Esdeveniments

Mòdul professional 6: desenvolupament web en entorn client

Esdeveniments

Mòdul professional 6: desenvolupament web en entorn client

UF1: Sintaxi del llenguatge.
Objectes predefinits del llenguatge.

UF2: Estructures definides pel programador.
Objectes.

UF3: Esdeveniments.
Manejament de formularis.
Model d'objectes del document.

UF4: Comunicació asíncrona client-servidor.

Esdeveniments

En aquesta primera part de la unitat formativa 3 tractarem els **esdeveniments**.

UF3: **Esdeveniments**.
Manejament de formularis.
Model d'objectes del document

Esdeveniments

Dividirem el contingut dels esdeveniments en els següents apartats:

- **Introducció**, per comprendre què son i com funcionen.
- Events del **ratolí.**
- Events del **teclat.**
- Events del **DOM.**
- Nous events en **HTML5.**
- Events de les **WebApi** estàndards.

Esdeveniments

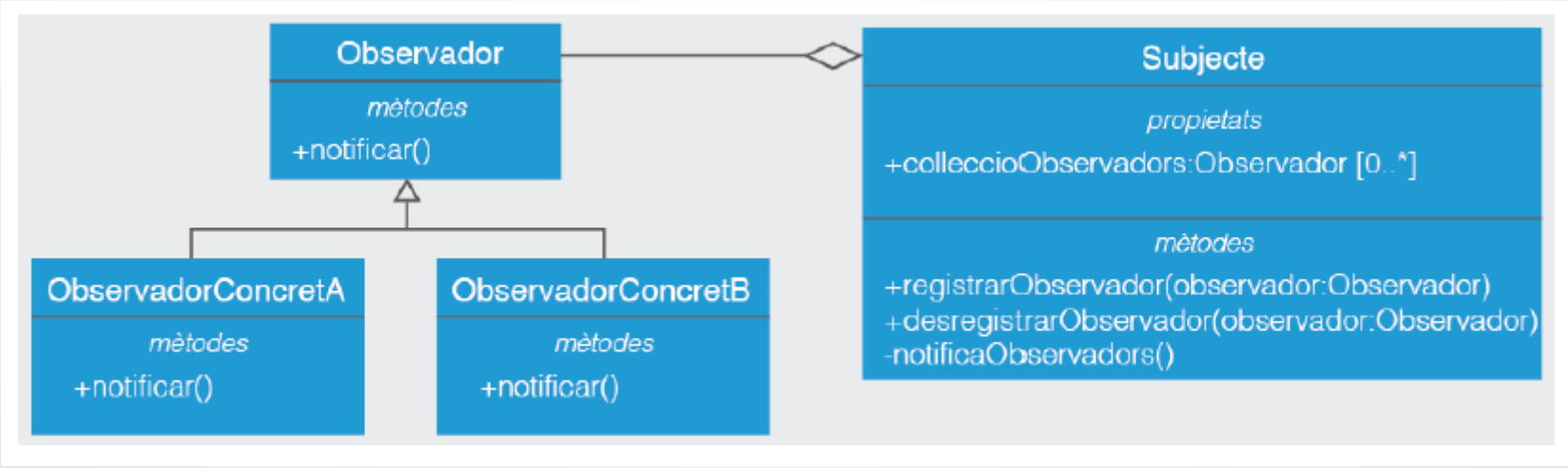
JavaScript

Introducció

Esdeveniments. Introducció 1/9

Els esdeveniments és el resultat d'aplicar el model «observer» a la nostra programació.

Aquest patró s'utilitza quan necessitem un sistema capaç d'indicar quan s'ha produït un esdeveniment i de comunicar-lo només als interessats, que s'han de poder afegir i eliminar dinàmicament.



https://codepen.io/anon/pen/rpGrde?editors=0010

Esdeveniments. Introducció

Una cosa semblant és el que fa el sistema d'esdeveniments (en endavant «events»), però en aquest cas els Subjectes són els elements de la pàgina, per exemple els botons, les imatges, els paràgrafs...

I els observadors, en lloc de ser objectes son funcions que son invocades i s'anomenen **callbacks**. Aquestes funcions poden (opcional) rebre **un paràmetre** que serà l'esdeveniment que s'ha generat.

Veurem a continuació uns quants exemples...

Esdeveniments. Introducció

Començarem veient com podem registrar-nos com a observadors d'un event:

- El paper de l'operació notificar() el realitza la funció alert
- El subjecte és el botó que ens notificarà el moment que es dispari l'event click

```
<button onclick="alert('clic a l\'element');">
    Fes clic aquí!
</button>                                          //01
```

Esdeveniments. Introducció

Recordeu que dos diapositives enrrere esmentavem que podem passar un parametre i que, aquest, correspondrà al esdeveniment que s'ha generat? Aquí ho podeu veure en funcionament:

```
<script>
     function mostraMissatge(e) {
          console.log(e);
          alert("Detectat event tipus: " + e.type);
          }
</script>
<button onclick="mostraMissatge(event);">
     Fes clic aquí!
</button>                                                    //02
```

Executeu el font i observa a la consola el contingut de l'objecte e i dels camps altKey, clientX, screenX,...

Esdeveniments. Introducció

Tot i que es possible afegir codi per reaccionar a *events* directament incrustats al codi HTML, **es considera una mala pràctica**.

Separar sempre hem d'intentar el codi CSS, HTML i JavaScript en blocs diferents i no fer-los *inline*. A més a més com a fitxers externs preferible carregar-los és (css i js).

Esdeveniments. Introducció

Afegir funcions a events

- Obtenim una referència als elements (amb l'**id**)
- Definim **esdeveniments** i els hi assignem una **funció**

```
<button id="primer">1</button>
<button>A</button>
<button id="segon">2</button>
<button id="tercer">3</button>
<script>
   primerElement = document.getElementById("primer"),
   segonElement = document.getElementById("segon");
   primerElement.onclick = function (e) {
      alert("Clic al primer element de tipus: " + e.type);
      };
   segonElement.onclick = function (e) {
      alert("Clic al segon element de tipus: " + e.type);
      };
</script>                                                    //03
```

Esdeveniments. Introducció

Fer-ho així però, té un problema...

```
<button id="primer">1</button>
<script>
   primerElement = document.getElementById("primer");

   primerElement.onclick = function (e) {
      alert("Primera funció");
      }

   primerElement.onclick = function (e) {
      alert("Segona funció");
      }
</script>                                                    //04
```

Veieu el que hi passa? Executeu el codi...

Esdeveniments. Introducció

És molt possible que necessitem afegir dos o més observadors a un subjecte pel mateix *event* o que ho hàgim de fer mes endavant, aixi que la **forma més recomanable** de fer-ho es amb el metode addEventListener:

target.addEventListener(type, listener)

target és l'element al qual s'ha de disparar l'*event perquè* aquest sigui detectat pel **subjecte** (un botó, un paràgraf, una imatge,...).

type és una cadena amb el tipus **d'*event*** que volem escoltar (click, dblclick, load,...).

listener és la funció ***callback*** que serà cridada quan es dispari l'*event* observat.

Esdeveniments. Introducció 9/9

Podem afegir diferents funcions al mateix element i esdeveniment:

```
<button id="primer">1</button>
<script>
   function mostrarMissatge1(e) { alert("primera funció"); }
   function mostrarMissatge2(e) { alert("segona funció");  }

   primerElement = document.getElementById("primer");

   primerElement.addEventListener("click", mostrarMissatge1);
   primerElement.addEventListener("click", mostrarMissatge2);
   </script>
</script>                                                    //05
```

També existeix la funció removeEventListener que fa just el contrari. Prova-la fent que, un cop s'ha executat la primera funció, no es torni a executar més.

Esdeveniments

JavaScript

Events del ratolí

Esdeveniments. Events de ratolí 1/3

Tipus(jQuery)	Operació	Descripció
click	onclick	**Clic** amb qualsevol botó
dblclick	ondblclick	**Doble clic** amb qualsevol botó
mousedown	onmousedown	**Prémer** qualsevol botó
mouseup	onmouseup	**Deixar** anar qualsevol botó
mousemove	onmousemove	El cursor es **mou** per sobre
mouseenter	onmouseenter	El cursor **entra** dins de l'àrea ocupada
mouseleave	onmouseleave	El cursor **abandona** l'àrea ocupada
mouseover	onmouseover	El cursor **entra** dins l'àrea ocupada **o dels seus fills**
mouseout	onmouseout	El cursor **abandona** l'àrea ocupada **o dels seus fills**

Exemple click + click jQuery
Exemple mouseenter i mouseover

Esdeveniments. Events de ratolí 2/3

El següent codi ens permet detectar el moviement del ratolí per sobre d'una imatge:

```
<img id="a" src="img/descarga.jpg">
<script>
   function moviment(e) {
      console.log("El aratolí s'ha mogut");
      }
   var element = document.getElementById("a");
   element.addEventListener("mousemove", moviment);
   </script>                                          //07
```

Amplia el codi per a que t'aparegi per pantalla la posició horitzontal i vertical del cursor, els cops que es detecta l'esdeveniment i la velocitat del ratolí (entre 0 i 10).

Esdeveniments. Events de ratolí 3/3

Tels events que es mostren a continuació no son esdeveniments de ratolí, però els incloem en aquest apartat per resultar més adient. Es tracta d'esdeveniments que només succeeixen amb la interacció amb pantalles tàctils (**tauletes i telèfons mòbils**) i cada cop s'utilitzen més.

Tipus(~~jQuery~~) **Operació**
Descripció

touchstart ontouchstart
Es toca la pantalla encara que s'estigui tocant en un altre punt

touchend ontouchend
Es deixa de tocar la pantalla, encara que hi hagi altres punts pressionats

touchmove ontouchmove
Un dels dits es mou sobre la pantalla

touchcancel ontouchcancel
Es produeix una disrupció (ex:hi ha més punts tocats dels que s'admeten)

Crea una pàgina web on es mostri el nombre de punts que s'estan tocant, i que es mostri un error (sense alert) si es produeix el touchcancel

Esdeveniments

JavaScript

Events del teclat

Esdeveniments. Events de teclat 1/4

Tipus(jQuery)	Operació	Descripció
keydown	onkeydown	Es prem una tecla
keypress	onkeypress	Es prem una tecla (caràcter)
keyup	onkeyup	Es deixa anar la tecla

Per poder disparar *events* de teclat, un element ha de complir alguna de les següents condicions:

- Ha de ser capaç de rebre el *focus*.
- Ha de contenir un element fill capaç de rebre el *focus*.
- L'element body pot disparar events.

Si s'acompleixen alguna d'aquestes condicions i l'element, o algun dels seus descendents, és el *focus* llavors disparara els *events* de teclat.

Esdeveniments. Events de teclat 2/4

Amb el següent codi podreu observar la diferència entre els events keydown i keypress:

```
<body id="all"></body>
<script>
   function notificaObservador(e) {
      console.log("Event "+ e.type)
      console.log(e);
      }
   var element = document.getElementById("all");
   element.addEventListener("keypress", notificaObservador);
   element.addEventListener("keydown", notificaObservador);
   element.addEventListener("keyup", notificaObservador);
   </script>                                                    //09
```

Executa el codi i comprova quan es generen cada un dels esdeveniments pitjan la tecla «a» i la tecla «control».

Esdeveniments. Events de teclat 3/4

Els events es propagen per tots aquells que s'han registrat per atendre'ls i, en el cas anerior, si afeguim un altre element que escolti també els esdeveniments, capturaran l'esdeveniment tots dos.

```
<body id="all"><textarea id="ta"></textarea></body>
<script>
   function notificaObservador(e) {
      console.log("Event "+ e.type)
      console.log(e);
      }
   var element = document.getElementById("all");
   element.addEventListener("keypress", notificaObservador);
   element.addEventListener("keydown", notificaObservador);
   element.addEventListener("keyup", notificaObservador);
   var ta = document.getElementById("ta");
   ta.addEventListener("keypress", notificaObservador);
   ta.addEventListener("keydown", notificaObservador);
   ta.addEventListener("keyup", notificaObservador);
  </script>                                                    //10
```

Executa el codi i modifica'l per saber quin element captura primer l'esdeveniment.
Els events tenen el mètode **stopPropagation()** que el que fan és aturar la propagació a la resta d'observadors. Modifica el codi per a que en el cas del event keypress no es propagi.

Esdeveniments. Events de teclat 4/4

Hi haurà cops que ens interessarà que un event, un cop executada la funció que tingui assignada, no volguem que faci el que el sistema té predeterminat fer (per exemple, no enviar un formulari al servidor...). Per fer això executarem el mètode **preventDefault()** del event corresponent.

El següent codi ens permet executar una funció determinada quan pitgem les tecles alfanumeriques del teclat:

```
<body>   <textarea id="ta"></textarea>   </body>
<script>
   function notificaObservador(e) {
      console.log("Event "+ e.type);
      console.log(e);
      }
   var element = document.getElementById("ta");
   element.addEventListener("keypress", notificaObservador);
   </script>                                                    //11
```

Modifica el codi per a que en el cas que ens pitgin numeros en el textarea, aquests no surtin per pantalla. Dit d'una altra manera, que només hi poguem posar lletres.

Esdeveniments

JavaScript

Events del DOM

Esdeveniments. Events del DOM 1/2

Tipus(jQuery)	Operació	Descripció
DOCUMENT		
load (ready)	onload	Finalitza la càrrega de la pàgina
unload	onunload	Es descarrega (tanca) la pàgina
error	onerror	Es produeix un error
FORMULARIS		
select	onselect	Se selecciona un text
change	onchange	Canvia el valor i perd el focus
submit	onsubmit	S'envia un formulari
reset (~~jQuery~~)	onreset	Es reinicia un formulari
ELEMENTS		
focus	onfocus	Rep el focus
blur	onblur	Perd el focus
resize	onresize	Canvia mida (només «window»)

Esdeveniments. Events del DOM 2/2

En el següent exemple comparem els events keypress i change i si ho executeu veureu els cops que es genera cadascun d'ells.

```
<body>   <textarea id="ta"></textarea>   </body>
<script>
   function notificaObservador(e) {
      console.log("Event "+ e.type);
      console.log(e);
      if (e.type = "change") console.log(e.srcElement.value);
      }
   var element = document.getElementById("ta");
   element.addEventListener("keypress", notificaObservador);
   element.addEventListener("change", notificaObservador);
</script>                                                          //12
```

Si introduiu el vostre nom al textarea, quants cops es genera l'event **keypress**? I l'event **change**?
La instrucció **«if»** que s'ha afeguit, que fa? És pot fer d'una altra manera? És millor fer-ho així?

Esdeveniments

JavaScript

Nous events HTML5

Esdeveniments. Events HTML5 1/3

Tipus(jQuery)	Operació	Descripció
input	oninput	Cada cop que es canvia
invalid	oninvalid	Intenta enviar formulari erroni
canplay	oncanplay	**És possible** fer play
ended	onended	**Finalitza** la reproducció
pause	onpause	És fa **pausa** a la reproducció
play	onplay	**S'inicia** la reproducció
waiting	onwaiting	Falta temporal de dades

Esdeveniments. Events HTML5 2/3

A diferència del event **change**, l'event **input** es dispara cada cop que canvia un element. Comproveu en el següent exemple

```
<body>   <textarea id="ta"></textarea>   </body>
<script>
   function notificaObservador(e) {
      console.log("Event "+ e.type);
      console.log(e);
      }
   var element = document.getElementById("ta");
   element.addEventListener("change", notificaObservador);
   element.addEventListener("input", notificaObservador);
   </script>                                                    //13
```

Esdeveniments. Events HTML5 3/3

Els nous esdeveniments de control d'audio i vídeo ens permeten capturar els esdevenimnts esmentats anterioriorment i que podeu veure en aquest exemple

```
<body>
   <video width="400" controls id="video">
      <source src="img/videoa.mp4" type="video/mp4">
      </video>
</body>
   <script>
      function notificaObservador(e) {
         console.log("Event "+ e.type);
       }
      var element = document.getElementById("video");
      element.addEventListener("canplay", notificaObservador);
      element.addEventListener("ended", notificaObservador);
      element.addEventListener("pause", notificaObservador);
      element.addEventListener("play", notificaObservador);
      element.addEventListener("waiting", notificaObservador);
      </script>
```

Comproveu quan es disparen els events waiting, i canplay, play, pause i ended. Fe-ho amb un vídeo (mp4 o ogg) CURT.

Esdeveniments

JavaScript

Events WebApi

Esdeveniments. Events WebAPI 1/4

API XMLHttpRequest

Permet fer peticions asíncrones al servidor web.

Tipus(jQuery)	Operació	Descripció
abort (~~jQuery)~~	onabort	Es cancel·la la descàrrega
error	onerror	Es produeix un error
load	onload	Finalitza la descàrrega amb èxit
loadend (~~jQuery)~~	onloadend	S'atura la descàrrega
loadstart(~~jQuery)~~	onloadstart	S'inicia la descàrrega
readystatechange	onreadystatechange	Canvia el «readyState»

Es veurà en profunditat en la UF4

API Web Storage

Amb HTML5 podem guardar en el navegador informació, de forma similar a les cookies, per a cada pàgina (objecte localStorage) i per a cada pàgina i sessió (objecte sessionStorage).

L'event associat a aquesta api és «**storage**» (només és permet l'observador «**window**») i es dispara quan hi ha canvis al magatzem de dades.

L'event no és dispara en la pestanya que genera el canvi sinó en la resta que capturin l'esdeveniment.

Esdeveniments. Events WebAPI 3/4

API WebSocket

Amb aquesta API podem obrir una connexió amb un servidor i, sense tallar la connexió, enviar-hi i rebre-hi dades.

Tipus(~~jQuery~~)	**Operació**	**Descripció**
open	onopen	S'estableix una connexió
close	onclose	Es tanca la connexió
message	onmessage	Es rep un missatge
error	error	S'ha tancat la connexió amb pèrdua

Esdeveniments. Events WebAPI 4/4

API Web Workers

Amb aquesta API podem executar processos en segon pla (multifils d'execució).

Tipus(~~jQuery~~)	**Operació**	**Descripció**
message	onmessage	Hi ha missatge per a l'aplicació