Características

- Objetos que permiten manejar conjuntos de datos.
- Se les llaman también listas, vectores, arreglos.
- Son dinámicos, su tamaño se puede modificar después de la declaración.
- **Son heterogéneos. Cada elemento del array puede ser de un tipo diferente.**
- La primera posición siempre es 0

Declarar Arrays

- Declarar array vacio:

```
let a = [];
let a = new Array();
```

- Declarar con valores:

```
let a = new Array(3,4,5,6,7);
let b =[1,true,[1,2,3],”cad”];
let c = [3,4,5,6,7];
let d= Array(100).fill(false);

console.log(a[1]); // muestra 4
console.log(a); // muestra [3,4,5,6,7]
```

- Para asignar valores:

```
a[0] = ”Rojo”;
a[1] = “Azul”;

console.log(a[1]); // muestra azul
```

Valores indefinidos

✔ Ejemplo:

```
let a =[“Saul”, “Rocio”];
a[3]=”Maria”;
console.log(a[2]);     // escribe “undefined”
```

✔ Ejemplo:

```
let a = [“Saul”, “Rocio”, ,“Maria”];
console.log(a[2]);     // escribe “undefined”
```

Borrar elementos

```
let dias=[“Lunes”,”Martes”,”Miercoles”];
dias[3]=”Jueves”;
delete dias[2]; → Ojo, no afecta a la longitud de la cadena
console.log(dias);   //  Como queda el array? ¿Qué longitud tiene? ¿que valor tiene dias[2]?
```

Arrays heterogéneos

- ✔ Ejemplo.

```
let a =[3, 4, ”Hola”, true, Math.random()];
```

- ✔ Ejemplo:

```
let b=[3, 4, ”Hola”, [ 99, 55, 33]];
console.log(b[3][1]); // Que devuelve?
console.log(b[2][1]); // Que devuelve?
```

Recorridos de Arrays

- **Con for**

```
let notas =[5,7,3,8,5,3,8];
for(let i =0; i< notas.length; i++)
{
    console.log(“La nota ” + i + “ es ” +  notas[i]);
}
```

Y si hay elementos indefinidos en el array?

```
notas =[5,7,,,5,3,8];
for(let i =0; i< notas.length; i++)
{
    console.log(“La nota ” + i + “ es ” +  notas[i]);
}
```

- Hay que controlar los elementos undefined

```
notas =[5,7,,,5,3,8];
for(let i =0; i< notas.length; i++)
{
    if(notas[i]!=undefined)
        console.log(“La nota ” + i + “ es ” +  notas[i]);
}
```

- **Con for .. in**
  - Se saltan los indefinidos

```
notas =[5,7,,,5,3,8];
for(let i in notas)
{
    console.log("La nota " + i + " es " +  notas[i]);
}
```

- **Con for .. of**
  - Surge a partir del ES2015
  - NO Se saltan los indefinidos

```
notas =[5,7,,,5,3,8];
for(let i of notas)
{
    console.log(i);   //¿Que muestra?
}

notas =[5,7,,,5,3,8];
for(let i of notas)
{
    if(i!=undefined)
        console.log(i);
}
```

Métodos de Arrays

- **Length.** Devuelve la longitud del array

```
dias =["Lunes","Martes","Miércoles","Jueves","Viernes","Sábado","Domingo"];
console.log(dias.length);
```

- **Instanceof.** Permite saber si el objeto es un array. Devuelve un valor booleano.

```
let a =[1,2,3,4,5,6,7];
let b="Hola";
console.log(a instanceOf Array); // Que devuelve?
console.log(b instanceOf Array); // Que devuelve?
```

- **Push**: Añade elementos al final de un array

  let colores=[“verde”];

  colores.push(“blanco”); // el array tendrá dos elementos

- **Pop**: retira del array el último elemento

  colores.pop(); // el array solo tendrá un elemento, “verde”

- **Shift**: Quita el primer elemento de un array
- **Unshift**: Añade un elemento al inicio del array

  colores.unshift(“naranja”); // el array tendrá: “naranja, verde”

  colores.shift(); // el array tendrá verde

- **concat**: Para unir dos arrays en uno nuevo. No se modifica ninguno de los arrays originales:

  b.concat(a);

- **slice**: Obtiene un subarray, indicando el indice del primer elemento que deseamos obtener y el índice del final (el último no se incluye y es optativo)

  let nombres=[“Juana”,”Pedro”,”Miguel”,”Ana”,”Pepa”];

  let masculinos = nombres.slice(1,3); // masculinos contiene [“Pedro”,”Miguel”]

- **splice**: permite añadir y eliminar elementos al array: splice(start[, deleteCount[, item1[, item2[, ...]]]])

  Se indica el índice desde donde comienza la eliminación y cuantos elementos eliminamos.

  Eje: colores.splice(1,1); //Borra el elemento de la posición 1.

  let colores =[‘blanco’,’negro’,’azul’,’lila’];

  let removed = colores.splice(2,0,’verde’); // añade el color verde en la posición 2

  colores=[‘blanco’,’negro’,’verde’,’azul’,’lila’];

  removed = colores.splice(3,1); // elimino el color azul

- **Join**: permite convertir un array en un string con todos los elementos del array separados por coma. Se puede indicar otro separador.

  colores.join();

  colores.join(“-”);

- **IndexOf**: Busca el índice de un elemento de un array. Si no lo encuentra, devuelve -1. Permite indicar el inicio de la búsqueda.
- **LastIndexOf**. Empieza a buscar desde el último elemento. Permite indicar el inicio de la búsqueda
- **Includes**. Busca un elemento y devuelve true si lo encuentra o false en caso contrario
- **Reverse**: Invierte un array.
- **Sort**: Permite ordenar los elementos de un array.
- **ForEach**: Sirve para recorrer arrays y puede llamar a una función por cada elemento que puede recibir 3 parametros: (elemento actual, [indice, array objetivo]) // [] opcionales

  ✔ Ejemplo de recorrido con foreach;

```
let ranks = ['A', 'B', 'C'];
ranks.forEach(function (e) {
    console.log(e);
});
```

```
let ranks = ['A', 'B', 'C'];
ranks.forEach((e) => {
    console.log(e);
});
```

✔ Ejemplo de recorrido con foreach; → ¿**Cómo obtenemos el índice de cada elemento**?

```
ranks.forEach(function (e,index) {
        console.log(e + " indice: " +index);
});
```

- **Filter**: devuelve un array nuevo con los elementos que cumplan una condición
  - ✔ Ejemplo: Dado un array de números, seleccionar los pares.

```
const numbers = [1, 2, 3];
const evens = numbers.filter(number => number % 2 === 0); // es par
console.log(evens); // [2]
```

- **Map:** Devuelve un nuevo array con los valores que se calculen en cada elemento
  - ✔ Ejemplo: Dado un array de números, devolver otro con sus valores dobles:

```
const arr =[2,3,4]
const newArr = arr.map(el => el * el);
return console.log(`Array original ${arr} \nArray con el doble ${newArr}`);
```

Arrays multidimensionales

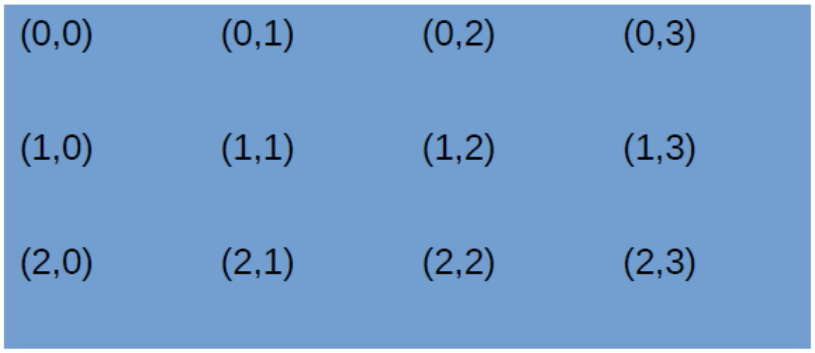
✔ Ejemplos de array de una dimensión:

```
let notas = new Array();
let notas = new Array('red');          // Crea un array con 1 elemento
let notas = new Array(9,8,10,7,6);  // Crea un array con 5 elementos
let colores = ['green', 'white'];      // Crea un array con 2 elementos
let notas = new Array(14);             // Cuidado, Que hace?
```

✔ Ejemplo: Si queremos crear un array con 3 temperaturas:

```
var temp= new Array(3);
temp[0]= 12;
temp[1]=10;
temp[2]=11;
```

- Si queremos crear un array bidimensional, por ej una tabla de 3 filas y 4 columnas:




```
var matriz = new Array(3) // creamos un array de 3 posiciones
matriz[0] = new Array(4); // en la primera posición creamos un array con 4 posiciones
matriz[1] = new Array(4); // en la segunda posición creamos un array con 4 posiciones
matriz[2] = new Array(4); // en la primera posición creamos un array con 4 posiciones
```

1. Con el constructor Array:

```
const M = 3, N = 4;
var arr = new Array(M);          // crea una array vacía de longitud `M`
for (var i = 0; i < M; i++) {
    arr[i] = new Array(N);       // hacer de cada elemento una array
}
```

2. Usando la función map:

```
const M = 3, N = 4;
var arr = Array(M).fill().map(() => Array(N));
```

Objetos literales

- Objetos en los que se pueden definir directamente sus propiedades y sus métodos:

```
let perro = {
nombre:"Scott",
color:"Cafe",
edad: 5,
macho: true
};

console.info(perro.nombre); // Scott
console.info(perro.edad); // 5
console.info(perro['nombre']); // Scott
```

```
console.info(perro['edad']); // 5
```

- Un objeto tiene **propiedades** y **métodos**. Accedemos a ellos a través **de un punto**.
  - **Objeto.propiedad;**
    - Ejemplo:

      coche.color=”verde”; // modificamos la propiedad color del coche
  - **Objeto.metodo();**
    - Ejemplo:
    - coche.acelerar(25);
- Métodos de un objeto:

```
let perro = {
    nombre:"Scott",
    color:"Cafe",
    edad: 5,
    macho: true,
    ladrar: function(){
        return(`${this.nombre} puede ladrar`)
    }
};
console.log(perro.ladrar()); // Scott puede ladrar
```

- **Inserción** de una nueva propiedad

```
let perro = {
    nombre:"Scott",
    color:"Cafe",
    edad: 5,
    macho: true,
    ladrar: function(){
        return(`${this.nombre} puede ladrar`)
    }
};
perro.tamaño = "Grande";
console.log(perro);
```

- **Modificar** una propiedad:

```
perro.edad = 8;
console.log(perro);
```

- **Eliminar** una propiedad: delete nombre_del_objeto.clave;
  - Ejemplo:

```
delete perro.color;
console.log(perro);
```

- **Recorrer** propiedades de uno objeto:

  Dado un objeto literal 'punto':

```
for (let prop in punto)
{
    console.log (`${prop} tiene el valor ${punto[prop]}`);
}
```

  - Si no queremos mostrar las funciones:

```
for (let prop in punto)
{
    if(typeof punto[prop] != "function")
        console.log (`${prop} tiene el valor ${punto[prop]}`);
}
```

- **Objeto this**
  - Hace referencia al objeto actual. Nos permite llegar al objeto propietario de la propiedad o del método.

Ejercicios

1. Dado un array de números: const arr = [2, 3, 4, 5, 0]:
   1. Escribir por consola la suma del array. Hacerlo con el método forEach y arrow functions.
   2. Escribir por consola la media.
   3. Obtener otro array con el triple de cada elemento y mostrarlo por consola.
   4. Obtener el mismo array con el triple de cada elemento y mostrarlo por consola.

2. Crear un array con 3 palabras que se introducirán por mensajes al usuario palabra a palabra (3 veces)

   Si cancela, se insertará una cadena vacía en el array.

   Escribir por consola y por pantalla, el array inicial y el array filtrando sólo las que comienzan por la letra C.

   Si no hay ninguna, escribir: “No hay ninguna palabra que comience por C.”

3. Crea una tabla bidimensional de longitud 5x15. Carga la tabla con dos únicos valores 0 y 1, donde, el valor uno ocupará las posiciones de los elementos que delimitan la tabla, es decir, las mas externas, mientras que el resto de los elementos contendrán el valor 0.

   Imprimir la tabla por pantalla:

   1111111111111111
   100000000000001
   100000000000001
   100000000000001
   1111111111111111

   Utilizar un método que cree la tabla: creaMatriz() y otro que la pinte por pantalla: pintaTabla()

4. Crear un array con 5 nombres de alumnos. Crearlos en el código como variable global. Crear una página que tenga 3 botones y un textarea donde se muestre el contenido de las notas cuando se pulse el botón 3.
   1. botón ‘Aprobado general’: que al pulsar ponga la calificación de 5 a todos los alumnos
   2. botón ‘Actualizar nota’: que pida código del alumno y nota para actualizarla.
   3. botón ‘Mostrar notas actuales’: que muestre los alumnos y sus notas

5. Realizar programa donde el usuario introduce una palabra y devuelva el número total de vocales contenidas. Escribir por pantalla el resultado: “La palabra X tiene Y vocales”.

   Utilizar la función forEach (PISTA: pasar de cadena a array.)

6. Dado un array con los 7 días de la semana, mostrar por pantalla la longitud de cada palabra, así como el día de la semana más largo. Escribir un fichero js externo para javascript. Usar forEach.

7. Hacer un programa que pida por pantalla al usuario una palabra y la invierta. Debe mostrar por pantalla la palabra original y la invertida:

   Ejemplo: La palabra hello invertida es olleh

8. Crear una función con flecha, que dado un array de números: [1,3,2,5,7,4], devuelva dos arrays, uno con los números pares y otro con los impares. Mostrar por pantalla ambos arrays. Usar filter.

   Realizar las siguientes comprobaciones: que se reciba un array, que no esté vacio y que sólo contenga números.

9. Crear un objeto literal llamado factura con las propiedades:
   1. numero, cliente, divisa, subtotal e IVA, dándole valores a cada uno de ellos.
   2. Tendrá también un método que calcula el total (subtotal + iva)
   3. Imprimir por consola: La factura X(numero) tiene un importe de Y(subtotal) Z(divisa)

10. Crea un objeto literal llamado ‘user’ vacío.
    1. Agrega la propiedad *name* con el valor John.
    2. Agrega la propiedad *surname* con el valor Smith.
    3. Cambia el valor de name a Peter.
    4. Elimina la propiedad name del objeto.

11. Escribir una función flecha isEmpty, que indique por consola si un objeto literal está vacío: no tiene ninguna propiedad.

    Ejemplo: Debe devolver true si se prueba sobre el objeto : let punto={};

12. Tenemos un objeto que almacena los salarios de nuestro equipo:

```
let salarios = {
    John: 100,
    Ann: 160,
    Peter: 130
};
```

    Escribe el código para sumar todos los salarios y mostrar por pantalla el resultado de la suma. En el ejemplo de arriba nos debería dar 390. Si salarios está vacío entonces el resultado será 0.

13. Dado un objeto literal llamado ‘menu’, crea una función flecha multiplyNumeric(menu) que multiplique todas las propiedades numéricas de ‘menu’ por 2.

    Por ejemplo:

```
// Antes de la llamada
    let menu = {
        width: 200,
        height: 300,
        title: "Mi menú"
    };

multiplyNumeric(menu);
// Después de la llamada
    menu = {
        width: 400,
        height: 600,
        title: "Mi menú"
    };
```

14. Escribir un objeto literal *cliente*, con propiedades nombre, email y direccion. Donde direccion es otro objeto literal con las propiedades: calle, num y ciudad.
    1. Asignarle valores
    2. Modificar la ciudad de la direccion.

15. Crear un objeto literal *gato c*on las propiedades: nombre (cadena), duerme(booleano), edad(number), y enemigos (array de cadenas con sus enemigos). Dad valores a cada uno.
    1. Mostrar por consola su primer enemigo
    2. Agregarle una nueva propiedad: color y asignarle valor.
    3. Actualizar la edad del gato
    4. Borrar la propiedad duerme

16. Crear un objeto literal gato: con nombre, duerme, edad, enemigos (como en el ejercicio anterior) y con un método comer() que imprima por consola: “Ahora está comiendo.”

    1. Añadir al ejercicio el método comerAlgo que tendrá un parámetro de entrada y tendrá que mostrar por consola: “Ahora esta comiendo xxxxx”

    2. Añadir al ejercicio el método quienCome, que tendrá la comida como parámetro de entrada y mostrará por consola: “El_nombre_del_gato esta comiendo comida_parametro”

    3. Mostrar por consola sus propiedades

17. Crear una página HTML con un botón para llamar a la función flecha crearObjeto. Esta función debe definir un objeto Taxi con 5 propiedades: tipoMotor, numeroPasajeros, carga, velocidad y ruedas. Tendrá un método saludar, que mostrará un mensaje alert de la forma: ‘Hola soy un taxi de X ruedas y Y pasajeros’. Después de crear el objeto taxi, invocar la función saludar.