# INDICE

# INTRODUCCIÓN

## Ejercicio 1:

Crea un array llamado elementos que tenga 10 datos de tipo string (tipo primitivo).
Sobre el array creado realiza las siguientes acciones.

- Visualiza por pantalla todo el contenido del array, separando cada dato en líneas distintas.
- Añade al array un dato más. (mediante el uso [longitud]
- Añade al array dos datos más mediante utilizando un solo método.
- Añade un dato más al principio del array.
- Localiza un cierto dato dentro del array.
- Elimina los últimos tres datos del array.
- Crea un sub-array llamado array_recortado con los datos del array elementos, comprendidos entre la posición 4 y 8 (ambos inclusive).
- Crea un nuevo array llamado elementos_MYCLS con los datos del array elementos en mayúsculas.

## Ejercicio 2:

Sigue los pasos del vídeo y genera el código necesario para cargar un fichero de texto.

A continuación se va a modificar el código para que en vez de aparecer en la página html el contenido del fichero, se cargue en un array llamado datos. Para ello, se debe añadir la declaración de la variable tipo array:
var datos = new Array();

Y a continuación, se asigna el contenido de cada línea del fichero al array, teniendo en cuenta hay que dividir en partes todo el string de contenido, como la separación es línea a línea, se utiliza como delimitador '\n'-
datos = contenido.split('\n');
Por último, visualizamos el array datos.

# EJERCICIOS

## Ejercicio 1

### Código:

```
const elementos = [
    "Rojo",
    "Azul",
    "Verde",
    "Amarillo",
    "Naranja",
    "Violeta",
    "Negro",
    "Blanco",
    "Gris",
    "Marrón"
];

function ejercicio1() {
    // Visualiza por pantalla todo el contenido del array, separando cada dato en líneas distintas.
    console.log(elementos);

    // Añade al array un dato más. (mediante el uso [longitud]
    elementos[elementos.length] = "Cián";
    console.log(elementos);

    // Añade al array dos datos más mediante utilizando un solo método.
    elementos.push("Morado", "Rosa");
    console.log(elementos);

    // Añade un dato más al principio del array.
    elementos.unshift("Color");
    console.log(elementos);

    // Localiza un cierto dato dentro del array.
    console.log(elementos.indexOf("Cián"));

    // Elimina los últimos tres datos del array.
    elementos.splice(elementos.length - 3, 3);
    console.log(elementos);

    // Crea un sub-array llamado array_recortado con los datos del array elementos, comprendidos entre la posición 4 y 8 (ambos inclusive).
    const array_cortado = elementos.slice(4, 9);
    console.log(array_cortado);

    // Crea un nuevo array llamado elementos_MYCLS con los datos del array elementos en mayúsculas.
    const elementos_MYCLS = Array.from(elementos).map(e => e.toUpperCase());
    console.log(elementos_MYCLS);
}
```

### Explicación:

Establezco un array inicial denominado elementos, que contiene diez colores. Primero examino el contenido de la función ejercicio1(). Después, agrego el color "Cian" utilizando la longitud del array como índice y luego incorporo "Morado" y "Rosa" al mismo tiempo con push(). Coloco "Color" al comienzo del array usando unshift(). Encuentro la posición del color "Cian" usando indexOf(). Procedo a suprimir los tres últimos elementos utilizando el método splice(). Para crear un nuevo array llamado "array_cortado", utilizo slice(4, 9) para extraer una parte del original. Por último, creo el array elementos_MYCLS al aplicar toUpperCase() sobre el array principal para transformar todos sus componentes a mayúsculas.

# Ejercicio 2

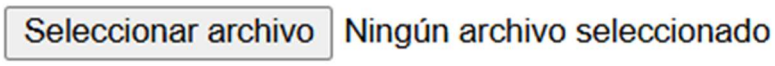
## Código:

```
document.addEventListener('DOMContentLoaded', function() {
    const fileInput = document.getElementById('fileInput');
    const contenedor = document.getElementById('contenedorContenido');

    fileInput.addEventListener('change', function(event) {
        const archivo = event.target.files[0];

        if (!archivo) {
            contenedor.textContent = 'No se ha seleccionado ningún fichero.';
            return;
        }

        const reader = new FileReader();

        reader.onload = function(e) {
            const contenido = e.target.result;

            var datos = contenido.split('\n').map(linea => linea.replace('\r', ''));


            contenedor.innerHTML = '<pre>' + JSON.stringify(datos, null, 2) + '</pre>';
        };

        reader.onerror = function(e) {
            contenedor.textContent = 'Error al leer el fichero: ' + e.target.error;
        };

        reader.readAsText(archivo);
    });
});
```

## Explicación:

Cuando el **DOM** está cargado, **obtengo** las referencias al input de tipo archivo (`fileInput`) y al contenedor donde **mostraré** el resultado. **Añado** un escuchador al evento **`change`** del input. Al seleccionar un archivo, **compruebo** que exista, y si no, **muestro** un mensaje y **salgo**. **Instancio** un nuevo objeto **`FileReader`**. Defino su evento **`onload`** para que, al cargar, **divida** el contenido del fichero por saltos de línea (`\n`), **elimine** los retornos de carro (`\r`) y luego **muestre** el *array* resultante en el contenedor dentro de etiquetas `<pre>` y **formateado** como JSON. También **defino** el evento **`onerror`** para **capturar** y **mostrar** posibles errores de lectura. Finalmente, **inicio** la lectura del archivo como texto con `reader.readAsText(archivo)`.

# RESULTADOS






```
[
  "PK\u0003\u0004",
  "\u0000\u0000\u0000\u0000\u0000\u0000e_O\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0005\u0000\u0000\u0000uxos/PK\u0003\u0004\u0014\
  "A3@U@@@@@@l@tL  @@Z](!@(7\u0006@!@X@@@ē\u0018@\u001f@@@>@R@\t\u0000@@;\u001a@a\u0010;@\u0003@p;v\u001c@\u0002wu\u000eQ@!;X\u0003%kTō@@
  "a@\u000e$u@@\"\u00001@~E#@]\u0017@P@O@+@K@@T$@@A\b@@@@0@@@ .@@T2W@H@@@\u001a@R@\u0006R@M\u001an@@l\u0019@\\2@@@@@<@@\f@h@\u001
  "9%y@\u0000@i@U@n@\u0002@\u0016@@\u0016@'\u000e>@@@@.\u001c@@@%@@\u001f}8@\\m@O@@@DCq*@@@@&\u001eY<@%@@@@+2@@;@k@@Z{e@c,V@@n^/@@s\u00
  "\u0000\u0000\u0000\u0000\u0000\u0000e_O\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\t\u0000\u0000\u0000\u0000uxos/css/PK\u0003\u0004\u0014\
  "@\u000f@@@@v@.He@@@T@'@@ @@~@~@@@i@do}^9K@^m@v@@@@@M@B'@c@@j _\u0017@@@|@@l@Wl@s\u000b\u0005=?\u001f@mU7F澶ma@@~@e\u000ea@@@
  "\u0013@@s7>#@@\u000f@\u001e\u0011@p@@3@\u0013@@'L@@%\\g@TC@) [\u0006@@Q@:@@Dr\u0003\"@\u0018@\u001e\"T@\u0011@&@ \u0011@\u001a@\u001d@@@
  "\"0@l@Π@@@2Y@@\u0013uQ&@@d@\u0006f@\u000e$@@@9d\u0019l\u001a@@@nS@MJ@]@l.\u002b@Z@\u0010\"@'@@M@I\u0018M@д@ђ}U@Z@@P@Myk\u0005Q@@\u6
  "ך\u0012@[\u0018@@@@v@|@@6\u0015&n@@@v@@\u001al@@@@Zv@Ża7@8@l@\u000b@A@g@@@@ @B\u000b@@=@+h@@@@\u000b@\u001fUZ@\u0015@@7D@a[d@
  "%@@G@@:i@jGJ+Rlw.@ @\u000b@@@7W_@@~\\\\@x{@@MUCz@l@@#1@:@)@JI@\u001a\u000fs\u001b@@2-\u0011@\u0011@{+X@@\u0005\u001b\u001d@[@\u0002@s
  "@ 6E@%{b\u0001z}@\t@@h@@@\u0004@@@\u000e{l@@@\\@@3@+\f@\u0016(@=\"@\u0015'@@\u0013@%@@=-@#3/v<@G@@ \u001bv@\u0017@\u001b@5,@@;@@j\u0
  "@\t@Pp@@V{@@\u0013Çm@\u0012Ř@@\tk@@\u0012@z@$M@\u000fm@\"@q@n@@&xB*0{@lVd=@v\u00048@@|@fZb@@01rY\u0000W\u000e@@IU\u0002Sv@\u001b@-D{r \
  "^(@v@@+@\u000f@4i @@@@@n\u001f@\u001e@5S@@NON(@O\f\u0003@bh\b.5@`t@ÓZy@Hn@d@K@@0]@@\u001c\u000f\\xW@@>lk@@@@]@I@@kQ@1@@m=@@5sa
  "@@@`@@\u0004@:@X\u0004@@L(@b@\u000f\bZ\u000f@^@W@@\u001bxk@W@@@\u00186\\q@@5\u001d@\u0013@ŧ@@f\u0010@x@\b@\u0005@@@+@\tz@@ö@rk=H2b
  "@\u0018.\u0015;@,h6@@7@\u0003&@D\u0013@\t(\u000eu@@\\\u0017@@\u0013f@#9@w@@@9@@@51\r@\u0003s@n h\u0017\\;@m@0@@) @2@xE:@@Hr\u0002@\u001c
  "a@@^@,\u0003\u0007@T@r@\u003K8@[d@@\u001d\u0004\u0013`T @ddr\u000e@@@X 4@\u0019,yT@@*p@@@)v]=B@@@@*@@\u000b@@]@@@@@\u000e/#",
```


# BIBLIOGRAFIA

MDN Web Docs (Mozilla Developer Network) Aplicación: Para comprender el empleo de métodos contemporáneos de JavaScript, como map() y Array.from(), que posibilitan la manipulación de conjuntos de elementos del DOM, así como para corroborar las API para acceder a los nodos del DOM (parentElement, nextElementSibling), reviso la documentación oficial. También examino los métodos básicos de manejo de arreglos (slice, splice, push y unshift) que utilizo en el ejercicio1().
Enlace: https://developer.mozilla.org/es/docs/Web/JavaScript

Uso de W3Schools: Para afianzar la sintaxis y el funcionamiento de los selectores que empleo para encontrar los elementos estructurales (listas ul/li, contenedores div y encabezados h2) en el código HTML, repaso las guías sobre manipulación del DOM y navegación entre nodos.
Enlace: https://www.w3schools.com/js/js_htmldom.asp

Uso de recursos audiovisuales (YouTube): Para entender cómo se aplica la API File de HTML5, la interfaz FileReader y el método readAsText(), que posibilitan la lectura de archivos locales seleccionados por el usuario sin requerir un servidor, veo este video.
Enlace: https://www.youtube.com/watch?v=3MG-fhHcNTM