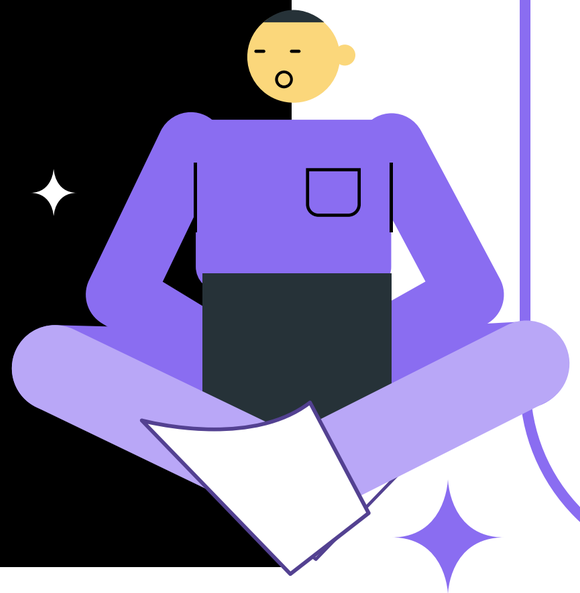
DIGITALHOUSE

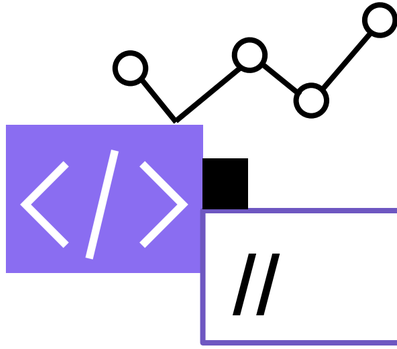
Los arrays

Índice

01 Arrays

Los arrays nos permiten generar una **colección de datos** ordenados.

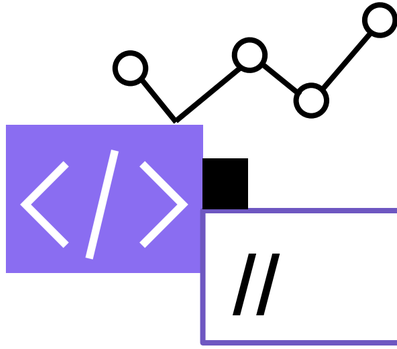




Estructura de un array

Utilizamos corchetes `[]` para indicar el **inicio** y el **fin** de un array. Utilizamos comas `,` para **separar** sus elementos.
Dentro, podemos almacenar la cantidad de elementos que queramos sin importar el tipo de dato de cada uno.
Es decir, podemos tener en un mismo array datos de tipo string, number, boolean y todos los demás.

```
{}  let miArray = ['Star Wars', true, 23];
```



Posiciones dentro de un array

Cada dato de un array ocupa una posición numerada conocida como índice. La primera posición de un array es siempre 0.

```
{}  let pelisFavoritas = ['Star Wars', 'Kill Bill', 'Alien'];
```

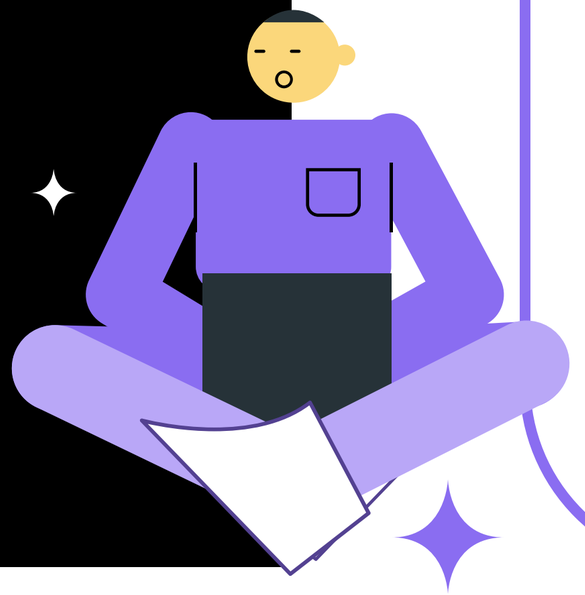
0 1 2

Para acceder a un elemento puntual de un array, nombramos al array y, **dentro de los corchetes**, escribimos el **índice** al cual queremos acceder.

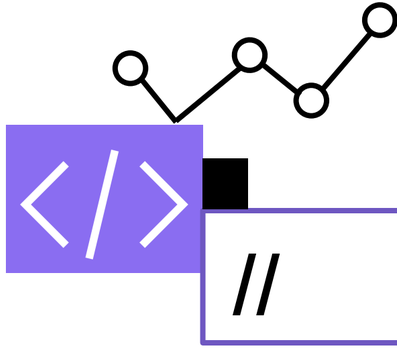
```
{}  pelisFavoritas[2];
    // accedemos a la película Alien, el índice 2 del array
```

02 Métodos de arrays I

Para JavaScript los arrays son un tipo especial de objetos.
Por esta razón disponemos de muchos **métodos** muy útiles a la hora de trabajar con la información que hay adentro.



Ya vimos antes que una función es un bloque de código que nos permite agrupar funcionalidad para usarla muchas veces.
Cuando una **función le pertenece a un objeto**, en este caso nuestro array, la llamamos **método**.



.length

Una propiedad útil de los arrays es su longitud, o cantidad de elementos. Podemos saber el número de elementos usando la propiedad **length**.

```
{}  let pelisFavoritas = ['Star Wars', 'Kill Bill', 'Alien'];
```

0 + 1 + 2 = 3

Para acceder al total de elementos de **un array**, nombramos al array y **seguido de un punto .**, escribiremos **la palabra length**.

```
{}  pelisFavoritas.length;
    // Devuelve 3, el número de elementos del array
```

.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('Violeta'); // retorna 4
console.log(colores); //
['Rojo','Naranja','Azul','Violeta']

colores.push('Gris','Oro');
console.log(colores);
// ['Rojo','Naranja','Azul','Violeta','Gris','Oro']
```

.pop()

Elimina el último elemento de un array.

- **No recibe** parámetros.
- **Devuelve** el elemento eliminado.

```
let series = ['Mad Men','Breaking Bad','The Sopranos'];

// creamos una variable para guardar lo que devuelve .pop()
let ultimaSerie = series.pop();

console.log(series); // ['Mad men', 'Breaking Bad']
console.log(ultimaSerie); // ['The Sopranos']
```

.shift()

Elimina el primer elemento de un array.

- **No recibe** parámetros.
- **Devuelve** el elemento eliminado.

```
let nombres = ['Frida','Diego','Sofía'];

// creamos una variable para guardar lo que devuelve
.shift()
let primerNombre = nombres.shift();

console.log(nombres); // ['Diego', 'Sofia']
console.log(primerNombre); // ['Frida']
```

.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'];

marcas.unshift('Ford');
console.log(marcas); // ['Ford', 'Audi']

marcas.unshift('Ferrari','BMW');
console.log(marcas); // ['Ferrari','BMW','Ford',
'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','Jueves'];

let separadosPorComa = dias.join();
console.log(separadosPorComa); //
'Lunes,Martes,Jueves'

let separadosPorGuion = dias.join(' - ');
console.log(separadosPorGuion); // 'Lunes - Martes -
Jueves'
```

.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 frutas = ['Manzana','Pera','Frutilla'];
frutas.indexOf('Frutilla');
// Encontró lo que buscaba. Devuelve 2, el índice del elemento

frutas.indexOf('Banana');
// No encontró lo que buscaba. Devuelve -1
```

.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','Lanús','Boca'];

clubes.lastIndexOf('Boca');
// Encontró lo que buscaba. Devuelve 3

clubes.lastIndexOf('River');
// No encontró lo que buscaba. Devuelve -1
```

.includes()

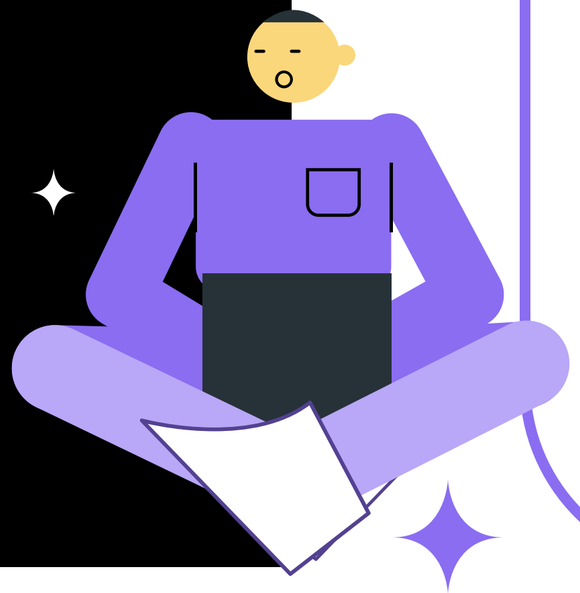
También similar a **.indexOf()**, con la salvedad que retorna un booleano.

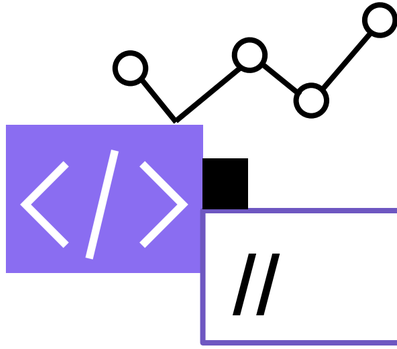
- **Recibe** un elemento a buscar en el array.
- **Retorna** true si encontró lo que buscábamos, false en caso contrario.

```
let frutas = ['Manzana','Pera','Frutilla'];

frutas.includes('Frutilla');
// Encontró lo que buscaba. Devuelve true

frutas.includes('Banana');
// No encontró lo que buscaba. Devuelve false
```

03 Métodos de strings

Para JavaScript los strings son como un array de caracteres. Por esta razón disponemos de **propiedades** y **métodos** muy útiles a la hora de trabajar con la información que hay adentro.





Los strings en JavaScript

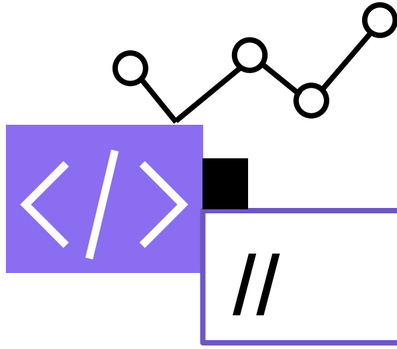
En muchos sentidos, para JavaScript, un **string** no es más que **un array de caracteres**. Al igual que en los arrays, la primera posición siempre será 0.

```
{}  let nombre = 'Fran';
```

0 1 2 3

Para acceder a un carácter puntual de un string, nombramos al string y, **dentro de los corchetes**, escribimos el **índice** al cual queremos acceder.

```
{}  nombre[2];
    // accedemos a la letra a, el índice 2 del string
```



.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

let arrayNombres = ['Bart', 'Lisa', 'Moe'];
arrayNombres.length; // devuelve 3

arrayNombres[0].length; // Corresponde a 'Bart', devuelve 4
```

.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! Estamos programando';

saludo.indexOf('Estamos'); // devuelve 7
saludo.indexOf('vamos'); // devuelve -1, no lo encontró
saludo.indexOf('o'); // encuentra la letra 'o' que está en la posición 2, devuelve 2 y corta la ejecución
```

.slice()

Corta el string y devuelve una parte del string donde se aplica.

- Recibe 2 números como parámetros:
  -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'
frase.slice(13); // devuelve 'Rules!'
frase.slice(-10); // ¿Qué devuelve? ¡A investigar!
```

.trim()

Elimina los espacios que estén al principio y al final de un string.

- **No recibe** parámetros.
- No quita los espacios del medio.

```
let nombreCompleto = '   Homero Simpson   ';
nombreCompleto.trim(); // devuelve 'Homero Simpson'

let nombreCompleto = '   Homero   J.   Simpson   ';
nombreCompleto.trim(); // devuelve 'Homero   J.
Simpson'
```

.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 = 'And bingo was his name, oh!';

cancion.split(' ');
// devuelve ['And', 'bingo', 'was', 'his', 'name,' ,
'oh!']

cancion.split(', ');
// devuelve ['And bingo was his name', 'oh!']
```

.replace()

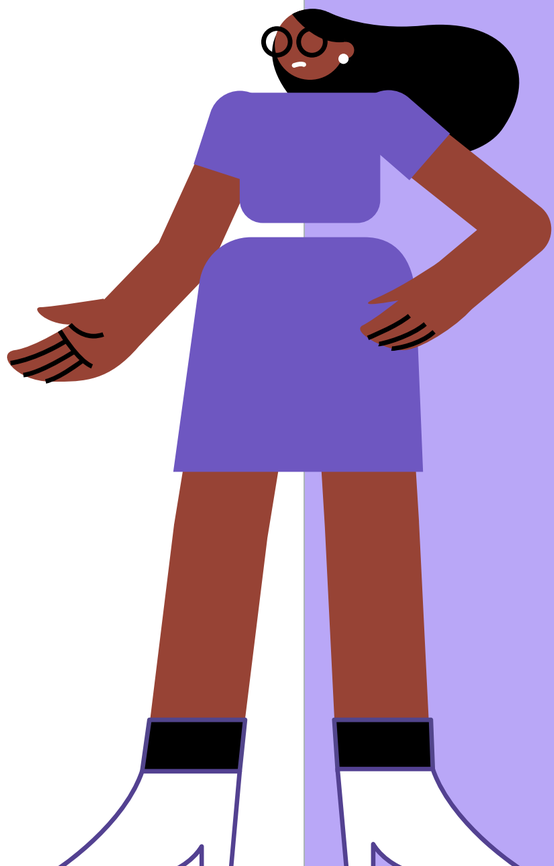
Reemplaza una parte del string por otra.

- **Recibe** 2 strings como parámetros:
  -El strings 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!'
frase.replace('Phy', 'JS'); // devuelve 'Aguante JSton!'
```

Conclusiones

Si bien cada **método** realiza una acción muy **simple**, cuando los **combinamos** podemos lograr resultados mucho más complejos y útiles.



DIGITALHOUSE

¡Muchas gracias!