

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





***¿DUDAS DEL ON-BOARDING?***

**MIRALO AQUI**



**Clase 06. JAVASCRIPT**

# ***ARRAYS***



## ***OBJETIVOS DE LA CLASE***

- Comprender y familiarizarse con funciones nativas de JS para operar Arrays.
- Reconocer cómo trabajar con Array de objetos.
- Identificar distintos algoritmos al trabajar con colecciones.

# 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.

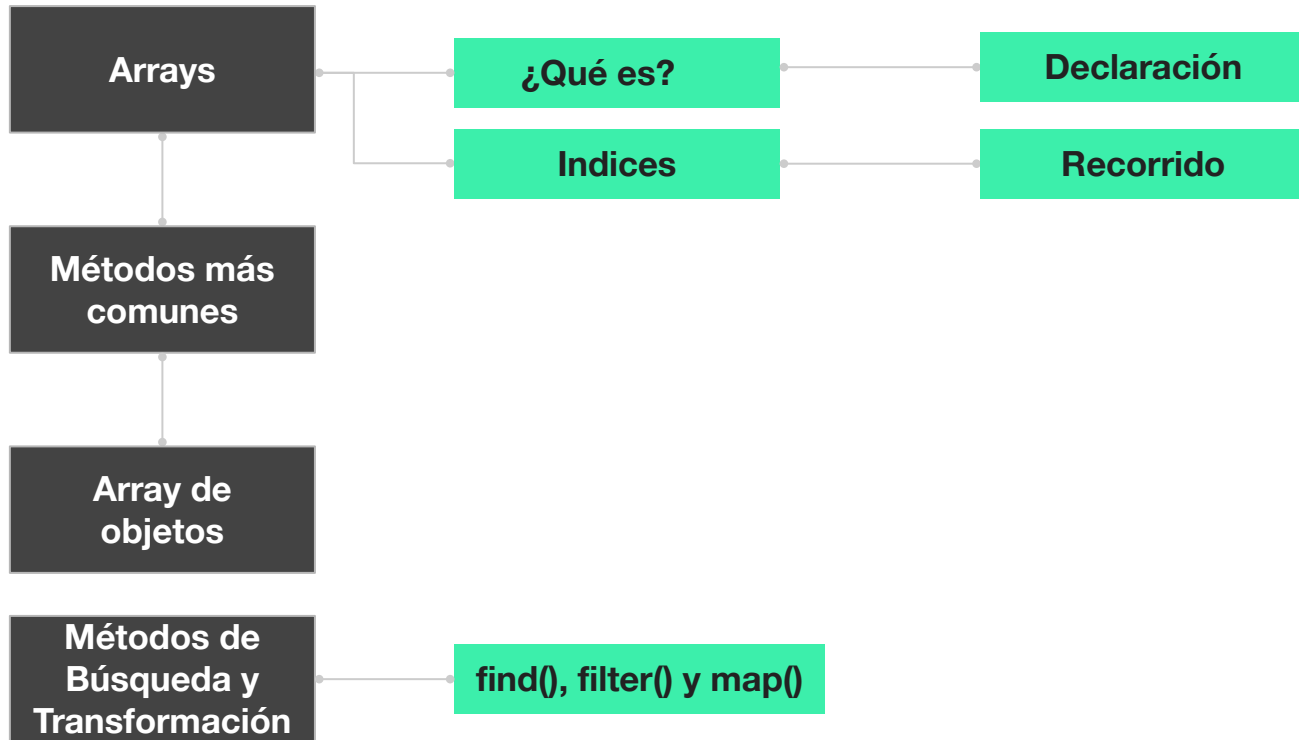
- **Constructor de un objeto:** en JS, es una función donde se inicializa el mismo y todas sus propiedades.
- **Método de un objeto:** es técnicamente una función, sólo que se limita a poder ser ejecutada únicamente desde el mismo objeto.

**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.

# ***MAPA DE CONCEPTOS***

# MAPA DE CONCEPTOS CLASE 6

¡Para  
recordar!



# ***CRONOGRAMA DEL CURSO***

## Clase 5



### Objetos



EJEMPLOS EN VIVO



CREAR UN OBJETO Y  
UTILIZARLO



INCORPORAR OBJETOS

## Clase 6



### Arrays



EJEMPLOS EN VIVO



CREAR UN ALGORITMO  
CON ARRAYS



INCORPORAR ARRAYS



PRIMERA ENTREGA DEL  
PROYECTO FINAL

## Clase 7



### Storage y JSON



EJEMPLOS EN VIVO



EJERCITAR JSON Y  
STORAGE





# ***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](#)

# ***ARRAYS***



# ¿QUÉ ES ARRAY?

Un Array es una **objeto que almacena una lista de elementos**. Puede ser un conjunto de números, strings, booleanos, objetos o hasta una lista de listas.

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

Los elementos del array tienen un orden, de 0 (el primer elemento del array) hasta el último elemento.

# ***DECLARACIÓN DE ARRAY***

Para declarar un variable y asignar un array empleamos los corchetes ([ ]) y dentro definimos todos los valores separados por coma.

Los arrays en Javascript empiezan siempre en la posición 0, así que 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

Podemos acceder a los elementos empleando Array indicando su posición. A los números de las posiciones que usamos para acceder a los elementos del array se los puede llamar índices.

```
const  numeros = [1,2,3,4,5];  
let resultado1  = numeros[0] + numeros[2]; // 1 + 3 = 4;  
let resultado2  = numeros[1] + numeros[4]; // 2 + 5 = 7;  
let resultado3  = numeros[1] + numeros[1]; // 2 + 2 = 4;
```

# ***RECORRIDO DEL ARRAY***

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

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 COMUNES***

# ***ARRAY***

## ***Métodos y propiedades más comunes***

- Conocer el largo de un array: `length`
- Pasar a String: `toString()`
- Agregar elementos: `push()`
- Juntar los elementos separándolos por un caracter: `join()`
- Combinar dos Arrays en uno: `concat()`
- Generar un nuevo Array a partir de un rango de posiciones: `slice()`



# ***LENGTH***

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

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

# ***TO STRING***

El método `toString` convierte un Array a un String, compuesto por cada uno de los elementos del Array separados por comas.

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

# ***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); //El array ahora tiene 4 posiciones
```

# JOIN

Mediante el método `join` podemos juntar todos los elementos de un Array en una cadena String, indicando como parámetro el separador para esos elementos:

```
const otroArray = ["hola", 22, "mundo"];  
console.log(otroArray.join("*")); //Imprime "hola*22*mundo"
```

# ***CONCAT***

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

```
const miArray    = ["ford", 45];  
const otroArray  = ["hola", 22, "mundo"];  
const nuevoArray = miArray.concat(otroArray);  
// nuevoArray ahora es igual a [ford,45,hola,22,mundo]
```

# ***SLICE***

Devuelve una copia de una parte del Array dentro de un nuevo Array, empezando por 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']
```

# ***EJEMPLO APLICADO: CARGAR ARRAY CON ENTRADAS***

```
//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  
en vivo



***¡VAMOS A PRACTICAR LO VISTO!***





***BREAK***

**¡5/10 MINUTOS Y VOLVEMOS!**

# ***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" });
```

# ***FOR...OF***

La sentencia for...of permite recorrer un objeto iterable (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);  
}
```

# typeof

El operador typeof devuelve una cadena que indica el tipo de dato del parámetro. El parámetro puede ser una cadena, número, variable, u objeto. Sirve para detectar qué tipo de dato es el parámetro. Por ejemplo, chequear antes de quitar espacios, que efectivamente sea un string.

```
let miFuncion = (a,b) => a + b;
let forma = " redonda ";
let tamano = 1;
console.log ( typeof miFuncion ); //imprime function
console.log ( typeof forma ); //imprime string
console.log ( typeof tamano ); //imprime number

if (typeof forma == 'string')
    forma = forma.trim();
```

## ***EJEMPLO APLICADO: OBJETOS, PRODUCTO Y 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();
```

# ***MÉTODOS DE BÚSQUEDA Y TRANSFORMACIÓN***

# MÉTODO FIND

El método find() devuelve el valor del primer elemento del Array que satisface la función de comprobación enviada por parámetro. Si ningún valor satisface la función de comprobación, se devuelve undefined.

```
const numeros    = [1, 2, 3, 4, 5];  
//La función parámetro generalmente es una función flecha sin cuerpo.  
const encontrado = numeros.find(elemento => elemento > 3); //Encuentra 4  
  
const nombres    = ["Ana", "Ema", "Juan"];  
const encontrado2 = nombres.find(elemento => elemento === "Ema"); //Encuentra "Ema"  
const encontrado3 = nombres.find(elemento => elemento === "Alan"); //undefined
```



# MÉTODO FILTER

El método `filter()` crea un nuevo Array con todos los elementos que cumplan la función de comprobación enviada por parámetro. Generalmente, se obtiene un Array con menos elementos que la lista a filtrar.

```
const numeros = [1, 2, 3, 4, 5];  
const filtro1 = numeros.filter(elemento => elemento > 3); //Encuentra [4,5]  
const filtro2 = numeros.filter(elemento => elemento < 4); //Encuentra [1,2,3]  
  
const nombres = ["Ana", "Ema", "Juan", "Elia"];  
//Filtrar nombre que incluyen la letra "n" Encuentra ["Ana","Juan"]  
const filtro3 = nombres.filter(elemento => elemento.includes("n"));
```

# MÉTODO MAP

El método `map()` se utiliza para crear un nuevo Array con todos los elementos del Array original transformados según las operaciones de la función enviada por parámetro. El nuevo Array obtenido tiene la misma cantidad de elementos que el array original pero con los valores modificados.

```
const numeros = [1, 2, 3, 4, 5];  
const porDos = numeros.map(x => x * 2); // porDos = [2, 4, 6, 8, 10]  
const masCien = numeros.map(x => x + 100); // porDos = [102, 104, 106, 108, 110]  
  
const nombres = ["Ana", "Ema", "Juan", "Elia"];  
const lengths = nombres.map(nombre => nombre.length); //lengths = [3, 3, 4, 4]
```

## ***EJEMPLO APLICADO: BUSCANDO Y FILTRANDO OBJETOS***

```
const productos = [{ id: 1, producto: "Arroz", precio: 125 },  
                    { id: 2, producto: "Fideo", precio: 70 },  
                    { id: 3, producto: "Pan" , precio: 50},  
                    { id: 4, producto: "Flan" , precio: 100}];  
  
const buscarPan = productos.find(producto => producto.id === 3);  
console.log(buscarPan); //{id: 3, producto: "Pan", precio: 50}  
  
const baratos = productos.filter(producto => producto.precio < 100);  
console.log(baratos); //  
[ {id: 2, producto: "Fideo", precio: 70}, {id: 3, producto: "Pan", precio: 50} ]  
  
const aumentos = productos.map(producto => producto.precio += 30);  
console.log(aumentos);  
//[155, 100, 80, 130] y el valor de cada producto cambio.
```

Ejemplo  
en vivo



***¡VAMOS A PRACTICAR LO VISTO!***

***CODER HOUSE***



# ***INCORPORAR ARRAYS***

Traslada al proyecto integrador el concepto de objetos, visto en la clase de hoy

# 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  
entregable



**>> 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**



# ***ORDENAR UN ARRAY DE OBJETOS***

Escribe una función para ordenar un array de objetos.

# ORDENAR UN ARRAY DE OBJETOS

**Formato:** código fuente en JavaScript

[Sublime Text](#) o [VisualStudio](#).

**Sugerencia:** recuerda que para ordenar una estructura de datos, los elementos deben ser del mismo tipo. Puedes emplear la función sort() para armar el algoritmo.

[https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos\\_globales/Array/sort](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Array/sort)

Desafío  
Complementario



**>> Consigna: codifica una función cuyas instrucciones permitan ordenar una colección (array). Preferentemente, recibir el criterio de ordenamiento por parámetro, y mostrar el resultado del procesamiento en una salida.**

**>>Aspectos a incluir en el entregable:**

Archivo HTML y archivo JavaScript referenciado, que incluya la definición un array de objetos, la declaración y llamada de una función que ordene la colección.

**>>Ejemplo de criterio de ordenamiento:**

- 1) Array de objetos "Productos". Ordenar por menor precio.
- 2) Array de objetos "Personas". Ordenar por mayor edad.
- 3) Array de objetos "Date". Ordenar por menor fecha.





# ***PRIMERA ENTREGA DEL PROYECTO FINAL***

Deberás entregar **la Estructura HTML y CSS del proyecto, las variables de JS necesarias y los objetos de JS**, correspondientes a la primera entrega de tu proyecto final.

# PRIMERA ENTREGA DEL PROYECTO FINAL

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

**Sugerencia:** Si bien, por el momento solo podemos hacer entradas con `prompt()` y salidas con `alert()` o `console.log()`, es suficiente para empezar a pensar el proceso a simular en términos de entradas, variables, estructuras, funciones, métodos y salidas. Verificar Rúbrica

Proyecto  
Final



## >>Objetivos Generales:

1. Codificar la funcionalidad inicial del simulador.
2. Identificar el flujo de trabajo del script en términos de captura de entradas ingresadas por el usuario, procesamiento esencial del simulador y notificación de resultados en forma de salida.

## >>Objetivos Específicos:

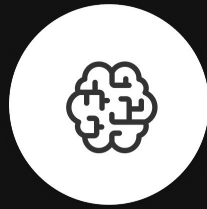
1. Capturar entradas mediante `prompt()`.
2. Declarar variables y objetos necesarios para simular el proceso seleccionado.
3. Crear funciones y/o métodos para realizar operaciones (suma, resta, concatenación, división, porcentaje, etc).
4. Efectuar una salida, que es el resultado de los datos procesados, la cual puede hacerse por `alert()` o `console.log()`.

-

# ***PRIMERA ENTREGA DEL PROYECTO FINAL***

## **>>Se debe entregar:**

- Estructura HTML del proyecto.
- Variables de JS necesarias.
- Funciones esenciales del proceso a simular.
- Objetos de JS.



## ***¡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.***

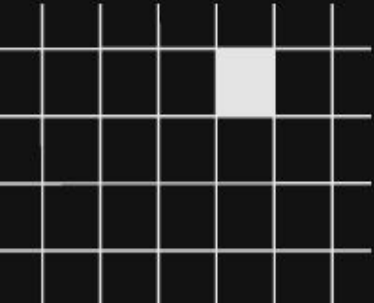
***Documentación ARRAY.***

Disponible en [nuestro repositorio](#).



# ***¡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***