Relación E

Manejo básico de eventos

Nota

Esta relación trabaja el modelo de eventos del navegador: tipos de eventos habituales, formas de asignar manejadores (atributos HTML, propiedad `on*`, `addEventListener`), el objeto `Event`, propagación y delegación de eventos.

Ejercicio 1. Cree una página con un botón y asigne un manejador al evento `click` de tres formas distintas:

a) Directamente en el atributo HTML (`onclick="..."`).
b) Asignando una función a la propiedad `onclick` del elemento desde JavaScript.
c) Usando `addEventListener('click', ...)`.

Cada manejador debe mostrar un mensaje diferente para identificar cuál se ha disparado. A continuación, en una segunda versión de la página, asigne **dos manejadores distintos** al mismo evento `click` usando `addEventListener` y compruebe que ambos se ejecutan. ¿Ocurre lo mismo con los métodos a) y b)? Coméntelo.

Ejercicio 2. Cree una función JS `mostrarInfo(evento)` que reciba un objeto `Event` y muestre por consola la siguiente información extraída de él:

- Tipo de evento (`type`).
- Elemento que disparó el evento (`target.tagName` e `target.id`).
- Elemento al que está asociado el manejador (`currentTarget.tagName`).
- Coordenadas del ratón si es un evento de ratón.
- Tecla pulsada si es un evento de teclado.

Asóciele esta función como manejador a: un `click` sobre un párrafo, un `mouseover` sobre un `<div>` y un `keydown` sobre un campo de texto.

Ejercicio 3. Cree una página con 5 párrafos. Usando un único bucle (o `forEach`), asigne a cada párrafo un manejador del evento `click` que muestre en un `alert` el texto del párrafo pulsado. A continuación, cree una segunda versión usando **delegación de eventos**: un único manejador asignado al contenedor padre que, al recibir el `click`, determine sobre qué párrafo se hizo clic mediante `event.target` y actúe en consecuencia.

Ejercicio 4. Explore los eventos de teclado (`keydown`, `keyup`, `keypress` —este último obsoleto, pero conviene conocerlo—). Cree un campo de texto y asigne manejadores para los tres. En cada manejador, muestre por consola las propiedades `key`, `code` y `keyCode` del evento. Cree además una versión en la que, mientras el usuario escribe en el campo, se cuente en tiempo real el número de caracteres y se muestre junto al campo.

Ejercicio 5. Explore los eventos de ratón: `mouseenter`, `mouseleave`, `mousemove`, `mousedown`, `mouseup` y `dblclick`. Cree un `<div>` de tamaño apreciable (use CSS inline) y asigne manejadores para cada uno de los eventos anteriores. Cada manejador debe registrar en consola el nombre del evento y, para `mousemove`, las coordenadas relativas al elemento (`offsetX`,

`offsetY`). Muestre también en un párrafo las coordenadas actuales del ratón dentro del `<div>`.

Ejercicio 6. Cree una página con un formulario que tenga al menos un campo de texto y un `select`. Trabaje los siguientes eventos de formulario:

- `focus` y `blur` en el campo de texto: cambie el color de borde al enfocar y al desenfocar.
- `input` en el campo de texto: actualice en tiempo real un párrafo adyacente con el contenido actual del campo.
- `change` en el `select`: muestre en consola la opción seleccionada.
- `submit` en el formulario: prevenga el comportamiento por defecto con `event.preventDefault()` y muestre los valores del formulario en un `alert`.

Ejercicio 7. Estudie la propagación de eventos (*bubbling* y *capturing*). Cree tres `<div>` anidados (abuelo, padre, hijo) con estilos que los diferencien visualmente. Asigne un manejador `click` a cada uno. Compruebe en consola el orden de propagación. A continuación:

- Use `event.stopPropagation()` en el hijo y compruebe el efecto.
- Use el tercer argumento de `addEventListener` para activar el modo *capture* y observe el cambio en el orden.

Ejercicio 8. Cree una lista de tareas básica con la siguiente funcionalidad:

- Un campo de texto y un botón "Añadir".
- Al hacer clic en el botón, se añade un nuevo `<li>` a la lista con el texto del campo.
- Cada `<li>` debe tener un botón "Eliminar" que, al pulsarlo, elimine ese elemento de la lista (use delegación de eventos o manejadores individuales).
- Al hacer doble clic sobre un `<li>`, el texto debe quedar tachado (`text-decoration: line-through`) para indicar que la tarea está completada.

Ejercicio 9. Cree una función `unaVez(handler)` que reciba una función manejadora y devuelva una nueva función que ejecute `handler` solo la primera vez y no haga nada en las siguientes llamadas. Pruébela asignando al `click` de un botón un manejador que muestre un `alert`. Compruebe que el alert solo aparece en el primer clic. Compare con el uso nativo de `addEventListener` con la opción `{ once: true }`.

Ejercicio 10. Cree una página con un formulario de búsqueda (campo de texto y botón). Implemente un efecto de *debounce*: en lugar de procesar cada pulsación de tecla, espere a que el usuario haya dejado de escribir durante al menos 500 ms antes de lanzar una "búsqueda" (puede simularse con un `console.log` del término buscado). Use `setTimeout` y `clearTimeout` para implementar el debounce manualmente.

Ejercicio 11. Cree una página con un elemento que responda a los eventos de arrastrar (`dragstart`, `drag`, `dragend`) y una zona de destino que responda a `dragover` y `drop`. Al soltar el elemento en la zona de destino, muévalo al DOM de dicha zona. Use `event.preventDefault()` donde sea necesario. Muestre mensajes en consola para cada fase del arrastre.

Ejercicio 12. Explore el objeto `CustomEvent` para crear eventos personalizados. Cree un

evento llamado `"datosActualizados"` que lleve en su propiedad `detail` un objeto con un array de datos estadísticos. Dispárelo manualmente con `dispatchEvent` sobre un elemento del DOM. Asigne a ese mismo elemento un manejador que, al recibirlo, genere una tabla en el documento con los datos del `detail`.

Ejercicio 13. Cree una página con una galería de al menos 6 `<div>` de colores dispuestos en cuadrícula. Usando delegación de eventos sobre el contenedor:

- Al hacer clic en un div, se selecciona (añade clase `"seleccionado"` y un borde visible).
- Al hacer clic en un div ya seleccionado, se deselecciona.
- Se debe mantener un contador visible en la página del número de divs actualmente seleccionados.
- Al pasar el ratón por encima de un div, se muestra su índice de posición en el contenedor (use `Array.from` e `indexOf`).