

***RECUERDA PONER A GRABAR LA
CLASE***





¿DUDAS DEL ON-BOARDING?

MIRALO AQUI



Clase 06. JAVASCRIPT

ARRAYS

GLOSARIO:

Clase 5

Objeto: en programación, y también en JS, un objeto es una colección de datos relacionados y/o funcionalidad, que generalmente consta de variables y funciones, denominadas propiedades y métodos cuando están dentro de objetos. cuando necesitamos enviarle a la función algún valor o dato para que luego la misma lo utilice en sus operaciones, estamos hablando de los parámetros de la función.

- **Constructor de un objeto:** en JS, es una función donde se inicializa el mismo y todas sus propiedades.
- **Método de un objeto:** es técnicamente una función, sólo que se limita a poder ser ejecutada únicamente desde el mismo objeto.

Invocar: en programación, una invocación o llamada a una función, implica pasarle el control de la ejecución del programa, así como los argumentos o parámetros que requiere para realizar su tarea.



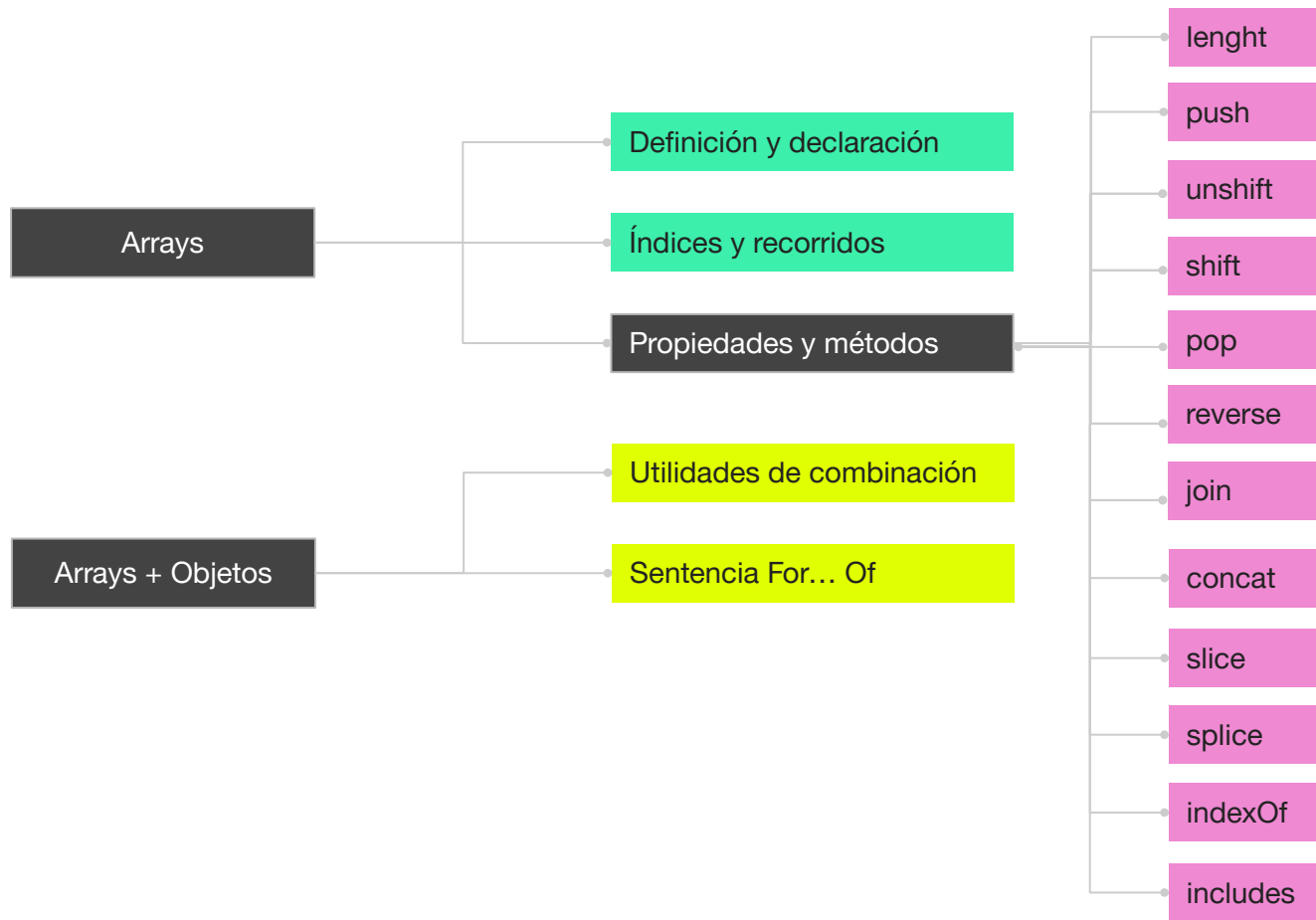
OBJETIVOS DE LA CLASE

- Definir **Arrays** en JavaScript
- Aprender cómo acceder a un **Array** y recorrerlo.
- Reconocer **propiedades y métodos** más comunes del Array.
- Comprender las utilidades de la combinación **Arrays + Objetos**.

MAPA DE CONCEPTOS

MAPA DE CONCEPTOS CLASE 6

¡Para
recordar!



MÓDULOS DE TRABAJO

MÓDULO 0 NIVELACIÓN

CLASE 0 -
INTRODUCCIÓN A JAVASCRIPT

MÓDULO 1 CONCEPTOS BÁSICOS

CLASE 1 -
CONCEPTOS GENERALES:
SINTAXIS Y VARIABLES

CLASE 2 -
CONTROL DE FLUJOS

CLASE 3 -
CICLOS E ITERACIONES

CLASE 4 -
FUNCIONES

- **Desafío entregable**

MÓDULO 2 OBJETOS & ARRAYS

CLASE 5 -
OBJETOS

CLASE 6 -
ARRAYS

CLASE 7 -
FUNCIONES DE ORDEN SUPERIOR

- **1ra pre-entrega**



HERRAMIENTAS DE LA CLASE

Les compartimos algunos recursos para acompañar la clase

- Guión de clase N° 6 [aquí](#).
- Quizz de clase N° 6 [aquí](#)
- Booklet de Javascript [aquí](#)
- FAQs de Javascript [aquí](#)

¿Arrancamos?



ARRAYS



¿QUÉ ES UN ARRAY?

Un Array es un **tipo de dato que sirve para almacenar valores en forma de lista.**

Puede ser una *colección* de números, strings, booleanos, objetos o hasta una lista de listas.

Los **valores** del array pueden ser distintos y es posible agregar o quitar elementos en todo momento.

Los elementos del array **tienen un índice**, que va desde el 0 (el primer elemento del array) hasta el último elemento.

DECLARACIÓN DEL ARRAY

Para declarar una variable y asignar un array empleamos los **corchetes** **[]** y dentro de ellos definimos todos los valores separados por coma.

Los arrays en Javascript empiezan siempre en la posición 0. Un array que tenga, por ejemplo, 10 elementos, tendrá posiciones de 0 a 9.

```
// Declaración de array vacío
const arrayA = [];

// Declaración de array con números
const arrayB = [1,2];

// Declaración de array con strings
const arrayC = ["C1","C2","C3"];

// Declaración de array con booleanos
const arrayD = [true,false,true,false];

// Declaración de array mixto
const arrayE = [1,false,"C4"];
```

ACCESO AL ARRAY

Los elementos dentro de un array tienen un **índice** que determina su **posición** en el mismo.

Así, es posible acceder a los elementos dentro de un array a través de su posición:

```
const numeros = [1,2,3,4,5];  
console.log( numeros[0] ) // 1;  
console.log( numeros[3] ) // 4;  
let resultado = numeros[1] + numeros[2]  
console.log( resultado ) // 5;
```

RECORRIDO DEL ARRAY

Decimos que estamos recorriendo un Array cuando empleamos un **bucle** para acceder a cada elemento por separado.

Los Array en JavaScript son **objetos iterables**, lo que permite usar distintas estructuras para iterar sobre ellos.

```
const numeros = [1, 2, 3, 4, 5];  
for (let index = 0; index < 5; index++) {  
    alert(numeros[index]);  
}
```

ARRAY: MÉTODOS Y PROPIEDADES

LENGTH

Al igual que en un String, la **propiedad length** nos sirve para obtener el largo de un Array, es decir, para identificar **cuántos elementos tiene**.

```
const miArray = ["marca", 3 , "palabra"];  
console.log( miArray.length ); //imprime 3
```

LENGTH

Es común utilizarlo para definir el límite de una iteración sobre un array, ya que la **propiedad length** me permite saber explícitamente la longitud del mismo:

```
const numeros = [1, 2, 3, 4, 5, 6, 7, 8]

for (let i= 0; i < numeros.length; i++) {
  alert(numeros[i]);
}
```

AGREGAR ELEMENTOS

Para sumar un elemento a un Array ya existente, se utiliza el **método push**, pasando como parámetro el valor (o variable) a agregar.

```
const miArray = ["marca", 3, "palabra"]  
miArray.push('otro elemento')  
  
console.log(miArray.length) // ⇒ 4  
console.log(miArray)  
//["marca", 3, "palabra", "otro elemento"]
```

AGREGAR ELEMENTOS

El **método push ()** agrega elementos *al final* del array. Si queremos agregar *al inicio* del array, utilizamos el método **unshift()** de forma similar:

```
const miArray = ["marca", 3, "palabra"]

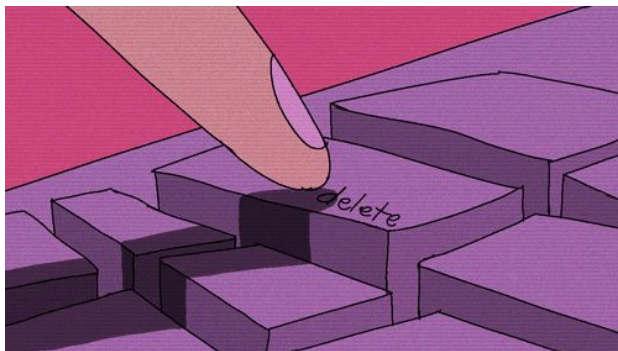
miArray.unshift('otro elemento')

console.log(miArray)
//["otro elemento", "marca", 3, "palabra"]
```

QUITAR ELEMENTOS

De forma inversa, existen métodos para eliminar elementos del array.

Si queremos eliminar el primer elemento del array utilizamos el **método `shift()`**; y si queremos eliminar el último elemento, el **método `pop()`**.



POP y SHIFT

```
const nombres = ["Luis", "Ana", "Julia", "Juan"]  
  
nombres.pop()  
console.log(nombres) // ["Luis", "Ana", "Julia"]  
  
nombres.shift()  
console.log(nombres) // ["Ana", "Julia"]
```

SPLICE

El **método splice()** permite eliminar uno o varios elementos de un array en cualquier posición. Funciona con 2 parámetros: el primero es el *índice* donde se ubica el método para trabajar, y el segundo es la *cantidad de elementos a eliminar* desde esa posición.

```
const nombres = ['Rita', 'Pedro', 'Miguel', 'Ana', 'Vanesa'];  
nombres.splice(1, 2)  
  
console.log(nombres)  
// ['Rita', 'Ana', 'Vanesa']
```

JOIN

Mediante el **método join** podemos generar un *string* con todos los elementos del array, separados por el valor que pasamos por parámetro:

```
const nombres = ["Luis", "Ana", "Julia", "Juan"]

console.log( nombres.join(", ") )
// Luis, Ana, Julia

console.log( nombres.join("*") )
// Luis*Ana*Julia
```


CONCAT

Mediante el **método concat** podemos combinar dos Arrays en un único Array resultante:

```
const perros    = ["Pupy", "Ronnie"]  
const gatos    = ["Mishi", "Garfield", "Zuri"]  
const mascotas = perros.concat(gatos)  
console.log(mascotas)  
// ["Pupy", "Ronnie", "Mishi", "Garfield", "Zuri"]
```

SLICE

El **método slice** devuelve una copia de una parte del Array dentro de un nuevo Array, empezando por el inicio hasta fin (fin no incluido).

El Array original no se modificará.

```
const nombres    = ['Rita', 'Pedro', 'Miguel', 'Ana', 'Vanesa'];  
const masculinos = nombres.slice(1, 3); // Nuevo array desde la posición 1 a 3.  
// masculinos contiene ['Pedro','Miguel']
```

INDEXOF

El **método indexOf()** nos permite obtener **el índice de un elemento en un array**. Recibe por parámetro el elemento que queremos buscar en el array y, en caso de existir, nos retorna su *índice*. Si el *elemento no existe* nos retornará como **valor: -1**

```
const nombres = ['Rita', 'Pedro', 'Miguel', 'Ana', 'Vanesa'];
```

```
console.log( nombres.indexOf('Rita') ) // ⇒ 0
```

```
console.log( nombres.indexOf('Ana') ) // ⇒ 3
```

```
console.log( nombres.indexOf('Julieta') ) // ⇒ -1
```

INCLUDES

Similar al anterior, el **método includes** me permite saber si un elemento que recibo por parámetro existe o no dentro de un array, retornando un valor booleano en caso afirmativo o negativo:

```
const nombres = ['Rita', 'Pedro', 'Miguel', 'Ana', 'Vanesa']

console.log( nombres.includes('Rita') ) // ⇒ true
console.log( nombres.includes('Miguel') ) // ⇒ true
console.log( nombres.includes('Julietta') ) // ⇒ false
```

REVERSE

Como su nombre lo indica, el **método reverse()** invierte el orden de los elementos dentro de un array.

```
const nombres = ['Rita', 'Pedro', 'Miguel', 'Ana', 'Vanesa']  
nombres.reverse()  
console.log( nombres )  
// ⇒ ['Vanesa', 'Ana', 'Miguel', 'Pedro', 'Rita']
```

REVERSE

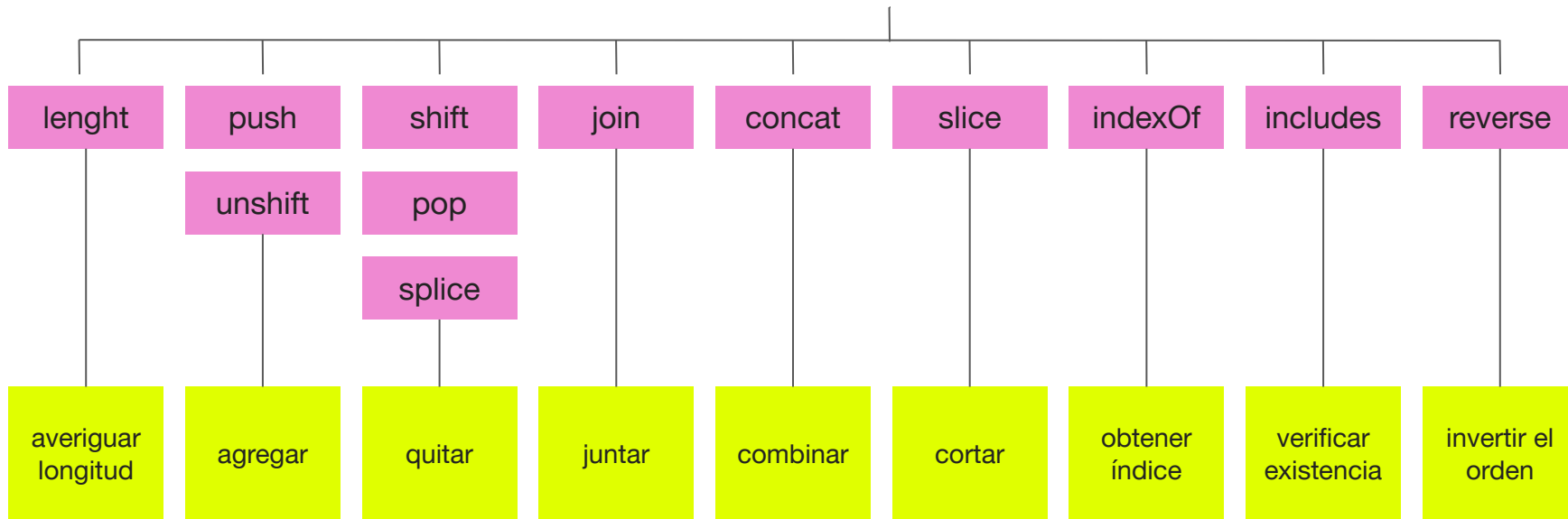
Cuidado porque el método reverse es *destructivo*, o sea que modifica el array original, al igual que los métodos para agregar o quitar elementos.





En síntesis,

MÉTODOS Y PROPIEDADES MÁS COMUNES





BREAK

¡5/10 MINUTOS Y VOLVEMOS!

Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

CODER HOUSE

EJEMPLO APLICADO: CARGAR ARRAY CON ENTRADAS

Ejemplo
en vivo



```
//Declaración de array vacío y variable para determinar cantidad
const listaNombres = [];
let cantidad = 5;
//Empleo de do...while para cargar nombres en el array por prompt()
do{
    let entrada = prompt("Ingresar nombre");
    listaNombres.push(entrada.toUpperCase());
    console.log(listaNombres.length);
}while(listaNombres.length != cantidad)
//Concatenamos un nuevo array de dos elementos
const nuevaLista = listaNombres.concat(["ANA", "EMA"]);
//Salida con salto de línea usando join
alert(nuevaLista.join("\n"));
```

EJEMPLO APLICADO: ELIMINAR CUALQUIER ELEMENTO

Ejemplo
en vivo



```
const nombres = ['Rita', 'Pedro', 'Miguel', 'Ana', 'Vanesa']

// recibo el elemento a borrar por parámetro
const eliminar = (nombre) => {
  // busco su índice en el array
  let index = nombres.indexOf(nombre)

  // si existe, o sea es distinto a -1, lo borro con
  splice
  if (index !== -1 ) {
    nombres.splice(index, 1)
  }
}

eliminar('Pedro')
```

ARRAYS DE OBJETOS

ARRAY DE OBJETOS

Los array pueden usarse para almacenar **objetos personalizados**. Podemos asignar objetos literales o previamente instanciados en la declaración del array o agregar nuevos objetos usando el **método push** y el **constructor**.

```
const objeto1 = { id: 1, producto: "Arroz" };  
const array    = [objeto1, { id: 2, producto: "Fideo" }];  
array.push({ id: 3, producto: "Pan" });
```



Thanks, brain.

¡ARRAYS + OBJETOS!

La combinación de arrays con objetos genera **estructuras complejas de datos.**

¡Los métodos de arrays y las herramientas para recorrerlos nos permiten acceder y manipular todos estos datos de forma **precisa y prolija!**

CODER HOUSE

FOR...OF

La **sentencia for...of** permite **recorrer** un array ejecutando un bloque de código por cada elemento del objeto.

```
const productos = [{ id: 1, producto: "Arroz" },  
                    { id: 2, producto: "Fideo" },  
                    { id: 3, producto: "Pan" }];  
  
for (const producto of productos) {  
    console.log(producto.id);  
    console.log(producto.producto);  
}
```

FOR...OF

Es un iterador que recorre el array de principio a fin, y en cada iteración accedemos al elemento en cuestión a través de la **referencia** que declaramos.

Por cada iteración se ejecuta el bloque de código que definimos entre llaves.

```
for (const referencia of array) { ... }
```

Es una forma más clara y rápida de ejecutar una acción por cada elemento de un array.


```
class Producto {
  constructor(nombre, precio) {
    this.nombre = nombre.toUpperCase();
    this.precio = parseFloat(precio);
    this.vendido = false;
  }

  sumaIva() {
    this.precio = this.precio * 1.21;
  }
}

//Declaramos un array de productos para almacenar objetos
const productos = [];

productos.push(new Producto("arroz", "125"));
productos.push(new Producto("fideo", "70"));
productos.push(new Producto("pan", "50"));

//Iteramos el array con for...of para modificarlos a todos
for (const producto of productos)
  producto.sumaIva();
```

EJEMPLO APLICADO: OBJETOS, PRODUCTO Y ARRAY

Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!



CODER HOUSE



INCORPORAR ARRAYS

Armar tu estructura de datos

INCORPORAR ARRAYS

Formato: Página HTML y código fuente en JavaScript. Debe identificar el apellido del estudiante en el nombre de archivo comprimido por “claseApellido”.

Sugerencia: Los Array cumplen el papel de listas en el programa. Principalmente, los usamos para agrupar elementos de un mismo tipo. Siempre que sea posible emplear los métodos disponibles para trabajar con ellos

Desafío
Complementario



>> Consigna: Traslada al proyecto integrador el concepto de objetos, visto en la clase de hoy. A partir de los ejemplos mostrados la primera clase, y en función del tipo de simulador que hayas elegido, deberás:

- Incorporar al menos un Array en tu proyecto.
- Utilizar algunos de los métodos o propiedades vistos en la clase.

>>Aspectos a incluir en el entregable:

Archivo HTML y Archivo JS, referenciado en el HTML por etiqueta `<script src="js/miarchivo.js"></script>`, que incluya la definición de un algoritmo en JavaScript que emplee array para agrupar elementos similares.

>>Ejemplo:

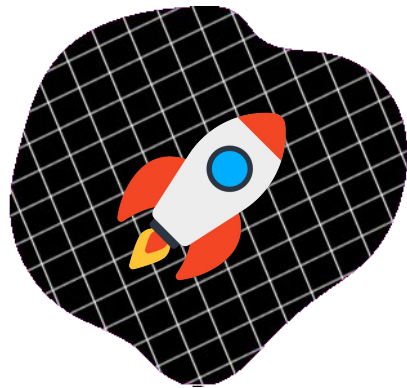
Podemos crear arrays para los objetos identificados en el simulador la clase anterior, Ejemplo: Array de Productos, Array de Personas, Array de Libros, Array de Autos, Array de Comidas, Array de Bebidas, Array de Tareas, etc.

CODER HOUSE

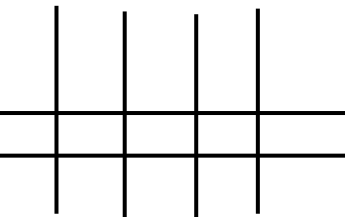
PREPARÁNDONOS PARA LA PRE-ENTREGA N° 1

En la clase 7 finaliza el segundo módulo y se entregarán las consignas de la primera pre-entrega del curso. La misma, incluirá temas vistos en las clases previas.





- La pre-entrega se compone de temas vistos hasta el momento, más otros que verán durante el **módulo completo** 💪.
- Te recomendamos ir avanzando con los "Hands On" y "Desafíos Complementarios" ✨
- Recuerden que recién la consigna del desafío se entrega ¡en la clase N° 7! 🙌 **Y tendrán hasta 7 días para resolver el desafío y subirlo.**

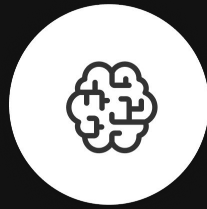


PRE-ENTREGA N° 1

Compuesto por...



- Estructura HTML del proyecto. ✓
- Variables de JS necesarias. ✓
- Funciones esenciales del proceso a simular. ✓
- Objetos de JS ✓
- Arrays - Desafío complementario **HOY**
- Métodos de búsqueda y filtrado sobre el Array - Desafío complementario **HOY**



¡PARA PENSAR!

¿Te gustaría comprobar tus conocimientos de la clase?

Te compartimos a través del chat de zoom
el enlace a un breve quiz de tarea.

Para el profesor:

- *Acceder a la carpeta "Quizzes" de la camada*
 - *Ingresar al formulario de la clase*
 - *Pulsar el botón "Invitar"*
 - *Copiar el enlace*
- *Compartir el enlace a los alumnos a través del chat*

¿PREGUNTAS?





RECURSOS:

- Array |

Los apuntes de Majo (Página 21 a 24).

- Estructuras de Datos: Objetos y Arreglos |
Eloquent JavaScript(ES).

- Práctica guiada |
Proyecto: Vida Electrónica.

- Documentación |

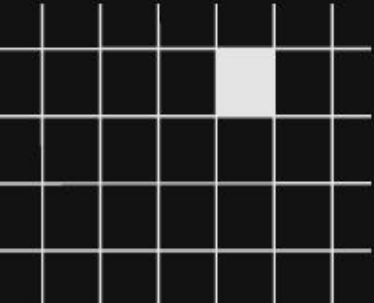
Documentación STRING.

Documentación ARRAY.



¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Métodos y propiedades en Array.
 - Función del typeof.
 - Métodos de búsqueda y transformación
- 



OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDO LA EDUCACIÓN

CODER HOUSE