Unidad 6. Integración de contenido interactivo.

jQuery: Gestión de eventos

Tabla de contenido

GESTIÓN DE EVENTOS

jQuery provee métodos para asociar **controladores de eventos** (en inglés event handlers) a selectores.

Cuando un evento ocurre, la función provista es ejecutada. Dentro de la función, la palabra clave **this** hace referencia al elemento en que el evento ocurre.

Para más detalles sobre los eventos en jQuery, puede consultar http://api.jquery.com/category/events/.

La función del controlador de eventos puede recibir un **objeto**. Este objeto puede ser utilizado para determinar la naturaleza del evento o, por ejemplo, prevenir el comportamiento predeterminado de éste. Para más detalles sobre el objeto del evento, visite http://api.jquery.com/category/events/event-object/.

Manejo de eventos

jQuery dispone de varias funciones relacionadas con la gestión de los eventos. De hecho, jQuery tiene un modelo de eventos muy completo que facilita la programación de interacciones entre el usuario y los objetos de la página web.

Para cada evento disponible hay dos posibilidades de manejo:

1. Se le puede **pasar una función** que determine su comportamiento.
2. No se le pasa función, entonces **se ejecuta el evento JavaScript** del elemento. jQuery tiene tantas funciones como eventos estándar de JavaScript. El nombre de cada función es el mismo que el del evento, pero sin el habitual prefijo "on" de los eventos:

Un ejemplo del primer caso es el siguiente:

```
$("p").click(function () { alert($(this).text());});
```

Este código muestra una ventana de alerta cuando se hace clic (onclick) en cualquier párrafo del documento. Esta función ha sido definida expresamente para atender este evento.

```
$ ("p").click() ;
```

En este caso, se ejecuta la función click() que es la asociada por defecto al evento onclick.

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 $.fn.on de jQuery ($.fn.bind en versiones anteriores a jQuery 1.7). El método $.fn.on 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.on

```
$('p').on('click', function() {
console.log('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.on con uno o más pares de nombres claves/valores. Cada clave será el nombre del evento mientras que cada valor será la función a ejecutar cuando ocurra el evento.

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


Segundo ejemplo:

<!doctype html>
<html lang="en">
```

```
<head>
  <meta charset="utf-8">
  <title>on demo</title>
  <style>
  .test {
    color: #000;
    padding: .5em;
    border: 1px solid #444;
  }
  .active {
    color: #900;
  }
  .inside {
    background-color: aqua;
  }
  </style>
  <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
</head>
<body>

<div class="test">test div</div>

<script>
$( "div.test" ).on({
  click: function() {
    $( this ).toggleClass( "active" );
  }, mouseenter: function() {
    $( this ).addClass( "inside" );
  }, mouseleave: function() {
    $( this ).removeClass( "inside" );
  }
});
</script>

</body>
</html>
```

Desvincular Eventos

Para desvincular (en ingles *unbind*) un controlador de evento, puede utilizar el método $.fn.off 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.

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

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

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

LISTA DE EVENTOS DE RATÓN, TECLADO Y VENTANA

En esta sección se muestras los diferentes eventos (ratón, teclado y ventana) que atiende jQuery para la interacción del usuario con objetos de la web. Además, en la última parte se describen los eventos asociados a las ventanas.

Eventos de ratón

Los eventos de ratón ejecutan una función cuando el usuario utiliza el puntero del ratón en un elemento determinado.

La estructura básica de todos los eventos es:

Elemento.Evento (función (handler));

Que se puede traducir fácilmente a la versión:

Elemento.on("evento", función(handler));

Sobre un elemento, cuando el usuario lanza un evento lo atiende (se ejecuta) la función. A continuación se describen los eventos de ratón disponibles. En la descripción se supone una configuración de ratón para diestros, donde el botón izquierdo de ratón es el que se usa para seleccionar.

- **.blur().** Se lanza cuando el elemento pierde el foco.
- .**click()**. Se lanza un evento clic cuando el usuario, colocado sobre un elemento (objeto) presiona el botón izquierdo del ratón y lo levanta, todo dentro del mismo objeto. Cualquier elemento HTML puede recibir este evento.
- **.dblclick().** Es el llamado doble clic. Este evento se lanza cuando se repite dos veces la secuencia de un clic, es decir, sobre un elemento (objeto) el usuario presiona el botón izquierdo del ratón, lo levanta, lo vuelve a presionar y lo levanta, toda la secuencia dentro del mismo objeto. Cualquier elemento HTML puede recibir este evento.
- **.focusin()**. Se lanza este evento cuando un elemento (objeto) gana el foco. Por ejemplo, cuando se selecciona una caja de texto para poder escribir sobre ella. No todos los elementos HTML pueden recibir el foco.
- **.focusout().** Es el evento contrario a .focusin(), se lazan cuando un elemento (objeto) pierde el foco. A diferencia del método blur (), el método focusOut () también se dispara si cualquier elemento hijo pierde el foco.
- **.mousedown()**. Este evento se lanza cuando el usuario presiona el botón del ratón sobre un elemento (objeto). La diferencia con respecto a click() es que éste no se lanza hasta que se suelta el botón dentro de ese mismo objeto. Cualquier elemento HTML puede recibir este evento.

- **.mouseup()**. Este evento se lanza cuando el usuario levanta el botón de ratón (previamente presionado). Cualquier elemento HTML puede recibir este evento.
- **.mouseenter()**. Se lanza este evento cuando el puntero del ratón entra en un elemento (objeto). No hace falta presionar ningún botón para que este evento se lance cuando el puntero entra en la región definida por el elemento. Cualquier elemento HTML puede recibir este evento.
- **.mouseleave()**.Se lanza este evento cuando el puntero del ratón sale de la región delimitada por un elemento (objeto). Es el evento opuesto a mouseenter(). Cualquier elemento HTML puede recibir este evento.
- **.mousemove()**. Este evento se lanza cuando el puntero de ratón se mueve dentro de un elemento. Cualquier elemento HTML puede recibir este evento.
- **.mouseover()**. Es parecido a mouseenter(), pero se diferencia cuando hay elementos anidados (por ejemplo, un <p> dentro de un hijo anidado dentro de un padre): se lanza el mouseover() del padre al pasar de <p> a hijo o de hijo a <p> o de hijo al padre.
- **.mouseout()**. Es el opuesto de mouseover(). Es parecido al mouseleave() pero con la diferencia con objeto anidados mostrada con mouseover().

El siguiente ejemplo, cuando se ejecuta, muestra la diferencia entre mouseover() y mouseenter().

```
<!DOCTYPE html>
<html>
     <head>
          <style>
               div.first {
                    position:relative;background-
                    color:#3f3;left:0px;
                    height:4OOpx;width:400px;}
               div.second {
                    position:relative;background-
                    color:#a3f;top:100px;left:100px;height:
                    200px; width:200px;}
               p {background:#33CC99;}
          </style>
          <script                    type="text/javascript"
          src="jquery.js"></script><script
          type="text/javascript">
               $( function() {
```

```
                    Var cuenta_enter=0;
                    Var cuenta_over=0;
               $("div.first").mouseenter(function() {
               $
               ("p:first").text("con_mouseEnter"+cuenta_ent
               er++);
               });
               $("div.first").mouseover(function() {
                    $("p:last").text("con          mouseOver
               "+cuenta over++);
                    });
               });
          </script>
     </head>
     <body>
          <div class=”first”>
               <div class=”second”>
                    <p></p>
                    <p></p>
               </div>
          </div>
     </body>
</html>
```

En el ejemplo, al pasar el ratón de los párrafos <p> al hijo aumenta la cuenta de cuenta_over. Lo mismo al pasar del hijo al padre. Sin embargo, la cuenta_enter solo se incremente al pasar de fuera de los al padre.

Eventos de teclado

Los eventos de teclado ejecutan una función cuando el usuario utiliza las teclas del teclado en un determinado elemento.

La estructura básica de todos los eventos es:

Elemento.Evento (función (handler));

Que se traduce fácilmente a su versión:

Elemento.on (“evento”, función (handler));

Sobre un elemento, cuando el usuario lanza un evento lo atiende (se ejecuta) la función. A continuación se describen los eventos de teclado disponibles. La mayoría de

ellos van asociados a formularios ya que suelen ser usados cuando el usuario entra mediante teclado.

- **.focusin().** Se lanza este evento cuando un elemento (objeto) gana el foco. Por ejemplo, cuando se selecciona una caja de texto para poder escribir sobre ella. No todos los elementos HTML pueden recibir el foco.
- **.focusout()**. Es el evento contrario a focusin(), se lazan cuando un elemento (objeto) pierde el foco.
- **.keydown()**. Se lanza el evento cuando el usuario presiona una tecla. El evento se lanza cuando el elemento tiene el foco. Para determinar que tecla ha sido presionada se examina el objeto event que se le pasa a la función que atiende el evento. jQuery ofrece la propiedad .which de event para recuperar el código de la tecla que se presiona- Si lo que se desea es obtener el texto que se ha introducido es mejor opción usar .keypress().
- **.keyup().** Se lanza el evento cuando el usuario libera una tecla. El evento se lanza cuando el elemento tiene el foco.
- **.keypress()**. Se lanza el evento cuando el usuario presiona una tecla pero con la idea de registrar qué carácter se ha pulsado. También se puede conseguir con .keydown(), sin embargo si se pulsa una tecla y se mantiene, keydown() solo registra un evento para esa tecla, mientras que keypress() registra uno por cada carácter introducido.

El siguiente ejemplo muestra un sencillo detector de qué tecla se ha presionado en cada momento. El parámetro e es de tipo event y es el que contiene la tecla pulsada, obtenida con e.which. Con e.preventDefault(); se consigue no se escriba nada en el textarea, es decir se inhabilita el comportamiento habitual del evento, que es escribir las teclas en el textarea.

```
<!DOCTYPE html>
<html>
<head>
<style>
p (background:#33CC99; }
</style>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jqu
ery.min.js"></script>
<script type="text/javascript">
$( function(){
$('#objetivo' ).keypress (function (e){
      e.preventDefault();
      $("#salidapress").html(e.which+";"+
            String.fromCharCode(e. which) )
```

```
      });
});
</script>
</head>
<body>
<form>
<fieldset>
<label for="objetivo">Teclea cualquier cosa</label>
<input id="objetivo"   type="text"/>
<div id="salidapress"></div>
</fieldset>
</form>
</body>
</html>
```

Eventos de ventana

Los eventos principales de ventana son: scroll y resize. La estructura básica de los dos es la misma que en los eventos anteriores de ratón y teclado.

- **.scroll()**. Este evento se atiende cuando sobre un elemento que tiene barras de desplazamiento se mueve una de ellas. El evento se suele aplicar a elemento "ventana" pero también sobre frames o elementos con la propiedad CSS overflow puesta a scroll.
- **.resize().** El evento se lanza cuando a un elemento tipo ventana se le cambia el tamaño.

El siguiente ejemplo muestra el comportamiento de ambos eventos. Scroll() se activa cuando se mueven las barras de desplazamiento o se pulsa en el texto "PULSA Y lanzo el evento pero no hago el scroll (desplazamiento)". Al activarse aparece un texto "se hace Scroll", pero este se desvanece (fadeout). Por otro lado, cuando se modifica el tamaño de la ventana del navegador, si el ancho de la ventana es mayor de 500 px se pone un color de fondo para la caja de texto y si es menor otro.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {overflow: scroll;
     width: 300px;
     height: 100px;
     font-family: Arial, Helvetica, sans-serif;
     color: #888888;
     padding: 7px 3px 0 4px;
```

```
     font-size: 12px;


}
span {display:inline;}
</style>
<script type="text/javascript"
src="jquery-1.6.4.js"></script>
<script type="text/javascript">
          $( function() {
               $('#prueba').scroll(function() {
               $("span#textoevento").css("display",
"inline").fadeOut("slow"); });
               $('#repite').click(function() {  $
('#prueba').scroll();
               });
               $(window).resize(function() {

                    if ($(window).width()>500)
                    {
                         $
('div').css("background","#99CC99");
                    }
                    else
                    {
                         $
('div').css("background","#9999CC");
                    }

               });
          });
</script>
</head>

<body>

<div id="prueba" >
Loremipsumdolor sitamet, consecteturadipisicingelit,
sed do eiusmodtemporincididuntutlabore et dolore magna
aliqua. Utenim ad minim veniam, quisnostrud exercitation
ullamcolaborisnisi ut aliquip ex eacommodoconsequat.
Duisauteirure dolor in reprehenderit in voluptatevelit
essecillumdoloreeufugiatnullapariatur. Excepteur
sintoccaecatcupidatat non proident, sunt in culpa qui
officiadeseruntmollitanim id estlaborum.
</div>
<span id="textoevento">
  Se hace Scroll
</span>
<span id="repite"> PINCHA Y lanzo el evento pero no hago el
scroll (desplazamiento)
```

```
</span>
</body>
</html>
```

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);
```

en el siguiente ejemplo se cancela la acción por defecto de un enlace usando el método .preventDefault() en función del destino del enlace.

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

En el siguiente ejemplo se cancela la acción por defecto de un enlace usando el método .preventDefault() para todos los enlaces de una página.

```
$( "body" ).on( "click", "a", function( event ) {
  event.preventDefault();

});
```

VINCULAR UN EVENTO UTILIZANDO EL MÉTODO $.FN.ON CON INFORMACIÓN ASOCIADA

$(selector).on('lista de eventos separada por comas', **{dato: valor},** función manejadora);

```
$('input').on(
        'click blur',           // es posible vincular múltiples eventos
                                // al elemento
        { nombre: 'Pedro' },    // se puede pasar información asociada
                                //como argumento

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

Ejemplo:

```
function greet( event ) {
  alert( "Hello " + event.data.name );
}
$( "button" ).on( "click", {name: "Karl"}, greet );
$( "button" ).on( "click", {name: "Addy"}, greet );
```

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 vínculo (*binding*). De esta forma, podrá llamar a la función cuando lo desee en lugar de ejecutar $.fn.trigger.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>on demo</title>
  <style>
  p {
```

```
    color: red;
  }
  span {
    color: blue;
  }
  </style>
  <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
</head>
<body>

<p>Has an attached custom event.</p>
<button>Trigger custom event</button>
<span style="display:none;"></span>

<script>
$( "p" ).on( "myCustomEvent", function( event, myName ) {
  $( this ).text( myName + ", hi there!" );
  $( "span" )
    .stop()
    .css( "opacity", 1 )
    .text( "myName = " + myName )
    .fadeIn( 30 )
    .fadeOut( 1000 );
});
$( "button" ).click(function () {
  $( "p" ).trigger( "myCustomEvent", [ "John" ] );
});
</script>

</body>
</html>
```

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**. 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.

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.on

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

Segundo ejemplo:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>on demo</title>
  <style>
  p {
    background: yellow;
    font-weight: bold;
    cursor: pointer;
    padding: 5px;
  }
  p.over {
    background: #ccc;
  }
  span {
    color: red;
  }
  </style>
  <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
</head>
<body>

<p>Click me!</p>
<span></span>

<script>
var count = 0;
$( "body" ).on( "click", "p", function() {
  $( this ).after( "<p>Another paragraph! " + (++count) + "</p>" );
});
</script>

</body>
</html>
```

Desvincular Eventos Delegados

Si necesita remover eventos delegados utilice el método $.fn.off para eventos conectados con $.fn.on. Al igual que cuando se realiza un vínculo, opcionalmente, se puede pasar el nombre de una función vinculada.

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

FUNCIONES AUXILIARES DE EVENTOS

jQuery ofrece dos funciones auxiliares para el trabajo con eventos:

$.fn.hover

Este evento se lanza cuando el puntero entra sobre un elemento o sale de él. Su principal ventaja es que permite definir una función para cuando el puntero entra en el elemento y otra para cuando salga. Es un evento muy usado con menús, cuando se desea hacer, por ejemplo, seleccionar opciones de menú, destacando alguna de sus propiedades o menús que se despliegan. Su sintaxis es:

Elemento.hover (función_entrada (handler), función_salida (handler));

El siguiente ejemplo muestra un submenú que aparece al colocarse con el ratón en la opción "localización".

```
<!DOCTYPE html>
<html>
     <head>
          <script   type="text/javascript"   src="jquery.js>
          </script>
          <script type="text/javascript">
               $( function() {
                    $("li#localizacion").hover(
                    function(){
                         $("span#opcion_loc").show();
                    },
                    function () {
                         $("span#opcion_loc").hide();
                    });
               });
          </script>
          <style>
               ul{ margin-left:20px; color:blue; }
               li { cursor:default; }
               span { display:none }
          </style>
     </head>
     <body>
```

```
            <ul>
                  <li id="localizador!">Localización</li>
                  <span id="opcion_loc">
                        <ul>
                              <li>Mapa</li>
                              <li>Dirección</li>
                              <li>Teléfonos</li>
                        </ul>
                  </span>
                  <li id="recetas">Recetas</li>
      </body>
</html>
```

$.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 dificultar la depuración del código.

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

EJEMPLOS

1. **Crear una "Sugerencia" para una Caja de Ingreso de Texto**

Abra el archivo /ejercicios/ejercicio1.html en el navegador. La tarea a realizar es utilizar el texto del elemento label y aplicar una "sugerencia" en la caja de ingreso de texto. Los pasos a seguir son los siguientes:

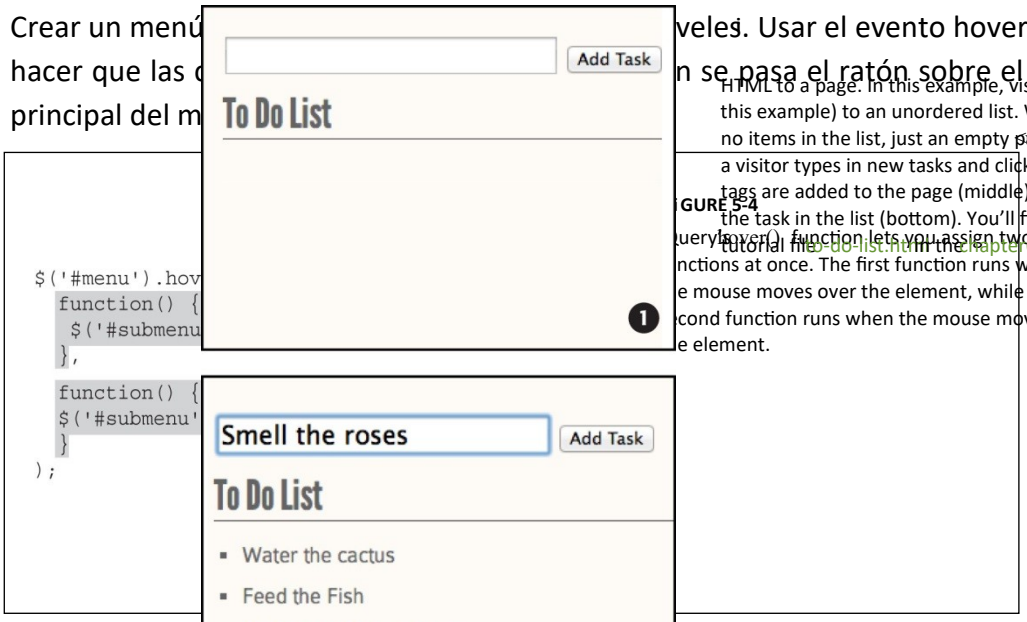
1. Establecer el valor del elemento *input* igual al valor del elemento *label*.
2. Añadir la clase "hint" al elemento *input*.
3. Eliminar el elemento *label*.

4. Vincular un evento *focus* en el *input* para eliminar el texto de sugerencia y la clase “hint”.
5. Vincular un evento *blur* en el *input* para restaurar el texto de sugerencia y la clase “hint” en caso que no se haya ingresado algún texto.

¿Qué otras consideraciones debemos tener en cuenta si se desea aplicar esta funcionalidad a un sitio real?

2. **Crear un menú dinámico.**

   Crear un menú veles. Usar el evento hover para hacer que las c n se pasa el ratón sobre el nivel principal del m

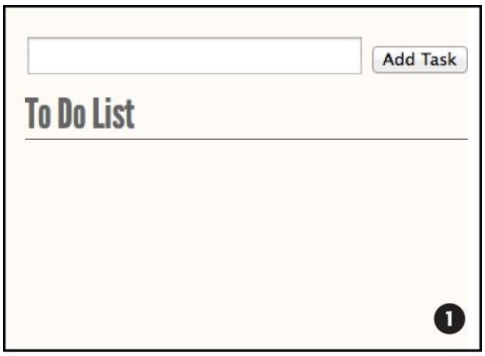


HTML to a page. In this example, visitors can add new items (tasks, in this example) to an unordered list. When the page first loads there are no items in the list, just an empty pair of <ul> tags (top). As a visitor types in new tasks and clicks the Add Task button, new <li> tags are added to the page (middle). To mark a task done, just click the task in the list (bottom). You’ll find this working example in the tutorial files in the chapter05

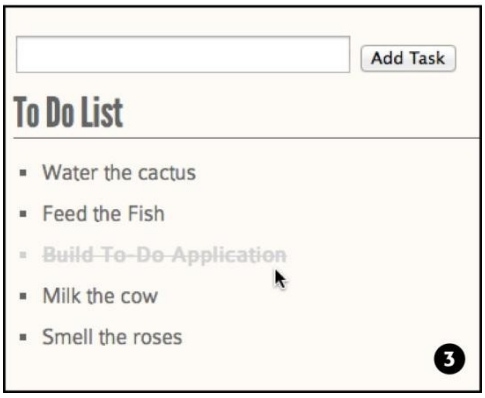
GURE 5-4

uery's hover() function lets you assign two nctions at once. The first function runs when e mouse moves over the element, while the cond function runs when the mouse moves off e element.

3. **Crear una**


Add Task
To Do List
1


Smell the roses
Add Task
To Do List
Water the cactus
Feed the Fish
Build To-Do Application
Milk the cow
2


Add Task
To Do List
Water the cactus
Feed the Fish
Build To-Do Application
Milk the cow
Smell the roses
3