APRENDERAPROGRAMAR.COM

# EVENTOS JAVASCRIPT. TIPOS. PROPAGACIÓN. MODELOS. MANEJADOR O EVENT HANDLER. CONFIRM. EJEMPLO. (CU01157E)

Sección: Cursos
Categoría: Tutorial básico del programador web: JavaScript desde cero

Fecha revisión: 2029

**Resumen:** Entrega nº57 del Tutorial básico "JavaScript desde cero".

Autor: César Krall

## CONCEPTO DE EVENTO JAVASCRIPT

Un evento es algo que ocurre y puede ser detectado. Esta definición de partida es quizás demasiado simple. Podríamos decir que si un usuario hace click sobre un botón en una página web se produce un evento (el usuario ha hecho click), pero también que si el usuario estornuda se produce un evento, lo cual realmente no entra dentro del ámbito de la programación JavaScript, por lo que hemos de acotar esta definición.


JavaScript es un lenguaje que puede realizar tareas propias de la programación tradicional como ejecuciones secuenciales de instrucciones o bucles iterativos, pero está especialmente preparado para ser un lenguaje de respuesta dinámica a los eventos que genera el usuario. Por ello muchas veces se alude a JavaScript como un lenguaje dirigido por eventos o como un tipo de programación basada en eventos. Según este modelo de programación, la aplicación web está "a la espera" de que el usuario haga algo (pulsar un botón, un link, cerrar una ventana, etc.) y es cuando existe una acción del usuario cuando se ejecuta un fragmento de código. Esto permite que las aplicaciones web sean interactivas en base al ciclo acción del usuario – reacción de la aplicación.

Podríamos hablar de distintos tipos de eventos:

**a) Eventos interceptables y eventos no interceptables:** un evento interceptable es aquel que puede ser detectado por el sistema operativo o el navegador y generar la ejecución de un fragmento de código JavaScript como consecuencia de ello. Estos son los eventos JavaScript con los que trabajaremos. Un evento no interceptable sería aquel que no puede ser detectado, como que el usuario se rasque la cabeza o que reciba un mensaje de correo electrónico en una aplicación local.

**b) Eventos disparados por el usuario y eventos disparados por el sistema:** cada vez que el usuario realiza una acción interceptable, como hacer click ó pulsar una tecla, decimos que se dispara un evento generado por el usuario. Pero además de eventos disparados por el usuario, existen eventos disparados por el sistema operativo o por el navegador. Por ejemplo, cuando termina la carga de una página web en el navegador se dispara el evento de sistema <<onload>> (carga terminada). También podemos incluir instrucciones específicas JavaScript para que se ejecuten cuando tenga lugar un evento de sistema.

## TIPOS DE EVENTOS. OBJETIVO O TARGET DE EVENTO.

Cada evento tiene un **nombre**, al que también se llama "**tipo de evento**". Por ejemplo al evento correspondiente a hacer click se le identifica como "onclick". Normalmente los nombres se forman anteponiendo el prefijo on con el nombre de lo que ocurre en inglés. Así por ejemplo "onload" indica "cuando termina la carga del documento HTML".

Cada evento está asociado a un objeto que es quien recibe o detecta el evento, al que se denomina target u objetivo del evento. Así, el evento onclick del ejemplo anterior tiene como objetivo un nodo HTML de imagen, y un nodo HTML div. El evento onload tiene como target el objeto window u objeto global.

**MODELOS DE EVENTOS. PROPAGACIÓN.**

Al igual que definir la estructura de un documento HTML es complejo y por ello se creó el DOM (Document Object Model), para poder describirla y manejarla, existen modelos de eventos para describir y manejar los eventos.

El problema reside en que a lo largo del tiempo se crearon distintos modelos de eventos propuestos por distintas organizaciones (Netscape, Microsoft, W3C, etc.). Distintos navegadores adoptaron distintos modelos de eventos, lo cual hacía difícil que el código programado y útil en un navegador funcionara en otro que usaba un modelo de eventos distinto.

Un modelo de eventos implica como aspecto importante qué eventos son detectados y cuáles son sus nombres. Pero esta no es la única cuestión relacionada con los eventos. Otra cuestión importante es en qué orden se ejecutan dos eventos que suceden simultáneamente. Por ejemplo, si tenemos un evento onclick definido para un div, y otro evento onclick definido para una imagen dentro del div, al hacer click sobre la imagen se producen dos eventos simultáneamente: el evento onclick correspondiente al div y el evento onclick correspondiente a la imagen. ¿Qué código de respuesta se debería ejecutar primero, el correspondiente al div o el correspondiente a la imagen? Al orden en que se van disparando los eventos se le denomina propagación de eventos. La propagación de eventos puede ser desde dentro hacia fuera (p.ej. primero el evento de la imagen y luego el del div), desde fuera hacia dentro (p.ej. primero el evento del div y luego el de la imagen), o para casos más complejos, según otras formas de establecer el orden.

Cuestiones como esta también están comprendidas en el modelo de eventos. Al existir distintos modelos de eventos, el resultado era (y en cierta medida es) que el orden en que se ejecutaban los eventos dependía del navegador que estuviéramos utilizando.

Con el paso del tiempo se ha ido produciendo cierta estandarización entre navegadores, aunque esta todavía no es completa. Nosotros en este curso no vamos a entrar a estudiar en profundidad ningún modelo de eventos en concreto, sólo haremos referencias puntuales a ellos cuando lo consideremos necesario. Nos vamos a limitar a describir la forma más estandarizada y aceptada por la industria (los fabricantes de navegadores) y la comunidad de programadores para trabajar con eventos.

Pero debes tener en cuenta lo siguiente: distintos navegadores pueden responder de distinta manera a un mismo código. Algunos navegadores pueden no reconocer el código que sí es reconocido por otros navegadores. Algunos navegadores pueden considerar como eventos algo que otros navegadores simplemente ignoran y no son capaces de capturar.

¿Debemos preocuparnos por esto? Pues la verdad es que no, ya que eso no nos aportará ninguna solución. La estrategia que recomendamos es tratar de usar el código más inter-compatible que podamos y sepamos, siendo conscientes de que hoy por hoy es imposible conseguir que una página web se muestre exactamente igual y responda exactamente igual en todos los navegadores.

Conociendo que no podemos cambiar esta situación, nuestro objetivo será crear páginas web cuyo funcionamiento sea correcto en el mayor número de navegadores posible.

## MANEJADOR DE EVENTOS O EVENT HANDLER COMO ATRIBUTO DE ELEMENTO HTML. CONFIRM.

¿Cómo indicar en el código que una serie de instrucciones deben ejecutarse cuando tiene lugar un evento?

La forma más tradicional sería la consistente en añadir un atributo a la etiqueta HTML que recibe el evento, también llamado "captura de eventos en línea":

```
<etiqueta nombreEvento="acción a ejecutar"> ... </etiqueta>
```

Donde acción a ejecutar es un fragmento de código JavaScript, que en caso de comprender varias sentencias deben ir separadas por punto y coma.

```
<a onclick="alert('Cambiamos la url')" href="http://aprenderaprogramar.com">Aprende a programar</a>
```

En este ejemplo podemos decir que onclick es el nombre del evento (aunque a veces se prefiere decir que el nombre del evento es click). También podemos decir que es un atributo de la etiqueta <a>, y podemos decir que es el manejador del evento: el elemento que dice qué ha de ejecutarse cuando se produzca el evento.

Una posibilidad interesante de este tipo de manejadores de eventos es que podemos pasar a una función el elemento HTML que recibe el evento usando la palabra clave this. Por ejemplo:

```
<a onclick="llamarFuncion(this)" href="http://aprenderaprogramar.com">Aprende a programar</a>
```

Al invocar a this estamos pasando el elemento HTML, nodo del DOM, <a>, lo cual puede ser de interés en numerosas ocasiones en que nos interese ejecutar una serie de acciones sobre el elemento que recibe la acción.

Con esta forma de manejo nos podemos plantear una pregunta: si hay una acción predeterminada cuando se pulsa el link (dirigirnos a la url de destino) ¿Qué se ejecuta primero, el código de respuesta o la acción predeterminada? Se estandarizó que en primer lugar se ejecuta el script de respuesta al evento, y luego la acción predeterminada.

Pero como en programación todo es posible, se planteó la cuestión: ¿Por qué no idear una forma de hacer que la ejecución predeterminada se produzca sólo cuando nosotros queramos?

La solución que se le dio a esto fue permitir que el manejador del evento devolviera un valor booleano: true (por defecto) si se debía ejecutar la acción predeterminada, o false para evitar que se ejecutara la acción predeterminada.

```
<etiqueta nombreEvento="acción a ejecutar; return valorBooleano"> ... </etiqueta>
```

```
<a onclick="alert('Link desactivado'); return confirm('Va a acceder a aprenderaprogramar.com')"
href="http://aprenderaprogramar.com">Aprende a programar</a>
```

La ejecución de la acción predeterminada se podía dejar en manos del usuario a través de un mensaje de confirmación usando la función global de JavaScript confirm, de modo que el manejador devuelve true si el usuario pulsa “Aceptar” o false si el usuario pulsa en “Cancelar”.

```
<a onclick="return confirm('Va a acceder a aprenderaprogramar.com ¿Está seguro?')"
href="http://aprenderaprogramar.com">Aprende a programar</a>
```

Algunos eventos, como el cierre de una ventana, pueden generar la ejecución de un código JavaScript, pero no se puede impedir en este caso la acción predeterminada porque sería “ir contra los deseos del usuario”. Quizás se pudiera impedir, pero no tendría lógica.

Esta forma de introducir manejadores de eventos, digamos que la más antigua de todos, es reconocida por todos los navegadores, al igual que los cuatro eventos “mágicos” (en el sentido de que cuando se introdujeron fueron un gran avance en el desarrollo web) onclick, onload, onmouseover, onmouseout.

Desde la época del navegador Netscape en que se introdujeron los manejadores de eventos básicos y los eventos básicos hasta hoy en día, ha habido una larga y complicada evolución.

Hoy día podemos decir que la captura de eventos en línea, aunque es compatible con todos los navegadores, tiene un serio inconveniente: nos obliga a mezclar aspectos de comportamiento o respuesta dinámica con aspectos de estructura del documento HTML. Como sabemos, hoy día hay consenso en que es ventajoso separar la estructura (HTML) de la presentación (CSS) y del comportamiento (JavaScript).

## MANEJADORES DE EVENTOS COMO PROPIEDADES DE OBJETOS

Podemos independizar los manejadores de eventos usando código JavaScript para crearlos manejando la propiedad de nodos HTML del DOM que representa el atributo de la etiqueta HTML.

Lo entenderemos mejor con un ejemplo:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><title>Ejemplo aprenderaprogramar.com</title><meta charset="utf-8">
</head>
```

```
<body>
<div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
<div style="color:blue;" id ="pulsador" onclick="ejemplo()"> Probar </div>
<a href="http://aprenderaprogramar.com">Aprende a programar.</a>
<a href="http://aprenderaprogramar.es">Inténtalo, será divertido.</a>
</body>
</html>
<script type="text/javascript">
var elemsA = document.getElementsByTagName('a');
for (var i=0;i<elemsA.length;i++) {
elemsA[i].onclick = function() {return confirm('Ud. va a ser transferido de url hacia ' + this.href);}
}
</script>
```

Sobre este ejemplo tenemos que llamar la atención sobre algunas cuestiones:

a) El código JavaScript está incluido detrás del código HTML. ¿Por qué? Porque si es incluido antes del código HTML, se ejecuta antes de que se haya cargado la página, y al no haberse cargado los elementos HTML no es posible establecer la propiedad onclick de los mismos (es decir, getElementsByTagName('a') nos devuelve cero elementos cuando todavía no se ha cargado la página web.

b) A cada nodo de tipo link (a) se le asigna como propiedad onclick una función, en este caso haciendo uso de return y confirm, de modo que el usuario puede elegir entre aceptar el ir al link elegido o cancelar.

c) this dentro de la función anónima a ejecutar ante un evento onclick sobre un elemento a hace referencia al elemento propietario de la función, en este caso el propio nodo de tipo link a, de modo que podemos rescatar la url de destino escribiendo this.href.

Una forma alternativa de escribir el script sería:

```
<script type="text/javascript">
var elemsA = document.getElementsByTagName('a');
for (var i=0;i<elemsA.length;i++) {
elemsA[i].onclick = transferir;}
function transferir() {
return confirm('Ud. va a ser transferido de url hacia ' + this.href);
}
</script>
```

En este caso usamos una asignación a la función. Tener en cuenta que no escribimos elemsA[i].onclick = transferir();} porque esto daría lugar a que se ejecutar la función (invocación). Al escribir la asignación sin paréntesis la función queda asignada pero no se ejecuta.

Finalmente tenemos una alternativa "más lógica" que nos permite no tener que poner el script después del código HTML. La idea es la siguiente: el evento onload nos informa de cuándo se ha completado la carga de la página web. En ese momento dispararemos una función que se encargará de introducir los manejadores de eventos que queramos. Como en el momento en que se dispara la función la carga de

la página ya se ha realizado, no es necesario que el código JavaScript para realizar este proceso quede "debajo" del código HTML.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><title>Ejemplo aprenderaprogramar.com</title><meta charset="utf-8">
<script type="text/javascript">
window.onload = function () {
var elemsA = document.getElementsByTagName('a');
for (var i=0;i<elemsA.length;i++) {
elemsA[i].onclick = transferir;}
function transferir() { return confirm('Ud. va a ser transferido de url hacia ' + this.href);}
}
</script>
</head>
<body><div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
<div style="color:blue;" id ="pulsador" onclick="ejemplo()"> Probar </div>
<a href="http://aprenderaprogramar.com">Aprende a programar.</a>
<a href="http://aprenderaprogramar.es">Inténtalo, será divertido.</a>
</body>
</html>
```

Recordar que al escribir window.onload = algo(); la función algo() se ejecutará porque mediante esta instrucción estamos indicando que esa función debe ejecutarse cuando se produzca el evento onload sobre el objeto window.

Por ejemplo window.onload = alert ('La página ha terminado de cargar'); nos permite mostrar un mensaje de alerta advirtiendo de que la página ha terminado de cargar.

**EJERCICIO**

A partir del siguiente código HTML, crea un script que cumpla los requisitos indicados más abajo:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html> <head><title>Portal web - aprenderaprogramar.com</title>
<meta name="description" content="Portal web aprenderaprogramar.com">
<meta name="keywords" content="aprender, programar, cursos, libros"><meta charset="utf-8">
</head>
 <body> <p><a href="principal.html" title="Página principal" >Ir a la pagina principal</a></p>
<h1>Novedades</h1>
<p>Aquí presentamos las novedades del sitio.</p>
<h3>Lanzamos el producto X-FASHION</h3>
<p>Este producto permite estirar la piel hasta dejarla como la de un bebé.</p>
<p><img src="http://i.imgur.com/tq5Bs.png" title="Imagen del producto" alt="imagen producto x-fashion" /></p>
<h3>Mejoramos el producto T-MOTION</h3>
<p>Hemos lanzado una nueva versión del producto T-MOTION</p>
<p><img src="http://i.imgur.com/SpZyc.png" title="Imagen del proudcto tmotion" alt="imagen producto tmotion" /></p>
</body>
</html>
```

a) Mediante el control de los eventos onmouseover y onmouseout, debemos hacer que cuando el usuario pase el ratón sobre las etiquetas h1 y h3, el color del texto pase a ser orange y cuando deje de pasarlo el texto quede en marrón.

b) Mediante el control del evento onmouseover, debemos hacer que cuando el usuario pase el ratón sobre un párrafo, el color de fondo del párrafo sea amarillo y cuando deje de pasarlo no haya color de fondo.

c) El código JavaScript debe estar situado entre las etiquetas <head> ... </head>

Para comprobar si tus respuestas y código son correctos puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01158E

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=78&Itemid=206