EJERCICIOS UT7 — ASINCRONÍA (SIN BACKEND)

EJERCICIO 1 — CARGAR MOVIMIENTOS DESDE JSON (FETCH + ASYNC/AWAIT)

CONTEXTO

Tienes un archivo `movimientos.json` con una lista de movimientos de ajedrez.

ENUNCIADO

Al cargar la página:

1. Usa `fetch` y `async/await` para cargar el archivo `movimientos.json`.
2. Guarda los movimientos en un estado (`useState`).
3. Muestra en pantalla la lista de movimientos en un `<ul>`.
4. Mientras se cargan los datos, muestra el texto:
   "Cargando movimientos…"

SE EVALÚA

- Uso correcto de `useEffect`
- Manejo de estado
- Renderizado dinámico
- Separación lógica / render

EJERCICIO 2 — BUSCADOR DE MOVIMIENTOS (AJAX SIN BACKEND)

CONTEXTO

Usando los movimientos cargados del ejercicio anterior.

ENUNCIADO

1. Añade un `<input>` para buscar movimientos.
2. Cada vez que cambie el texto:
   - Filtra los movimientos que contengan ese texto.
3. Muestra solo los resultados filtrados.

No vuelvas a hacer `fetch`. Usa el estado.

SE EVALÚA

- `useEffect` con dependencias
- Pensar en estado
- Renderizado reactivo

EJERCICIO 3 — MANEJO DE ERRORES EN FETCH

CONTEXTO

Simula que el archivo puede fallar.

ENUNCIADO

1. Intenta cargar un archivo JSON.
2. Si ocurre un error:
    - Muestra un mensaje en rojo:
      “Error cargando los datos”
3. Si todo va bien, muestra el contenido.

SE EVALÚA

- `try / catch`
- Control de errores
- Renderizado condicional

EJERCICIO 4 — MOSTRAR INFORMACIÓN SEGÚN EL TURNO (USEEFFECT)

CONTEXTO

El turno cambia entre `"blancas"` y `"negras"`.

ENUNCIADO

1. Guarda el turno en un estado.
2. Usa `useEffect` para:
    - Mostrar un mensaje automático:
      “Turno de las blancas” / “Turno de las negras”

3. Cambia el turno con un botón.

SE EVALÚA

- `useEffect` sin fetch
- Relación estado → interfaz
- Comprensión del ciclo de vida

EJERCICIO 5 — CARGA CONDICIONAL DE DATOS

CONTEXTO

Solo se cargan movimientos cuando el usuario lo pide.

ENUNCIADO

1. No cargues datos al inicio.
2. Añade un botón: **"Cargar movimientos"**
3. Al pulsarlo:
    - Ejecuta `fetch`
    - Muestra los movimientos
4. Deshabilita el botón después de cargar.

SE EVALÚA

- Control de cuándo ocurre la asincronía
- Buen uso del estado
- Evitar efectos innecesarios

EJERCICIO 6 — INDICADOR DE CARGA Y DATOS REALES

CONTEXTO

Simula una carga lenta.

ENUNCIADO

1. Añade un estado `loading`.
2. Mientras se cargan los datos:
    - Muestra "Cargando…"
3. Cuando termine:
    - Oculta el mensaje
    - Muestra los datos

Puedes usar `setTimeout` para simular retardo.

EJERCICIO 7 — USEEFFECT COMO OBSERVADOR

CONTEXTO

Cada vez que se hace un movimiento, se quiere mostrar un mensaje.

ENUNCIADO

1. Guarda el último movimiento en un estado.
2. Usa `useEffect` para:
    - Mostrar un mensaje automático cuando cambie.
3. No llames a la función desde el botón directamente.

SE EVALÚA

- Mentalidad React
- Separación de responsabilidades
- Evitar código imperativo

EJERCICIO 8 — FORMATO DE DATOS JSON

CONTEXTO

Tienes este formato:

```
[
  { "from": "e2", "to": "e4", "piece": "pawn" }
]
```

ENUNCIADO

1. Carga el JSON.
2. Transforma los datos para mostrarlos como:
   “Peón de e2 a e4”
3. Muestra la lista transformada.

SE EVALÚA

- Comprensión de formatos
- Procesado de datos
- Lógica previa al render

EJERCICIO 9 — COMPARACIÓN AJAX CLÁSICO VS FETCH

ENUNCIADO

1. Implementa la carga de un JSON:
    - Una vez con `XMLHttpRequest`
    - Otra con `fetch`
2. Muestra el mismo resultado.
3. Comenta brevemente cuál es más legible.

SE EVALÚA

- Conocer AJAX
- Comparar tecnologías
- Capacidad crítica

EJERCICIO 10 — INTEGRACIÓN ASINCRONÍA + REACT (AJEDREZ)

CONTEXTO

Un historial de movimientos que se carga desde JSON.

ENUNCIADO

1. Carga los movimientos al iniciar.
2. Muestra el historial.
3. Permite ocultarlo / mostrarlo con un botón.
4. Usa estado y renderizado condicional.

SE EVALÚA

- Visión global
- React + asincronía
- Código limpio y mantenible