Día 1 — Ejercicio explicado a nivel granular

Incluye explicación detallada de métodos y parámetros (addEventListener, querySelector, callbacks).

Objetivo: Conectar JavaScript a la página, ubicar un elemento del DOM y responder al clic del botón ‘Ordenar ahora’.

Ficha técnica — Instrucciones y parámetros clave

• Element.addEventListener(tipoEvento, listener, opciones?)
  - ¿Qué es? Método de EventTarget (Element, Document, Window) para registrar un manejador de eventos.
  - Parámetros:
    • tipoEvento: String (p. ej., "click", "input", "submit").
    • listener: Función callback que recibe el objeto Event (p. ej., MouseEvent). Firma típica: function (event) { ... }.
    • opciones (opcional): Boolean (useCapture, obsoleto) o un objeto con { capture: boolean, once: boolean, passive: boolean, signal: AbortSignal }.
  - Retorno: undefined. Se pueden registrar múltiples listeners del mismo tipo sobre el mismo elemento.
  - Nota: once=true ejecuta la función una sola vez; passive=true indica que no llamará preventDefault() en eventos de scroll/touch.

• document.querySelector(selectorCSS)
  - ¿Qué es? Devuelve el primer elemento que coincide con el selector CSS dado; si no hay coincidencia, devuelve null.
  - Parámetros: selectorCSS (String), por ejemplo ".btn-ordenar", "#id", "header nav a".
  - Retorno: Element | null.

• function onClick(event) { ... } (callback nombrado)
  - ¿Qué es? Declaración de una función de callback con nombre. Nombrarla ayuda al depurar (stack traces legibles).
  - Parámetros: event (Event o subtipo, p. ej. MouseEvent). Contiene info del evento (target, timeStamp, etc.).
  - Retorno: undefined (salvo que explicitamente retorne algo, lo cual no usa addEventListener).

index.html — Enlazar JavaScript al final del <body>

```
<!-- ... resto de tu HTML ... -->

  <!-- Enlace a nuestro JavaScript: ponerlo al final del body asegura que el DOM ya existe -->
  <script src="js/app.js"></script>
</body>
</html>
```

js/app.js — Código explicado línea por línea

```
// 1) console.log(...) — permite imprimir en la consola del navegador para verificar conexión.
console.log("app.js CARGADO: si ves esto, JS está funcionando");

// 2) document.querySelector(".btn-ordenar")
//    - Busca en el DOM el primer elemento que coincida con el selector CSS ".btn-ordenar".
//    - Parámetro: selectorCSS (String). Retorno: Element | null.
const botonOrdenar = document.querySelector(".btn-ordenar");

// 3) Validación defensiva: si no existe, no seguimos para evitar errores al registrar el evento.
if (!botonOrdenar) {
  // Nada que hacer; la página no tiene el botón aún.
  // return termina el bloque 'if' y salta el registro de eventos.
```

```
} else {
  // 4) addEventListener("click", function onClick(event) { ... })
  //    - tipoEvento: "click" (String) → evento de ratón o tap.
  //    - listener:   función callback que se ejecutará cuando ocurra el evento.
  //    - event:      objeto MouseEvent con información del click (target, coords, etc.).
  botonOrdenar.addEventListener("click", function onClick(event) {
    // 5) alert(mensaje) — muestra un diálogo modal con texto. Útil como prueba rápida de
wiring.
    alert("¡Gracias! Pronto agregaremos el formulario de pedido.");
  });
}

// Alternativa robusta: esperar a DOMContentLoaded si el script se carga en <head>
// document.addEventListener("DOMContentLoaded", function onReady() {
//   const boton = document.querySelector(".btn-ordenar");
//   if (!boton) return;
//   boton.addEventListener("click", function onClick(e) {
//     alert("¡Listo! (DOMContentLoaded)");
//   });
// });
```