Explicación par Tarea 3, de la UNIDAD 3 del Módulo de DWEC.

Alumno: Celestino F. Amigo

Actividad 1. Objetos predefinidos

Realiza las siguientes operaciones utilizando los diferentes objetos predefinidos:

- **1.1.** Objetos predefinidos **Window y Screen**:
    - Diseña una función que al llamarla muestre en una etiqueta H1 el tamaño de tu pantalla, el tamaño del navegador y el tamaño útil de la página web.
    - Haz que el valor se actualice automáticamente cada segundo utilizando la función **setInterval()**.

Solución:

Para ello creamos primero la parte de HTML con la etiqueta:

```
    <h1 id="datos_navegador"></h1>
```

Y ahora vamos con la parte del script:

Para la pantalla, que siempre mide lo mismo:

```
<script>

        //  Dimensiones de la PANTALLA. NO VARÍAN.
        const altura_pantalla = window.screen.availHeight; // Altura disponible de la pantalla (excluye barra de tareas)
        const anchura_pantalla = window.screen.availWidth; // Ancho disponible de la pantalla (excluye barra de tareas)
```

Como el resto de dimensiones sí que varían al hacer más o menos pequeño el explorador, creamos una función que actualice esos valores:

```
function muestraMensaje () {

        //Almacenamos el encabezado en una constante para modificar más tarde su contenido
        const datos_navegador=document.getElementById("datos_navegador");

        // Como window es el objeto por defecto, en este caso voy a usarlo. En el siguiente no, para que se vea.
        //  Dimensiones del NAVEGADOR (Ventana Completa)
        const altura_navegador = window.outerHeight; // Altura total del navegador, incluyendo bordes y barras de
herramientas
        const anchura_navegador = window.outerWidth;  // Ancho total del navegador, incluyendo bordes y barras de
herramientas

        //Sin uso de window por defecto.
        //  Dimensiones del VIEWPORT (Espacio Útil para el Contenido)
        const altura_utilnavegador = innerHeight; // Altura del área de contenido (viewport)
        const anchur_utilnavegador = innerWidth;  // Ancho del área de contenido (viewport)

        datos_navegador.innerHTML=`La altura de la pantalla es ${altura_pantalla} px, la anchura es ${anchura_pantalla}
px.<br>
        La altura del navegador (con bordes) es de ${altura_navegador} px, y su anchura ${anchura_navegador} px.<br>
        El espacio útil en el navegador es de ${altura_utilnavegador} px de altura, y ${anchur_utilnavegador} px de
anchura.`;
        }
```

Por último establecemos que se llame a la función cada 1000ms (1 segundo):

```
//lanzamos el temporizador que lanza la funcion cada 1 segundo.
        setInterval(muestraMensaje,1000 );
```

- **1.2.** Objetos predefinidos **Location e History:**
    - Muestra la URL de la página actual.
    - Haz un botón que al pulsarlo cambie la página actual por la de Google. Debe permitir volver a la página anterior con el botón "ir a página anterior" del navegador.
    - Haz un botón que al pulsarlo cambie la página actual por la de Yahoo. En este caso, al pulsar el botón "ir a página anterior" del navegador no se debe volver a la página de la tarea.

Primero creamos las etiquetas en el HTML.

```
<p id="textoUrl"></p>
    <!--Botones para lanzar la función-->
    <button type="button" onclick="iraGoogle()" id="iraGoogle">Google</button>
    <button type="button" onclick="iraYahoo()" id="iraYahoo">Yahoo</button>
```

Para ver la localización en la que estamos:

```
 //URL de la web
        const url=window.location;
        //Obtención del elemento donde mostramos la información.
        const infoURL=document.getElementById("textoUrl");
        infoURL.innerHTML=`La URL de la web es: ${url}`;
```

Función para el botón de Google. Simplemente con window.location.href:

```
function iraGoogle(){
            //Propiedad href. Le estamos diciendo cual es el location que tiene.
            window.location.href="https://www.google.com/";
        }
```

Función para Yahoo, olvidando la página anterior: Simplemente con window.location.replace

```
function iraYahoo() {
            //Método. En este caso se borra el historial no permitiendo volver.
            window.location.replace("https://es.search.yahoo.com/");
        }
```

- **1.3.** Objetos predefinidos **navigator**:
    - Muestra el sistema operativo y el navegador del usuario.
    - Además, debe mostrar si se ha accedido a la web desde un móvil o no.

Mejor mirar el ejercicio. Los métodos de la lección están deprecated y busqué otra manera de hacerlo:

```
   <p id="info"></p>
    <script>

        /*Por los apuntes del tema, tendríamos que hacer algo de este estilo:
```

```
const cliente= window.navigator.appName;
const plataforma=window.navigator.platform;

const informacion = document.getElementById("info");

informacion.innerHTML=`El navegador utilizado es ${cliente}, y la plataforma utilizada es ${plataforma}`

PERO AL ESTAR DEPRECATED, HE BUSCADO OTRA MANERA DE HACERLO MÁS CORRECTA*/



// Obtener el elemento donde se mostrará la información
const informacion = document.getElementById("info");

// --- 1. Obtener el User Agent (cadena que identifica el navegador y SO) ---
const userAgent = navigator.userAgent;

// --- 2. Determinar si es un Dispositivo Móvil ---
// Se considera móvil si el userAgent contiene ciertas palabras clave comunes.
const esMovil = /Mobi|Android|iPhone|iPad|iPod|BlackBerry|Windows Phone/i.test(userAgent);

// --- 3. Extraer información del Navegador y Sistema Operativo (Estimación) ---
let navegador = "Desconocido";
let sistemaOperativo = "Desconocido";

// Detección de Sistema Operativo
if (userAgent.includes("Win")) {
    sistemaOperativo = "Windows";
} else if (userAgent.includes("Mac")) {
```

```
        sistemaOperativo = "macOS / iOS";
    } else if (userAgent.includes("Linux")) {
        sistemaOperativo = "Linux / Android";
    } else if (userAgent.includes("X11")) {
        sistemaOperativo = "UNIX";
    }

    // Detección de Navegador
    if (userAgent.includes("Chrome") && !userAgent.includes("Edg")) {
        navegador = "Google Chrome";
    } else if (userAgent.includes("Firefox")) {
        navegador = "Mozilla Firefox";
    } else if (userAgent.includes("Edg")) {
        navegador = "Microsoft Edge";
    } else if (userAgent.includes("Safari") && !userAgent.includes("Chrome")) {
        navegador = "Apple Safari";
    }

    // --- 4. Construir el mensaje ---
    const mensajeMovil = esMovil ?
        "**SÍ** se ha accedido desde un dispositivo móvil." :
        "**NO** se ha accedido desde un dispositivo móvil.";

    informacion.innerHTML = `
        <p>
            **Navegador Identificado:** <strong>${navegador}</strong><br>
            **Sistema Operativo Estimado:** <strong>${sistemaOperativo}</strong><br>
            **Acceso Móvil:** ${mensajeMovil}
        </p>
```

```
            <p>
                User Agent Completo para referencia: ${userAgent}
            </p>
        `;


    </script>
```

Actividad 2. Date

Desarrolla una página web que utilizando el objeto **Date** permita:

- **2.1.** Introducir una fecha y mostrar si es anterior o posterior a la fecha actual.
- **2.2.** Mostrar en un input de tipo date la fecha actual más 20 días. Por ejemplo, si es el 1 de noviembre de 2023 deberá mostrar 21/11/2023. *Pista: valueAsDate*

Por un lado el HTML:

```
<form>
        <!--Input para introducir la fecha a evaluar-->
        <label for="fechaInput">Fecha</label>
        <input name="fechaInput" id="fechaUsuario" type="date" value="Fecha">
        <!--Botón que lanza la función que evalúa la fecha-->
        <button type="button" onclick="evaluaFecha()">Evalúa la fecha</button>
        <!--Aquí diremos cual es la fecha de hoy-->
        <p id="fechaHoy"></p>
        <br>
```

```
        <!--Aquí veremos si la fecha es menor o mayor a la fechaActual-->
        <h2 id="comparaFechas"></h2>
        <br>
        <br>
        <!--Aquí veremos cuanto es la fecha actual más 20 días-->
        <p id="fechasumada"></p>
        </form>
```

El Script:

Creando un objeto new Dare() tenemos la fecha de hoy. Solo tenemos que extraer los valores y almacenarlos en la variable correspondiente.

```
script>
            //Solicitamos la fecha de hoy
            const fechaHoy= new Date();

            //Fecha a día de hoy:
            //Obtenemos el día de hoy
            const diaActual=fechaHoy.getDate();
            //Los meses van de 0 a 11, por lo que hay que sumar 1 para que el mes correcto
            const mesActual=fechaHoy.getMonth()+1;
            //Obtenemos el año actual. Hay que usar getFullYear(), ya que getYear() no funciona bien (leer en internet)
            const anioActual=fechaHoy.getFullYear();

            //Constante que captura el párrafo donde se va a mostar la fecha de hoy
            const hoy=document.getElementById("fechaHoy");

            //Sacamos el contenido a la página modificando el ese párrafo
```

```
hoy.innerHTML=`La fecha actual es ${diaActual}/${mesActual}/${anioActual}`
```

Para comparar la fecha, solo tenemos que comparar la fecha de hoy del punto anterior, con la fecha introducida por el usuario. Para ello creamos nuevo objeto Date() con el string introducido por el usuario, y lo almacenamos en una nueva constante. En función del resultado, arrojaos un resultado u otro.

```
//Almacenamos el objeto del dom en el que presentaremos el resultado
const fechasComparadas=document.getElementById("comparaFechas");

function evaluaFecha(){

//Para que evalúe la fecha introducida, tienen que estar dentro de la función.
//Si no, nada más abrir la página se carga el valor en el inputo y no vuelve a cambiar.
//Almacenamos en valor de la fecha a evaluar, introducida en el input. Usamos value para almacenar el valor
//Se extrae como String
const fechaInputString=document.getElementById("fechaUsuario").value;
//Para convertirlo a formato Date, creamos un nuevo objeto.
const fechaInput= new Date(fechaInputString);


if (fechaInput<fechaHoy){
    fechasComparadas.innerHTML="La fecha introducida es menor a la fecha de Hoy"
}

//Para este else, tuve que hacerlo así, puesto que al contener las horas
//horas, minutos, segundos y milisegundos, nunca entraba a este bloque.
else if (fechaHoy.getDate()==fechaInput.getDate()&&
fechaHoy.getMonth()==fechaInput.getMonth()&&
fechaHoy.getFullYear()==fechaInput.getFullYear()){
```

```
            fechasComparadas.innerHTML="La fecha introducida es igual a la de hoy."
        }
         else {
            fechasComparadas.innerHTML="La fecha introducida es superior a la fecha de hoy"
        }
    }

    //Para el apartado 2.2., tenemos que aumentar la fecha actual en 20 días.
```

Para el apartado 2.2. Solo tenemos que sumar 20 días al día de hoy. Seleccionamos con getDate() que se sume a los días, y js automáticamente detecta la fecha correcta.

```
        const fechaAumentadaTexto=document.getElementById("fechasumada");

        let hoyMas20 = new Date();
        hoyMas20.setDate(hoyMas20.getDate()+20);

        let diaMas20=hoyMas20.getDate();
        let mesMas20=hoyMas20.getMonth()+1;
        let anioMas20=hoyMas20.getFullYear();

    fechaAumentadaTexto.innerHTML=`La fecha de hoy más 20 días es ${diaMas20}/${mesMas20}/${anioMas20}`;

    </script>
```

Actividad 3. String

Desarrolla una página que dada una cadena de texto, introducida mediante un área de texto, utilizando las funcionalidades de **String** permita calcular:

- La longitud del texto en caracteres.
- La longitud del texto en palabras (se asume que están separadas todas por un espacio " ").
- El número de frases introducidas (se asume que todas las frases están separadas por un punto ".").
- Devuelve los primeros 10 caracteres del texto introducido en mayúscula.
- El número de apariciones de la palabra "alumno" o "alumna" en el texto. Debe ser robusto al uso de mayúsculas.
- Remplazar todas las apariciones de la palabra "alumno" o "alumna" por tu nombre.

Primero, el HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <textarea id="textoEvaluado" rows="10" cols="80"></textarea>
    <br>
    <div>
```

```
    <button type="button" id="longitud" onclick="compLongitud()">Apartado 3.1. Longitud</button>
    <button type="button" id="palabras" onclick="compPalabras()">Apartado 3.2. Num palabras</button>
    <button type="button" id="frases" onclick="compFrases()">Apartado 3.3. Num frases</button>
    <button type="button" id="mayus" onclick="diezEnMayus()">Apartado 3.4. Mayúsculas</button>
    <button type="button" id="busca" onclick="buscaPalabra()">Apartado 3.5. Busca palabras</button>
    <button type="button" id="busca" onclick="todasJuntas()">LANZA TODO A LA VEZ</button>
    </div>
    <div>
    <p id="p1">Apartado 3.1</p>
    <p id="p2">Apartado 3.2</p>
    <p id="p3">Apartado 3.3</p>
    <p id="p4">Apartado 3.4</p>
    <p id="p4b">Apartado 3.4b</p>
    <p id="p5">Apartado 3.5</p>
    </div>
    <script src="../js/Apartado 3.js"></script>
</body>
</html>
```

Ahora vamos al js:

1º.

- Creamos una función que obtenga del DOM el texto introducido por el usuario y lo almacenamos en una constante, para poder utilizarlo en el resto de funciones de la aplicación:

```
/**Función que obtiene el texto del DOM
     * Obtiene el valor actual del contenido del elemento <textarea> con el ID "textoEvaluado",
    * eliminando cualquier espacio en blanco al inicio o al final.
```

```
    * * @returns {string} El contenido de la textarea.
    */
        function obtenerTexto() {
        // Usa .trim() para evitar contar espacios iniciales/finales
        const elementoTexto=document.getElementById("textoEvaluado");
        return elementoTexto.value;
        }
```

- Creamos otra función que será utilizada par mostrar el resultado:

```
/** Función que actualiza el contenido de texto de un elemento HTML específico por su ID.
    * @param id - El ID del elemento HTML (p. ej., "p1", "p5") cuyo texto se va a modificar.
    * @param  mensaje - El nuevo contenido de texto que se mostrará dentro del elemento.
    * @return no tiene valor de retorno
    */
    function mostrarResultado(id, mensaje) {
        document.getElementById(id).innerText = mensaje;
    }
```

2º - Funciones solicitadas por el ejercicio.

a) Longitud del texto:

```
// Apartado 3.1. Longitud
    function compLongitud() {
        const textoActual = obtenerTexto();
        //Longitud total de la cadena, almacenada en const longTexto.
        const longTexto = textoActual.length;
```

```
        //Lleva el resultado al HTML.
        mostrarResultado("p1", `Apartado 3.1. El tamaño del texto es ${longTexto}`);
    }
```

b) Número de palabras:

```
// Apartado 3.2. Número de Palabras
    function compPalabras() {
        const textoActual = obtenerTexto();
        // Array de elementos separado por espacios.
        const palabras = textoActual.split(" ");
        //Cuenta la cantidad de elementos
        const numPalabras = palabras.length;

        //llevarlo al HTML
        mostrarResultado("p2", `Apartado 3.2. El número de palabras es: ${numPalabras}`);
    }
```

c) Número de frases:

```
// Apartado 3.3. Número de Frases (Lógica Original: split("."))
    function compFrases() {
        const textoActual = obtenerTexto();
        // Almacena en el array el número de frases, separadas por punto.
        const frases = textoActual.split(".");
        // Cuenta el número de elementos en el array.
        const numFrases = frases.length;
```

```
        //llevarlo al HTML
        mostrarResultado("p3", `Apartado 3.3. El número de frases es: ${numFrases}`);
    }
```

d) 10 primeros caracteres en mayúsculas, dos formas:

```
// Apartado 3.4. Mayúsculas
    function diezEnMayus() {
        const textoActual = obtenerTexto();

        // 3.4.a: Primeros 10 caracteres (incluyendo espacios) a mayúsculas
        const carMayus = textoActual.substring(0, 10).toUpperCase();

        // 3.4.b: Primeros 10 caracteres (SIN espacios) a mayúsculas. (miré en internet como quitarlos)
        const textoActualCompactado = textoActual.replace(/\s/g, "");
        const carMayusb = textoActualCompactado.substring(0, 10).toUpperCase();

        mostrarResultado("p4", `Apartado 3.4.a. Los diez primeros caracteres mayúsculos (con espacios) son:
${carMayus}`);
        mostrarResultado("p4b", `Apartado 3.4.b. Los diez primeros caracteres mayúsculos (sin espacios) son:
${carMayusb}`);
    }
```

e) Búsqueda y reemplazo de las palabras alumno/alumna, por mi nombre:

```
 // Apartado 3.5. Búsqueda de Palabras
```

```
function buscaPalabra() {
    //Se obtiene el texto, como antes
    const textoActual = obtenerTexto();
    const nombreReemplazo = "Celestino";

    // Función auxiliar de conteo (no llamada al DOM)
    function contarAlumnado(texto) {
        const textoEnMinusculas = texto.toLowerCase();
        // Contamos ocurrencias de "alumno" y "alumna".
        const conteoAlumno = textoEnMinusculas.split("alumno").length - 1; //explicación del -1 abajo, que no lo veía
        const conteoAlumna = textoEnMinusculas.split("alumna").length - 1;
        return conteoAlumno + conteoAlumna;

        // Ojo, como vamos a separar alumno y alumna de la cadena, por ejemplo en en caos de alumno:
        // Como vamos a "cortar alumno del array del resto del texto, el texto resultante será igual
        // al número de alumnos más uno. Por ej: let var1="Soy un alumno nuevo"
        // let conteo=var1.split("almumno"). conteo sería igual a  ["Soy un", "nuevo"] (2, en vez de 1).
    }

    // Función auxiliar de reemplazo (no llamada al DOM)
    function reemplazarAlumnado(texto, reemplazo) {
        // Usamos RegEx con bandera 'gi' (global e insensible)
        const regex = /alumn[oa]/gi;
        return texto.replace(regex, reemplazo);
    }

    const conteo = contarAlumnado(textoActual);
    const textoReemplazado = reemplazarAlumnado(textoActual, nombreReemplazo);
```

```
        let mensaje = `Apartado 3.5. En total se encontraron ${conteo} referencias a "alumno/a".\n\n
        Texto original con reemplazo (${nombreReemplazo}):\n${textoReemplazado}`;

        mostrarResultado("p5", mensaje);
    }
```

f) Función que lanza todas las funciones a la vez:

```
//Lanzar todas las funciones juntas.
    function todasJuntas(){

    compLongitud();
    compPalabras();
    compFrases();
    diezEnMayus();
    buscaPalabra();
```

Actividad 4. Almacenar lista de nombres

Crea una página que permita añadir nombres a una lista con un input. Esta lista debe mostrarse en la página Web. Además, debe:

- La lista guardarse en memoria y mantenerse aunque se cierre la página y se vuelva a abrir en una nueva pestaña.
- Se debe poder vaciar la lista con un botón.

Primero el HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Actividad 4: Lista Simple con LocalStorage</title>
</head>
<body>

    <h1>Gestor de Nombres</h1>

    <div>
        <input type="text" id="inputNombre" placeholder="Nombre a añadir">
        <button id="btnAñadir">Añadir</button>
        <button id="btnLimpiar">Limpiar Todo</button>
    </div>

    <hr>

    <h2>Lista de Nombres</h2>
```

```
    <ul id="listaUsuario"> </ul>

    <script src="../js/Apartado 4.js"></script>

</body>
</html>
```

Y ahora el JS:

1º - Fijamos las constantes que vamos a usar par vincular a los elementos del DOM:

```
   //--- 1. CONSTANTES ---
        // Referencias a los elementos del HTML (la "interfaz de usuario").
        const inputLista = document.getElementById('inputNombre');
        const listaMostrar = document.getElementById('listaUsuario');
        const btnAnadir = document.getElementById('btnAñadir');
        const btnLimpiar = document.getElementById('btnLimpiar');
```

2º - Nombre que vamos a utilizar para localstorage:

```
       // Nombre clave único para guardar y recuperar la lista en la memoria local del navegador.
        const CLAVE_ALMACENAMIENTO = 'miListaDeNombres';
```

3º - Cargar las funciones de inicio con la carga del DOM:

```
 //----INICIAR LAS FUNCIONES DE LA WEB AL CARGAR EL DOM ----
        /**Configura la aplicación.*/
        function iniciarApp() {
            // A. Carga los nombres guardados (localstorage) inmediatamente al abrir la página.
            mostrarLista();

            // B. Asigna las funciones a los botones (Event Listeners).
            btnAnadir.addEventListener('click', añadir);//recordar que al ir en addEventListener siempre va sin (), que
se me olvida y no funciona.
            btnLimpiar.addEventListener('click', limpiar);
        }

        // CASI LO MÁS IMPORTANTE. SE ME OLVIDA. Espera a que la página HTML esté completamente cargada antes de iniciar
la app.
        document.addEventListener('DOMContentLoaded', iniciarApp);
```

4º - Obtener (si la hay) la lista con los datos guardados en el lavegador:

Si hay una lista con datos, al estar en JSON habrá que transformarlos a array. (JSON.parse)

```
// --- 2. PERSISTENCIA DE DATOS (localStorage) ---

        /**Carga la lista de nombres desde localStorage, si la hay.
         * @returns {Array} Un array de nombres (o un array vacío si no hay nada guardado).*/

        function obtenerLista() {
            // localStorage.getItem() recupera los datos como una cadena JSON.
```

```
            const listaJSON = localStorage.getItem(CLAVE_ALMACENAMIENTO);

              let lista; // Variable para almacenar el resultado final.

            // Comprobamos si listaJSON tiene un valor (no es null, undefined, o una cadena vacía)
            if (listaJSON) {
            // Si hay datos (la condición es verdadera), los convertimos de JSON a Array.
            lista = JSON.parse(listaJSON);
            } else {
            // Si no hay datos (la condición es falsa, generalmente porque es null), inicializamos con un Array vacío.
            lista = [];
            }

            return lista; //retorna la lista, que será un array en cualquier caso
        }
```

5º- Función que guarda datos en la lista: Al guardar los datos, hay que pasarlos de array a JSON (con stringify)

```
/**Guarda el array de nombres actual en localStorage.
         * @param {Array} lista - El array de nombres a guardar.*/

        function guardarLista(lista) {
            // JSON.stringify() convierte el Array en una cadena JSON para poder guardarse.
            const listaParaAlmacenar = JSON.stringify(lista);

            // Almaceno los datos en local. Podría haberlo hecho todo en un paso, pero así me parece más
            // claro paso a paso. localStorage.setItem(CLAVE_ALMACENAMIENTO, JSON.stringify(lista) );
            localStorage.setItem(CLAVE_ALMACENAMIENTO, listaParaAlmacenar );
        }
```

6º- Función de vista que muestra el HTML:

```
// --- 3. FUNCIONES DE VISTA (Interactúan con el HTML) ---

   /**Actualiza la lista en el HTML, construyendo todos los <li> como una cadena de texto.
   * Usa el método .forEach() y la propiedad innerHTML para insertar el HTML de golpe.*/

   function mostrarLista() {
       // Obtiene la lista de nombres del almacenamiento (que es un Array de JS).
       const listaGuardada = obtenerLista();

       // Inicializamos una variable para construir la cadena HTML de todos los <li>.
       let contenidoHTML = '';

       // RECORRIDO USANDO .forEach()
       // Por cada 'nombre' en el array, ejecutamos el código.
       listaGuardada.forEach(nombre => {
           // Concatenamos (añadimos) a la cadena 'contenidoHTML' la estructura <li>...</li>
           // Se crean manualmente las etiquetas HTML como parte del texto.
           contenidoHTML += `<li>${nombre}</li>`;
       });

       // NSERCIÓN EN EL DOM (El reemplazo de createElement/appendChild)

       // listaMostrar (que es la referencia al <ul>) limpia y reemplaza su contenido
       // inyectando la cadena de texto HTML que acabamos de construir.
       listaMostrar.innerHTML = contenidoHTML;
   }
```

7º- Función para el botón Añadir:

```
// --- 4. FUNCIONES DE EVENTO (BOTONES) ---

        /**Se ejecuta al pulsar "Añadir"*/
        function añadir() {
            //El valor de la lista sin espacios
            const nuevoNombre = inputLista.value.trim();

            if (nuevoNombre) {
                // 1. Cargar: Obtiene la lista actual.
                const lista = obtenerLista();

                // 2. Modificar: Añade el nuevo nombre al array (el modelo de datos).
                lista.push(nuevoNombre);

                // 3. Guardar: Persiste el array modificado en localStorage.
                guardarLista(lista);

                // 4. Actualizar: Dibuja la nueva lista en la pantalla.
                mostrarLista();

                // 5. Limpiar: Deja el input vacío para el siguiente nombre.
                inputLista.value = '';
            } else {
                alert("No has escrito nada");
            }
        }
```

8º - Función para limpiar todo:

```
/** Se ejecuta al pulsar "Limpiar Todo".*/
        function limpiar() {
            if (confirm("¿Seguro que quieres borrar TODOS los nombres guardados?")) {
                // localStorage.removeItem() borra permanentemente la clave.
                localStorage.removeItem(CLAVE_ALMACENAMIENTO);

                // La vista se actualiza, mostrando ahora una lista vacía.
                mostrarLista();
            }
        }
```