APRENDERAPROGRAMAR.COM

¿QUÉ SIGNIFICA JAVASCRIPT VOID (0) Y JAVASCRIPT: EN HREF? ¿QUÉ DIFERENCIA RETURN FALSE Y PREVENTDEFAULT? (CU01184E)

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

Fecha revisión: 2029

Resumen: Entrega nº84 del Tutorial básico "JavaScript desde cero".

Autor: César Krall

ALGUNOS DETALLES JAVASCRIPT

Vamos a detenernos a estudiar algunos detalles de JavaScript.

BURBUJEO EN LA PROPAGACIÓN DE EVENTOS

Cuando hablamos sobre los eventos en apartados anteriores del curso indicamos que pueden producirse varios eventos simultáneamente y que en este caso la respuesta a los eventos sigue un orden. Si no se indica otra cosa, el evento "burbujea" desde dentro hacia fuera y a esto se le llamaba bubbling. Por ejemplo, si tenemos tres div, uno interno, otro intermedio y otro externo, y definimos una función de respuesta al evento click sobre los elementos div, cuando hacemos click en el div interno en primer lugar se ejecuta la función de respuesta para el div interno, luego para el div intermedio y luego para el div externo.

Ejecuta este código y comprueba sus resultados:

```
<!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">
<style type="text/css"> *{font-family:sans-serif;}
div {margin:20px; padding:20px; background-color: yellow; border-style:solid;}</style>
<script type="text/javascript">
window.onload = function () {
var elemsDiv = document.querySelectorAll('div');
for (var i=0;i<elemsDiv.length;i++) {elemsDiv[i].addEventListener("click", decirHola);}
}
function decirHola() {alert('Hola soy el '+this.className);}
</script>
</head>
<body><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3>
<div class="externo"> Soy el div externo:
        <div class="intermedio"> Soy el div intermedio
                <div class="interno"> Soy el div interno</div>
        </div>
</div>
</body></html>
```

El resultado esperado es que si hacemos click en el div interno se muestre por pantalla: Hola soy el interno, Hola soy el intermedio, Hola soy el externo.

Aquí comprobamos que primero se ejecuta la respuesta el evento asociado al elemento más interno, y seguidamente las respuestas de elementos envolventes sucesivamente hasta llegar a completar todos los eventos.

Podemos evitar que el evento burbujee indicando que no debe propagarse hacia elementos externos. Esto lo podemos hacer cambiando la función decirHola. Cámbiala por este código y comprueba lo que ocurre:

```
function decirHola(elEvento) {
alert('Hola soy el '+this.className);
 elEvento.stopPropagation();
}
```

Ahora el único evento que se ejecuta es el primero (más interno), y al encontrarnos con stopPropagation() queda detenida la propagación.

EVITAR LA ACCIÓN DE DEFECTO

Recordar que algunos eventos tienen una acción de defecto, por ejemplo el evento submit de un formulario tiene como acción de defecto el envío del formulario, o el evento click sobre un link tiene como acción de defecto el cargar la página referenciada por el link en el navegador.

La acción por defecto puede ser anulada introduciendo preventDefault() en la función manejadora del evento. Escribe este código y comprueba sus resultados:

```
<!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">
<style type="text/css"> *{font-family:sans-serif;}
div {margin:20px; padding:20px; background-color: yellow; border-style:solid;}</style>
<script type="text/javascript">
window.onload = function () {
var elemsLink = document.querySelectorAll("div a");
var elemsDiv = document.querySelectorAll('div');
for (var i=0;i<elemsLink.length;i++) {
elemsLink[i].addEventListener("click", decirHola);
elemsDiv[i].addEventListener("click", decirHola);
}
}
function decirHola(elEvento) {
elEvento.preventDefault();
alert('Hola soy el '+this.parentNode.className);
elEvento.stopPropagation();
}
</script>
</head>
<body>
<h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3>
<div class="externo"> Soy el div externo: <a href="http://aprenderaprogramar.com"> Aquí un link </a>
<div class="intermedio"> Soy el div intermedio <a href="http://aprenderaprogramar.es"> Aquí un link </a>
<div class="interno"> Soy el div interno <a href="http://aprendeaprogramar.es"> Aquí un link </a> </div>
</div>
</div>
</body></html>
```

El resultado esperado es que si hacemos click en el link más interno se muestre únicamente <<Hola soy el interno>>. Sin embargo, no se carga la página web referenciada por el link debido al preventDefault(), y no se ejecuta la respuesta a los eventos asociados a los div debido a que el evento no se propaga debido a stopPropagation().

AL ESTILO TRADICIONAL

Todavía se pueden encontrar muchos programadores o código existente donde para prevenir la acción de defecto se usa una sintaxis más tradicional basada en definir la respuesta al evento dentro del código HTML y prevenir la acción de defecto escribiendo return false. Ejecuta este código y comprueba sus resultados:

```
<!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">
<style type="text/css"> *{font-family:sans-serif;}
div {margin:20px; padding:20px; background-color: yellow; border-style:solid;}</style>
<script type="text/javascript">

function decirHola(elEvento, that) {
alert('Hola soy el '+that.parentNode.className);
return false;
}

</script>
</head>
<body>
<h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3>
<div class="externo" onclick="return decirHola(event, this);"> Soy el div externo: <a onclick="return decirHola(event,
this);" href="http://aprenderaprogramar.com"> Aquí un link </a>
<div class="intermedio" onclick="return decirHola(event, this);"> Soy el div intermedio <a onclick="return
decirHola(event, this);" href="http://aprenderaprogramar.es"> Aquí un link </a>
<div class="interno" onclick="return decirHola(event, this);"> Soy el div interno <a onclick="return decirHola(event, this);"
href="http://aprendeaprogramar.es"> Aquí un link </a> </div>
</div>
</div>
</body></html>
```

En lugar de incluir el return al final de la función también podríamos introducirlo en línea con esta sintaxis: onclick="decirHola(event, this); return false;"

De este modo, la sentencia return queda en línea después de invocar la ejecución de la función de respuesta sin necesidad del return en la llamada ni dentro de la propia función.

En este caso al pulsar en el link interno se nos muestra: <<Hola soy el interno, Hola soy el intermedio, Hola soy el externo, Hola soy el >> que se corresponde con los eventos click en el link a, más click en cada uno de los tres div (suponiendo un total de 4 elementos). La propiedad className no está definida para el nodo padre del div más externo, de ahí que nos aparezca como una cadena vacía.

Fíjate en que en JavaScript return false tiene el mismo efecto que preventDefault(). En JavaScript return false no evita la propagación del evento, del mismo modo que preventDefault() tampoco la evita. Si quisiéramos evitar la propagación del evento hemos de introducir stopPropagation().

¿QUÉ SIGNIFICA JAVASCRIPT VOID?

Cuando se revisa código HTML y JavaScript con frecuencia nos encontramos con expresiones del tipo:

```
<a href="javascript:void(0);">  Aquí un texto </a>
```

¿Qué significa javascript:void(0)?

Para entender esto primeramente nos vamos a referir a qué significa javascript: en el contexto del atributo href cuando escribimos javascript: estamos indicando que en lugar de llevar a una dirección web, se ejecute el código javascript que vaya indicado a continuación de los dos puntos.

Escribe este código y comprueba sus resultados:

```
<!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">
function mostrarMensaje() {
alert('Aham, lo que parecía un link en realidad lo que está haciendo es ejecutar una función javascript!');
}
</script>
</head>
<body><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3>
<a href="javascript:mostrarMensaje();"> Pulsa aquí por favor </a>
</body></html>
```

Aquí tenemos más o menos claro que lo que se indica es: cuando se pulse sobre el enlace, ejecuta la función JavaScript indicada.

Un efecto parecido se puede lograr si escribimos Pulsa aquí por favor

Sin embargo, escribir como destino del href # mueve la vista al comienzo del documento html (para ver este efecto tiene que tratarse de un enlace colocado en una página web que tenga un scroll vertical).

Hay una forma sencilla de evitar el retorno al comienzo de la página en este caso: hacer que el evento devuelva false, con lo que la acción de defecto (ir al comienzo de la página no se ejecutará). El código sería este:

 Pulsa aquí por favor

Piensa en otra alternativa como esta: Pulsa aquí por favor

¿Qué hará este código? Este código ejecutará el código javascript después de los dos puntos. Después de los dos puntos tenemos una doble barra que es el símbolo de comentario en JavaScript. ¿Entonces que hará? Pues no hará nada, porque ejecutar un comentario equivale a no hacer nada.

Este tipo de código normalmente lo encontraremos en situaciones como esta:

 Pulsa aquí para imprimir

¿Qué se está haciendo aquí?

a) Se crea el aspecto de un link con el objetivo de "avisar" al usuario de que puede hacer click sobre ese elemento.

b) El link en sí no lleva a ningún lado

c) Como respuesta al evento click se ejecuta una función JavaScript.

Este planteamiento no parece muy adecuado: incluir un link que no lleva a ninguna parte simplemente para que el usuario sepa que puede pulsar ahí para ejecutar algo. ¿Tiene sentido? Posiblemente no demasiado, porque no tiene lógica crear un link que no linka a ningún lado: es confuso. La alternativa recomendada sería incluir el texto dentro de un elemento html sin necesidad de escribir una etiqueta de link, y dotarla del estilo adecuado (como si fuera un link, pero sin serlo) para que el usuario sepa que puede pulsar ahí. La idea sería de este tipo:

<span style="comoSiFueraLink" onclick="imprimir()> Pulsa aquí para imprimir

Esto es más correcto: aquí está claro lo que se quiere hacer y se hace de una forma coherente.

En muchas ocasiones nos encontraremos algo como esto:

Pulsa aquí para logarte

Void es un operador JavaScript que evalúa (ejecuta) la expresión que se le pasa como argumento y a continuación devuelve <<undefined>>. En este caso, javascript:void(0) ejecuta 0, que en realidad no tiene ningún efecto, y devuelve undefined. ¿Y qué significa esto? Pues lo mismo que href="javascript://", es decir, que simplemente se crea un link que no enlaza con ningún lado (o podríamos decir que enlaza con undefined, lo cual es enlazar con ningún lado).

¿Entonces tiene void alguna utilidad? Puede tenerla. Como hemos visto, nos puede servir para decir que el link no lleve a ningún lado, aunque por otro lado como respuesta al evento sí puede que se ejecuten ciertos efectos colaterales.

Otra utilidad sería dar lugar a la ejecución de algo devolviendo undefined. Por ejemplo, ejecuta este código y comprueba sus resultados:

```
<!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>
<h2>Cursos aprenderaprogramar.com</h2>
<h3>Ejemplos JavaScript</h3>
<a href="javascript:void(document.body.style.backgroundColor='yellow');">
 Pulsa aquí para cambiar a color de fondo amarillo
</a>
</body>
</html>
```

Aquí tenemos un link que no lleva a ningún lado (ya que void termina devolviendo undefined), y sin embargo existe un efecto colateral derivado de la expresión que se le pasa como argumento a void.

Obtendríamos el mismo efecto escribiendo esto:

Pulsa aquí para cambiar a color de fondo amarillo

Que posiblemente es más claro (se dice que “oscurece menos el código”). Pero más claro todavía es no crear un link si realmente no es un link, por tanto preferimos usar esto otro:

Pulsa aquí para cambiar a color de fondo amarillo

Obviamente para usar esto tendremos que haber definido el estilo CSS .comoSiFueraLink para darle la apariencia al texto de un link.

Algunos expertos consideran que javascript void(0) es una mala práctica de programación (de hecho hay una expresión acuñada en referencia a ello: <<javascript void(0) must die>>).

Nosotros no recomendamos su uso, pero es útil conocer su significado porque ha sido usado y sigue usándose, con lo cual es posible que nos encontremos código que se use cuando estemos revisando código desarrollado por otros programadores.

RESUMEN

Con javascript:void(0) y otras expresiones similares se busca que el usuario perciba como algo donde puede hacer click un texto, por ejemplo “Imprimir”, y para ello se simula un link. De esta forma el navegador mostrará el texto con un color y cursor especial si se pasa el ratón encima de él. Esto no es una idea muy afortunada y sin embargo ha sido bastante usada debido a que es simple.

EJERCICIO

Escribe este código en un editor y responde a las siguientes preguntas:

```
<!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><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3>
<a href="javascript:void(0);" onclick="if (confirm('Are you sure?')) { var f = document.createElement('form');
 f.style.display = 'none'; this.parentNode.appendChild(f); f.method = 'POST'; f.action = this.href;var m =
document.createElement('input');
 m.setAttribute('type', 'hidden'); m.setAttribute('name', '_method'); m.setAttribute('value', 'delete'); f.appendChild(m);
 var s = document.createElement('input'); s.setAttribute('type', 'hidden'); s.setAttribute('name', 'authenticity_token');
 s.setAttribute('value', 'aprenderaprogramar.com='); f.appendChild(s);f.submit(); };return false;">Pulsa aquí para
proceder</a>
</body></html>
```

a) ¿Para qué se utiliza aquí javascript: void(0)?

b) Explica paso a paso qué es lo que hace el código JavaScript que se ha incluido como respuesta al evento click.

c) ¿Qué implica el uso de return false; en este código?

d) Modifica el código para que la apariencia y resultados sean iguales pero sin usar un elemento <a> ...

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

Próxima entrega: CU01185E

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