat[2] = "Nasty silence";
    moderat[3] = "Berlin";
```

concat() en este caso he **concatenado la segunda matriz a la primera**, pero se pueden encadenar una serie de elementos separados por comas, o incluso varias matrices.

```
moderatII.concat(moderat);

console.log(moderatII.concat(moderat));

console.log(moderatII);
```

```
console.log(moderat);
```

Como se puede ver, sin embargo, la matriz **moderat** no ha sido alterada.

Empleado sin un parámetro, **join()** devuelve una mera cadena en la que los valores del array están separados por comas. Sin embargo, se puede especificar una cadena que sirva como separador.

```
moderatII.join('-');

console.log(moderatII);
```

slice() extrae una copia de una sección especificada de una matriz, aunque a diferencia de **splice()** no la modifica.

Para este método el primer parámetro es obligatorio y el segundo opcional:

- Definidos ambos naturales, el primero es el índice desde el que debe comenzarse la copia (se incluye ese elemento en la misma), y el segundo el índice en el que debe detenerse (y este elemento que no se incluye).

- Definido sólo uno, cuenta como el índice desde el que comenzar la copia, que abarcará los elementos hasta el final de la matriz.

```
moderatII.slice(2,4);

console.log(moderatII);
```

```
moderatIII.slice(3);

console.log(moderatIII);
```

toString() este método es similar a **join()**, sólo que no admite parámetros, y que el resultado es un objeto String, con sus propias propiedades y métodos.

```
moderatII.toString()

console.log(moderatII);
```

Localizar un valor en un array

Por último, recojo dos métodos muy útiles para localizar valores en un array.

```
let modeSelector = ["Who", "Fentanyl", "Tom", "Dy"]
```

indexOf() devuelve el índice del primer elemento que coincide con el parámetro proporcionado.

```
modeSelector.indexOf('Who');
```

Si no se da coincidencia, devuelve -1.

```
modeSelector.indexOf('Pastis');
```

lastIndexOf() devuelve el índice del último elemento que coincide con el parámetro proporcionado.

```
modeSelector.lastIndexOf('Who');
```

Si no se da coincidencia, también devuelve -1:

```
modeSelector.lastIndexOf('Pastis');
```

En realidad estos dos métodos son un poco limitados, pero son útiles precisamente cuando se está comprobando la **no existencia** de un elemento en un Array.

includes() determina si una matriz incluye un determinado elemento, devuelve true o false según corresponda.

```
const listNumbers = [1, 2, 3];

console.log(listNumbers.includes(2));
// expected output: true

const listFilms = ['Reservoir dogs', 'The Big Lewoski', 'End Gar

console.log(listFilms.includes('dogs'));
// expected output: true

console.log(listFilms.includes('og'));
// expected output: false
```