Eventos © ADRINFOR S.L.

www.adrformacion.com © ADRINFOR S.L.
Héctor García González
www.adrformacion.com © ADRINFOR S.L.
Héctor García González
www.adrformacion.com © ADRINFOR S.L.
Héctor García González

Indice

www.adrformacion.com © ADRINFOR S.L. Héctor García González

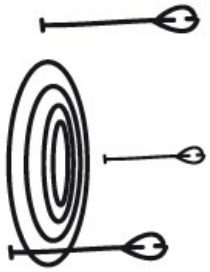
acion.com © ADRINFOR S.L.
a González

Eventos

Introducción

Cuando hablamos de eventos en el entorno de la programación nos referimos a la delegación de ejecución de código cuando una acción, generalmente provocada por una interacción del usuario, ocurre en la página.

jQuery incluye un conjunto de métodos con los que podremos manejar diferentes tipos de eventos del navegador, tales como cambios en los elementos de un formulario, pulsaciones del ratón en un elemento de la página, así como diferentes tipos de acciones que pueden ocurrir en una página.



Un evento puede ser un clic,
mover el ratón, pulsar una tecla...
o que la flecha alcance la diana ;)

De este modo no sólo podremos asignar controladores a eventos sino también controlar el funcionamiento estándar de los mismos e incluso prevenir su ejecución.

Conceptos básicos sobre eventos en jQuery

Objetos de tipo Evento

Cada vez que se ejecuta un evento en jQuery, la función controladora del mismo siempre recibirá un objeto de tipo Evento del que podremos obtener información sobre el mismo.

Propiedades estandarizadas de un objeto de tipo Evento

Tal y cómo hemos mencionado anteriormente, todos los eventos capturados con jQuery obtienen en la llamada al controlador un objeto de tipo Evento, que contiene toda la información relativa al evento ocurrido en la página. Este objeto ofrecerá un amplio conjunto de propiedades, que detallaremos a continuación:

Propiedades generales del evento

www.adrformacion.com © ADRINFOR S.L. Héctor García González

target

Esta propiedad contendrá una referencia al elemento del DOM que inició el evento.

relatedTarget

Contendrá una referencia de otro elemento del DOM que se vea involucrado en el evento.

Esta propiedad tomará valor para los métodos de ratoón mouseout y mouseover, indicando en cada uno de ellos el elemento que recibirá el puntero, en el primer caso, y el elemento que lo perderá en el segundo.

type

La propiedad type contendrá una descripción del tipo de evento: click, dblclick, mouseover, mouseout, change...

currentTarget

La propiedad currentTarget contendrá una referencia al elemento del DOM en el que el evento está "ascendiendo" (bubbling).

Cuando se dice que un evento está "bubbling" se hace referencia al desencadenamiento de los eventos producidos por la ascendencia en el árbol del DOM a los demás elementos de que se a iniciado un evento.

Por ejemplo, si en nuestro código nos encontramos con una etiqueta "a" contenida dentro de un "div", el evento click del “a” ascenderá a través del DOM, propagando a su vez el evento click del "div".

timeStamp

El atributo timestamp contendrá una marca de hora correspondiente a la diferencia de tiempo, en milisegundos, entre el instante de creación del evento y el 1 de enero de 1970.

Este atributo no es válido para el navegador Firefox.

Propiedades de eventos del teclado y del ratón

which

La propiedad which tendrá diferente significado dependiendo del tipo de evento capturado.

Para los eventos de teclado, esta propiedad indicará el código de la tecla presionada.

Por otro lado, para los eventos de ratón, indicará diferentes valores dependiendo del botón del ratón utilizado:

- 1 para el botón izquierdo
- 2 para el botón central
- 3 para el botón derecho.

altKey, ctrlKey, shiftKey

Estas propiedades indicarán, relativamente, si en el momento de capturar el evento, se encontraban pulsadas la teclas del teclado Alt, Ctrl o Shift.

button
Esta propiedad contendrá, para los eventos de ratón, un valor indicativo del botón del ratón utilizado en el evento: • 1 para el botón izquierdo • 2 para el botón central • 3 para el botón derecho. Es recomendable no utilizar esta propiedad, haciendo uso cuando sea posible de la propiedad which.

Propiedades de los objetos de tipo Evento

Captura de Eventos con jQuery

jQuery es una librería muy versátil. Como hemos visto en otras ocasiones, existen diferentes formas de realizar una acción. Del mismo modo, también podemos encontrar varias formas de capturar eventos con jQuery.

El primero de los modos de realizar esta captura es la captura con métodos directos. En este tipo de captura, utilizaremos un método para cada evento, indicando como método el nombre del evento y proporcionándole como parámetro una función anónima, que se ejecutará cuando el evento ocurra.

Otras formas de capturar los eventos es con el uso de métodos de captura. Entre estos, se encuentran los siguientes:

- Método .on()
- Método .one()
- Método .bind()
- Método .live()
- Método .delegate()

Captura de un evento con métodos directos

jQuery dispone de métodos para capturar directamente la mayoría de los eventos disponibles. Estos métodos tienen el nombre del evento que capturan, por ejemplo:

```
 $("a").click(function(e){
  //acciones a realizar
});
```

Con la aplicación del método .click() capturaremos el evento de pulsación sobre los elementos de tipo "a" de la página. Así, cada vez que un elemento "a" sea pulsado, se ejecutarán las acciones definidas en la función proporcionada como parámetro al método.

Los métodos directos podrán funcionar también como desencadenantes del evento, permitiéndonos en cualquier momento poder lanzar la ejecución de cada uno de los eventos que se encuentren capturados. Para que estos métodos tengan ese comportamiento, realizaremos una llamada a los mismos sin proporcionarles un controlador:

```
 $("a").click(function(e){
  //acciones a realizar
});

//ejecutará el evento sobre el primer elemento de tipo a
$("a:first").click();
```

Captura de eventos con el método .on()

El método .on() ofrece una mayor versatilidad que la captura con métodos directos. Las propiedades extra que nos brinda este método son:

- vinculación de múltiples eventos en una única llamada
- opción de proporcionar información adicional al evento

Esta segunda característica nos permitirá personalizar eventos, proveyendo de información adicional al controlador del evento.

Ejemplo: captura del evento click de los elementos a

```
 $("a").on("click", (function(e){
  //acciones a realizar
});
```

Al capturar un evento, la acción indicada será realizada cada vez que éste ocurra.

Veamos otro ejemplo de cómo capturar más de un tipo de evento con un único controlador. En este ejemplo, vamos a capturar los eventos mouseover y mouseout de los elementos a

```
 $("a").on("mouseover mouseout", function(e){
  //acciones a realizar
  console.log(e.type);
});
```

Como comentamos anteriormente, otra forma de llamar al método "on" para la captura de un evento es proporcionándole un parámetro adicional, con el que indicar datos adicionales al controlador. El siguiente fragmento de código capturará el evento click de los elementos a, proporcionando un parámetro adicional al controlador.

```
 $("a").on("click", {extradata: "valor"},function(e){
  //acciones a realizar
  console.log(e.data.extradata);
});
```

En versiones de jQuery anteriores a la 1.7, esta tarea podía realizarse con el método .bind()

```
 $("a").bind("click", function(e){
  //acciones a realizar
});
```

Captura de eventos con el método .one()

El método .one() permite otro tipo de captura de los eventos de la página. Este método realizará la captura del evento una única vez. Es decir, la primera vez que ocurra el evento, las acciones indicadas en el controlador serán ejecutadas, pero no en sucesivas ocurrencias de este evento.

```
 $("a").one("click", function(e){
  //acciones a realizar
  alert("evento capturado una vez con el metodo .one()");
});
```

HTML

Vamos a capturar el evento click de los elementos div con la clase "elem", utilizando el método .one(). De este modo, el evento solamente se capturará una vez sobre cada div.

```
    <div id="contenedor">
      <div class="elem">E1</div>
      <div class="elem">E2</div>
      <div class="elem">E3</div>
      <div class="elem">E4</div>
      <div class="elem">E5</div>
   </div>
```

Código

Así, al ocurrir el evento, el elemento pulsado verá modificado su color de fondo, así como el cursor por defecto. Además, el evento dejará de capturarse, y no volverá a ser ejecutado en siguientes pulsaciones.

```
 <script type="text/javascript">
$(document).ready(function(){
   $("div.elem").one( "click", function() {
      $(this).css({
         background: "red",
         cursor: "auto"
      });
   });
});
</script>
```

Captura de eventos con el método .bind()

El método .bind() permite capturar uno o varios eventos sobre los elementos incluidos en la selección sobre la que se aplique.

Este método fue la forma originaria de capturar los eventos en las primeras versiones de jQuery, siendo utilizado hasta la versión 1.7, donde, a pesar de seguir funcionando, se aconsejó reemplazarla por el método .on().

Ejemplo: captura del evento click de los elementos a

```
 $("a").bind("click", (function(e){
  //acciones a realizar
});
```

Como puedes observar, su sintáxis es idéntica a la del método .on().

Captura de eventos con el método .live()

El método .live() nos permite capturar un evento de todos los elementos que cumplan con el selector sobre el que se captura, tanto de los existentes en un momento dado en el DOM como los futuros que sean añadidos al mismo.

Aunque este método se encuentra marcado como "desaprobado" (deprecated) desde la versión 1.7, es recomendable conocerlo, ya que si nos encontramos con una versión anterior de la librería, puede darse la necesidad de que tengamos que recurrir a este método.

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

  $("a").one("click", function(e){
    //acciones a realizar
    alert("evento capturado una vez con el metodo .one()");
  });

  $("<a href='#'>Enlace</a>").appendTo($("body"));

});
```

Captura de eventos con el método .delegate()

Por medio del método .delegate() podremos vincular un controlador sobre uno o varios eventos.

El comportamiento de este método es algo más complejo que los anteriores:

1. Se aplicará sobre una selección, que definirá el ámbito en el que localizar los elementos finales sobre los que se capturarán los eventos
2. Se indicará como primer parámetro un selector que dentro del ámbito anteriormente establecido, buscará aquellos elementos que cumplan con el mismo y que serán los elementos finales cuyos eventos sean capturados.
3. Se le proporcionará un segundo parámetro indicando el tipo de evento a capturar
4. Por último, el tercer parámetro será una función, que se ejecutará cada vez que se produzca el evento

El uso del método .delegate() está desaconsejado desde la versión 1.7 de jQuery. En su lugar, se recomienda utilizar el método .on() para capturar los eventos sobre los elementos de una selección.

Métodos de captura de eventos

Prevenir comportamiento por defecto del evento

Cabe destacar que capturar un evento no "cancelará" automáticamente el evento genérico del objeto. Por ejemplo, si capturamos el evento "click" de los enlaces de la página, éstos seguirán realizando su función habitual, de acceso a la URL indicada en su propieddad href.

Este comportamiento puede ser desactivado preveniendo el comportamiento por defecto de los controles. Disponemos de dos métodos para prevenir este comportamiento:

event.preventDefault()

Llamando al método preventDefault() del evento proporcionado en la función

```
 $("a").on("click", (function(e){
  //desactivar funcionalidad por defecto del evento
  e.preventDefault();

  //realizar las acciones deseadas en el evento
});
```

Devolviendo false en la función controladora

Forzando a que la función controladora devuelva el valor false

```
 $("a").on("click", (function(e){
  //realizar las acciones deseadas en el evento
  //desactivar funcionalidad por defecto del evento
  return false;
});
```

A diferencia del método event.preventDefault(), el método return false; evitará que el evento sea propagado por el resto de elementos padres del DOM. Para conseguir la misma funcionalidad, además de llamar a event.preventDefault() deberemos llamar al método event.stopPropagation()

Eventos disponibles en jQuery

Para estudiar los diferentes eventos que podemos capturar con jQuery vamos a dividirlos en cuatro subgrupos según la naturaleza de los mismos:

- Eventos del ratón
- Eventos del teclado
- Eventos de elementos de formularios
- Eventos del navegador

Eventos del ratón

Los eventos del ratón son aquellos que, los visitantes de una página, disparan a partir de su interacción en el sitio por medio del ratón.

Evento click

El evento click se disparará cuando los visitantes del sitio realicen un clic del ratón sobre los elementos que tengan "capturado" este evento.

Evento dblclick

El evento dblclick de un elemento será disparado cuando un elemento reciba un doble ckic (dos clics consecutivos).

HTML

Vamos a capturar el evento dblclick del botón con identificador "idDbl".

```
<div id="contenedor">
  <div class="elem">E1</div>
  <input type="button" id="idDbl">
</div>
```

www.adrformacion.com © ADRINFOR S.L. Héctor García González

Código

Cuando pulsemos dos clics consecutivos sobre el botón, los elementos de tipo "div" con la clase "elem" modificarán su asignación de la clase "dblClicked", añadiéndose cuando no la tengan asignada, o quitándose cuando sí la tengan asignada.

```
 <script type="text/javascript">
$(document).ready(function(){
    $("#idDbl").on( "dblClick", function() {
        $("div.elem").toggleClass("dblClicked");
    });
});
</script>
```

Evento mousedown

Este tipo de evento permitirá capturar la acción de presionar el botón de teclado, sin necesidad de que éste sea soltado. En otras palabras, el evento mousedown captura la "primera parte" del evento click.

Este evento registrará el evento de ambos botones del ratón, pudiendo diferenciar cuál de ellos provoca el evento analizando el atributo .which del objeto de tipo evento que se proporcione a la función controladora del evento.

Evento mouseup

El evento mouseup se ejecutará cuando, una vez pulsado un botón del ratón, el usuario decida "soltar" este botón.

HTML

Vamos a capturar los eventos "mouseup" y "mousedown" del botón con identificador "idDbl".

```
    <div id="spoiler" data-oculto="contenido oculto"></div>
   <div id="btnMostrar">Mostrar</div>
```

www.adrformacion.com © ADRINFOR S.L. Héctor García González

Código

Al presionar el botón #btnMostrar el div#spoiler mostrará el texto asignado en el atributo data-title. Además, al "soltar" el botón, el texto desaparecerá.

```
 <script type="text/javascript">
$(document).ready(function(){
    $("#btnMostrar").on( "mousedown", function() {
        $("#spoiler").text( $("#spoiler").attr("data-oculto") );
    })
    .on( "mouseup", function() {
        $("#spoiler").text("");
    });
});
</script>
```

www.adrformacion.com © ADRINFOR S.L. Héctor García González

Evento mouseenter

El evento de ratón mouseenter se producirá cuando el puntero del ratón se sitúe encima de un elemento de la página.

Evento mouseleave

Este evento se desencadenará cuando el puntero del ratón abandone el área ocupada por un elemento de la página.

Evento mousemove

El evento mousemove se registrará cuando el puntero del ratón se mueva sobre un elemento de la página.

Evento mouseout

El evento mouseout se producirá cuando el puntero del ratón abandone el área de un elemento. Este evento también se ejecutará cuando el puntero del ratón se mueva "fuera" de un elemento hijo.

Evento mouseover

El evento mouseover se producirá cuando el puntero del ratón entre dentro del área de un elemento. Del mismo modo que el evento mouseout, este evento también se producirá cuando el puntero se posicione sobre cualquiera de los descendientes del elemento del cual se capture el evento.

Método hover()

El método .hover() nos proporciona una forma abverviada de capturar los eventos mouseenter y mouseleave conjuntamente.

Este método recibirá como parámetros dos funciones anónimas:

- La primera de ellas, será ejecutada como si del evento mouseenter se tratase

- La segunda, se ejecutará como acción al evento mouseleave del objeto.

Esta función en realidad sirve para manejar dos eventos, cuando el ratón entra y sale de encima de un elemento. Por tanto espera recibir dos funciones en vez de una que se envía a la mayoría de los eventos.

HTML

Vamos a capturar los eventos "mouseup" y "mousedown" del botón con identificador "idDbl".

```
<div id="hoverDiv" >Antes de ponerse el rat&oacute;n encima</div>
```

Código

Al presionar el botón #btnMostrar el div#spoiler mostrará el texto asignado en el atributo data-title. Además, al "soltar" el botón, el texto desaparecerá.

```
 <script type="text/javascript">
$(document).ready(function(){
   $("#hoverDiv").hover( function() {
      $("#hoverDiv").text( "Despues de poner el rat&oacute;n encima" );
   },
   function() {
      $("#hoverDiv").text( "Antes de poner el rat&oacute;n encima" );
   });
});
</script>
```

www.adrformacion.com © ADRINFOR S.L. Héctor García González

Método toggle()

El método .toggle() permite indicar dos o más funciones que serán ejecutadas alternandose para cada clic que el usuario realice sobre un objeto.

Eventos del teclado

En este grupo de eventos, encontramos aquellos que tienen una relación principal con las interacciones que un usuario realiza por medio del teclado.

Evento keydown

El evento keydown se producirá en el momento en que se pulsa una tecla del teclado, sin necesidad de que ésta sea liberada.

Este evento se producirá en el momento justo en el que la tecla es presionada.

HTML

Vamos a limitar las pulsaciones permitiendo únicamente introducir números.

```
 Introduce un n&uacute;mero: <input type="text" id="inputNumber">
<p>Tecla pulsada: <span id="teclaPulsada"></span></p>
```

Código

Al presionar una tecla cuyo código no esté entre el 96 y el 105, prevenir la ejecución del evento.

```
 <script type="text/javascript">
$(document).ready(function(){
    $("#inputNumber").on("keydown",function(e){
        var code = e.keyCode || e.which;
        $("#teclaPulsada").text(code);
        //solamente permitir teclado numerico
        if(code < 96 || code > 105 ) {
            e.preventDefault();
            return false;
        }
        else{
            return true;
        }

    });
});
</script>
```

Evento keypress

Este evento se producirá cuando se realice el juego completo de pulsación: pulsar una tecla y "soltarla" o, dicho con otras palabras, como respuesta a una pulsación y la inmediata liberación de la misma.

Cabe tener en cuenta que también se producirá el evento si mantenemos la teclapulsada.

Evento keyup

El evento keyup se ejecutará en el momento en el que se deje de presionar una tecla que se encontraba pulsada.

Eventos de elementos de formularios

Este bloque de eventos contiene aquellos eventos relacionados principalmente con los elementos de formularios, como campos de entrada (input), selectores (select) e incluso los propios formularios (form).

Evento change

El evento change se producirá cuando el valor de un elemento sea modificado. Este evento se podrá capturar para los elementos de tipo input, select y textarea.

Para los elementos input de tipo checkbox y radio, y los elementos select, este evento se ejecutará inmediatamente después de que el usuario de la página realice un cambio en los mismos. Para otros elementos, como elementos input de tipo texto, el evento se producirá cuando éstos pierdan el foco.

Evento blur

El evento blur será capturado cuando un elemento pierda el foco o, dicho de otro modo, pierda el estado de activación.

El foco en informática se refiere a cuál de las ventanas o componentes gráficos de un escritorio (botones de comando, casillas de verificación, cuadros de texto, etc.) están en ese momento activos (a la escucha de eventos, tales como los provenientes del teclado o el ratón).

Fuente: Wikipedia

Originalmente este evento estaba únicamente orientado a elementos de formularios (por ello se categoríza en este grupo), pero podrá ser capturado para cualquier elemento del DOM.

Evento select

El evento select controlará cuándo, los usuarios de un sitio, realicen una selección del texto de un elemento.

Este evento se podrá capturar para elementos input de tipo "text" y elementos textarea.

Evento submit

El evento submit capturará la acción de envío de datos de un formulario. El evento submit podrá ser capturado únicamente para elementos de tipo formulario.

Para conseguir que un formulario lance esta acción, deberá ocurrir alguna de las siguientes acciones:

- El usuario pulsa sobre un elemento input de tipo "submit"
- El usuario pulsa sobre un elemento input de tipo "image"
- El usuario pulsa la tecla "Intro" del ratón teniendo activo un elemento de formulario (que no sea de tipo textarea)
- Un script provoca el envío del formulario.

Evento focusin

El evento focusin se producirá cuando un elemento o cualquiera de sus descendientes obtengan el foco (sean activados).

Evento focusout

El evento focusout ocurrirá siempre que un elemento del DOM pierda el foco activo. Del mismo modo que ocurría con el evento focusin, este evento también ocurrirá cuando uno de los descendientes del elementos en el que el evento se encuentra capturado pierdan el foco.

Evento focus

Por último dentro de los eventos de elementos de fomulario encontramos el evento focus. Este evento ocurrirá cuando el elemento en el que esté capturado el evento pierda el foco activo.

Este elemento se diferencia del evento focusin en que el evento focusin también se produce cuando un descendiente del elemento sobre el que se captura el evento recibe el foco.

Eventos del navegador

En este grupo de eventos, vamos a encontrar aquellos eventos que, pudiendo ser provocados bien por el propio funcionamiento del navegador o por acciones realizadas por los usuarios sobre el mismo, controlan aspectos que tienen una amplia vinculación con el funcionamiento del navegador.

Evento ready

Como hemos visto en anteriores ocasiones, el evento ready se aplica sobre el documento del sitio, capturando el momento exacto en el que el DOM de la página se encuentra preparado.

Este evento solamente podrá capturarse sobre el selector $(document), aunque también podrá hacerse referencia al mismo con la sintáxis $(), dado que siempre actuará sobre el objeto document de la página.

Evento resize

El evento resize, que se aplicará sobre el selector $(window), capturará el momento en el que el tamaño de la ventana sea modificado.

Dependiendo del navegador desde el que un usuario visualice la página en la que se capture este evento, el momento de "lanzamiento" del evento puede verse alterado.

Los navegadores Internet Explorer y aquellos basados en el motor WebKit, lanzarán este evento varias veces durante el tiempo en el que la ventana es redimensionada, mientras que en otros navegadores, como Opera, el evento solamente se lanzará una vez cuando el redimensionado de la página se de por finalizado.

Evento scroll

El evento scroll se producirá sobre los elementos del DOM cuando el usuario realiza scroll sobre los mismos (uso de la barra lateral para subir / bajar en el contenido del elemento).

Este evento se podrá capturar sobre el objeto window de la página, así como en otros elementos del DOM que dispongan del control de desplazamiento (generalmente, elementos con la propiedad de CSS overflow establecida con el valor scroll o auto.

Evento unload

El evento unload controlará el momento en que los usuarios de la página abandonen la misma por diversas razones:

- El usuario pulsa un enlace de la página, abandonando así la página original
- El usuario indica una nueva dirección en la barra de direcciones del navegador
- El usuario pulsa un de los botones "Atras" o "Adelante" en el navegador (o cualquier otra utilidad del mismo que haga "abandonar" la página actual)
- El usuario cierra la ventana o pestaña del navegador.

Este tipo de evento siempre será capturado sobre el selector $(window).

El desencadenamiento del evento unload puede ser diferente dependiendo del navegador y las versiones de los mismos, ya que en algunos de ellos se lanzarán cuando el usuario realice una de las acciones anteriormente mencionadas mientras que en otros pueden ocurrir cuando los elementos del DOM que componen la página haya sido destruidos.

Cancelación de eventos

Hasta ahora hemos visto como capturar los diferentes tipos de eventos ocurridos sobre la página y sus elementos. Pero, llegado un momento, puede surgirnos la necesidad de dejar de capturar un evento. Es en este momento en el que deberemos de calcelar la captura del evento, para que cuando éste ocurra, no se ejecute ninguna acción.

Método .die()

El método .die() desvinculará la captura de eventos que hayan sido capturados por medio del método .live() de los elementos existentes en la selección sobre la que se aplique.

Cuando llamemos al método .die() podremos hacerlo de dos modos:

- Indicando con un parámetro el tipo de evento a desasignar.
- Sin indicar ningún parámetro, removiendo así todos los eventos capturados por el método .live() sobre los elementos de la selección.

Método .unbind()

Con el método .unbind() podremos desasignar la captura de los eventos capturados con el método .bind()

Del mismo modo que el método .die() este método podrá aceptar un parámetro con el que indicar el evento exacto que deseemos desasignar. En caso de no indicar ester parámetro, todos los eventos que hubieran sido capturados con el método .bind() dejarán de encontrarse capturados a la espera de que una acción del usuario o del navegador "dispare" estos eventos.

Método .undelegate()

Otra forma de desvincular la captura de eventos es por medio del método .undelegate().

Este método permitirá desactivar la captura de eventos que se haya realizado con el método .delegate(). Su funcionamiento será similar al método anteriormente mencionado:

- Se aplica sobre una selección, que definirá el ámbito en el que localizar los elementos finales sobre los que desvincular la captura de eventos.
- Se indicará como primer parámetro un selector que, dentro del ámbito anteriormente establecido, buscará aquellos elementos que cumplan con el mismo y que serán los elementos finales sobre los que desvincular la captura.
- Opcionalmente, se le proporcionará un segundo parámetro indicando el tipo de evento a desvincular

Otra forma de llamar a este método es indicando únicamente el evento a desvincular. En este caso, el parámetro proporcionado hará referencia al evento que se desea desvincular, indicado por medio del nombre del mismo.

También podrá llamarse al método sin ningún parámetro. De este modo, todos los eventos que hubieran sido capturados con el método .delegate() en la selección sobre la que se aplique dejarán de capturarse.

Método .off()

Con el método .off() podremos remover la captura de eventos que se hubieran capturado por medio del método .on()

Al llamar al método .off() sin ningún argumento, conseguiremos que todos los eventos capturados sobre los elementos existentes en la selección sean removidos.

También se podrá proporcionar al método un parámetro con el que indicar el evento exacto del que se desee eliminar la captura.

jQuery.off(nombreEvento, selector)

Una funcionalidad más específica de este método es indicándole dos parámetros:

- El primero será el evento que deseemos remover.
- El segundo parámetro deberá ser un selector, que, en caso de ser exactamente el selector usado en el método .on() al capturar el evento sobre alguno de los elementos de la selección, eliminará la captura del evento sobre los mismos. Si queremos que se elimine la captura para todos los elementos, indicaremos la cadena "**".

Forzado de eventos

En un momento dado, puede ser necesario "forzar" un evento a partir de un comando. Al igual que ocurría con la captura de eventos, el forzado de los mismos también puede realizarse de dos modos diferentes:

- Con métodos directos
- Por medio del método .trigger()

Métodos directos

Los métodos directos, que anteriormente han sido vistos para capturar un tipo de evento sobre una selección, llevan el mismo nombre del evento a capturar o, en este caso, a ser forzado. Para forzar un evento por medio de este modo, haremos una llamada al método sin indicarle ningún parámetro.

Por ejemplo, para lanzar el evento click sobre un elemento cuyo identificador tenga el valor "inputText23" escribiremos el siguiente comando:

```
 $("#inputText23").on("click",function(e){
   alert($(this).val());
});

$("#inputText23").click();
```

De este modo, la función que se encuentra establecida como controladora del evento click será automáticamente ejecutada, mostrando una alerta con el valor del elemento.

Método .trigger()

Por medio del método .trigger(), podremos "disparar" la ejecución de un evento sobre algún elemento, sin que haya sido necesario que un usuario haya realizado ninguna acción sobre el mismo.

El método .trigger() podrá recibir dos argumentos:

Primer Parámetro

El primero de ellos, indicará al método el tipo de evento que se debe disparar.
 	
`$("#btnCalcular").on('click', function(e){`
`  alert($(this).attr('id'));`
`});`
 	
`$("#btnCalcular").trigger('click');`

www.adrformacion.com © ADRINFOR S.L. Héctor García González

Segundo Parámetro (opcional)

El segundo parámetro será opcional. Este parámetro será un array de valores, que podrán ser recogidos en la función controladora del mismo como parámetros de llamada a la función posteriores al objeto de tipo evento disponible en la misma .

```
 $("#btnCalcular").on('click', function(e, arg1, arg2){
   alert($(this).attr('id')); //mostrará la alerta "btnCalcular"
   alert(arg1); //mostrará la alerta "extraParam1"
   alert(arg2); //mostrará la alerta "extraParam2"
});

$("#btnCalcular").trigger('click', ['extraParam1','extraParam2']);
```

Creación de eventos personalizados

Además de los eventos básicos existentes jQuery permite definir eventos personalizados con los que podremos definir nuestros propios eventos, sin vinculación con una interacción del usuario sobre los elementos del DOM de la página.

A primera vista, los eventos personalizados pueden parecer innecesarios, ya que con los eventos básicos nos dan suficiente juego para poder cumplir con nuestras necesidades.

Los eventos personalizados permiten utilizar otro modo de programar, centrando la atención sobre el elemento en el que se van a realizar ciertas acciones. Al no estar directamente vinculados con una acción del usuario, los eventos personalizados se ejecutarán por medio del método .trigger() con el que podremos lanzar nuestros nuevos eventos personalizados.

Al centrar la atención sobre un elemento concreto, podremos aprovechar los siguientes beneficios:

- los eventos del elemento objetivo pueden ser lanzados por diferentes elementos reutilizando código.
- los eventos pueden ser lanzados desde cualquier otro elemento del DOM.
- los eventos se vinculan directamente con el elemento objetivo, consiguiendo así un código más entendible.

En la creación de eventos personalizados utilizaremos dos métodos vistos anteriormente: el método .on() y el método .trigger()

Método .on()

El método .on() recibirá como argumentos un tipo de evento y una función controladora del evento.

Método .trigger()

El método .trigger() recibirá como argumentos el tipo de evento (requerido) y, opcionalmente, un array con valores que serán pasados a la función controladora del eventos como argumentos adicionales.

Ejemplo:

```
 <div id="fichaPersona" id="fichaUsuario">
   <div class="infoPersonal"></div>
   <div class="imageContainer"></div>
   <div class="contactDetails"></div>
</div>

<input id="fichaRellenar1" value="Cumplimentar Ficha" />
```

```
 $("#fichaPersona").on("cumplimentar", function(e, infoPersonal, imageSRC, contactEmail,contactTlf){
   $(".infoPersonal", $(this)).html(infoPersonal);
   if(imageSRC){
      $newImage=$("<img />").attr("src", imageSRC);
      $(".imageContainer", $(this)).append($newImage);
   }

   if(contactEmail){
      $newLinkEmail=$("<a />", {'href': 'mailto:'+contactEmail, 'target':'_blank'}).text("Email: "+contactEmail);
      $(".contactDetails", $(this)).append($newLinkEmail);
   }

   if(contactTlf){
      $newLinkCallTo=$("<a />", {'href': 'callto:'+contactTlf, 'target':'_blank'}).text("Tlf: "+contactTlf);
      $(".contactDetails", $(this)).append($newLinkCallTo);
   }
});

 $("#fichaRellenar1").on("click", function(e){
   $("#fichaPersona").trigger("cumplimentar", ["John Doe","http://lorempixel.com/180/220/people/Aprendiendo-jQu
ery/","sample@mail.com","555959595"]);
});
```

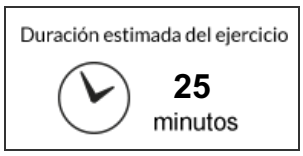
A la hora de programar eventos personalizados, puede ser de utilidad observar los mismos como si de métodos de los elementos del DOM se trataran. En esta similitud, el evento funcionaría como un método disponible para los elementos de la selección sobre la que se capture el evento (con el método de jQuery ,on()).

En nuestro ejemplo, el evento cumplimentar podría asimilarse con la existencia de dicho método en el elemento #fichaPersona.

www.adrformacion.com © ADRINFOR S.L. Héctor García González

Ejercicios

Ejercicio 1: Eventos del ratón



www.adrformacion.com © ADRINFOR S.L.
Héctor García González

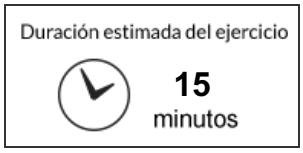
Crea un ejemplo básico de funcionamiento de tabs (fichas de contenido). Al abrer la página, solamente la primera pestaña será visible. Cuando se pulse sobre otro de los títulos de las pestañas, la anterior será modificaca, aplicándole el estilo css "display: none;" y, aquel que se corresponda con el pulsado, mostrado.

Utiliza como contenido de la página el siguiente HTML:

```
 <div id="tabs">
   <div id="titulos">
     <div class="tituloSelected">titulo A</div>
     <div>titulo B</div>
     <div>titulo C</div>
   </div>
   <div id="contenidos">
     <div class="contenidoSelected">Contenido A</div>
     <div style="display: none;">Contenido B</div>
     <div style="display: none;">Contenido C</div>
   </div>
</div>
```

www.adrformacion.com © ADRINFOR S.L.
Héctor García González

Ejercicio 2: Eventos del teclado



www.adrformacion.com © ADRINFOR S.L.
Héctor García González

Disponiendo de un formulario en HTML, utiliza los eventos del teclado de jQuery para evitar que se utilicen los atajosde teclado:

- Cortar - Ctrl + X
- Copiar - Ctrl + C
- Pegar - Ctrl + V

Debes deshabilitar esta función en todos los campos que permitan la escritura por teclado. El HTML del formulario es el siguiente:

www.adrformacion.com © ADRINFOR S.L.
Héctor García González

```
<form id="formRegistro" name="formRegistro" method="post" action="#">
  <p>Nombre <input name="nombre" id="nombre" type="text" /></p>
  <p>Apellidos <input name="apellidos" id="apellidos" type="text" /></p>
  <p>Email <input name="email" id="email" type="text" /></p>
  <p>Direcci&oacute;n <input name="direccion" id="direccion" type="text" /></p>
  <p>Calle <input name="calle" id="calle" type="text" /></p>
  <p>N&uacute;mero <input name="numero" id="numero" type="text" /></p>
  <p>Edad <input name="edad" id="edad" type="text" /></p>
  <p>
    Pa&iacute;s 
    <select name="pais" id="pais">
      <option selected="selected" value="ES">Espa&ntilde;a</option>
      <option value="FR">Francia</option>
      <option value="GE">Alemania</option>
      <option value="PT">Portugal</option>
    </select>
  </p>
  <p>G&eacute;nero <label><input name="genero" id="genero_h" type="radio" value="Hombre" /> 
Hombre</label> <label><input name="genero" type="radio" value="Mujer" />Mujer</label></p>
  <p>Comentarios</p>
  <p><textarea cols="60" name="comentarios" id="comentarios" rows="4"></textarea></p>
  <p>Contrase&ntilde;a  <input type="password" name="contrasena" id="contrasena" /></p>
  <p>Repetir Contrase&ntilde;a <input type="password" name="repetir_contrasena" id="repetir_contrasena" /></
p>
</form>
```

www.adrformacion.com © ADRINFOR S.L. Héctor García González