Eventos

Un evento es cualquier acción que ocurre en el navegador y que puede ser detectada por el JavaScript. Ejemplos comunes de eventos incluyen hacer clic en un botón, enviar un formulario, mover el ratón sobre un elemento, o presionar una tecla. Como respuesta a un evento específico, se ejecutará una función o código en JavaScript, lo que se conoce como manejador de eventos.

El encargado de crear la jerarquía de objetos que compone una página web es el DOM (Document Object Model). Por tanto es el DOM el encargado de gestionar los eventos.

Imaginemos que queremos crear una página web con un botón que, cuando se pulse, muestra un mensaje. El botón será el elemento html que desencadena el evento. El código JavaScript que manejará el evento 'click', se encargará de mostrar el mensaje.

Tipos de Eventos: Entre los más utilizados, se encuentran:

1.- **Eventos del ratón**

- click. Este evento se produce cuando pulsamos sobre el botón izquierdo del ratón. El manejador de este evento es **onclick**.
- dblclick. Este evento se acciona cuando hacemos un doble click sobre el botón izquierdo del ratón. El manejador de este evento es **ondblclick**.
- mousedown. Este evento se produce cuando pulsamos un botón del ratón. El manejador de este evento es **onmousedown**.
- mouseout. Este evento se produce cuando el puntero del ratón esta dentro de un elemento y este puntero es desplazado fuera del elemento. El manejador de este evento es **onmouseout**
- mouseover. Este evento al revés que el anterior se produce cuando el puntero del ratón se encuentra fuera de un elemento, y este se desplaza hacia el interior. El manejador de este evento es **onmouseover**.
- mouseup. Este evento se produce cuando soltamos un botón del ratón que previamente teníamos pulsado. El manejador de este evento es **onmouseup**.
- mousemove. Se produce cuando el puntero del ratón se encuentra dentro de un elemento. Es importante señalar que este evento se producirá continuamente una vez tras otra mientras el puntero del ratón permanezca dentro del elemento. El manejador de este evento es **onmousemove**

2.- Eventos del teclado

- keydown. Este evento se produce cuando pulsamos una tecla del teclado. Si mantenemos pulsada una tecla de forma continua, el evento se produce una y otra vez hasta que soltemos la misma. El manejador de este evento es **onkeydown**.
- keypress. Este evento se produce si pulsamos una tecla de un carácter alfanumérico (El evento no se produce si pulsamos enter, la barra espaciadora, etc…). En el caso de mantener una tecla pulsada, el evento se produce de forma continuada. El manejador de este evento es **onkeypress**.
- keyup. Este evento se produce cuando soltamos una tecla. El manejador de este evento es **onkeyup**.

3.- Eventos de formulario

- submit. Este evento se produce cuando pulsamos sobre un botón de tipo submit. El manejador es **onsubmit**.
- change. Ocurre cuando el valor de un elemento <input>, <select>, o <textarea> cambia. El manejador es **onchange**.
- focus. Este evento se produce cuando un elemento obtiene el foco. El manejador es **onfocus**.
- blur. Este evento se produce cuando un elemento pierde el foco. El manejador es **onblur**
- reset. Este evento se produce cuando pulsamos sobre un botón de tipo reset. El manejador es **onreset**.
- abort. Este evento se produce cuando el usuario detiene la descarga de un elemento antes de que haya terminado, actúa sobre un elemento <object>. El manejador es **onabort**
- error. El evento error se produce en el objeto Window cuando se ha producido un error en JavaScript. En el elemento <img> cuando la imagen no se ha podido cargar por completo y en el elemento <object> en el caso de que un elemento no se haya cargado correctamente. El manejador es **onerror**.
- select. Se acciona cuando seleccionamos texto de los cuadros de textos <input> y <textarea>. El manejador es **onselect**.
- resize. Este evento se produce cuando redimensionamos el navegador, actúa sobre el objeto Window. El manejador es **onresize**.
- scroll. Se produce cuando varía la posición de la barra de scroll en cualquier elemento que la tenga. El manejador es **onscroll**.

4.- Eventos de carga

- load. Se activa cuando una página o un recurso (como una imagen) se ha cargado completamente. El manejador es **onload**.
- unload. El evento unload actúa sobre el objeto Window cuando la pagina ha desaparecido por completo (por ejemplo, si pulsamos el aspa cerrando la ventana del navegador). También se acciona en el elemento <object> cuando desaparece el objeto. El manejador es onunload.
- DOMContentLoad. Se produce cuando el documento se ha cargado. La forma de asegurarte que la página está cargada es:

  window.addEventListener("DOMContentLoaded", inicio);

5.- Eventos de arrastre

- drag: Ocurre cuando un elemento es arrastrado.
- dragstart: Ocurre cuando se inicia el arrastre de un elemento.
- dragend: Ocurre cuando el arrastre de un elemento finaliza.
- dragover: Ocurre cuando un elemento arrastrado está sobre un área de destino.
- drop: Ocurre cuando un elemento arrastrado se deja caer en un área de destino.

Cómo manejar eventos

FORMA 1: Modelo de registro de eventos en línea.

- En el modelo de registro de eventos en línea (estandarizado por Netscape), el evento es añadido como un atributo más a la etiqueta HTML, como por ejemplo:

  <A href="pagina.html" onclick="alert('Has pulsado en el enlace')">Pulsa aqui

  Cuando hacemos click en el enlace, se llama al gestor de eventos onClick (al hacer click) y se ejecuta el script; que contiene en este caso una alerta de JavaScript. También se podría realizar lo mismo pero llamando a una función:

```
<A href="pagina.html" onClick="alertar()">Pulsa aqui</a>

function alertar(){
    alert("Has pulsado en el enlace");
}
```

- Este modelo no se recomienda, y aunque lo has visto en ejemplos que hemos utilizado hasta ahora, tiene el problema de que estamos mezclando la estructura de la página web con la programación de la misma, y lo que se intenta hoy en día es separar la programación en JavaScript, de la estructura HTML.

FORMA 2: Modelo de registro de eventos tradicional

- En este nuevo modelo el evento pasa a ser una propiedad del elemento, así que por ejemplo los navegadores modernos ya aceptan el siguiente código de JavaScript:

```
elemento.onclick = hacerAlgo; // cuando el usuario haga click en el objeto, se llamará a la
funcion hacerAlgo(){
---
}
```

- Esta forma de registro, no fue estandarizada por el W3C, pero debido a que fue ampliamente utilizada por Netscape y Microsoft, todavía es válida hoy en día. La ventaja de este modelo es que podremos asignar un evento a un objeto desde JavaScript, con lo que ya estamos separando el código de la estructura.
- Los nombres de los eventos sí que van siempre en minúsculas: elemento.onclick = hacerAlgo;
- Para eliminar un gestor de eventos de un elemento u objeto, le asignaremos null:

```
elemento.onclick = null;
```

- Otra gran ventaja es que, como el gestor de eventos es una función, podremos realizar una llamada directa a ese gestor, con lo que estamos disparando el evento de forma manual.

Por ejemplo:

```
elemento.onclick();   // Al hacer ésto estamos disparando el evento click de forma manual y se ejecutará
                      //la función hacerAlgo()
```

SIN PARÉNTESIS

- En el registro del evento no usamos paréntesis (). El método onclick espera que se le asigne una función completa. Si haces: element.onclick = hacerAlgo(); la función será ejecutada y el resultado que devuelve esa función será asignado a onclick. Pero ésto no es lo que queremos que haga, queremos que se ejecute la función cuando se dispare el evento.

FORMA 3: Modelo de registro avanzado de eventos

La forma moderna y flexible de manejar eventos es usando el método **addEventListener** sobre el elemento html que provoca el evento. Esto te permite agregar múltiples manejadores para un solo evento y separar el JavaScript del HTML. Ejemplo:

- Este método tiene tres argumentos: el tipo de evento, la función a ejecutar y un valor booleano (true o false), que se utiliza para indicar cuándo se debe capturar el evento: en la fase de captura (true) o de burbujeo (false): **elemento.addEventListener('evento', función, false|true)**

- Ejemplo: Cuando hagamos click sobre el enlace cuyo id es 'mienlace', se llama a la función alertar.

```
document.getElementById("mienlace").addEventListener('click',alertar,false);
function alertar(){
        alert("Te conectaremos con la página: "+this.href);
}
```

this hace referencia al objeto que ha provocado el evento. En este caso un enlace.

- Para **eliminar** un evento de un elemento, usaremos el método **removeEventListener**():

```
elemento.removeEventListener();
```

Obteniendo información del evento. Importante

Cuando se produce un evento, se puede pasar a la función el objeto '**event**'. Este objeto proporciona información sobre el evento que ocurrió. Este objeto contiene propiedades y métodos útiles como:

- **event.target:** Elemento que ha disparado el evento
- **event.type**: El tipo del evento. Por ejemplo, 'click'
- **event.preventDefault** : Evita la acción por defecto del evento, como por ejemplo al enviar un formulario)

Ejemplo: Supongamos que tenemos un botón <button id="**mienlace**">pulsa</button>
Si le asociamos un evento al hacer 'click' de la siguiente forma:

```
document.getElementById("mienlace").addEventListener("click", alertar);
```

La siguiente función alertar podrá hacer lo siguiente: (e es el objeto event, que puede tener cualquier nombre)

```
alertar = (e) => {
 alert(e.type);
 alert(e.target);
 e.target.style.color = "blue";
};
```

¿Qué hace esta función?

Ejemplo de preventDefault: En los formularios, los button o los submit automáticamente realizan un envío de formulario. Para evitar que haga ese envío por defecto, por ejemplo en las validaciones de un formulario, se utiliza preventDefault.

```
<form id=”miFormulario”>
    <input type=”text” id=”n1” size=30 />
    <button id=”b1”>validar</button>
</form>

validar = (e) =>{
    e.preventDefault();
    alert(“Ahora el formulario no recarga, hemos anulado el comportamiento por defecto”);
}

let $boton = document.getElementById(“b1”);
$boton.addEventListener(“click”, validar);
```

NOTA: Para poder usar bootstrap en los ejercicios en head ponemos:

```
<link
  rel="stylesheet"
  href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
/>
```

Ver estilos: https://getbootstrap.com/docs/3.4/css/

Ejercicios

1. Crear un formulario con una etiqueta label y un input type text para que escribáis vuestro nombre. Cuando hagáis **click** fuera del campo, tendréis que convertir el nombre a mayúsculas en el input. El valor nuevo debe quedarse señalado (marcado con el cursor). Usar función flecha. Utilizar addEventListener.

2. Crear una página html con dos input de tipo text. Cuando se escriba en ambos, el color del texto debe ser rojo y cuando pierda el foco, el color del texto será negro.

   Después, añadir un botón 'Púlsame' (button), con un evento, tal que al hacer click ponga el color del texto del botón en azul.
   Añadir otro botón 'Cambia', de forma que al pasar por encima ponga el borde del botón de color verde y cuando salga del botón, lo ponga de color naranja.

3. Crear un formulario con 6 checkbox y un botón 'validar' . Cuando pulsemos el botón, validará que hay 3 o más checkbox marcados. Mostrar por alert un mensaje que indique si hay o no 3 o más de 3 checkbox marcados.

4. Crear una página con dos botones: uno 'Sumar' y otro 'Restar'. Debajo hay un div que inicialmente muestra un 5. Al pulsar el botón Sumar, se sumará uno, y al pulsar el botón restar, se restará uno. El div tendrá que actualizarse en cada operación. Cuando se llegue al valor 3, el div debe mostrar además, un mensaje que indique 'Alcanzado valor 3'. Cuando vuelva a ser distinto de 3, el mensaje debe desaparecer.

5. Crear un estilo llamado boton que tenga una altura y anchura de 50px.
   El formulario tendrá 10 botones con ese estilo creado y con values del 0 al 9 respectivamente.
   Crear para todos ellos un evento tal que al pulsarlo muestre en una etiqueta div el valor del boton pulsado.

   NOTA: Para crear estilos: .boton {…..}

   Pista: crear los eventos **con un bucle for.**

6. Crear un formulario con 2 input de tipo de texto: Nombre y Apellidos. Y un botón validar.
   Se tiene que cumplir que:
   1. El nombre comienza por vocal.
   2. Si los apellidos tienen 2 o más de dos palabras.

   Poner debajo de cada input un mensaje en caso de no cumplir su validación.
   Una vez hecho, añadir que el mensaje aparezca con letra de color rojo.

7. Sobre el ejercicio anterior, modificar lo necesario para que una vez que se pulsa el botón, si se cumplen las dos validaciones, el formulario debe redirigir a la página de: https://ceu.es.

   En caso contrario, el comportamiento debe ser igual al ejercicio anterior.

8. Dado el formulario:

```
<form name="formulario" id="formulario" action="http://www.google.es" method="get">
    <label for="nombre">Nombre:</label>
    <input type="text" name="nombre" id="nombre" />

    <label for="apellidos">Apellidos:</label>
    <input type="text" name="apellidos" id="apellidos" />

    <label for="edad">Edad:</label>
    <input name="edad" type="text" id="edad" maxlength="3" />

    <label for="provincia">Provincia:</label>
    <select name="provincia" id="provincia">
        <option value="0" selected="selected">Seleccione Provincia</option>
            <option value="H">Huesca</option>
            <option value="ZA">Zaragoza</option>
            <option value="T">Teruel</option>
        </select>
    <fieldset>
        <div id="resultado"></div>
        <input type="reset" name="limpiar" id="limpiar" value="Limpiar" />
            <input type="submit" name="enviar" id="enviar" value="Enviar" />
    </fieldset>
</form>
```

Validar el formulario antes de enviarlo. Para ello se debe comprobar que el nombre, apellidos y edad tienen valor (no se dejan en blanco). La edad además debe ser un número y sus valores deben estar comprendidos entre 0 y 105. También se debe validar que se ha seleccionado alguna provincia.

Si se cumplen todas las validaciones, debe preguntarse al usuario con un mensaje si deseas enviar el formulario. En caso afirmativo, se enviará. En caso negativo no hará nada.

No se mostraran todos los errores. Sólo el primero que no cumpla la validación.

Cuando no se cumpla alguna validación debe aparecer el mensaje correspondiente en la etiqueta div (id=”resultado”) del final en color rojo.

9. Dada la página ‘window.html’ que contiene un título y un botón ‘Abrir Página’, crear código JavaScript de forma que, al pulsarlo se abra otra ventana: ‘datos.html’ de 400x300. Esta segunda pantalla tendrá:

   1. 2 cuadros de texto donde el usuario tendrá que introducir su nombre y sus apellidos
   2. 1 cuadro de tipo fecha (type=”date”) dónde seleccionará la fecha de nacimiento.
   3. 1 botón ‘Mostrar’, para que muestre **en la página principal,** los datos del usuario junto con su fecha de nacimiento en formato dd/mm/yyyy. Y deberá mostrar también, los años bisiestos que ha habido desde que nació.

   NOTA: Un año es bisiesto si el año es divisible por 4 pero no divisible por 100, o bien, el año debe ser divisible por 400.

   window.html:

```
<body>
   <h1>Ventana Principal</h1>
   <button id="abrirPagina">Abrir Página</button>
   <div id="resultado"></div>
</body>
```

10. Dada la página index.html con un temporizador de cuenta regresiva que inicia en 30 segundos. El usuario tendrá tres botones: **Iniciar**, **Pausar** y **Reiniciar**. El temporizador se mostrará en pantalla y disminuirá de 1 en 1 segundo. Cuando el temporizador llegue a 0, mostrará "¡Tiempo agotado!".

   Index.html:

```
<body>
            <h1>Cuenta Regresiva</h1>
             <div id="temporizador">30</div>
             <button id="iniciar">Iniciar</button>
             <button id="pausar">Pausar</button>
             <button id="reiniciar">Reiniciar</button>
</body>
```