Eventos DOM

JavaScrypt

Eventos

Hasta ahora, todas las aplicaciones y scripts que se han creado tienen algo en común: se ejecutan desde la primera instrucción hasta la última de forma secuencial.

Gracias a las estructuras de control de flujo (**if, for, while**) es posible modificar ligeramente este comportamiento y repetir algunos trozos del script y saltarse otros trozos en función de algunas condiciones.

Eventos

Este tipo de aplicaciones son poco útiles, ya que no interactúan con los usuarios y no pueden responder a los diferentes ***eventos*** que se producen durante la ejecución de una aplicación.

Afortunadamente, las aplicaciones web creadas con el lenguaje JavaScript pueden utilizar el modelo de ***programación basada en eventos***.

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.

Eventos

Así, por ejemplo, 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**"*.

Tipos de eventos

En este modelo, cada elemento o etiqueta XHTML define 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 diferentes y un mismo elemento XHTML puede tener asociados varios eventos diferentes.

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**.

Tipos de eventos

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

Tipos de eventos

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

Evento	Descripción	Elementos para los que está definido
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>

Tipos de eventos

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.

Tipos de eventos

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.

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.

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, por lo que la aplicación puede *responder* ante cualquier evento que se produzca durante su ejecución.

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

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. 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="alert('Gracias por pinchar');" />
```

Manejadores como atributos XHTML

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**.

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 **(alert('Gracias por pinchar');)**, ya que solamente se trata de mostrar un mensaje.

Manejadores como atributos XHTML

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:

```
<div onclick="alert('Has pinchado con el ratón');“
 onmouseover="alert('Acabas de pasar el ratón por encima');"> Puedes pinchar sobre este elemento o simplemente pasar el ratón por encima
</div>
```

Manejadores como atributos XHTML

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

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

Manejadores como atributos XHTML

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.

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.

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.

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

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.

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.

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

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.

Manejadores de eventos y variable this

Elemento ** ** original:

```
<div id="contenidos" style="width:150px;
height:60px; border:thin solid silver">
Sección de contenidos...
</div>
```

Manejadores de eventos y variable this

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('contenido
s').style.borderColor='black';"
onmouseout="document.getElementById('contenidos'
).style.borderColor='silver';">
 Sección de contenidos...
</div>
```

Manejadores de eventos y variable this

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.

Manejadores de eventos y variable this

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

```
<div id="contenidos" style="width:150px;
height:60px; border:thin solid silver"
onmouseover="this.style.borderColor='black';"
onmouseout="this.style.borderColor='silver';">
 Sección de contenidos...
</div>
```