Teoría de Manejo de Eventos en JavaScript

En JavaScript, hay varias formas de declarar eventos, cada una con su uso específico. Los métodos principales son:

Declaración en Línea (inline): El evento se declara directamente en el HTML usando el atributo del evento en el elemento.

1. **Declaración en Línea (inline)**: El evento se declara directamente en el HTML usando el atributo del evento en el elemento.

```
<button onclick="alert('Clicked!')">Click me</button>
```

2. **Como Propiedad del Elemento**: En el código JavaScript, el evento se asocia directamente a la propiedad del elemento.

```
element.onclick = function() { alert("Clicked!"); };
```

3. **Mediante `addEventListener`**: Asocia un evento usando el método `addEventListener`, permitiendo adjuntar múltiples funciones a un mismo evento.

```
element.addEventListener("click", function() { alert("Clicked!"); });
```

Además, **el objeto del evento** contiene propiedades útiles, como:

- `.target`: Elemento que activó el evento.
- `.preventDefault()`: Evita la acción predeterminada del evento.
- `.stopPropagation()`: Evita que el evento se propague a otros elementos.

Teoría y Sintaxis de Eventos en JavaScript

Tipo de Evento	Nombre del Evento	Descripción	Ejemplo de Sintaxis
Ratón	`click`, `dblclick`	Click/doble clic en un elemento.	`element.onclick = function() { ... }`
	`mouseenter`, `mouseleave`	Entrada/salida del cursor en un elemento.	`element.onmouseenter = function() { ... }`
	`mouseover`, `mouseout`	Entrada/salida del cursor en un elemento o alguno de sus hijos.	`element.onmouseover = function() { ... }`
	`mousedown`, `mouseup`	Presión/liberación de un botón del ratón.	`element.onmousedown = function() { ... }`
	`drag`, `drop`	Movimiento y liberación de un elemento arrastrable.	`element.ondragstart = function() { ... }`
Teclado	`keydown`, `keyup`	Tecla mantenida/pulsada/liberada por el usuario.	`element.onkeydown = function() { ... }`
Formulario	`focus`, `blur`	Elemento obtiene/pierde el foco.	`element.onfocus = function() { ... }`
	`submit`, `reset`	Envía o borra los datos de un formulario.	`form.onsubmit = function() { ... }`
Navegador	`load`, `unload`	Página cargada o cerrada.	`window.onload = function() { ... }`
Propiedades de Eventos	`.preventDefault()`	Evita el comportamiento por defecto de un evento.	`event.preventDefault()`
	`.stopPropagation()`	Detiene la propagación del evento a otros elementos.	`event.stopPropagation()`
	`.target`	Elemento específico que activó el evento.	`console.log(event.target)`

Ejercicio 1: Click y Doble Click

Ejemplo: Cambiar el tamaño de un div al hacer clic y al hacer doble clic

```
<div id="box" style="width: 100px; height: 100px; background-color: lightgray;"></div>
<script>
    let box = document.getElementById("box");
    box.onclick = function() { box.style.width = "150px"; };
    box.ondblclick = function() { box.style.width = "100px"; };
</script>
```

Reto: Haz que al hacer clic en el div, cambie tanto el color como el tamaño, y al hacer doble clic, vuelva a sus valores iniciales.

Ejercicio 2: Hover con mouseenter y mouseleave

Ejemplo: Cambiar el color de texto de un párrafo cuando el ratón pasa por encima

```
<p id="hoverText">Hover over me!</p>
<script>
    let text = document.getElementById("hoverText");
    text.onmouseenter = function() { text.style.color = "blue"; };
    text.onmouseleave = function() { text.style.color = "black"; };
</script>
```

Reto: Haz que al pasar el ratón sobre el texto, el tamaño también cambie junto al color, y vuelve a sus valores iniciales al salir.

Ejercicio 3: preventDefault en Formularios

Ejemplo: El método `preventDefault()` evita la acción predeterminada de un evento. En formularios, puede usarse para evitar su envío hasta que se cumpla una condición. Prevenir el envío de un formulario hasta que un campo determinado esté relleno.

```
<form id="myForm">
    <input type="text" id="name" placeholder="Enter your name">
    <button type="submit">Submit</button>
</form>
<script>
    document.getElementById("myForm").onsubmit = function(event) {
        if (document.getElementById("name").value === "") {
            event.preventDefault();
            alert("Name is required.");
        }
    };
</script>
```

Reto: Añade validación para que no se permita enviar el formulario si otro campo de correo electrónico está vacío o mal formado

Ejercicio 4: keydown y keyup

Ejemplo: `keydown` se activa al presionar una tecla y `keyup` cuando se libera. Estos eventos permiten capturar y manipular entradas de teclado en tiempo real. Mostrar un mensaje cuando se presiona la tecla Enter en un campo de texto.

```
<input type="text" id="inputField" placeholder="Press Enter">
<script>
    document.getElementById("inputField").onkeydown = function(event) {
        if (event.key === "Enter") alert("Enter pressed!");
    };
</script>
```

Reto: Crea un contador que aumente cada vez que se presiona la tecla de espacio y se reinicie cuando se suelte.

Ejercicio 5: focus y blur en Formularios

Ejemplo: `focus` y `blur` son eventos de formulario que indican cuando un campo obtiene o pierde el foco, útiles para cambiar el estilo del campo. Mostrar un mensaje al obtener el foco y ocultarlo al perderlo.

```
<input type="text" id="nameField" placeholder="Enter your name">
<p id="message" style="display: none;">Please enter your name.</p>
<script>
    let nameField = document.getElementById("nameField");
    let message = document.getElementById("message");
    nameField.onfocus = function() { message.style.display = "block"; };
    nameField.onblur = function() { message.style.display = "none"; };
</script>
```

Reto: Haz que al enfocar el campo, cambie el texto de `placeholder`, y que vuelva al original al desenfocarlo.

Ejercicio 6: addEventListener y removeEventListener

Ejemplo: `addEventListener` permite agregar múltiples listeners a un evento, mientras `removeEventListener` quita un listener. Esto es útil para modular el manejo de eventos. Cambiar el color del fondo de una sección al hacer clic y eliminar el evento tras dos clics.

```
<button id="changeTextButton">Click me</button>
<script>
    let button = document.getElementById("changeTextButton");
    let clickCount = 0;
    function changeText() {
        button.textContent = "Clicked!";
        clickCount++;
        if (clickCount >= 2) button.removeEventListener("click", changeText);
    }
    button.addEventListener("click", changeText);
</script>
```

Reto: Usa `addEventListener` para activar un contador que se incrementa en cada clic, y luego desactiva el evento una vez que llegue a 5.

Ejercicio 7: Propagación de Eventos con stopPropagation

Ejemplo: `stopPropagation()` evita que el evento se propague a otros elementos. Útil en estructuras anidadas donde se requiere que el evento solo afecte al elemento inmediato. Crear dos divs anidados, donde solo el más interno responda al evento `click`.

```
<div id="outer" style="padding: 30px; background-color: lightblue;">
    Outer
    <div id="inner" style="padding: 30px; background-color: lightcoral;">Inner</div>
</div>
<script>
    document.getElementById("outer").onclick = function() { alert("Outer clicked!"); }
    document.getElementById("inner").onclick = function(event) {
        alert("Inner clicked!");
        event.stopPropagation();
    };
</script>
```

Reto: Añade un tercer nivel de `div` anidado y haz que el `click` solo dispare el alert en el `div` más interno.

Ejercicio 8: Resize con Eventos de Navegador

Ejemplo: El evento `resize` se activa al cambiar el tamaño de la ventana del navegador, útil para ajustar el diseño en respuesta al tamaño de pantalla. Mostrar el tamaño de la ventana en un párrafo que se actualice en cada cambio.

```
<p id="windowSize"></p>
<script>
    function updateSize() {
        document.getElementById("windowSize").textContent = `Width: ${window.innerWidth}, Height: ${window.innerHeight}`;
    }
    window.addEventListener("resize", updateSize);
    updateSize();
</script>
```

Reto: Cambia el color de fondo a rojo si el ancho de la ventana es menor a 600px, y a verde si es mayor.

Ejercicio 9: `keydown`, `keyup` y `keypress` para detectar teclas

Ejemplo: **keydown**: se activa cuando se mantiene presionada una tecla. **keyup**: se dispara al liberar la tecla. **keypress**: registra la tecla pulsada y repite el evento mientras se mantenga presionada (en desuso y poco recomendado en algunos navegadores). Mostrar un mensaje cuando el usuario presiona la tecla “Enter”.

```
<input type="text" id="inputBox" placeholder="Escribe algo">
<p id="message"></p>
<script>
    document.getElementById("inputBox").addEventListener("keyup", (event) => {
        if (event.key === "Enter") {
            document.getElementById("message").textContent = "¡Tecla Enter presionada!";
        }
    });
</script>
```

Reto: Detecta cuando el usuario escribe la letra “a” y muestra un mensaje en pantalla.

Ejercicio 10: Evento `input` en Formularios

Ejemplo: El evento `input` se activa cuando se cambia el contenido de un input, incluyendo texto y selección. Mostrar la longitud del texto ingresado en un párrafo aparte

```
<input type="text" id="textInput" placeholder="Escribe algo">
<p id="textLength"></p>
<script>
    document.getElementById("textInput").addEventListener("input", (event) => {
        document.getElementById("textLength").textContent = `Longitud: ${event.target.value.length}`;
    });
</script>
```

Reto: Agrega una advertencia si la longitud supera los 10 caracteres.

Ejercicio 11: `focus` y `blur` en Formularios

Ejemplo: **focus**: se dispara cuando el elemento gana el foco. **blur**: se activa cuando el elemento pierde el foco. Restaurar el color de fondo al perder el foco.

```
<input type="text" id="inputFocus" placeholder="Click aquí">
<script>
    const input = document.getElementById("inputFocus");
    input.addEventListener("focus", () => input.style.backgroundColor = "yellow");
    input.addEventListener("blur", () => input.style.backgroundColor = "");
</script>
```

Reto: Cambia el color de fondo a rojo si el usuario deja el input vacío.

Ejercicio 12: `submit` y `preventDefault()`

Ejemplo: **submit**: permite capturar el envío de formularios. **preventDefault()**: evita el comportamiento por defecto, como recargar la página en el envío de un formulario. Mostrar un mensaje en pantalla en lugar de enviar el formulario.

```
<form id="myForm">
    <input type="text" id="nameInput" placeholder="Nombre">
    <button type="submit">Enviar</button>
</form>
<p id="message"></p>
<script>
    document.getElementById("myForm").addEventListener("submit", (event) => {
        if (document.getElementById("nameInput").value === "") {
            event.preventDefault();
            document.getElementById("message").textContent = "Por favor, ingresa un nombre.";
        }
    });
</script>
```

Reto: Haz que el mensaje sea rojo si el campo está vacío y verde si tiene texto.

Ejercicio 13: `mouseenter` y `mouseleave` para Hovers

Ejemplo: **mouseenter**: activa cuando el cursor entra en un elemento. **mouseleave**: activa cuando el cursor sale de un elemento. Mostrar un mensaje temporal al pasar el cursor por encima de un elemento.

```
<button id="hoverButton">Pasa el cursor aquí</button>
<p id="hoverMessage"></p>
<script>
    const button = document.getElementById("hoverButton");
    button.addEventListener("mouseenter", () => document.getElementById("hoverMessage").textContent = "¡Hola!");
    button.addEventListener("mouseleave", () => document.getElementById("hoverMessage").textContent = "");
</script>
```

Reto: Agrega un mensaje diferente al poner el cursor y al retirarlo.

Ejercicio 14: Propiedad `pageX` y `pageY` en Eventos de Ratón

Ejemplo: Actualizar las coordenadas en tiempo real mientras el ratón se mueve.

```
<p id="coordinates"></p>
<script>
    document.addEventListener("mousemove", (event) => {
        document.getElementById("coordinates").textContent = `X: ${event.pageX}, Y: ${event.pageY}`;
    });
</script>
```

Reto: Restringe la visualización de las coordenadas a una sección específica de la pantalla.

Ejercicio 15: `target` y `currentTarget` en Eventos

Ejemplo: **target**: el elemento que disparó el evento. **currentTarget**: el elemento actual que maneja el evento. Mostrar el texto del elemento clicado

```
<ul id="list">
    <li>Elemento 1</li>
    <li>Elemento 2</li>
    <li>Elemento 3</li>
</ul>
<script>
    document.getElementById("list").addEventListener("click", (event) => {
        if (event.target.tagName === "LI") {
            alert(`Has clicado: ${event.target.textContent}`);
        }
    });
</script>
```

Reto: Agregar un borde rojo al elemento clicado y eliminarlo cuando se clica en otro.

Ejercicio 16: `stopPropagation()`

Ejemplo: Evita que un evento se propague a otros elementos padres. Mostrar un mensaje en el padre y otro en el hijo

```html
<div id="parent" style="padding:20px; background-color:lightgray;">
    Padre
    <div id="child" style="padding:20px; background-color:lightblue;">
        Hijo
    </div>
</div>
<script>
    document.getElementById("parent").addEventListener("click", () => alert("Clic en padre"));
    document.getElementById("child").addEventListener("click", (event) => {
        alert("Clic en hijo");
        event.stopPropagation();
    });
</script>
```

Reto: Agrega otro nivel anidado y evita que el evento se propague hasta el abuelo.

Ejercicio 17: `DOMContentLoaded`

Ejemplo: **DOMContentLoaded**: se activa cuando el contenido HTML ha sido completamente cargado, antes de los recursos externos. Ocultar un mensaje de carga una vez el contenido se haya cargado.

```html
<p id="loadingMessage">Cargando...</p>
<script>
    document.addEventListener("DOMContentLoaded", () => {
        document.getElementById("loadingMessage").style.display = "none";
    });
</script>
```

Reto: Muestra un saludo en lugar del mensaje de carga una vez el contenido esté listo.

Ejercicio 18: `scroll` en el Navegador

Ejemplo: **scroll**: se activa cuando la página o un elemento desplazable es desplazado. Mostrar el porcentaje de scroll que ha hecho el usuario

```html
<p id="scrollPercentage"></p>
<script>
    document.addEventListener("scroll", () => {
        const scrollTop = window.scrollY;
        const docHeight = document.documentElement.scrollHeight - window.innerHeight;
        const scrollPercent = (scrollTop / docHeight) * 100;
        document.getElementById("scrollPercentage").textContent = `Desplazado: ${scrollPercent.toFixed(0)}%`;
    });
</script>
```

Reto: Cambia el color de fondo a medida que el usuario se desplaza por la página.

Ejercicio 19: `dragenter` y `dragleave` para Elementos Arrastrables

Ejemplo: **dragenter**: se dispara cuando un elemento arrastrable entra en un área de destino. **dragleave**: se activa cuando el elemento arrastrable sale del área de destino. Mostrar el porcentaje de scroll que ha hecho el usuario.

```
<div id="dropZone" style="width:200px; height:200px; background-color:lightgray;">Arrastra aquí</div>
<script>
    const dropZone = document.getElementById("dropZone");
    dropZone.addEventListener("dragenter", () => dropZone.style.backgroundColor = "lightgreen");
    dropZone.addEventListener("dragleave", () => dropZone.style.backgroundColor = "lightgray");
</script>
```

Reto: Muestra una alerta al soltar el elemento arrastrado dentro del contenedor.

Ejercicio 20: `drop` para Completar Arrastres

Ejemplo: **drop**: se activa cuando un elemento arrastrable es soltado en el área de destino. Necesita habilitarse con `dragenter` y `dragover` para que el área sea un “destino de arrastre” válido. Cambiar el texto de un contenedor cuando un elemento se suelta en él..

```
<div id="dropArea" style="width:200px; height:200px; background-color:lightblue;">Suelta aquí</div>
<script>
    const dropArea = document.getElementById("dropArea");
    dropArea.addEventListener("dragover", (e) => e.preventDefault()); // Permite el drop
    dropArea.addEventListener("drop", () => dropArea.textContent = "Elemento soltado aquí.");
</script>
```

Reto: Cambia el color de fondo del contenedor al soltar el elemento y restaura el color original después de 2 segundos.

```
setTimeout(() => {
    dropArea.style.backgroundColor = "lightblue";
}, 2000); // 2000 milisegundos = 2 segundos
```