PROCESO DIRECCIÓN DE FORMACIÓN PROFESIONAL INTEGRAL CENTRO TECNOLOGIA Y MANOFACTURA AVANZADA
GUIA DE TRABAJO # 1

Objetivo de la clase: Que el estudiante comprenda y aplique técnicas nativas de JavaScript para manipular elementos en pantalla mediante Drag & Drop, evaluar intersecciones espaciales (colisiones) y controlar la reproducción de medios auditivos, integrando estas herramientas en interfaces dinámicas.

Módulo 1: HTML5 Drag & Drop API

La API de Drag & Drop de HTML5 permite a los usuarios hacer clic y arrastrar elementos hacia áreas de colocación específicas (drop zones).

- **Preparación del elemento:** Para que un elemento sea arrastrable, debe tener el atributo HTML draggable="true".
- **El objeto dataTransfer:** Es el núcleo de la operación. Permite guardar la información del elemento que se está arrastrando para luego recuperarla en el momento de soltarlo.
- **Eventos clave:**
  - dragstart: Se dispara al iniciar el arrastre. Aquí se define qué datos se transfieren (e.dataTransfer.setData).
  - dragover: Se dispara continuamente cuando un elemento arrastrable está sobre una zona de colocación. **Importante:** Por defecto, el navegador no permite soltar elementos. Se debe llamar a e.preventDefault() en este evento para habilitar el drop.
  - drop: Se dispara cuando se suelta el elemento. Aquí se procesa la acción (e.dataTransfer.getData).

El secreto del Drag & Drop (e.preventDefault): Por defecto, los navegadores están programados para *no* permitir que se suelten elementos arbitrarios en cualquier parte de una página (si arrastras una imagen a una pestaña, el navegador intenta abrirla, no moverla de div). Por eso, en los eventos dragover y drop, es **obligatorio** usar event.preventDefault(). Esto le dice al navegador: *"Desactiva tu comportamiento por defecto, yo me encargo de gestionar lo que pasa al soltar el elemento"*.

Ejemplo

El Cubo Viajero

Demostración de cómo mover un cuadro de un contenedor a otro.

HTML

```
<div id="origen" class="caja" ondragover="permitirDrop(event)" ondrop="soltar(event)">
  <div id="item" draggable="true" ondragstart="arrastrar(event)" style="background: red; width: 50px; height: 50px;"></div>
</div>
<div id="destino" class="caja" ondragover="permitirDrop(event)" ondrop="soltar(event)" style="border: 2px dashed black; width: 100px; height: 100px;"></div>

<!-- JavaScript -->
<script>
  function arrastrar(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
```

```
 }
 function permitirDrop(ev) {
   ev.preventDefault(); // Permite que se pueda soltar
 }
 function soltar(ev) {
   ev.preventDefault();
   let data = ev.dataTransfer.getData("text");
   ev.target.appendChild(document.getElementById(data));
 }
</script>
```

Ejercicio Práctico 1

Instrucciones: Crea una interfaz con tres lenguajes de programación (cajas arrastrables con los textos: "PHP", "JavaScript", "Python") y dos áreas de destino ("Frontend" y "Backend"). El estudiante debe programar el arrastre libre de estos elementos hacia cualquiera de las cajas destino.

Módulo 2: Detección de Colisiones 2D

En el desarrollo web (fuera de Canvas), la detección de colisiones suele basarse en el método **AABB (Axis-Aligned Bounding Box)**. Consiste en evaluar si las "cajas" rectangulares que envuelven a dos elementos HTML se superponen.

- **getBoundingClientRect():** Este método devuelve el tamaño de un elemento y su posición relativa al viewport (top, right, bottom, left, width, height).
- **Lógica de Intersección:** Dos rectángulos (A y B) **no** colisionan si:
  - La parte inferior de A está por encima de la parte superior de B.
  - La parte superior de A está por debajo de la parte inferior de B.
  - La parte derecha de A está a la izquierda de la parte izquierda de B.
  - La parte izquierda de A está a la derecha de la parte derecha de B.
- Si *ninguna* de esas condiciones es verdadera, entonces hay una colisión.

La matemática de las Colisiones (AABB): El método getBoundingClientRect() dibuja una "caja invisible" alrededor del elemento HTML. La lógica que usamos no busca "cuándo se tocan", sino que hace lo opuesto: **comprueba cuándo es físicamente imposible que se estén tocando** (ej. si mi lado derecho está más a la izquierda que tu lado izquierdo, no nos tocamos). Si todas esas comprobaciones de "imposibilidad" son falsas, la única conclusión matemática es que hay una intersección.

Ejemplo

Evaluador de Superposición Estática

JavaScript

```
function verificarColision(elemento1, elemento2) {
  const rect1 = elemento1.getBoundingClientRect();
  const rect2 = elemento2.getBoundingClientRect();

  const colision = !(
    rect1.top > rect2.bottom ||
    rect1.right < rect2.left ||
    rect1.bottom < rect2.top ||
    rect1.left > rect2.right
  );

  if (colision) {
    console.log("¡Impacto detectado!");
  }
}
```

Ejercicio Práctico 2

Instrucciones: Crea dos elementos posicionados de forma absoluta. Usa botones (Arriba, Abajo, Izquierda, Derecha) para alterar las propiedades top y left del primer div. En cada movimiento, ejecuta una función que verifique si toca al segundo div. Si colisionan, el segundo div debe cambiar su color de fondo a rojo.

Módulo 3: Control de Sonido Multimedia

El control de sonido dinámico es esencial para dar feedback en interfaces interactivas. Usaremos la API nativa de Audio de HTML5 controlada por JavaScript.

- **El objeto Audio:** Permite instanciar sonidos dinámicamente desde JS sin necesidad de etiquetas <audio> en el HTML: const miSonido = new Audio('ruta.mp3');.
- **Métodos principales:**
  - .play(): Inicia la reproducción (devuelve una Promesa).
  - .pause(): Detiene la reproducción temporalmente.
- **Propiedades útiles:**
  - .currentTime: Establece o devuelve la posición actual (en segundos). Para reiniciar un efecto de sonido y que pueda sonar rápido varias veces seguidas, se usa sonido.currentTime = 0;.
  - .volume: Rango de 0.0 (silencio) a 1.0 (máximo).
  - **Políticas de Autoplay en Audio**: Es crucial enseñarles que los navegadores modernos (Chrome, Safari) bloquean el audio si no hay una **interacción previa del usuario** (un clic o tocar una tecla). Por eso, no podemos hacer que una página suene nada más cargar; el sonido siempre debe desencadenarse a partir de un evento del DOM (click, keydown).

Ejemplo

Efecto de Clic de Interfaz

JavaScript

```
const sonidoClick = new Audio('clic.mp3'); // Asumir que el archivo existe
const boton = document.getElementById('btnAccion');

boton.addEventListener('click', () => {
  sonidoClick.currentTime = 0; // Reinicia por si el usuario hace clics muy rápidos
  sonidoClick.play();
});
```

Ejercicio Práctico

Instrucciones: Construye una alerta en pantalla simulando la llegada de un nuevo registro a una base de datos. Cuando el usuario presione un botón "Simular Nuevo Registro", debe aparecer un mensaje temporal en la interfaz y al mismo tiempo reproducirse un sonido corto. Incluye un control <input type="range"> que permita ajustar el volumen del sonido antes de que se reproduzca.

Librerías No Nativas (Herramientas de la Industria)

Es muy valioso para los estudiantes saber que, aunque entender el código nativo (Vanilla JS) es obligatorio para la lógica, en el mundo laboral se suelen utilizar librerías que resuelven problemas complejos (como el soporte para pantallas táctiles en móviles o animaciones fluidas).

Aquí tienes las más recomendadas para que se las menciones o las integres en proyectos futuros:

1. Para Drag & Drop

- **SortableJS:** Es la librería "Vanilla" más popular. Es increíblemente ligera, no requiere dependencias y maneja listas reordenables, arrastre entre diferentes listas, y tiene soporte nativo para pantallas táctiles (algo que la API nativa de HTML5 hace terriblemente mal en celulares).
- **dnd-kit (o React Beautiful DnD):** Si en tu plan de estudios llegas a tocar frameworks modernos como React, estas son las librerías estándar en la industria. dnd-kit es muy modular y excelente para crear tableros complejos, listas de tareas o reordenamiento de filas en tablas de bases de datos.

2. Para Detección de Colisiones y Físicas

- **Matter.js:** Es un motor de físicas 2D completo para la web. En lugar de calcular rectángulos manualmente, tú le dices a Matter.js "crea un círculo y dale gravedad", y la librería se encarga de que rebote, choque y ruede de manera hiperrealista. Excelente para interactividad muy avanzada.
- **Kaboom.js o Phaser:** Si el objetivo de las colisiones tiene una orientación hacia la gamificación de la interfaz, estas librerías facilitan la creación de escenarios donde los elementos colisionan, se destruyen o rebotan de forma nativa.

3. Para Control de Sonido Avanzado

- **Howler.js:** Trabajar con la API nativa de Audio puede traer dolores de cabeza cuando se manejan muchos sonidos al mismo tiempo o se cruzan navegadores antiguos. Howler.js es la solución definitiva de la industria: carga los audios más rápido, maneja "sprites de audio" (un solo archivo grande con muchos soniditos adentro para no saturar la red) y tiene control espacial (sonido 3D).