El objeto event

Cuando se produce un evento, no es suficiente con asignarle una función responsable de procesar ese evento. Normalmente, la función que procesa el evento necesita información relativa al evento producido: la tecla que se ha pulsado, la posición del ratón, el elemento que ha producido el evento, etc.

El objeto `event` es el mecanismo definido por los navegadores para proporcionar toda esa información. Se trata de un objeto que se crea automáticamente cuando se produce un evento y que se destruye de forma automática cuando se han ejecutado todas las funciones asignadas al evento.

Internet Explorer permite el acceso al objeto event a través del objeto `window`:

```
elDiv.onclick = function() {
  var elEvento = window.event;
}
```

El estándar DOM especifica que el objeto `event` es el único parámetro que se debe pasar a las funciones encargadas de procesar los eventos. Por tanto, en los navegadores que siguen los estándares, se puede acceder al objeto `event` a través del array de los argumentos de la función:

```
elDiv.onclick = function() {
  var elEvento = arguments[0];
}
```

También es posible indicar el nombre argumento de forma explícita:

```
elDiv.onclick = function(elEvento) {
  ...
}
```

El funcionamiento de los navegadores que siguen los estándares puede parecer "*mágico*", ya que en la declaración de la función se indica que tiene un parámetro, pero en la aplicación no se pasa ningún parámetro a esa función. En realidad, los navegadores que siguen los estándares crean automáticamente ese parámetro y lo pasan siempre a la función encargada de manejar el evento.

1. Propiedades y métodos

A pesar de que el mecanismo definido por los navegadores para el objeto `event` es similar, existen numerosas diferencias en cuanto las propiedades y métodos del objeto.

1.1. Propiedades definidas por Internet Explorer

La siguiente tabla recoge las propiedades definidas para el objeto `event` en los navegadores de la familia Internet Explorer:

Propiedad/Método	Devuelve	Descripción
altKey	Boolean	Devuelve `true` si se ha pulsado la tecla ALT y `false` en otro caso
button	Número entero	El botón del ratón que ha sido pulsado. Posibles valores: 0 – Ningún botón pulsado 1 – Se ha pulsado el botón izquierdo 2 – Se ha pulsado el botón derecho 3 – Se pulsan a la vez el botón izquierdo y el derecho 4 - Se ha pulsado el botón central 5 – Se pulsan a la vez el botón izquierdo y el central 6 - Se pulsan a la vez el botón derecho y el central 7 – Se pulsan a la vez los 3 botones
cancelBubble	Boolean	Si se establece un valor `true`, se define el flujo de eventos de tipo *bubbling* (rellamadas)
clientX	Número entero	Coordenada X de la posición del ratón respecto del área visible de la ventana
clientY	Número entero	Coordenada Y de la posición del ratón respecto del área visible de la ventana
ctrlKey	Boolean	Devuelve `true` si se ha pulsado la tecla CTRL y `false` en otro caso
fromElement	Element	El elemento del que sale el ratón (para ciertos eventos de ratón)
keyCode	Número entero	En el evento `keypress`, indica el carácter de la tecla pulsada (ASCII). En los eventos `keydown` y `keyup` indica el código numérico de la tecla pulsada
offsetX	Número entero	Coordenada X de la posición del ratón respecto del elemento que origina el evento
offsetY	Número entero	Coordenada Y de la posición del ratón respecto del elemento que origina el evento
repeat	Boolean	Devuelve `true` si se está produciendo el evento `keydown` de forma continuada y `false` en otro caso
returnValue	Boolean	Se emplea para cancelar la acción predefinida del evento
screenX	Número entero	Coordenada X de la posición del ratón respecto de la pantalla completa
screenY	Número entero	Coordenada Y de la posición del ratón respecto de la pantalla completa
shiftKey	Boolean	Devuelve `true` si se ha pulsado la tecla SHIFT y `false` en otro caso
srcElement	Element	El elemento que origina el evento
toElement	Element	El elemento al que entra el ratón (para ciertos eventos)
type	Cadena texto	El nombre del evento
x	Número entero	Coordenada X de la posición del ratón respecto del elemento padre del elemento que origina el evento
y	Número entero	Coordenada Y de la posición del ratón respecto del elemento padre del elemento que origina el evento

Todas las propiedades salvo `repeat` son de lectura/escritura y por tanto, su valor se puede leer o establecer.

1.2. Propiedades definidas por la especificación DOM

La siguiente tabla recoge las propiedades definidas para el objeto `event` en los navegadores que siguen los estándares:

Propiedad/Método	Devuelve	Descripción
altKey	Boolean	Devuelve `true` si se ha pulsado la tecla ALT y `false` en otro caso
bubbles	Boolean	Indica si el evento pertenece al flujo de eventos de *bubbling*
button	Número entero	El botón del ratón que ha sido pulsado. Posibles valores: 0 – Ningún botón pulsado 1 – Se ha pulsado el botón izquierdo 2 – Se ha pulsado el botón derecho 3 – Se pulsan a la vez el botón izquierdo y el derecho 4 - Se ha pulsado el botón central 5 – Se pulsan a la vez el botón izquierdo y el central 6 - Se pulsan a la vez el botón derecho y el central 7 – Se pulsan a la vez los 3 botones
cancelable	Boolean	Indica si el evento se puede cancelar
cancelBubble	Boolean	Indica si se ha detenido el flujo de eventos de tipo *bubbling*
charCode	Número entero	El código unicode del carácter correspondiente a la tecla pulsada
clientX	Número entero	Coordenada X de la posición del ratón respecto del área visible de la ventana
clientY	Número entero	Coordenada Y de la posición del ratón respecto del área visible de la ventana
ctrlKey	Boolean	Devuelve `true` si se ha pulsado la tecla CTRL y `false` en otro caso
currentTarget	Element	El elemento que es objeto del evento
detail	Número entero	El número de veces que se han pulsado los botones del ratón
eventPhase	Número entero	La fase a la que pertenece el evento: 0 – Fase capturing 1 – En el elemento destino 2 – Fase bubbling
isChar	Boolean	Indica si la tecla pulsada corresponde a un carácter
keyCode	Número entero	Indica el código numérico de la tecla pulsada
metaKey	Número entero	Devuelve `true` si se ha pulsado la tecla META y `false` en otro caso
pageX	Número entero	Coordenada X de la posición del ratón respecto de la página
pageY	Número entero	Coordenada Y de la posición del ratón respecto de la página
preventDefault()	Función	Se emplea para cancelar la acción predefinida del evento
relatedTarget	Element	El elemento que es el objetivo secundario del evento (relacionado con los eventos de ratón)
screenX	Número entero	Coordenada X de la posición del ratón respecto de la pantalla completa
screenY	Número entero	Coordenada Y de la posición del ratón respecto de la pantalla completa

shiftKey	Boolean	Devuelve true si se ha pulsado la tecla SHIFT y false en otro caso
stopPropagation()	Función	Se emplea para detener el flujo de eventos de tipo *bubbling*
target	Element	El elemento que origina el evento
timeStamp	Número	La fecha y hora en la que se ha producido el evento
type	Cadena de texto	El nombre del evento

Al contrario de lo que sucede con Internet Explorer, la mayoría de propiedades del objeto event de DOM son de sólo lectura. En concreto, solamente las siguientes propiedades son de lectura y escritura: altKey, button y keyCode.

La tecla META es una tecla especial que se encuentra en algunos teclados de ordenadores muy antiguos. Actualmente, en los ordenadores tipo PC se asimila a la tecla Alt o a la *tecla de Windows*, mientras que en los ordenadores tipo Mac se asimila a la tecla Command.

2. Similitudes y diferencias entre navegadores

2.1. Similitudes

En ambos casos se utiliza la propiedad type para obtener el tipo de evento que se trata:

```
function procesaEvento(elEvento) {
  if(elEvento.type == "click") {
    alert("Has pulsado el ratón");
  }
  else if(elEvento.type == "mouseover") {
    alert("Has movido el raton");
  }
}

elDiv.onclick = procesaEvento;
elDiv.onmouseover = procesaEvento;
```

Mientras que el manejador del evento incluye el prefijo on en su nombre, el tipo de evento devuelto por la propiedad type prescinde de ese prefijo. Por eso en el ejemplo anterior se compara su valor con click y mouseover y no con onclick y onmouseover.

Otra similitud es el uso de la propiedad keyCode para obtener el código correspondiente al carácter de la tecla que se ha pulsado. La tecla pulsada no siempre representa un carácter alfanumérico. Cuando se pulsa la tecla ENTER por ejemplo, se obtiene el código 13. La barra espaciadora se corresponde con el código 32 y la tecla de borrado tiene un código igual a 8.

Una forma más inmediata de comprobar si se han pulsado algunas teclas

especiales, es utilizar las propiedades `shiftKey`, `altKey` y `ctrlKey`.

Para obtener la posición del ratón respecto de la parte visible de la ventana, se emplean las propiedades `clientX` y `clientY`. De la misma forma, para obtener la posición del puntero del ratón respecto de la pantalla completa, se emplean las propiedades `screenX` y `screenY`.

2.2. Diferencias

Una de las principales diferencias es la forma en la que se obtiene el elemento que origina el evento. Si un elemento `<div>` tiene asignado un evento `onclick`, al pulsar con el ratón el interior del `<div>` se origina un evento cuyo objetivo es el elemento `<div>`.

```
// Internet Explorer
var objetivo = elEvento.srcElement;

// Navegadores que siguen los estándares
var objetivo = elEvento.target;
```

Otra diferencia importante es la relativa a la obtención del carácter correspondiente a la tecla pulsada, ya que tienen asociados dos códigos diferentes: el primero es el código de la tecla que ha sido presionada y el otro código es el que se refiere al carácter pulsado.

El primer código es un código de tecla interno para JavaScript. El segundo código coincide con el código ASCII del carácter. De esta forma, la letra a tiene un código interno igual a 65 y un código ASCII de 97. Por otro lado, la letra A tiene un código interno también de 65 y un código ASCII de 95.

En Internet Explorer, el contenido de la propiedad `keyCode` depende de cada evento. En los eventos de "pulsación de teclas" (`onkeyup` y `onkeydown`) su valor es igual al código interno. En los eventos de "escribir con teclas" (`onkeypress`) su valor es igual al código ASCII del carácter pulsado.

Por el contrario, en los navegadores que siguen los estándares la propiedad `keyCode` es igual al código interno en los eventos de "pulsación de teclas" (`onkeyup` y `onkeydown`) y es igual a 0 en los eventos de "escribir con teclas" (`onkeypress`).

En la práctica, esto supone que en los eventos `onkeyup` y `onkeydown` se puede utilizar la misma propiedad en todos los navegadores:

```
function manejador(elEvento) {
  var evento = elEvento || window.event;
  alert("["+evento.type+"] El código de la tecla pulsada
es " +                              evento.keyCode);
}
```

```
document.onkeyup = manejador;
document.onkeydown = manejador;
```

En este caso, si se carga la página en cualquier navegador y se pulsa por ejemplo la tecla a, se muestra el siguiente mensaje:

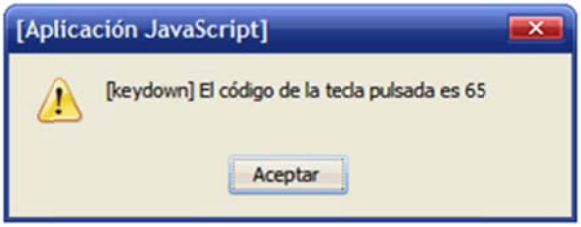

Figura 7.4. Mensaje mostrado en el navegador Firefox

Figura 7.5. Mensaje mostrado en el navegador Internet Explorer

La gran diferencia se produce al intentar obtener el carácter que se ha pulsado, en este caso la letra a. Para obtener la letra, en primer lugar se debe obtener su código ASCII. Como se ha comentado, en Internet Explorer el valor de la propiedad `keyCode` en el evento `onkeypress` es igual al carácter ASCII:

```
function manejador() {
  var evento = window.event;

  // Internet Explorer
  var codigo = evento.keyCode;
}

document.onkeypress = manejador;
```

Sin embargo, en los navegadores que no son Internet Explorer, el código anterior es igual a 0 para cualquier tecla pulsada. En estos navegadores que siguen los estándares, se debe utilizar la propiedad `charCode`, que devuelve el código de la tecla pulsada, pero solo para el evento `onkeypress`:

```
function manejador(elEvento) {
  var evento = elEvento;

  // Navegadores que siguen los estándares
  var codigo = evento.charCode;
}

document.onkeypress = manejador;
```

Una vez obtenido el código en cada navegador, se debe utilizar la función `String.fromCharCode()` para obtener el carácter cuyo código ASCII se pasa como parámetro. Por tanto, la solución completa para obtener la tecla pulsada en cualquier navegador es la siguiente:

```
function manejador(elEvento) {
  var evento = elEvento || window.event;
  var codigo = evento.charCode || evento.keyCode;
  var caracter = String.fromCharCode(codigo);
}

document.onkeypress = manejador;
```

Una de las propiedades más interesantes es la posibilidad de impedir que se complete el comportamiento normal de un evento. En otras palabras, con JavaScript es posible no mostrar ningún carácter cuando se pulsa una tecla, no enviar un formulario después de pulsar el botón de envío, no cargar ninguna página al pulsar un enlace, etc. El método avanzado de impedir que un evento ejecute su acción asociada depende de cada navegador:

```
// Internet Explorer
elEvento.returnValue = false;

// Navegadores que siguen los estándares
elEvento.preventDefault();
```

En el modelo básico de eventos también es posible impedir el comportamiento por defecto de algunos eventos. Si por ejemplo en un elemento `<textarea>` se indica el siguiente manejador de eventos:

```
<textarea onkeypress="return false;"></textarea>
```

En el `<textarea>` anterior no será posible escribir ningún carácter, ya que el manejador de eventos devuelve `false` y ese es el valor necesario para impedir que se termine de ejecutar el evento y por tanto para evitar que la letra se escriba.

Así, es posible definir manejadores de eventos que devuelvan `true` o `false` en función de algunos parámetros. Por ejemplo se puede diseñar un limitador del número de caracteres que se pueden escribir en un `<textarea>`:

```
function limita(maximoCaracteres) {
  var elemento = document.getElementById("texto");
  if(elemento.value.length >= maximoCaracteres ) {
    return false;
  }
  else {
```

```
    return true;
  }
}
<textarea id="texto" onkeypress="return limita(100);">
</textarea>
```

El funcionamiento del ejemplo anterior se detalla a continuación:

1. Se utiliza el evento `onkeypress` para controlar si la tecla se escribe o no.
2. En el manejador del evento se devuelve el valor devuelto por la función externa `limita()` a la que se pasa como parámetro el valor 100.
3. Si el valor devuelto por `limita()` es `true`, el evento se produce de forma normal y el carácter se escribe en el `<textarea>`. Si el valor devuelto por `limita()` es false, el evento no se produce y por tanto el carácter no se escribe en el `<textarea>`.
4. La función `limita()` devuelve `true` o `false` después de comprobar si el número de caracteres del `<textarea>` es superior o inferior al máximo número de caracteres que se le ha pasado como parámetro.

El objeto `event` también permite detener completamente la ejecución del flujo normal de eventos:

```
// Internet Explorer
elEvento.cancelBubble = true;

// Navegadores que siguen los estándares
elEvento.stopPropagation();
```

Al detener el flujo de eventos pendientes, se invalidan y no se ejecutan los eventos que restan desde ese momento hasta que se recorren todos los elementos pendientes hasta el elemento `window`.