Día 5 — Ejercicio explicado a nivel granular

Explicación línea por línea y parámetros/retornos de cada instrucción clave.

Objetivo: Mejorar micro■UX: desactivar el botón mientras 'envía', simular espera, resetear formulario y resaltar promo.

Ficha técnica — Instrucciones y parámetros clave

• form.addEventListener("submit", async (event) => { ... })
  - Listener asíncrono para poder usar 'await'.
• await new Promise(r => setTimeout(r, ms))
  - Simula latencia de servidor; 'ms' en milisegundos.
• submitBtn.disabled: boolean
  - true mientras 'envía' para prevenir múltiples envíos.
• submitBtn.textContent: string
  - Feedback textual del estado del proceso.
• form.reset(): void
  - Restablece todos los controles del formulario.
• msg.focus(): void
  - Llama el foco al mensaje para cierre perceptible (accesibilidad).

index.html — Fragmento con comentarios línea a línea

```
<!-- Línea 1: Formulario mínimo para demo de micro■UX -->
<form id="form-pedido">
  <button type="submit">Enviar pedido</button>
</form>
<p id="msg-resultado" aria-live="polite"></p>
<div id="promo">2×1 en bebidas calientes (16:00–18:00)</div>
```

js/app.js — Código explicado línea por línea

```
// Línea 1: Espera a que el DOM esté listo (opcional si el script va al final del body)
document.addEventListener("DOMContentLoaded", () => {
  const form  = document.getElementById("form-pedido");       // HTMLFormElement|null
  const msg   = document.getElementById("msg-resultado");     // HTMLElement|null
  const promo = document.getElementById("promo");             // HTMLElement|null

  if (!form) return; // defensa: si no hay formulario, no hay flujo

  // Línea 2: Listener 'submit' ASÍNCRONO para poder usar 'await'
  form.addEventListener("submit", async (event) => {
    event.preventDefault(); // no recargues la página

    // Línea 3: Identificar y bloquear el botón de envío
    const submitBtn = form.querySelector("button[type='submit']"); // HTMLButtonElement|null
    submitBtn.disabled = true;             // boolean: true → bloqueado
    submitBtn.textContent = "Enviando…";  // estado textual para el usuario

    // Línea 4: Simula latencia: new Promise(resolve => setTimeout(resolve, 1200))
    await new Promise(r => setTimeout(r, 1200));

    // Línea 5: Mensaje final y accesibilidad (llevar foco al feedback)
    msg.textContent = "¡Pedido enviado! Gracias.";
    msg.tabIndex = -1;  // temporalmente enfocable
    msg.focus?.();      // si el navegador lo permite

    // Línea 6: Restaurar estado inicial
    form.reset();
    submitBtn.disabled = false;
    submitBtn.textContent = "Enviar pedido";
  });

  // Línea 7: Micro-feedback visual: resaltar promo si existe
```

```
  promo?.classList.add("resaltado"); // añade clase CSS
});
```