Universidad Tecnológica Metropolitana

DIVISIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN

División Tecnologías de la Información y Comunicación

Coordinación de Servicios de Tecnologías de la Información y Comunicación

Universidad Tecnológica Metropolitana

4F

ACTIVIDAD # 1-4

Nombre alumno:
Fernando David Sanchez Sacnhez

Nombre del profesor(a): Ruth Dominguez

Fecha de entrega: Mérida, Yucatán a martes09 de octubre de 2024

Practica 1.

1. Declaración de Variables Globales

El código comienza definiendo dos arreglos vacíos para almacenar los productos disponibles y los productos retirados:

```
// Variables para almacenar los productos disponibles y
let productosDisponibles = [];
let productosRetirados = [];
```

- productosDisponibles: Almacena los productos que actualmente están disponibles.
- productosRetirados: Almacena los productos que han sido retirados de la lista de disponibles.

2. Función agregarProducto

Esta función se encarga de agregar un nuevo producto a la lista de productos disponibles. A continuación, se detalla cada paso:

```
// Función para agregar productos a la lista
function agregarProducto() {
    const nombreProducto = `Producto${productosDisponibles.length + 1}`;
    const cantidad = Math.floor(Math.random() * 20) + 1; // Cantidad aleatoria entre 1 y 20
    const precio = (Math.random() * 50 + 1).toFixed(2);  // Precio aleatorio entre 1 y 50

    const nuevoProducto = {
        id: Date.now(), // ID único basado en el tiempo actual
        nombre: nombreProducto,
        cantidad: cantidad,
        precio: precio
    };

    productosDisponibles.push(nuevoProducto);
    actualizarUI();
}
```

Paso 1: Genera un nombre de producto automático (Producto1, Producto2, etc.), basado en la longitud del arreglo productosDisponibles.

```
    const nombreProducto = `Producto${productosDisponibles.length + 1}`;
```

Paso 2: Genera una **cantidad aleatoria** entre 1 y 20 para el producto.

```
const cantidad = Math.floor(Math.random() * 20) + 1; // Cantidad aleat
```

Paso 3: Genera un **precio aleatorio** entre 1 y 50 con dos decimales.

```
const precio = (Math.random() * 50 + 1).toFixed(2);
```

Paso 4: Crea un objeto nuevoProducto con las siguientes propiedades:

- id: ID único basado en el tiempo actual (Date.now()).
- nombre: Nombre generado del producto.
- cantidad: Cantidad aleatoria.
- precio: Precio aleatorio.

```
const nuevoProducto = {
    id: Date.now(), // ID único basado en el tiempo actual
    nombre: nombreProducto,
    cantidad: cantidad,
    precio: precio
};
```

Paso 5: Añade el nuevo producto a la lista productosDisponibles y llama a actualizarUI para reflejar los cambios en la interfaz.

```
    productosDisponibles.push(nuevoProducto);
    actualizarUI();
}
```

3. Función eliminarProducto

Esta función elimina un producto de la lista de productosDisponibles y lo mueve a la lista de productosRetirados usando su ID como referencia.

```
function eliminarProducto(id) {
    // Buscar el índice del producto con el ID proporcionado
    const index = productosDisponibles.findIndex(producto => producto.id === id);

    if (index !== -1) {
        // Mover el producto a la lista de retirados
        productosRetirados.push(productosDisponibles[index]);
        // Eliminar el producto de la lista de disponibles
        productosDisponibles.splice(index, 1);
        actualizarUI();
    }
}
```

4. Función actualizarUI

Esta función actualiza la interfaz de usuario para reflejar la lista de productos disponibles y retirados, así como las cantidades y precios.

```
function actualizarUI() {
    const listaProductos = document.getElementById('listaProductos');
    const listaProductosRetirados = document.getElementById('listaProductosRetirados');
    const totalProductos = document.getElementById('totalProductos');
    const productosRetiradosElem = document.getElementById('productosRetirados');

    // Actualizar lista de productos disponibles
    listaProductos.innerHTML = '';
    productosDisponibles.forEach(producto => {
        const li = document.createElement('li');
        li.innerHTML = `<span>${producto.nombre}</span> - Cantidad: ${producto.cantidad} - Precio: $${producto.precio}
                        <button onclick="eliminarProducto(${producto.id})">Retirar</button>`;
        listaProductos.appendChild(li);
    });

    // Actualizar lista de productos retirados
    listaProductosRetirados.innerHTML = '';
    productosRetirados.forEach(producto => {
        const li = document.createElement('li');
        li.innerHTML = `<span>${producto.nombre}</span> - Cantidad: ${producto.cantidad} - Precio: $${producto.precio}`;
        listaProductosRetirados.appendChild(li);
    });

    // Actualizar contadores de productos
    totalProductos.textContent = `Productos disponibles: ${productosDisponibles.length}`;
    productosRetiradosElem.textContent = `Productos retirados: ${productosRetirados.length}`;
}
```

5. Inicialización de la Interfaz

Por último, el código llama a actualizarUI() para inicializar la interfaz al cargar la página:

```
// Inicializar la interfaz de usuario al cargar la página
actualizarUI();
```

Practica 2.

1. Función generarNumeros

Esta función es la encargada de generar los 20 números aleatorios y separarlos en números pares e impares, para luego actualizar la interfaz de usuario.

```
function generarNumeros() {
    const numerosAleatorios = [];

    // Generar 20 números aleatorios y agregarlos a la lista
    for (let i = 0; i < 20; i++) {
        const numero = Math.floor(Math.random() * 100) + 1; // Generar número aleatorio entre 1 y 100
        numerosAleatorios.push(numero);
    }

    // Separar la lista en pares e impares
    const numerosPares = numerosAleatorios.filter(numero => numero % 2 === 0);
    const numerosImpares = numerosAleatorios.filter(numero => numero % 2 !== 0);

    // Actualizar la interfaz de usuario
    mostrarNumeros(numerosPares, 'columnaPares');
    mostrarNumeros(numerosImpares, 'columnaImpares');
}
```

2. Función mostrarNumeros

Esta función se encarga de mostrar una lista de números en un contenedor HTML específico, que es indicado por el parámetro columnaId.

```
// Función para mostrar los números en la columna correspondiente
function mostrarNumeros(listaNumeros, columnaId) {
    const columna = document.getElementById(columnaId);
    columna.innerHTML = ''; // Limpiar la lista

    // Agregar cada número como un elemento de lista
    listaNumeros.forEach(numero => {
        const li = document.createElement('li');
        li.textContent = numero; // Asignar el número al elemento de lista
        columna.appendChild(li);
    });
}
```

3. Estructura Esperada del HTML

Para que este código funcione correctamente, se deben definir dos columnas en el archivo HTML con los id columnaPares y columnaImpares. Un ejemplo de cómo podría verse el HTML es el siguiente:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Números Pares e Impares - Modo Oscuro</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <h1>Generador de Números Pares e Impares</h1>
        <div class="buttons">
            <button onclick="generarNumeros()">Generar 20 Números</button>
        </div>

        <div class="number-list">
            <h2>Números Generados</h2>
            <div class="column-container">
                <div class="column">
                    <h3>Números Pares</h3>
                    <ul id="columnaPares"></ul>
                </div>
                <div class="column">
                    <h3>Números Impares</h3>
                    <ul id="columnaImpares"></ul>
                </div>
            </div>
        </div>
    </div>

    <script src="app.js"></script>
</body>
</html>
```

Practica 3.

1. Declaración de Listas Globales

El código comienza con la creación de dos arreglos vacíos para almacenar los alumnos aprobados y los reprobados:

```
// Listas para almacenar los alumnos aprobados y reprobados
let listaAprobados = [];
let listaReprobados = [];
```

Paso 2: Valida los datos ingresados. Si el nombre está vacío o la calificación no es un número válido o está fuera del rango de 0 a 10, muestra un mensaje de alerta y termina la función.

```
// Validar que el nombre y la calificación no estén vacíos y que la calificación esté en
if (nombre === '' || isNaN(calificacion) || calificacion < 0 || calificacion > 10) {
    alert('Por favor ingresa un nombre válido y una calificación entre 0 y 10.');
    return;
}
```

Paso 3: Clasifica al alumno según su calificación:

- Si la calificación es igual o mayor a 7, se agrega a listaAprobados.
- Si la calificación es menor a 7, se agrega a listaReprobados

```
// Clasificar al alumno como aprobado o reprobado
if (calificacion >= 7) {
    listaAprobados.push({ nombre, calificacion });
} else {
    listaReprobados.push({ nombre, calificacion });
}
```

Paso 4: Limpia los campos de entrada de texto (nombre y calificacion) después de agregar el alumno.

```
// Limpiar los campos de entrada
document.getElementById('nombre').value = '';
document.getElementById('calificacion').value = '';
```

Paso 5: Llama a la función actualizarUI para mostrar las listas de aprobados y reprobados en la interfaz de usuario.

```
    // Actualizar la inte
    actualizarUI();
}
```

3. Función actualizarUI

Esta función es la encargada de actualizar la interfaz de usuario para mostrar las listas de alumnos aprobados y reprobados en sus respectivos contenedores.

```
// Función para actualizar la lista de aprobados y reprobados en la interfaz
function actualizarUI() {
    const listaAprobadosElem = document.getElementById('listaAprobados');
    const listaReprobadosElem = document.getElementById('listaReprobados');

    // Limpiar las listas
    listaAprobadosElem.innerHTML = '';
    listaReprobadosElem.innerHTML = '';

    // Mostrar lista de aprobados
    listaAprobados.forEach(alumno => {
        const li = document.createElement('li');
        li.classList.add('aprobado'); // Asignar estilo de aprobado
        li.textContent = `${alumno.nombre} - Calificación: ${alumno.calificacion}`;
        listaAprobadosElem.appendChild(li);
    });

    // Mostrar lista de reprobados
    listaReprobados.forEach(alumno => {
        const li = document.createElement('li');
        li.classList.add('reprobado'); // Asignar estilo de reprobado
        li.textContent = `${alumno.nombre} - Calificación: ${alumno.calificacion}`;
        listaReprobadosElem.appendChild(li);
    });
}
```

Practica 4.

1. Declaración de Listas Globales

El código comienza con la declaración de dos arreglos vacíos para almacenar los productos disponibles y los productos que han sido eliminados.

```
// Listas para almacenar los productos y los eliminados
let productos = [];
let productosEliminados = [];
```

2. Función generarCodigoUnico

Esta función genera un código numérico único de 4 dígitos aleatorios para cada producto.

```
// Función para generar un código único de al menos 3 dígitos
function generarCodigoUnico() {
    return Math.floor(Math.random() * 9000) + 1000; // Genera un
}
```

3. Función agregarProducto

Esta función permite agregar un nuevo producto a la lista de productos con un nombre y un precio proporcionado por el usuario.

```
// Función para agregar un producto a la lista
function agregarProducto() {
    const nombre = document.getElementById('nombreProducto').value.trim();
    const precio = parseFloat(document.getElementById('precioProducto').value);

    // Validar que el nombre y el precio no estén vacíos y sean válidos
    if (nombre === '' || isNaN(precio) || precio <= 0) {
        alert('Por favor ingresa un nombre y un precio válidos.');
        return;
    }

    // Generar un código único para el producto
    let codigo;
    do {
        codigo = generarCodigoUnico();
    } while (productos.some(producto => producto.codigo === codigo)); // Asegurarse d

    // Crear un nuevo producto y agregarlo a la lista
    const nuevoProducto = {
        codigo: codigo,  // Asignar el código único
        nombre: nombre,
        precio: precio
    };

    productos.push(nuevoProducto);

    // Limpiar los campos de entrada
    document.getElementById('nombreProducto').value = '';
    document.getElementById('precioProducto').value = '';

    // Actualizar la lista y el costo total
    actualizarUI();
}
```

4. Función eliminarProducto

Esta función permite eliminar un producto de la lista de productos usando su código único y moverlo a la lista de productosEliminados.

```
function eliminarProducto() {
    const codigo = parseInt(document.getElementById('codigoEliminar').value);

    // Buscar el producto y moverlo a la lista de eliminados
    const index = productos.findIndex(producto => producto.codigo === codigo);
    if (isNaN(codigo) || index === -1) {
        alert('Por favor ingresa un código de producto válido.');
        return;
    }

    // Mover el producto eliminado a la lista de productos eliminados
    productosEliminados.push(productos.splice(index, 1)[0]);

    document.getElementById('codigoEliminar').value = '';
    actualizarUI();
}
```

5. Función restaurarProducto

Esta función permite restaurar un producto eliminado a la lista principal de productos usando su código único.

```
// Función para restaurar un producto de la lista de eliminados a la lista principal
function restaurarProducto(codigo) {
    const index = productosEliminados.findIndex(producto => producto.codigo === codigo);

    // Mover el producto de nuevo a la lista principal
    productos.push(productosEliminados.splice(index, 1)[0]);
    actualizarUI();
}
```

.

6. Función ordenarPorNombre

Esta función permite ordenar la lista de productos por su nombre en orden alfabético.

```
// Función para ordenar los productos por nombre
function ordenarPorNombre() {
    productos.sort((a, b) => a.nombre.localeCompare(b.nombre));
    actualizarUI();
}
```

7. Función actualizarUI

Esta función se encarga de actualizar la interfaz de usuario para mostrar las listas de productos y productos eliminados, así como calcular el costo total de los productos actuales.

```
// Función para actualizar la interfaz de usuario
function actualizarUI() {
    const listaProductosElem = document.getElementById('listaProductos');
    const listaEliminadosElem = document.getElementById('listaEliminados');
    const costoTotalElem = document.getElementById('costoTotal');

    listaProductosElem.innerHTML = '';
    listaEliminadosElem.innerHTML = '';

    // Mostrar productos actuales
    productos.forEach(producto => {
        const li = document.createElement('li');
        li.textContent = `Código: ${producto.codigo} - ${producto.nombre} - Precio: $${producto.precio.toFixed(2)}`;
        listaProductosElem.appendChild(li);
    });

    // Mostrar productos eliminados
    productosEliminados.forEach(producto => {
        const li = document.createElement('li');
        li.textContent = `Código: ${producto.codigo} - ${producto.nombre} - Precio: $${producto.precio.toFixed(2)}`;
        const restoreButton = document.createElement('button');
        restoreButton.textContent = 'Restaurar';
        restoreButton.onclick = () => restaurarProducto(producto.codigo);
        li.appendChild(restoreButton);
        listaEliminadosElem.appendChild(li);
    });

    // Calcular el costo total
    const costoTotal = productos.reduce((total, producto) => total + producto.precio, 0);
    costoTotalElem.textContent = `$${costoTotal.toFixed(2)}`;
}
```

8. Inicialización

Llama a actualizarUI() para actualizar la interfaz al cargar la página.

```
// Inicializar la interfaz de usuario
actualizarUI();
```