#### Gestión de eventos III

avanzado

#### Captura y Burbuja

- Estas son dos etapas distintas de la ejecución de un evento.
- La primera captura (capture en Inglés) se ejecuta antes del comienzo del evento,
- Mientras que la segunda, burbuja (bubbling en Inglés), se ejecuta después de que el evento se activó.
- Estos dos parámetros definen el sentido de propagación de los eventos.

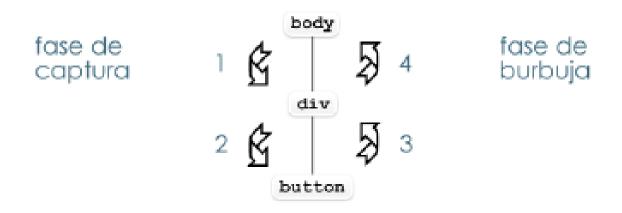
#### Ejemplo de propagación de un evento

```
Código: HTML

<div>
<span> texto </span>
</div>
Si asignamos un evento click a cada uno de los dos elementos ¿cuál se ejecuta primero?
```

#### Ejemplo de propagación de un evento

• La respuesta esta en el método de propagación que utilicemos.



# Código html

- <div id="capt1">
- <span id="capt2">Instrucciones para la fase de captura. </ span>
- </div>
- <div id="burb1">
- <span id="burb2">Instrucciones para la fase de burbuja. </span>
- </div>

#### script

#### Extraemos los nodos

- capt1 = document.getElementById ('capt1')
- capt2 = document.getElementById ('capt2')
- burb1 = document.getElementById ('burb1')
- burb2 = document.getElementById ('burb2');

#### Script:addEventListener(False/True)

```
capt1.addEventListener ('click', function () {alert ("El
evento de div acaba de desencadenarse.");},true);
capt2.addEventListener ('click', function () {alert ("El
evento de span acaba de desencadenarse.");},true);
```

```
burb1.addEventListener ('click', function () {alert ("El
evento de div acaba de desencadenarse.");},false);
burb2.addEventListener('click', function() {alert ("El
evento de span acaba de desencadenarse.");},false);
```

- Obtener el elemento del actual evento disparado.
- Una de las propiedades más importantes de nuestro objeto de destino se llama target.

#### **Ejemplo:**

```
<span id="pulsame"> Pulse aquí </ span>
<script>
var pulsame = document.getElementById ('pulsame');
pulsame.addEventListener ('click', function (e)
{e.target.innerHTML = 'Ha hecho clic en';}, false);
</script>
```

- Obtener el elemento en el origen del evento desencadenante.
- La solución es simple: utilizar la propiedad currentTarget en lugar de target. Probar el ejemplo primero con target y después con currentTarget.

Ejemplo:

```
<div id="padre1">Padre
<div id="child1"> Niño N º 1 </div>
<div id="child2"> Niño N ° 2 </div>
</div>
</div>
<script>
var padre1 = document.getElementById ('padre1')
result = document.getElementById ('resultado');
padre1.addEventListener ('mouseover', function (e) {result.innerHTML = "El desencadenador del evento \ "MouseOver \" tiene la id: "+ e.target.id;
}, false);
</script>
```

- Recuperar la posición del cursor
- Ejemplo:

```
<div id="posicion"> </div>
<script>
var posicion = document.getElementById ('posicion');
document.addEventListener ('mousemove', function (e) {
posicion.innerHTML = 'Posición X' + e.clientX + "px <br/>
  Posición
Y: '+ e.clientY +' px ';}, false);
</script>
```

- Recuperar el elemento en relación a un evento de ratón: relatedTarget y se utiliza con los eventos mouseover y mouseout.
- En el caso:
- mouseout, proporciona el objeto del elemento en el cual el cursor se ha introducido.
- mouseover, dará el objeto del elemento del que el cursor acaba de salir.

- <div id="padre1">
- Padre: 1 <br />
- Mouseover en el niño
- <div id="child1"> Niño N º 1 </ div>
- </div>
- <div id="padre2">
- Padre 2 <br />
- Mouseout en el niño
- <div id="child2"> Niño N ° 2 </ div>
- </div>
- <script>
- var child1 = document.getElementById ('child1')
- child2 = document.getElementById ('child2')
- resultado = document.getElementById ('resultado');
- child1.addEventListener ('mouseover', function (e) {result.innerHTML = "El elemento de la izquierda justo antes del cursor, que no entra sobre el niño # 1 es: "+ e.relatedTarget.id;}, false);
- child2.addEventListener ('mouseout', function (e) {result.innerHTML = "El elemento volado inmediatamente después del cursor que ha dejado el niño # 2 es: "+ e.relatedTarget.id;}, false);
- </script>

- Recuperar las teclas pulsadas por el usuario
- Los eventos KeyUp y KeyDown están diseñados para capturar las pulsaciones de teclado.(letras, crtl...etc)
- El evento keypress, tiene un propósito completamente diferente: capturará las teclas que escriben un carácter.
- Su ventaja radica en su capacidad para detectar las combinaciones de teclas.

si se ejecuta la combinación:

 Mayús + A, el evento keypress detectará una A mayúscula donde los eventos KeyDown y KeyUp se dispararán dos veces, una vez para la tecla Shift y una segunda vez para la tecla A.

- Si tuviéramos que enumerar todas las propiedades que pueden aportar valor, habría tres: keyCode, charCode y which.
- Estas propiedades devuelven el código ASCII correspondiente a la tecla pulsada. Sin embargo, la propiedad keyCode es más que suficiente en todos los casos.

#### Práctica:

Realizar una tabla de tres celdas en las cuales ponga: 1-pulsa 2-Presiona 3-suelta.

Asignaremos con addEventlistener los eventos correspondientes a cada elemento y visualizaremos el keycode de cada event en un elemento preparado para ello.

Después de probar esto convertir el resultado con fromCharCode.

 Bloquear la acción por defecto de ciertos eventos. <a id="link" href="http://www.barzanallana.es"> Pulse aquí</ a> <script> var link = document.getElementById ('link'); link.addEventListener ('click', function (e) { e.preventDefault () // Se bloquea la acción predeterminada de este evento alert ('Ha hecho clic en');}, false); </script>