Unitat Didàctica 5

Mecanismes de manejament d'esdeveniments

Tipus d’esdeveniments

- Els esdeveniments són elements de Javascript que ens proporciona el navegador perquè quan es produïsca un esdeveniment s’execute un codi associat a aquest.

- El nom de cada esdeveniment es construïx mitjançant el prefix **on**, seguit del nom en anglés de l’acció associada a l’esdeveniment.

- Algunes accions llancen més d’un esdeveniment, per exemple l’acció submit d’un formulari → onmouseover, onmousedown, onmouseup, onclick i onsubmit i a més a més tenen associada una tasca a desentrollar (enviar el formulari). Es pot impedir aquesta tasca si tornem false (**return false;**) en la funció manejadora

Tipus d'esdeveniments

- La totalitat d'esdeveniments disponibles està descrita en :
  - https://www.w3schools.com/tags/ref_eventattributes.asp
  - http://www.w3schools.com/jsref/dom_obj_event.asp

    - onfocus: obtindre el focus.
    - onblur: perdre e focus.
    - onchange: fer un canvi.
    - onclick: fer un click.
    - ondblclick: fer doble click.
    - onkeydown: polsar una tecla (sense soltar-la).
    - onkeyup: soltar una tecla polsada.
    - onkeypress: polsar una tecla.

Tipus d’esdeveniments

- onload: carregar una pàgina.
- onunload: abandonar una pàgina.

- onmousedown: fer clic de ratolí (sense soltar-lo).
- onmouseup: soltar el botó del ratolí prèviament polsat.

- onmouseover: entrar a un element amb el ratolí.
- onmouseout: eixir d’un element amb el ratolí.

- onsubmit: enviar les dades d’un formulari.
- onreset: resetear les dades d’un formulari.

- onselect: seleccionar un text.

- onresize: modificar el tamany de la página.

Observadors d’esdeveniments

- Les funcions o codi **JavaScript** que es definixen per a cada esdeveniment es denominen "*observador d’esdeveniments*" *i* en **JavaScript** hi ha algunes formes diferents d’indicar els observadors:
  - Observadors com a atributs dels elements HTML.
    - *Complica el codi font de la pàgina, dificultant la modificació i manteniment de la pàgina*.
  - Observadors d’esdeveniments DOM nivell 0.
  - Observadors d’esdeveniments DOM nivell 2.

Observadors d'esdeveniments com a atributs HTML

- Es tracta del mètode més senzill.
  - Cada esdeveniment s'especifica dins de l'etiqueta HTML com a un atribut més de l'etiqueta.
  - El valor ha de ser codi JavaScript a executar.

    <input type="button" value="Pinchame y verás "onclick="alert('Gracias por pinchar');" />

    *<input type="button" value="Pinchame y verás" onclick="**muestraMensaje()**" />*

- Per al manejament d'esdeveniments , es pot utilitzar ***this*** per a referir-se a l'element HTML que ha provocat l'esdeveniment. Si s'utilitza una funció s'ha de passar explícitament (Exemple 5.1)

Observadors (DOM nivell 0).

- Aquesta tècnica consistix en assignar una funció observadora a la propietat DOM de l'esdeveniment .
  - element.onxxx = nom_funció
  - element[“onxxx”] = nom_funció
  - És simple i **funciona en tots els navegadors**.
  - **No es pot passar paràmetres**, ja que no es fa la crida a la funció
- Han d'existir els elements del DOM, és per això que **la pàgina ha d'estar completament carregada**. L'assignació de la funció observadora ha de fer-se en **window.onload**

```
window.onload = function() {
        var btn = document.getElementById(“myBtn”);
        btn.onclick = function(){   alert(“Clicked”); };.....
}
<input type="button" id=”myBtn” value="púlsame">
```

Observadors (DOM nivell 0).

- Si volem llevar la funció observadora de l'esdeveniment s'ha de assignar a **null**

```
btn.onclick = null;
```

- Si volem llançar un esdeveniment per codi, es pot cridar a la funció del mateix nom que l'ésdevenimet.

```
form.submit();
```

- Si utilitzem aquest mètode, la funció observadora es considera que perteneïx a l'element i és per això que s'executa en el seu ambit, no necessitem passar **this**.

```
window.onload = function() {
    var btn = document.getElementById("myBtn");
    btn.onclick = function(){ alert(this.id); }
};
```

Observadors (DOM Nivell 2).

- El DOM Nivell 2 definix dos mètodes per a associar i eliminar els observadors dels esdeveniments: **addEventListener**() i **removeEventListener**()
- Aquests mètodes accepten tres arguments :
  - El nom de l'esdeveniment sense **on**.
  - La funció a assignar.
  - Si la funció observadora es va a utilitzar en la fase de propagació (false) o de captura (true).
- La funció observadora també se considera que pertany a l'element i és per això que s'executa en el seu àmbit, no necessitem passar **this**.

Propagar esdeveniments.

Els navegadors inclouren dos mecanismes anomenats fluxos d'esdeveniments o "events flow", **event bubbling** (propagacio) i **event capturing** (captura>

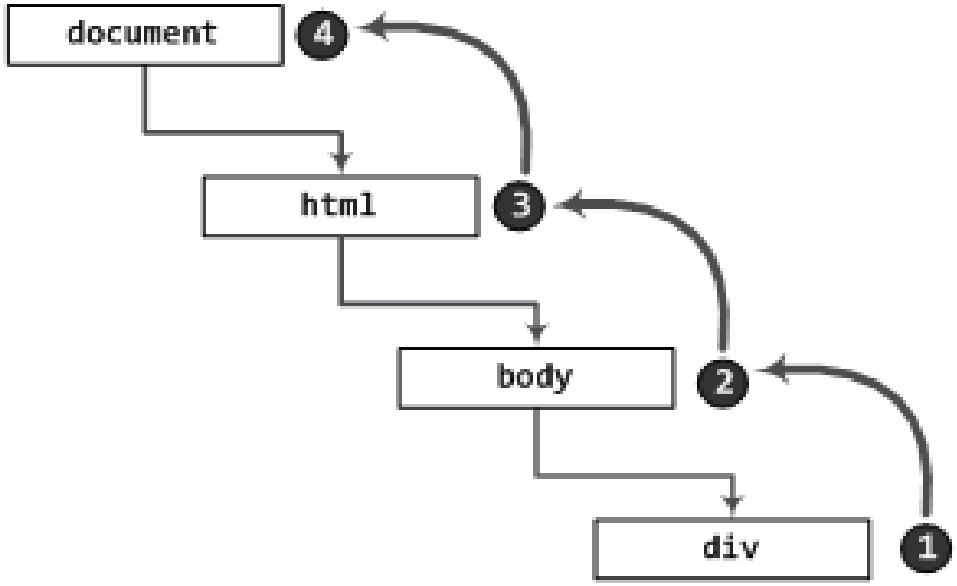
El flux d'esdeveniment permet que elements diferents puguen respondre a un mateix esdeveniment

Exemple :

```
<html onclick="procesaEvento()">
    <head><title>Ejemplo de flujo de eventos</title></head>
    <body onclick="procesaEvento()">
        <div onclick="procesaEvento()">Pincha aqui</div>
    </body
</html>
```

event bubbling

- En aquest model de flux d'esdeveniments, l'ordre que se seguix es des de l'element més específic al menys específic.



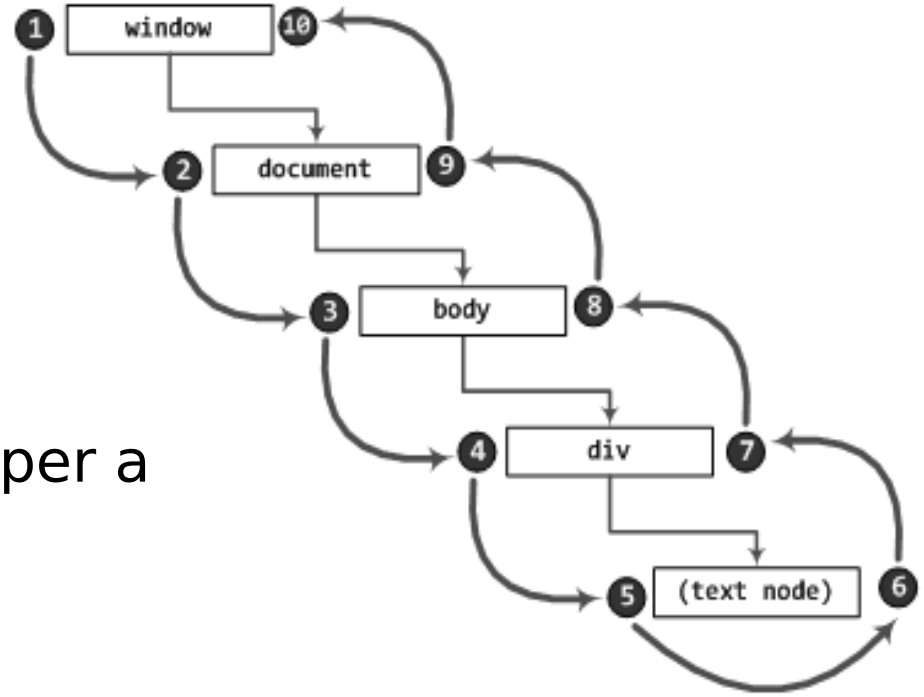
event capturing

- En aquest altre model, el flux d'esdeveniments es definix des de l'element menys específic al més específic, al contrari que "event bubbling"

- **Aquest model , IE no el suporta**, és per això que sempre s'ha d'utilitzar event bubbling (propagació) per compatibilitat de les nostres aplicacions, és a dir tercer **paràmetre addEventListener false**

Esdeveniments DOM

- El flux d'esdeveniments definit en l'especificació DOM suporta tant el bubbling com el capturing, però el "capturing" s'executa en primer lloc.



* Podem utilitzar el mètode **.stopPropagation()** per a detindre el flux

DOM nivell 2

- DOM nivell 2 pot assignar dos observadors a un mateix **esdeveniment** en un mateix **element**.
- Aquests s'executaran en l'ordre que foren assignats.

```
var fnClick1 = function(){alert("Botón pulsado");}
var fnClick2 = function(){alert("Otra vez Botónpulsado");}
var oDiv =document.getElementById("etiqueta");
oDiv.addEventListener("click", fnClick1, false);
oDiv.addEventListener("click", fnClick2, false);
```

DOM Nivel 2 està suportat en IE9+, Firefox, Safari, Chrome y Opera.

Internet Explorer

- En IE versions 8 i inferiors, cada element i objecte window té dos mètodes: **attachEvent() y detachEvent()**.
- **AttachEvent**(): s'utilitza per a connectar un observador a un esdeveniment.
- **DetachEvent**(): servix per al contrari.

- S'utilitzen de la mateixa forma que els anteriors excepte que l'esdeveniment es posa **on…** i també es pot assignar més d'una funció al mateix esdeveniment

Solució cross browser

```
window.onload = function() {

elemento = document.getElementById(“id”);

If (elemento.addEventListener) { // navegadores DOM
    elemento.addEventListener(tipoEvento, funcion, false);
  }
  else if (elemento.attachEvent) { // Internet Explorer
    elemento.attachEvent("on"+tipoEvento, funcion);
  }
  else { // resto de navegadores
    elemento["on"+tipoEvento] = funcion;
  }
};
```

Solució cross browser

```
if (elemento.removeEventListener) { // navegadoresDOM
        elemento.removeEventListener(tipoEvento,funcion,
                                     false);

}

else if (elemento.detachEvent) { // InternetExplorer

        elemento.detachEvent("on"+tipoEvento, funcion);
}

else { // resto de navegadores

        elemento["on"+tipoEvento] = null;
};
```

Obtenint informació de l'esdeveniment (objecte event)

- **JavaScript** permet obtindre informació sobre el ratolí i el teclat mitjançant un objecte especial anomenat **event**.
- En els navegadors tipus **Internet Explorer**, l'objecte **event** se obté directament mitjançant:

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

- Per una altra part, en la resta dels navegadors, l'objecte **event** s'obté a partir de l'argument que el navegador crea automàticament:

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

- El següent codi mostra la forma correcta d'obtindre l'objecte **event** en qualsevol navegador:

```
function manejadorEventos(elEvento) {
    var evento = elEvento || window.event;
}
```

Información sobre el evento

Informació de l'object event :

https://www.w3schools.com/jsref/obj_event.asp

- La propietat ***type*** indica el tipus d'esdeveniment produït, això és útil quan una mateixa funció s'utilitza per a manejar més d'un tipus d'esdeveniment produït,
- La propietat type ens torna el nom de l'esdeveniment però sense el prefix **on** : **var tipo = evento.type;**
- La propietat ***target*** ens torna l'element que llançà l'esdeveniment.
- El mètode **event.preventDefault()** impedix que l'acció predefinida a aquest esdeveniment tinga lloc (útil per a onsubmit)

Esdeveniments de ratolí

- Les coordenades més senzilles són aquelles que es referixen a la posició del punter respecte a la finestra del navegador i s'obtenen mitjançant les propietats ***clientX*** **y** ***clientY*** :

```
function muestraInformacion(elEvento){
    var evento = elEvento || window.event;
    var coordenadaX = evento.clientX;
    var coordenadaY = evento.clientY;
    alert("Has pulsado el ratón en la posición: " + coordenadaX + ", " +coordenadaY);
}
document.onclick = muestraInformacion;
```

Esdeveniments de ratolí

- Les coordenades de la posició del punter del ratolí respecte de la pantalla completa del ordinador de l'usuari se obtenen de la mateixa manera, mitjançant les propietats ***screenX*** **i** ***screenY***:

```
var coordenadaX =evento.screenX;
var coordenadaY =evento.screenY;
```

Esdeveniments de ratolí

- Moltes vegades és necessari obtindre las coordenades que corresponen a la posición del ratolí **respecte l'origen de la página**.
- Aquestes coordenades no sempre coincidixen amb les coordenades respecte a la finestra del navegador, ja que l'usuari pot haver fet scroll en la pàgina.
- Internet Explorer no proporciona aquestes coordenades de forma directa, mentes que la resta de navegadors sí ho fan mitjançant (**pageX, pageY**).

Esdeveniments de ratolí

- És necessari detectar si el navegador és Internet Explorer i en aquest cas fer un càlcul senzill :

```
function myFunction(event) {
 var evento = event || window.event;
 var ie = (navigator.userAgent.toLowerCase().indexOf('msie')!=-1);
 var ie = (event == null );
if (ie) {
        coordenadaX = evento.clientX +  document.body.scrollLeft;
        coordenadaY = evento.clientY + document.body.scrollTop;}
else {
            coordenadaX = evento.pageX;
            coordenadaY = evento.pageY;   }

      alert("Has pulsado el ratón en la posición: " + coordenadaX + ", " +
coordenadaY + " respecto de la página web");
      }
```

Esdeveniments de ratol

- La **variable ie** és **true** si el navegador on s'executa el script és de tipus Explorer (qualsevol versió) i és **false** en qualsevol altre cas.

- Para la resta de navegadors, les coordenades respecte de l'origen de la pàgina s'obtenen mitjançant les propietats ***pageX*** **y** ***pageY***.

- En cas d'Internet Explorer, se obtenen sumant la posició respecte de la finestra del navegador (***clientX***, ***clientY***) **i el desplaçament sufrit per la pàgina (** ***document.body.scrollLeft***, ***document.body.scrollTop*** **)**

- **.scrollLeft i .scrollTop** són propietats de l'objecte **document de DOM**, i es podria utilitzar també les propietas **.pageXOffset i .pageYOffset de l'objecte window de BOM** (es preferix DOM)

Esdeveniments de teclat

- De tots els esdeveniments disponibles en JavaScript, els esdeveniments relacionats amb el teclat són **els més incompatibles entre diferents navegadors** i per això, els més **difícils de manejar.**

- A més a més hi ha tres esdeveniments diferents per a les pulsacions de les tecles (***onkeyup*, *onkeypress* y *onkeydown***)

- Finalment, hi ha dos tipus de tecles: **les tecles *normals*** (com lletres, números i símboles normals) i **les tecles *especials*** (com ENTER, Alt, Shift, etc.)

Esdeveniments de teclat

- Quan un usuari polsa una tecla normal, es produixen tres esdeveniments seguits i en aquest ordre : **onkeydown**, **onkeypress** y **onkeyup**
- L'esdeveniment **onkeydown** correspon amb el fet de polsar una tecla i no soltar-la
- L'esdeveniment **onkeyup** correspon amb el fet de soltar una tecla que estava polsada
- L'esdeveniment **onkeypress** és la pròpia pulsació de la tecla

- L'objecte **event** té dos propietats per a accedir a la informació de la tecla polsada : **keyCode i charCode**.

Esdeveniments de teclat

- Esdeveniments ***keydown i keyup***: En tots els navegadors :
  - **keyCode**: còdi intern de la tecla **charCode**: no definitt

- Esdeveniment ***keypress***:
  - **Internet Explorer**:
    - ***keyCode***: el **códi del caràcter de la tecla que s'ha polsat**
    - ***charCode***: no definit
  - Resta de navegadors:
    - ***keyCode***: per a les tecles normals no definit (algunes versions poden tornar el mateix que charCode) . Per a les tecles especials el còdi intern de la tecla.
    - ***charCode***: per a les tecles normals, el **códi del caràcter de la tecla que s'ha polsat.** Per a les tecles especials, 0. i

Esdeveniments de teclat

Analisi Exemple 5.2

- ***keyup, keydown keyCode*** Si es polsa la mateixa tecla majúscula o minúscula s'obté el mateix codi. **No ens servix per a saber el caràcter polsat**
- **Navegador distint d'IE** : En l'esdeveniment ***keypress***, el valor de la propietat ***charCode*** canvia, ja que el caràcter a no és el mateix que el caràcter A. En aquest cas el valor de *charCode* coincidix amb **el còdi ASCII del caràcter polsat.**

  El caràcter polsat es pot obtindre a partir de :

  String.fromCharCode(charCode)

- **Internet Explorer keyCode**, en l'esdeveniment **keypress**, proporciona **el còdi ASCII del caràcter polsat.**

Esdeveniments de teclat

- Finalment, les propietats ***altKey***, ***ctrlKey*** i ***shiftKey*** emmagatzemen un valor booleà que indica si alguna de aquestas tecles estava polsada al produir-se l'esdeveniment del teclat.

- Aquestas tres propietats funcionen de la mateixa manera en tots els navegadors:

```
if(evento.altKey) {
alert('Estaba pulsada la tecla ALT');
}
```