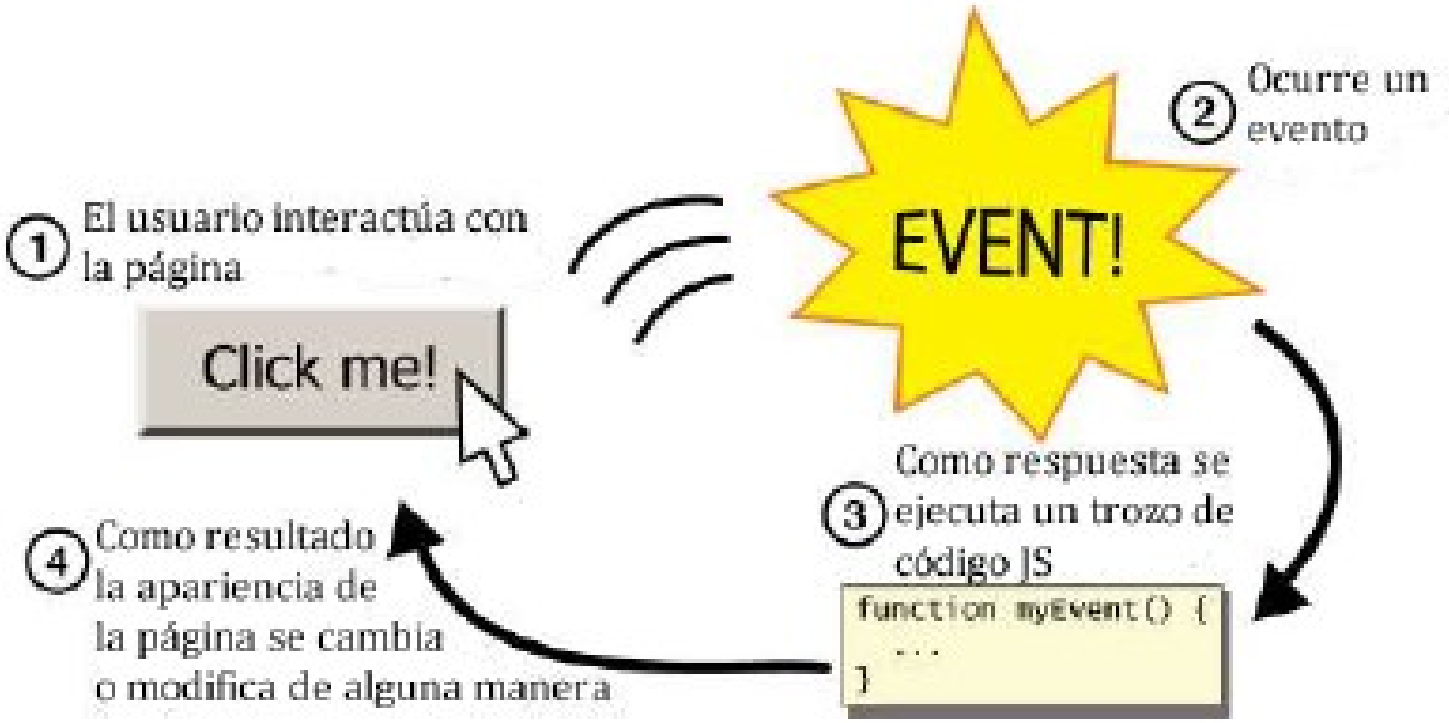
Eventos en JavaScript

Módulo: **Desarrollo Web en entorno cliente**

CFGS Desarrollo de Aplicaciones Web

Introducción

- Los programas JavaScript responden a acciones de los usuarios que llamamos eventos.
- Ayudan a que las aplicaciones web sean más dinámicas e interactivas.



Manejadores de eventos

Las funciones o código JS que se definen para cada evento se denominan “manejadores de eventos” (event handlers) y se pueden definir como:

- Manejadores como atributos de los elementos HTML (inline)
- Manejadores de forma no intrusiva, donde se asigna el nombre de la función a la propiedad Evento de un nodo DOM

Forma inline

- Definir un manejador en un atributo HTML

```
<input type=”button” value=”Pulsa aquí” onClick=”alert(‘hola’);/>
```

- Especificar un valor de retorno, devolviendo true o false se puede anular el comportamiento por defecto de algunos eventos.

```
<a href="hola.htm" onclick="alert('hola');return false;">hola</a>
```

- Inconveniente: Se mezcla JS con HTML.

Forma no intrusiva

- Se accede al objeto, si la etiqueta tiene un id, a través de document.getElementById.
- El objeto tiene propiedades que se corresponden con los atributos HTML. Por ejemplo, un input tendría la propiedad value, etc.
- Ventaja: No se mezcla JS con HTML.

```
<p id="par">Clícame</p>
<script type="text/javascript">
function saludo() { alert("Hola")}
document.getElementById(“par”).onclick = saludo;
//también podríamos definir la función “sobre la marcha” (anónima)
document.getElementById(“par”).onclick = function() {alert(“Hola”)}
</script>
```

Manejadores de eventos y variable this

- Podemos usar la variable **this** para referirse al elemento HTML que ha provocado el evento.

- SIN this

```
<div id="contenidos" style="width:150px; height:60px; border:thin solid silver"
onmouseover="document.getElementById('contenidos').style.borderColor='black';"
onmouseout="document.getElementById('contenidos').style.borderColor='silver';">
Sección de contenidos... </div>
```

- CON this

```
<div id="contenidos" style="width:150px; height:60px; border:thin solid
silver" onmouseover="this.style.borderColor='black';"
onmouseout="this.style.borderColor='silver';">
Sección de contenidos... </div>
```

```
<!DOCTYPE>
<html>
   <head>
      <title>Formas de registrar eventos</title>
      <style type = "text/css">
         div { padding: 5px;
               margin: 10px;
               border: 3px solid #0000BB;
               width: 12em }
      </style>
      <script type="text/javascript">
          // función que gestiona el evento onclick
         function handleEvent() {
            alert("Se ha maneja el evento satisfactoriamente.");
         }
         // registro de evento con el modelo tradicional
         function registerHandler() {
            var traditional = document.getElementById("traditional");
            traditional.onclick = handleEvent; // registro del manejador
         }
      </script>
   </head>
   <body onload="registerHandler()">
     <!-- Registro inline -->
      <div id = "inline" onclick = "handleEvent()">
         Modelo de registro Inline</div>
     <!-- El controlador de eventos se ha registrado con la función registerHandler -->
      <div id = "traditional">Modelo Tradicional</div>
   </body>
</html>
```

Tipos de eventos

Evento	Se aplica a ...	Ocurre cuando ...	Handler
abort	Imágenes	El usuario aborta la carga de una imagen (por ejemplo haciendo click sobre un enlace o en el boton del navegador) .	onAbort
blur	ventanas, frames, y todos los elementos de formularios	El usuario cambia el foco a otro elemento (ventana, frame, o elemento del formulario).	onBlur
click	Casi todos los elementos HTML	El usuario hace click en algo. Devolver false para cancelar la acción por defecto	onClick
change	campos de texto, area de texto, listas de selección.	El usuario cambia el valor de un elemento.	onChange
error	imágenes, ventanas	La carga de un documento o imagen causa un error	onError
focus	ventanas, frames, y todos los elementos de formularios	El usuario pasa el foco a otro elemento (ventana, frame, o elemento del formulario).	onFocus
load	body	El usuario carga la pagina.	onLoad

A evitar…

Es un error común tratar de obtener un elemento de una página antes de que la página se haya cargado.

- El evento onload se dispara siempre y cuando un elemento finaliza su carga satisfactoriamente.
- Si un script icnluido en el head intenta acceder a un nodo DOM del body, getElementById devolverá null debido a que el body todavía no se ha cargado.

Obtener información del objeto event

Se puede obtener información sobre el ratón y el teclado a través del objeto especial llamado **event**. El problema es que no todos los navegadores se comportan igual.

- Navegadores IE, el objeto event se obtiene:

```
var evento = window.event;
```

- Resto, el objeto event se obtiene mágicamente a partir del argumento que el navegador crea automáticamente:

```
function manejadorEventos(elEvento) { var evento = elEvento; }
```

draw.html

```
<script type = "text/javascript">
   <!--
   //initialization function to insert cells into the table
   function createCanvas() {
      var side = 100;
      var tbody = document.getElementById("tablebody");

      for (var i=0; i<side; i++) {
         var row = document.createElement("tr");

         for (var j=0; j<side; j++) {
            var cell = document.createElement("td");
            cell.onmousemove = processMouseMove;
            row.appendChild(cell);
         }
         tbody.appendChild( row );
      }
   }

   //  evento onmousemove
   function processMouseMove(e) {
      // obtener el objeto event para IE
      if (!e)
         var e = window.event;

      // si se pulsa la tecla Ctrl, convertir la celda a azul
      if ( e.ctrlKey )
         this.style.backgroundColor = "blue";

      // si se pulsa la tecla Shift, convertir la
      if ( e.shiftKey )
         this.style.backgroundColor = "red";
   }
   // -->
</script>
```

Asignamos `processMouseMove` como manejador de evento para el evento de las celdas `onmousemove`

Obtenemos el objeto `Event` para el resto de navegadores

Obtiene el objeto `Evento` para IE

Averigua que tecla es pulsada y asigna el color aprpopiado

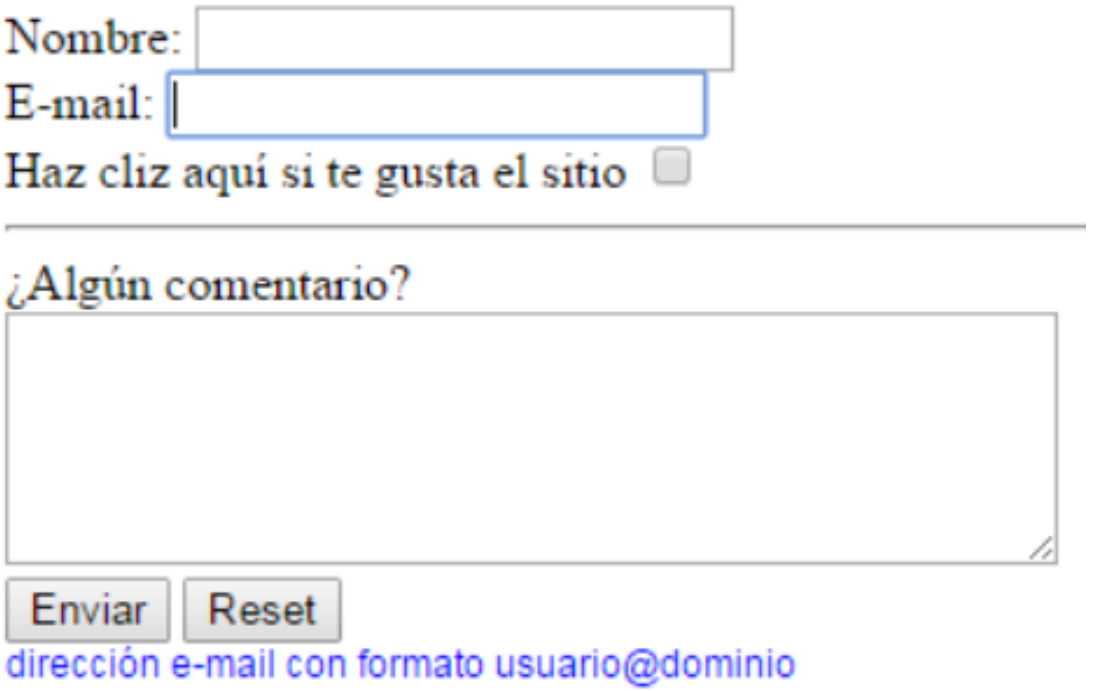
`this` hace referencia a la celda que recibe el evento

Algunas propiedades del objeto event

Property	Description
`altKey`	This value is `true` if the *Alt* key was pressed when the event fired.
`cancelBubble`	Set to `true` to prevent the event from bubbling. Defaults to `false`.
`clientX` and `clientY`	The coordinates of the mouse cursor inside the client area (i.e., the active area where the web page is displayed, excluding scrollbars, navigation buttons, etc.).
`ctrlKey`	This value is `true` if the *Ctrl* key was pressed when the event fired.
keyCode	The ASCII code of the key pressed in a keyboard event. See Appendix D for more information on the ASCII character set.
`screenX` and `screenY`	The coordinates of the mouse cursor on the screen coordinate system.
`shiftKey`	This value is `true` if the *Shift* key was pressed when the event fired.
`type`	The name of the event that fired, without the prefix "`on`".

Ejemplo de procesado de un formulario con onfocus y onblur

- **onfocus**: evento que se lanza cuando el elemento obtiene el foco
- **onblur:** evento que se lanza cuando el elemento pierde el foco



onfocusblur.html

```
<body>
    <form id="myForm" action="">
        <div>
        Nombre: <input type="text" name="name" onfocus="helpText(0)" onblur="helpText(6)" /><br/>
        E-mail: <input type="text" name="e-mail" onfocus="helpText(1)" onblur="helpText(6)" /><br/>
        Haz cliz aquí si te gusta el sitio <input type="checkbox" name="like" onfocus="helpText(2)"
onblur="helpText(6)" /><br/><hr />

        ¿Algún comentario?<br/> <textarea name="comments" rows= "5" cols="45" onfocus="helpText(3)"
onblur="helpText(6)"></textarea>
        <br/>
        <input type="submit" value="Enviar" onfocus="helpText(4)"
        <input type="reset" value="Reset" onfocus= "helpText(5)"
        </div>
    </form>
    <div id="tip" class = "tip"></div>

      <script type="text/javascript">
    <!--
        var helpArray =
          [ "Introduce tu nombre.", // elemento 0
            "dirección e-mail con formato usuario@dominio" , // elemento 1
            "Marcar si te ha gustado nuestro sitio.", // elemento 2
            "Introduce cualquier comentario " , // elemento 3
            "Este botón envía el formulario para su procesamiento", // elemento 4
            "Botón para limpiar el formulario.", // elemento 5
            "" ]; // elemento 6
        function helpText(messageNum) {
          document.getElementById("tip").innerHTML = helpArray[ messageNum ];
        }
    // -->
    </script>
</body>
```

Cuando el usuario hace click dentro de un campo, el evento `onfocus` se dispara, pasándole el número de mensaje apropiado para `HelpText` con el fin de mostrar el mensaje de ayuda

Muestra el mensaje de ayuda correspondiente en el `div` que se encuentra al final del documento

Cuando un elemento pierde el foco, se dispara el evento `onblur`, y se llama a `helpText(6)`, que escribe el mensaje vacío

addEventListener()

Dispone de tres parámetros: tipo de evento, listener o función asociada y useCapture.

```
<script>
        element.addEventListener("click", modifyText, false);
 </script>
```

Los eventos pueden ser activadas en dos ocasiones: al comienzo ("captura") o al final ("burbuja"), y esto se indica con la opción useCapture. Por defecto, el valor es false.

https://developer.mozilla.org/en/DOM/element.addEventListener

addEventListener()

```
document.body.addEventListener('click',       //Type
      function (event) {      //Listener
            console.log("hola");
      },
false); // useCapture
document.body.addEventListener('dblclick',
      function(event) {
            console.log("mundo");},
      false);
```

- En el parámetro *type*, no necesitas escribir el prefijo "on" para cada tipo de evento.
- Utilizando addEventListener(), los manejadores de eventos NO SE SOBREESCRIBEN. Esto quiere decir que puedes utilizar múltiples listener para un mismo elemento.

Eventos: http://www.w3schools.com/jsref/dom_obj_event.asp

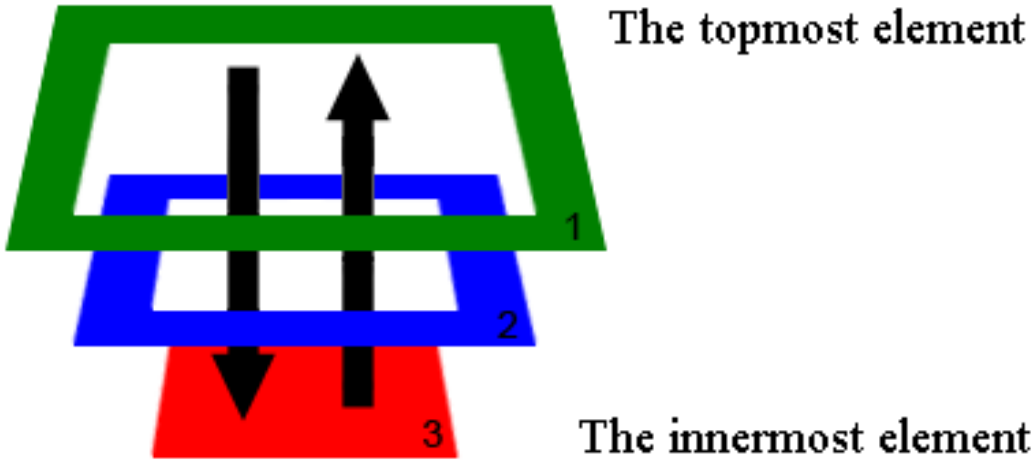
removeEventListener()

Dispone de tres parámetros: tipo de evento, listener o función asociada y useCapture. No se eliminan eventos asociados a funciones anónimas.

```
<div id="myDIV">
      <p>Haz clic sobre el botón para eliminar el evento.</p>
      <button onclick="removeHandler()" id="myBtn">Try it</button>
</div>
<p id="demo"></p>
<script>
document.getElementById("myDIV").addEventListener("mousemove", myFunction);
function myFunction() {
   document.getElementById("demo").innerHTML = Math.random();
}
function removeHandler() {
   document.getElementById("myDIV").removeEventListener("mousemove", myFunction);
}
</script>
```

Fases de captura de los eventos

En todos los navegadores (excepto IE<9) tienen dos estados para el procesamiento de los eventos: fase de captura y fase de burbujeo.



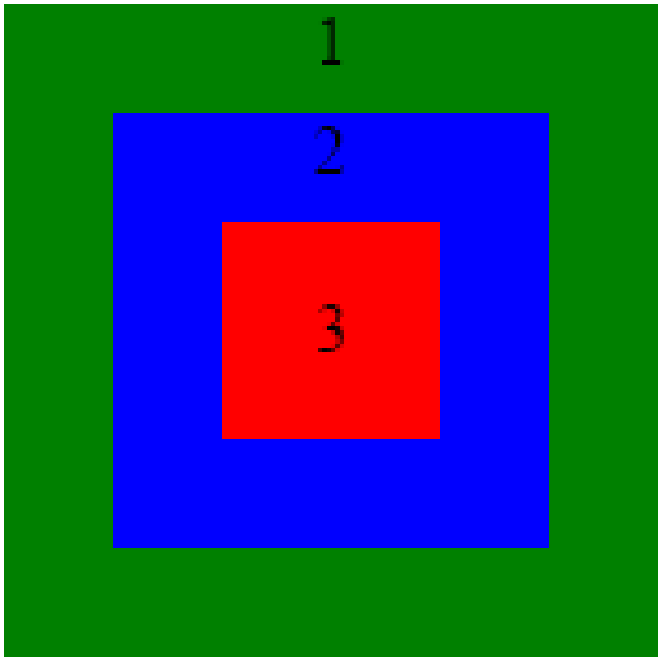
Fase de captura: 1 -> 2 -> 3 (parámetro useCapture = true)

Fase de burbujeo: 3 -> 2-> 1 (parámetro useCapture = false)

Fase de burbujeo

```
<!DOCTYPE HTML>
<html>
<body>
<link type="text/css" rel="stylesheet" href="example.css">

<div class="d1" onclick="highlight(this)">1
    <div class="d2" onclick="highlight(this)">2
        <div class="d3" onclick="highlight(this)">3
        </div>
    </div>
</div>
<script>
function highlight(elem) {
    elem.style.backgroundColor='yellow'
    alert(elem.className)
    elem.style.backgroundColor = ''
}
</script>
</body>
</html>
```



El burbujeo garantiza que al hacer click sobre el div3 se disparará el evento onclick del elemento 3 (conocido como target – event.target) en primer lugar, luego el elemento 2 y, finalmente, el elemento 1.

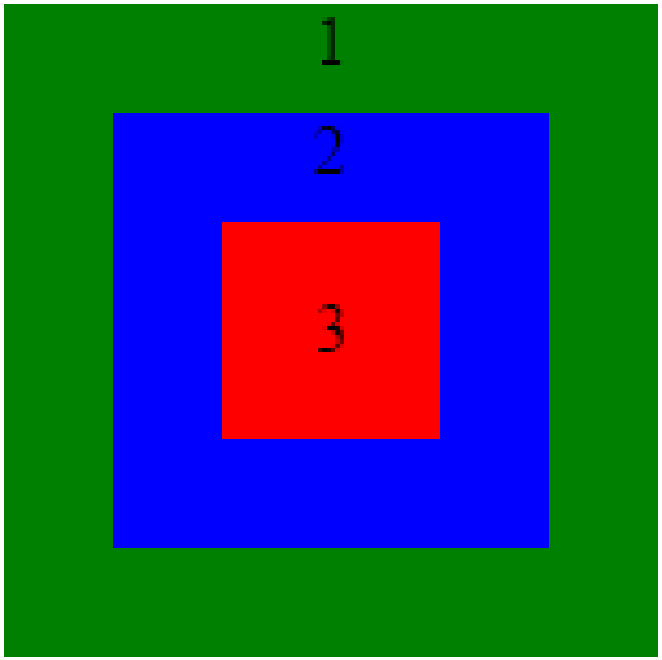
http://javascript.info/tutorial/bubbling-and-capturing

Fase de captura

```
var divs = document.getElementsByTagName('div')

for(var i=0; i<divs.length; i++) {
  divs[i].addEventListener("click", highlightThis, true)
}
```

En este caso se disparará el evento onclic del elemento1, luego el 2 y por el último el 3.



Evento bubbling

- Es el proceso por el cual eventos disparados en elementos secundarios suben (“como las burbujas“) hasta sus elementos padres.
- Cuando un evento se dispara en un elemento, se entrega primero (si lo hay) al manejador del evento del elemento, y a continuación, al manejador de evento del elemento padre (si existe).
- Si solo se va a gestionar el evento de un elemento secundario, debes cancelar el burbujeo del evento en el código fuente del manejador de dicho elemento secundario mediante la propiedad cancelBubble del objeto event para IE < 9 o event.stopPropagation() para navegadores adaptados a la W3C.

```
event.stopPropagation ? event.stopPropagation() : (event.cancelBubble=true)
```

bubbling.html

```
<!-- Cancelar burbujeo de eventos. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
    <head>
        <title>Burbujeo de Eventos</title>
        <script type='text/javascript'>
        <!--
            function documentClick() {
                alert( "Haz hecho click en el documento." );
            }

            function bubble(e) {
                alert('Habrá burbujeo.');
                e.cancelBubble = false;
            }

            function noBubble(e){
                alert('NO Habrá burbujeo.');
                e.cancelBubble = true;
            }

            function registerEvents() {
                document.onclick = documentClick;
                document.getElementById('bubble').onclick = bubble;
                document.getElementById('noBubble').onclick = noBubble;
            }
        // -->
        </script>
    </head>
    <body onload='registerEvents()'>
        <p id='bubble'>Bubbling habilitado.</p>
        <p id='noBubble'>Bubbling NO habilitado.</p>
    </body>
</html>
```

No cancela el bubbling que se produce por defecto

Cancela el Evento bubbling

Registra un Evento para el objeto document

Registra Eventos para eventos haciendo click en 2 elementos p, que son hijos del objeto document