



# Arrays

# ¿Qué son los Arrays?

- Hasta el momento podíamos guardar un único dato a la vez, en una variable.
- Si queríamos guardar dos nombres, lo hacíamos así:

```
let nombre1 = "Jose"
```

```
let nombre2 = "Pepe"
```

# ¿Qué son los Arrays?

- Un arrays (también llamado arreglo, vector, o matriz) es una colección de datos, los cuales se agrupan en una sola variable.
- En JavaScript los arrays son **objetos** (object)

```
> let nombres = ["Juan", "Pedro"]  
< undefined  
 > typeof nombres  
< "object"  
> |
```

- En el código anterior estamos declarando un array de nombre “nombres”, con dos elementos del tipo string.

# Como declaro un Array

- Podemos hacerlo de tres formas distintas:

```
> const unArray = Array.of("Juan", "Pepe");  
< undefined  
> unArray  
< ▶ (2) ["Juan", "Pepe"]  
  
> const otroArray = ["Juan", "Pepe"];  
< undefined  
> otroArray  
< ▶ (2) ["Juan", "Pepe"]  
  
> const tercerArray = new Array("Juan", "Pepe");  
< undefined  
> tercerArray  
< ▶ (2) ["Juan", "Pepe"]  
>
```

# Como declaro un Array: Array.of()

- Con array.of() creamos una instancia del objeto Array
- Le debemos pasar como parámetro cada uno de los elementos que queramos que tenga nuestro array a la hora de declararlo.
- `let notas = ,Array.of(1, 4, 3, 6, 11, 5);`    `// [1, 4, 3, 6, 11, 5]`

```
> let frutas = Array.of('Manzana', 'Pera', 'Banana');  
< undefined  
-----  
> frutas  
< ▶ (3) ["Manzana", "Pera", "Banana"]  
-----  
> |
```

# Como declaro un array: new Array()

- Cuando uso new Array, lo que estoy haciendo es invocar al **constructor** de la **clase** Array
- (Cuando veamos POO estos conceptos quedarán mas claros)
- A diferencia de Array.of(), que lo que hacemos es llamar al método of de la clase Array
- El constructor de Array recibirá como parámetro el total de elementos que queremos que tenga, o también podemos pasarle cada uno de los elementos (como en Array.of())
-

# Como declaro un array: new Array()

```
> const x = new Array(2);
```

```
< undefined
```

---

```
> x
```

```
< ▶ (2) [empty × 2]
```

---

```
> const y = new Array('a', 'b');
```

```
< undefined
```

---

```
> y
```

```
< ▶ (2) ["a", "b"]
```

# Diferencia entre new Array() y Array.of()

- La diferencia entre ambas formas es como maneja los parámetros de tipo entero

- Ejemplo:

`Array.of(3);` // crea un array con un único elemento: el entero 3

`new Array(3);` // crea un array de tamaño 3

Es decir, un array con tres posiciones, vacías.

- Esto solo ocurre si usamos `new Array()` con un solo parametro entero.
- `new Array(1, 2, 3)`

Este código crea un array con tres elementos: [1, 2, 3]



# Como declaro un array: []

- En lugar de recurrir a la clase Array, puedo usar la notación literal de los arrays
- `let frutas = ["Manzana", "Pera", "Kiwi"];`
- `let numeros = [2, 3, 5, 7, 8];`

# Acceder a cada elemento de un Array

- Para acceder a cada uno de los elementos de un array, lo puedo hacer a través de su índice.
- Los índices comienzan desde el cero.
- Consideremos el siguiente array:

```
let arr = [6, 4, 2];
```



- Para acceder al primer elemento lo hacemos así: `arr[0]`
- Para acceder al segundo elemento: `arr[1]`
- Para acceder al tercer elemento: `arr[2]`

# Acceder a cada elemento de un Array

```
let arr = [6, 4, 2];
```

- // Accedemos al valor de la posición 0 (valor: 6) :  
console.log(arr[0]); // 6
- console.log(arr[1]);
- console.log(arr[2]);

# Analicemos el siguiente código:

```
let arr = new Array(3);  
arr[1] = 4;  
arr[2] = 2;  
arr[0] = arr[1] + arr[2];
```

- Tarea: ¿Qué hace el código de arriba?  
¿Cual es la expresión final del array arr?

# Cosas a considerar de los Arrays

- En **JavaScript** los arrays son **Objetos**  
Por ende, tienen propiedades y métodos
- En **JavaScript**, podemos tener un array con varios elementos de distintos tipos:

```
> const miArray = [null, 1, "Hola", 12, false, undefined, []]  
< undefined  
  
> miArray  
< ▶ (7) [null, 1, "Hola", 12, false, undefined, Array(0)]  
>
```

- En el código anterior definimos un array de nombre miArray que contiene enteros, booleanos, strings, valores null, undefined, e incluso un objeto Array



# Propiedades de los Arrays

- Al ser un objeto, tiene varias propiedades y métodos, los cuales mas adelante veremos con detalle
- Pero ahora, vamos a ver algunas propiedades y métodos básicos:

# Length

- Con la propiedad length accedemos al tamaño del array

```
> const nombres = ["Maria", "Juan", "Sofia", "Nico"];  
< undefined  
-----  
> nombres.length  
< 4  
-----  
> |
```

- Notar que los índices van del 0 al 3, pero el tamaño del array es de 4 elementos.
- Notar que length al ser una propiedad y no un método, no debemos indicar los paréntesis ()

# Analizar

- 1) ¿Cual es el valor de la propiedad length?:

```
> let clientes = [];  
< undefined
```

- 2) ¿Cual es el valor de la propiedad length?:

```
> let personas = new Array(3);  
< undefined
```

- 3) ¿Cual es el índice máximo del array anterior?



# Respuestas

```
> let clientes = [];  
< undefined  
> clientes.length;  
< 0  
> let personas = new Array(3);  
< undefined  
> personas.length  
< 3  
> personas  
< ▶ (3) [empty × 3]  
>
```

El índice máximo de `personas[]` es el numero 2

# Método Array.concat()

- concat() se utiliza para unir dos o mas arrays.
- Este método no modidica ningún array, sino que recibe dos arrays como parámetro, y retorna un nuevo array resultante de la unión de los arrays.
- Ejemplo (sacado de la documentación de javascript):

```
1 const array1 = ['a', 'b', 'c'];
2 const array2 = ['d', 'e', 'f'];
3 const array3 = array1.concat(array2);
4
5 console.log(array3);
6 // expected output: Array ["a", "b", "c", "d", "e", "f"]
7
```

# Array.push()

- El método push añade uno o mas elementos al final del array

```
> const menu = ["Milanesas", "Carne", "Hamburguesas"]
< undefined
> menu.push("Helado")
< 4
> menu
< ▶ (4) ["Milanesas", "Carne", "Hamburguesas", "Helado"]
> menu.push("Puré", "Café")
< 6
> menu
< ▶ (6) ["Milanesas", "Carne", "Hamburguesas", "Helado", "Puré", "Café"]
>
```

- Notar que push() nos devuelve el nuevo valor de la propiedad length

# Array.toString()

- El método toString devuelve una representación en string del array:

```
< ▶ (6) ["Milanesas", "Carne", "Hamburguesas", "Helado", "Puré", "Café"]  
> menu.toString()  
< "Milanesas,Carne,Hamburguesas,Helado,Puré,Café"  
> |
```

- Otro ejemplo:

```
> const arr = Array.of('a', 1, false, 'b');  
< undefined  
> arr.toString()  
< "a,1,false,b"  
> |
```

# Array.join()

- Este método hace prácticamente lo mismo que toString()
- Tienen algunas minúsculas diferencias en cuanto al rendimiento.

```
> const arr = Array.of('a', 1, false, 'b');  
< undefined  
-----  
> arr.toString()  
< "a,1,false,b"  
-----  
> arr.join()  
< "a,1,false,b"  
-----  
> |
```

# Ejercicios:

- 1)
  - a) Declare un Array de tamaño 20
  - b) Asigne a los primeros 10 elementos con los múltiplos del 2
  - c) Asigne a los elementos restantes con las letras del abecedario (en orden)
  - d) Imprimir por consola los primeros 13 elementos del array
- 2) a) Crear un array con los nombres de 5 personas.
  - b) Al array creado anteriormente, agregarle tres valores, con los apellidos de las personas.



# Próxima clase:

- Vamos a ver como recorrer (iterar) arreglos
- Vamos a seguir viendo otros métodos de los arreglos
- Haremos mas ejercicios.