GUÍA TEÓRICO-PRÁCTICA:

UNIDAD I

SEMANA 1: FUNDAMENTOS DE JAVASCRIPT

TEMA 1.1: INTRODUCCIÓN A JAVASCRIPT - VARIABLES, TIPOS DE DATOS Y OPERADORES

Teoría

¿Qué es JavaScript? JavaScript es un lenguaje de programación interpretado, dinámico y orientado a objetos que permite agregar interactividad a las páginas web. Fue creado por Brendan Eich en 1995.

Variables: Una variable es un contenedor para almacenar datos. En JavaScript existen tres formas de declarar variables:

```
1  var nombre = "Juan";      // Declaración antigua (ámbito de función)
2  let edad = 25;            // Declaración moderna (ámbito de bloque)
3  const PI = 3.1416;        // Constante (no se puede reasignar)
```

Tipos de Datos:

- **String (Cadena):** "Hola Mundo", 'JavaScript'
- **Number (Número):** 25, 3.14, -10
- **Boolean (Booleano):** true, false
- **Undefined:** Variable declarada sin valor
- **Null:** Ausencia intencional de valor
- **Object:** Colección de propiedades
- **Array:** Lista ordenada de valores

Pregunta de reflexión 1: ¿Cuál es la diferencia entre let y const? ¿Cuándo usarías cada una?

Operadores:

- **Aritméticos:** + (suma), - (resta), * (multiplicación), / (división), % (módulo), ** (potencia)

- **Asignación:** =, +=, -=, *=, /=
- **Comparación:** == (igualdad), === (igualdad estricta), !=, !==, >, <, >=, <=
- **Lógicos:** && (AND), || (OR), ! (NOT)

Ejemplo

javascript

```javascript
// Declaración de variables
let nombre = "María";
let edad = 28;
const altura = 1.65;

// Operaciones aritméticas
let suma = 10 + 5;        // 15
let resta = 10 - 5;       // 5
let multiplicacion = 10 * 5; // 50
let division = 10 / 5;    // 2
let modulo = 10 % 3;      // 1 (residuo)

// Comparaciones
let esMayor = edad > 18;  // true
let esIgual = 5 == "5";   // true (comparación débil)
let esEstricto = 5 === "5"; // false (comparación estricta)
```

Ejercicio 1.1

Crea un programa que:

1. Declare variables para: tu nombre, tu edad, tu altura
2. Realice las siguientes operaciones:
    - Suma tu edad con 10
    - Multiplica tu altura por 100
    - Calcula el módulo de tu edad entre 5
3. Muestra los resultados usando console.log()

TEMA 1.2: SALIDA BÁSICA - console.log() Y alert()

Teoría

JavaScript necesita formas de mostrar información. Las dos más básicas son:

console.log(): Muestra mensajes en la consola del navegador (herramientas de desarrollador). Es ideal para depuración y no interrumpe al usuario .

javascript

```javascript
console.log("Mensaje en consola");
console.log("Valor:", variable);
console.log("Suma:", 5 + 3);
```

alert(): Muestra una ventana emergente con un mensaje. Detiene la ejecución hasta que el usuario hace clic en "Aceptar"

javascript

```javascript
alert("Hola, bienvenido");
alert("Tu resultado es: " + resultado);
```

Otras salidas útiles:

- **console.warn():** Muestra advertencias en amarillo
- **console.error():** Muestra errores en rojo
- **confirm():** Muestra cuadro de diálogo con "Aceptar" y "Cancelar"
- **prompt():** Solicita entrada de texto al usuario

Pregunta de reflexión 2: ¿Cuándo es más apropiado usar console.log() en lugar de alert()?

Ejemplo

javascript

```javascript
// Usando console.log
let nombre = "Carlos";
let edad = 30;

console.log("Nombre:", nombre);
console.log("Edad:", edad);
console.log("Tipo de dato de nombre:", typeof nombre);
console.log("Tipo de dato de edad:", typeof edad);

// Usando alert
alert("Bienvenido " + nombre);
alert("Tienes " + edad + " años");

// Combinando con prompt
let ciudad = prompt("¿En qué ciudad vives?");
console.log("El usuario vive en:", ciudad);
alert("¡Qué bonito es " + ciudad + "!");
```

Ejercicio 1.2

Crea un programa que:

1. Use prompt() para preguntar el nombre del usuario
2. Use prompt() para preguntar la edad
3. Use console.log() para mostrar ambos datos con mensajes descriptivos
4. Use alert() para dar la bienvenida al usuario con su nombre

javascript

```javascript
// Tu código aquí:


```

TEMA 1.3: CONECTAR JS CON HTML (<script>)

Teoría

Existen tres formas de incluir JavaScript en HTML:

1. JavaScript Interno (Internal): El código se escribe dentro de etiquetas <script> en el HTML .

```html
<!DOCTYPE html>
<html>
<head>
    <title>Mi Página</title>
</head>
<body>
    <h1>Hola Mundo</h1>

    <script>
        console.log("JavaScript interno");
        alert("Página cargada");
    </script>
</body>
</html>
```

2. JavaScript Externo (External): El código se escribe en un archivo .js separado y se enlaza con src .

```html
<!-- HTML -->
<script src="miarchivo.js"></script>
```

```javascript
// miarchivo.js
console.log("JavaScript externo");
```

3. JavaScript en Línea (Inline): El código se escribe directamente en los atributos HTML (no recomendado).

```html
<button onclick="alert('Click!')">Haz click</button>
```

Ubicación del script:

- **En <head>:** Se ejecuta antes de cargar el contenido (puede causar errores si busca elementos que aún no existen)
- **Al final de <body>:** Recomendado, asegura que el DOM esté cargado
- **Con atributo defer:** <script src="app.js" defer></script> (se ejecuta después de cargar el HTML)
- **Con atributo async:** <script src="app.js" async></script> (se ejecuta apenas se descarga)

Pregunta de reflexión 3: ¿Por qué es mejor colocar el script al final del body o usar defer?

Ejemplo

html

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Ejemplo Script</title>
    <!-- Script externo con defer -->
    <script src="main.js" defer></script>
</head>
<body>
    <h1 id="titulo">Bienvenido</h1>

    <!-- Script interno -->
    <script>
        console.log("El DOM está cargado");
        document.getElementById("titulo").style.color = "blue";
    </script>
</body>
</html>
```

javascript

```javascript
// main.js
console.log("Archivo externo cargado");
let mensaje = "Hola desde archivo externo";
console.log(mensaje);
```

Ejercicio 1.3

Crea un archivo HTML que:

1. Tenga un título <h1> con tu nombre
2. Incluya un script interno que cambie el color del título a rojo
3. Incluya un script externo (archivo separado) que muestre un alert() de bienvenida
4. Ambos scripts deben usar console.log() para mostrar mensajes

archivo.html:

```html
<!DOCTYPE html>
<html>
<head>
    <title>Mi Ejercicio</title>

</head>
<body>
    <h1 id="miNombre"></h1>


</body>
</html>
```

archivo.js:

```javascript
// Tu código aquí:

```

TEMA 1.4: CREACIÓN DE UNA CALCULADORA SIMPLE

Teoría

Una calculadora básica requiere:

1. **Entrada de datos:** Obtener números del usuario
2. **Operaciones:** Realizar cálculos aritméticos
3. **Salida:** Mostrar resultados

Estructura básica:

```javascript
// 1. Obtener valores
let numero1 = parseFloat(prompt("Ingresa el primer número"));
let numero2 = parseFloat(prompt("Ingresa el segundo número"));

// 2. Realizar operaciones
let suma = numero1 + numero2;
let resta = numero1 - numero2;
let multiplicacion = numero1 * numero2;
let division = numero1 / numero2;

// 3. Mostrar resultados
console.log("Suma:", suma);
console.log("Resta:", resta);
alert("Resultado de la suma: " + suma);
```

Consideraciones:

- Usar parseFloat() o parseInt() para convertir strings a números
- Validar que los valores no sean NaN (Not a Number)
- Manejar división entre cero

Pregunta de reflexión 4: ¿Qué sucede si el usuario ingresa texto en lugar de números? ¿Cómo lo evitarías?

Ejemplo

```javascript
// Calculadora completa
let num1 = parseFloat(prompt("Ingresa el primer número:"));
let num2 = parseFloat(prompt("Ingresa el segundo número:"));

// Validación
if (isNaN(num1) || isNaN(num2)) {
    alert("Error: Debes ingresar números válidos");
} else {
    // Operaciones
    let suma = num1 + num2;
    let resta = num1 - num2;
    let mult = num1 * num2;
    let div = num2 !== 0 ? num1 / num2 : "No se puede dividir entre cero";

    // Mostrar resultados
    console.log("=== RESULTADOS ===");
    console.log("Suma:", suma);
    console.log("Resta:", resta);
    console.log("Multiplicación:", mult);
    console.log("División:", div);

    alert("Suma: " + suma + "\nResta: " + resta);
}
```

Ejercicio 1.4

Crea una calculadora que:

1. Pida dos números al usuario
2. Pregunte qué operación quiere realizar (+, -, *, /)
3. Realice la operación correspondiente
4. Muestre el resultado con alert() y console.log()
5. Maneje el error de división entre cero

```javascript
// Tu código aquí:
```

SEMANA 2: EVENTOS Y FUNCIONES

TEMA 2.1: EVENTOS BÁSICOS - onclick, onload

Teoría

Eventos: Los eventos son acciones que ocurren en el navegador que JavaScript puede detectar y responder .

Eventos principales:

onclick: Se activa cuando el usuario hace clic en un elemento.

```html
<button onclick="miFuncion()">Haz click</button>

<script>
function miFuncion() {
    alert("Button clicked!");
}
</script>
```

onload: Se activa cuando la página o un elemento ha terminado de cargar.

```html
<body onload="inicializar()">
    <h1>Mi Página</h1>
</body>

<script>
function inicializar() {
    console.log("Página cargada completamente");
}
</script>
```

Otros eventos comunes:

- **onmouseover/onmouseout:** Cuando el mouse entra/sale de un elemento
- **onfocus/onblur:** Cuando un campo recibe/pierde el foco
- **onchange:** Cuando cambia el valor de un input
- **onkeydown/onkeyup:** Cuando se presiona/libera una tecla
- **onsubmit:** Cuando se envía un formulario

Pregunta de reflexión 5: ¿Cuál es la diferencia entre onclick en HTML y addEventListener? (Lo veremos más adelante)

Ejemplo

html

```html
<!DOCTYPE html>
<html>
<body onload="bienvenida()">
    <h1 id="titulo">Eventos en JavaScript</h1>

    <button onclick="cambiarColor()">Cambiar Color</button>
    <button onmouseover="agrandar()" onmouseout="normalizar()">
        Pasa el mouse
    </button>

    <div id="caja" style="width:100px; height:100px; background:blue;">
    </div>

    <script>
        function bienvenida() {
            console.log("Bienvenido a la página");
        }

        function cambiarColor() {
            let colores = ["red", "green", "yellow", "purple"];
            let random = Math.floor(Math.random() * colores.length);
            document.getElementById("titulo").style.color = colores[random];
        }

        function agrandar() {
            document.getElementById("caja").style.width = "150px";
            document.getElementById("caja").style.height = "150px";
        }
```

```
29
30          function normalizar() {
31              document.getElementById("caja").style.width = "100px";
32              document.getElementById("caja").style.height = "100px";
33          }
34      </script>
35  </body>
36  </html>
```

Ejercicio 2.1

Crea una página HTML que:

1. Muestre un alert() de bienvenida cuando cargue la página (onload)
2. Tenga un botón que al hacer clic (onclick) cambie el texto de un párrafo
3. Tenga una imagen que cambie de tamaño cuando pases el mouse sobre ella (onmouseover/onmouseout)

html

```
1  <!DOCTYPE html>
2  <html>
3  <body onload="">
4      <h1>Mis Eventos</h1>
5      <p id="miParrafo">Texto original</p>
6
7      <button onclick="">Cambiar texto</button>
8
9      <img id="miImagen" src="https://via.placeholder.com/150"
10         onmouseover="" onmouseout=""
11         style="transition: 0.3s;">
12
13     <script>
14         function bienvenida() {
15
16         }
17
18         function cambiarTexto() {
19
20         }
21
22         function cambiarTamaño() {
23
24         }
25     </script>
26 </body>
27 </html>
```

TEMA 2.2: FUNCIONES EN JS - DEFINICIÓN Y LLAMADA

Teoría

Funciones: Una función es un bloque de código reutilizable que realiza una tarea específica .

Formas de declarar funciones:

1. Declaración de función (Function Declaration):

```javascript
function saludar(nombre) {
    return "Hola " + nombre;
}

let mensaje = saludar("María");
console.log(mensaje); // "Hola María"
```

2. Expresión de función (Function Expression):

```javascript
const sumar = function(a, b) {
    return a + b;
};

let resultado = sumar(5, 3);
console.log(resultado); // 8
```

3. Arrow Function (ES6):

```javascript
const multiplicar = (a, b) => {
    return a * b;
};

// O más corto:
const dividir = (a, b) => a / b;

console.log(multiplicar(4, 2)); // 8
```

Partes de una función:

- **Palabra clave:** function
- **Nombre:** Identificador de la función
- **Parámetros:** Variables que recibe la función
- **Cuerpo:** Código entre llaves {}
- **Retorno:** Valor que devuelve (return)

Pregunta de reflexión 6: ¿Cuál es la diferencia entre console.log() y return en una función?

Ejemplo

javascript

```javascript
// Función sin parámetros ni retorno
function saludar() {
    console.log("¡Hola!");
}
saludar(); // Llamada a la función

// Función con parámetros
function sumar(a, b) {
    let resultado = a + b;
    console.log("La suma es:", resultado);
}
sumar(5, 3); // 8

// Función con retorno
function restar(a, b) {
    return a - b;
}
let diferencia = restar(10, 4);
console.log(diferencia); // 6

// Función con parámetros por defecto
function despedir(nombre = "amigo") {
    return "Adiós " + nombre;
}
console.log(despedir()); // "Adiós amigo"
console.log(despedir("Carlos")); // "Adiós Carlos"

// Arrow function
const cuadrado = (x) => x * x;
console.log(cuadrado(5)); // 25
```

Ejercicio 2.2

Crea las siguientes funciones:

1. saludar(nombre) - Retorne "Hola [nombre], bienvenido"
2. calcularIMC(peso, altura) - Retorne el IMC (peso / altura²)
3. esMayorDeEdad(edad) - Retorne true si edad >= 18, false si no
4. calcularAreaCirculo(radio) - Retorne el área (π × radio²)

```javascript
// Tu código aquí:
```

TEMA 2.3: USO DE getElementById Y innerHTML

Teoría

getElementById(): Método que selecciona un elemento del DOM por su atributo id .

```javascript
let elemento = document.getElementById("miId");
```

innerHTML: Propiedad que obtiene o establece el contenido HTML dentro de un elemento .

```javascript
// Leer contenido
let contenido = document.getElementById("parrafo").innerHTML;

// Modificar contenido
document.getElementById("parrafo").innerHTML = "Nuevo texto";
```

Otros métodos de selección:

- getElementsByClassName(): Selecciona por clase

- getElementsByTagName(): Selecciona por etiqueta
- querySelector(): Selecciona usando selectores CSS
- querySelectorAll(): Selecciona todos los elementos que coincidan

Pregunta de reflexión 7: ¿Por qué es importante que los IDs sean únicos en una página?

Ejemplo

html

```html
<!DOCTYPE html>
<html>
<body>
    <h1 id="titulo">Título Original</h1>
    <p id="parrafo">Párrafo inicial</p>
    <div id="contenedor"></div>

    <button onclick="modificar()">Modificar Contenido</button>

    <script>
        function modificar() {
            // Cambiar texto del h1
            document.getElementById("titulo").innerHTML = "Título Modificado";

            // Cambiar estilo
            document.getElementById("titulo").style.color = "red";

            // Agregar HTML complejo
            document.getElementById("contenedor").innerHTML =
                "<p>Nuevo <strong>párrafo</strong> con <em>formato</em></p>" +
                "<ul><li>Item 1</li><li>Item 2</li></ul>";

            // Leer contenido
            let texto = document.getElementById("parrafo").innerHTML;
            console.log("Contenido del párrafo:", texto);
        }
    </script>
</body>
</html>
```

Ejercicio 2.3

Crea una página con:

1. Un <h1> con id="titulo"

2. Un <p> con id="descripcion"
3. Un con id="resultado"
4. Tres botones que:
    - Botón 1: Cambie el título a "JavaScript es genial"
    - Botón 2: Cambie la descripción a "Aprendiendo DOM"
    - Botón 3: Agregue una lista de 3 elementos en el div resultado

html

```html
<!DOCTYPE html>
<html>
<body>
    <h1 id="titulo">Título Inicial</h1>
    <p id="descripcion">Descripción inicial</p>
    <div id="resultado"></div>

    <button onclick="">Cambiar título</button>
    <button onclick="">Cambiar descripción</button>
    <button onclick="">Mostrar lista</button>

    <script>
        function cambiarTitulo() {

        }

        function cambiarDescripcion() {

        }

        function mostrarLista() {

        }
    </script>
</body>
</html>
```

TEMA 2.4: MANIPULACIÓN DEL DOM - CAMBIAR TEXTO Y COLORES

Teoría

Manipulación del DOM: El Document Object Model (DOM) es una representación en árbol de la estructura HTML que JavaScript puede modificar.

Cambiar texto:

javascript

```javascript
// Usando innerHTML (puede contener HTML)
elemento.innerHTML = "<strong>Texto en negrita</strong>";

// Usando textContent (solo texto, más seguro)
elemento.textContent = "Texto plano";

// Usando innerText (similar a textContent pero respeta CSS)
elemento.innerText = "Texto visible";
```

Cambiar estilos CSS:

javascript

```javascript
// Cambiar una propiedad
elemento.style.color = "red";
elemento.style.fontSize = "20px";
elemento.style.backgroundColor = "blue";

// Múltiples propiedades
elemento.style.cssText = "color: red; font-size: 20px;";

// Agregar/quitar clases
elemento.classList.add("miClase");
elemento.classList.remove("otraClase");
elemento.classList.toggle("activa");
```

Pregunta de reflexión 8: ¿Cuál es la diferencia entre innerHTML y textContent? ¿Cuándo usar cada uno?

Ejemplo

html

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        .destacado {
            background-color: yellow;
            font-weight: bold;
            padding: 10px;
        }
        .oculto {
            display: none;
        }
    </style>
</head>
<body>
    <h1 id="titulo">Manipulación del DOM</h1>
    <p id="texto">Texto original</p>
    <div id="caja" style="width: 200px; height: 200px; background: lightblue;">
    </div>

    <button onclick="cambiarTexto()">Cambiar Texto</button>
    <button onclick="cambiarColor()">Cambiar Color</button>
    <button onclick="agregarClase()">Agregar Clase</button>
    <button onclick="toggleOculto()">Ocultar/Mostrar</button>

    <script>
        function cambiarTexto() {
            document.getElementById("texto").innerHTML =
                "Nuevo texto con <strong>formato</strong>";
        }

        function cambiarColor() {
            let caja = document.getElementById("caja");
            let colores = ["red", "green", "blue", "purple", "orange"];
            let random = Math.floor(Math.random() * colores.length);
            caja.style.backgroundColor = colores[random];
        }

        function agregarClase() {
            document.getElementById("titulo").classList.add("destacado");
        }

        function toggleOculto() {
            document.getElementById("texto").classList.toggle("oculto");
        }
    </script>
</body>
</html>
```

Ejercicio 2.4

Crea una página que permita:

1. Cambiar el color de fondo de la página aleatoriamente
2. Cambiar el tamaño de fuente de un párrafo (pequeño, mediano, grande)
3. Poner en negrita y cambiar el color de un título
4. Mostrar/ocultar un elemento con un botón

html

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        .grande { font-size: 24px; }
        .pequeno { font-size: 12px; }
        .negrita { font-weight: bold; }
        .oculto { display: none; }
    </style>
</head>
<body>
    <h1 id="miTitulo">Mi Título</h1>
    <p id="miParrafo">Este es un párrafo de ejemplo.</p>
    <div id="miDiv">Este div se puede ocultar</div>

    <button onclick="">Cambiar color fondo</button>
    <button onclick="">Tamaño pequeño</button>
    <button onclick="">Tamaño mediano</button>
    <button onclick="">Tamaño grande</button>
    <button onclick="">Estilo título</button>
    <button onclick="">Ocultar/Mostrar div</button>

    <script>
        function cambiarColorFondo() {

        }

        function tamanoPequeno() {
```

```
29
30          }
31
32          function tamanoMediano() {
33
34          }
35
36          function tamanoGrande() {
37
38          }
39
40          function estiloTitulo() {
41
42          }
43
44          function toggleDiv() {
45
46          }
47      </script>
48  </body>
49  </html>
```

SEMANA 3: ESTRUCTURAS DE CONTROL

TEMA 3.1: CONDICIONALES - if, else, switch

Teoría

Estructuras condicionales: Permiten ejecutar diferentes bloques de código según se cumplan o no ciertas condiciones.

if:

```javascript
if (condicion) {
    // Código si la condición es verdadera
}
```

if-else:

```javascript
if (condicion) {
    // Código si es verdadero
} else {
    // Código si es falso
}
```

if-else if-else:

```javascript
if (condicion1) {
    // Código si condicion1 es verdadera
} else if (condicion2) {
    // Código si condicion2 es verdadera
} else {
    // Código si ninguna es verdadera
}
```

switch:

```javascript
switch(expresion) {
    case valor1:
        // Código si expresion === valor1
        break;
    case valor2:
        // Código si expresion === valor2
        break;
    default:
        // Código si no coincide ningún caso
}
```

Pregunta de reflexión 9: ¿Cuándo es mejor usar switch en lugar de múltiples if-else?

Ejemplo

javascript

```javascript
// Ejemplo if-else
let edad = 18;

if (edad >= 18) {
    console.log("Eres mayor de edad");
} else {
    console.log("Eres menor de edad");
}

// Ejemplo if-else if
let nota = 85;
let calificacion;

if (nota >= 90) {
    calificacion = "A";
} else if (nota >= 80) {
    calificacion = "B";
} else if (nota >= 70) {
    calificacion = "C";
} else if (nota >= 60) {
    calificacion = "D";
} else {
    calificacion = "F";
}
console.log("Calificación:", calificacion);

// Ejemplo switch
let dia = 3;
let nombreDia;

switch(dia) {
    case 1:
        nombreDia = "Lunes";
        break;
    case 2:
        nombreDia = "Martes";
        break;
    case 3:
        nombreDia = "Miércoles";
        break;
    case 4:
        nombreDia = "Jueves";
        break;
    case 5:
        nombreDia = "Viernes";
        break;
    default:
        nombreDia = "Fin de semana";
}
console.log("Día:", nombreDia);

// Switch con strings
let fruta = "manzana";
switch(fruta) {
    case "manzana":
        console.log("La manzana es roja");
        break;
    case "banana":
        console.log("La banana es amarilla");
        break;
    default:
        console.log("Fruta desconocida");
}
```

Ejercicio 3.1

Crea un programa que:

1. **Pida al usuario un número del 1 al 7 y use switch para mostrar el día de la semana**
2. **Pida la edad y use if-else para determinar si es niño (0-12), adolescente (13-17), adulto (18-64) o adulto mayor (65+)**

3. **Pida una calificación (0-100) y determine si está aprobado (>=60) o reprobado**

```javascript
// Tu código aquí:
```

TEMA 3.2: OPERADORES LÓGICOS (&&, ||)

Teoría

Operadores lógicos: Permiten combinar múltiples condiciones.

AND (&&):

Ambas condiciones deben ser verdaderas.

```javascript
if (edad >= 18 && tieneLicencia) {
    console.log("Puede conducir");
}
```

OR (||):

Al menos una condición debe ser verdadera.

```javascript
if (esMiembro || tieneCupon) {
    console.log("Tiene descuento");
}
```

NOT (!):

Invierte el valor booleano.

```javascript
if (!estaConectado) {
    console.log("No hay conexión");
}
```

Combinaciones:

```javascript
if ((edad >= 18 && tieneLicencia) || esConductorProfesional) {
    console.log("Puede trabajar como conductor");
}
```

Pregunta de reflexión 10: ¿Qué resultado da true && false || true? Explica el orden de evaluación.

Ejemplo

```javascript
// Ejemplos AND
let edad = 25;
let tieneLicencia = true;

if (edad >= 18 && tieneLicencia) {
    console.log("Puede conducir legalmente");
}

// Ejemplos OR
let dia = "sábado";

if (dia === "sábado" || dia === "domingo") {
    console.log("¡Es fin de semana!");
}

// Ejemplos NOT
let estaApagado = false;

if (!estaApagado) {
    console.log("El sistema está encendido");
}

// Combinación compleja
let usuario = "admin";
let password = "12345";
let intentos = 2;

if (usuario === "admin" && password === "12345" && intentos < 3) {
    console.log("Acceso concedido");
} else {
    console.log("Acceso denegado");
}

// Tabla de verdad
console.log("true && true:", true && true);     // true
console.log("true && false:", true && false);   // false
console.log("true || false:", true || false);   // true
console.log("false || false:", false || false); // false
console.log("!true:", !true);                   // false
console.log("!false:", !false);                 // true
```

Ejercicio 3.2

Crea un sistema de validación que:

1. Verifique si un usuario puede entrar a una película (edad >= 18 O si viene con adulto)
2. Valide un formulario: el email NO debe estar vacío Y debe contener "@"
3. Determine si hace buen día: temperatura entre 20-30°C Y no llueve O es fin de semana

```javascript
// Tu código aquí:
```

TEMA 3.3: RECORRER ARRAYS CON BUCLES

Teoría

Arrays (Arreglos):

Colección ordenada de valores .

```javascript
let frutas = ["manzana", "banana", "naranja"];
let numeros = [1, 2, 3, 4, 5];
let mixto = ["texto", 42, true, null];
```

Acceso a elementos:

```javascript
let primera = frutas[0];      // "manzana"
let ultima = frutas[frutas.length - 1]; // "naranja"
```

Recorrer arrays con for:

```javascript
for (let i = 0; i < frutas.length; i++) {
    console.log(frutas[i]);
}
```

Recorrer arrays con forEach:

```javascript
frutas.forEach(function(fruta) {
    console.log(fruta);
});

// O con arrow function
frutas.forEach(fruta => console.log(fruta));
```

Pregunta de reflexión 11: ¿Cuál es la diferencia entre for y forEach? ¿Cuándo usar cada uno?

Ejemplo

```javascript
// Crear array
let colores = ["rojo", "verde", "azul", "amarillo"];

// Acceder a elementos
console.log(colores[0]);        // "rojo"
console.log(colores.length);    // 4

// Modificar elementos
colores[1] = "naranja";
console.log(colores);           // ["rojo", "naranja", "azul", "amarillo"]

// Agregar elementos
colores.push("morado");         // Agrega al final
colores.unshift("rosa");        // Agrega al inicio
console.log(colores);

// Recorrer con for
console.log("=== Usando for ===");
for (let i = 0; i < colores.length; i++) {
    console.log(i + ": " + colores[i]);
}

```

```javascript
// Recorrer con forEach
console.log("=== Usando forEach ===");
colores.forEach((color, indice) => {
    console.log(indice + ": " + color);
});

// Operaciones con arrays
let numeros = [1, 2, 3, 4, 5];

// Sumar todos los elementos
let suma = 0;
numeros.forEach(num => suma += num);
console.log("Suma:", suma);     // 15

// Encontrar el mayor
let mayor = numeros[0];
for (let num of numeros) {
    if (num > mayor) {
        mayor = num;
    }
}
console.log("Mayor:", mayor);   // 5
```

Ejercicio 3.3

Crea un programa que:

1. Declare un array con 5 nombres
2. Recorra el array con for y muestre cada nombre
3. Recorra el array con forEach y muestre los nombres en mayúsculas
4. Encuentre y muestre el nombre más largo
5. Cuente cuántos nombres tienen más de 5 letras

```javascript
// Tu código aquí:
```

TEMA 3.4: PROYECTO - LISTA DE TAREAS DINÁMICA

Teoría

Proyecto integrador: Aplicaremos: variables, funciones, eventos, manipulación del DOM, arrays y condicionales.

Características:

- Agregar tareas
- Marcar tareas como completadas
- Eliminar tareas
- Contador de tareas

Pregunta de reflexión 12: ¿Cómo organizarías el código para que sea más fácil de mantener?

Ejemplo

html

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        body { font-family: Arial; max-width: 600px; margin: 50px auto; }
        .tarea { padding: 10px; margin: 5px 0; background: #f0f0f0;
                 display: flex; justify-content: space-between; align-items: center; }
        .completada { text-decoration: line-through; color: gray; }
        button { padding: 5px 10px; cursor: pointer; }
        #inputTarea { width: 70%; padding: 10px; }
        #btnAgregar { padding: 10px 20px; }
    </style>
</head>
<body>
    <h1>Lista de Tareas</h1>
    <input type="text" id="inputTarea" placeholder="Nueva tarea...">
    <button id="btnAgregar" onclick="agregarTarea()">Agregar</button>
    <div id="listaTareas"></div>
    <p>Total: <span id="contador">0</span> tareas</p>

    <script>
        let tareas = [];

        function agregarTarea() {
            let input = document.getElementById("
```

```
26          let texto = input.value.trim();
27
28          if (texto === "") {
29              alert("Ingresa una tarea");
30              return;
31          }
32
33          let tarea = {
34              id: Date.now(),
35              texto: texto,
36              completada: false
37          };
38
39          tareas.push(tarea);
40          input.value = "";
41          renderizarTareas();
42      }
43
44      function toggleTarea(id) {
45          let tarea = tareas.find(t => t.id === id);
46          tarea.completada = !tarea.completada;
47          renderizarTareas();
48      }
49
50      function eliminarTarea(id) {
51          tareas = tareas.filter(t => t.id !== id);
52          renderizarTareas();
53      }
54
55      function renderizarTareas() {
56          let contenedor = document.getElementById("listaTareas");
57          contenedor.innerHTML = "";
58
59          tareas.forEach(tarea => {
60              let div = document.createElement("div");
61              div.className = "tarea";
62              div.innerHTML = `
63                  <span class="${tarea.completada ? 'completada' : ''}">
64                      ${tarea.texto}
65                  </span>
66                  <div>
67                      <button onclick="toggleTarea(${tarea.id})">
68                          ${tarea.completada ? '↩' : '✓'}
69                      </button>
```

```
                    <button onclick="eliminarTarea(${tarea.id})">X</button>
                </div>
            `;
            contenedor.appendChild(div);
        });

        document.getElementById("contador").textContent = tareas.length;
    }
</script>
</body>
</html>
```

Ejercicio 3.4

Mejora la lista de tareas agregando:

1. Un campo para priorizar tareas (alta, media, baja)
2. Un filtro para mostrar solo tareas pendientes o completadas
3. La fecha de creación de cada tarea
4. Un botón para eliminar todas las tareas completadas

javascript

```
// Tu código aquí:
```

SEMANA 4: BUCLES Y ARRAYS

TEMA 4.1: BUCLES - for, while

Teoría

Bucles (Loops): Permiten repetir un bloque de código múltiples veces.

for:

Se usa cuando se conoce el número de iteraciones.

```javascript
for (inicializacion; condicion; actualizacion) {
    // Código a repetir
}

for (let i = 0; i < 5; i++) {
    console.log("Iteración:", i);
}
```

while:

Se usa cuando no se conoce el número de iteraciones.

```javascript
while (condicion) {
    // Código a repetir
}

let i = 0;
while (i < 5) {
    console.log("Iteración:", i);
    i++;
}
```

do-while:

Ejecuta al menos una vez.

```javascript
do {
    // Código
} while (condicion);
```

Pregunta de reflexión 13: ¿Cuándo es mejor usar while en lugar de for?

Ejemplo

javascript

```javascript
// Ejemplo for
console.log("=== Contar del 1 al 5 ===");
for (let i = 1; i <= 5; i++) {
    console.log(i);
}

// For decremental
console.log("=== Cuenta regresiva ===");
for (let i = 5; i >= 1; i--) {
    console.log(i);
}
console.log("¡Despegue!");

// For con array
let frutas = ["manzana", "banana", "naranja"];
console.log("=== Frutas ===");
for (let i = 0; i < frutas.length; i++) {
    console.log(frutas[i]);
}

// While
console.log("=== While ===");
let contador = 0;
while (contador < 3) {
    console.log("Contador:", contador);
    contador++;
}

// Do-while
console.log("=== Do-while ===");
let numero;
do {
    numero = Math.floor(Math.random() * 10);
    console.log("Número:", numero);
} while (numero !== 5);
console.log("¡Encontré el 5!");

// Break y continue
```

```javascript
console.log("=== Break ===");
for (let i = 0; i < 10; i++) {
    if (i === 5) break;
    console.log(i);
}

console.log("=== Continue ===");
for (let i = 0; i < 5; i++) {
    if (i === 2) continue;
    console.log(i);
}
```

Ejercicio 4.1

Crea programas que:

1. Use for para mostrar la tabla de multiplicar del 7
2. Use while para sumar números hasta que la suma sea mayor a 100
3. Use for anidado para crear un triángulo de asteriscos
4. Use break para salir de un bucle cuando encuentre un número negativo

```javascript
// Tu código aquí:
```

TEMA 4.2: ARRAYS BÁSICOS

Teoría

Métodos de arrays:

Agregar/eliminar:

- push(): Agrega al final
- pop(): Elimina del final
- unshift(): Agrega al inicio
- shift(): Elimina del inicio
- splice(): Agrega/elimina en cualquier posición

Buscar:

- indexOf(): Encuentra la posición de un elemento
- includes(): Verifica si existe un elemento
- find(): Encuentra el primer elemento que cumpla una condición
- filter(): Filtra elementos que cumplan una condición

Transformar:

- map(): Transforma cada elemento
- slice(): Copia una porción del array
- concat(): Une arrays
- join(): Convierte array a string
- reverse(): Invierte el array
- sort(): Ordena el array

Pregunta de reflexión 14: ¿Cuál es la diferencia entre map() y forEach()?

Ejemplo

javascript

```javascript
let numeros = [1, 2, 3, 4, 5];
let frutas = ["manzana", "banana", "naranja"];

// Agregar elementos
frutas.push("uva");           // ["manzana", "banana", "naranja", "uva"]
frutas.unshift("pera");       // ["pera", "manzana", "banana", "naranja", "uva"]

// Eliminar elementos
let ultima = frutas.pop();    // Elimina "uva"
let primera = frutas.shift(); // Elimina "pera"

// splice(posición, cantidad a eliminar, elementos a agregar)
frutas.splice(1, 0, "kiwi");  // Agrega "kiwi" en posición 1

// Buscar
console.log(frutas.indexOf("banana"));  // Posición
console.log(frutas.includes("manzana")); // true

// Filter
let pares = numeros.filter(n => n % 2 === 0);
console.log(pares); // [2, 4]

// Map
let cuadrados = numeros.map(n => n * n);
console.log(cuadrados); // [1, 4, 9, 16, 25]

// Find
let mayorQue3 = numeros.find(n => n > 3);
console.log(mayorQue3); // 4

// Join
let texto = frutas.join(", ");
console.log(texto); // "manzana, kiwi, banana, naranja"

// Sort
let desordenado = [3, 1, 4, 1, 5, 9];
desordenado.sort((a, b) => a - b);
console.log(desordenado); // [1, 1, 3, 4, 5, 9]

// Slice
let copia = numeros.slice(1, 4);
console.log(copia); // [2, 3, 4]
```

Ejercicio 4.2

Dado el array productos = [{nombre: "Laptop", precio: 1000}, {nombre: "Mouse", precio: 25}, {nombre: "Teclado", precio: 50}]:

1. Usa filter() para obtener productos menores a $100
2. Usa map() para crear un array solo con los nombres
3. Usa find() para encontrar el producto "Mouse"
4. Usa reduce() para sumar todos los precios
5. Agrega un nuevo producto con push()

javascript

```javascript
let productos = [
    {nombre: "Laptop", precio: 1000},
    {nombre: "Mouse", precio: 25},
    {nombre: "Teclado", precio: 50}
];

// Tu código aquí:
```

TEMA 4.3: INTEGRACIÓN DE OBJETOS EN FORMULARIOS

Teoría

Objetos: Colección de propiedades (pares clave-valor) .

```javascript
let persona = {
    nombre: "Juan",
    edad: 30,
    ciudad: "Madrid"
};

// Acceder a propiedades
console.log(persona.nombre);  // "Juan"
console.log(persona["edad"]); // 30

// Modificar
persona.edad = 31;

// Agregar
persona.profesion = "Ingeniero";

// Eliminar
delete persona.ciudad;
```

Formularios y objetos: Capturar datos de formularios y convertirlos en objetos.

Pregunta de reflexión 15: ¿Por qué es útil almacenar datos de formularios en objetos?

Ejemplo

html

```html
<!DOCTYPE html>
<html>
<body>
    <form id="miFormulario">
        <input type="text" id="nombre" placeholder="Nombre" required>
        <input type="email" id="email" placeholder="Email" required>
        <input type="number" id="edad" placeholder="Edad" required>
        <select id="ciudad">
            <option value="">Selecciona ciudad</option>
            <option value="madrid">Madrid</option>
            <option value="barcelona">Barcelona</option>
        </select>
        <button type="submit">Guardar</button>
    </form>

    <div id="resultado"></div>

    <script>
        let usuarios = [];

        document.getElementById("miFormulario").addEventListener("submit",
            function(e) {
            e.preventDefault();

            // Crear objeto desde formulario
            let usuario = {
                nombre: document.getElementById("nombre").value,
```

```
            edad: parseInt(document.getElementById("edad").value),
            ciudad: document.getElementById("ciudad").value
        };

        // Validar
        if (usuario.edad < 18) {
            alert("Debes ser mayor de edad");
            return;
        }

        // Guardar
        usuarios.push(usuario);

        // Mostrar
        mostrarUsuarios();

        // Limpiar formulario
        this.reset();
    });

    function mostrarUsuarios() {
        let html = "<h2>Usuarios registrados:</h2><ul>";
        usuarios.forEach(u => {
            html += `<li>${u.nombre} - ${u.email} - ${u.ciudad}</li>`;
        });
        html += "</ul>";
        document.getElementById("resultado").innerHTML = html;
    }
</script>
</body>
</html>
```

Ejercicio 4.3

Crea un formulario de registro de libros que:

1. Capture: título, autor, año, género
2. Valide que todos los campos estén completos
3. Guarde cada libro como un objeto en un array
4. Muestre la lista de libros registrados
5. Calcule y muestre cuántos libros hay por género

```html
<form id="formLibro">
    <input type="text" id="titulo" placeholder="Título" required>
    <input type="text" id="autor" placeholder="Autor" required>
    <input type="number" id="anio" placeholder="Año" required>
    <select id="genero">
        <option value="ficcion">Ficción</option>
        <option value="ciencia">Ciencia</option>
        <option value="historia">Historia</option>
    </select>
    <button type="submit">Agregar Libro</button>
</form>
<div id="listaLibros"></div>

<script>
    let libros = [];

    document.getElementById("formLibro").addEventListener("submit", function(e) {
        e.preventDefault();

        // Tu código aquí:


    });
</script>
```

TEMA 4.4: PROYECTO - FORMULARIO CON VALIDACIÓN

Teoría

Validación de formularios:

- Validación del lado del cliente (JavaScript)
- Expresiones regulares para emails
- Validación de campos requeridos
- Mensajes de error personalizados

Pregunta de reflexión 16: ¿Por qué es importante validar tanto en el cliente como en el servidor?

Ejemplo

```javascript
function validarFormulario(datos) {
    let errores = [];

    // Validar nombre
    if (datos.nombre.trim().length < 3) {
        errores.push("El nombre debe tener al menos 3 caracteres");
    }

    // Validar email con regex
    let emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (!emailRegex.test(datos.email)) {
        errores.push("Email inválido");
    }

    // Validar contraseña
    if (datos.password.length < 8) {
        errores.push("La contraseña debe tener al menos 8 caracteres");
    }

    // Validar que las contraseñas coincidan
    if (datos.password !== datos.confirmarPassword) {
        errores.push("Las contraseñas no coinciden");
    }

    return errores;
}
```

Ejercicio 4.4

Crea un formulario de registro completo con:

1. Campos: nombre, email, teléfono, contraseña, confirmar contraseña
2. Validaciones:
    - Nombre: mínimo 3 caracteres
    - Email: formato válido
    - Teléfono: 10 dígitos
    - Contraseña: mínimo 8 caracteres, 1 mayúscula, 1 número

- Confirmar contraseña: debe coincidir

3. Mostrar mensajes de error debajo de cada campo
4. Si todo es válido, mostrar mensaje de éxito

javascript

```javascript
// Tu código aquí:
```

SEMANA 5: OBJETOS Y CSS

TEMA 5.1: OBJETOS BÁSICOS EN JS

Teoría

Objetos literales:

javascript

```javascript
let persona = {
    nombre: "Ana",
    edad: 25,
    saludar: function() {
        return "Hola, soy " + this.nombre;
    }
};

console.log(persona.saludar()); // "Hola, soy Ana"
```

Constructor de objetos:

```javascript
function Persona(nombre, edad) {
    this.nombre = nombre;
    this.edad = edad;
    this.saludar = function() {
        return "Hola, soy " + this.nombre;
    };
}

let juan = new Persona("Juan", 30);
console.log(juan.saludar());
```

Pregunta de reflexión 17: ¿Qué es this en JavaScript?

Ejemplo

javascript

```javascript
// Objeto literal
let libro = {
    titulo: "JavaScript Avanzado",
    autor: "Carlos Pérez",
    anio: 2024,
    disponible: true,
    prestar: function() {
        if (this.disponible) {
            this.disponible = false;
            return "Libro prestado";
        }
        return "Libro no disponible";
    }
};

console.log(libro.prestar());
console.log(libro.disponible); // false

// Objeto con métodos
let calculadora = {
    valor: 0,
    sumar: function(n) { this.valor += n; return this; },
    restar: function(n) { this.valor -= n; return this; },
    multiplicar: function(n) { this.valor *= n; return this; },
    resultado: function() { return this.valor; }
};

console.log(calculadora.sumar(5).multiplicar(2).resultado()); // 20
```

Ejercicio 5.1

Crea un objeto cuentaBancaria con:

1. Propiedades: titular, saldo
2. Métodos: depositar(cantidad), retirar(cantidad), consultarSaldo()
3. Valida que no se pueda retirar más del saldo disponible
4. Crea 2 cuentas diferentes y realiza operaciones

```javascript
// Tu código aquí:
```

TEMA 5.2: PROPIEDADES Y MÉTODOS

Teoría

Propiedades: Atributos o características de un objeto.

Métodos: Funciones que pertenecen a un objeto.

Acceso dinámico:

```javascript
let obj = { clave: "valor" };
let propiedad = "clave";
console.log(obj[propiedad]); // "valor"
```

Pregunta de reflexión 18: ¿Cómo iteras sobre las propiedades de un objeto?

Ejemplo

javascript

```javascript
let producto = {
    nombre: "Laptop",
    precio: 1000,
    stock: 10,
    descuento: 0.1,

    precioConDescuento: function() {
        return this.precio * (1 - this.descuento);
    },

    agregarStock: function(cantidad) {
        this.stock += cantidad;
    },

    vender: function(cantidad) {
        if (cantidad <= this.stock) {
            this.stock -= cantidad;
            return true;
        }
        return false;
    },

    getInfo: function() {
        return `${this.nombre} - $${this.precioConDescuento()}
                (Stock: ${this.stock})`;
    }
};

// Iterar propiedades
for (let clave in producto) {
    if (typeof producto[clave] !== 'function') {
        console.log(clave + ":", producto[clave]);
    }
}

console.log(producto.getInfo());
```

Ejercicio 5.2

Crea un objeto estudiante con:

1. Propiedades: nombre, calificaciones (array), materia
2. Métodos:
    - agregarCalificacion(nota)
    - calcularPromedio()
    - obtenerEstado() (retorna "Aprobado" si promedio >= 60)
    - mostrarInformacion()
3. Crea 3 estudiantes y muestra sus promedios

```javascript
// Tu código aquí:
```

TEMA 5.3: INTERACTIVIDAD CON CSS

Teoría

Manipular CSS con JavaScript:

```javascript
// style.property
elemento.style.color = "red";
elemento.style.fontSize = "20px";

// classList
elemento.classList.add("activo");
elemento.classList.remove("inactivo");
elemento.classList.toggle("visible");

// getComputedStyle
let estilo = window.getComputedStyle(elemento);
let color = estilo.color;
```

Pregunta de reflexión 19: ¿Cuál es la ventaja de usar clases CSS en lugar de modificar estilos directamente?

Ejemplo

html

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        .caja {
            width: 100px;
            height: 100px;
            background: blue;
            transition: all 0.3s;
        }
        .caja.grande {
            width: 200px;
            height: 200px;
        }
        .caja.rojo {
            background: red;
        }
        .caja.oculto {
            opacity: 0;
            transform: scale(0);
        }
        .animar {
            animation: mover 2s infinite;
        }
        @keyframes mover {
            from { transform: translateX(0); }
            to { transform: translateX(100px); }
        }
    </style>
```

```
</head>
<body>
    <div id="miCaja" class="caja"></div>
    <button onclick="agrandar()">Agrandar</button>
    <button onclick="cambiarColor()">Color</button>
    <button onclick="toggleOculto()">Ocultar</button>
    <button onclick="animar()">Animar</button>

    <script>
        let caja = document.getElementById("miCaja");

        function agrandar() {
            caja.classList.toggle("grande");
        }

        function cambiarColor() {
            caja.classList.toggle("rojo");
        }

        function toggleOculto() {
            caja.classList.toggle("oculto");
        }

        function animar() {
            caja.classList.toggle("animar");
        }
    </script>
</body>
</html>
```

Ejercicio 5.3

Crea una página con:

1. Un elemento que cambie de color al pasar el mouse
2. Un botón que active/desactive un modo oscuro (cambia fondo y texto)
3. Un slider que controle el tamaño de un elemento
4. Una animación que se active al hacer clic

```javascript
// Tu código aquí:
```

TEMA 5.4: DISEÑO VISUAL DE APLICACIÓN

Teoría

Principios de diseño:

- Consistencia visual
- Feedback al usuario
- Jerarquía visual
- Espaciado y alineación

Pregunta de reflexión 20: ¿Qué hace que una interfaz sea "usable"?

Ejemplo

Aplicación completa de notas con diseño profesional.

Ejercicio 5.4

Diseña una aplicación de:

- Lista de contactos
- Calculadora visual
- Reloj digital Elige una y aplícale:

1. Diseño responsive
2. Animaciones CSS
3. Validaciones visuales
4. Feedback al usuario

```javascript
// Tu código aquí:
```

SEMANA 6: EVENTOS AVANZADOS

TEMA 6.1: EVENTOS AVANZADOS - onchange, onkeyup

Teoría

- **onchange: Se activa cuando cambia el valor de un elemento y pierde el foco.**
- **onkeyup: Se activa cuando se libera una tecla.**
- **onkeydown: Se activa cuando se presiona una tecla.**
- **oninput: Se activa cuando cambia el valor de un input (en tiempo real).**

Pregunta de reflexión 21: ¿Cuál es la diferencia entre onchange y oninput?

Ejemplo

```html
<input type="text" id="buscador" onkeyup="buscar()"
       placeholder="Escribe para buscar...">

<input type="select" id="pais" onchange="cambiarPais()">

<script>
function buscar() {
    let texto = document.getElementById("buscador").value;
    console.log("Buscando:", texto);
    // Filtrar resultados en tiempo real
}

function cambiarPais() {
    let pais = document.getElementById("pais").value;
    console.log("País seleccionado:", pais);
}
</script>
```

Ejercicio 6.1

Crea:

1. **Un buscador que filtre una lista mientras escribes (onkeyup)**
2. **Un formulario que valide campos en tiempo real (oninput)**
3. **Un select que muestre información diferente según la opción (onchange)**

javascript

```javascript
// Tu código aquí:
```

TEMA 6.2: USO DE addEventListener

Teoría

addEventListener: Forma moderna y recomendada de agregar eventos .

javascript

```javascript
elemento.addEventListener("click", funcion);
elemento.addEventListener("click", () => {
    console.log("Click");
});

// Remover evento
elemento.removeEventListener("click", funcion);

// Múltiples eventos al mismo elemento
elemento.addEventListener("click", funcion1);
elemento.addEventListener("click", funcion2);

// Event object
elemento.addEventListener("click", (e) => {
    console.log(e.target);      // Elemento clickeado
    console.log(e.type);        // Tipo de evento
    e.preventDefault();         // Prevenir comportamiento default
    e.stopPropagation();        // Detener propagación
});
```

Pregunta de reflexión 22: ¿Por qué es mejor addEventListener que onclick?

Ejemplo

```javascript
let boton = document.getElementById("miBoton");

// Agregar evento
boton.addEventListener("click", function(e) {
    console.log("Click en:", e.target);
});

// Evento en múltiples elementos
document.querySelectorAll(".item").forEach(item => {
    item.addEventListener("click", manejarClick);
});

// Delegación de eventos
document.getElementById("lista").addEventListener("click", (e) => {
    if (e.target.matches(".eliminar")) {
        e.target.parentElement.remove();
    }
});

// Eventos de teclado
document.addEventListener("keydown", (e) => {
    if (e.key === "Enter") {
        console.log("Presionó Enter");
    }
});

// Evento de scroll
window.addEventListener("scroll", () => {
    if (window.scrollY > 100) {
        console.log("Scroll > 100px");
    }
});
```

Ejercicio 6.2

Crea:

1. **Un botón que cambie de color con addEventListener**
2. **Delegación de eventos para una lista dinámica**

3. **Atajo de teclado (presionar "S" para guardar)**
4. **Detectar si el usuario está haciendo scroll hacia arriba o abajo**

```javascript
// Tu código aquí:
```

TEMA 6.3: DOCUMENTACIÓN TÉCNICA BREVE

Teoría

Comentarios en JavaScript:

```javascript
// Comentario de una línea

/* Comentario
   de múltiples
   líneas */

/**
 * Documentación JSDoc
 * @param {string} nombre - El nombre del usuario
 * @returns {string} Saludo personalizado
 */
function saludar(nombre) {
    return "Hola " + nombre;
}
```

Pregunta de reflexión 23: ¿Por qué es importante documentar el código?

Ejercicio 6.3

Documenta todas las funciones que creaste en ejercicios anteriores usando JSDoc.

```javascript
/**
 * Descripción de la función
 * @param {tipo} parametro - Descripción
 * @returns {tipo} Descripción del retorno
 */
```

TEMA 6.4: CIERRE DE UNIDAD

Repaso general de todos los temas vistos.

SEMANA 7: EVALUACIÓN PRÁCTICA

TEMA 7.1: EVALUACIÓN PRÁCTICA

Proyecto final integrador.

🎯 RETOS FINALES DE LA UNIDAD 1

RETO 1: "SISTEMA DE GESTIÓN DE BIBLIOTECA"

Objetivo: Crear una aplicación completa para gestionar una biblioteca usando todos los conceptos aprendidos.

Requisitos funcionales:

Parte 1 - Estructura de datos:

1. **Crea un array de objetos libros con propiedades:**
    - **id (único)**
    - **titulo**
    - **autor**
    - **anio**
    - **genero (ficcion, ciencia, historia, otros)**
    - **disponible (boolean)**
    - **vecesPrestadas (contador)**

Parte 2 - Funcionalidades: 2. Implementa las siguientes funciones:

- **agregarLibro(titulo, autor, anio, genero): Agrega un nuevo libro**

- buscarLibro(titulo): Busca un libro por título (parcial o total)
- prestarLibro(id): Marca el libro como no disponible e incrementa vecesPrestadas
- devolverLibro(id): Marca el libro como disponible
- eliminarLibro(id): Elimina un libro del array
- getEstadisticas(): Retorna objeto con:
    - totalLibros
    - librosDisponibles
    - librosPrestados
    - libroMasPrestado

Parte 3 - Interfaz de usuario: 3. Crea un HTML con:

- Formulario para agregar libros (con validación)
- Campo de búsqueda en tiempo real (onkeyup)
- Lista de libros mostrando:
    - Título, autor, año
    - Estado (disponible/prestado) con color diferente
    - Botones: Prestar, Devolver, Eliminar
- Sección de estadísticas que se actualice automáticamente

Parte 4 - Características avanzadas: 4. Agrega:

- **Validación de formulario completa**
- **Confirmación antes de eliminar**
- **Filtro por género (select onchange)**
- **Ordenamiento (por título, año, veces prestadas)**
- **Persistencia en localStorage (opcional)**
- **Animaciones CSS al agregar/eliminar**
- **Diseño responsive**

Entregables:

- Código HTML completo
- Código JavaScript documentado con JSDoc
- Capturas de pantalla de la aplicación funcionando
- README explicando cómo usar la aplicación

Criterios de evaluación: ✓ Uso correcto de arrays y objetos (20%) ✓ Manipulación del DOM (20%) ✓ Eventos y validaciones (20%) ✓ Funciones y lógica de programación (20%) ✓ Diseño y experiencia de usuario (20%)

RETO 2: "APLICACIÓN DE CLIMA CON INTERFAZ DINÁMICA"

Objetivo: Crear una aplicación interactiva que simule un sistema de consulta del clima con interfaz dinámica.

Requisitos funcionales:

Parte 1 - Estructura de datos:

1. Crea un objeto ciudades que contenga:
    - Nombre de la ciudad
    - Temperatura (número aleatorio entre -10 y 40)
    - Condición (soleado, nublado, lluvioso, nevando)
    - Humedad (porcentaje)
    - Viento (km/h)
    - Historial (array de temperaturas de los últimos 7 días)

Parte 2 - Funcionalidades: 2. Implementa:

- agregarCiudad(nombre): Agrega ciudad con datos aleatorios
- actualizarClima(ciudad): Actualiza temperatura y condición aleatoriamente
- getPromedioTemperatura(ciudad): Calcula promedio del historial
- getCiudadMasCalida(): Retorna la ciudad con mayor temperatura

- filtrarPorCondicion(condicion): Retorna ciudades con esa condición
- generarRecomendacion(ciudad): Retorna recomendación según clima:
    - Si hace calor (>30): "Usa ropa ligera y hidrátate"
    - Si hace frío (<10): "Abrígate bien"
    - Si llueve: "Lleva paraguas"
    - Si nieva: "Precaución en las calles"

Parte 3 - Interfaz interactiva: 3. Crea:

- Selector de ciudad (dropdown)
- Visualización del clima actual con:
    - Icono según condición (usa emojis o imágenes)
    - Temperatura grande y destacada
    - Detalles (humedad, viento)
    - Color de fondo que cambie según:
        - Soleado: degradado amarillo/naranja
        - Nublado: gris
        - Lluvioso: azul oscuro
        - Nevando: blanco/azul claro
- Gráfico simple del historial (barras con divs)
- Recomendación dinámica

Parte 4 - Interactividad avanzada: 4. Agrega:

- Actualización automática cada 10 segundos (setInterval)
- Animación de transición entre ciudades
- Búsqueda predictiva (autocomplete)
- Modo oscuro/claro (toggle)
- Exportar datos a JSON (descargar archivo)

- Teclas de acceso rápido:
    - Flechas arriba/abajo: cambiar ciudad
    - "R": refrescar clima
    - "N": nueva ciudad
- Validación de entrada
- Mensajes de error personalizados

Parte 5 - Objetos y métodos avanzados: 5. Implementa:

- Constructor o clase para Ciudades
- Métodos encadenados (method chaining)
- Uso de this correctamente
- Closures para contadores privados
- Callbacks o promesas (simuladas)

Entregables:

- Código completo y documentado
- Diagrama de flujo de la aplicación
- Video demo de 2 minutos mostrando funcionalidades
- Archivo README.md con:
    - Instrucciones de instalación
    - Lista de características
    - Capturas de pantalla
    - Lecciones aprendidas

Criterios de evaluación: ✓ Estructura de datos y objetos (15%) ✓ Funciones y algoritmos (20%) ✓ Manipulación del DOM y eventos (20%) ✓ CSS dinámico y animaciones (15%) ✓ Validaciones y manejo de errores (15%) ✓ Creatividad y funcionalidades extra (15%)

Bonus points (+10%):

- **Usar API real del clima (OpenWeatherMap)**
- **Geolocalización del usuario**
- **Gráficos con Chart.js**
- **PWA (Progressive Web App)**

BIBLIOGRAFÍA

Bibliografía Fundamental:

1. MDN Web Docs. (2024). *JavaScript Guide*. Mozilla Developer Network. https://developer.mozilla.org/es/docs/Web/JavaScript/Guide
2. MDN Web Docs. (2024). *Document Object Model (DOM)*. https://developer.mozilla.org/es/docs/Web/API/Document_Object_Model
3. Flanagan, D. (2020). *JavaScript: The Definitive Guide*. 7th Edition. O'Reilly Media.
4. Zakas, N. C. (2021). *Understanding ECMAScript 6*. No Starch Press.
5. W3Schools. (2024). *JavaScript Tutorial*. https://www.w3schools.com/js/
6. MDN Web Docs. (2024). *Introduction to events*. https://developer.mozilla.org/es/docs/Learn/JavaScript/Building_blocks/Events
7. JavaScript.info. (2024). *Introduction to Events*. https://javascript.info/introduction-browser-events
8. MDN Web Docs. (2024). *Functions*. https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Functions
9. Crockford, D. (2008). *JavaScript: The Good Parts*. O'Reilly Media.
10. MDN Web Docs. (2024). *Document.getElementById()*. https://developer.mozilla.org/es/docs/Web/API/Document/getElementById

11. Duckett, J. (2014). *JavaScript & jQuery: Interactive Front-End Web Development*. Wiley.
12. MDN Web Docs. (2024). *Element.innerHTML*. https://developer.mozilla.org/es/docs/Web/API/Element/innerHTML
13. MDN Web Docs. (2024). *Using the W3C DOM Level 1 Core*. https://developer.mozilla.org/es/docs/Web/API/Document_Object_Model
14. Eich, B. (2008). *Popularity: The success of the web platform*. O'Reilly Radar.
15. MDN Web Docs. (2024). *Making decisions in your code — conditionals*. https://developer.mozilla.org/es/docs/Learn/JavaScript/Building_blocks/conditionals
16. Simpson, K. (2015). *You Don't Know JS: Types & Grammar*. O'Reilly Media.
17. MDN Web Docs. (2024). *Logical operators*. https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/Logical_Operators
18. MDN Web Docs. (2024). *Array*. https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Array
19. Bevacqua, N. (2015). *JavaScript Application Design*. Manning Publications.
20. MDN Web Docs. (2024). *Loops and iteration*. https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Loops_and_iteration
21. Haverbeke, M. (2018). *Eloquent JavaScript*. 3rd Edition. No Starch Press.
22. MDN Web Docs. (2024). *Working with objects*. https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Working_with_Objects

23. MDN Web Docs. (2024). *EventTarget.addEventListener()*. https://developer.mozilla.org/es/docs/Web/API/EventTarget/addEventListener

Bibliografía Complementaria:

24. Resig, J., Bibeault, B., & Marak, J. (2013). *Secrets of the JavaScript Ninja*. Manning Publications.
25. Osmani, A. (2012). *Learning JavaScript Design Patterns*. O'Reilly Media.
26. Wikipedia. (2024). *JavaScript*. https://es.wikipedia.org/wiki/JavaScript
27. ECMA International. (2024). *ECMAScript® 2024 Language Specification*. https://ecma-international.org/publications-and-standards/standards/ecma-262/
28. FreeCodeCamp. (2024). *JavaScript Algorithms and Data Structures*. https://www.freecodecamp.org/
29. Codecademy. (2024). *Learn JavaScript*. https://www.codecademy.com/learn/introduction-to-javascript
30. JavaScript.info. (2024). *The Modern JavaScript Tutorial*. https://javascript.info/

Nota para el estudiante: Esta guía está diseñada para ser completada de manera progresiva durante 7 semanas. Se recomienda:

- Practicar diariamente al menos 1 hora
- Resolver todos los ejercicios básicos antes de avanzar
- Consultar la documentación oficial (MDN) cuando tengas dudas
- No copiar código, entiende la lógica detrás de cada solución
- Los retos finales requieren integrar TODO lo aprendido

Diseñado por: Lic. Carlos M. Galdámez
AVC El Salvador