Hoy por hoy casi todos los que trabajamos en entorno web utilizamos JQuery como framework base para la mayoría de nuestros proyectos y a veces es importante conocerlo a detalle para sacarle el mayor partido posible . Hoy hablaré un poco de la gestión de eventos que es algo que todos utilizamos pero que no todos conocemos en profundidad. Utilizamos continuamente eventos como (click, change, submit etc) . Ahora bien ¿como funcionan realmente?. Vamos a ver un ejemplo sencillo .



En este ejemplo disponemos de una lista con un único elemento y un link dentro de ella cuando pulsamos el link nos saldrá un mensaje de Javascript “has pulsado el boton” . A continuación se muestra el código fuente

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
<script type="text/javascript" src="jquery-1.10.2.min.js">
</script>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
<title>Insert title here</title>
<script type="text/javascript" src="jquery-1.10.2.min.js">
</script>
<script type="text/javascript">
$(document).ready(function(evento) {

$("a").click(function(evento) {

 alert("has pulsado el boton");
 evento.preventDefault();
 });
});
</script>
</head>
<body>
<ul>
<li>
<p>Parrafo dentro de una lista
<a href="destino.php">ir a pagina de destino</a>
</p>
</li>
</ul>
</body>
</html>
```

Lo que mas nos interesa en este caso es el código de Javascript (JQuery) que hemos utilizado para ligar el evento “click” con el mensaje de “alert”. Como podemos ver invoca a la función de alert y luego cancela el evento.

```
$(document).ready(function(evento) {
```

```
$("a").click(function(evento) {

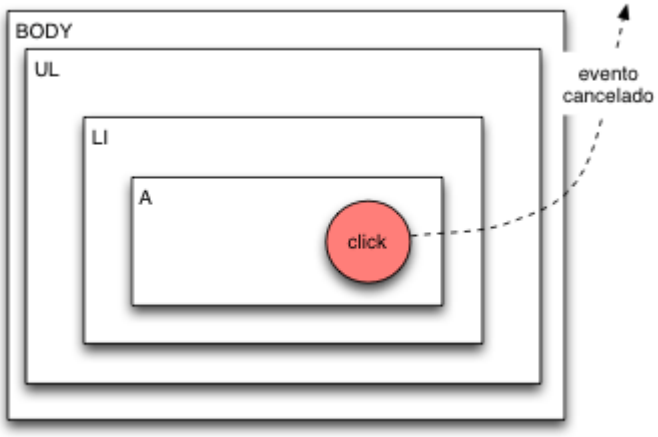
 alert("has pulsado el boton");
 evento.preventDefault();
 });
});
```

Cancelar acción

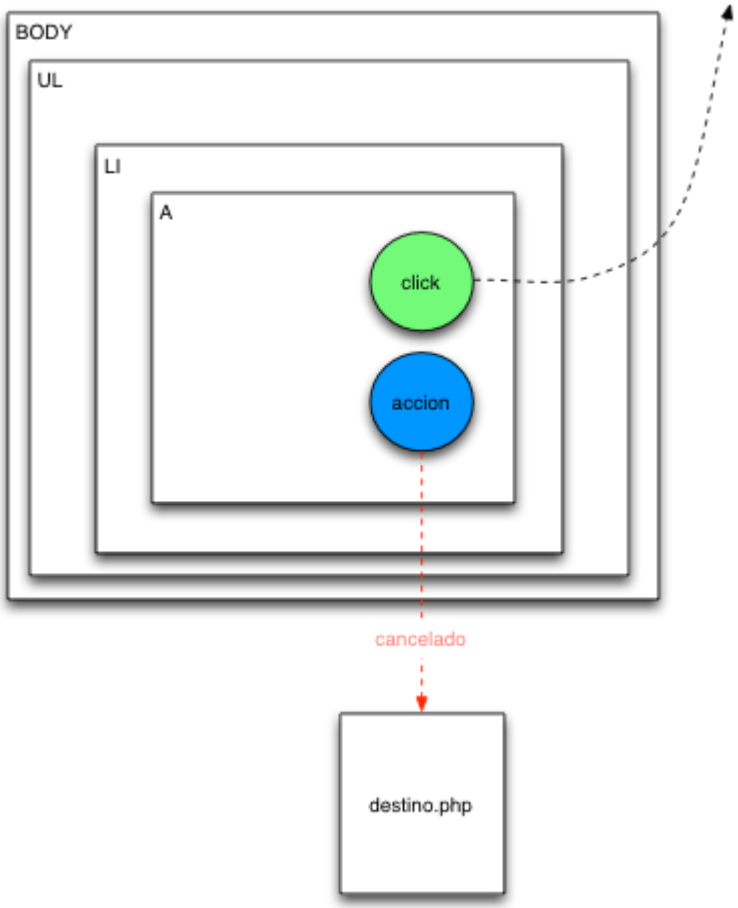
Ahora bien ¿que hace exactamente la linea evento.preventDefault()? . Para la mayor parte de la gente la linea simplemente cancela el evento y al pulsar el enlace no pasamos a la página de destino y simplemente sale el mensaje “has pulsado el boton”.



Así pues esta es la idea que tenemos de lo que sucede .

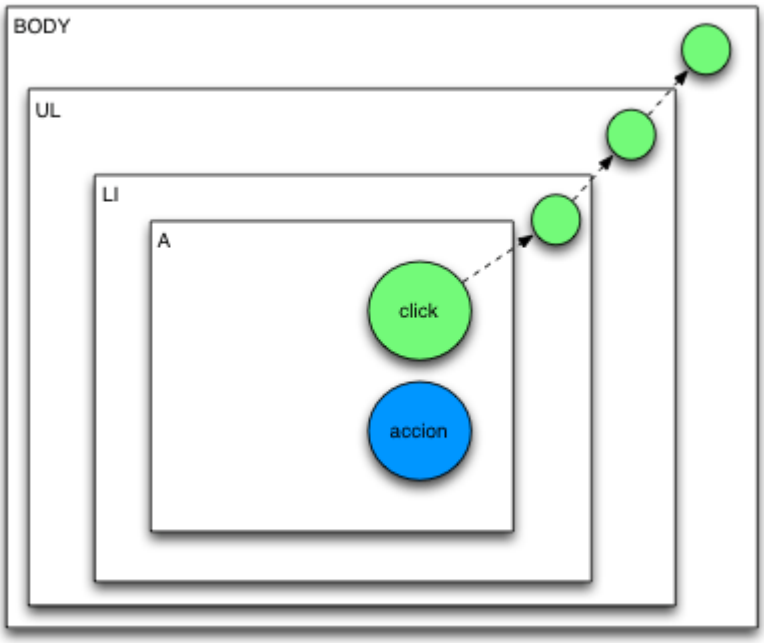


Sin embargo no es del todo correcta ,en la realidad sucede algo distinto . Cuando nosotros pulsamos el boton lanzamos el evento click , pero ademas se ejecuta la acción de salto de pagina al haber pulsado en un link . Es esta acción la que cancelamos cuando usamos el método evento.preventDefault() de JQuery.



Event Bubling

¿Que sucede entonces con el evento que hemos lanzado? . Pues sigue fluyendo hacia arriba por la estructura de etiquetas HTML. Esto nos permitirá capturarlo si fuera necesario a nivel de otra etiqueta.



Vamos a ver un sencillo ejemplo en código de como el evento sube como una burbuja hasta la etiqueta <html> y es capturado por todas.

```
$(document).ready(function(evento) {

		$("a").click(function(evento) {

			alert("has pulsado el boton");
			evento.preventDefault();
		});

		$("li").click(function(evento) {
			console.log("el evento llega al LI ");
		});

		$("ul").click(function(evento) {
```

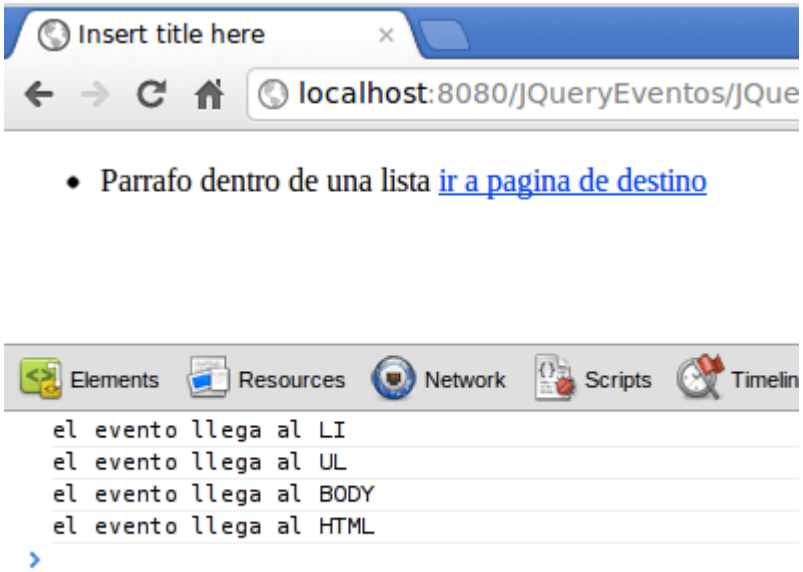
```
                console.log("el evento llega al UL ");
        });

        $("body").click(function(evento) {

                console.log("el evento llega al BODY ");
        });
        $("html").click(function(evento) {

                console.log("el evento llega al HTML ");
        });
});
```

Vemos a continuación como a traves de la consola se imprimen los distintos mensajes de captura del evento en cada una de las etiquetas . Este comportamiento es conocido habituralmente como “event bubbling”.



Cancelar Evento

Si queremos cancelar la propagación del evento podemos definir el código de la siguiente forma

```
$("a").click(function(evento) {

alert("has pulsado el boton");
 return false;
 });
```

La linea de return false cancela tanto la acción de salto de página como el propio evento click .

Cancelar propagación

Puede ser que en algún momento queramos tener un control mas exhaustivo de la gestión de eventos .En ese caso podemos cancelar el bubbling del evento a la altura que necesitemos utilizando el método stopPropagation() .

```
$(document).ready(function(evento) {

$("a").click(function(evento) {

alert("has pulsado el boton");
 evento.preventDefault();
 });

$("li").click(function(evento) {
```

```
 console.log("el evento llega al LI ");
 evento.stopPropagation();
 });

$("ul").click(function(evento) {

console.log("el evento llega al UL ");
 });

$("body").click(function(evento) {

console.log("el evento llega al BODY ");
 });
 $("html").click(function(evento) {

console.log("el evento llega al HTML ");
 });
 });
```

Hemos visto en este articulo las diferencias entre las siguientes opciones a nivel de control de eventos y su propagación.

- return false;
- evento.preventDefault()
- evento.stopPropagation()

Conocer las diferencias de cada uno nos será útil en el trabajo del día a día.