EVENTOS

Las aplicaciones web creadas con el lenguaje JavaScript pueden utilizar el modelo de programación basada en eventos.

En este tipo de programación, los scripts se dedican a esperar a que el usuario "haga algo" (que pulse una tecla, que mueva el ratón, que cierre la ventana del navegador). A continuación, el script responde a la acción del usuario normalmente procesando esa información y generando un resultado.

Los eventos hacen posible que los usuarios transmitan información a los programas. JavaScript define numerosos eventos que permiten una interacción completa entre el usuario y las páginas/aplicaciones web. La pulsación de una tecla constituye un evento, así como pinchar o mover el ratón, seleccionar un elemento de un formulario, redimensionar la ventana del navegador, etc.

JavaScript permite asignar una función a cada uno de los eventos. De esta forma, cuando se produce cualquier evento, JavaScript ejecuta su función asociada. Este tipo de funciones se denominan "event handlers" en inglés y suelen traducirse por "manejadores de eventos".

1. *Tipos de* eventos

El nombre de cada evento 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.

La siguiente tabla resume los eventos más importantes definidos por JavaScript:

Evento	Descripción	Elementos para los que está definido
onblur	Deseleccionar el elemento	<button>, <input>, <label>, <select>, <textarea>, <body>
onchange	Deseleccionar un elemento que se ha modificado	<input>, <select>, <textarea>
onclick	Pinchar y soltar el ratón	Todos los elementos
ondblclick	Pinchar dos veces seguidas con el ratón	Todos los elementos
onfocus	Seleccionar un elemento	<button>, <input>, <label>, <select>, <textarea>, <body>
onkeydown	Pulsar una tecla (sin soltar)	Elementos de formulario y <body>
onkeypress	Pulsar una tecla	Elementos de formulario y <body>
onkeyup	Soltar una tecla pulsada	Elementos de formulario y <body>
onload	La página se ha cargado completamente	<body>
onmousedown	Pulsar (sin soltar) un botón del ratón	Todos los elementos
onmousemove	Mover el ratón	Todos los elementos
onmouseout	El ratón *"sale"* del elemento (pasa por encima de otro elemento)	Todos los elementos
onmouseover	El ratón *"entra"* en el elemento (pasa por encima del elemento)	Todos los elementos
onmouseup	Soltar el botón que estaba pulsado en el ratón	Todos los elementos
onreset	Inicializar el formulario (borrar todos sus datos)	<form>
onresize	Se ha modificado el tamaño de la ventana del navegador	<body>
onselect	Seleccionar un texto	<input>, <textarea>
onsubmit	Enviar el formulario	<form>
onunload	Se abandona la página (por ejemplo al cerrar el navegador)	<body>

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.

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. Si queremos validar los datos de un formulario antes de que sean enviados y no enviarlos si son incorrectos, escribiré lo siguiente en la etiqueta del formulario.

<form onsubmit=return validarDatos()>

Dónde validarDatos es una función que retorna true si los datos son correctos y false si no lo son. Si los datos son incorrectos la función devuelve false, y se anula el evento onsubmit.

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.**

onchange se produce cuando el usuario cambia el valor de un elemento de texto (<input type="text"> o <textarea>). También se produce cuando el usuario selecciona una opción en una lista desplegable (<select>). Sin embargo, el evento sólo se produce si después de realizar el cambio, el usuario *pasa* al siguiente campo del formulario, lo que técnicamente se conoce como que *"el campo de formulario ha perdido el foco"*.

2.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. De esta forma, cuando se produce un evento se ejecuta el código indicado.

Las funciones o código JavaScript que se definen para cada evento se denominan "manejador de eventos".

Hemos visto que dado un elemento HTML identificado por un id podemos acceder después desde javascript

var e = document.getElementById ("id");

Dependiendo del tipo de elemento podemos acceder y ver o cambiar determinadas propiedades

e.value	**(ej. para un <input text>)**
e.innerHTML	**(ej. para un , <p>,...)**
e.style.propiedad	**propiedades de estilo**

2.1. La variable this

Dentro del código de un evento, JavaScript crea automáticamente la variable this, que hace referencia al elemento que ha provocado el evento.

Ejemplo: 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.

```
<div id="contenidos" style="width:150px; height:60px; border:thin solid silver"
onMouseOver="this.style.borderColor='black';"
onMouseOut="this.style.borderColor='silver';">
Pasa el ratón por encima
</div>
```

2.2. *Manejadores de eventos como funciones externas*

La definición de los manejadores de eventos en los atributos es el método más sencillo pero menos aconsejable de tratar con los eventos en JavaScript. 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.

Si se realizan aplicaciones complejas, como por ejemplo la validación de un formulario, es aconsejable agrupar todo el código JavaScript en una función externa y llamar a esta función desde el elemento correspondiente.

Siguiendo con el ejemplo anterior que muestra un mensaje al pinchar sobre un botón:

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

Utilizando funciones externas se puede transformar en:

```
function muestraMensaje() {
  alert('Gracias por pinchar');
}
<input type="button" value="Pinchame y verás" onclick="muestraMensaje()" />
```

La llamada a la función se realiza de la forma habitual, indicando su nombre seguido de los paréntesis y de forma opcional, incluyendo todos los argumentos y parámetros que se necesiten El principal inconveniente de este método es que en las funciones externas no se puede seguir utilizando la variable this y por tanto, es necesario pasar esta variable como parámetro a la función:

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

```
    case 'black':
    elemento.style.borderColor = 'silver';
    break;
    }
}
<div style="width:150px; height:60px; border:thin solid silver"
onMouseOver="resalta(this)" onMouseOut="resalta(this)">
Sección de contenidos...
</div>
```

En el ejemplo anterior, la función externa es llamada con el parámetro this, que dentro de la función se denomina elemento.

3. Ejemplos

Cada vez que pulsemos un botón mostraremos un contador:

```
<script language="JavaScript">
contador=0;
function incrementar()
{
 contador++;
 alert('El contador ahora vale :' + contador);
}
</script>
<body>
<form>
 <input type="button" onClick="incrementar()" value="incrementar">
</form>
</body>
```

Hacer un formulario con tres botones y después mostrar el mensaje de cuál se ha pulsado

```
<html>
<head>
<title>Problema</title>
<script language="javascript">

function decirboton(k)
{
   alert('Has pulsado el boton'+k);
}
</script>

</head>
<body>
```

```
<form>
  <input type="button" onClick="decirboton(1)" value="1">
  <input type="button" onClick="decirboton(2)" value="2">
  <input type="button" onClick="decirboton(3)" value="3">
</form>
</body>
</html>
```

Uso particular de onclick

onclick puede recibir un valor true o false tras haber ejecutado una función y a su vez retornar este valor que recibe. En el caso de que el valor devuelto sea false, si onclick estaba asociado a un link no se efectuará el ir a ese link. Si estaba asociado a un botón submit, no se enviarán los datos del formulario.

 ir a google
No abrirá la página

Combinando el mensaje de confirmación de window.confirm con un evento onclick podemos ofrecer al usuario la posibilidad de cancelar ciertas acciones, como la visita a una página o enviar los datos del formulario

 ir a google

CORRESPONDENCIA ENTRE LAS PROPIEDADES DE LOS ESTILOS EN CSS Y EN JAVASCRIPT

Propiedad CSS	Propiedad DOM en Javascript
background	background
background-attachment	backgroundAttachment
background-color	backgroundColor
background-image	backgroundImage
background-position	backgroundPosition
background-repeat	backgroundRepeat
border	border
border-color	borderColor
border-style	borderStyle

border-top	borderTop
border-right	borderRight
border-left	borderLeft
border-bottom	borderBottom
border-top-color	borderTopColor
border-right-color	borderRightColor
border-bottom-color	borderBottomColor
border-left-color	borderLeftColor
border-top-style	borderTopStyle
border-right-style	borderRightStyle
border-bottom-style	borderBottomStyle
border-left-style	borderLeftStyle
border-top-width	borderTopWidth
border-right-width	borderRightWidth
border-bottom-width	borderBottomWidth
border-left-width	borderLeftWidth
border-width	borderWidth
clear	clear
clip	clip
color	color
display	display
float	cssFloat
font	font
font-family	fontFamily
font-size	fontSize
font-style	fontStyle
font-variant	fontVariant

font-weight	fontWeight
height	height
left	left
letter-spacing	letterSpacing
line-height	lineHeight
list-style	listStyle
list-style-image	listStyleImage
list-style-position	listStylePosition
list-style-type	listStyleType
margin	margin
margin-top	marginTop
margin-right	marginRight
margin-bottom	marginBottom
margin-left	marginLeft
overflow	overflow
padding	padding
padding-top	paddingTop
padding-right	paddingRight
padding-bottom	paddingBottom
padding-left	paddingLeft
position	position
text-aling	textAling
text-decoration	textDecoration
text-indent	textIndent
text-transform	textTransform
top	top
vertical-align	verticalAlign

visibility	visibility
white-space	whiteSpace
width	width
word-spacing	wordSpacing
z-index	zIndex