STRINGS

Los strings en JavaScript En muchos sentidos, para JavaScript, un string no es más que un array de caracteres. Al igual que en los arrays, la primera posición siempre será **0.**Para acceder a un carácter puntual de un string, nombramos al string y, dentro de los corchetes, escribimos el índice al cual queremos acceder.

```
let nombre = 'Franco';
nombre [2] //----> devuelve el caracter en indice 2, la letra a.
```

PROPIEDADES

.length: Esta propiedad retorna la cantidad total de caracteres del string, incluidos los espacios. Como es una propiedad, al invocarla, no necesitamos los paréntesis.

```
let miSerie = 'Mad men';
miSerie.length;  //---> devuelve 7.
```

MÉTODOS

Cuando una función le pertenece a un objeto, en este caso nuestro string, la llamamos método.

.slice(): Corta el string y devuelve una parte del string donde se aplica. Recibe 2 números como parámetros (pueden ser negativos):
*El índice desde donde inicia el corte.
*El índice hasta donde hacer el corte (es opcional).
Retorna la parte correspondiente al corte.

```
let frase = 'Breaking Bad rules!';
frase.slice(9,12);  //-----> devuelve 'Bad'.
```

.indexOf(): Busca, en el string, el string que recibe como parámetro.
*Recibe un elemento a buscar en el array.
*Retorna el primer índice donde encontró lo que buscábamos. Si no lo encuentra, retorna un -1.

```
let saludo = 'Hola cómo estás?';
saludo.indexOf('cómo'); //----> devuelve 5.
saludo.indexOf('bienvenido') //----> devuelve -1, no lo encontró.
```

.trim(): Elimina los espacios que estén al principio y al final de un string.
*No recibe parámetros.
*No quita los espacios del medio.

```
let nombre = ' Homero Simpsons ';
nombre.trim() //----> devuelve 'Homero Simpsons'
```

.split(): Divide un string en partes.
*Recibe un string que usará como separador de las partes.
*Devuelve un array con las partes del string.

```
let cancion = 'Oops I did it again';
cancion.split(' '); //---> devuelve [ 'Oops', 'I', 'did', 'it', 'again']
```

.replace() : Reemplaza una parte del string por otra.
Recibe dos strings como parámetros:
*El string que queremos buscar.
*El string que usaremos de reemplazo.
*Retorna un nuevo string con el reemplazo.

```
let  frase = 'Aguante Phyton';
frase.replace('Phyton', 'JS') //---> devuelve 'Aguante JS'
```

MÁS MÉTODOS:
https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/String

ARRAYS

Los arrays nos permiten generar una colección de datos ordenados. Utilizamos corchetes [] para indicar el inicio y el fin de un array. Utilizamos comas , para separar sus elementos. Cada dato de un array ocupa una posición numerada conocida como índice. La primera posición de un array es siempre **0.**Para acceder a un elemento puntual de un array, nombramos al array y, dentro de los corchetes, escribimos el índice al cual queremos acceder.

```
let peliculas = ['Avatar', 'Titanic', 'Toy Story', 'Bee Movie'];
peliculas [1] // ----> accedemos a Titanic.
```

PROPIEDADES

.length: Podemos saber el número de elementos usando la propiedad length.

```
let peliculas = ['Avatar', 'Titanic', 'Toy Story', 'Bee Movie'];
peliculas.length //---> devuelve 4, la cantidad de elementos del array.
```

MÉTODOS

BÁSICOS

.push(): **Agrega** uno o varios elementos al **final** del array.
*Recibe uno o más elementos como parámetros.
*Retorna la nueva longitud del array.

```
let colores = ['Rojo', 'Naranja', 'Azul'];
colores.push('Verde') //---> devuelve 4, la nueva longitud del array.
console.log(colores) //---> ['Rojo', 'Naranja', 'Azul', 'Verde']
```

.pop(): **Elimina** el **último** elemento de un array.
*No recibe parámetros.
*Devuelve el elemento eliminado.

```
let colores = ['Rojo', 'Naranja', 'Azul'];
let eliminado = colores.pop()
console.log(eliminado) // -----> devuelve 'Azul'.
```

.shift(): Elimina el **primer** elemento de un array.
*No recibe parámetros.
*Devuelve el elemento eliminado.

```
let nombres = ['Ana', 'Maria', 'Pepe'];
let primerNombre = nombres.shift()
console.log(primerNombre); //---> devuelve 'Ana'
```

.unshift(): Agrega uno o varios elementos al **principio** de un array.
*Recibe uno o más elementos como parámetros.
*Retorna la nueva longitud del array.

```
let marcas = ['Audi'];
let nuevas = marcas.unshift('Ford', 'Fiat')
console.log(nuevas) // ---> Devuelve ['Ford', 'Fiat', 'Audi']
```

.join(): Une los elementos de un array utilizando el **separador** que le especifiquemos. Si no lo especificamos, utiliza comas.
*Recibe un separador (string), es opcional.
*Retorna un string con los elementos unidos.

```
let dias = ['lunes', 'martes', 'viernes']
let separados = dias.join (' - ');
console.log(separados) //---> Devuelve ['lunes - martes - viernes']
```

.indexOf(): Busca en el array el **elemento** que recibe como parámetro.
*Recibe un elemento a buscar en el array.
*Retorna el primer índice donde encontró lo que buscábamos. Si no lo encuentra, retorna un -1.

```
let dias = ['lunes', 'martes', 'viernes']
dias.indexOf('martes')// ---> Devuelve 1, el indice del elemento.
```

.lastIndexOf(): Similar a .indexOf(), con la salvedad de que **empieza buscando el elemento por el final del array** (de atrás hacia adelante). En caso de haber elementos repetidos, devuelve la posición del primero que encuentre (o sea el último si miramos desde el principio).

```
let clubes = ['Racing', 'Boca', 'Lanus', 'Boca'];
clubes.lastIndexOf('Boca') //---> devuelve 3.
```

.includes(): También similar a .indexOf(), con la salvedad que **retorna un booleano.**
*Recibe un elemento a buscar en el array.
*Retorna true si encontró lo que buscábamos, false en caso contrario.

```
let clubes = ['Racing', 'Boca', 'Lanus', 'Boca'];
```

AVANZADOS

.map(): Este método **recibe una función como parámetro** (callback). Recorre el array y **devuelve un nuevo array modificado**. Las modificaciones serán aquellas que programemos en nuestra función de callback.

```
let numeros = [2,4,6,8];
let elDoble = numeros.map(function(numero){
return numero *2;})
con arrow function:
let elDoble = numeros.map(numero=>numero*2);
console.log(elDoble); //---> Devuelve [4,8,12,16]
```

.filter(): Este método también **recibe una función** como parámetro. Recorre el array y **filtra los elementos según una condición** que exista en el callback. **Devuelve un nuevo array** que contiene únicamente los elementos que hayan cumplido con esa condición. Es decir, que nuestro nuevo array puede contener menos elementos que el original.

```
let edades = [22,8,17,14,30];
let mayores = edades.filter(edad => edad>18);
console.log(mayores) //---> Devuelve [22,30]
```

.reduce(): Este método recorre el array y **devuelve un único valor.** Recibe un **callback** que se va a ejecutar sobre cada elemento del array. Este, a su vez, **recibe cuatro argumentos**: un **acumulador, elemento actual, índice actual, array que esté recorriendo.**

```
let numeros = [5,17,16];
let suma = numeros.reduce((acumulador, numero)=>acumulador+numero);
console.log(suma); //---> Devuelve 28.
```

.forEach(): La finalidad de este método es iterar sobre un array. **Recibe un callback** como parámetro y, a diferencia de los métodos anteriores, este **no retorna nada. (modifica el array original)**

```
let paises = ['Argentina', 'Cuba', 'Peru'];
paises.forEach(pais => console.log(pais));
//Devuelve
Argentina
Cuba
Peru
```

.slice(): Este método **devuelve** (extrae) **una copia de una parte del array dentro de un array (subarray)**. Determina el **índice inicial y el final** (opcional). **El final no está incluido para el nuevo array**.
*Si el inicio es un valor negativo, extrae los últimos elementos del array desplazándose desde el final del mismo. Si es omitido, por defecto es 0. Si es mayor a array.length, devolverá array vacío.
*Si el final es negativo, realiza un desplazamiento al final. Por ejemplo, array.slice(3, -1) extrae desde el cuarto elemento hasta el penúltimo. Si es mayor a array.length o si es omitido, extrae hasta el final del array.

```
let numeros = [3,4,5,6,7];
let subArray = numeros.slice(0, 3);
console.log(subArray); //---> Devuelve [3,4,5].
```

.splice(): Este método nos sirve para **remover y/o agregar elementos de un array**. Recibe tres parámetros:
*inicio: el índice del primer elemento (de donde comenzará el cambio).
* cant (opcional): un entero que indica la cantidad de elementos a eliminar. Si se omite o si es mayor que (array.length - inicio), se eliminarán todos los elementos desde el inicio.
*items (opcional): indica los elementos que se agregarán en el array, desde inicio. Si se omite splice, solo elimina.

```
let numeros = [3,4,5,6,7];
numeros.splice(0, 0, 2);
console.log(numeros) //---> Devuelve [2,3,4,5,6,7]
```

.sort(): Este método nos sirve para **ordenar los elementos de un array**. Recibe un callback como parámetro (opcional) que especifica el modo de ordenamiento. Si es omitido, el array es ordenado con el valor de string (Unicode). **Convierte a string a cada elemento**. La función recibe dos parámetros, que son los elementos a comparar, el primero vs. el segundo elemento.

```
let numeros = [10,3,4,52,6,7];
numeros.sort((a,b)=>a-b);
console.log(numeros) //---> Devuelve [4,7,8,9,10,52]
```

MAS INFORMACIÓN DE SORT:
https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Array/sort

.find(): Este método **devuelve el valor del primer elemento de un array que cumple con una función** especificada (callback). Recibe un callback que se ejecuta sobre cada índice del array hasta que encuentre uno que devuelve un valor verdadero, y toma tres parámetros:
* elemento: se utiliza un alias para representar el valor del elemento actual que está procesando el array.
* index (opcional): posición del elemento actual que se está procesando en el array.
* array: que está siendo recorrido.

```
let criptos = [
{nombre: 'Bitcoin', simbolo: 'BTC'},
{nombre: 'Ethereum', simbolo: 'ETH'},
{nombre: 'Cardano', simbolo: 'ADA'}
];
let resultado = criptos.find(elemento =>elemento.nombre === 'Bitcoin');
console.log(resultado); //---> Devuelve {nombre: 'Bitcoin', simbolo: 'BTC'}
```