JQuery

eventos

.click()

El mas común de todos es `.click()` este registra el comportamiento del clic del mouse sobre el elemento seleccionado, su sintaxis:

Ejemplo
Este ejemplo es bastante sencillo, es un boton que cuando se le haga clic se escondera en 500 milisegundos.

```
<style>
*{font-family:sans-serif;}
button{padding:10px 20px;font-size:14px;}
</style>

<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js"></script>
<script>
$(document).ready(function () {

    $('#ejemplo1').click(function () {
        $(this).hide(500); // 'this' es un selector que se usa para seleccionar el elemento ya seleccionado anteriormente, en este caso #ejemplo1
    });

});
</script>

<button id="ejemplo1">Hola</button>
```

.hover()

.hover() es el evento que registra cuando el cursor pasa por encima del elemento seleccionado, su sintaxis es:

```
$('elemento seleccionado').hover(function () {

    // Que hacer cuando el cursor pase sobre elemento

});
```

A este evento también se le puede definir que hacer cuando el cursor deje de pasar sobre el elemento seleccionado, en tal caso su sintaxis cambiaria a:

```
$('elemento seleccionado').hover(function () {

    // Que hacer cuando el cursor pase sobre el elemento

}, function (){

    // Que hacer cuando el cursor deje de pasar sobre el elemento

});
```

Ejemplo
Haremos un ejemplo usando la segunda sintaxis:

```
<style>
*{font-family:sans-serif;}
button{padding:10px 20px;font-size:14px;position:relative;}
.acerca{
width:400px;
font-size:16px;
}
</style>

<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js"></script>
<script>
$(document).ready(function () {

   $('#ejemplo2').hover(function () {
      $(this).animate( { opacity: 0.3 }, 400)
   }, function () {
```

```
        $(this).animate( { opacity: 1.0 }, 400)
    });

});
</script>

<button id="ejemplo2">Hola</button>
```

El resultado es un botón que se aclara cuando el cursor pasa por encima pero regresa a su estado original cuando el cursor deja de pasar por el botón.

En este ejemplo usamos el método `.animate()` que realiza una animación usando propiedades CSS (las que van entre corchetes), luego de la coma, le defnimos en milisegundos el tiempo que queremos que dure la animación.

.toggle()

El evento `.toggle()` es parecido a `.click()` registra el comportamiento de un click, solo que este actúa como un interruptor con dos estados: 'prendido' o 'apagado', su sintaxis:

```
$('elemento seleccionado').toggle(function () {

    // Que hacer cuando se haga clic sobre el elemento

},function () {

    // Que hacer cuando se haga clic otra vez sobre el elemento

});
```

Ejemplo

Haremos un ejemplo sencillo con `.toggle()`.

```
<style>
*{font-family:sans-serif;}
```

```
button{padding:10px 20px;font-size:14px;position:relative;}
</style>
<script type="text/javascript" src="https://ajax.googleapis.com/ajax/
libs/jquery/1.4.4/jquery.min.js"></script>
<script>
$(document).ready(function () {

    $('#ejemplo3').toggle(function () {

        $(this).animate({left:'200px'}, 300)

    }, function () {

        $(this).animate({left:'0px'}, 300)

    });

});
</script>

<button id="ejemplo3">Hola</button>
```

El resultado es un botón que cuando se le haga click se moverá 200px a la derecha y cuando se le haga click otra vez regresara a su estado original y así sucesivamente.
Hay que notar que donde se declaran los estilos (linea 3) el elemento a animar tiene que estar en`position:relative` para que la propiedad left funcione.

Vincular Eventos a Elementos

jQuery ofrece métodos para la mayoría de los eventos — entre ellos $.fn.click, $.fn.focus, $.fn.blur,$.fn.change, etc. Estos últimos son formas reducidas del método $.fn.bind de jQuery. El método bind es útil para vincular (en inglés *binding*) la misma función de controlador a múltiples eventos, para cuando se desea proveer información al controlador de evento, cuando se está trabajando con eventos personalizados ó cuando se desea pasar un objeto a múltiples eventos y controladores.

Vincular un evento utilizando el método $.fn.bind method

```
$('p').bind('click', function() {
     console.log('click');
});
```

Vincular un evento utilizando el método $.fn.bind con información asociada

```
$('input').bind(
    'click change',  // es posible vincular múltiples eventos al
elemento
    { foo : 'bar' }, // se debe pasar la información asociada como
argumento

    function(eventObject) {
        console.log(eventObject.type, eventObject.data);
        // registra el tipo de evento y la información asociada {
foo : 'bar' }
    }
);
```

Vincular Eventos para Ejecutar una vez
A veces puede necesitar que un controlador particular se ejecute solo una vez — y después de eso, necesite que ninguno más se ejecute, o que se ejecute otro diferente. Para este propósito jQuery provee el método $.fn.one.

Cambiar controladores utilizando el método $.fn.one

```
$('p').one('click', function() {
    console.log('Se clickeó al elemento por primera vez');
    $(this).click(function() { console.log('Se ha clickeado nuevamente'); }
); });
```

El método $.fn.one es útil para situaciones en que necesita ejecutar cierto código la primera vez que ocurre un evento en un elemento, pero no en los eventos sucesivos.
Desvincular Eventos
Para desvincular (en ingles *unbind*) un controlador de evento, puede utilizar el método $.fn.unbind pasándole el tipo de evento a desconectar. Si se pasó como adjunto al evento una función nombrada, es posible aislar la desconexión de dicha función pasándola como segundo argumento.

Desvincular todos los controladores del evento click en una selección

```
$('p').unbind('click');
```

```
Desvincular un controlador particular del evento click
```

```
var foo = function() { console.log('foo'); };
var bar = function() { console.log('bar'); };

$('p').bind('click', foo).bind('click', bar);
$('p').unbind('click', bar); // foo esta atado aún al evento click
```

Vinculación de Múltiples Eventos

Muy a menudo, elementos en una aplicación estarán vinculados a múltiples eventos, cada uno con una función diferente. En estos casos, es posible pasar un objeto dentro de $.fn.bind con uno o más pares de nombres claves/valores. Cada nombre clave será el nombre del evento mientras que cada valor será la función a ejecutar cuando ocurra el evento.

Vincular múltiples eventos a un elemento

```
$('p').bind({
      'click': function() { console.log('clickeado'); },
      'mouseover': function() { console.log('sobrepasado'); }
});
```

El Objeto del Evento

Como se menciona en la introducción, la función controladora de eventos recibe un objeto del evento, el cual contiene varios métodos y propiedades. El objeto es comúnmente utilizado para prevenir la acción predeterminada del evento a través del método *preventDefault*. Sin embargo, también contiene varias propiedades y métodos útiles:
pageX, pageY
La posición del puntero del ratón en el momento que el evento ocurrió, relativo a las zonas superiores e izquierda de la página.
type
El tipo de evento (por ejemplo "click").
which
El botón o tecla presionada.
data
Alguna información pasada cuando el evento es ejecutado.
target
El elemento DOM que inicializó el evento.
preventDefault()
Cancela la acción predeterminada del evento (por ejemplo: seguir un enlace).
stopPropagation()
Detiene la propagación del evento sobre otros elementos.
Por otro lado, la función controladora también tiene acceso al elemento DOM que inicializó el

evento a través de la palabra clave this. Para convertir a dicho elemento DOM en un objeto jQuery (y poder utilizar los métodos de la biblioteca) es necesario escribir $(this), como se muestra a continuación:

var $this = $(this);

Cancelar que al hacer click en un enlace, éste se siga

```
$('a').click(function(e) {
    var $this = $(this);
    if ($this.attr('href').match('evil')) {
        e.preventDefault();
        $this.addClass('evil');
    } });
```

Ejecución automática de Controladores de Eventos

A través del método $.fn.trigger, jQuery provee una manera de disparar controladores de eventos sobre algún elemento sin requerir la acción del usuario. Si bien este método tiene sus usos, no debería ser utilizado para simplemente llamar a una función que pueda ser ejecutada con un click del usuario. En su lugar, debería guardar la función que se necesita llamar en una variable, y luego pasar el nombre de la variable cuando realiza el vinculo (*binding*). De esta forma, podrá llamar a la función cuando lo desee en lugar de ejecutar $.fn.trigger.

Disparar un controlador de eventos de la forma correcta

```
var foo = function(e) {
    if (e) {
        console.log(e);
    } else {
        console.log('esta ejecucción no provino desde un evento');
    } };


$('p').click(foo);

foo(); // en lugar de realizar $('p').trigger('click')
```