Eventos sobre elementos creados on the fly

Todos conocemos como asignar la detección de un evento en jQuery a un elemento de la página. Simplemente referenciamos el elemento mediante el selector adecuado y le atribuimos el método on(), con el evento que deseamos detectar y la función de CallBack que deseamos que se ejecute como respuesta a dicho evento (si quieres conocer más sobre los posibles selectores para referenciar un elemento, te sugiero este enlace que, no me cabe duda, encontrarás muy útil).
El problema es que sólo podemos detectar los eventos que se producen en elementos que ya existen en la página cuando esta se carga. Si, una vez cargado el DOM, se añaden nuevos elementos, la detección habitual de eventos no funciona. En este artículo vamos a conocer este problema y la forma de resolverlo.

EL PROBLEMA
Bien. El problema es el siguiente. Supongamos que tenemos una página que contiene ciertos elementos. Durante la visita a la misma, y como respuesta a determinadas acciones del usuario, u otro tipo de eventos, se crean nuevos elementos de forma dinámica. Si tratamos de detectar eventos sobre los elementos que existían en la página, no hay problema. Sin embargo, si tratamos de detectar eventos sobre los elementos nuevos, creados dinámicamente después de cargada la página, jQuery parece no reconocerlos, y no detecta eventos. Veamos un ejemplo en no_funciona.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>jQuery</title>
</head>
<body>
 <div id="contenedor_de_botones" style="width: 200px; height: auto; padding: 4px; border: 1px solid black;">
  <input type="button" class="botones" value="botón 1" style="width:100%;">
 </div>
 <script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
 <script type="text/javascript">
  var numero_de_boton = 1;
  $('.botones').on('click', function(){
   numero_de_boton ++;
   var botonNuevo = $('<input type="button" class="botones" value="botón ' + numero_de_boton + '" style="width:100%;">');
   botonNuevo.appendTo($('#contenedor_de_botones'));
  });
 </script>
</body>
</html>
```

Como ves, todo es código es extremadamente simple. Se crea un contenedor en el que se incluye un botón, al que le hemos agregado la clase botones, para poder usarla como selector, es decir, esa clase no aporta ningún estilo específico, sino que sirve sólo para referenciar el elemento. Este documento, según se renderiza a la carga, muestra lo que ves en la imagen situada a la izquierda de este texto. Muy simple, como puedes comprobar.

Si pulsas este botón, tal como se ve al examinar el jQuery de la página, se crea un nuevo botón que se añade al contenedor. Cada vez que lo pulses, se agregará un nuevo botón. Así, si lo pulsas tres veces, se crean tres botones nuevos, quedando con el aspecto que ves a la derecha. Sin embargo, si pulsas alguno de los botones "nuevos", verás que no funciona. El botón original sí añade nuevos botones (jQuery sí detecta el evento, y responde a él), pero los botones nuevos no funcionan. Sin embargo, todo parece estar correcto. Hemos programado el evento sobre la clase botones y los botones nuevos también tienen esa clase, así que, pensamos, también deberían actuar igual.

La raíz del problema está en que, cuando cargamos jQuery, el DOM es parseado, y el framework "sabe" lo que ya hay. Cuando se crea un elemento dinámicamente, no se reparsea el DOM, por lo que, a efectos de jQuery, este elemento no exite. Por lo tanto, no lo reconoce, aunque el selector parezca estar correcto.

LA SOLUCIÓN

Afortunadamente, el método on() nos permite usar otro parámetro, de modo que podamos hacer que se reparsee dinámicamente una parte del documento, de modo que jQuery reconozca los elementos "nuevos" que forman parte de ella. Lo que hacemos es usar cómo selector el padre de los elementos que nos interesan. En este ejemplo, será el contenedor de los botones (el div llamado contenedor_de_botones). En el método on() añadimos el selector por el que nos referimos a los botones (la clase botones), entre el evento (click) y la función de CallBack. El código lo tienes en solucionado.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>jQuery</title>
</head>
<body>
 <div id="contenedor_de_botones" style="width: 200px; height: auto; padding: 4px; border: 1px solid black;">
  <input type="button" class="botones" value="botón 1" style="width:100%;">
 </div>
 <script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
 <script type="text/javascript">
  var numero_de_boton = 1;
  $('#contenedor_de_botones').on('click', '.botones', function(){
   numero_de_boton ++;
   var botonNuevo = $('<input type="button" class="botones" value="botón ' + numero_de_boton + '" style="width:100%;">');
   botonNuevo.appendTo($('#contenedor_de_botones'));
  });
 </script>
</body>
</html>
```

Observa cómo lo hemos hecho en la línea que aparece resaltada. Si pruebas este código, verás que, al pulsar el botón inicial, se crean nuevos botones. Si ahora pulsas cualquiera de ellos, actúan del mismo modo. Es decir, ahora jQuery es capaz de reconocer todos los elementos y reaccionar a sus eventos.

Este es un claro ejemplo de problemas que surgen en el día a día del desarrollo web, y que son mucho más fáciles de solucionar de lo que parece en principio. Los dos listados de este artículo puedes descargarlos en este enlace.