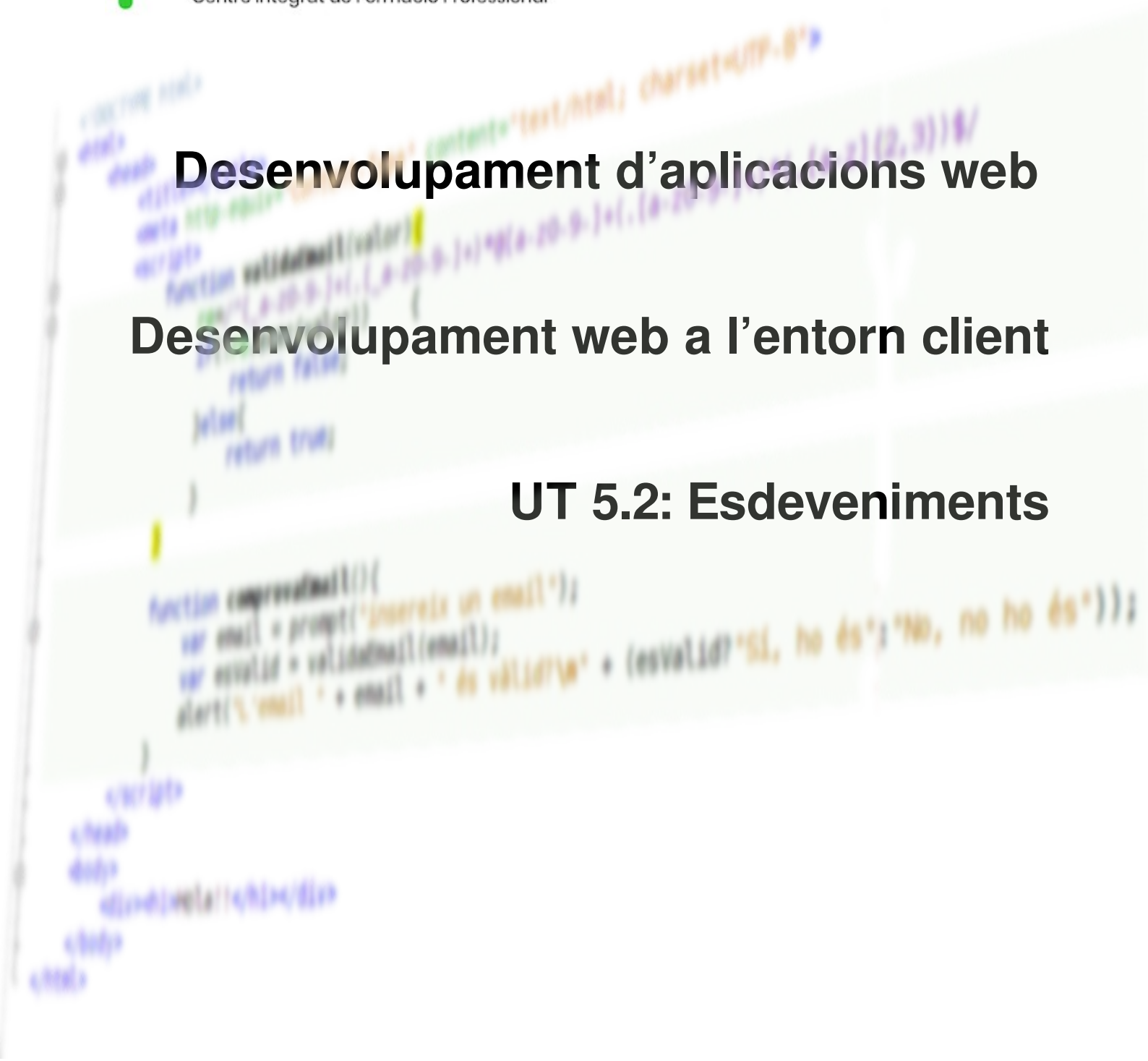
PAU CASESNOVES
Centre Integrat de Formació Professional

Desenvolupament d'aplicacions web

Desenvolupament web a l'entorn client

UT 5.2: Esdeveniments



Joan Pons Tugores

Índex de continguts

Esdeveniments

En informàtica un esdeveniment fa referència a una succés que ha tengut lloc, dins del mateix programa, iniciat per l'usuari, per maquinari, ... al que hem de donar resposta. Per exemple un esdeveniment té lloc quan l'usuari pitja un botó o un arxiu s'ha acabat de carregar.

La programació per esdeveniments és típica de les interfícies d'usuari. La forma de programar és totalment distinta de la programació estructurada que féreu a primer. A primer escriviu el codi en l'ordre en el que s'havia d'executar. Ara en canvi, programarem què volem que faci el navegador quan passi tal cosa, però no podem saber per endavant en quin ordre s'executaran els esdeveniments, ni tan sols si un esdeveniment arribarà a tenir lloc mai.

A partir de la versió 4 HTML incorpora la capacitat de generar i capturar esdeveniments. La resposta a aquests esdeveniments s'ha de programar en algun llenguatge d'script, Javascript en el nostre cas.

Una pàgina HTML pot llençar esdeveniments, per exemple en passar per sobre d'un element, en canviar el contingut d'un camp d'un formulari, en enviar un formulari, ...

Podem assignar una funció que respongui a l'esdeveniment. Aquestes funcions s'anomenen ***listener***. Ho podem fer de tres maneres diferents:

- ~~En línia: incloure el codi de l'esdeveniment dins l'element HTML.~~ **Prohibit!** Dificulta molt el seguiment del codi de la pàgina. S'ha de separar HTML de Javascript i de CSS.

```
<a href="pagina.html" onClick="alert('Has pitjat l'enllaç')">Pitja aquí</a>
```

- **Assignació directa**: Els esdeveniments són també propietats de l'element HTML. Es pot fer una assignació directa a aquests esdeveniments. D'aquesta manera només una funció pot respondre a l'esdeveniment. Només podem tenir una funció que respongui a un determinat esdeveniment. S'afegeix on davant el nom de l'esdeveniment.

```
document.getElementById("...").onclick=function(){...};
```

- **Assignació amb addEventListener**: amb aquesta funció podem assignar diferents funcions al mateix esdeveniment i tenir un major control sobre com s'executen aquestes funcions i les dels elements pare del que dispara l'esdeveniment.

```
document.getElementById("...").addEventListener("click", funcio1, true|false);

document.getElementById("...").addEventListener("click", funcio2, true|false);
```

Objecte Event

En capturar un esdeveniment podem accedir a informació sobre aquest esdeveniment. Per exemple, si capturam un click del ratolí podem comprovar quin botó del ratolí s'ha pitjat o en quina coordenada es trobava el punter quan s'ha pitjat.

Si necessitam aquestes o altres dades de l'esdeveniment podem accedir a un objecte que el navegador passa com a primer argument a la funció que respon a l'esdeveniment.

```
document.getElementById("boto").onclick=function(esdeveniment){
  ...
};
```

O bé

```
function responClick(esdeveniment){
  ...
}
document.getElementById("boto").onclick=responClick;
```

D'aquesta manera dins la funció que respon a l'esdeveniment tenim accés a l'objecte de tipus *Event* que ens passa el *Javascript*.

Propietats de l'objecte Event

Totes són només de lectura.

- **target**: l'element que ha desencadenat l'esdeveniment.
- **currentTarget**: l'element que està tractant l'esdeveniment.
- **type**: el nom de l'esdeveniment.
- **cancelable**: True si es pot anul·lar l'acció per defecte de l'esdeveniment.

Hi ha diversos tipus d'esdeveniments i cada un d'ells té una sèrie de propietats addicionals:

MouseEvent

- **button**: El botó pitjat: 0 esquerra, 1 roda, 2 dret.
- **clientX**, **clientY**, **x, y**: Les coordenades relatives a la finestra on s'ha pitjat el botó.
- **offsetX**, **offsetY**: Les coordenades relatives a l'element que ha disparat l'esdeveniment.
- **screenX**, **screenY**: Les coordenades relatives a la pantalla.
- **altKey**, **ctrlKey**, **metaKey**, **shiftKey**: Tornen true si la tecla alt, ctrl, windows, majúscules estava pitjada quan ha tengut lloc l'esdeveniment.
- **detail**: torna el nombre de vegades que s'ha clickat el mouse.

KeyboardEvent

- **key:** una cadena amb el valor de la tecla pitjada; té en compte si s'ha pitjat juntament amb majúscules o no: "W", "F1", "CAPS LOCK"
- **altKey**, **ctrlKey**, **metaKey**, **shiftKey**: Tornen true si la tecla alt, ctrl, windows, majúscules estava pitjada quan ha tengut lloc l'esdeveniment.
- **location**: Ens permet saber, per exemple, si hem pitjat la tecla ctrl dreta o esquerra.

Mètodes de l'objecte Event

Més endavant.

Catàleg d'esdeveniments

Els distints objectes poden llençar distints esdeveniments. A continuació teniu els més habituals:

Esdeveniments de window

- **load**: té lloc quan la pàgina s'ha acabat de carregar, i per tant el DOM està format i tots els scripts, imatges, ... carregats. És molt útil quan tenim un script que ha d'inicialitzar d'alguna manera la pàgina. Ens assegura que tots els elements existeixen.

```
window.onload=function(){}
```

- **beforeunload**: es dispara abans que la pàgina es descarregui, per exemple perquè l'usuari ha navegat a una altra pàgina. La funció ha de tornar un valor distint de null per que el navegador demani confirmació per abandonar la pàgina a l'usuari. En alguns navegadors es mostra el valor retornat per la funció quan es demana confirmació.

Esdeveniments de document

- **DOMContentLoaded**: Es dispara quan l'arbre de la pàgina s'ha completat i els scripts amb *defer* o de tipus mòdul s'han executat. No espera que s'hagin descarregat les imatges, els fulls d'estil o scripts asíncrons.

Esdeveniments de img

- **abort**: té lloc quan s'ha interromput la càrrega d'una imatge.

Esdeveniments de form

- **reset**: té lloc just abans de reinicialitzar el formulari. El podem utilitzar per demanar confirmació a l'usuari.
- **submit**: es dispara just abans d'enviar les dades del formulari on digui l'atribut *action* del formulari. Bon lloc per fer validacions.

No s'haurien d'utilitzar mai l'onclick dels botons per aquestes tasques. Es pot fer un *reset* o un *submit* d'altres maneres i amb *l'onclick* no les controlaríem, en canvi aquests esdeveniments es disparen sigui quina sigui la forma en que s'ha fet el *submit* o el *reset*.

Esdeveniments dels controls d'un formulari

- **focus**: el cursor ha entrar a l'element, amb el tabulador o perquè l'usuari hi ha clicat dins. No es propaga.
- **focusin**: el cursor ha entrar a l'element, amb el tabulador o perquè l'usuari hi ha clicat dins. Es propaga.
- **blur**: l'element deixa de tenir el focus, l'usuari ha sortit de l'element. No es propaga.
- **focusout**: l'element deixa de tenir el focus, l'usuari ha sortit de l'element. Es propaga.
- **change**: onblur però a més el contingut del camp ha d'haver canviat.
- **input**: immediatament quan el valor del control ha canviat, sense esperar que l'element perdi el focus.
- **select**: es dispara quan hem seleccionat el contingut de l'element.

Esdeveniments de teclat

- **keydown**: una tecla s'ha pitjat estant el focus a l'element. S'ha pitjat, però encara no s'ha amollat.
- **keyup**: una tecla s'ha amollat estant el focus a l'element. S'ha pitjat, i s'ha amollat.
- **keypress**: hi ha hagut una pulsació de tecla, és a dir, la tecla s'ha pitjat i s'ha amollat. Està marcat com a ***Deprecated*** per tant pot ser que deixi de funcionar en qualsevol actualització del navegador. Es recomana utilitzar k*eydown*.

És a dir, que una pulsació de tecla genera tres esdeveniments: down, up i press.

Esdeveniments del ratolí

- **click**: S'ha fet click a l'element amb el botó principal del ratolí (esquerra normalment). Es genera també un esdeveniment mousedown i un mouseup.
- **dblclick**: S'ha fet doble click a l'element amb el botó principal del ratolí (esquerra normalment). Es generen també esdeveniments onclick, ...
- **contextmenu:** S'ha fet click a l'element amb el botó secundari del mouse.
- **mouseover**: El punter del ratolí entra a l'element o a qualsevol dels fills.
- **mouseenter**: El punter del ratolí entra a l'element.
- **mouseout**: El punter del ratolí surt de l'element o entra o surt de qualsevol dels seus fills.
- **mouseleave**: El punter del ratolí surt de l'element.
- **mousemove**: El punter del ratolí es mou sobre l'element.
- **mousedown**: Qualsevol botó del ratolí es pitja dins l'element.
- **mouseup**: Qualsevol botó del ratolí s'amolla dins l'element.

Esdeveniments dels elements

A més dels esdeveniments que hem vist fins ara qualsevol element del document pot disparar altres esdeveniments.

- **Copy**: es dispara quan l'usuari inicia al navegador l'acció de copiar. L'acció per defecte d'aquest esdeveniment és copiar la selecció al porta papers. Amb aquest esdeveniment podem modificar el porta papers, però no podem llegir el que hi hagi.

```
const source = document.getElementById("source");
source.addEventListener("copy", (event) => {
  //Recull la selecció.
  const selection = document.getSelection();
  //Modifica la selecció i l'afegeix al porta papers
  event.clipboardData.setData("text/plain",
          selection.toString().toUpperCase());
  //Evita l'acció per defecte de l'esdeveniment
  event.preventDefault();
});
```

- **cut**: Com *copy*. Pot modificar la selecció abans de posar-la al porta papers. Si anul·lam l'acció per defecte haurem de programar l'eliminació de la selecció.

- **paste**: Aferra el contingut del porta papers a l'element que el dispara si és editable. Podem modificar el contingut aferrat aturant el comportament per defecte.

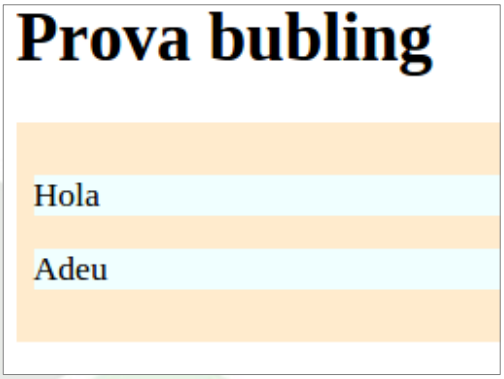
```
const target = document.querySelector("target");

target.addEventListener("paste", (event) => {
  event.preventDefault();
```

```
  // Recull el contingut del portapapers
  let paste = event.clipboardData.getData("text");
  // El modifica
  paste = paste.toUpperCase();
  const selection = window.getSelection();
  if (!selection.rangeCount) return;
  selection.deleteFromDocument();
  selection.getRangeAt(0).insertNode(document.createTextNode(paste));
  selection.collapseToEnd();
});
```

Bubbling o propagació d'esdeveniments

Suposem que tenim la següent estructura a la pàgina html, un div que conté dos paràgrafs, i que volem programar *l'onclick* del div.

```
<h1>Prova bubling</h1>
<div id="capa">
    <p id="hola">Hola</p>
    <p id="adeu">Adeu</p>
</div>
```



A la imatge s'han pintat els paràgrafs de blau cel i el div de beig. Realment l'única part del div que es veu i on hauria de clicar l'usuari és l'espai entre paràgrafs i la vora. Segurament no volem exigir a l'usuari tanta punteria, per això es va idear el *bubbling* o *propagació d'esdeveniments*.

El *bubbling* consisteix en que si un esdeveniment té lloc a un element, s'executa el *listener*, la funció associada a l'esdeveniment d'aquell element, i després aquest esdeveniment es propaga a l'element que el conté i així successivament fins arribar a l'arrel. Per això es diu *bubbling*, perquè l'esdeveniment puja com una bombolla per el DOM.

Això vol dir que en el nostre cas si programam *l'onclick* del *div* i clicam sobre un paràgraf, s'executarà *l'onclick* del *div*.

1. Si el paràgraf té un listener per l'onclick s'executarà.
2. Es propagarà l'esdeveniment. Es dispararà l'onclick del *div*. S'executarà el listener que tengui assignat.
3. Es propagarà l'esdeveniment. Es dispararà l'onclick del pare del *div ...*

addEventListener

Com ja hem dit abans a la funció que respon a un esdeveniment se l'anomena *listener*.

L'objecte *Element* que ens torna *getElementById* té dos mètodes relacionats amb la gestió d'esdeveniments:

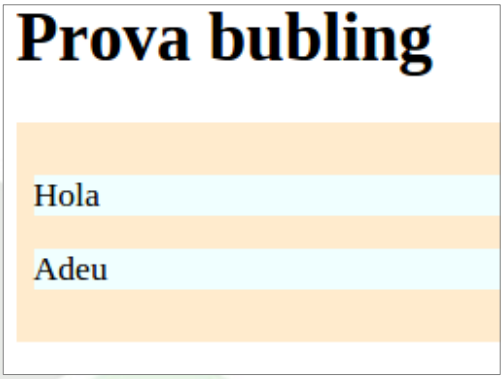
- **addEventListener**: permet afegir un listener a un esdeveniment de l'element. Té tres paràmetres:
  - El nom de l'esdeveniment, sense 'on'
  - El nom de la funció externa o una funció anònima
  - *useCapture*: true o false. Opcional (més endavant)

  Amb *addEventListener* podem associar tantes funcions com volguem al mateix esdeveniment. **S'executaran totes en l'ordre en el que s'han afegit**.

- **removeEventListener**: permet eliminar un *listener* d'un esdeveniment de l'element sempre que s'hagi definit com una funció externa amb nom. Té dos paràmetres:
  - El nom de l'esdeveniment, sense 'on'
  - El nom de la funció externa

Trickling o captura d'esdeveniments

```
<h1>Prova bubling</h1>
<div id="capa">
    <p id="hola">Hola</p>
    <p id="adeu">Adeu</p>
</div>
```



Suposem que tenim la mateixa estructura a la pàgina html, una capa que conté dos paràgrafs, i que volem programar *l'onclick* de la capa i els paràgrafs.

El comportament per defecte és el de *bubbling*. Primer s'executa el *listener* de l'esdeveniment de l'element intern i després de l'extern.

Si volem invertir l'ordre per defecte ho podem fer utilitzant *addEventListener* i posant el tercer paràmetre, *useCapture*, a *true*. Això és el que s'anomena captura d'esdeveniments o *trickling* (degoteig).

La captura d'esdeveniment no significa que si picam, en aquest cas, sobre el div s'executin els esdeveniments dels dos paràgrafs. Significa que si picam sobre un paràgraf primer s'executa el listener del div i després el del paràgraf on hem clicat. Si clicam sobre el div només s'executa el listener del div.

Això vol dir que en el nostre cas si programam *l'onclick* del *div* i clicam sobre un paràgraf, s'executarà *l'onclick* del *div*.

1. Si l'element *html* té un listener per l'onclick l'executarà.
2. Si l'element *body* té un listener per l'onclick l'executarà.
3. L'esdeveniment es propagarà per tots els antecessors del paràgraf fins arribar al *div* i si tenen un o més listeners s'executarà.
4. Es disparará l'onclick del *div*. S'executarà el listener que tengui assignat.
5. Es propagarà l'esdeveniment. Es disparará l'onclick del paràgraf. Acabarà.

Mètodes de l'objecte Event

- **.preventDefault**: Si l'esdeveniment es cancel·lable l'acció per defecte associada a l'esdeveniment s'anul·la. Es equivalent a que la funció *listener* faci *return false*. NO atura la propagació o la captura d'esdeveniments.

  Per exemple, si ho posam dins un *onsubmit* el formulari no s'enviarà. O si el posam dins un *oncontextmenu* no mostrarà el menú contextual.

- **.stopPropagation**: evita que l'esdeveniment es propagui, arribi als elements que contenen l'element que l'executa. De vegades ens pot ser útil la propagació d'esdeveniments, però d'altres pot ser que no volguem aquest comportament. Per exemple, en el cas anterior, si programam *l'onclick* dels paràgrafs i volem que si l'usuari pitja sobre un paràgraf no s'executi el click del *div*. No evita que s'executin els altres *listeners* que hi pugui haver associats al mateix esdeveniment de l'element.

- **.stopImmediatePropagation**: Evita que altres *listeners* del mateix esdeveniment del mateix element s'executin. Atura també la propagació als elements que contenen l'actual.