1 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 o cuando se desea pasar un objeto a múltiples eventos y controladores.

Vincular un evento utilizando un método reducido

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

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 incular 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' }
} );
```

1.2 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.

1.3 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'); }
});
```

2. Incrementar el rendimiento con la delegación de eventos

Cuando trabaje con jQuery, frecuentemente añadirá nuevos elementos a la página, y cuando lo haga, necesitará vincular eventos a dichos elementos — eventos que ya estaban vinculados a elementos en la página. En lugar de repetir la tarea cada vez que se añade un elemento, es posible utilizar la delegación de eventos para hacerlo. Con ella, podrá enlazar un evento a un elemento contenedor, y luego, cuando el evento ocurra, podrá ver en que elemento sucede. Si todo esto suena complicado, afortunadamente jQuery lo hace fácil a través de los métodos $.fn.live y $.fn.delegate.

La delegación de eventos posee algunos beneficios, incluso si no se tiene pensando añadir más elementos a la página. El tiempo requerido para enlazar controladores de eventos a cientos de elementos no es un trabajo trivial; si posee un gran conjunto de elementos, debería considerar utilizar la delegación de eventos a un elemento contenedor.

Delegar un evento utilizando $.fn.delegate

```
$('#myUnorderedList').delegate('li', 'click', function(e) {
    var $myListItem = $(this);
    // ...
});
```

Delegar un Evento utilizando $.fn.live

```
$('#myUnorderedList li').live('click', function(e) {
    var $myListItem = $(this);
    // ...
});
```

2.1. Desvincular eventos delegados

Si necesita eliminar eventos delegados, no puede hacerlo simplemente desvinculándolos. Para eso, utilice el método $.fn.undelegate para eventos conectados con $.fn.delegate, y $.fn.die para eventos conectados con $.fn.live.

Desvincular eventos delegados

```
$('#myUnorderedList').undelegate('li', 'click');
$('#myUnorderedList li').die('click');
```

3. Funciones auxiliares de eventos

jQuery ofrece dos funciones auxiliares para el trabajo con eventos:

3.1. $.fn.hover

El método $.fn.hover permite pasar una o dos funciones que se ejecutarán cuando los eventos mouseenter y mouseleave ocurran en el elemento seleccionado. Si se pasa una sola función, está será ejecutada en ambos eventos; en cambio si se pasan dos, la primera será ejecutada cuando

ocurra el evento mouseenter, mientras que la segunda será ejecutada cuando ocurra mouseleave.

Nota A partir de la versión 1.4 de jQuery, el método requiere obligatoriamente dos funciones.

La función auxiliar hover

```
$('#menu li').hover(function() {
    $(this).toggleClass('hover');
});
```

3.2. $.fn.toggle

Al igual que el método anterior, $.fn.toggle recibe dos o más funciones; cada vez que un evento ocurre, la función siguiente en la lista se ejecutará. Generalmente, $.fn.toggle es utilizada con solo dos funciones. En caso que utiliza más de dos funciones, tenga cuidado, ya que puede ser dificultar la depuración del código.

La función auxiliar toggle

```
$('p.expander').toggle(
    function() {
        $(this).prev().addClass('open');
    },
    function() {
        $(this).prev().removeClass('open');
    }
);
```