Curso de Javascript

Unidad Didáctica 15: Eventos

http://cursosdedesarrollo.com/

Ayuntamiento
de Vitoria-Gasteiz
Vitoria-Gasteizko
Udala

Índice de contenidos

http://cursosdedesarrollo.com/

Introducción

En la programación tradicional, las aplicaciones se ejecutan secuencialmente de principio a fin para producir sus resultados

Sin embargo, en la actualidad el modelo predominante es el de la programación basada en eventos

http://cursosdedesarrollo.com/

Introducción

Los scripts y programas esperan sin realizar ninguna tarea hasta que se produzca un evento

Una vez producido, ejecutan alguna tarea asociada a la aparición de ese evento y cuando concluye, el script o programa vuelve al estado de espera

http://cursosdedesarrollo.com/

Introducción

JavaScript permite realizar scripts con ambos métodos de programación: secuencial y basada en eventos

Los eventos de JavaScript permiten la interacción entre las aplicaciones JavaScript y los usuarios

http://cursosdedesarrollo.com/

Introducción

Cada vez que se pulsa un botón, se produce un evento

Cada vez que se pulsa una tecla, también se produce un evento

http://cursosdedesarrollo.com/

Introducción

No obstante, para que se produzca un evento no es obligatorio que intervenga el usuario, ya que por ejemplo, cada vez que se carga una página, también se produce un evento

Introducción

El nivel 1 de DOM no incluye especificaciones relativas a los eventos JavaScript

El nivel 2 de DOM incluye ciertos aspectos relacionados con los eventos y el nivel 3 de DOM incluye la especificación completa de los eventos de JavaScript

http://cursosdedesarrollo.com/

Introducción

Desafortunadamente, la especificación de nivel 3 de DOM se publicó en el año 2004, más de diez años después de que los primeros navegadores incluyeran los eventos

Introducción

Por este motivo, muchas de las propiedades y métodos actuales relacionados con los eventos son incompatibles con los de DOM

De hecho, navegadores como Internet Explorer tratan los eventos siguiendo su propio modelo incompatible con el estándar

Introducción

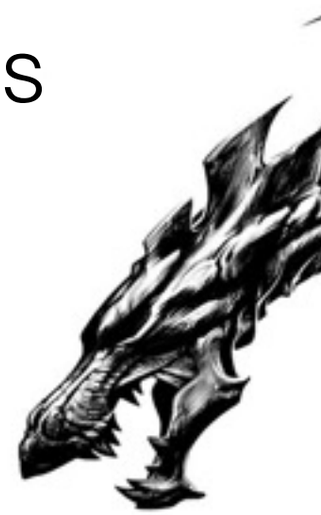
El modelo simple de eventos se introdujo en la versión 4 del estándar HTML y se considera parte del nivel más básico de DOM

Aunque sus características son limitadas, es el único modelo que es compatible con todos los navegadores y por tanto, el único que permite crear aplicaciones que funcionan de la misma manera en todos los navegadores

Tipos de Eventos

Cada elemento XHTML tiene definida su propia lista de posibles eventos que se le pueden asignar

Un mismo tipo de evento (por ejemplo, pinchar el botón izquierdo del ratón) puede estar definido para varios elementos XHTML y un mismo elemento XHTML puede tener asociados diferentes eventos



Eventos Disponibles

El nombre de los eventos se construye mediante el prefijo on, seguido del nombre en inglés de la acción asociada al evento

Así, el evento de pinchar un elemento con el ratón se denomina onclick y el evento asociado a la acción de mover el ratón se denomina onmousemove

http://cursosdedesarrollo.com/

Eventos Disponibles

Evento	Descripción	Elementos para los que está definido
`onblur`	Un elemento pierde el foco	`<button>`, `<input>`, `<label>`, `<select>`, `<textarea>`, `<body>`
`onchange`	Un elemento ha sido modificado	`<input>`, `<select>`, `<textarea>`
`onclick`	Pulsar y soltar el ratón	Todos los elementos
`ondblclick`	Pulsar dos veces seguidas con el ratón	Todos los elementos
`onfocus`	Un elemento obtiene el foco	`<button>`, `<input>`, `<label>`, `<select>`, `<textarea>`, `<body>`
`onkeydown`	Pulsar una tecla y no soltarla	Elementos de formulario y `<body>`

http://cursosdedesarrollo.com/

Eventos Disponibles

`onkeypress`	Pulsar una tecla	Elementos de formulario y `<body>`
`onkeyup`	Soltar una tecla pulsada	Elementos de formulario y `<body>`
`onload`	Página cargada completamente	`<body>`
`onmousedown`	Pulsar un botón del ratón y no soltarlo	Todos los elementos
`onmousemove`	Mover el ratón	Todos los elementos
`onmouseout`	El ratón "sale" del elemento	Todos los elementos
`onmouseover`	El ratón "entra" en el elemento	Todos los elementos
`onmouseup`	Soltar el botón del ratón	Todos los elementos

http://cursosdedesarrollo.com/

Eventos Disponibles

`onreset`	Inicializar el formulario	`<form>`
`onresize`	Modificar el tamaño de la ventana	`<body>`
`onselect`	Seleccionar un texto	`<input>`, `<textarea>`
`onsubmit`	Enviar el formulario	`<form>`
`onunload`	Se abandona la página, por ejemplo al cerrar el navegador	`<body>`

http://cursosdedesarrollo.com/

Eventos Disponibles

Los eventos más utilizados en las aplicaciones web tradicionales son onload para esperar a que se cargue la página por completo, los eventos onclick, onmouseover, onmouseout para controlar el ratón y onsubmit para controlar el envío de los formularios

Eventos Disponibles

Algunos eventos de la tabla anterior (onclick, onkeydown, onkeypress, onreset, onsubmit) permiten evitar la "acción por defecto" de ese evento. Más adelante se muestra en detalle este comportamiento, que puede resultar muy útil en algunas técnicas de programación

Eventos Disponibles

Las acciones típicas que realiza un usuario en una página web pueden dar lugar a una sucesión de eventos. Al pulsar por ejemplo sobre un botón de tipo <input type="submit"> se desencadenan los eventos onmousedown, onclick, onmouseup y onsubmit de forma consecutiva

http://cursosdedesarrollo.com/

Eventos Disponibles

MANEJADORES DE EVENTOS

Un evento de JavaScript por sí mismo carece de utilidad. Para que los eventos resulten útiles, se deben asociar funciones o código JavaScript a cada evento

http://cursosdedesarrollo.com/

Eventos Disponibles

De esta forma, cuando se produce un evento se ejecuta el código indicado, por lo que la aplicación puede responder ante cualquier evento que se produzca durante su ejecución

http://cursosdedesarrollo.com/

Eventos Disponibles

Las funciones o código JavaScript que se definen para cada evento se denominan manejador de eventos (event handlers en inglés) y como JavaScript es un lenguaje muy flexible, existen varias formas diferentes de indicar los manejadores:

- Manejadores como atributos de los elementos XHTML.
- Manejadores como funciones JavaScript externas.
- Manejadores "semánticos".

http://cursosdedesarrollo.com/

Eventos Disponibles

MANEJADORES COMO ATRIBUTOS XHTML

Se trata del método más sencillo y a la vez menos profesional de indicar el código JavaScript que se debe ejecutar cuando se produzca un evento

http://cursosdedesarrollo.com/

Eventos Disponibles

En este caso, el código se incluye en un atributo del propio elemento XHTML. En el siguiente ejemplo, se quiere mostrar un mensaje cuando el usuario pinche con el ratón sobre un botón:

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

Eventos Disponibles

En este método, se definen atributos XHTML con el mismo nombre que los eventos que se quieren manejar. El ejemplo anterior sólo quiere controlar el evento de pinchar con el ratón, cuyo nombre es onclick. Así, el elemento XHTML para el que se quiere definir este evento, debe incluir un atributo llamado onclick

Eventos Disponibles

El contenido del atributo es una cadena de texto que contiene todas las instrucciones JavaScript que se ejecutan cuando se produce el evento. En este caso, el código JavaScript es muy sencillo (console.log('Gracias por pinchar');), ya que solamente se trata de mostrar un mensaje

http://cursosdedesarrollo.com/

Eventos Disponibles

En este otro ejemplo, cuando el usuario pincha sobre el elemento se muestra un mensaje y cuando el usuario pasa el ratón por encima del elemento, se muestra otro mensaje:

Puedes pinchar sobre este elemento o simplemente pasar el ratón por encima

Eventos Disponibles

Este otro ejemplo incluye una de las instrucciones más utilizadas en las aplicaciones JavaScript más antiguas:

```
<body onload="console.log('La página se ha cargado completamente');">

...

</body>
```

http://cursosdedesarrollo.com/

Eventos Disponibles

El mensaje anterior se muestra después de que la página se haya cargado completamente, es decir, después de que se haya descargado su código HTML, sus imágenes y cualquier otro objeto incluido en la página

Eventos Disponibles

El evento onload es uno de los más utilizados ya que, como se vio en el capítulo de DOM, las funciones que permiten acceder y manipular los nodos del árbol DOM solamente están disponibles cuando la página se ha cargado completamente

Eventos Disponibles

MANEJADORES DE EVENTOS Y VARIABLE THIS

JavaScript define una variable especial llamada this que se crea automáticamente y que se emplea en algunas técnicas avanzadas de programación

http://cursosdedesarrollo.com/

Eventos Disponibles

En los eventos, se puede utilizar la variable this para referirse al elemento XHTML que ha provocado el evento

Eventos Disponibles

Esta variable es muy útil para ejemplos como el siguiente:

Cuando el usuario pasa el ratón por encima del , el color del borde se muestra de color negro. Cuando el ratón sale del , se vuelve a mostrar el borde con el color gris claro original.

Sección de contenidos...

Eventos Disponibles

Si no se utiliza la variable this, el código necesario para modificar el color de los bordes, sería el siguiente:

```
<div id="contenidos" style="width:150px; height:60px; border:thin
solid silver"
onmouseover="document.getElementById('contenidos').style.border
Color='black';"
onmouseout="document.getElementById('contenidos').style.borderC
olor='silver';">

Sección de contenidos...

</div>
```


Eventos Disponibles

El código anterior es demasiado largo y demasiado propenso a cometer errores. Dentro del código de un evento, JavaScript crea automáticamente la variable this, que hace referencia al elemento XHTML que ha provocado el evento

Eventos Disponibles

Así, el ejemplo anterior se puede reescribir de la siguiente manera:

Sección de contenidos...

Eventos Disponibles

El código anterior es mucho más compacto, más fácil de leer y de escribir y sigue funcionando correctamente aunque se modifique el valor del atributo id del

Eventos Disponibles

MANEJADORES DE EVENTOS COMO FUNCIONES EXTERNAS

La definición de manejadores de eventos en los atributos XHTML es un método sencillo pero poco aconsejable para tratar con los eventos en JavaScript

http://cursosdedesarrollo.com/

Eventos Disponibles

El principal inconveniente es que se complica en exceso en cuanto se añaden algunas pocas instrucciones, por lo que solamente es recomendable para los casos más sencillos

http://cursosdedesarrollo.com/

Eventos Disponibles

Cuando el código de la función manejadora es más complejo, como por ejemplo la validación de un formulario, es aconsejable agrupar todo el código JavaScript en una función externa que se invoca desde el código XHTML cuando se produce el evento

Eventos Disponibles

De esta forma, el siguiente ejemplo:

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

Se puede transformar en:

```
function muestraMensaje() {

console.log('Gracias por pinchar');

}

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

http://cursosdedesarrollo.com/

Eventos Disponibles

En las funciones externas no es posible utilizar la variable this de la misma forma que en los manejadores insertados en los atributos XHTML. Por tanto, es necesario pasar la variable this como parámetro a la función manejadora

Eventos Disponibles

```
function resalta(elemento) {
  switch(elemento.style.borderColor) {
    case 'silver':
    case 'silver silver silver silver':
    case '#c0c0c0':
      elemento.style.borderColor = 'black';
      break;
    case 'black':
    case 'black black black black':
    case '#000000':
      elemento.style.borderColor = 'silver';
      break;
  }
}
```

http://cursosdedesarrollo.com/

Eventos Disponibles

<div style="padding: .2em; width: 150px; height: 60px; border: thin solid silver"
onmouseover="resalta(this)"
onmouseout="resalta(this)">

Sección de contenidos...

http://cursosdedesarrollo.com/

Eventos Disponibles

En el ejemplo anterior, a la función externa se le pasa el parámetro this, que dentro de la función se denomina elemento

Al pasar this como parámetro, es posible acceder de forma directa desde la función externa a las propiedades del elemento que ha provocado el evento

Eventos Disponibles

Por otra parte, el ejemplo anterior se complica por la forma en la que los distintos navegadores almacenan el valor de la propiedad borderColor

Mientras que Firefox almacena (en caso de que los cuatro bordes coincidan en color) el valor simple black, Internet Explorer lo almacena como black black black black y Opera almacena su representación hexadecimal #000000

Eventos Disponibles

MANEJADORES DE EVENTOS SEMÁNTICOS

Utilizar los atributos XHTML o las funciones externas para añadir manejadores de eventos tiene un grave inconveniente: "ensucian" el código XHTML de la página

Eventos Disponibles

Como es conocido, al crear páginas web se recomienda separar los contenidos (XHTML) de la presentación (CSS). En lo posible, también se recomienda separar los contenidos (XHTML) de la programación (JavaScript)

http://cursosdedesarrollo.com/

Eventos Disponibles

Mezclar JavaScript y XHTML complica excesivamente el código fuente de la página, dificulta su mantenimiento y reduce la semántica del documento final producido

http://cursosdedesarrollo.com/

Eventos Disponibles

Afortunadamente, existe un método alternativo para definir los manejadores de eventos de JavaScript. Esta técnica consiste en asignar las funciones externas mediante las propiedades DOM de los elementos XHTML

Eventos Disponibles

Así, el siguiente ejemplo:

<input id="pinchable" type="button" value="Pinchame y verás"
onclick="console.log('Gracias por pinchar');" />

Se puede transformar en:

function muestraMensaje() {

console.log('Gracias por pinchar');

}

document.getElementById("pinchable").onclick = muestraMensaje;

<input id="pinchable" type="button" value="Pinchame y verás" />

http://cursosdedesarrollo.com/

Eventos Disponibles

El código XHTML resultante es muy "limpio", ya que no se mezcla con el código JavaScript. La técnica de los manejadores semánticos consiste en:

1. Asignar un identificador único al elemento XHTML mediante el atributo id.

2. Crear una función de JavaScript encargada de manejar el evento.

3. Asignar la función a un evento concreto del elemento XHTML mediante DOM.

http://cursosdedesarrollo.com/

Eventos Disponibles

Otra ventaja adicional de esta técnica es que las funciones externas pueden utilizar la variable this referida al elemento que original el evento

Asignar la función manejadora mediante DOM es un proceso que requiere una explicación detallada

Eventos Disponibles

En primer lugar, se obtiene la referencia del elemento al que se va a asignar el manejador:

document.getElementById("pinchable");

A continuación, se asigna la función externa al evento deseado mediante una propiedad del elemento con el mismo nombre del evento:

document.getElementById("pinchable").onclick = ...

http://cursosdedesarrollo.com/

Eventos Disponibles

Por último, se asigna la función externa. Como ya se ha comentado en capítulos anteriores, lo más importante (y la causa más común de errores) es indicar solamente el nombre de la función, es decir, prescindir de los paréntesis al asignar la función:

```
document.getElementById("pinchable").onclick =
muestraMensaje;
```


Eventos Disponibles

Si se añaden los paréntesis al final, en realidad se está invocando la función y asignando el valor devuelto por la función al evento onclick de elemento

http://cursosdedesarrollo.com/

Eventos Disponibles

El único inconveniente de este método es que los manejadores se asignan mediante las funciones DOM, que solamente se pueden utilizar después de que la página se ha cargado completamente

De esta forma, para que la asignación de los manejadores no resulte errónea, es necesario asegurarse de que la página ya se ha cargado

http://cursosdedesarrollo.com/

Eventos Disponibles

Una de las formas más sencillas de asegurar que cierto código se va a ejecutar después de que la página se cargue por completo es utilizar el evento onload:

```
window.onload = function() {

document.getElementById("pinchable").onclick =
muestraMensaje;

}
```

http://cursosdedesarrollo.com/

Eventos Disponibles

La técnica anterior utiliza una función anónima para asignar algunas instrucciones al evento onload de la página (en este caso se ha establecido mediante el objeto window)

http://cursosdedesarrollo.com/

Eventos Disponibles

De esta forma, para asegurar que cierto código se va a ejecutar después de que la página se haya cargado, sólo es necesario incluirlo en el interior de la siguiente construcción:

```
window.onload = function() {

...

}
```

http://cursosdedesarrollo.com/

El Flujo de Eventos

Además de los eventos básicos que se han visto, los navegadores incluyen un mecanismo relacionado llamado flujo de eventos o "event flow”

El flujo de eventos permite que varios elementos diferentes puedan responder a un mismo evento

El Flujo de Eventos

Si en una página HTML se define un elemento con un botón en su interior, cuando el usuario pulsa sobre el botón, el navegador permite asignar una función de respuesta al botón, otra función de respuesta al que lo contiene y otra función de respuesta a la página completa

http://cursosdedesarrollo.com/

El Flujo de Eventos

De esta forma, un solo evento (la pulsación de un botón) provoca la respuesta de tres elementos de la página (incluyendo la propia página)

http://cursosdedesarrollo.com/

El Flujo de Eventos

El orden en el que se ejecutan los eventos asignados a cada elemento de la página es lo que constituye el flujo de eventos. Además, existen muchas diferencias en el flujo de eventos de cada navegador

http://cursosdedesarrollo.com/

El Flujo de Eventos

EVENT BUBBLING

En este modelo de flujo de eventos, el orden que se sigue es desde el elemento más específico hasta el elemento menos específico

http://cursosdedesarrollo.com/

El Flujo de Eventos

En los próximos ejemplos se emplea la siguiente página HTML:

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

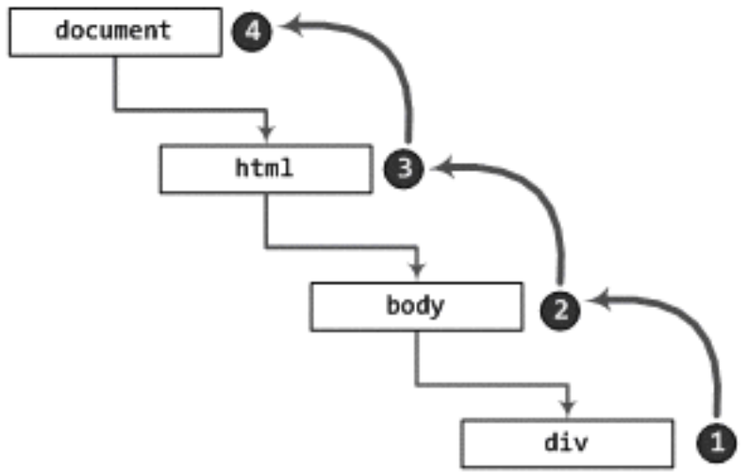
http://cursosdedesarrollo.com/

El Flujo de Eventos

Cuando se pulsa sobre el texto "Pincha aquí" que se encuentra dentro del , se ejecutan los siguientes eventos en el orden que muestra el siguiente esquema:

http://cursosdedesarrollo.com/

El Flujo de Eventos



http://cursosdedesarrollo.com/

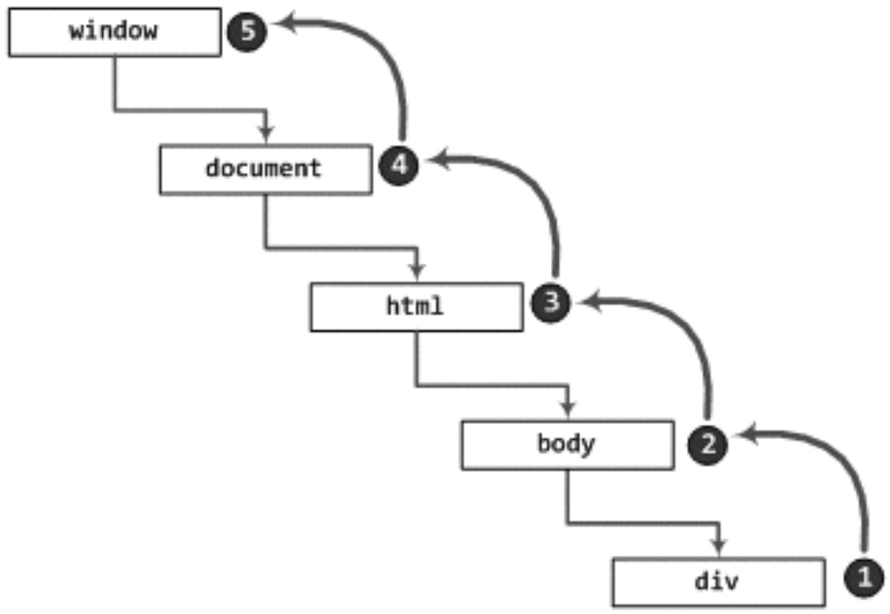
El Flujo de Eventos

El primer evento que se tiene en cuenta es el generado por el que contiene el mensaje. A continuación el navegador recorre los ascendentes del elemento hasta que alcanza el nivel superior, que es el elemento document

El Flujo de Eventos

Este modelo de flujo de eventos es el que incluye el navegador Internet Explorer. Los navegadores de la familia Mozilla (por ejemplo Firefox) también soportan este modelo, pero ligeramente modificado. El anterior ejemplo en un navegador de la familia Mozilla presenta el siguiente flujo de eventos

El Flujo de Eventos



http://cursosdedesarrollo.com/

El Flujo de Eventos

Aunque el objeto window no es parte del DOM, el flujo de eventos implementado por Mozilla recorre los ascendentes del elemento hasta el mismo objeto window, añadiendo por tanto un evento más al modelo de Internet Explorer

El Flujo de Eventos

EVENT CAPTURING

En ese otro modelo, el flujo de eventos se define desde el elemento menos específico hasta el elemento más específico. En otras palabras, el mecanismo definido es justamente el contrario al "event bubbling". Este modelo lo utilizaba el desaparecido navegador Netscape Navigator 4.0

http://cursosdedesarrollo.com/

El Flujo de Eventos

EVENTOS DOM

El flujo de eventos definido en la especificación DOM soporta tanto el bubbling como el capturing, pero el "event capturing" se ejecuta en primer lugar

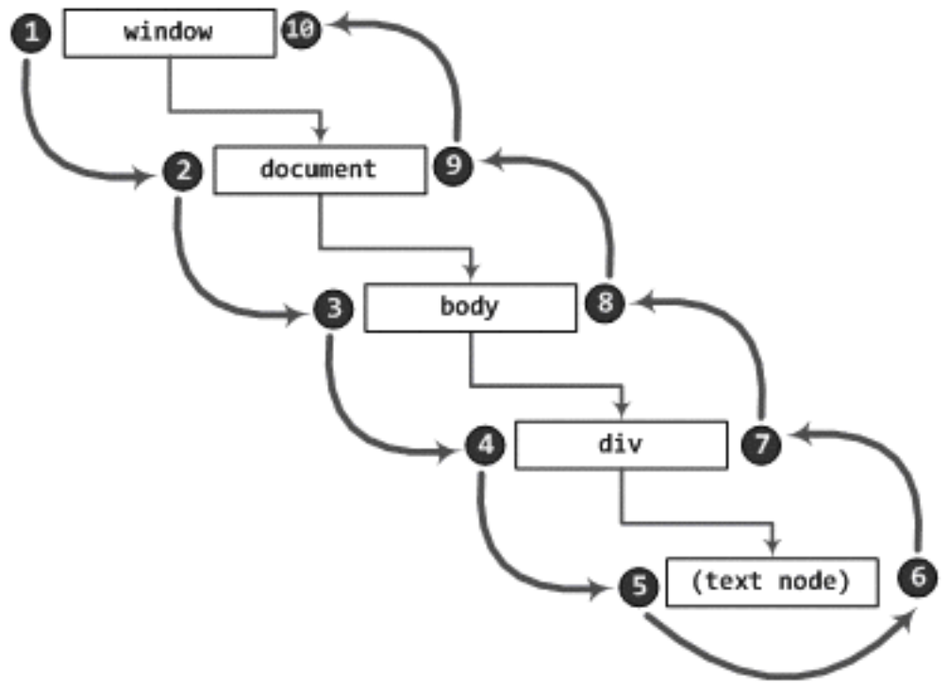
http://cursosdedesarrollo.com/

El Flujo de Eventos

Los dos flujos de eventos recorren todos los objetos DOM desde el objeto document hasta el elemento más específico y viceversa. Además, la mayoría de navegadores que implementan los estándares, continúan el flujo hasta el objeto window

http://cursosdedesarrollo.com/

El Flujo de Eventos



http://cursosdedesarrollo.com/

El Flujo de Eventos

El elemento más específico del flujo de eventos no es el que desencadena la ejecución de los eventos, sino el nodo de tipo TextNode que contiene el . El hecho de combinar los dos flujos de eventos, provoca que el nodo más específico pueda ejecutar dos eventos de forma consecutiva

Handlers y Listeners

En las secciones anteriores se introdujo el concepto de "event handler" o manejador de eventos, que son las funciones que responden a los eventos que se producen

http://cursosdedesarrollo.com/

Handlers y Listeners

Además, se vieron tres formas de definir los manejadores de eventos para el modelo básico de eventos:

1. Código JavaScript dentro de un atributo del propio elemento HTML

2. Definición del evento en el propio elemento HTML pero el manejador es una función externa

3. Manejadores semánticos asignados mediante DOM sin necesidad de modificar el código HTML de la página

http://cursosdedesarrollo.com/

Handlers y Listeners

Cualquiera de estos tres modelos funciona correctamente en todos los navegadores disponibles en la actualidad. Las diferencias entre navegadores surgen cuando se define más de un manejador de eventos para un mismo evento de un elemento

http://cursosdedesarrollo.com/

Handlers y Listeners

La forma de asignar y "desasignar" manejadores múltiples depende completamente del navegador utilizado

http://cursosdedesarrollo.com/

Handlers y Listeners

MANEJADORES DE EVENTOS DE DOM

La especificación DOM define otros dos métodos similares a los disponibles para Internet Explorer y denominados addEventListener() y removeEventListener() para asociar y desasociar manejadores de eventos

http://cursosdedesarrollo.com/

Handlers y Listeners

La principal diferencia entre estos métodos y los anteriores es que en este caso se requieren tres parámetros: el nombre del "event listener", una referencia a la función encargada de procesar el evento y el tipo de flujo de eventos al que se aplica

Handlers y Listeners

El primer argumento no es el nombre completo del evento como sucede en el modelo de Internet Explorer, sino que se debe eliminar el prefijo on. En otras palabras, si en Internet Explorer se utilizaba el nombre onclick, ahora se debe utilizar click

Handlers y Listeners

Si el tercer parámetro es true, el manejador se emplea en la fase de capture. Si el tercer parámetro es false, el manejador se asocia a la fase de bubbling

http://cursosdedesarrollo.com/

Handlers y Listeners

A continuación, se muestran los ejemplos anteriores empleando los métodos definidos por DOM:

```
function muestraMensaje() {

console.log("Has pulsado el ratón");
}

var elDiv = document.getElementById("div_principal");

elDiv.addEventListener("click", muestraMensaje, false);

// Más adelante se decide desasociar la función al evento

elDiv.removeEventListener("click", muestraMensaje, false);
```


Handlers y Listeners

Asociando múltiples funciones a un único evento:

```
function muestraMensaje() {

console.log("Has pulsado el ratón");

}

function muestraOtroMensaje() {

console.log("Has pulsado el ratón y por eso se muestran estos mensajes");

}

var elDiv = document.getElementById("div_principal");

elDiv.addEventListener("click", muestraMensaje, true);

elDiv.addEventListener("click", muestraOtroMensaje, true);
```

http://cursosdedesarrollo.com/

Handlers y Listeners

Si se asocia una función a un flujo de eventos determinado, esa función sólo se puede desasociar en el mismo tipo de flujo de eventos. Si se considera el siguiente ejemplo:

```
function muestraMensaje() {
    console.log("Has pulsado el ratón");
}
var elDiv = document.getElementById("div_principal");
elDiv.addEventListener("click", muestraMensaje, false);
// Más adelante se decide desasociar la función al evento
elDiv.removeEventListener("click", muestraMensaje, true);
```

http://cursosdedesarrollo.com/

Handlers y Listeners

La última instrucción intenta desasociar la función muestraMensaje en el flujo de eventos de capture, mientras que al asociarla, se indicó el flujo de eventos de bubbling

Aunque la ejecución de la aplicación no se detiene y no se produce ningún error, la última instrucción no tiene ningún efecto

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

http://cursosdedesarrollo.com/

El objeto Event

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

http://cursosdedesarrollo.com/

El objeto 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];

}
```

http://cursosdedesarrollo.com/

El objeto Event

También es posible indicar el nombre argumento de forma explícita:

```
elDiv.onclick = function(event) {

    ...

}
```

http://cursosdedesarrollo.com/

El objeto Event

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

http://cursosdedesarrollo.com/

El objeto Event

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

http://cursosdedesarrollo.com/

El objeto Event

PROPIEDADES DEFINIDAS POR DOM

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

http://cursosdedesarrollo.com/

El objeto Event

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 `izquierdo2- Se ha pulsado el botón derecho3- Se pulsan a la vez el botón izquierdo y el derecho4- Se ha pulsado el botón central5- Se pulsan a la vez el botón izquierdo y el central6- Se pulsan a la vez el botón derecho y el central7` – Se pulsan a la vez los 3 botones

http://cursosdedesarrollo.com/

El objeto Event

`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 el objetivo del evento

http://cursosdedesarrollo.com/

El objeto Event

`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

http://cursosdedesarrollo.com/

El objeto Event

`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

http://cursosdedesarrollo.com/

El objeto Event

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

El objeto Event

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

El objeto Event

SIMILITUDES Y DIFERENCIAS ENTRE NAVEGADORES

En este caso veremos las similitudes y diferencias entre los distintos navegadores principales

http://cursosdedesarrollo.com/

El objeto Event

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") {

console.log("Has pulsado el raton");

}

else if(elEvento.type == "mouseover") {

console.log("Has movido el raton");

}

}

elDiv.onclick = procesaEvento;

elDiv.onmouseover = procesaEvento;
```

http://cursosdedesarrollo.com/

El objeto Event

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

El objeto Event

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

http://cursosdedesarrollo.com/

El objeto Event

Una forma más inmediata de comprobar si se han pulsado algunas teclas especiales, es utilizar las propiedades shiftKey, altKey y ctrlKey

http://cursosdedesarrollo.com/

El objeto Event

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

El objeto Event

DIFERENCIAS

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

El objeto Event

// Internet Explorer

var objetivo = elEvento.srcElement;

// Navegadores que siguen los estandares

var objetivo = elEvento.target;

http://cursosdedesarrollo.com/

El objeto Event

Otra diferencia importante es la relativa a la obtención del carácter correspondiente a la tecla pulsada. Cada tecla pulsada tiene 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

http://cursosdedesarrollo.com/

El objeto Event

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

El objeto Event

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

http://cursosdedesarrollo.com/

El objeto Event

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)

El objeto Event

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

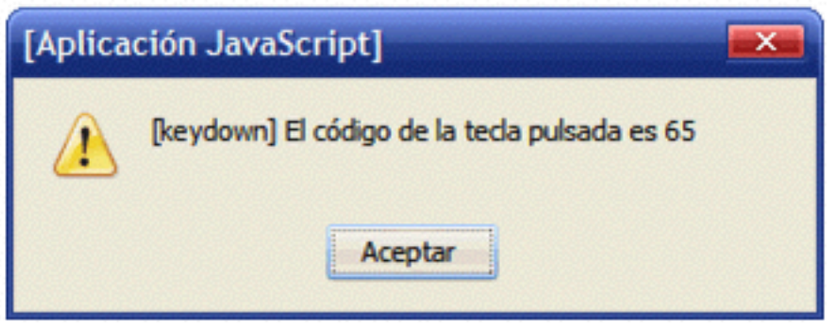
```
function manejador(elEvento) {
    var evento = elEvento || window.event;

    console.log("["+evento.type+"] El código de la tecla pulsada es " +
    evento.keyCode);

}

document.onkeyup = manejador;

document.onkeydown = manejador;
```

http://cursosdedesarrollo.com/

El objeto Event

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

El objeto Event



http://cursosdedesarrollo.com/

El objeto Event

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;
```

http://cursosdedesarrollo.com/

El objeto Event

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;
```

http://cursosdedesarrollo.com/

El objeto Event

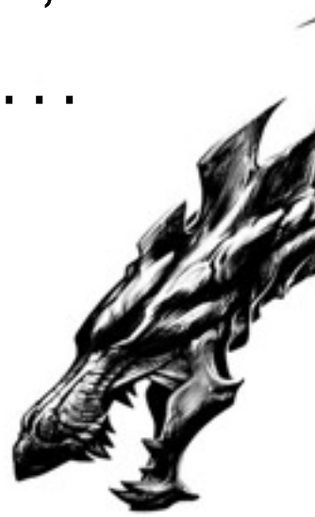
Una vez obtenido el código en cada navegador, se debe utilizar la función String.fromCharCode() para obtener el carácter cuyo código ASCII se pasa como parámetro. Por tanto, la solución completa para obtener la tecla pulsada en cualquier navegador es la siguiente:

```
function manejador(elEvento) {

    var evento = elEvento || window.event;

    var codigo = evento.charCode || evento.keyCode;

    var caracter = String.fromCharCode(codigo);

}

document.onkeypress = manejador;
```

http://cursosdedesarrollo.com/

El objeto Event

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 objeto Event

El método avanzado de impedir que un evento ejecute su acción asociada depende de cada navegador:

// Navegadores que siguen los estandares

elEvento.preventDefault();

El objeto Event

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>

El objeto Event

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

El objeto Event

Así, es posible definir manejadores de eventos que devuelvan true o false en función de algunos parámetros

http://cursosdedesarrollo.com/

El objeto Event

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>
```

http://cursosdedesarrollo.com/

El objeto Event

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

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

// Navegadores que siguen los estandares

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.

Tipos de Eventos

La lista completa de eventos que se pueden generar en un navegador se puede dividir en cuatro grandes grupos. La especificación de DOM define los siguientes grupos:

- Eventos de ratón: se originan cuando el usuario emplea el ratón para realizar algunas acciones.
- Eventos de teclado: se originan cuando el usuario pulsa sobre cualquier tecla de su teclado.
- Eventos HTML: se originan cuando se producen cambios en la ventana del navegador o cuando se producen ciertas interacciones entre el cliente y el servidor.
- Eventos DOM: se originan cuando se produce un cambio en la estructura DOM de la página. También se denominan "eventos de mutación".

http://cursosdedesarrollo.com/

Tipos de Eventos

Evento Descripción

- click Se produce cuando se pulsa el botón izquierdo del ratón. También se produce cuando el foco de la aplicación está situado en un botón y se pulsa la tecla ENTER
- dbIclickSe produce cuando se pulsa dos veces el botón izquierdo del ratón
- mousedown Se produce cuando se pulsa cualquier botón del ratón
- mouseout Se produce cuando el puntero del ratón se encuentra en el interior de un elemento y el usuario mueve el puntero a un lugar fuera de ese elemento
- mouseover Se produce cuando el puntero del ratón se encuentra fuera de un elemento y el usuario mueve el puntero hacia un lugar en el interior del elemento
- mouseup Se produce cuando se suelta cualquier botón del ratón que haya sido pulsado
- mousemove Se produce (de forma continua) cuando el puntero del ratón se encuentra sobre un elemento

http://cursosdedesarrollo.com/

Tipos de Eventos

Todos los elementos de las páginas soportan los eventos de la tabla anterior

http://cursosdedesarrollo.com/

Tipos de Eventos

PROPIEDADES

El objeto event contiene las siguientes propiedades para los eventos de ratón:

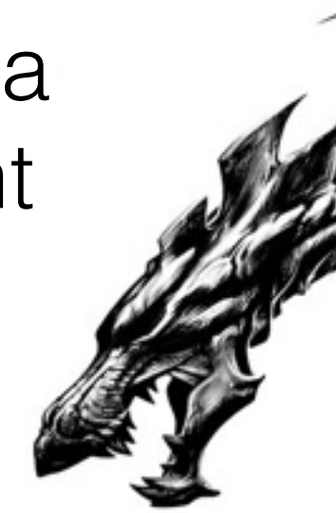
- Las coordenadas del ratón (todas las coordenadas diferentes relativas a los distintos elementos)
- La propiedad type
- La propiedad srcElement (Internet Explorer) o target (DOM)
- Las propiedades shiftKey, ctrlKey, altKey y metaKey (sólo DOM)
- La propiedad button (sólo en los eventos mousedown, mousemove, mouseout, mouseover y mouseup)

http://cursosdedesarrollo.com/

Tipos de Eventos

Los eventos mouseover y mouseout tienen propiedades adicionales. Internet Explorer define la propiedad fromElement, que hace referencia al elemento desde el que el puntero del ratón se ha movido y toElement que es el elemento al que el puntero del ratón se ha movido. De esta forma, en el evento mouseover, la propiedad toElement es idéntica a srcElement y en el evento mouseout, la propiedad fromElement es idéntica a srcElement

http://cursosdedesarrollo.com/



Tipos de Eventos

En los navegadores que soportan el estándar DOM, solamente existe una propiedad denominada relatedTarget. En el evento mouseout, relatedTarget apunta al elemento al que se ha movido el ratón. En el evento mouseover, relatedTarget apunta al elemento desde el que se ha movido el puntero del ratón

Tipos de Eventos

Cuando se pulsa un botón del ratón, la secuencia de eventos que se produce es la siguiente: mousedown, mouseup, click. Por tanto, la secuencia de eventos necesaria para llegar al doble click llega a ser tan compleja como la siguiente: mousedown, mouseup, click, mousedown, mouseup, click, dblclick

http://cursosdedesarrollo.com/

Tipos de Eventos

EVENTOS DE TECLADO

Los eventos que se incluyen en esta clasificación son los siguientes:

Evento Descripción

- keydown Se produce cuando se pulsa cualquier tecla del teclado. También se produce de forma continua si se mantiene pulsada la tecla

- keypress Se produce cuando se pulsa una tecla correspondiente a un carácter alfanumérico (no se tienen en cuenta telas como SHIFT, ALT, etc.). También se produce de forma continua si se mantiene pulsada la tecla

- keyup Se produce cuando se suelta cualquier tecla pulsada

http://cursosdedesarrollo.com/

Tipos de Eventos

El objeto event contiene las siguientes propiedades para los eventos de teclado:

- La propiedad keyCode
- La propiedad charCode (sólo DOM)
- La propiedad srcElement (Internet Explorer) o target (DOM)
- Las propiedades shiftKey, ctrlKey, altKey y metaKey (sólo DOM)

Tipos de Eventos

Cuando se pulsa una tecla correspondiente a un carácter alfanumérico, se produce la siguiente secuencia de eventos: keydown, keypress, keyup. Cuando se pulsa otro tipo de tecla, se produce la siguiente secuencia de eventos: keydown, keyup. Si se mantiene pulsada la tecla, en el primer caso se repiten de forma continua los eventos keydown y keypress y en el segundo caso, se repite el evento keydown de forma continua

http://cursosdedesarrollo.com/

Tipos de Eventos

EVENTOS HTML

Evento Descripción

- load Se produce en el objeto window cuando la página se carga por completo. En el elemento <img> cuando se carga por completo la imagen. En el elemento` cuando se carga el objeto

- unload Se produce en el objeto window cuando la página desaparece por completo (al cerrar la ventana del navegador por ejemplo). En el elemento <object> cuando desaparece el objeto.

- abort Se produce en un elemento <object> cuando el usuario detiene la descarga del elemento antes de que haya terminado

- error Se produce en el objeto window cuando se produce un error de JavaScript. En el elemento <img> cuando la imagen no se ha podido cargar por completo y en el elemento <object> cuando el elemento no se carga correctamente

- select Se produce cuando se seleccionan varios caracteres de un cuadro de texto (<input> y <textarea>)

- change Se produce cuando un cuadro de texto (<input> y <textarea>) pierde el foco y su contenido ha variado. También se produce cuando varía el valor de un elemento <select>

http://cursosdedesarrollo.com/

Tipos de Eventos

EVENTOS HTML

Evento Descripción

- submit Se produce cuando se pulsa sobre un botón de tipo submit (<input type="submit">)
- reset Se produce cuando se pulsa sobre un botón de tipo reset (<input type="reset">)
- resize Se produce en el objeto window cuando se redimensiona la ventana del navegador
- scroll Se produce en cualquier elemento que tenga una barra de scroll, cuando el usuario la utiliza. El elemento <body> contiene la barra de scroll de la página completa
- focus Se produce en cualquier elemento (incluido el objeto window) cuando el elemento obtiene el foco
- blur Se produce en cualquier elemento (incluido el objeto window) cuando el elemento pierde el foco

http://cursosdedesarrollo.com/

Tipos de Eventos

Uno de los eventos más utilizados es el evento load, ya que todas las manipulaciones que se realizan mediante DOM requieren que la página esté cargada por completo y por tanto, el árbol DOM se haya construido completamente

Tipos de Eventos

El elemento <body> define las propiedades scrollLeft y scrollTop que se pueden emplear junto con el evento scroll

http://cursosdedesarrollo.com/

Tipos de Eventos

EVENTOS DOM

Aunque los eventos de este tipo son parte de la especificación oficial de DOM, aún no han sido implementados en todos los navegadores. La siguiente tabla recoge los eventos más importantes de este tipo:

Evento Descripción

- DOMSubtreeModified Se produce cuando se añaden o eliminan nodos en el subárbol de un documento o elemento
- DOMNodeInserted Se produce cuando se añade un nodo como hijo de otro nodo
- DOMNodeRemoved Se produce cuando se elimina un nodo que es hijo de otro nodo
- DOMNodeRemovedFromDocument Se produce cuando se elimina un nodo del documento
- DOMNodeInsertedIntoDocument Se produce cuando se añade un nodo al documento

http://cursosdedesarrollo.com/

Conclusiones

Hemos visto cómo incluir y manejar los eventos disponibles desde Javascript

http://cursosdedesarrollo.com/

Datos de Contacto

http://www.cursosdedesarrollo.com
info@cursosdedesarrollo.com

http://cursosdedesarrollo.com/

Licencia

David Vaquero Santiago