U4. Eventos y Formularios
Parte I. Conceptos básicos sobre eventos

Desarrollo Web en Entorno Cliente

Contenidos

- Modelo de gestión de eventos DOM 1
- Modelo de gestión de eventos DOM 2
  - Asociar funciones a eventos
  - Quitar funciones de eventos
  - Ventajas
  - Disparo de eventos
- Tipos de eventos
  - Eventos de ratón
  - Eventos de teclado
  - Eventos de página (relacionados con formularios)
  - Eventos de DOM

Modelos de gestión de eventos

- Los eventos son mecanismo que se accionan cuando el usuario realiza un cambio sobre una página web.
- 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.

Modelos de gestión de eventos

- Para poder controlar un evento se necesita un manejador.
- El manejador es la palabra reservada que indica la acción que va a manejar.
- En el caso del evento *click*, el manejador sería `onClick`.
- Ejemplo:

```
<IMG SRC="mundo.jpg" onclick="alert('Click en imagen');">
```

Modelos de gestión de eventos

Manejo de eventos asignando función a elemento en HTML:

```
<html>
  <head>
    <title>Página de Evento</title>
    <script>
      function func1() {
        alert("Click en imagen");
      }
    </script>
  </head>
  <body>
    <IMG SRC="mundo.jpg" onclick="func1();”>
  </body>
</html>
```

Modelos de gestión de eventos

Manejo de eventos asignando función a elemento en JS:

```
<html>
<head><title>Página de Evento</title></head>
<body><img src="mundo.jpg">
<script>
      document.getElementsByTagName('img')[0].onclick =
      function() {
        alert("Click en imagen");
      }
</script>
</body>
</html>
```

Modelos de gestión de eventos

Manejo de eventos asignando función a manejador de manera genérica (no a un elemento concreto) en JS:

```
<html>
  <head>
    <title>Página de Evento</title>
    <script>
      onclick = function func1() {
        alert("Click en imagen");
      }
    </script>
  </head>
  <body><IMG SRC="mundo.jpg"></body>
</html>
```

Modelos de gestión de eventos

Manejo de eventos asignando función a manejador de manera genérica (no a un elemento concreto) en JS:

```
<html>
  <head>
    <title>Página de Evento</title>
    <script>
      onclick = function func1(e) {
        alert("Click en " + e.target);
      }
    </script>
  </head>
  <body><h1>Título</h1><img src="mundo.jpg"></body>
</html>
```

Objeto event (e)

- El objeto es el objeto que pasamos como parámetro a las funciones que empleamos para manejar el evento y tiene numerosos métodos y propiedades que son útiles:
  - e.target: quién provocó el evento
  - e.type: de qué tipo es el evento
  - Otros:

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

Modelo gestión eventos DOM 2

- La especificación del DOM de nivel 2 ofrece una manera alternativa de registrar los eventos sobre un objeto determinado mediante **addEventListener()** .
- Este método tiene tres argumentos:
  - el tipo de evento
  - la función a ejecutar
  - un valor booleano que se utiliza para indicar cuándo se debe capturar el evento. Por defecto es false.
- Ejemplo:

elemento.addEventListener('evento', funcion, [false|true])

Asociar eventos DOM 2

- Por ejemplo para asociar la función alertar() al evento click sobre el párrafo con id “mienlace”, haríamos:

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

Asociar eventos DOM 2

- También podemos emplear funciones anónimas:

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

Quitar eventos DOM 2

- Para eliminar un evento de un elemento, usaremos el método **removeEventListener()** :
  - elemento.removeEventListener('evento', función, false|true);

Ejemplo

```
mienlace.removeEventListener("click", alertar);
```

- Para cancelar la ejecución asociada por defecto a un evento, este modelo nos proporciona el método del objeto event **preventDefault()** .

Ejemplo que impide que se abra el enlace:

```
mienlace.addEventListener("click", function(event){
    event.preventDefault()
});
```

Otro ejemplo

Ventajas gestión eventos DOM 2

- Asociar varias funciones a un mismo evento:

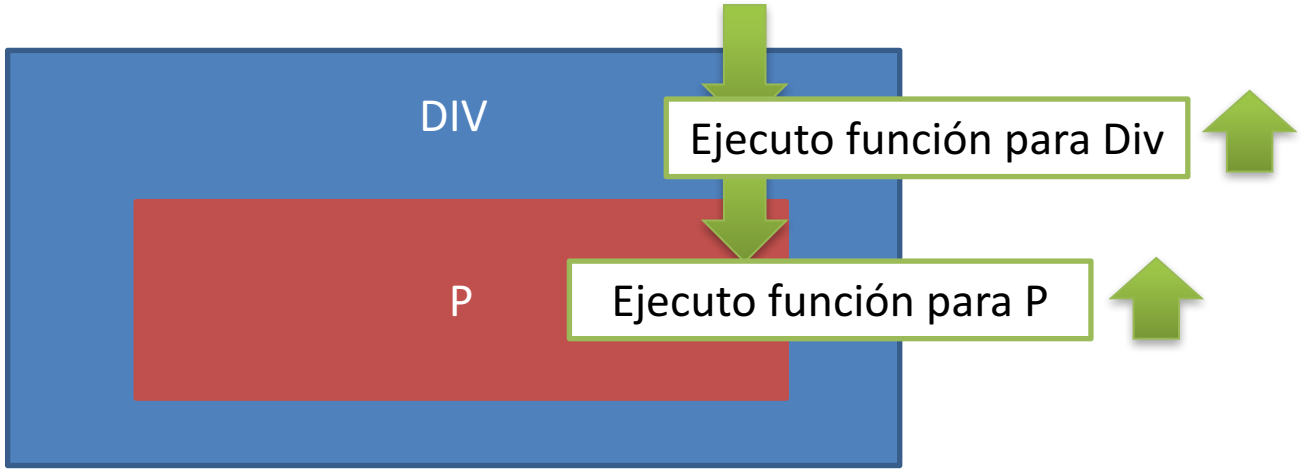
```
mienlace.addEventListener('click',alertar,false);
mienlace.addEventListener('click',avisar,false);
mienlace.addEventListener('click',chequear,false);
```

(También podemos asociar varias funciones a distintos eventos)

- Desasociar eventos de funciones e impedir su ejecución.
- Modificar algunos parámetros de la llamada de eventos. Por ejemplo, cuándo se captura (tercer parámetro que pasamos).

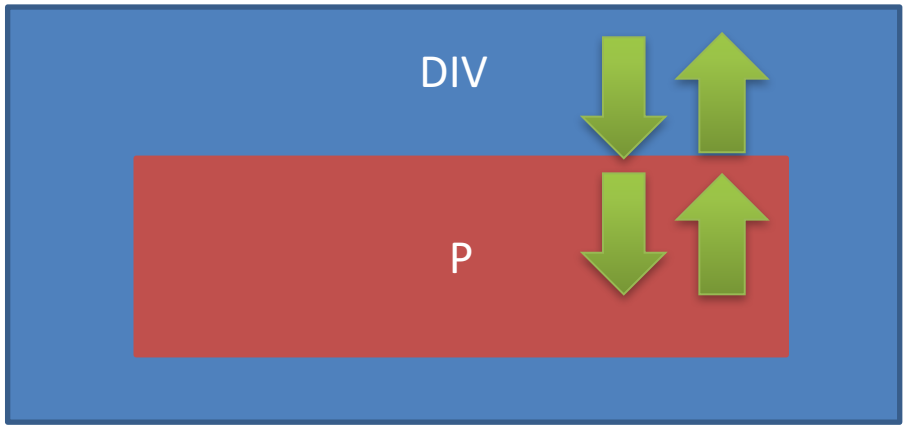
Disparo de eventos

- Cuando se produce un evento, primero se produce la fase de captura hasta llegar al elemento de destino, y luego se producirá la fase de burbujeo hacia arriba.
  - Aquí tenéis un diagrama
  - Aquí podeis ver un ejemplo de qué implica eso.



Disparo de eventos

- Podemos cambiar cuando queremos que se registre el evento:
  - en la fase de captura (ponemos true en el tercer parámetro)
  - en la fase de burbujeo (opción por defecto que equivale a false en el tercer parámetro).



Disparo de eventos

- Para detener la propagación del evento en la fase de burbujeo, disponemos del método **stopPropagation()** .
- En la fase de captura es imposible detener la propagación.

Tipos de eventos

- La especificación DOM define cuatro grupos de eventos dividiéndolos según su origen:
  - Eventos del ratón.
  - Eventos del teclado.
  - Eventos HTML.
  - Eventos DOM.

Eventos de ratón

- 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`.

Eventos de ratón

- Eventos del ratón (2):
  - **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`**.**

Eventos de teclado

- 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`.

Consideraciones eventos de teclado

- Hay dos tipos de códigos diferentes:
  - los códigos de teclado: tecla física del teclado que ha sido pulsada. Los obtenemos con
  - los códigos de caracteres: número asociado a cada carácter, según el juego de caracteres Unicode.
- Hay algunas teclas no imprimibles que no generan evento keypress, sino únicamente eventos keydown y keyup. En cambio, algunas teclas no imprimibles sí generan evento keypress.

Recomendaciones eventos de teclado

- Recomendaciones:
  - Usar **keypress** (no los eventos keydown ni keyup), para una tecla “normal” y consultar la tecla con la propiedad **event.key**.
  - Usar **keyup** para teclas no imprimibles (como flecha de cursor,…) empleando para ello la propiedad **event.keyCode**.
    - En Firefox, podemos tener problemas con event.which con el evento onkeypress. Tenemos las 2 opciones así:

      var x = event.which || event.keyCode;

    - Los caracteres especiales más habituales, tienen una propiedad dedicada: altKey, ctrlKey, metaKey or shiftKey.

Eventos de página

- Eventos HTML (1):
  - **Load**. El evento *load* hace referencia a la carga de distintas partes de la página. Este se produce en el objeto `Window` cuando la página se ha cargado por completo. En el elemento `<img>` actúa cuando la imagen se ha cargado. En el elemento `<object>` se acciona al cargar el objeto completo. 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`.
  - **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`.

Eventos de página

- Eventos HTML (2):
  - **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`.
  - **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`.

Eventos de página

- Eventos HTML relacionados con **Formularios**
  - **Reset**. Este evento se produce cuando pulsamos sobre un botón de tipo reset. El manejador es `onreset`.
  - **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`.
  - **Select**. Se acciona cuando seleccionamos texto de los cuadros de textos `<input>` y `<textarea>`. El manejador es `onselect`.
  - **Change**. Este evento se produce cuando los cuadros de texto `<input>` y `<textarea>` pierden el foco y el contenido que tenían ha variado. También se producen cuando un elemento `<select>` cambia de valor. El manejador es `onchange`.
  - **Submit**. Este evento se produce cuando pulsamos sobre un botón de tipo submit. El manejador es `onsubmit`.

Eventos DOM

- Eventos DOM:
  - **DOMSubtreeModified**. Este evento se produce cuando añadimos o eliminamos nodos en el subárbol de un elemento o documento.
  - **DOMNodeInserted**. Este evento se produce cuando añadimos un nodo hijo a un nodo padre.
  - **DOMNodeRemoved**. Este evento se produce cuando eliminamos un nodo que tiene nodo padre.
  - **DOMNodeRemovedFromDocument**. Este evento se produce cuando eliminamos un nodo del documento.
  - **DOMNodeInsertedIntoDocument**. Este evento se produce cuando añadimos un nodo al documento.