Tutorial Completo de JavaScript en Español 🚀

Asistencia sobre aplicaciones Especificas - Profe: Agustin Gatica

📚 TABLA DE CONTENIDOS

🔧 Configuración Inicial

Instalación de VSCode

1. Descarga VSCode desde https://code.visualstudio.com
2. Instala las extensiones recomendadas:
   - **Live Server** (Ritwick Dey)
   - **Prettier - Code Formatter**
   - **ES7+ React/Redux/React-Native snippets**

Estructura de Proyecto

```
mi-proyecto/
├── index.html
├── style.css
└── script.js
```

Primer archivo HTML

index.html

```
<!DOCTYPE html>
<html lang="es">
```

```html
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mi Proyecto JavaScript</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h1>Bienvenido a JavaScript</h1>
    <p id="mensaje">Aquí veremos ejemplos increíbles</p>

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

🎯 Fundamentos de JavaScript

1. Variables y Tipos de Datos

script.js

```javascript
// VAR (evitar usar - obsoleto)
var nombre = "Juan";

// LET (usar en la mayoría de casos)
let edad = 25;
let ciudad;

// CONST (usar cuando el valor no cambie)
const PI = 3.14159;
const pais = "España";

// Tipos de datos
let numero = 42;           // number
let texto = "Hola";        // string
let booleano = true;       // boolean
let nulo = null;           // null
let indefinido;            // undefined
let objeto = { edad: 30 }; // object
let arreglo = [1, 2, 3];   // array (es un objeto)
let funcion = () => {};    // function

console.log(typeof numero);    // "number"
console.log(typeof texto);     // "string"
```

2. Operadores Básicos

```
// Aritméticos
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
let potencia = 2 ** 3;    // 8

// Incremento y Decremento
let contador = 5;
contador++;               // 6
contador--;               // 5
contador += 3;            // 8
contador -= 2;            // 6

// Comparación
console.log(5 == "5");    // true (compara valor)
console.log(5 === "5");   // false (compara tipo y valor) ✅ MEJOR
console.log(5 != "5");    // false
console.log(5 !== "5");   // true ✅ MEJOR

// Lógicos
let resultado = true && false;  // false (AND)
let resultado2 = true || false; // true (OR)
let resultado3 = !true;         // false (NOT)
```

3. Strings (Cadenas de Texto)

```
// Formas de declarar strings
let str1 = "Hola";
let str2 = 'Mundo';
let str3 = `Hola Mundo`; // Template literals ✅ MEJOR

// Template literals (muy útiles)
let nombre = "Pedro";
let edad = 30;
console.log(`Mi nombre es ${nombre} y tengo ${edad} años`);

// Métodos de strings
let texto = "JavaScript";
console.log(texto.length);           // 10
console.log(texto.toUpperCase());    // "JAVASCRIPT"
console.log(texto.toLowerCase());    // "javascript"
console.log(texto.charAt(0));        // "J"
console.log(texto.indexOf("Script")); // 4
console.log(texto.slice(0, 4));      // "Java"
console.log(texto.includes("Script")); // true
console.log(texto.replace("Java", "Type")); // "TypeScript"
console.log(texto.trim());           // elimina espacios
```

4. Arreglos (Arrays)

```
// Declarar arreglos
let frutas = ["manzana", "plátano", "naranja"];
let numeros = [1, 2, 3, 4, 5];
let mixto = [1, "texto", true, null];

// Acceder a elementos
console.log(frutas[0]);    // "manzana"
console.log(frutas.length); // 3

// Métodos de arreglos
frutas.push("pera");       // agrega al final
console.log(frutas);       // ["manzana", "plátano", "naranja", "pera"]

frutas.pop();              // elimina el último
console.log(frutas);       // ["manzana", "plátano", "naranja"]

frutas.shift();            // elimina el primero
frutas.unshift("piña");    // agrega al inicio

// Métodos útiles
console.log(frutas.includes("manzana")); // true
console.log(frutas.indexOf("plátano"));  // 1
console.log(frutas.join(", "));          // "piña, plátano, naranja"

// Desestructuración
let [primera, segunda, tercera] = frutas;
console.log(primera); // "piña"

// Iterar arrays
frutas.forEach((fruta, indice) => {
    console.log(`${indice}: ${fruta}`);
});

// map() - transformar elementos
let numerosDobles = numeros.map(n => n * 2);
console.log(numerosDobles); // [2, 4, 6, 8, 10]

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

// find() - encontrar primer elemento
let primero = numeros.find(n => n > 3);
console.log(primero); // 4

// reduce() - reducir a un valor
let suma = numeros.reduce((acumulador, actual) => {
    return acumulador + actual;
```

```
}, 0);
console.log(suma); // 15
```

5. Objetos

```
// Declarar objetos
let persona = {
    nombre: "Ana",
    edad: 28,
    ciudad: "Madrid",
    hobbies: ["lectura", "viajes"],
    saludar: function() {
        console.log(`Hola, soy ${this.nombre}`);
    }
};

// Acceder a propiedades
console.log(persona.nombre);        // "Ana"
console.log(persona["edad"]);       // 28

// Modificar propiedades
persona.edad = 29;
persona["ciudad"] = "Barcelona";

// Agregar propiedades
persona.email = "ana@email.com";

// Eliminar propiedades
delete persona.email;

// Métodos
persona.saludar();

// Iterar propiedades
for (let clave in persona) {
    console.log(clave, persona[clave]);
}

// Desestructuración
let { nombre, edad } = persona;
console.log(nombre); // "Ana"

// Object.keys(), Object.values()
console.log(Object.keys(persona));   // ["nombre", "edad", "ciudad", ...]
console.log(Object.values(persona)); // ["Ana", 29, "Barcelona", ...]
```

6. Condicionales

```
// If/Else
let edad = 18;

if (edad >= 18) {
    console.log("Eres mayor de edad");
} else if (edad >= 13) {
    console.log("Eres adolescente");
} else {
    console.log("Eres niño");
}

// Operador ternario ✅ MÁS LIMPIO
let mensaje = edad >= 18 ? "Mayor de edad" : "Menor de edad";

// Switch
let dia = 3;
switch (dia) {
    case 1:
        console.log("Lunes");
        break;
    case 2:
        console.log("Martes");
        break;
    case 3:
        console.log("Miércoles");
        break;
    default:
        console.log("Día inválido");
}
```

7. Bucles

```
// FOR tradicional
for (let i = 0; i < 5; i++) {
    console.log(i);
}

// WHILE
let contador = 0;
while (contador < 5) {
    console.log(contador);
    contador++;
}

// DO-WHILE
let x = 0;
do {
    console.log(x);
    x++;
} while (x < 5);
```

```
// FOR-OF (para valores)
let array = ["a", "b", "c"];
for (let valor of array) {
    console.log(valor);
}

// FOR-IN (para claves/índices)
for (let indice in array) {
    console.log(indice, array[indice]);
}
```

8. Funciones

```
// Función tradicional
function saludar(nombre) {
    return `Hola, ${nombre}`;
}
console.log(saludar("Juan")); // "Hola, Juan"

// Parámetros por defecto
function sumar(a = 0, b = 0) {
    return a + b;
}
console.log(sumar(3, 5));   // 8
console.log(sumar(3));      // 3

// Función flecha (arrow function) ✅ MÁS MODERNO
const restar = (a, b) => {
    return a - b;
};
console.log(restar(10, 3)); // 7

// Arrow function simplificada
const multiplicar = (a, b) => a * b;
console.log(multiplicar(4, 5)); // 20

// Función con múltiples líneas
const procesar = (numero) => {
    const doble = numero * 2;
    const resultado = doble + 10;
    return resultado;
};
console.log(procesar(5)); // 20

// Funciones que retornan funciones
const crear_multiplicador = (factor) => {
    return (numero) => numero * factor;
};
const triplicar = crear_multiplicador(3);
console.log(triplicar(5)); // 15
```

```
// Spread operator (...)
const sumarTodos = (...numeros) => {
    return numeros.reduce((a, b) => a + b, 0);
};
console.log(sumarTodos(1, 2, 3, 4, 5)); // 15
```

🎨 Manipulación del DOM

1. Seleccionar Elementos

```
// getElementById - por ID (más rápido)
let elemento1 = document.getElementById("mensaje");

// querySelector - por selector CSS (muy flexible)
let elemento2 = document.querySelector("h1");
let elemento3 = document.querySelector(".clase");
let elemento4 = document.querySelector("#id");

// querySelectorAll - todos los elementos
let elementos = document.querySelectorAll("p");

// getElementsByClassName
let elementosPorClase = document.getElementsByClassName("clase");

// getElementsByTagName
let parrafos = document.getElementsByTagName("p");

// 💡 TIP: Preferir querySelector y querySelectorAll
```

Actualiza tu HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Manipulación del DOM</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h1 id="titulo">Mi Primer Proyecto</h1>
    <p id="parrafo" class="texto">Este es un párrafo</p>
    <div id="contenedor">
        <button id="boton">Haz clic</button>
    </div>

    <script src="script.js"></script>
```

```
</body>
</html>
```

2. Modificar Contenido

```
// textContent - solo texto
document.getElementById("titulo").textContent = "Nuevo Título";

// innerHTML - HTML completo (cuidado con seguridad)
document.getElementById("contenedor").innerHTML = "<p>Nuevo contenido</p>";

// innerText - texto visible (respeta CSS)
document.getElementById("parrafo").innerText = "Texto modificado";

// 💡 TIP: textContent es más seguro que innerHTML
```

3. Modificar Atributos

```
// getAttribute
let valorId = document.querySelector("button").getAttribute("id");

// setAttribute
let boton = document.getElementById("boton");
boton.setAttribute("data-accion", "guardar");
boton.setAttribute("disabled", "true");

// removeAttribute
boton.removeAttribute("disabled");

// Acceso directo a atributos comunes
boton.id = "boton-nuevo";
boton.className = "clase1 clase2";
boton.title = "Haz clic aquí";

// dataset (acceder a data-* atributos)
console.log(boton.dataset.accion); // "guardar"
```

4. Modificar Clases

```
let elemento = document.getElementById("parrafo");

// Agregar clase
elemento.classList.add("activo");

// Eliminar clase
elemento.classList.remove("activo");
```

```
// Alternar clase
elemento.classList.toggle("activo");

// Verificar si tiene clase
if (elemento.classList.contains("activo")) {
    console.log("Tiene la clase activo");
}

// Múltiples clases
elemento.classList.add("clase1", "clase2", "clase3");
```

5. Crear y Eliminar Elementos

```
// Crear elemento
let nuevoParrafo = document.createElement("p");
nuevoParrafo.textContent = "Soy un párrafo nuevo";
nuevoParrafo.classList.add("nuevo");

// Agregar al DOM
document.body.appendChild(nuevoParrafo);

// Agregar en posición específica
document.getElementById("contenedor").insertBefore(nuevoParrafo, boton);

// innerHTML para agregar HTML
document.getElementById("contenedor").innerHTML += "<span>Nuevo span</span>";

// Eliminar elemento
let elemento = document.getElementById("parrafo");
elemento.remove(); // Forma moderna

// O con parentNode
// elemento.parentNode.removeChild(elemento);

// Clonar elemento
let clon = nuevoParrafo.cloneNode(true); // true = clonar hijos
document.body.appendChild(clon);
```

6. Ejemplo Práctico: Lista de Tareas

HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Lista de Tareas</title>
    <link rel="stylesheet" href="style.css">
</head>
```

```
<body>
    <div id="app">
        <h1>Mi Lista de Tareas</h1>
        <input type="text" id="inputTarea" placeholder="Agregar nueva tarea...">
        <button id="btnAgregar">Agregar</button>
        <ul id="listaTareas"></ul>
    </div>

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

JavaScript:

```
let tareas = [];
let inputTarea = document.getElementById("inputTarea");
let btnAgregar = document.getElementById("btnAgregar");
let listaTareas = document.getElementById("listaTareas");

// Función para agregar tarea
function agregarTarea() {
    let textoTarea = inputTarea.value.trim();

    if (textoTarea === "") {
        alert("Por favor escribe una tarea");
        return;
    }

    // Agregar a array
    tareas.push({
        id: Date.now(),
        texto: textoTarea,
        completada: false
    });

    // Limpiar input
    inputTarea.value = "";

    // Actualizar lista
    mostrarTareas();
}

// Función para mostrar tareas
function mostrarTareas() {
    listaTareas.innerHTML = "";

    tareas.forEach((tarea) => {
        let li = document.createElement("li");
        li.innerHTML = `
            <span class="${tarea.completada ? 'completada' : ''}">
                ${tarea.texto}
```

```
            </span>
            <button class="btn-completar" data-id="${tarea.id}">
                ${tarea.completada ? "Desmarcar" : "Completar"}
            </button>
            <button class="btn-eliminar" data-id="${tarea.id}">Eliminar</button>
        `;
        listaTareas.appendChild(li);
    });
}

// Función para completar tarea
function completarTarea(id) {
    tareas = tareas.map(tarea => {
        if (tarea.id === id) {
            tarea.completada = !tarea.completada;
        }
        return tarea;
    });
    mostrarTareas();
}

// Función para eliminar tarea
function eliminarTarea(id) {
    tareas = tareas.filter(tarea => tarea.id !== id);
    mostrarTareas();
}

// Event listeners
btnAgregar.addEventListener("click", agregarTarea);
inputTarea.addEventListener("keypress", (e) => {
    if (e.key === "Enter") {
        agregarTarea();
    }
});

listaTareas.addEventListener("click", (e) => {
    if (e.target.classList.contains("btn-completar")) {
        let id = parseInt(e.target.dataset.id);
        completarTarea(id);
    }
    if (e.target.classList.contains("btn-eliminar")) {
        let id = parseInt(e.target.dataset.id);
        eliminarTarea(id);
    }
});
```

🎯 Eventos

1. Event Listeners Básicos

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

// Escuchar clic
boton.addEventListener("click", () => {
    console.log("¡Fue clickeado!");
});

// Escuchar enter en input
document.getElementById("inputTarea").addEventListener("keypress", (evento) => {
    if (evento.key === "Enter") {
        console.log("Se presionó Enter");
    }
});

// Eventos de mouse
boton.addEventListener("mouseenter", () => {
    console.log("Ratón entró");
});

boton.addEventListener("mouseleave", () => {
    console.log("Ratón salió");
});

boton.addEventListener("mouseover", () => {
    console.log("Ratón encima");
});

boton.addEventListener("mouseout", () => {
    console.log("Ratón afuera");
});

// Eventos de formulario
let input = document.getElementById("inputTarea");

input.addEventListener("focus", () => {
    console.log("Input enfocado");
});

input.addEventListener("blur", () => {
    console.log("Input desenfocado");
});

input.addEventListener("input", (e) => {
    console.log("Valor:", e.target.value);
});

input.addEventListener("change", (e) => {
    console.log("Input cambió:", e.target.value);
});
```

2. Objeto Evento

```
boton.addEventListener("click", (evento) => {
    console.log(evento.target);           // El elemento que disparó el evento
    console.log(evento.type);             // "click"
    console.log(evento.key);              // Tecla presionada (si aplica)
    console.log(evento.clientX);          // Coordenada X del mouse
    console.log(evento.clientY);          // Coordenada Y del mouse
    console.log(evento.timeStamp);        // Tiempo del evento

    // Prevenir comportamiento por defecto
    evento.preventDefault();

    // Detener propagación del evento
    evento.stopPropagation();
});

// Eventos de formulario
let form = document.querySelector("form");
form.addEventListener("submit", (e) => {
    e.preventDefault(); // Evitar enviar el formulario
    console.log("Formulario enviado");
});
```

3. Event Delegation (Delegación de Eventos)

```
// En lugar de agregar listener a cada elemento...
// Agregamos a un contenedor padre

let contenedor = document.getElementById("listaTareas");

contenedor.addEventListener("click", (e) => {
    if (e.target.classList.contains("btn-eliminar")) {
        console.log("Eliminar:", e.target.dataset.id);
    }
    if (e.target.classList.contains("btn-completar")) {
        console.log("Completar:", e.target.dataset.id);
    }
});

// 💡 TIP: Más eficiente que agregar listeners a cada elemento
```

🎨 Manipulación de CSS

1. Modificar Estilos Inline

```
let elemento = document.getElementById("titulo");

// Cambiar un estilo
elemento.style.color = "red";
elemento.style.fontSize = "30px";
elemento.style.backgroundColor = "#f0f0f0";
elemento.style.padding = "20px";

// Múltiples estilos
elemento.style.cssText = `
    color: blue;
    font-size: 25px;
    background-color: yellow;
    padding: 15px;
    border-radius: 5px;
`;
```

2. Usar Clases de CSS (Método Recomendado)

style.css:

```
/* Clases base */
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    margin: 0;
    padding: 20px;
    background-color: #f5f5f5;
}

#contenedor {
    max-width: 600px;
    margin: 0 auto;
    background-color: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
}

/* Clases de estado */
.activo {
    background-color: #4CAF50;
    color: white;
    font-weight: bold;
}

.desactivado {
    background-color: #ccc;
    cursor: not-allowed;
}
```

```css
.visible {
    display: block;
    animation: fadeIn 0.3s ease-in;
}

.oculto {
    display: none;
}

.completada {
    text-decoration: line-through;
    color: #999;
}

/* Animaciones */
@keyframes fadeIn {
    from {
        opacity: 0;
    }
    to {
        opacity: 1;
    }
}

@keyframes slideDown {
    from {
        transform: translateY(-20px);
        opacity: 0;
    }
    to {
        transform: translateY(0);
        opacity: 1;
    }
}
```

JavaScript:

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

// Agregar clase
elemento.classList.add("activo");

// Eliminar clase
elemento.classList.remove("desactivado");

// Alternar clase
elemento.classList.toggle("visible");

// Verificar si tiene clase
if (elemento.classList.contains("activo")) {
    console.log("Tiene clase activo");
```

```
}

// Agregar/eliminar múltiples clases
elemento.classList.add("activo", "visible");
elemento.classList.remove("desactivado", "oculto");

// Reemplazar clase
elemento.classList.replace("oculto", "visible");
```

3. Obtener Estilos Computados

```
let elemento = document.getElementById("titulo");

// Obtener estilos calculados
let estilosCalculados = window.getComputedStyle(elemento);

console.log(estilosCalculados.color);
console.log(estilosCalculados.fontSize);
console.log(estilosCalculados.backgroundColor);
console.log(estilosCalculados.display);

// Acceso directo
console.log(window.getComputedStyle(elemento).getPropertyValue("color"));
```

4. Animaciones con JavaScript

```
// Cambio de opacidad gradual
function fadeOut(elemento, duracion = 1000) {
    let opacidad = 1;
    let velocidad = 1000 / duracion;

    let intervalo = setInterval(() => {
        opacidad -= velocidad / 100;
        elemento.style.opacity = opacidad;

        if (opacidad <= 0) {
            clearInterval(intervalo);
            elemento.style.display = "none";
        }
    }, 10);
}

function fadeIn(elemento, duracion = 1000) {
    elemento.style.display = "block";
    let opacidad = 0;
    let velocidad = 1000 / duracion;

    let intervalo = setInterval(() => {
        opacidad += velocidad / 100;
```

```
        elemento.style.opacity = opacidad;

        if (opacidad >= 1) {
            clearInterval(intervalo);
        }
    }, 10);
}

// Uso
let boton = document.getElementById("boton");
boton.addEventListener("click", () => {
    fadeOut(document.getElementById("titulo"));
});

// Usar CSS animations es más eficiente
// Agregar clase con animación CSS es la mejor práctica
```

5. Ejemplo Completo: Tema Oscuro/Claro

HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Tema Oscuro</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div id="contenedor">
        <h1>Mi Aplicación</h1>
        <p>Contenido de la aplicación</p>
        <button id="btnTema">Tema Oscuro 🌙</button>
    </div>

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

style.css:

```
body {
    transition: background-color 0.3s, color 0.3s;
    background-color: white;
    color: black;
    font-family: Arial, sans-serif;
    padding: 20px;
}
```

```css
body.dark-theme {
    background-color: #1e1e1e;
    color: white;
}

#contenedor {
    max-width: 600px;
    margin: 0 auto;
    padding: 20px;
    background-color: #f0f0f0;
    border-radius: 8px;
    transition: background-color 0.3s;
}

body.dark-theme #contenedor {
    background-color: #2d2d2d;
}

#btnTema {
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
    transition: background-color 0.3s;
}

#btnTema {
    background-color: #333;
    color: white;
}

body.dark-theme #btnTema {
    background-color: #ffb300;
    color: black;
}
```

script.js:

```js
let btnTema = document.getElementById("btnTema");
let body = document.body;

// Cargar preferencia guardada
let temaDark = localStorage.getItem("temaDark") === "true";
if (temaDark) {
    body.classList.add("dark-theme");
    btnTema.textContent = "Tema Claro 🌞";
}

// Cambiar tema
btnTema.addEventListener("click", () => {
```

```
    body.classList.toggle("dark-theme");
    temaDark = body.classList.contains("dark-theme");

    btnTema.textContent = temaDark ? "Tema Claro 🌞" : "Tema Oscuro 🌙";

    // Guardar preferencia
    localStorage.setItem("temaDark", temaDark);
});
```

💡 Mejores Prácticas

1. Separación de Responsabilidades

```
// ❌ MAL: Todo en una función
function procesarClick() {
    let valor = document.getElementById("input").value;
    // validar...
    // procesar...
    // mostrar...
    // guardar...
}

// ✅ BIEN: Funciones separadas
function validarEntrada(valor) {
    return valor.trim().length > 0;
}

function procesarDatos(valor) {
    return valor.toUpperCase();
}

function mostrarResultado(resultado) {
    document.getElementById("resultado").textContent = resultado;
}

function guardarEnLocal(clave, valor) {
    localStorage.setItem(clave, JSON.stringify(valor));
}

function procesarClick() {
    let valor = document.getElementById("input").value;

    if (!validarEntrada(valor)) {
        alert("Entrada inválida");
        return;
    }

    let resultado = procesarDatos(valor);
```

```
    mostrarResultado(resultado);
    guardarEnLocal("ultimo_resultado", resultado);
}
```

2. DRY (Don't Repeat Yourself)

```
// ❌ MAL: Código repetido
let boton1 = document.getElementById("boton1");
boton1.style.padding = "10px";
boton1.style.borderRadius = "5px";
boton1.style.cursor = "pointer";
boton1.addEventListener("click", () => console.log("1"));

let boton2 = document.getElementById("boton2");
boton2.style.padding = "10px";
boton2.style.borderRadius = "5px";
boton2.style.cursor = "pointer";
boton2.addEventListener("click", () => console.log("2"));

// ✅ BIEN: Reutilizar código
function crearBoton(id, callback) {
    let boton = document.getElementById(id);
    boton.classList.add("boton-estandar");
    boton.addEventListener("click", callback);
    return boton;
}

crearBoton("boton1", () => console.log("1"));
crearBoton("boton2", () => console.log("2"));
```

3. Usar Constantes

```
// ✅ BIEN
const API_URL = "https://api.ejemplo.com";
const TIMEOUT = 5000;
const CLAVES_LOCAL_STORAGE = {
    USUARIO: "usuario_actual",
    TEMA: "tema",
    TAREAS: "lista_tareas"
};

// Usar en todo el código
localStorage.setItem(CLAVES_LOCAL_STORAGE.TEMA, "oscuro");
```

4. Validar Entrada del Usuario

```
function validarEmail(email) {
    const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    return regex.test(email);
}

function validarPassword(password) {
    // Al menos 8 caracteres, 1 mayúscula, 1 número
    const regex = /^(?=.*[A-Z])(?=.*\d).{8,}$/;
    return regex.test(password);
}

function validarTeléfono(telefono) {
    const regex = /^[0-9]{9,}$/;
    return regex.test(telefono);
}

// Usar validaciones
let email = "usuario@ejemplo.com";
if (!validarEmail(email)) {
    console.log("Email inválido");
}
```

5. Manejo de Errores

```
// Try-Catch
function parsearJSON(texto) {
    try {
        let resultado = JSON.parse(texto);
        return resultado;
    } catch (error) {
        console.error("Error al parsear JSON:", error.message);
        return null;
    }
}

// Validación antes de procesar
function procesarArray(arr) {
    if (!Array.isArray(arr)) {
        console.error("Se esperaba un array");
        return [];
    }
    return arr.map(item => item * 2);
}
```

6. Usar localStorage para Guardar Datos

```
// Guardar
function guardarDatos(clave, datos) {
```

```
    try {
        localStorage.setItem(clave, JSON.stringify(datos));
    } catch (error) {
        console.error("Error guardando datos:", error);
    }
}

// Obtener
function obtenerDatos(clave, porDefecto = null) {
    try {
        let datos = localStorage.getItem(clave);
        return datos ? JSON.parse(datos) : porDefecto;
    } catch (error) {
        console.error("Error obteniendo datos:", error);
        return porDefecto;
    }
}

// Eliminar
function eliminarDatos(clave) {
    localStorage.removeItem(clave);
}

// Limpiar todo
function limpiarLocalStorage() {
    localStorage.clear();
}

// Uso
guardarDatos("usuario", { nombre: "Juan", edad: 30 });
let usuario = obtenerDatos("usuario");
console.log(usuario); // { nombre: "Juan", edad: 30 }
```

7. Nombres Descriptivos

```
// ❌ MAL
let d = new Date();
let x = document.getElementById("btn");
function f(a, b) {
    return a + b;
}

// ✅ BIEN
let fechaActual = new Date();
let botonAgregar = document.getElementById("btn-agregar");
function sumarNumeros(numero1, numero2) {
    return numero1 + numero2;
}
```

8. Funciones Pequeñas y Enfocadas

```
// ✅ BIEN: Cada función hace una cosa
function obtenerPrecio(producto) {
    return producto.precio;
}

function aplicarDescuento(precio, porcentaje) {
    return precio * (1 - porcentaje / 100);
}

function calcularImpuesto(precio, tasaImpuesto) {
    return precio * (1 + tasaImpuesto / 100);
}

function calcularPrecioFinal(producto, descuento, impuesto) {
    let precio = obtenerPrecio(producto);
    precio = aplicarDescuento(precio, descuento);
    precio = calcularImpuesto(precio, impuesto);
    return precio;
}
```

9. Usar Logging Apropiadamente

```
// Development vs Production
const LOG_LEVEL = {
    DEBUG: 0,
    INFO: 1,
    WARN: 2,
    ERROR: 3
};

const CURRENT_LEVEL = LOG_LEVEL.DEBUG;

function log(mensaje, nivel = LOG_LEVEL.INFO) {
    if (nivel >= CURRENT_LEVEL) {
        switch (nivel) {
            case LOG_LEVEL.DEBUG:
                console.log("%cDEBUG:", "color: blue;", mensaje);
                break;
            case LOG_LEVEL.INFO:
                console.log("%cINFO:", "color: green;", mensaje);
                break;
            case LOG_LEVEL.WARN:
                console.warn("%cWARN:", "color: orange;", mensaje);
                break;
            case LOG_LEVEL.ERROR:
                console.error("%cERROR:", "color: red;", mensaje);
                break;
        }
```

```
    }
}

// Uso
log("Aplicación iniciada", LOG_LEVEL.INFO);
log("Variable x =", LOG_LEVEL.DEBUG, { x: 5 });
```

10. Usar async/await para Operaciones Asincrónicas

```
// ❌ VIEJO: Callbacks
function obtenerDatos(callback) {
    setTimeout(() => {
        callback({ nombre: "Juan" });
    }, 1000);
}

// ❌ VIEJO: Promises
function obtenerDatos() {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            resolve({ nombre: "Juan" });
        }, 1000);
    });
}

// ✅ MODERNO: async/await
async function obtenerDatos() {
    try {
        const respuesta = await fetch("https://api.ejemplo.com/usuarios");
        const datos = await respuesta.json();
        return datos;
    } catch (error) {
        console.error("Error:", error);
    }
}

// Usar
async function main() {
    let datos = await obtenerDatos();
    console.log(datos);
}

main();
```

🚀 Proyecto Final: Aplicación de Notas

Aquí te presento una aplicación completa que integra todo lo aprendido:

index.html:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Aplicación de Notas</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="contenedor">
        <header>
            <h1>📝 Mis Notas</h1>
            <button id="btnTema" class="btn-tema">🌙 Tema Oscuro</button>
        </header>

        <main>
            <div class="form-notas">
                <input
                    type="text"
                    id="inputTitulo"
                    placeholder="Título de la nota..."
                    maxlength="50"
                >
                <textarea
                    id="inputContenido"
                    placeholder="Contenido de la nota..."
                    rows="5"
                ></textarea>
                <button id="btnGuardar" class="btn-guardar">Guardar Nota</button>
            </div>

            <div class="filtros">
                <input
                    type="text"
                    id="inputBuscar"
                    placeholder="Buscar notas..."
                >
                <select id="selectOrden">
                    <option value="reciente">Más Reciente</option>
                    <option value="antiguo">Más Antiguo</option>
                    <option value="a-z">A - Z</option>
                </select>
            </div>

            <div id="listaNotas" class="lista-notas">
                <!-- Las notas se generarán aquí -->
            </div>
        </main>
    </div>
```

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

style.css:

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #f5f5f5;
    color: #333;
    transition: background-color 0.3s, color 0.3s;
    padding: 20px;
}

body.dark-theme {
    background-color: #1e1e1e;
    color: #e0e0e0;
}

.contenedor {
    max-width: 900px;
    margin: 0 auto;
}

header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 30px;
}

header h1 {
    font-size: 2.5rem;
}

.btn-tema {
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    background-color: #ffb300;
    color: black;
    cursor: pointer;
    font-size: 1rem;
    transition: background-color 0.3s;
}
```

```css
body.dark-theme .btn-tema {
    background-color: #333;
    color: white;
}

.form-notas {
    background-color: white;
    padding: 20px;
    border-radius: 8px;
    margin-bottom: 20px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
}

body.dark-theme .form-notas {
    background-color: #2d2d2d;
    box-shadow: 0 2px 10px rgba(0,0,0,0.5);
}

.form-notas input,
.form-notas textarea {
    width: 100%;
    padding: 12px;
    margin-bottom: 15px;
    border: 2px solid #ddd;
    border-radius: 5px;
    font-family: inherit;
    font-size: 1rem;
    transition: border-color 0.3s;
}

body.dark-theme .form-notas input,
body.dark-theme .form-notas textarea {
    background-color: #3d3d3d;
    color: white;
    border-color: #555;
}

.form-notas input:focus,
.form-notas textarea:focus {
    outline: none;
    border-color: #4CAF50;
    box-shadow: 0 0 5px rgba(76, 175, 80, 0.3);
}

.btn-guardar {
    width: 100%;
    padding: 12px;
    background-color: #4CAF50;
    color: white;
    border: none;
    border-radius: 5px;
    font-size: 1rem;
```

```
    cursor: pointer;
    transition: background-color 0.3s;
    font-weight: bold;
}

.btn-guardar:hover {
    background-color: #45a049;
}

.btn-guardar:active {
    transform: scale(0.98);
}

.filtros {
    display: flex;
    gap: 15px;
    margin-bottom: 20px;
}

.filtros input,
.filtros select {
    flex: 1;
    padding: 10px;
    border: 2px solid #ddd;
    border-radius: 5px;
    font-size: 1rem;
}

body.dark-theme .filtros input,
body.dark-theme .filtros select {
    background-color: #2d2d2d;
    color: white;
    border-color: #555;
}

.lista-notas {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(280px, 1fr));
    gap: 20px;
}

.nota {
    background-color: white;
    padding: 15px;
    border-radius: 8px;
    box-shadow: 0 2px 8px rgba(0,0,0,0.1);
    transition: transform 0.3s, box-shadow 0.3s;
    animation: slideDown 0.3s ease-out;
}

body.dark-theme .nota {
    background-color: #2d2d2d;
    box-shadow: 0 2px 8px rgba(0,0,0,0.5);
```

```css
}

.nota:hover {
    transform: translateY(-5px);
    box-shadow: 0 5px 15px rgba(0,0,0,0.2);
}

.nota-titulo {
    font-size: 1.3rem;
    font-weight: bold;
    margin-bottom: 10px;
    color: #4CAF50;
    word-break: break-word;
}

.nota-contenido {
    margin-bottom: 10px;
    line-height: 1.5;
    color: #666;
    max-height: 100px;
    overflow-y: auto;
    word-break: break-word;
}

body.dark-theme .nota-contenido {
    color: #b0b0b0;
}

.nota-fecha {
    font-size: 0.9rem;
    color: #999;
    margin-bottom: 10px;
}

body.dark-theme .nota-fecha {
    color: #777;
}

.nota-botones {
    display: flex;
    gap: 10px;
}

.btn-editar,
.btn-eliminar {
    flex: 1;
    padding: 8px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 0.9rem;
    transition: background-color 0.3s;
    font-weight: bold;
```

```css
}

.btn-editar {
    background-color: #2196F3;
    color: white;
}

.btn-editar:hover {
    background-color: #0b7dda;
}

.btn-eliminar {
    background-color: #f44336;
    color: white;
}

.btn-eliminar:hover {
    background-color: #da190b;
}

.vacio {
    grid-column: 1 / -1;
    text-align: center;
    padding: 40px;
    color: #999;
    font-size: 1.1rem;
}

@keyframes slideDown {
    from {
        opacity: 0;
        transform: translateY(-20px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}

/* Modal para editar */
.modal {
    display: none;
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0,0,0,0.5);
    z-index: 1000;
    align-items: center;
    justify-content: center;
}
```

```css
.modal.activo {
    display: flex;
}

.modal-contenido {
    background-color: white;
    padding: 30px;
    border-radius: 8px;
    width: 90%;
    max-width: 500px;
    box-shadow: 0 5px 20px rgba(0,0,0,0.3);
}

body.dark-theme .modal-contenido {
    background-color: #2d2d2d;
}

.modal-contenido h2 {
    margin-bottom: 20px;
}

.modal-contenido input,
.modal-contenido textarea {
    width: 100%;
    padding: 12px;
    margin-bottom: 15px;
    border: 2px solid #ddd;
    border-radius: 5px;
}

body.dark-theme .modal-contenido input,
body.dark-theme .modal-contenido textarea {
    background-color: #3d3d3d;
    color: white;
    border-color: #555;
}

.modal-botones {
    display: flex;
    gap: 10px;
}

.modal-botones button {
    flex: 1;
    padding: 12px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-weight: bold;
    font-size: 1rem;
}

.btn-guardar-cambios {
```

```
    background-color: #4CAF50;
    color: white;
}

.btn-cancelar {
    background-color: #ccc;
    color: black;
}
```

script.js:

```
// ===== CONSTANTES Y VARIABLES =====
const CLAVES_LOCAL = {
    NOTAS: "notas_guardadas",
    TEMA: "tema_oscuro"
};

let notas = [];
let notaEnEdicion = null;

// ===== ELEMENTOS DEL DOM =====
const inputTitulo = document.getElementById("inputTitulo");
const inputContenido = document.getElementById("inputContenido");
const btnGuardar = document.getElementById("btnGuardar");
const btnTema = document.getElementById("btnTema");
const listaNotas = document.getElementById("listaNotas");
const inputBuscar = document.getElementById("inputBuscar");
const selectOrden = document.getElementById("selectOrden");

// ===== FUNCIONES DE ALMACENAMIENTO =====
function guardarEnLocal() {
    try {
        localStorage.setItem(CLAVES_LOCAL.NOTAS, JSON.stringify(notas));
    } catch (error) {
        console.error("Error guardando en localStorage:", error);
    }
}

function cargarDelLocal() {
    try {
        let notasGuardadas = localStorage.getItem(CLAVES_LOCAL.NOTAS);
        notas = notasGuardadas ? JSON.parse(notasGuardadas) : [];
    } catch (error) {
        console.error("Error cargando de localStorage:", error);
        notas = [];
    }
}

// ===== FUNCIONES DE NOTAS =====
function crearNota(titulo, contenido) {
    return {
```

```
        id: Date.now(),
        titulo: titulo.trim(),
        contenido: contenido.trim(),
        fecha: new Date().toLocaleString("es-ES")
    };
}

function agregarNota() {
    let titulo = inputTitulo.value;
    let contenido = inputContenido.value;

    if (!validarNota(titulo, contenido)) {
        return;
    }

    let nota = crearNota(titulo, contenido);
    notas.unshift(nota); // Agregar al inicio

    guardarEnLocal();
    limpiarForm();
    mostrarNotas();

    mostrarMensaje("✓ Nota guardada correctamente");
}

function validarNota(titulo, contenido) {
    titulo = titulo.trim();
    contenido = contenido.trim();

    if (titulo === "") {
        alert("Por favor, escribe un título");
        inputTitulo.focus();
        return false;
    }

    if (contenido === "") {
        alert("Por favor, escribe el contenido");
        inputContenido.focus();
        return false;
    }

    if (titulo.length > 50) {
        alert("El título no puede exceder 50 caracteres");
        return false;
    }

    return true;
}

function limpiarForm() {
    inputTitulo.value = "";
    inputContenido.value = "";
    inputTitulo.focus();
```

```
}

function eliminarNota(id) {
    if (confirm("¿Estás seguro de que deseas eliminar esta nota?")) {
        notas = notas.filter(nota => nota.id !== id);
        guardarEnLocal();
        mostrarNotas();
        mostrarMensaje("✓ Nota eliminada");
    }
}

function editarNota(id) {
    notaEnEdicion = notas.find(nota => nota.id === id);
    if (notaEnEdicion) {
        inputTitulo.value = notaEnEdicion.titulo;
        inputContenido.value = notaEnEdicion.contenido;
        inputTitulo.focus();
        btnGuardar.textContent = "Actualizar Nota";
        btnGuardar.style.backgroundColor = "#2196F3";
    }
}

function guardarCambios() {
    if (notaEnEdicion) {
        let titulo = inputTitulo.value;
        let contenido = inputContenido.value;

        if (!validarNota(titulo, contenido)) {
            return;
        }

        notaEnEdicion.titulo = titulo.trim();
        notaEnEdicion.contenido = contenido.trim();

        guardarEnLocal();
        notaEnEdicion = null;
        limpiarForm();
        btnGuardar.textContent = "Guardar Nota";
        btnGuardar.style.backgroundColor = "#4CAF50";
        mostrarNotas();

        mostrarMensaje("✓ Nota actualizada");
    } else {
        agregarNota();
    }
}

// ===== FUNCIONES DE BÚSQUEDA Y ORDENAMIENTO =====
function filtrarNotas() {
    let busqueda = inputBuscar.value.toLowerCase();
    let orden = selectOrden.value;

    let notasFiltradas = notas.filter(nota =>
```

```
        nota.titulo.toLowerCase().includes(busqueda) ||
        nota.contenido.toLowerCase().includes(busqueda)
    );

    notasFiltradas = ordenarNotas(notasFiltradas, orden);

    mostrarNotasPersonalizadas(notasFiltradas);
}

function ordenarNotas(array, orden) {
    let copia = [...array];

    switch (orden) {
        case "reciente":
            return copia.reverse();
        case "antiguo":
            return copia;
        case "a-z":
            return copia.sort((a, b) =>
                a.titulo.localeCompare(b.titulo)
            );
        default:
            return copia;
    }
}

// ===== FUNCIONES DE VISUALIZACIÓN =====
function mostrarNotas() {
    if (notas.length === 0) {
        listaNotas.innerHTML = '<div class="vacio">📭 No tienes notas aún. ¡Crea una!</div>';
        return;
    }

    filtrarNotas();
}

function mostrarNotasPersonalizadas(arrayNotas) {
    if (arrayNotas.length === 0) {
        listaNotas.innerHTML = '<div class="vacio">🔍 No se encontraron notas</div>';
        return;
    }

    listaNotas.innerHTML = arrayNotas.map(nota => `
        <div class="nota">
            <div class="nota-titulo">${escaparHTML(nota.titulo)}</div>
            <div class="nota-contenido">${escaparHTML(nota.contenido)}</div>
            <div class="nota-fecha">${nota.fecha}</div>
            <div class="nota-botones">
                <button class="btn-editar" onclick="editarNota(${nota.id})">
                    ✏️ Editar
                </button>
                <button class="btn-eliminar" onclick="eliminarNota(${nota.id})">
                    🗑️ Eliminar
```

```
                </button>
            </div>
        </div>
    `).join("");
}

function escaparHTML(texto) {
    const div = document.createElement("div");
    div.textContent = texto;
    return div.innerHTML;
}

function mostrarMensaje(mensaje) {
    // Crear elemento temporal para mensaje
    let msgElement = document.createElement("div");
    msgElement.textContent = mensaje;
    msgElement.style.cssText = `
        position: fixed;
        top: 20px;
        right: 20px;
        background-color: #4CAF50;
        color: white;
        padding: 15px 20px;
        border-radius: 5px;
        animation: slideDown 0.3s ease-out;
        z-index: 2000;
    `;
    document.body.appendChild(msgElement);

    setTimeout(() => {
        msgElement.style.animation = "fadeOut 0.3s ease-out";
        setTimeout(() => msgElement.remove(), 300);
    }, 2000);
}

// ===== FUNCIÓN DE TEMA =====
function cambiarTema() {
    document.body.classList.toggle("dark-theme");
    let esTemaOscuro = document.body.classList.contains("dark-theme");
    localStorage.setItem(CLAVES_LOCAL.TEMA, esTemaOscuro);
    btnTema.textContent = esTemaOscuro ? "🌞 Tema Claro" : "🌙 Tema Oscuro";
}

function cargarTema() {
    let temaOscuro = localStorage.getItem(CLAVES_LOCAL.TEMA) === "true";
    if (temaOscuro) {
        document.body.classList.add("dark-theme");
        btnTema.textContent = "🌞 Tema Claro";
    }
}

// ===== EVENT LISTENERS =====
btnGuardar.addEventListener("click", guardarCambios);
```

```
btnTema.addEventListener("click", cambiarTema);

inputBuscar.addEventListener("input", filtrarNotas);
selectOrden.addEventListener("change", filtrarNotas);

inputTitulo.addEventListener("keypress", (e) => {
    if (e.key === "Enter") {
        inputContenido.focus();
    }
});

inputContenido.addEventListener("keypress", (e) => {
    if (e.ctrlKey && e.key === "Enter") {
        guardarCambios();
    }
});

// ===== INICIALIZACIÓN =====
document.addEventListener("DOMContentLoaded", () => {
    cargarDelLocal();
    cargarTema();
    mostrarNotas();
});
```