

Guía Teórica N°05

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.

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*.

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"*.

Tipos de Eventos:

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**.

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
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.

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"*.

Se recomienda utilizar los manejadores a través de funciones externas o de manera semántica, evitando así tener código JavaScript en documento HTML y permitiendo así una mejor interpretación del código por parte de aquellos programadores que vayan a leerlo.

- **Como Funciones Externas:** La definición de los manejadores de eventos en los atributos XHTML 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 XHTML.

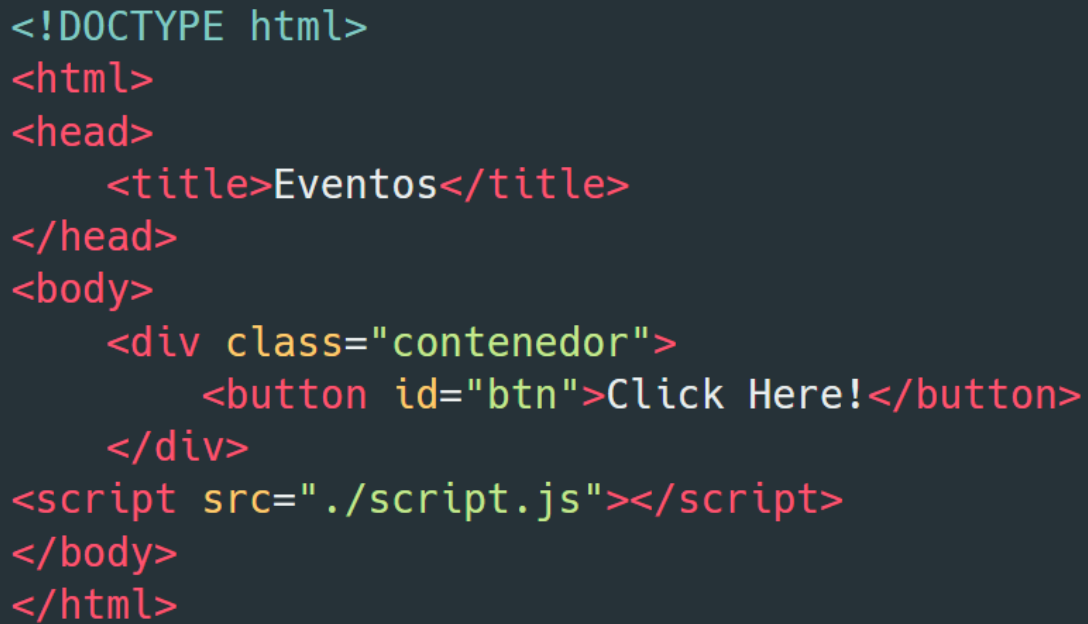
- **Manejadores de eventos semánticos:** Como es conocido, una de las buenas prácticas básicas en el diseño de páginas y aplicaciones web es la separación de los contenidos (XHTML) y su aspecto o presentación (CSS). Siempre que sea posible, también se recomienda separar los contenidos (XHTML) y su comportamiento o programación (JavaScript).

Mezclar el código JavaScript con los elementos XHTML solamente contribuye a complicar el código fuente de la página, a dificultar la modificación y mantenimiento de la página y a reducir la semántica del documento final producido.

Afortunadamente, existe un método alternativo para definir los manejadores de eventos de JavaScript. Esta técnica es una evolución del método de las funciones externas, ya que se basa en utilizar las propiedades DOM de los elementos XHTML para asignar todas las funciones externas que actúan de manejadores de eventos.


A continuación, veremos un ejemplo haciendo uso de los manejadores eventos definiéndole a un elemento `<button>` el evento click, donde cada vez que el usuario haga click sobre el botón, se disparará un ***alert*** con un mensaje indicado.

- **Código HTML:**



```
<!DOCTYPE html>
<html>
<head>
  <title>Eventos</title>
</head>
<body>
  <div class="contenedor">
    <button id="btn">Click Here!</button>
  </div>
<script src="./script.js"></script>
</body>
</html>
```

- Código JavaScript:



```
const btn = document.querySelector('#btn');
const handleClick = () => {
  alert('Hello!!');
}
btn.onclick = handleClick;
```


addEventListener():

Registra un evento a un objeto en específico. El Objeto específico puede ser un simple elemento en un archivo, el mismo documento.

Para registrar más de un evento, puedes llamar `addEventListener()` para el mismo elemento pero con diferentes tipos de eventos o parámetros de captura.

Históricamente, este método traía consigo varias incompatibilidades con diferentes navegadores puesto a que no todos soportaban esta nueva definición para agregar eventos a los elementos HTML pero actualmente, esta situación se ha venido solucionando a tal punto en donde la mayoría de navegadores aceptan esa forma de hacer escuchar eventos y se está volviendo un estándar al momento de trabajar con la programación Orientada a Eventos en JavaScript.

Basándonos en el ejemplo anterior, haciendo uso de este método, el código JavaScript resultaría de la siguiente manera:



```
const btn = document.querySelector('#btn');  
btn.addEventListener('click', () => {  
    alert('Hello!!');  
});
```

¡Nos vemos en la próxima!

Hemos aprendido los fundamentos de JavaScript y la manipulación directa del DOM, pero esto no implica que nuestro camino de aprendizaje haya finalizado. Aún queda mucho por aprender, pero no te desanimes, con lo aprendido hasta ahora tienes las capacidades de desarrollar páginas web dinámicas, pero todo dependerá de qué tan creativo seas al momento de escribir tu código.

No tengas miedo, proponte realizar un mini proyecto donde coloques todo lo aprendido en el curso de HTML, CSS, Programación en JavaScript y en este.

Recuerda que cualquier ayuda que necesites, podrás contactar con nosotros a través del correo electrónico educa2ucv@gmail.com o también puedes escribirnos directamente en los medios de comunicación que te hemos facilitado (Facebook, Instagram, Twitter, WhatsApp, Telegram o Discord).

¡GRACIAS POR CONFIAR EN NOSOTROS!