Actividades unidad 5: Gestión de eventos en JavaScript II

1. Delegación de eventos en una tabla dinámica

Una aplicación interna muestra un listado de usuarios en una tabla. Cada fila tiene botones de acción.
Dado el **Ej1.html** realiza las siguientes tareas:

1. Añade un solo listener al <tbody> usando delegación de eventos.
2. Detecta si se ha pulsado Editar o Borrar usando event.target.
3. Muestra por consola (console.log) el nombre del usuario afectado.
4. Explica en un comentario por qué event.currentTarget no es el botón.

2. Diferencia entre target y currentTarget

En un panel de administración hay bloques clicables con contenido interno.
Dado el **Ej2.html** realiza las siguientes tareas:

1. Añade un listener a cada .card.
2. Al hacer clic en el botón, muestra por consola:
    - event.target.tagName.
    - event.currentTarget.className.
3. Cambia el fondo solo de la card pulsada (modificando estilos o clases CSS en el DOM).
4. Justifica en un comentario por qué no se debe usar event.target para cambiar el estilo.

3. Captura vs burbujeo en un menú desplegable

Un menú corporativo necesita controlar el orden de ejecución de eventos.
Dado el **Ej3.html** realiza las siguientes tareas:

1. Añade un listener al <nav> en fase de captura.
2. Añade otro listener a los <li> en fase de burbujeo.
3. Haz clic en "Web" y muestra por consola el orden real de ejecución.
4. Usa comentarios para explicar el flujo del evento.

4. Formulario con validación por delegación

Un formulario se valida dinámicamente mientras el usuario escribe.
Dado el **Ej4.html** realiza las siguientes tareas:

1. Añade un único listener input al formulario (no uno por cada input).
2. Detecta qué campo se está modificando usando event.target.name.
3. Valida:
    - Nombre: mínimo 3 caracteres.
    - Password: mínimo 6 caracteres.
4. Marca en rojo solo el input inválido (modificando clases CSS en el DOM).

5. Gestión de botones en una lista de tareas

Una app tipo ToDo usa delegación para manejar tareas creadas dinámicamente.
Dado el **Ej5.html** realiza las siguientes tareas:

1. Añade un único listener al elemento <ul id="tareas"> usando delegación de eventos.
2. Si se pulsa el botón ✔, marca la tarea como completada, realizando un cambio visual en el DOM (por ejemplo, cambiando el color del texto o añadiendo una clase CSS).
3. Si se pulsa el botón ✖, elimina la tarea correspondiente del DOM.
4. Al pulsar el botón Nueva tarea, crea dinámicamente una nueva tarea dentro del <ul> con la misma estructura HTML que las existentes.
5. Comprueba que los botones ✔ y ✖ funcionan correctamente también en las tareas creadas dinámicamente sin añadir nuevos listeners.

6. Justifica, mediante un comentario en el código, por qué el listener no se pierde al crear nuevas tareas y qué papel tiene event.currentTarget.

6. Prevención de eventos no deseados

En un dashboard hay enlaces dentro de tarjetas clicables.
Dado el **Ej6.html** realiza las siguientes tareas:

1. Añade un listener a la tarjeta (.card) que muestre por consola el mensaje:"Card pulsada".
2. Añade un listener al enlace <a> que muestre por consola el mensaje:"Enlace pulsado".
3. Al hacer clic en el enlace, evita que se ejecute el evento asociado a la card. Para ello, usa event.stopPropagation().
4. Explica, mediante un comentario en el código, para qué se utiliza event.stopPropagation() y qué problema resuelve en este ejercicio.

7. Simulación de sistema de notificaciones

Un sistema muestra notificaciones clicables.
Dado el **Ej7.html** realiza las siguientes tareas:

1. Usa delegación de eventos en #notificaciones.
2. Al hacer clic en una notificación (no en el botón “Cerrar”), muestra por consola el texto visible de la notificación (por ejemplo "Nuevo mensaje 1").
3. Al hacer clic en el botón Cerrar, elimina solo esa notificación del DOM.
4. Usa closest() junto con event.target para identificar correctamente la notificación padre del botón pulsado.

8. Auditoría de eventos

Se necesita una utilidad de depuración para saber dónde se producen los clics.
Dado el **Ej8.html** realiza las siguientes tareas:

1. Añade un listener en fase de captura a #app.
2. Registra por consola:
    - event.target
    - event.currentTarget
    - Fase del evento.
3. Explica en un comentario para qué sería útil este sistema en un entorno real.
4. Indica en un comentario qué cambiaría si el listener estuviera en burbujeo.