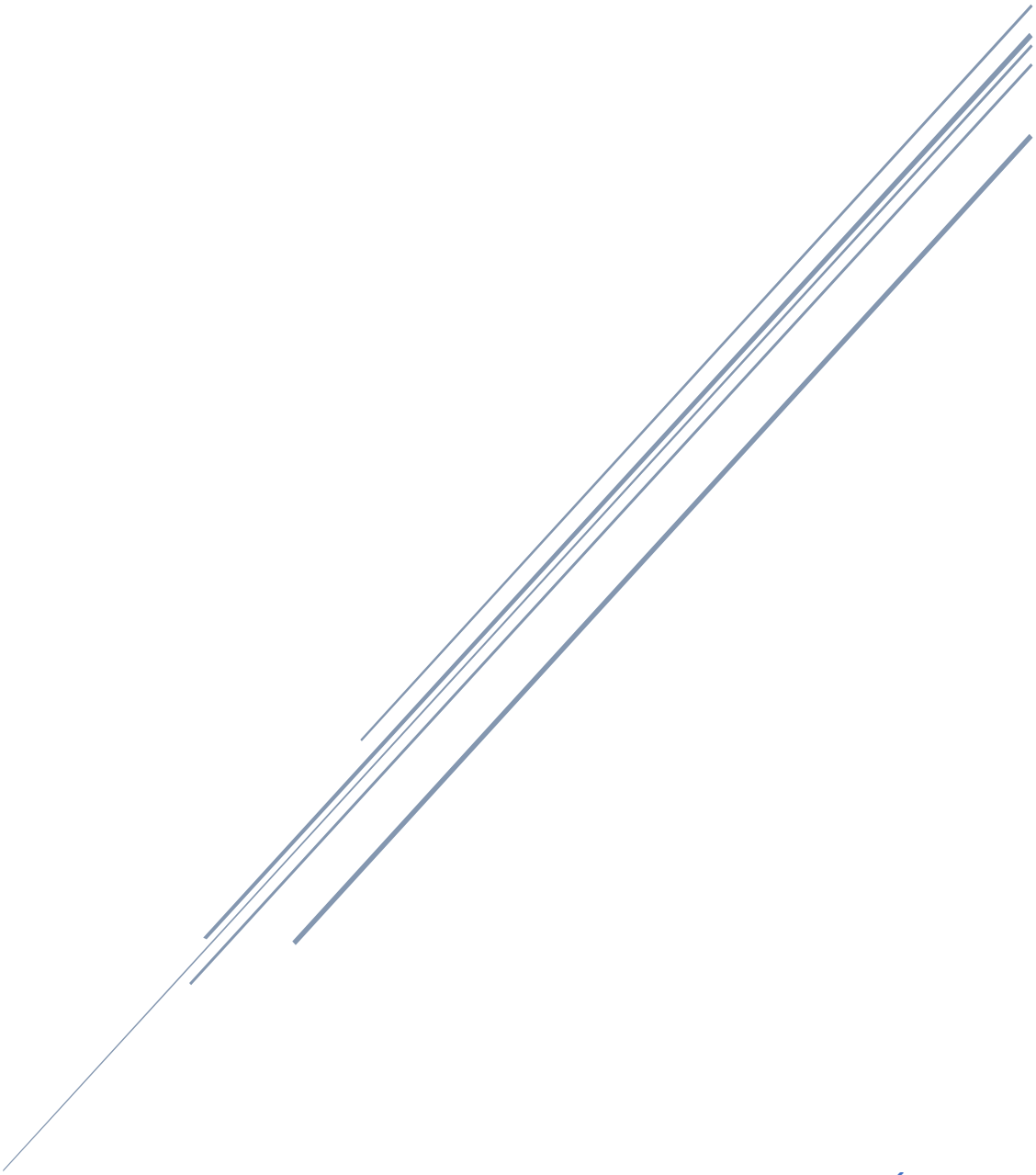
EJERCICIOS DE EVENTOS JS

Parte teórico



I.E.S. MAR DE CÁDIZ

2ª DESARROLLO DE APLICACIONES WEB

Índice

1. ¿Qué son los manejadores de eventos? 2
2. Define Manejadores como atributos XHTML e indica un ejemplo de código 2
3. Define Manejadores de eventos con la variable this e indica un ejemplo de código 2
4. Define Manejadores de eventos como funciones externas e indica un ejemplo de código... 3
5. Define Manejadores de eventos semánticos e indica un ejemplo de código. 3
6. ¿Qué es el Event bubbling? 4
7. ¿Qué es el Event capturing? 4
8. ¿Que es el Event Dom? 5
9. Define el objeto event, sus propiedades y métodos 5
10. ¿Qué tipos de eventos existen segun el DOM? indica cada uno de los grupos existentes y ejemplos de cada grupo. 7

1. ¿Qué son los manejadores de eventos?

Un manejador de evento es una función que se ejecuta en caso de que el evento sea activado.

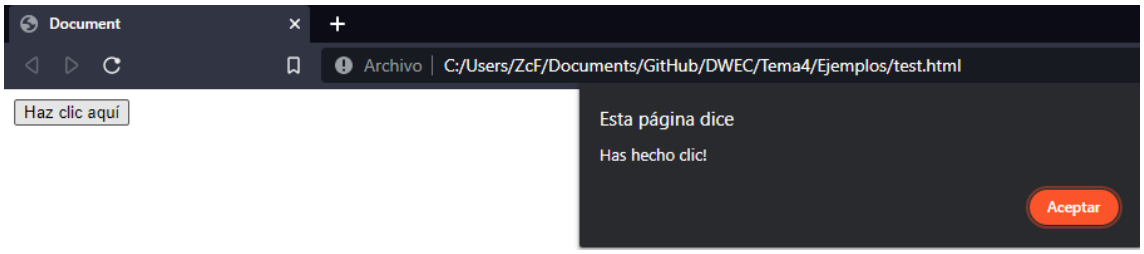
Algunos ejemplos de manejador de evento son: contextmenu, mouseover, mousedown, mouseup, keydown, keypress…

2. Define Manejadores como atributos XHTML e indica un ejemplo de código

En este caso, el manejador de evento se incluye en un atributo del propio elemento HTML.

En el siguiente ejemplo,aAl hacer clic en el botón se activa el manejador del evento click (on-click), el cual ejecuta la función o método asignado (en este caso el método alert()).

```
<body>
    <input value="Haz clic aquí" onclick="alert('Has hecho clic!')" type="button" />
</body>
```



3. Define Manejadores de eventos con la variable this e indica un ejemplo de código

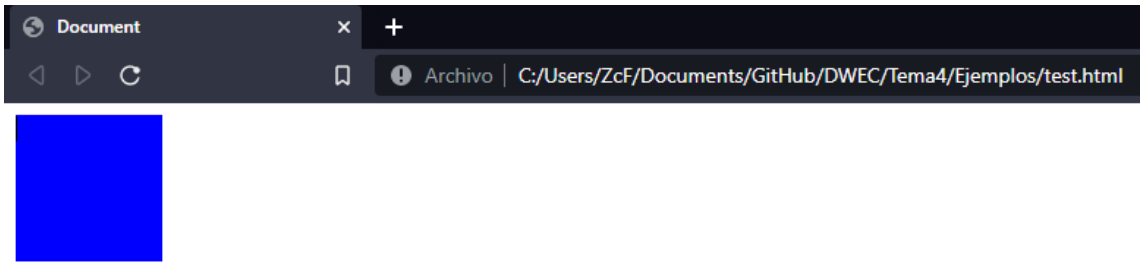
Se utiliza la variable this para referirse al elemento XHTML que ha provocado el evento. Esta variable es muy útil para ejemplos como el siguiente:

Si no se utiliza la variable this, el código necesario para modificar el color de fondo es el siguiente:

```
<body>
    <div id="caja" style="width: 100px; height: 100px; background-color: black;"
        onmouseover="document.getElementById('caja').style.backgroundColor='red';"
        onmouseout="document.getElementById('caja').style.backgroundColor='blue';">
    </div>
</body>
```

El código anterior es demasiado largo y demasiado propenso a errores. En el código del evento, JavaScript crea automáticamente la variable this, que hace referencia al elemento XHTML que generó el evento. Por lo tanto, el ejemplo anterior se puede reescribir de la siguiente manera:

```
<body>
    <div id="caja" style="width: 100px; height: 100px; background-color: black;"
        onmouseover="this.style.backgroundColor='red';"
        onmouseout="this.style.backgroundColor='blue';">
    </div>
</body>
```



4. **Define Manejadores de eventos como funciones externas e indica un ejemplo de código**

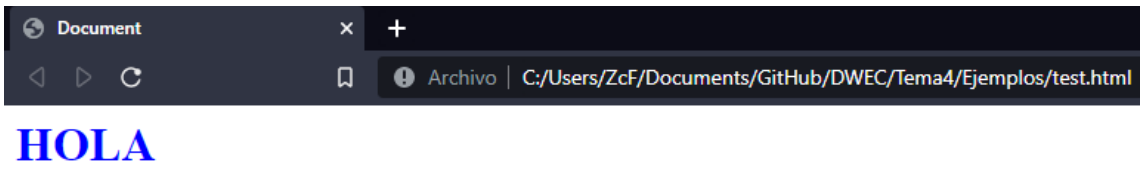
Con este método, lo que hacemos es extraer las instrucciones de JavaScript y ponerlas en una función externa, incluyendo el nombre de la función que se llamará cuando se ejecute el evento.

Aunque nos encontramos con el problema de no poder utilizar la variable this dentro de la función exterior, por lo que es necesario pasar dicha variable como argumento.

En el siguiente ejemplo, si el ratón esta encima del h1, el h1 será de color rojo, y sino azul.

```
<h1 onmouseover="dentro(this)" onmouseout="fuera(this)">HOLA</h1>

<script>
    function dentro(elemento) {
        elemento.style.color = 'red';
    }
    function fuera(elemento) {
        elemento.style.color = 'blue';
    }
</script>
```



5. **Define Manejadores de eventos semánticos e indica un ejemplo de código.**

Esta técnica consiste en asignar las funciones externas mediante las propiedades DOM de los elementos XHTML. En esta técnica, código XHTML resultante es muy "limpio", ya que no se mezcla con el código JavaScript

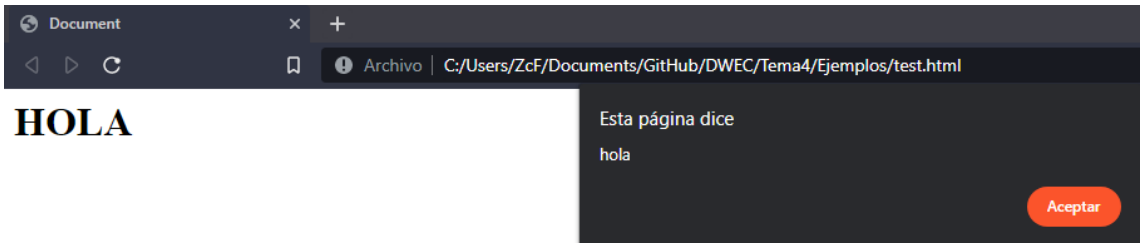
Podemos dividir el código en tres pares:

- La primera, es nuestro elemento HTML al que le hemos puesto una id llamada evento.
- La segunda, son las funciones que van a provocar la acción que necesitamos
- La tercera, son las llamadas a las funciones.

En el siguiente ejemplo, muestra una alerta al cliquear el evento.

```html
<h1 id="evento">HOLA</h1>

<script>
    function clikea() {
        alert('hola');
    }
    document.getElementById('evento').onclick = clikea;
</script>
```



6. ¿Qué es el Event bubbling?

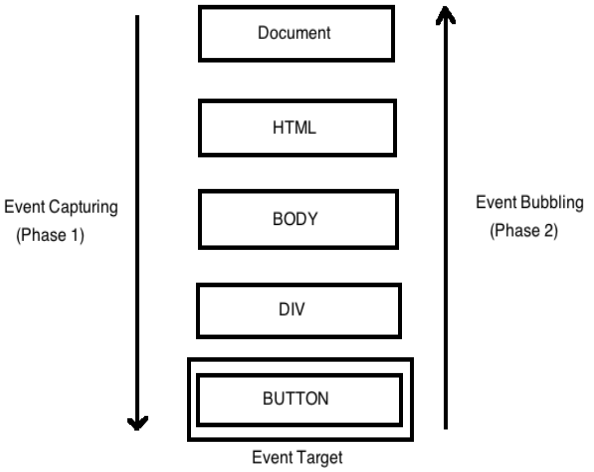
El Event bubbling es un concepto que se refiere al orden de propagación de eventos en el árbol jerárquico de etiquetas HTML. Este concepto afirma que los eventos se propagan desde el elemento HTML que llama al evento hacia arriba.

Es decir, si tenemos un evento que se declara en un button, todos los elementos HTML padres en su rama del árbol serán afectados. Esto significa que hacer clic en un botón también es hacer clic en sus contenedores.

7. ¿Qué es el Event capturing?

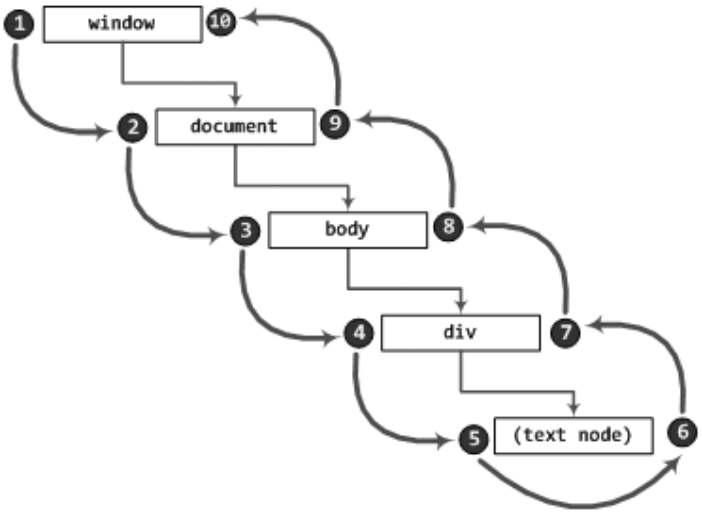
En este modelo, el mecanismo definido es justamente el contrario al "event bubbling".

Por defecto, JavaScript se comporta con Event bubbling, pero podemos alterar el orden de los eventos cambiando al modo captura. Esto se logra añadiendo la propiedad capture:true.



8. ¿Qué es el Event Dom?

El flujo de eventos definido en la especificación DOM soporta tanto el bubbling como el capturing, pero el "event capturing" se ejecuta en primer lugar.



9. Define el objeto event, sus propiedades y métodos

El objeto event es el mecanismo definido por los navegadores para proporcionar toda esa información. Se trata de un objeto que se crea automáticamente cuando se produce un evento y que se destruye de forma automática cuando se han ejecutado todas las funciones asignadas al evento.

Métodos del objeto Event

Métodos	Descripción
`preventDefault()`	Cancela cualquier acción asociada por defecto a un evento.
`stopPropagation()`	Evita que un evento burbujee. Por ejemplo si tenemos un `divA` que contiene un `divB` hijo. Cuando asignamos un evento de click a `divA`, si hacemos click en `divB`, por defecto se dispararía también el evento en `divA` en la fase de burbujeo. Para evitar ésto se puede llamar a `stopPropagation()` en `divB`. Para ello creamos un evento de click en `divB` y le hacemos `stopPropagation()` .

Propiedades del objeto Event

Propiedades	Descripción
`altKey, ctrlKey, metaKey, shiftKey`	Valor booleano que indica si están presionadas alguna de las teclas `Alt`, `Ctrl`, `Meta` o `Shift` en el momento del evento.
`bubbles`	Valor booleano que indica si el evento burbujea o no.
`button`	Valor `integer` que indica que botón del ratón ha sido presionado o soltado, 0=izquierdo, 2=derecho, 1=medio.
`cancelable`	Valor booleano que indica si el evento se puede cancelar.
`charCode`	Indica el carácter **Unicode** de la tecla presionada.
`clientX, clientY`	Devuelve las coordenadas de la posición del ratón en el momento del evento.
`currentTarget`	El elemento al que se asignó el evento. Por ejemplo si tenemos un evento de click en un `divA` que contiene un hijo `divB`. Si hacemos click en `divB`, `currentTarget` referenciará a `divA` (el elemento dónde se asignó el evento) mientras que `target` devolverá `divB`, el elemento dónde ocurrió el evento.
`eventPhase`	Un valor integer que indica la fase del evento que está siendo procesada. Fase de captura (1), en destino (2) o fase de burbujeo (3).
`layerX, layerY`	Devuelve las coordenadas del ratón relativas a un elemento posicionado absoluta o relativamente. Si el evento ocurre fuera de un elemento posicionado se usará la esquina superior izquierda del documento.
`pageX, pageY`	Devuelve las coordenadas del ratón relativas a la esquina superior izquierda de una página.
`relatedTarget`	En un evento de "`mouseover`" indica el nodo que ha abandonado el ratón. En un evento de "`mouseout`" indica el nodo hacia el que se ha movido el ratón.
`screenX, screenY`	Devuelve las coordenadas del ratón relativas a la pantalla dónde se disparó el evento.
`target`	El elemento dónde se originó el evento, que puede diferir del elemento que tenga asignado el evento. Véase `currentTarget`.
`timestamp`	Devuelve la hora (en milisegundos desde `epoch`) a la que se creó el evento. Por ejemplo cuando se presionó una tecla. No todos los eventos devuelven `timestamp`.
`type`	Una cadena de texto que indica el tipo de evento "`click`", "`mouseout`", "`mouseover`", etc.
`which`	Indica el Unicode de la tecla presionada. Idéntico a `charCode`, excepto que esta propiedad también funciona en Netscape 4.

10. ¿Qué tipos de eventos existen según el DOM? indica cada uno de los grupos existentes y ejemplos de cada grupo.

- ***Eventos de ratón***

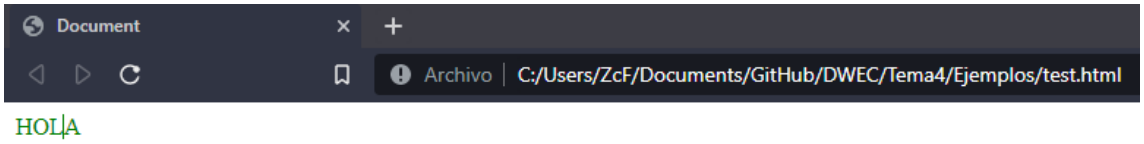
Se originan cuando el usuario emplea el ratón para realizar algunas acciones.

En el siguiente ejemplo, cambia de color al "P" cuando tenemos el ratón pulsado.

```
<p id="myP" onmousedown="mouseDown()" onmouseup="mouseUp()">
    HOLA
</p>

<script>
    function mouseDown() {
        document.getElementById("myP").style.color = "red";
    }

    function mouseUp() {
        document.getElementById("myP").style.color = "green";
    }
</script>
```



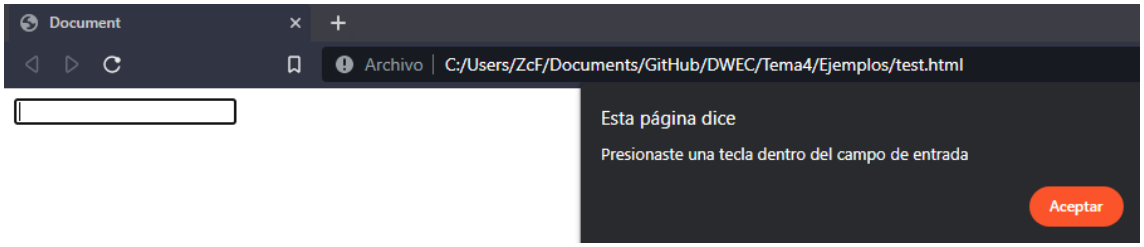
- ***Eventos de teclado***

Se originan cuando el usuario pulsa sobre cualquier tecla de su teclado.

En el siguiente ejemplo, muestra una alerta al teclear dentro del input.

```
<input type="text" onkeydown="myFunction()">

<script>
    function myFunction() {
        alert("Presionaste una tecla dentro del campo de entrada");
    }
</script>
```



- ***Eventos HTML***

Se originan cuando se producen cambios en la ventana del navegador o cuando se producen ciertas interacciones entre el cliente y el servidor.

En el siguiente ejemplo, resetea el contenido del input al pulsar el botón de resetear.

```
<form id="myForm">
    Nombre: <input type="text" name="fname"><br>
    Apellido: <input type="text" name="lname"><br><br>
    <input type="button" onclick="myFunction()" value="Reset form">
</form>

<script>
    function myFunction() {
        document.getElementById("myForm").reset();
    }
</script>
```

- ***Eventos DOM***

Se originan cuando se produce un cambio en la estructura DOM de la página. También se denominan "eventos de mutación".

En el siguiente ejemplo, mueve un elemento de una lista a otra lista.

```
<ul id="myList1">
    <li>Coffee</li>
    <li>Tea</li>
</ul>
<ul id="myList2">
    <li>Water</li>
    <li>Milk</li>
    <li>Juice</li>
</ul>

<p>Haga clic en el botón para mover elementos de una lista a otra:</p>

<button onclick="myFunction()">Mover</button>

<script>
    function myFunction() {
        const node = document.getElementById("myList2").lastElementChild;
        const list = document.getElementById("myList1");
        list.insertBefore(node, null);
    }
</script>
```