

INDICE

| | |
|--------------------|---|
| INTRODUCCIÓN | 1 |
| EJERCICIO..... | 3 |
| Código:..... | 3 |
| Explicación: | 3 |
| Resultado: | 3 |
| PREGUNTAS..... | 4 |

INTRODUCCIÓN

Ejercicio 3:

Ejercicio: Análisis del flujo de eventos (captura y burbujeo) en tu formulario

Partiendo del formulario que diseñaste en el ejercicio 1 del tema (el formulario que contiene tus campos de entrada, botones, etc), realiza un estudio práctico sobre cómo funciona el flujo de eventos en el DOM, comparando:

Captura (useCapture = true)

Burbujeo (useCapture = false)

Tareas a realizar:

Añade tres listeners al formulario que ya creaste:

Uno en el elemento form.

Uno en un contenedor interno (por ejemplo un <div> que agrupe varios campos).

Uno en un elemento hijo, por ejemplo un <input> o un <button>.

A cada uno añádele el mismo evento (por ejemplo "click"), pero cambia el tercer parámetro de addEventListener():

En el formulario, usa captura (tercer parámetro true).

En el contenedor, usa burbujeo (tercer parámetro false).

En el elemento hijo, prueba ambos valores: primero con true, luego con false, observando la diferencia.

Dentro de cada listener, muestra en la consola:

El nombre del elemento que recibió el listener.

El valor actual de event.eventPhase (1 = captura, 2 = target, 3 = burbujeo).

El orden en el que se disparan los eventos.

Prueba haciendo clic en distintos puntos del formulario y anota el orden de ejecución.

Responde a las siguientes preguntas en un breve informe:

¿Qué listeners se ejecutan primero en la fase de captura?

¿Cuándo actúa exactamente el "target"?

¿Qué listeners se ejecutan en la fase de burbujeo?

¿Cómo cambia el comportamiento cuando alteras el valor true/false del addEventListener() del elemento hijo?

¿Qué efecto tiene event.stopPropagation() si lo aplicas en uno de los listeners?

Conclusión final

Explica con tus palabras la diferencia entre:

La fase de captura

La fase target

La fase de burbujeo

Y cuándo sería recomendable usar cada una.

EJERCICIO

Código:

```
const formulario = document.getElementById("formulario");
const div = document.getElementById("div");
const hijoDiv = document.getElementById("url");

formulario.addEventListener("click", useCapture, true);
div.addEventListener("click", useCapture, false);
hijoDiv.addEventListener("click", useCapture, true);

function useCapture(e) {
    let el = e.target;
    let name = "Nombre: " + el.id;
    let current = "Current Target: " + e.currentTarget.id;
    let value = "Fase del evento: " + e.eventPhase;

    console.log(name);
    console.log(current);
    console.log(value);
}
```

Explicación:

Declaramos constantes, añadimos los listeners y finalmente creamos la función que nos muestra la información del evento pulsado en este caso ya que es “click”.

Resultado:

The screenshot shows a browser's developer tools open to the 'Console' tab. On the left, there is a form with fields for password, date of start programming, URL, and a reset button. On the right, the console lists numerous log entries from the script. The logs show the state of variables 'name', 'current', and 'value' at different points during the event capture phase (1, 2, or 3).

| Variable | Valor | Fase del evento | Fuente |
|-----------------|------------|-----------------|--------------|
| Nombre | Contraseña | 1 | script.js:15 |
| Current Target | formulario | 1 | script.js:16 |
| Fase del evento | 1 | 1 | script.js:17 |
| Nombre | Dificultad | 1 | script.js:15 |
| Current Target | formulario | 1 | script.js:16 |
| Fase del evento | 1 | 1 | script.js:17 |
| Nombre | dia | 1 | script.js:15 |
| Current Target | formulario | 1 | script.js:16 |
| Fase del evento | 1 | 1 | script.js:17 |
| Nombre | url | 1 | script.js:15 |
| Current Target | formulario | 1 | script.js:16 |
| Fase del evento | 1 | 1 | script.js:17 |
| Nombre | url | 1 | script.js:15 |
| Current Target | url | 1 | script.js:16 |
| Fase del evento | 2 | 1 | script.js:17 |
| Nombre | url | 1 | script.js:15 |
| Current Target | div | 1 | script.js:16 |
| Fase del evento | 3 | 1 | script.js:17 |
| Nombre | restart | 1 | script.js:15 |
| Current Target | formulario | 1 | script.js:16 |
| Fase del evento | 1 | 1 | script.js:17 |
| Nombre | restart | 1 | script.js:15 |
| Current Target | div | 1 | script.js:16 |
| Fase del evento | 3 | 1 | script.js:17 |

PREGUNTAS

¿Qué listeners se ejecutan primero en la fase de captura?

Los primeros que se ejecutan son los event listeners que se encuentren habilitados en el modo de captura y que estén asociados a los elementos que son los ancestros más lejanos del elemento que recibió el clic.

Es decir, como el recorrido del evento comienza desde la raíz: documento \rightarrow html \rightarrow body, el elemento más externo en esta cadena de descenso será el primero en activar su listener de captura.

¿Cuándo actúa exactamente el "target"?

El *target* (objetivo) del evento se activa justo cuando el flujo de la interacción alcanza el elemento específico sobre el cual el usuario ha realizado el clic.

¿Qué listeners se ejecutan en la fase de burbujeo?

Durante esta etapa, se activan los event listeners que no fueron configurados para la fase de captura. La ejecución procede de forma ascendente, desde el elemento objetivo hacia los elementos más externos, siempre y cuando no sean listeners definidos para la captura.

¿Cómo cambia el comportamiento cuando alteras el valor true/false del addEventListener() del elemento hijo?

- Si se configura con captura (true): El listener del elemento se activa mientras el evento está viajando hacia abajo, antes de llegar al objetivo. Sin embargo, si ese elemento es el objetivo, su listener también se ejecuta en la fase target (objetivo).
- Si se configura con burbujeo (false o por defecto): Su listener se activa cuando el evento ya está en la fase target o cuando comienza su camino ascendente (burbujeo), dependiendo de si ese elemento fue o no el objetivo principal del clic.

¿Qué efecto tiene event.stopPropagation() si lo aplicas en uno de los listeners?

El impacto de esta función es detener el avance del evento a través de las fases restantes. Si se aplica durante la captura, el evento se cancela y no logra alcanzar ni la fase target ni la burbujeo. Si se utiliza en la fase target, el flujo no continúa hacia la fase de burbujeo. Si se aplica en la burbujeo, el evento dejará de ascender a través de los ancestros.

Conclusión Final

Fase de Captura: El evento desciende desde la capa más amplia del documento (elementos más exteriores) hacia el elemento interno.

Fase de Target (Objetivo): El evento arriba al elemento exacto donde se produjo la interacción. En este punto, todos los listeners asociados al elemento, ya sean de captura o de burbujeo, son ejecutados.

Fase de Burbujeo: El evento comienza a ascender desde el elemento objetivo hacia los elementos contenedores más externos. Es aquí donde se activan los listeners con la configuración estándar (los que no son de captura).

El modo de captura se emplea cuando se necesita interceptar o interactuar con el evento antes de que llegue al elemento final; la fase target representa el momento de reacción directa al clic en su ubicación precisa; y la burbujeo es útil cuando un componente principal (como un contenedor) debe manejar eventos originados en sus hijos, o cuando se desea que una acción se desencadene mientras el evento se propaga hacia arriba a través de la jerarquía.