Unidad 8
Ejercicios resueltos

CESUR
Tu Centro Oficial de FP

Ejercicios resueltos

EJERCICIO 1: VALIDADOR JSON

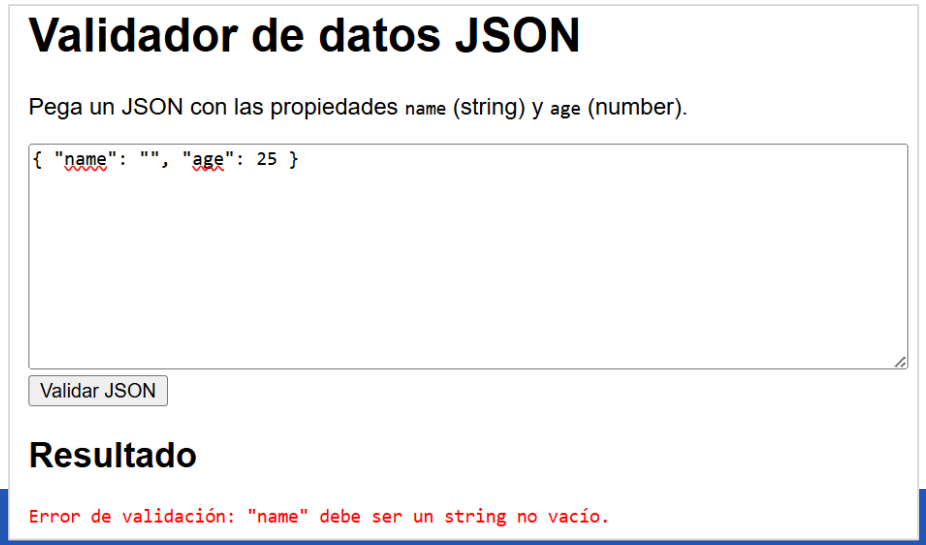
Crea una página web que pide un JSON por pantalla, lo parsea con una función y valida los datos usando gestión de errores con try…catch y excepciones personalizadas.

El try…catch en el manejador del botón diferencia entre:

- Errores de validación (datos incorrectos pero JSON bien formado).
- Errores de sintaxis (JSON mal formado).

Validaciones a considerar:

1) Json nulo
2) Falta la propiedad name o su valor
3) Falta la propiedad age o su valor
4) Age debe ser un número positivo



Ejercicios resueltos

```
<style>
  body { font-family: sans-serif; max-width: 600px; margin: 2rem auto; }
  textarea { width: 100%; height: 150px; }
  .error { color: red; }
  .ok { color: green; }
</style>
```

```
  <script>
    // Clase de error de validación personalizada
class ValidationError extends Error {
  constructor(message) {
    super(message);
    this.name = "ValidationError";
  }
}
// Función que parsea y valida el JSON
function parseAndValidateJSON(jsonString) {
  let data;
  // 1. Intentar parsear el JSON
  try {
    data = JSON.parse(jsonString); // puede lanzar SyntaxError
  } catch (e) {
    // Re-lanzar un error más descriptivo
    throw new Error("JSON inválido: " + e.message);
  }
  // 2. Validaciones adicionales
  if (typeof data !== "object" || data === null || Array.isArray(data)) {
    throw new ValidationError("El JSON debe representar un objeto no nulo.");
  }
  if (!("name" in data)) {
    throw new ValidationError('Falta la propiedad obligatoria "name".');
  }
  if (!("age" in data)) {
    throw new ValidationError('Falta la propiedad obligatoria "age".');
  }
  if (typeof data.name !== "string" || data.name.trim() === "") {
    throw new ValidationError('"name" debe ser un string no vacío.');
  }
  if (typeof data.age !== "number" || !Number.isFinite(data.age)) {
    throw new ValidationError('"age" debe ser un número.');
  }
  if (data.age < 0) {
    throw new ValidationError('"age" no puede ser negativo.');
  }
  // Si todo está bien, devolvemos el objeto ya validado
  return data;
}
```

EJERCICIO 1 – SOLUCIÓN: VALIDADOR JSON I

```
// Manejo de eventos en la página
document.addEventListener("DOMContentLoaded", () => {
  const textarea = document.getElementById("jsonInput");
  const button = document.getElementById("validateBtn");
  const result = document.getElementById("result");
  button.addEventListener("click", () => {
    const jsonText = textarea.value;
    try {
      const user = parseAndValidateJSON(jsonText);
      result.className = "ok";
      result.textContent =
        "JSON válido.\n\nObjeto parseado:\n" + JSON.stringify(user, null, 2);
    } catch (err) {
      result.className = "error";

      if (err instanceof ValidationError) {
        result.textContent = "Error de validación: " + err.message;
      } else if (err instanceof SyntaxError) {
        // Por si llegara algún SyntaxError directamente
        result.textContent = "Error de sintaxis JSON: " + err.message;
      } else {
        // Otros errores genéricos
        result.textContent = "Error: " + err.message;
      }
    }
  });
});
  </script>
</head>
<body>
  <h1>Validador de datos JSON</h1>
  <p>Pega un JSON con las propiedades <code>name</code> (string) y <code>age</code> (number).</p>
  <textarea id="jsonInput">
{ "name": "Ana", "age": 25 }
  </textarea>
  <br />
  <button id="validateBtn">Validar JSON</button>
  <h2>Resultado</h2>
  <pre id="result"></pre>
</body>
```

EJERCICIO 1 – SOLUCIÓN: VALIDADOR JSON II

Ejercicios resueltos

EJERCICIO 2:

Muestra un fondo con degradado y texto centrado.

Ejercicios resueltos

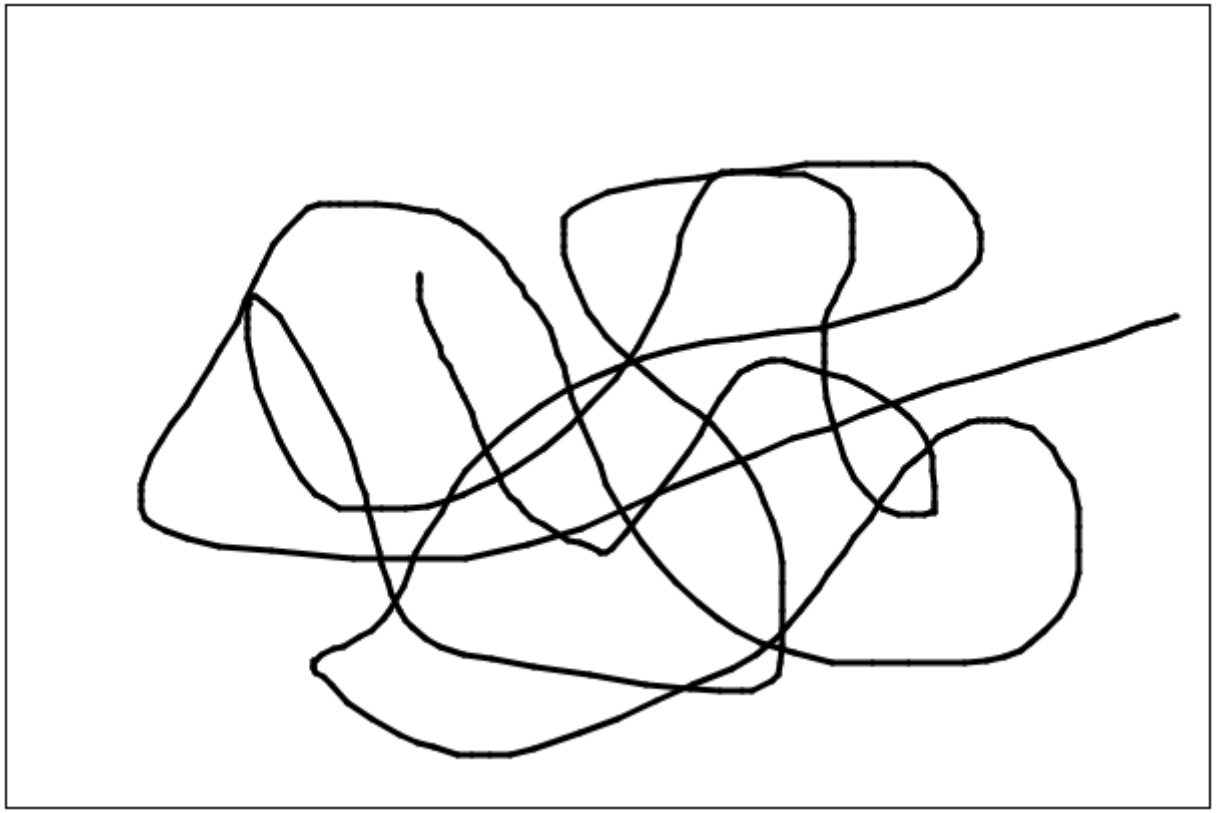
```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Texto y gradiente</title>
  <style>
    canvas { border: 1px solid #000; }
  </style>
</head>
<body>
  <canvas id="canvas" width="500" height="200"></canvas>

  <script>
    const canvas = document.getElementById("canvas");
    const ctx = canvas.getContext("2d");

    function draw() {
      // Degradado de fondo
      const grd = ctx.createLinearGradient(0, 0, canvas.width, canvas.height);
      grd.addColorStop(0, "pink");
      grd.addColorStop(0.5, "red");
      grd.addColorStop(1, "blue");

      ctx.fillStyle = grd;
      ctx.fillRect(0, 0, canvas.width, canvas.height);

      // Texto centrado
      ctx.font = "40px Arial";
      ctx.fillStyle = "white";
      ctx.textAlign = "center";
      ctx.textBaseline = "middle";
      ctx.fillText("Hola Canvas", canvas.width / 2, canvas.height / 2);
    }

    draw();
  </script>
</body>
</html>
```

EJERCICIO 2 - SOLUCIÓN

Ejercicios resueltos

EJERCICIO 3:

Mantén pulsado el ratón para dibujar líneas en el canvas.



Ejercicios resueltos

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Mini paint con Canvas</title>
  <style>
    canvas { border: 1px solid #000; cursor: crosshair; }
  </style>
</head>
<body>
  <canvas id="canvas" width="600" height="400"></canvas>
  <script>
    const canvas = document.getElementById("canvas");
    const ctx = canvas.getContext("2d");
    let painting = false;
    let lastX = 0;
    let lastY = 0;
    function startPosition(e) {
      painting = true;
      const rect = canvas.getBoundingClientRect();
      lastX = e.clientX - rect.left;
      lastY = e.clientY - rect.top;
    }
    function finishedPosition() {
      painting = false;
      ctx.beginPath(); // rompe el camino para no unir trazos
    }
    function draw(e) {
      if (!painting) return;
      const rect = canvas.getBoundingClientRect();
      const x = e.clientX - rect.left;
      const y = e.clientY - rect.top;
      ctx.lineWidth = 3;
      ctx.lineCap = "round";
      ctx.strokeStyle = "black";
      ctx.beginPath();
      ctx.moveTo(lastX, lastY);
      ctx.lineTo(x, y);
      ctx.stroke();
      lastX = x;
      lastY = y;
    }
    canvas.addEventListener("mousedown", startPosition);
    canvas.addEventListener("mouseup", finishedPosition);
    canvas.addEventListener("mouseleave", finishedPosition);
    canvas.addEventListener("mousemove", draw);
  </script>
</body>
</html>
```

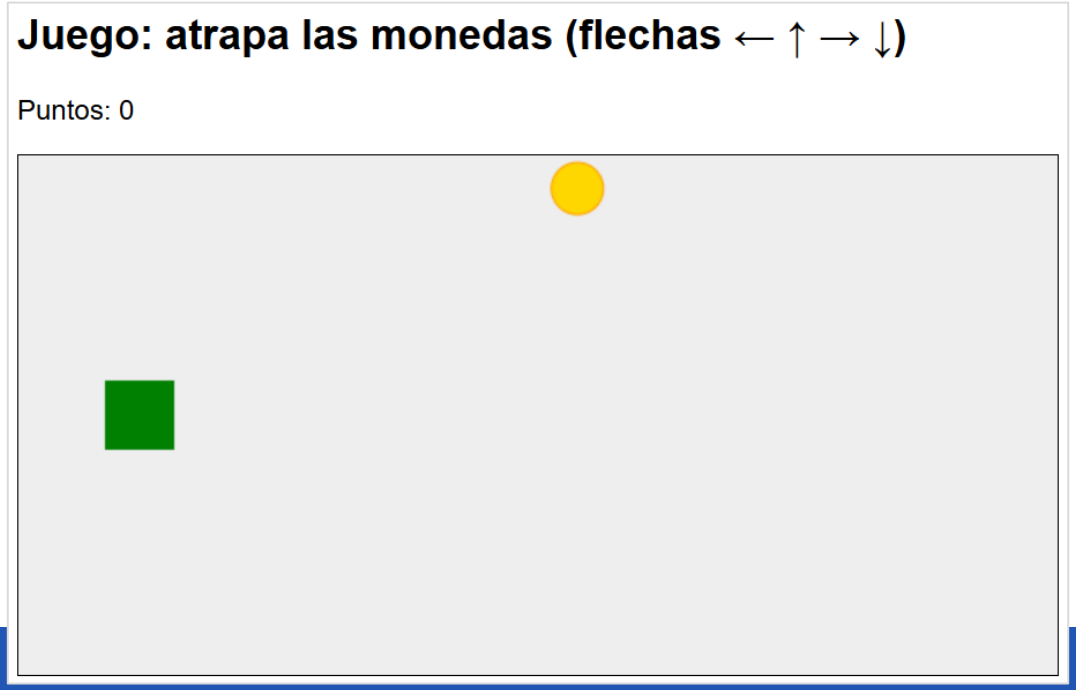
EJERCICIO 3 - SOLUCIÓN

Ejercicios resueltos

EJERCICIO 4:

Crea un juego con Canvas donde:

- El jugador es un cuadrado verde que se mueve con las flechas del teclado.
- Aparece una círculo amarillo en una posición aleatoria del canvas.
- Cuando el jugador toca la moneda, sumas 1 punto y la moneda aparece en otra posición.
- En pantalla se muestra la puntuación actual.



Ejercicios resueltos

EJERCICIO 4 – SOLUCIÓN I

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Juego: atrapar monedas</title>
  <style>
    canvas { border: 1px solid #000; background: #eee; }
    body { font-family: sans-serif; }
  </style>
</head>
<body>
  <h2>Juego: atrapa las monedas (flechas ← ↑ → ↓)</h2>
  <p>Puntos: <span id="score">0</span></p>
  <canvas id="game" width="600" height="300"></canvas>
  <script>
    const canvas = document.getElementById("game");
    const ctx = canvas.getContext("2d");
    const scoreEl = document.getElementById("score");
    const player = {
      x: 50,
      y: canvas.height / 2 - 20,
      w: 40,
      h: 40,
      speed: 4
    };
    const coin = {
      x: 0,
      y: 0,
      r: 15
    };
    let score = 0;
    const keys = {};
    // Posicionar la moneda en un lugar aleatorio
    function placeCoin() {
      coin.x = coin.r + Math.random() * (canvas.width  - coin.r * 2);
      coin.y = coin.r + Math.random() * (canvas.height - coin.r * 2);
    }
    placeCoin();
    document.addEventListener("keydown", e => {
      keys[e.key] = true;
    });
    document.addEventListener("keyup", e => {
      keys[e.key] = false;
    });
```

Ejercicios resueltos

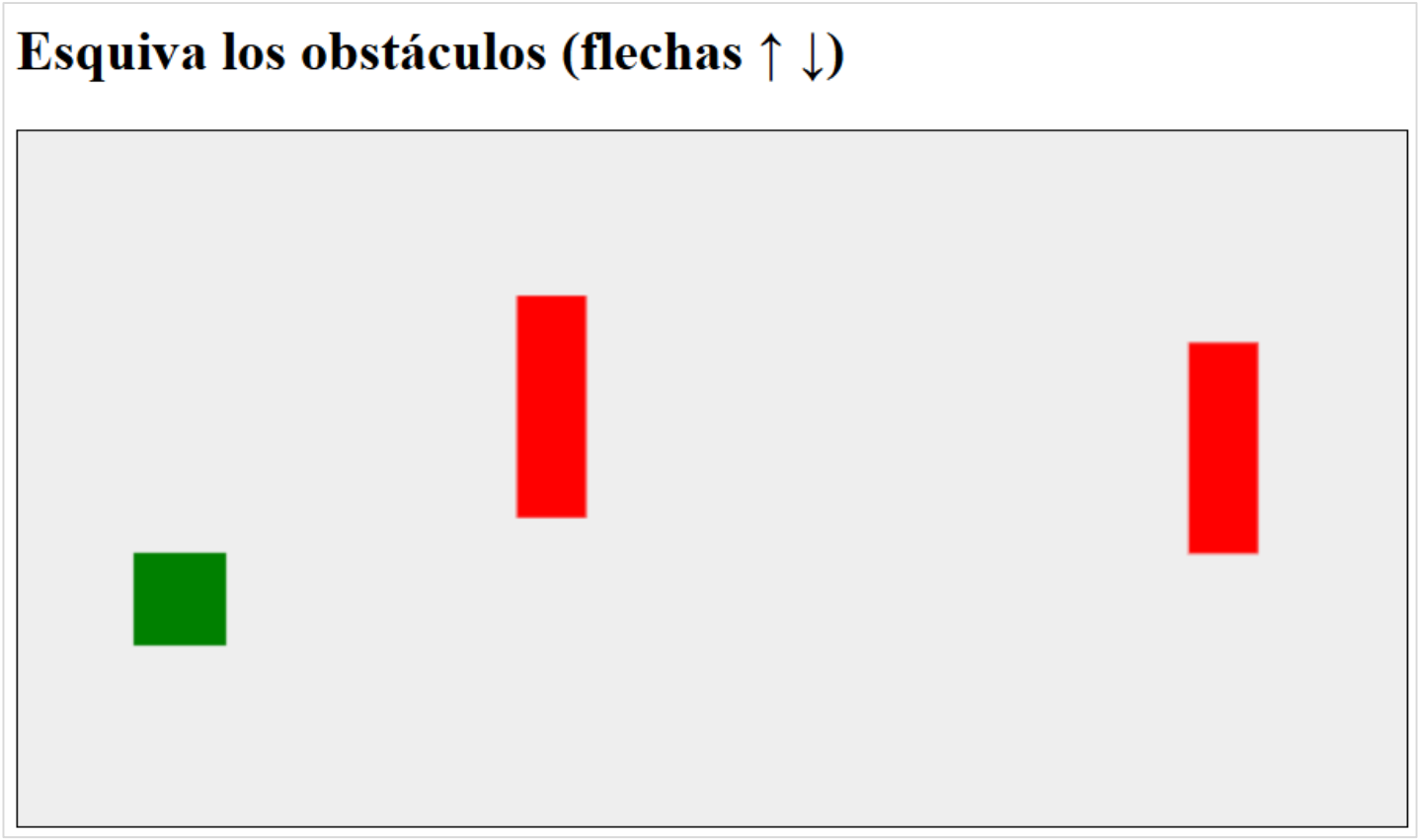
EJERCICIO 4 – SOLUCIÓN II

```
    function update() {
      // Mover jugador
      if (keys["ArrowLeft"])  player.x -= player.speed;
      if (keys["ArrowRight"]) player.x += player.speed;
      if (keys["ArrowUp"])    player.y -= player.speed;
      if (keys["ArrowDown"])  player.y += player.speed;
      // Limites
      if (player.x < 0) player.x = 0;
      if (player.y < 0) player.y = 0;
      if (player.x + player.w > canvas.width) {
        player.x = canvas.width - player.w;
      }
      if (player.y + player.h > canvas.height) {
        player.y = canvas.height - player.h;
      }
      // Colisión simple entre rectángulo (player) y círculo (coin)
      const closestX = Math.max(player.x, Math.min(coin.x, player.x + player.w));
      const closestY = Math.max(player.y, Math.min(coin.y, player.y + player.h));
      const dx = coin.x - closestX;
      const dy = coin.y - closestY;
      const distance = Math.sqrt(dx * dx + dy * dy);
      if (distance < coin.r) {
        // Ha cogido la moneda
        score++;
        scoreEl.textContent = score;
        placeCoin();
      }
    }
    function draw() {
      ctx.clearRect(0, 0, canvas.width, canvas.height);
      // Jugador
      ctx.fillStyle = "green";
      ctx.fillRect(player.x, player.y, player.w, player.h);
      // Moneda
      ctx.beginPath();
      ctx.arc(coin.x, coin.y, coin.r, 0, Math.PI * 2);
      ctx.fillStyle = "gold";
      ctx.fill();
      ctx.strokeStyle = "orange";
      ctx.stroke();
      ctx.closePath();
    }
    function loop() {
      update();
      draw();
      requestAnimationFrame(loop);
    }
    loop();
  </script>
</body>
</html>
```

Ejercicios resueltos

EJERCICIO 5:

Crea un juego en canvas: un cuadrado que esquiva obstáculos que vienen desde la derecha. Se controla con las flechas.



```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Juego: esquivar obstáculos</title>
  <style>
    canvas { border: 1px solid #000; background: #eee; }
  </style>
</head>
<body>
  <h2>Esquiva los obstáculos (flechas ↑ ↓)</h2>
  <canvas id="game" width="600" height="300"></canvas>
  <script>
    const canvas = document.getElementById("game");
    const ctx = canvas.getContext("2d");
    // Jugador
    const player = {
      x: 50,
      y: canvas.height / 2 - 20,
      w: 40,
      h: 40,
      speed: 4
    };
    // Obstáculos
    const obstacles = [];
    let frame = 0;
    let gameOver = false;
    const keys = {};
    document.addEventListener("keydown", e => {
      keys[e.key] = true;
    });
    document.addEventListener("keyup", e => {
      keys[e.key] = false;
    });
    function createObstacle() {
      const height = 40 + Math.random() * 60;
      const y = Math.random() * (canvas.height - height);
      const speed = 3 + Math.random() * 2;
      obstacles.push({
        x: canvas.width,
        y,
        w: 30,
        h: height,
        speed
      });
    }
```

```
    function update() {
      if (gameOver) return;
      // Mover jugador
      if (keys["ArrowUp"]) player.y -= player.speed;
      if (keys["ArrowDown"]) player.y += player.speed;
      // Límites
      if (player.y < 0) player.y = 0;
      if (player.y + player.h > canvas.height) {
        player.y = canvas.height - player.h;
      }
      // Crear obstáculos cada cierto tiempo
      frame++;
      if (frame % 60 === 0) {
        createObstacle();
      }
      // Mover obstáculos
      for (let i = obstacles.length - 1; i >= 0; i--) {
        const o = obstacles[i];
        o.x -= o.speed;
        // Quitar si sale de pantalla
        if (o.x + o.w < 0) {
          obstacles.splice(i, 1);
          continue;
        }
        // Colisión con el jugador
        if (
          player.x < o.x + o.w &&
          player.x + player.w > o.x &&
          player.y < o.y + o.h &&
          player.y + player.h > o.y
        ) {
          gameOver = true;
        }
      }
    }
    function draw() {
      ctx.clearRect(0, 0, canvas.width, canvas.height);
      // Dibujar jugador
      ctx.fillStyle = "green";
      ctx.fillRect(player.x, player.y, player.w, player.h);
      // Dibujar obstáculos
      ctx.fillStyle = "red";
      for (const o of obstacles) {
        ctx.fillRect(o.x, o.y, o.w, o.h);
      }
      // Texto de Game Over
      if (gameOver) {
        ctx.fillStyle = "black";
        ctx.font = "40px Arial";
        ctx.textAlign = "center";
        ctx.fillText("GAME OVER", canvas.width / 2, canvas.height / 2);
        ctx.font = "20px Arial";
        ctx.fillText(
          "Recarga la página para volver a jugar",
          canvas.width / 2,
          canvas.height / 2 + 40
        );
      }
    }
    function loop() {
      update();
      draw();
      requestAnimationFrame(loop);
    }
    loop();
  </script>
</body>
</html>
```

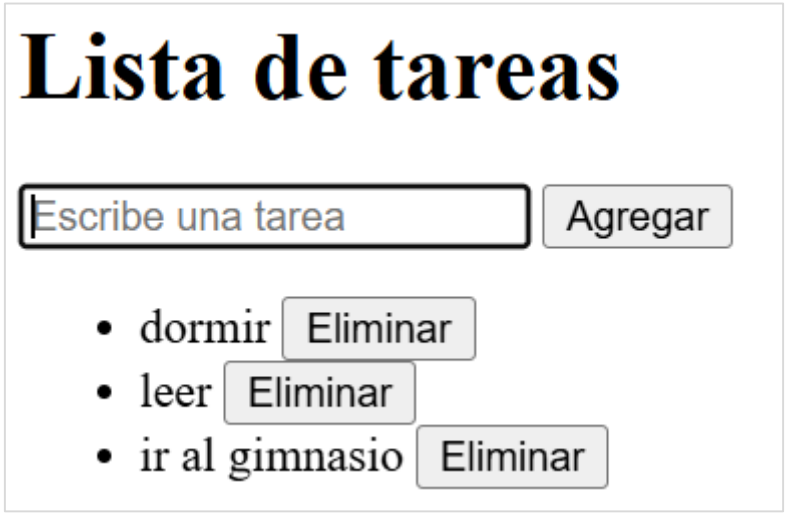
EJERCICIO 5 - SOLUCIÓN

Ejercicios resueltos

EJERCICIO 6:

Crea una pequeña aplicación de lista de tareas donde el usuario pueda:

- Escribir una tarea en un input.
- Añadirla a una lista que se muestra en pantalla.
- Guardar la lista en localStorage para que, al recargar la página, las tareas sigan apareciendo.



Ejercicios resueltos

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <title>Lista de tareas con Web Storage</title>
</head>
<body>
  <h1>Lista de tareas</h1>

  <input id="tareaInput" type="text" placeholder="Escribe una tarea" />
  <button id="btnAgregar">Agregar</button>

  <ul id="listaTareas"></ul>

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

EJERCICIO 6 – SOLUCIÓN I

Ejercicios resueltos

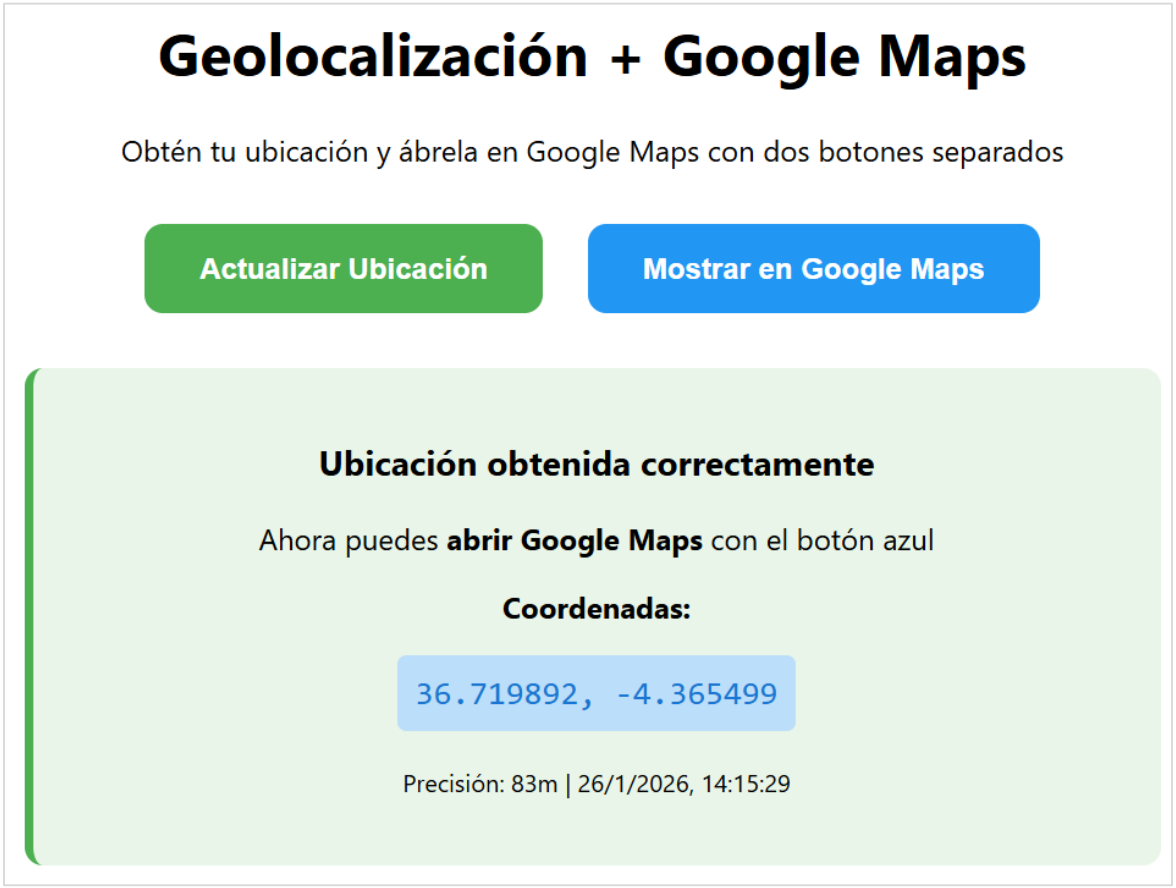
```
// Claves que usaremos en localStorage
const CLAVE_TAREAS = "tareas";
// Referencias a elementos del DOM
const inputTarea = document.getElementById("tareaInput");
const btnAgregar = document.getElementById("btnAgregar");
const listaTareas = document.getElementById("listaTareas");
// 1. Cargar tareas guardadas al iniciar
function cargarTareas() {
  // Obtenemos la cadena del localStorage y la convertimos a array
  const tareasGuardadas = JSON.parse(localStorage.getItem(CLAVE_TAREAS)) || [];
  tareasGuardadas.forEach(texto => {
    crearElementoTarea(texto);
  });
}
// 2. Guardar array de tareas en localStorage
function guardarTareas() {
  const tareas = [];
  // Recorremos los <li> actuales para reconstruir el array
  listaTareas.querySelectorAll("li").forEach(li => {
    tareas.push(li.firstChild.textContent); // el texto está en el primer nodo de texto
  });
  localStorage.setItem(CLAVE_TAREAS, JSON.stringify(tareas));
}
// 3. Crear un <li> con la tarea y botón eliminar
function crearElementoTarea(texto) {
  const li = document.createElement("li");
  li.textContent = texto + " ";
  const btnEliminar = document.createElement("button");
  btnEliminar.textContent = "Eliminar";
  btnEliminar.addEventListener("click", () => {
    listaTareas.removeChild(li);
    guardarTareas(); // actualizamos localStorage
  });
  li.appendChild(btnEliminar);
  listaTareas.appendChild(li);
}
// 4. Manejar el clic en "Agregar"
btnAgregar.addEventListener("click", () => {
  const texto = inputTarea.value.trim();
  if (texto === "") return;
  crearElementoTarea(texto);
  guardarTareas();     // guardamos la lista actualizada
  inputTarea.value = ""; // limpiamos el input
  inputTarea.focus();
});
// 5. Cargar las tareas al cargar la página
cargarTareas();
```

EJERCICIO 6 – SOLUCIÓN II

Ejercicios resueltos

EJERCICIO 7:

Crea un ejercicio con API Geolocation que incluya un botón para obtener la localización del usuario y que incluya un botón para mostrar la localización en GoogleMaps



```html
<body>
    <div class="container">
        <h1>Geolocalización + Google Maps</h1>
        <p>Obtén tu ubicación y ábrela en Google Maps con dos botones separados</p>

        <div class="btn-group">
            <button id="getLocation">Obtener Ubicación</button>
            <button id="showMap" disabled>Mostrar en Google Maps</button>
        </div>

        <div id="resultado"></div>
    </div>
    <script>
        let coordenadasActuales = null;
        const btnGetLocation = document.getElementById('getLocation');
        const btnShowMap = document.getElementById('showMap');
        const resultado = document.getElementById('resultado');
        // Botón 1: Obtener ubicación
        btnGetLocation.addEventListener('click', () => {
            if (!navigator.geolocation) {
                mostrarError('Geolocalización no soportada por este navegador');
                return;
            }
            btnGetLocation.textContent = 'Obteniendo...';
            btnGetLocation.disabled = true;
            resultado.style.display = 'none';
            const opciones = {
                enableHighAccuracy: true,
                timeout: 10000,
                maximumAge: 0
            };
            navigator.geolocation.getCurrentPosition(
                posicionExito,
                posicionError,
                opciones
            );
        });
        // Botón 2: Mostrar en Google Maps
        btnShowMap.addEventListener('click', () => {
            if (!coordenadasActuales) {
                mostrarError('Primero obtén tu ubicación');
                return;
            }
            const { lat, lng } = coordenadasActuales;
            const urlGoogleMaps = `https://www.google.com/maps?q=${lat},${lng}&ll=${lat},${lng}&z=16`;
            // Abrir en nueva pestaña
            window.open(urlGoogleMaps, '_blank');
        });
```

```html
<style>
    body {
        font-family: 'Segoe UI', sans-serif;
    }
    .container {
        background: white; padding: 40px;
        text-align: center;
    }
    .btn-group { margin: 30px 0; }
    button {
        padding: 15px 30px; margin: 0 10px;
        font-size: 16px; border: none; border-radius: 10px;
        cursor: pointer; transition: all 0.3s;
        font-weight: bold;
    }
    #getLocation {
        background: #4CAF50; color: white;
    }
    #getLocation:hover:not(:disabled) { background: #45a049; }
    #showMap {
        background: #2196F3; color: white;
    }
    #showMap:hover:not(:disabled) { background: #1976D2; }
    button:disabled {
        opacity: 0.5; cursor: not-allowed;
    }
    #resultado {
        margin-top: 30px; padding: 20px;
        background: #e8f5e8; border-radius: 10px;
        border-left: 5px solid #4CAF50; display: none;
    }
    .coords {
        font-family: monospace; font-size: 18px;
        color: #1976D2; background: #bbdefb;
        padding: 10px; border-radius: 5px; display: inline-block;
    }
    .error {
        background: #ffebee; border-left-color: #f44336;
        color: #c62828;
    }
</style>
```

EJERCICIO 7 – SOLUCIÓN I

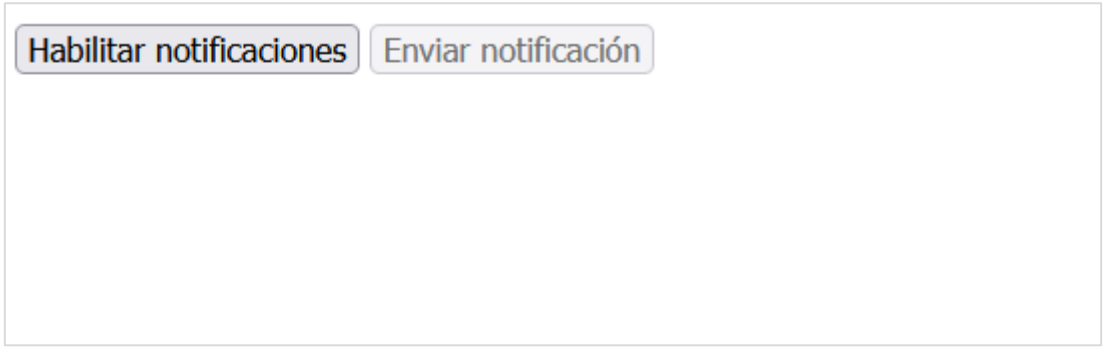
resueltos

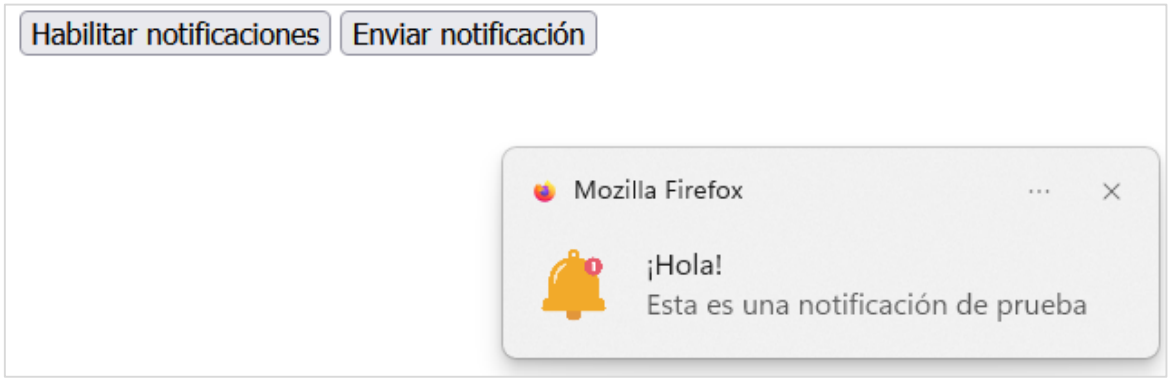
```
97          function posicionExito(position) {
98              const { latitude, longitude, accuracy } = position.coords;
99              coordenadasActuales = { lat: latitude, lng: longitude };
100             // Habilitar botón de Google Maps
101             btnShowMap.disabled = false;
102             resultado.innerHTML = `
103                 <h3>Ubicación obtenida correctamente</h3>
104                 <p>Ahora puedes <strong>abrir Google Maps</strong> con el botón azul</p>
105                 <div>
106                     <p><strong>Coordenadas:</strong></p>
107                     <div class="coords">${latitude.toFixed(6)}, ${longitude.toFixed(6)}</div>
108                 </div>
109                 <p><small>Precisión: ${Math.round(accuracy)}m | ${new Date(position.timestamp).toLocaleString()}</small></p>
110             `;
111             resultado.className = '';
112             resultado.style.display = 'block';
113             btnGetLocation.textContent = 'Actualizar Ubicación';
114             btnGetLocation.disabled = false;
115         }
116         function posicionError(error) {
117             let mensaje = '';
118             switch(error.code) {
119                 case error.PERMISSION_DENIED:
120                     mensaje = 'Permiso denegado. Habilita la ubicación en tu navegador.';
121                     break;
122                 case error.POSITION_UNAVAILABLE:
123                     mensaje = 'Ubicación no disponible.';
124                     break;
125                 case error.TIMEOUT:
126                     mensaje = 'Tiempo agotado. Inténtalo de nuevo.';
127                     break;
128                 default:
129                     mensaje = 'Error desconocido.';
130             }
131             mostrarError(mensaje);
132         }
133         function mostrarError(mensaje) {
134             resultado.innerHTML = `<p>${mensaje}</p>`;
135             resultado.className = 'error';
136             resultado.style.display = 'block';
137             btnGetLocation.textContent = 'Obtener Ubicación';
138             btnGetLocation.disabled = false;
139             btnShowMap.disabled = true;
140         }
141     </script>
142 </body>
143 </html>
```

EJERCICIO 7 – SOLUCIÓN II

EJERCICIO 8: NOTIFICACIONES

Crea una página que incluya dos botones uno que solicite activar las notificaciones y el otro para que envíe una notificación.
Inicialmente sólo está activado el de habilitar notificaciones, en el momento que se habiliten, se activa el botón de enviar notificación.





```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Ejercicio API Notification</title>
</head>
<body>
    <button id="enable">Habilitar notificaciones</button>
    <button id="notify">Enviar notificación</button>

    <script>
        const enableBtn = document.getElementById('enable');
        const notifyBtn = document.getElementById('notify');

        function askNotificationPermission() {
            Notification.requestPermission().then(function(permission) {
                if (permission === 'granted') {
                    console.log('Permiso concedido');
                    notifyBtn.disabled = false;
                } else {
                    console.log('Permiso denegado');
                }
            });
        }

        function sendNotification() {
            if (Notification.permission === 'granted') {
                new Notification('¡Hola!', {
                    body: 'Esta es una notificación de prueba',
                    icon: 'icono.png' // Asegúrate de tener un archivo icono.png en la misma carpeta
                });
            }
        }

        enableBtn.addEventListener('click', askNotificationPermission);
        notifyBtn.addEventListener('click', sendNotification);
        notifyBtn.disabled = true;
    </script>
</body>
</html>
```

EJERCICIO 8 - SOLUCIÓN

Gracias