ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS
INTRODUCCIÓN AL DESARROLLO WEB
2025 II
LABORATORIO 16 – JAVASCRIPT: EVENTOS Y JSON

I

OBJETIVOS

- Comprender la importancia de JavaScript en el desarrollo web
- Comprender los fundamentos de la orientación a objetos en JavaScript
- Solucionar ejercicios aplicando los conocimientos obtenidos

TIEMPO ESTIMADO: 2 horas

II

CONSIDERACIONES DE EVALUACIÓN

- Se deberán utilizar los conocimientos impartidos en las clases teóricas
- Deberá utilizar nombre de variables significativos
- Deberá realizar pruebas adicionales
- El alumno deberá indicar en su código con quien colaboró, así sea la IA
- El alumno será requerido de realizar modificaciones en su código y responder a preguntas sobre el mismo
- Los ejercicios deberán realizarse en el laboratorio, a su ritmo y subirlos al aula virtual como AVANCE antes de finalizar la clase
- Todos los ejercicios completos, deberán ser subidos al aula virtual como TAREA, para lo cual se les dará un tiempo adecuado y deben cumplir el deadline estipulado. Esto se deberá cumplir, aunque en el laboratorio ya se hayan terminado todos los ejercicios
- El formato a usar para los avances y tareas será .zip, que contenga todos los archivos requeridos
- Utilizar Git con GitHub para el manejo de versiones de su trabajo, ocasionalmente se le solicitará que compartan sus repositorios

III

POLITICA DE COLABORACION

La política del curso es simple, a menos que se exprese lo contrario en el laboratorio, siéntase libre de colaborar con sus compañeros en todos los laboratorios, pero debe notificar expresamente con quien ha colaborado. La colaboración con alumnos, que no están matriculados en el curso está prohibida. Los laboratorios y asignaciones han sido desarrollados para ayudarlo a comprender el material. Conozca su código y esté preparado para revisiones individuales de código. Durante las revisiones es probable que se le pida realizar modificaciones y justificar sus decisiones de programación. Cada uno de sus ejercicios debe iniciar de la siguiente forma:

```
// Laboratorio  Nro x - Ejerciciox
// Autor: mi nombre
// Colaboró : el nombre
// Tiempo :
```

INDICACIONES GENERALES

a. En cada sesión de laboratorio, los ejercicios propuestos deberán ser guardados en la misma carpeta/directorio
b. La carpeta deberá tener el nombre del Laboratorio y el nombre del alumno, así por ejemplo:
Laboratorio 11 – Juan Perez
c. Utilice nombres significativos
d. Su código deberá estar correctamente indentado y preferentemente documentado
e. Deberá ser debidamente probado

MARCO TEORICO

1. ¿Qué es un evento en el DOM?

Un evento es una acción que ocurre en la página web y que el navegador puede detectar, tales como clics, pulsaciones de teclas, movimiento del ratón, carga del documento, etc.

El modelo de eventos del DOM permite ejecutar código JavaScript en respuesta a esas acciones.

Ejemplo:

```
let boton = document.querySelector("#miBoton");
boton.addEventListener("click", () => {
  console.log("Se hizo clic en el botón");
});
```

2. Manejo de eventos

Existen tres formas principales de asignar eventos:

1) Atributo HTML (no recomendado):
```
<button onclick="alert('Hola JS')">Haz clic</button>
```

2) Propiedad del objeto (en JS):
```
boton.onclick = function() {
  alert("Hola JS");
};
```

3) addEventListener() (forma moderna y recomendada) (en JS):
```
boton.addEventListener("click", () => {
  alert("Hola JS");
});
```

3. Sintaxis general

```
element.addEventListener(tipo, funcion [, opciones]);
```

- tipo: nombre del evento (click, input, submit, etc.).
- función: código que se ejecuta al dispararse el evento.
- opciones (opcional): configuración

4. Tipos de eventos frecuentes

Categoría	Ejemplos
Mouse	click, dblclick, mouseover, mouseout
Teclado	keydown, keyup, keypress
Formulario	submit, change, focus, blur
Ventana / documento	load, resize, scroll
Entrada de datos	input, select

5. Métodos comunes

Método / Propiedad	Descripción
addEventListener(evento, función)	Asocia un evento a un elemento
removeEventListener(evento, función)	Quita un evento asociado
event.target	Devuelve el elemento que disparó el evento
preventDefault()	Evita el comportamiento por defecto (por ejemplo, envío de formulario)
stopPropagation()	Detiene la propagación del evento hacia padres

6. Propiedades del objeto evento

Propiedad / Método	Descripción
event.type	Tipo de evento (ej. "click")
event.target	Elemento que originó el evento
event.currentTarget	Elemento que maneja el evento
event.preventDefault()	Evita el comportamiento por defecto
event.stopPropagation()	Detiene la propagación del evento
event.clientX, event.clientY	Posición del mouse

7. Delegación de eventos

Permite manejar varios elementos similares con un único addEventListener en su contenedor.
Ejemplo:

```
document.querySelector("ul").addEventListener("click", (e) => {
  if (e.target.tagName === "li") {
    e.target.classList.toggle("resaltado");
  }
});
```

8. Ejemplos

1) Cuando el usuario hace clic, se ejecuta la función que cambia el texto del párrafo:

HTML

```
<button id="btnSaludo">Haz clic</button>
<p id="mensaje">Esperando clic...</p>
```

JS

```
const boton = document.getElementById("btnSaludo");
const mensaje = document.getElementById("mensaje");

boton.addEventListener("click", () => {
  mensaje.textContent = "¡Hola! Hiciste clic en el botón.";
});
```

2) El evento input se ejecuta cada vez que cambia el valor del campo de texto, permitiendo responder en tiempo real. Se usa mucho en validaciones o autocompletados:

HTML

```
<label>Escribe tu nombre: </label>
<input type="text" id="nombre">
<p id="saludo"></p>
```

JS

```
const input = document.getElementById("nombre");
const saludo = document.getElementById("saludo");

input.addEventListener("input", () => {
  saludo.textContent = `Hola, ${input.value}!`;
});
```

3) Cambiar color al pasar el mouse

HTML

```
<div id="caja">
  Pasa el mouse
</div>
```

JS

```
const caja = document.getElementById("caja");

caja.addEventListener("mouseover", () => {
  caja.style.backgroundColor = "skyblue";
});

caja.addEventListener("mouseout", () => {
  caja.style.backgroundColor = "lightgray";
});
```

4) Mostrar alerta al enviar un formulario

HTML

```
<form id="formulario">
  <input type="text" placeholder="Tu nombre">
  <button type="submit">Enviar</button>
</form>
```

JS

```
const form = document.getElementById("formulario");

form.addEventListener("submit", (event) => {
  event.preventDefault(); // evita recargar la página
  alert("Formulario enviado correctamente");
});
```

5) Delegación en lista

HTML

```
<ul id="lista">
  <li>Manzana</li>
  <li>Plátano</li>
  <li>Naranja</li>
</ul>
```

CSS

```
.resaltado {
  background-color: yellow;
}
```

JS

```
const lista = document.getElementById("lista");

lista.addEventListener("click", (event) => {
  if (event.target.tagName === "li") {
    event.target.classList.toggle("resaltado"); // alterna la clase
  }
});
```

9. JSON (JavaScript Object Notation)

JSON es un formato de intercambio de datos ligero, basado en texto. Permite representar objetos y estructuras de datos en un formato legible y fácil de transmitir entre cliente y servidor.
Es una cadena de texto con una estructura similar a los objetos de JavaScript, pero con requisitos más estrictos.

10. Sintaxis básica de JSON

Un JSON contiene pares clave–valor entre llaves { } o listas entre corchetes [ ].

```
{
  "nombre": "Ana",
  "edad": 25,
  "activo": true,
  "hobbies": ["leer", "viajar", "dibujar"]
}
```

Reglas:

- Las claves van entre comillas dobles
- Los valores pueden ser: texto, número, booleano, null, objeto o arreglo
- No se permiten comentarios ni comas al final
- No permite funciones

11. Conversión entre JSON y JavaScript:

Método	Descripción	Ejemplo
JSON.stringify(obj)	Convierte un objeto JS a cadena JSON	let texto = JSON.stringify(persona);
JSON.parse(cadena)	Convierte una cadena JSON a objeto JS	let obj = JSON.parse(texto);

Convertir un objeto JS → JSON:

```
let persona = { nombre: "Ana", edad: 25 };
let textoJSON = JSON.stringify(persona);
console.log(textoJSON); // '{"nombre":"Ana","edad":25}'
```

Convertir JSON → objeto JS

```
let cadena = '{"nombre":"Ana","edad":25}';
let obj = JSON.parse(cadena);
console.log(obj.nombre); // "Ana"
```

12. Persistencia con localStorage

Se puede hacer almacenamiento persistente del objeto JSON.

Ejemplo:
Convertir objeto a JSON (para guardar de forma persistente) y luego recuperarlo

```
const usuario = { nombre: "Carlos", nivel: "Avanzado", puntos: 2500 };

// Convertir objeto a JSON (para guardar)
const jsonUsuario = JSON.stringify(usuario);
localStorage.setItem("usuario", jsonUsuario);

// Recuperar y volver a objeto
const datos = JSON.parse(localStorage.getItem("usuario"));
console.log(datos.nombre); // "Carlos"
```

13. Usos comunes de JSON

- Enviar datos a un servidor (por ejemplo con fetch())
- Almacenar configuraciones o registros en localStorage
- Cargar datos externos (archivos .json)
- Intercambio entre APIs y front-end

VI

EJERCICIOS PROPUESTOS

1. Crear un directorio que tenga su nombre y un subdirectorio laboratorio16. Recomendación: usar minúsculas, sin espacios, sin tildes ni “ñ” y con guiones medios o bajos
2. Utilizar los atajos de teclado o combinación de teclas para agilizar su trabajo

Crear el html y js que manipule el DOM para cada ejercicio:

3. Cambiar contenido dinámicamente: crear un botón que al presionarlo cambie el texto de un párrafo, de “Texto original” a “Texto cambiado”
4. Anterior ejercicio, pero que al volver a hacer click vuelva al texto anterior (alternadamente)
5. Manipular clases CSS: que un botón active/desactive una clase “oscuro” en body para simular modo oscuro
6. Contador interactivo: crear botones “+”, “–” y un span que muestre el valor actual (empezar en 0)
   Evita que el contador baje de cero y mostrar mensaje
7. Agregar y eliminar elementos de una lista: permitir al usuario ingresar texto en un <input> y añadirlo como <li> dentro de una lista. Incluir un botón para borrar el último elemento
8. Validar formulario con DOM: validar que los campos “nombre” y “correo” no estén vacíos. Si hay error, mostrar mensajes debajo del input usando createElement("span") con estilo rojo
9. Galería con miniaturas: crear una galería donde al hacer clic en una miniatura (img) se muestre la imagen grande en un principal
10. Tabla dinámica desde JavaScript: generar una tabla HTML al presionar un botón, a partir de un arreglo de objetos

```
const productos = [
  { nombre: "Laptop", precio: 3500 },
  { nombre: "Mouse", precio: 80 },
];
```

11. Delegación de eventos: crear una lista <ul> que permita eliminar cualquier <li> al hacer clic sobre él, sin asignar eventos individualmente. Usa event.target dentro del listener del
12. Animación con DOM y CSS: hacer que al presionar un botón, un cuadrado ( ) se mueva horizontalmente usando classList.add("mover") y una animación CSS. Agrega un botón “Reiniciar” que quite la clase
13. Construir una pequeña interfaz CRUD (Crear, Leer, Actualizar y Eliminar)
    - Formulario para añadir usuarios (nombre, edad)
    - Tabla que muestra los usuarios agregados
    - Botones de editar y eliminar por fila
    - Usa querySelector, appendChild, dataset y event delegation
14. Declara un objeto persona con nombre, edad y ciudad. Convierte el objeto a JSON con JSON.stringify() y muéstralo en consola
15. Declara una cadena JSON y conviértela en objeto con JSON.parse(). Muestra uno de sus valores en el DOM (por ejemplo, dentro de un <p>)
16. Crea un arreglo de objetos productos (nombre, precio). Convierte a JSON y luego vuelve a objeto. Muestra los nombres en una lista <ul> generada dinámicamente
17. Guarda un objeto JSON con datos de usuario (nombre, correo, rol) en localStorage. Luego recupéralo y muéstralo en pantalla
18. Define una variable con un JSON que simule una lista de libros. Parsea el JSON y genera una tabla con sus títulos y autores usando el DOM
19. Convierte un objeto JSON en objeto JS, modifica un valor (por ejemplo: edad o precio), y vuelve a convertirlo a JSON actualizado
20. Cuando el usuario escriba su nombre en un <input> y haga clic en un botón, guarda los datos en un objeto, conviértelo a JSON y muéstralo en consola
21. Define un JSON con una lista de tareas (título, completada). Crea dinámicamente una lista en el DOM que muestre el estado de cada tarea (por ejemplo, con color verde si está completada)
22. Simulador de perfil
    - Crea un formulario con nombre, edad y país
    - Al enviar, guarda los datos como JSON en localStorage
    - Si el usuario recarga la página, muestra el perfil guardado en pantalla
23. Crear un repositorio remoto en GitHub y subir tu repositorio local. Compartir URL y pdf con captura de pantalla del código de los archivos js y de la ejecución

I. TAREA PARA LA CASA: Complete todos los ejercicios.

Crear un documento docx/pdf con la solución de los ejercicios.

Subir el documento a la tarea **Tarea 16** del Aula Virtual respetando las fechas indicadas.