Especialidad: Técnico en Programación

Módulo III. Submódulo2: Desarrolla aplicaciones que se ejecutan en el cliente.

Prof: Hilda Lucía Rodríguez Gómez

Competencia Profesional: Usa JavaScript para manejar eventos.

MANEJADORES DE EVENTOS

Un evento es una acción que realiza el usuario (hacer click, posicionarse con el ratón sobre un lugar determinado, enviar un formulario,...). ante la cual puede realizarse algún proceso; se capturan con los manejadores de eventos que son mecanismos mediante los que se detectan las acciones y llaman automáticamente a la función que haya asociada para que se realice. JavaScript reconoce una serie de eventos sobre sus objetos predefinidos. La interacción con el usuario se basa en que algunas de las instrucciones HTML admiten como parámetros a algunos de los manejadores de eventos de tal modo que estos se insertan allí y efectúan la llamada a la función asociada en el momento que se produce el evento.

En la siguiente tabla puedes ver los manejadores de eventos, la causa que provoca el evento, las etiquetas HTML en las que puede insertarse el manejador del evento y los objetos predefinidos de JavaScript que admiten el evento:

Evento	Causa del evento	Directivas asociadas	Objetos que lo admiten
onLoad	Se carga una página (entrar)	<BODY>, <IMG>	Image, window
onUnload	Se descarga una página (salir)	<BODY>	window
onMouseOver	El puntero pasa sobre algo	<A HREF>, <AREA>	Link
onMouseOut	El puntero sale de algo	<A HREF>, <AREA>	Link
onSubmit	Se envía un formulario	<FORM>	Form
onClick	Se pulsa el ratón sobre algo	<INPUT TYPE="…">, <AREA>, <A HREF>	Button, document, Checkbox, Link, Radio, Reset, Submit
onBlur	Se pierde el foco del cursor	<INPUT TYPE="…">, <TEXTAREA>	Button, Checkbox, FileUpload, Password, Radio, Reset, Select, Submit, Text, Textarea, window
onChange	Cambia el contenido de algún control de edición o se pierde el foco del cursor	<INPUT TYPE="…">, <TEXTAREA>	FileUpload, Select, Text, Textarea
onFocus	Recibir el foco del cursor	<INPUT TYPE="…">, <TEXTAREA>	Button, Checkbox, FileUpload, Password, Radio, Reset, Select, Submit, Text, Textarea, window
onSelect	Seleccionar texto en un control de edición	<INPUT TYPE="…">, <TEXTAREA>	Text, Textarea
onAbort	Interrumpir la carga de una imagen	<IMG>	Image
onDblClick	Se hace click doble		Document, Link
onDragDrop	Arrastrar y soltar un objeto en la ventana del navegador		Window
onError	Producirse algún error en la carga del documento o imagen	<BODY>,<IMG>	Image, window
onKeyDown	Pulsar una tecla		Document, Link, Textarea
onKeyPress	Mantener pulsada una tecla		Document, Link, Textarea
onKeyUp	Liberar una tecla		Document, Link, Textarea
onMouseDown	Pulsar un botón del ratón	<A HREF>, <AREA>	Button, document, Link
onMouseUp	Liberar un botón del ratón	<A HREF>, <AREA>	Button, document, Link
onMove	Mover la ventana		Window
onReset	Se pulsa sobre el botón reset		Form
onResize	Cambiar el tamaño de una ventana o marco		Window
onMouseMove	Mover el puntero por el documento		

Captura de Eventos

Existen tres diferentes formas de registrar un evento para un elemento HTML: en línea, registrar un manejador de evento como una propiedad del elemento o usar el método addEventListener**()**.

Manejadores de eventos en línea: se trata de utilizar los atributos provistos por HTML para registrar eventos para un elemento en particular. Esta técnica está en desuso, pero en ocasiones puede ser de utilidad. Ejemplos:

<A HREF="enlace.html" onClick= "Hacer_Algo(); ">Sucederá algo al hacer click aquí</A>

```
<!DOCTYPE html>
<html lang=”es”>
<head>
  <title>Este texto es el título del documento</title>
</head>
<body>
  <div id=”principal”>
    <p onclick=”alert(‘hizo clic!’)”>Hacer Clic</p>
    <p>No puede hacer clic</p>
  </div>
</body>
</html>
```

Manejadores de eventos como propiedades: Usando selectores Javascript podemos referenciar el elemento HTML y asignarle el manejador de eventos que queremos como si fuese una propiedad. Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <script>
    function mostraralerta(){
      alert('hizo clic!');
    }
    function hacerclic(){
      document.getElementsByTagName('p')[0].onclick=mostraralerta;
    }
    window.onload=hacerclic;
  </script>
</head>
<body>
  <div id=”principal”>
    <p>Hacer Clic</p>
    <p>No puede hacer Clic</p>
  </div>
</body>
</html>
```

Manejadores de eventos usando el método addEventListener(). Este método tiene tres argumentos: el nombre del evento, la función a ser ejecutada y un valor booleano (falso o verdadero) que indica cómo un evento será disparado en elementos superpuestos.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <script>
    function mostraralerta(){
      alert('hizo clic!');
    }
    function hacerclic(){
      var elemento=document.getElementsByTagName('p')[0];
      elemento.addEventListener(‘click’, mostraralerta, false);
    }
    window.addEventListener(‘load’, hacerclic, false);
  </script>
</head>
<body>
  <div id=”principal”>
    <p>Hacer Clic</p>
    <p>No puede hacer Clic</p>
  </div>
</body>
</html>
```

En resumen:

- Los manejadores de eventos se pueden definir con
  - **objeto.addEventListener(evento, manejador)**
- También se pueden definir como propiedades
  - **objeto.evento = manejador**
    - objeto: objeto DOM al que se asocia el evento
    - evento: nombre (onload, onclick, onmouseover, etc.)
    - manejador: función ejecutada al ocurrir un evento
- Ejemplos
  - img.addEventListener("onclick", function() {... código ...})
  - img.onclick=function() {... código...}

Definir nuevos eventos

Además, JavaScript permite definir eventos y asignarlos a objetos por encima de los objetos para los que están definidos. Para ello los objetos window, document y layer utilizan los siguientes métodos:

captureEvents(): Captura eventos del tipo que se especifique.
releaseEvents(): Ignora la captura del tipo especificado.
routeEvent(): Envía el evento capturado a u objeto.

Secuencia de captura, definición y activación de un gestor de eventos:

1.- Especificamos el tipo de eventos que queremos capturar, por ejemplo, los de tipo click. Se pueden especificar varios eventos, separados por | (OR).

```
window.captureEvent(Event.CLICK [| Event.* | Event.*])
```

2.- Creamos la función que realice las acciones asociadas a dicho evento.

```
function evento_nombre([parámetros]){
        acciones
}
```

3.-Asignamos al evento la función especificada.

```
window.onClick=evento_nombre;
```

También existe la posibilidad de utilizar eventos sobre objetos para los que no están definidos. Por ejemplo, las imágenes sólo admiten como eventos standard (en todos los navegadores) onLoad, onError y onAbort, pero muchas veces es necesario que soporten eventos como onClick, onmouseOver, etc., y en todo tipo de navegadores. Como sabemos que el objeto Link sí admite estos eventos, podemos hacer que la imagen contenga un enlace y así ya están disponibles. Sólo resta anular la dirección del enlace con el símbolo #, que no remite a ningún sitio:

```
<a href="#" onClick="Hacer_Algo();";return false;"><img src="imagen.gif" border="0"></a>
```