

Manual jQuery & jQueryUI



INSTITUTO TECNOLÓGICO DE CHIHUAHUA II

March 14, 2012

Elaborado por: •Isaac Ojeda Quintana
•Mariana Martínez Almeida

Tabla de contenido

1.	Introducción	4
1.1	Software a Utilizar	4
1.1.1	Descargar jQuery y jQueryUI	4
1.2	Añadir JavaScript a una Página	4
2.	Introducción a jQuery	6
2.1	Objetos en JavaScript	6
2.2	Conceptos Básicos de jQuery	7
2.2.1	\$(document).ready()	7
2.2.2	Selección de Elementos	7
2.2.3	Guardar Selecciones	8
2.2.4	Obtenedores (Getters) & Establecedores (Setters)	9
2.3	Ejemplo practico	9
3.	Eventos	14
3.1	Introducción	14
3.2	Vincular Eventos a Elementos	14
4.	Efectos	15
4.1	Introducción	15
4.2	Efectos Incorporados en la Biblioteca	15
4.2.1	Cambiar la Duración de los Efectos	15
4.2.1.1	Realizar una Acción Cuando un Efecto fue Ejecutado	16
4.3	Efectos Personalizados con \$.fn.animate	16
4.4	Control de los Efectos	16
4.5	Ejemplo practico (Eventos & Efectos)	17
5.	jQuery User Interface	21
5.1	Introducción	21
5.1.1	Librería y estilos de página	21
5.2	Módulos de jQuery UI	22
5.2.1	Interacciones	22
5.2.2	Widgets	22
5.3	Efectos	31
6	jQuery Plugins	32

6.1 Step Carousel.....	32
6.1.1 Estructura HTML.....	32
6.1.2 CSS.....	33
6.1.3 Configuración del Plugin.....	34
6.2 Galería fotográfica con “Glisse”	36
6.2.1 Estructura HTML.....	37
6.2.2 CSS.....	37
6.2.3 Configuración	38
Trabajos citados	40

1. Introducción

jQuery es una biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

jQuery se está convirtiendo rápidamente en una herramienta que todo desarrollador de interfaces web debería de conocer. El propósito de este manual es proveer un resumen de la biblioteca, de tal forma que para cuando lo haya terminado de leer, será capaz de realizar tareas básicas utilizando jQuery y tendrá una sólida base para continuar el aprendizaje.

1.1 Software a Utilizar

Para trabajar con el contenido de este manual necesitara varias de las siguientes herramientas:

- Un navegador web actualizado
- Un editor de textos planos(como Notepad++,Komodo,etc)
- Las bibliotecas jQuery y jQueryUI

1.1.1 Descargar jQuery y jQueryUI

Podemos visitar la página oficial de jQuery para obtener la versión más actual y estable: http://docs.jquery.com/Downloading_jQuery y <http://jqueryui.com/download>. En este último nos da la opción de seleccionar o crear un tema para jQueryUI, nos mostrara ejemplos de componentes de jQuery UI y seleccionaremos el que necesitemos.

1.2 Añadir JavaScript a una Página

Una vez descargado jQuery y el plugin jQueryUI debemos agregarlo a nuestra página web. Existen dos formas de insertar código JavaScript dentro de una pagina: escribiendo código en la misma o a través de un archivo externo utilizando la etiqueta script.

```
1  <!--
2  Taller jQuery & jQueryUI
3  -->
4  <!DOCTYPE html>
5  <html>
6  <head>
7      <title></title>
8      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
9      <script type="text/javascript">
10         //Codigo JavaScript
11     </script>
12 </head>
13 <body>
14
15     <h2>Taller de jQuery y jQueryUI</h2>
16
17
18 </body>
19 </html>
```

Dentro de esta etiqueta podemos empezar a escribir código JavaScript, la segunda opción es usando la misma etiqueta pero nuestro código JavaScript ira dentro de un archivo de texto plano con extensión “.js”.

```
<script type="text/javascript" src="js/jquery-1.6.2.min.js"></script>
<script type="text/javascript">
    //Codigo JavaScript
</script>
```

Dentro del atributo ‘src’ debe de ir la ruta donde se encuentra en este caso el archivo con extensión .js .Y así es como agregamos la biblioteca jQuery a nuestra página web, y en la etiqueta de abajo estaremos casi listos para comenzar a escribir código JavaScript utilizando jQuery

2. Introducción a jQuery

jQuery se encuentra escrito en JavaScript, un lenguaje de programación muy rico y expresivo. El manual está orientado a personas con conocimientos básicos en este lenguaje así que no se tomaran temas referentes a JavaScript, pero si usted esta interesado en aprender el lenguaje más en profundidad, puede leer el libro JavaScript: The Good Parts escrito por Douglas Crockford.

2.1 Objetos en JavaScript

Los objetos son elementos que pueden contener cero o más conjuntos de pares de nombres claves y valores asociados a dicho objeto. Los nombres claves pueden ser cualquier palabra o número válido. El valor puede ser cualquier tipo de valor: un número, una cadena, un arreglo, una función, incluso otro objeto.

[**Definición:** Cuando uno de los valores de un objeto es una función, ésta es nombrada como un método del objeto.] De lo contrario, se les llama propiedades.

Curiosamente, en JavaScript, casi todo es un objeto — arreglos, funciones, números, incluso cadenas — y todos poseen propiedades y métodos.

Creación de un “objeto literal”

```
<script type="text/javascript">
  //Codigo JavaScript

  var myObject = {
    sayHello : function() {
      console.log('hello');
    },

    myName : 'Rebecca'
  };

  myObject.sayHello(); // se llama al método sayHello,
  // el cual muestra en la consola 'hello'

  console.log(myObject.myName); // se llama a la propiedad myName,
  // la cual muestra en la consola 'Rebecca'

</script>
```

2.2 Conceptos Básicos de jQuery

2.2.1 \$(document).ready()

No es posible interactuar de forma segura con el contenido de una página hasta que el documento no se encuentre preparado para su manipulación. jQuery permite detectar dicho estado a través de la declaración `$(document).ready()` de forma tal que el bloque se ejecutará sólo una vez que la página este disponible.

El bloque `$(document).ready()`

```
<script type="text/javascript">
  //Codigo JavaScript
  $(document).ready(function() {
    alert('el documento está preparado');
  });
</script>
```

Existe una forma abreviada para `$(document).ready()` la cual podrá encontrar algunas veces; sin embargo, es recomendable no utilizarla en caso que este escribiendo código para gente que no conoce jQuery.

Forma abreviada para `$(document).ready()`

```
//Codigo JavaScript
$(function() {
  alert('el documento está preparado');
});
```

Además es posible pasarle a `$(document).ready()` una función nombrada en lugar de una anónima:

Pasar una función nombrada en lugar de una función anónima

```
//Codigo JavaScript
function miFuncion() {
  alert('el documento está preparado');
}
$(document).ready(function() {
  miFuncion();
});
```

2.2.2 Selección de Elementos

El concepto más básico de jQuery es el de “seleccionar algunos elementos y realizar acciones con ellos”. La biblioteca soporta gran parte de los selectores CSS3 y varios más no estandarizados. En <http://api.jquery.com/category/selectors/> se puede encontrar una completa referencia sobre los selectores de la biblioteca.

A continuación se muestran algunas técnicas comunes para la selección de elementos:

Selección de elementos en base a su ID

```
$('#miId'); // notar que los IDs deben ser únicos por página
```

Selección de elementos en base al nombre de clase

```
$('div.myClass'); // si se especifica el tipo de elemento,  
// se mejora el rendimiento de la selección
```

Selección de elementos por su atributo

```
$('input[name=first_name]'); // tenga cuidado, que puede ser muy lento
```

Selección de elementos en forma de selector CSS

```
$('#contents ul.people li');
```

Pseudo-selectores

```
$('a.external:first'); // selecciona el primer elemento <a>  
                        // con la clase 'external'  
$('tr:odd');           // selecciona todos los elementos <tr>  
                        // impares de una tabla  
$('#myForm :input');    // selecciona todos los elementos del tipo input  
                        // dentro del formulario #myForm  
$('div:visible');       // selecciona todos los divs visibles  
$('div:gt(2)');         // selecciona todos los divs excepto los tres primeros  
$('div:animated');      // selecciona todos los divs actualmente animados
```

2.2.3 Guardar Selecciones

Cada vez que se hace una selección, una gran cantidad de código es ejecutado. jQuery no guarda el resultado por sí solo, por lo tanto, si va a realizar una selección que luego se hará de nuevo, deberá guardar la selección en una variable.

Guardar selecciones en una variable

```
var $divs = $('div'); // Siempre que hablemos de un objeto(s)  
                    // por convención se antepone un "$" para  
                    // referirnos a elementos de jQuery
```

Una vez que la selección es guardada en la variable, se la puede utilizar en conjunto con los métodos de jQuery y el resultado será igual que utilizando la selección original.

2.2.4 Obtenedores (Getters) & Establecedores (Setters)

jQuery “sobrecarga” sus métodos, en otras palabras, el método para establecer un valor posee el mismo nombre que el método para obtener un valor. Cuando un método es utilizado para establecer un valor, es llamado método establecedor (en inglés setter). En cambio, cuando un método es utilizado para obtener (o leer) un valor, es llamado obtenido (en inglés getter).

El método \$.fn.html utilizado como establecedor

```
$('#h1').html('hello world');
```

El método html utilizado como obtenido

```
$('#h1').html();
```

2.3 Ejemplo practico

Vamos a hacer un ejemplo sencillo de selecciones de objetos y verificar que realmente obtuvimos un elemento HTML con jQuery.

Tendremos una pagina muy simple:

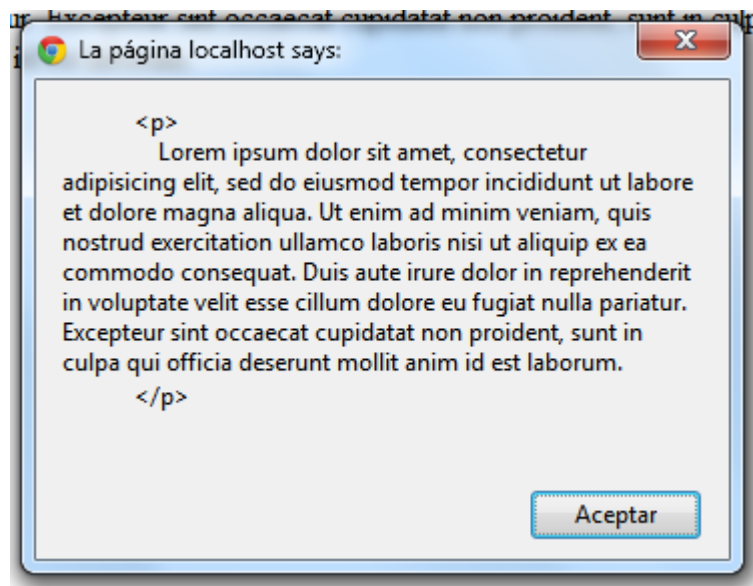
```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <script type="text/javascript" src="js/jquery-1.6.2.min.js"></script>
    <script type="text/javascript">
      //Codigo JavaScript
    </script>
  </head>
  <body>
    <h2>Taller de JQuery y JQueryUI</h2>
    <div class="miClase">
      <p>
        Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
      </p>
    </div>
    <div id="miId">
      <p>
        Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
      </p>
    </div>
  </body>
</html>
```

A simple vista ya podemos reconocer donde estamos agregando la biblioteca jQuery y en donde podemos empezar a escribir código JavaScript. Lo que haremos es obtener el valor del <div> que tiene la clase "miClase" y al <div> con el id "mild" le daremos un contenido que queramos (sustituyendo el contenido actual).

Dentro de la etiqueta <script> escribimos:

```
<script type="text/javascript">
  //Codigo JavaScript
  $(document).ready(function() {
    //obtenemos el elemento <div> con clase "miClase"
    var $miDiv=$( "div.miClase" );
    alert($miDiv.html());
  });
</script>
```

Y el resultado será:



Si se fijan nos esta incluyendo también las etiquetas <p> literalmente. Esto es porque estamos obteniendo todo el código HTML contenida en la etiqueta <div>, si queremos seleccionar solo el texto, tenemos que seleccionar exclusivamente el elemento <p>.

```
var $elementoP=$( "div.miClase p" );
alert($elementoP.html());
```

Y el resultado ya será diferente.

Ahora vamos a usar la función html como un setter, como esta sobrecargado es la misma función pero le tenemos que especificar el contenido HTML que queremos agregar.

```
<script type="text/javascript">
  //Codigo JavaScript
  $(document).ready(function(){
    //obtenemos el elemento <div> con clase "miClase"
    var $elementoP=$( "div.miClase p" );
    alert($elementoP.html());

    $("#miId p").html("Agregando contenido al elemento p del id 'miID'")
  });
</script>
```

De esta forma con el método html() con una cadena como parámetro estamos agregando contenido al elemento <p> que esta dentro de cualquier cosa que tenga el id "miId"

Para hacer selecciones un poco diferentes vamos a crear una tabla que tendrá varias columnas y filas:

```
</br>
<h2>Tabla random</h2>
<table border='1'>
  <thead>
    <tr>
      <th>Nombre</th>
      <th>Apellidos</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Isaac</td>
      <td>Ojeda Quintana</td>
    </tr>
    <tr>
      <td>Mariana</td>
      <td>Martinez Almeida</td>
    </tr>
  </tbody>
</table>
```

Esta la agregamos al final del último div

Tabla random

Nombre	Apellidos
Isaac	Ojeda Quintana
Mariana	Martinez Almeida

Lo que podemos ver aquí es que el elemento <table> tiene dos hijos (thead y tbody) que a su vez estos tienen hijos inmediatos que son los elementos <tr> (las filas) y estos los elementos th y td que son las columnas (th para títulos y td para contenido normal). Veremos como le haremos para cambiar el nombre de Isaac y sus apellidos por medio de jQuery, la verdad en este momento se me ocurren varios caminos por donde irme, tal vez poniéndole un "id" sería el camino mas fácil pero no es nada practico cuando el contenido HTML se genera por medio de PHP o ASP. Así que nos iremos llamando una serie de funciones de una forma encadenada y será así:

```
var nombre=prompt("Escribe un nombre");
//se busca el elemento td donde se encuentra el nombre del primer
//registro
$("table tbody tr:first").children("td:first").html(nombre);
```

Lo primero que hacemos es pedir que el usuario escriba un nombre y para insertarlo en la celda donde dice "Isaac". Lo primero que se hace en el selector principal es obtener el primer elemento <tr> que es hijo directo de <tbody> y que este mismo es hijo directo de <table> y esto se convierte en una función encadenada, quiere decir que se mandan a llamar mas funciones ya que cada función nos devuelve un objeto que contiene mas funciones. La función children nos esta devolviendo el primer elemento <td> y este como es el que ya contiene lo que queremos cambiar, mandamos a llamar la función html() dándole de parámetro una cadena que se capturo anteriormente, espero y esto sea de fácil comprensión ya que al momento de trabajar en el árbol que se genera en HTML es de vital importancia utilizar este tipo de funciones encadenadas.

Resultado:

Taller de JQuery y JQueryUI

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fu id est laborum.

Agregando contenido al elemento p del id 'miID'

Tabla random

Nombre	Apellidos
Isaac	Ojeda Quintana
Mariana	Martinez Almeida

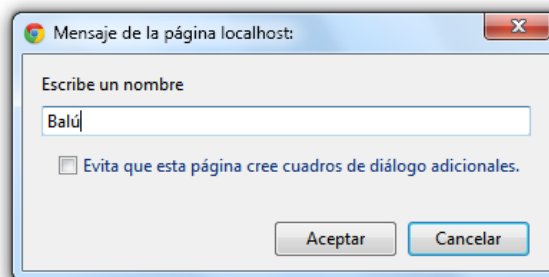


Tabla random

Nombre	Apellidos
Balú	Ojeda Quintana
Mariana	Martinez Almeida

->

Y para cambiar el apellido es de una manera similar:

```
var apellido=prompt("Escribe un apellido");  
//similar a la seleccion anterior, pero estando en el elemento  
//td se llama next() que nos devuelve el elemento siguiente a este  
$("table tbody tr:first").children("td:first").next().html(apellido);
```

Hacemos casi exactamente lo mismo, en este ejemplo lo que hacemos ahora es usar la función `next()`, que nos devuelve el elemento siguiente al actual, el actual es el primer `<td>` el siguiente `<td>` es donde se encuentra el apellido

NOTA: la mayoría (si no es que todas) de las funciones se pueden sobrecargar, en este caso `next()` si se le incluye una selección se ira a buscar el siguiente elemento con la selección dada por parámetro

3. Eventos

3.1 Introducción

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/>.

3.2 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() {  
    alert('diste click');  
});
```

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

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

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() { alert('clickeado'); },  
    'mouseover': function() { alert('sobrepasado'); }  
});
```

4. Efectos

4.1 Introducción

Con jQuery, agregar efectos a una página es muy fácil. Estos efectos poseen una configuración predeterminada pero también es posible proveerles parámetros personalizados. Además es posible crear animaciones particulares estableciendo valores de propiedades CSS.

Para una completa documentación sobre los diferentes tipos de efectos puede visitar la sección effects: <http://api.jquery.com/category/effects/>.

4.2 Efectos Incorporados en la Biblioteca

Los efectos más utilizados ya vienen incorporados dentro de la biblioteca en forma de métodos:

\$.fn.show

Muestra el elemento seleccionado.

\$.fn.hide

Oculto el elemento seleccionado.

\$.fn.fadeIn

De forma animada, cambia la opacidad del elemento seleccionado al 100%.

\$.fn.fadeOut

De forma animada, cambia la opacidad del elemento seleccionado al 0

\$.fn.slideDown

Muestra el elemento seleccionado con un movimiento de deslizamiento vertical.

\$.fn.slideUp

Oculto el elemento seleccionado con un movimiento de deslizamiento vertical.

\$.fn.slideToggle

Muestra o oculta el elemento seleccionado con un movimiento de deslizamiento vertical, dependiendo si actualmente el elemento está visible o no.

Uso básico de un efecto incorporado

```
$('h1').show();
```

4.2.1 Cambiar la Duración de los Efectos

Con la excepción de \$.fn.show y \$.fn.hide, todos los métodos tienen una duración predeterminada de la animación en 400ms. Este valor es posible cambiarlo.

Configurar la duración de un efecto

```
$('h1').fadeIn(300); // desvanecimiento en 300ms
$('h1').fadeOut('slow'); // utilizar una definición de velocidad interna
```


4.2.1.1 Realizar una Acción Cuando un Efecto fue Ejecutado

A menudo, querrá ejecutar una acción una vez que la animación haya terminado — ya que si ejecuta la acción antes que la animación haya acabado, puede llegar a alterar la calidad del efecto o afectar a los elementos que forman parte de la misma. [Definición: *Las funciones de devolución de llamada* (en inglés *callback functions*) proveen una forma para ejecutar código una vez que un evento haya terminado.] En este caso, el evento que responderá a la función será la conclusión de la animación. Dentro de la función de devolución, la palabra clave `this` hace referencia al elemento en donde el efecto fue ejecutado y al igual que sucede con los eventos, es posible transformarlo a un objeto jQuery utilizando `$(this)`.

Ejecutar cierto código cuando una animación haya concluido

```
$('#div.old').fadeOut(300, function() {  
    $(this).remove();  
});
```

4.3 Efectos Personalizados con \$.fn.animate

Es posible realizar animaciones en propiedades CSS utilizando el método `$.fn.animate`. Dicho método permite realizar una animación estableciendo valores a propiedades CSS o cambiando sus valores actuales.

Efectos personalizados con \$.fn.animate

```
$('#div.funtimes').animate(  
    {  
        left : "+=50",  
        opacity : 0.25  
    },  
    300, // duration  
    function() { alert('realizado'); // función de devolución de llamada  
});
```

4.4 Control de los Efectos

jQuery provee varias herramientas para el manejo de animaciones.

\$.fn.stop

Detiene las animaciones que se están ejecutando en el elemento seleccionado.

\$.fn.delay

Espera un tiempo determinado antes de ejecutar la próxima animación.

```
$('#h1').show(300).delay(1000).hide(300);
```

4.5 Ejemplo practico (Eventos & Efectos)

Para abarcar todo lo visto en los últimos dos capítulos haremos esta pequeña práctica que consta de puros elementos <p> (párrafos) y algunos div.

Utilizaremos la misma estructura de la practica pasada, asi que solo les mostrare la parte del <body>:

Efecto hide & show

Código HTML:

```
<body>
  <h2>Efectos y Animaciones en jQuery</h2>
  <h4>Efecto hide & show </h4>
  <p id="p1">
    Lorem ipsum dolor sit amet, consectetur
  </p>
```

Código JavaScript:

```
$("#p#p1").click(function() {
  $(this).hide("slow").delay(1000).show("slow", function() {
    alert("Este fue el efecto hide&show"); //callback
  })
});
```

A simple vista podemos ver que primero estamos seleccionando un elemento <p> que tiene un id=p1 y con la función click() estamos agregando un evento que obviamente actuara al darle un click sobre el elemento donde lo estamos aplicando, en este caso un elemento <p> con id=p1. Esto se podrá ver un poco confuso pero asi se hara en los demás ejemplos, entonces veremos solo este en partes:

```
$("#p#p1").click(function() {
  //codigo que se ejecutara al dar click
});
```

En la función click estamos dando de parámetro una función anónima que será la que se llamara al dar click. También podemos crear funciones no anónimas y pasarlas por parámetro:

```
var clickBotonUno= function clickBotonUno() {
  //codigo del evento
}
```

```
$("#p#p1").click(clickBotonUno);
```

Y el resultado será el mismo, ustedes decidirán si crean sus funciones no anónimas para tener mas organizado nuestro código y de alguna manera mas estructurado. Ok ahora veamos que hay dentro del evento click:

```
$(this).hide("slow").delay(1000).show("slow",function(){
    alert("Este fue el efecto hide&show m"); //callback
});
```

Primeramente estamos seleccionando el elemento actual (this), al principio seleccionamos el elemento <p> con id=p1 entonces ese elemento es el actual, seleccionamos con this y aplicamos la función hide() y estamos dando de parámetro la velocidad, recuerden podemos dar de parámetros los milisegundos o "slow" o "fast" que están por default. A continuación por medio de una función encadenada volvemos a llamar una función y en este caso es delay() que lo que hará es dormirse los milisegundos que le den por parámetro, en este caso dormirá la ejecución 1 segundo y al terminar manda a llamar la función show() y estamos dando 2 parametros y el segundo es opcional. Pero en este caso estamos agregando una función anónima (como la del click) pero esta función se llama "callback", lo que tiene de diferente como lo vimos anteriormente esta función se llama cuando se termina de ejecutar la animación, podemos correr nuestra pagina y al dar click veremos el resultado.

Efecto Fade Out & Fade In

Los siguientes efectos son exactamente lo mismo:

Codigo HTML:

```
<p id="p2">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do ei
</p>
```

Codigo JavaScript:

```
$("#p2").click(function(){
    $(this).fadeOut("slow").delay(1000).fadeIn("slow",function(){
        alert("Este fue el efecto fadeOut&fadeIn"); //callback
    })
});
```

Como pueden ver lo implementamos de la misma forma y así podemos hacerlo también con el efecto slideUp y slideDown.

Efecto FadeTo

Este es un poquito diferente ya que se le agrega un parámetro mas aparte del de la velocidad y este es la opacidad, la opacidad se mide de 0 a 1 (porcentaje), podemos poner 0.5 si queremos que la opacidad cambie a un 50% en los milisegundos que nosotros queramos.

Código HTML:

```
<h4>Efecto fadeTo</h4>
<div id="div1" style="background:yellow;width:300px;height:300px" ></div>
```

Código JavaScript:

```
$("#div#div1").click(function() {
    $(this).fadeTo(500,0).delay(1000).fadeTo(500,1,function() {
        alert("Este fue el efecto FadeTo"); //callback
    })
});
```

El primer parámetro son los milisegundos y el segundo es la opacidad y como siempre esta la opción de la función callback.

Animaciones Personalizadas

Esto ya es un poco diferente y si nos llevara mas líneas de código:

Código HTML:

```
<h4>Animaciones Personalizadas</h4>
<div id="div2" style="font-size:0px; background:#98bf21;height:100px;width:100px;position:relative">Hola</div>
```

Esto es una caja pequeña que contiene un color y las letras de 0 pixeles, esto para hacerlas grandes por medio de una animación.

Código JavaScript:

```
$("#div#div2").click(function() {

    $(this).animate({height:300},"slow");
    $(this).animate({width:300},"slow");
    $(this).animate({height:100},"slow");
    $(this).animate({width:100},"slow");
    $(this).animate({left:"70%"},"slow");
    $(this).animate({left:"0%"},"slow");
    $(this).animate({fontSize:"40px"},"slow");

});
```

En cada línea de código hacemos referencia al elemento actual que es un <div> con id=div2 y mandamos a llamar a la función animate() que en esta damos de parámetro lo que queremos modificarle al elemento por medio de estilos CSS y por otro parámetro la velocidad, por ejemplo en el primero vemos:

Originalmente la caja div tiene una altura (height) de 100 pixeles, pero con esta función la estamos cambiando a 300px de una manera lenta (Slow), y así consecutivamente vamos llamando a mas

funciones `animate` para ir creando nuestra animación personalizada. Como cada función también existe la posibilidad de agregar una función `callback`. En las siguientes animaciones modificamos la propiedad `"left"`, esta es la posición que estará respecto a la izquierda del objeto, puede ser en píxeles o en porcentaje, en este caso estamos alejando un 70% de la izquierda del objeto, y por último cambiamos el tamaño de la fuente (`fontSize`) a 40px. Podemos correr esta página y veremos el resultado 😊.

Para terminar, efecto `slideToggle()`

Esta es una función muy útil y por mi parte muy usada, crearemos lo siguiente:

Código HTML:

```
<h4>Efecto SlideToggle</h4>
<div id="div3" style="padding: 5;text-align: center;border:solid 1px #c3c3c3;">
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut
  </p>
</div>
<div id="div4" style="padding: 5;text-align: center;border:solid 1px #c3c3c3;">Ocultar/Mostrar</div>
```

Podemos correr nuestra página para ver como se ve este código HTML, simplemente agregamos estilos a las dos cajas `div`

Código JavaScript:

```
$("div#div4").click(function(){
  $("div#div3").slideToggle("slow");
});
```

Como ven, más sencillo que nunca y estoy seguro que no hay necesidad de explicarlo.

5. jQuery User Interface

5.1 Introducción

jQuery UI es una biblioteca de componentes para el framework jQuery que le añaden un conjunto de plug-ins, widgets y efectos visuales para la creación de aplicaciones web. Cada componente o módulo se desarrolla de acuerdo a la filosofía de jQuery5 (find something, manipulate it: encuentra algo, manipúlalo).

5.1.1 Librería y estilos de página

Para comenzar a utilizar jQuery UI debemos de incluir el archivo de JavaScript que ya hemos descargado de la pagina oficial en nuestra pagina y aparte los estilos de pagina del tema que hemos seleccionado al descargar la extensión. Lo incluiremos de la forma más normal:

```
<link type="text/css" rel="stylesheet" href="css/start/jquery-ui-1.8.16.custom.css"/>
<script type="text/javascript" src="js/jquery-1.6.2.min.js"></script>
<script type="text/javascript" src="js/jquery-ui-1.8.16.custom.min.js"></script>

<script type="text/javascript">
    $(function() {
        //documento listo
    });
</script>
```



5.2 Módulos de jQuery UI

5.2.1 Interacciones

jQuery UI añade comportamientos complejos a los elementos como lo son:

- **Draggable:** Hace al elemento arrastrable.
- **Droppable:** Permite que el elemento responda a elementos arrastrables.
- **Resizable:** Permite redimensionar el elemento.
- **Selectable:** Permite seleccionar entre una lista de elementos.
- **Sortable:** Ordena una lista de elementos.

Ejemplo: Draggable:

Supongamos que tenemos una caja <div> con contenido, podemos con `$(Div).draggable()`, ese div será totalmente movable con el cursor. Podemos encontrar un sinfín de ejemplos en el siguiente enlace: <http://jqueryui.com/demos/>, de una manera sencilla enseñan con demos lo que contiene jQuery UI, en este manual no tomaremos en cuenta las interacciones

5.2.2 Widgets

Este es de los módulos de jQuery mas interesantes y mas usados, ya que es la parte donde le damos un toque especial a nuestros sitios web con estos widgets.

Cada control tiene un conjunto de opciones configurables y se les pueden aplicar estilos [CSS](#).

- **Accordion:** Menú con efecto acordeón.
- **Autocomplete:** Caja con autocompletado.
- **Button:** Botón.
- **Dialog:** [Ventanas](#) con contenido.
- **Slider:** Elemento para elegir en un rango de valores.
- **Tabs:** [Pestañas](#).
- **Datepicker:** [Calendario](#) gráfico.
- **Progressbar:** Barra de progreso.

5.2.2.1 Accordion

Para usar estos widgets es demasiado fácil, solo hay que ser la estructura HTML y a veces CSS con los que fueron diseñados, en este caso para crear un acordeón hay que tener una caja <div> con un identificador ya que desde jQuery diremos que este será el acordeón:

Código HTML:

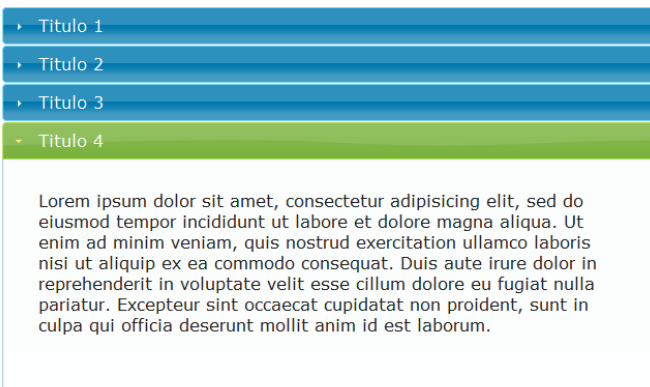
```
<div id="Acordion" style="width: 50%">
  <h3><a href="#">Titulo 1</a></h3>
  <div>
    <p>
      Lorem ipsum dolor sit amet, consectetur adipisicing elit, se
    </p>
  </div>
  <h3><a href="#">Titulo 2</a></h3>
  <div>
    <p>
      Lorem ipsum dolor sit amet, consectetur adipisicing elit, se
    </p>
  </div>
  <h3><a href="#">Titulo 3</a></h3>
  <div>
    <p>
      Lorem ipsum dolor sit amet, consectetur adipisicing elit, se
    </p>
  </div>
  <h3><a href="#">Titulo 4</a></h3>
  <div>
    <p>
      Lorem ipsum dolor sit amet, consectetur adipisicing elit, se
    </p>
  </div>
</div>
```

Hay que seguir obligatoriamente esta pequeña estructura y agregar el siguiente código

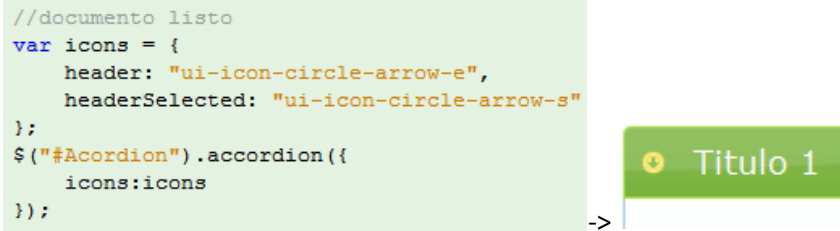
Código JavaScript:

```
//documento listo
$("#Acordion").accordion();
```

Y maravillosamente el resultado será este:



Como ya se dieron cuenta es relativamente sencillo hacer un menú con efecto de acordeón, claro esto es lo mas fundamental, recuerden que cada widget es personalizable y tiene un gran numero de métodos y atributos para modificar y personalizar el widget. Un ejemplo sencillo de personalización es agregarle iconos al acordeón.



5.2.2.2 AutoComplete

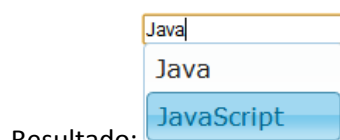
Este es uno de los widgets que tal vez sean los mas “difíciles” de implementar ya en un sistema o pagina en producción, ya que es necesario la tecnología Ajax pero difícil entre comillas he dicho. Veamos un ejemplo simple donde dentro de un input text nos a completara jQuery automáticamente con los posibles valores

Codigo HTML:

```
<input type="text" id="tags" />
```

Codigo JavaScript:

```
//arreglo de elementos que autocompletara
var availableTags = [
  "ActionScript",
  "AppleScript",
  "Asp",
  "BASIC",
  "C",
  "C++",
  "Clojure",
  "COBOL",
  "ColdFusion",
  "Erlang",
  "Fortran",
  "Groovy",
  "Haskell",
  "Java",
  "JavaScript",
  "Lisp",
  "Perl",
  "PHP",
  "Python",
  "Ruby",
  "Scala",
  "Scheme"
];
$( "#tags" ).autocomplete({
  source: availableTags
});
```



Resultado:

5.2.2.3 Button

Puede ser un `<input type= botón o un <button> o un <div>, cualquier cosa se puede llegar a convertir en un botón con la función button().`

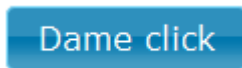
Código HTML:

```
<input type="button" class="boton" value="Dame click"/>
```

Código JavaScript:

```
$( ".boton" ).button();
```

Resultado:



Igual como lo vimos anteriormente, si queremos agregarle un evento de “click” lo hacemos como se vio en el capítulo 3.

Para crear un conjunto de botones y tengan como función seleccionar alguna opción (como los radiobutton), podemos hacer lo siguiente:

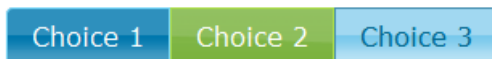
Código HTML:

```
<div id="radio">
  <input type="radio" id="radio1" name="radio" /><label for="radio1">Choice 1</label>
  <input type="radio" id="radio2" name="radio" checked="checked" /><label for="radio2">Choice 2</label>
  <input type="radio" id="radio3" name="radio" /><label for="radio3">Choice 3</label>
</div>
```

Código JavaScript:

```
$( "#radio" ).buttonset();
```

Resultado:



Y así tendremos un conjunto de botones con de tipo radiobutton

5.2.2.4 Datepicker

Este es uno de los componentes que mas he usado y de los que mas utilidad les encuentro (enseguida para mi el mas útil son los buttons y dialogs), este componente por lo regular se aplica en un input tipo text y este actúa al dar click sobre la caja de texto.

Codigo HTML:

```
<input type="text" class="calendario"/>
```

Código JavaScript:

```
//calendario  
$("#calendario").datepicker();
```

Resultado:



Podemos agregar diferentes opciones a este calendario, como por ejemplo permitir cambiar el mes y el año por medio de un comboBox, o que muestre un panel con botones, etc.

Código JavaScript:

```
//calendario  
$("#calendario").datepicker({  
  showButtonPanel:true,  
  changeMonth:true,  
  changeYear:true,  
  numberOfMonths:2,  
  showWeek:true  
});
```

Resultado:



5.2.2.5 Dialog

Con este componente estoy seguro que es el que mas vas a utilizar a la hora de desarrollar tu aplicación web, los diálogos (diálogos simples o diálogos modales) son de gran utilidad al momento de querer mostrar alguna advertencia, mensaje informativo o de error al usuario; y no solo eso, si no también mostrar formularios, imágenes, o cualquier contenido HTML que se te ocurra dentro de un dialogo modal (o simple), comencemos con este widget:

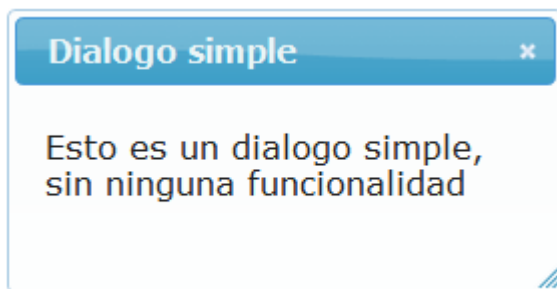
Código HTML:

```
<div id="dialogo" title='Dialogo simple'>
  <p>
    Esto es un dialogo simple, sin ninguna funcionalidad
  </p>
</div>
```

Código JavaScript:

```
//dialogo
$("#div#dialogo").dialog();
```

Resultado:



Como pueden ver el dialogo que se creo aparece en el centro de nuestra pagina, pero este puede ser arrastrado sobre cualquier elemento que actualmente tenga nuestra pagina.

Un dialogo mas funcional

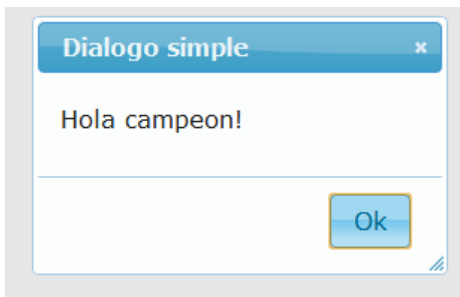
Código HTML:

```
<h4>Dialogos </h4>
<button id="btn1">Dame clic</button>
<div id="dialogo" title='Dialogo simple'>
  <p>
    Hola campeon!
  </p>
</div>
```

Codigo JavaScript:

```
//dialogo
$("#div#dialogo").dialog({
  autoOpen:false,
  show:"blind",
  hide:"explode",
  modal:true,
  buttons:{
    Ok:function(){
      $(this).dialog('close')
    }
  }
});
$("#btn1").click(function(){
  $("#div#dialogo").dialog('open')
  return false
})
```

Resultado:



Ahora tenemos un dialogo modal informativo, que tiene una animación tipo “blind” al mostrarse el dialogo, y al cerrar el dialogo tiene una animación “explode” (mas delante se verán todas las animaciones que proporciona jQuery ui). Al momento de crear el dialogo estamos estableciendo todas las propiedades que se ven en la imagen, si queremos agregar mas botones solo hay que separar los elementos con comas, como se hace regularmente con todas las propiedades de objetos de JavaScript.

Dialogo con un formulario

Codigo CSS:

Agregamos este fragmento de código CSS dentro de la etiqueta <head>

```
<style type="text/css">
  div#dialogForm label,div#dialogForm input { display:block; }
  input.text { margin-bottom:12px; width:95%; padding: .4em; }
  fieldset { padding:0; border:0; margin-top:25px; }
</style>
```

Código HTML:

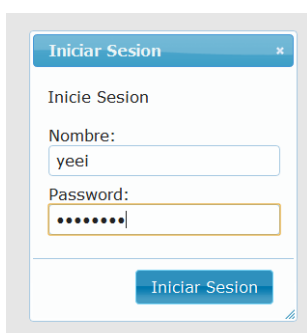
```
<div id="dialogForm" title='Iniciar Sesión'>
  <p>Inicie Sesión</p>
  <form>
    <fieldset>
      <label for="nombre">Nombre:</label>
      <input type="text" name="nombre" id="nombre" class="text ui-widget-content ui-corner-all"/>
      <label for="password">Password:</label>
      <input type="password" name="password" id="password" class="text ui-widget-content ui-corner-all" />
    </fieldset>
  </form>
</div>
<button id="btn2">Iniciar Sesión</button>
```

Aquí simplemente estamos agregando el div que será el dialogo y este puede contener todo tipo de código HTML, y este se mostrara en el dialogo. Aquí agregamos unas clases adicionales a los elementos <input>, las clases que comienzan con “ui” son clases que vienen incluidas en el archivo CSS que descargamos de jQuery UI, estos son estilos de paginas que cambian solo un poco nuestros elementos HTML dependiendo del tema que descargamos de jQuery UI, la clase text la definimos arriba en el apartado <style>.

Código JavaScript:

```
//dialog form
$("#div#dialogForm").dialog({
  autoOpen:false,
  modal:true,
  show:"blind",
  hide:"explode",
  buttons:{
    "Iniciar Sesión":function(){
      $(this).dialog('close')
    }
  }
});
//evento click que abra el dialogo
$("#button#btn2").click(function(){
  $("#div#dialogForm").dialog('open');
});
$("#button").button(); //hacemos todos
                        //los elementos
                        //<button> botones jQueryUI
```

Resultado:

A screenshot of a jQuery UI dialog box titled "Iniciar Sesión". The dialog has a blue header bar with the title and a close button. The main content area is white and contains the text "Inicie Sesión". Below this, there are two labels: "Nombre:" and "Password:". The "Nombre:" label is followed by a text input field containing the text "yeei". The "Password:" label is followed by a password input field containing seven dots. At the bottom of the dialog, there is a blue button labeled "Iniciar Sesión".

5.2.2.6 Tabs

Para terminar con los widgets mas usados en jQuery (porque hay algunos mas) tenemos el widget “tabs”, este widget nos permite crear las famosas tabs, permitiendo tener varias secciones y poder cambiar entre ellas con diferentes tabs, veamos como se hace:

Codigo HTML:

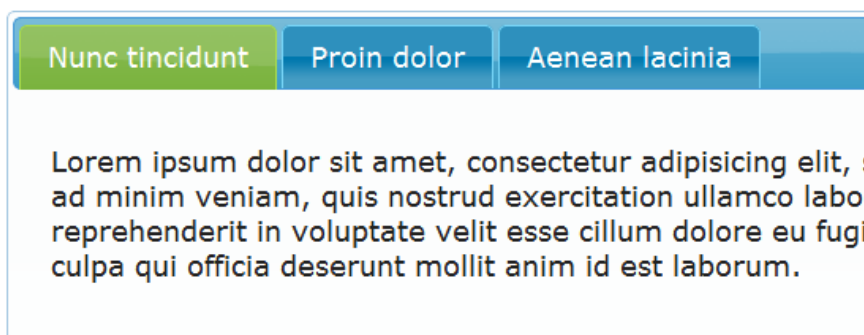
```
<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
    <li><a href="#tabs-3">Aenean lacinia</a></li>
  </ul>

  <div id="tabs-1">
    <p>
      Lorem ipsum dolor sit amet, consectetur adipisicing elit, se
    </p>
  </div>
  <div id="tabs-2">
    <p>
      sed do eiusmod tempor incididunt ut labore et dolore magna
    </p>
  </div>
  <div id="tabs-3">
    <p>
      Duis aute irure dolor in reprehenderit in voluptate velit e
    </p>
  </div>
</div>
```

Codigo JavaScript:

```
<script type="text/javascript">
  $(document).ready(function() {
    //documento listo
    $("#div#tabs").tabs();
  });
</script>
```

Resultado:



5.3 Efectos

Jquery por si solo ya ofrece una serie de efectos como lo vimos en el capitulo 4, jQuery UI ofrece aun mas efectos (que algunos ya se vieron en el capitulo anterior), los posibles efectos son: 'blind', 'bounce', 'clip', 'drop', 'explode', 'fold', 'highlight', 'puff', 'pulsate', 'scale', 'shake', 'size', 'slide', 'transfer'.

Veamos un ejemplo sencillo con un elemento <p> (recuerden, esto se puede aplicar a cualquier elemento HTML).

Codigo JavaScript:

```
$("p").click(function(){  
    $(this).hide("explode", "slow").delay(500).show("scale", "slow");  
})
```

Y si corremos nuestra página, al darle clic a un elemento <p> correrá inmediatamente el efecto “explode” desapareciendo de una manera lenta, y como lo hemos hecho anteriormente, solo para mostrar el ejemplo hacemos un delay de 500 milisegundos y posteriormente mostramos nuevamente el elemento pero ahora con un efecto “scale” de una manera lenta también, hay que probar todos los efectos para ver cual es el que realmente nos gusta.

Básicamente esto es lo mismo que la sección 4.2 pero aquí jQuery UI agrega una gran variedad de efectos. Y si solo se quiere correr un efecto y no desaparecerlo (porque algunos efectos son de “shake” o “highlight”) podemos usar la función effect() y usa lo mismos parámetros que las funciones anteriores, veamos un ejemplo con un dialogo:

Creemos un dialogo de la misma manera que lo hicimos anteriormente pero aplicamos este código para darle un efecto al darle click o que responda a algún evento

Código JavaScript:

```
$("div#Dialogo").dialog(); //se crea el dialogo  
$("button").click(function(){ //se crea el evento  
    $("div#Dialogo").effect("shake", "slow") //se aplica el efecto  
})
```

Corramos nuestra página y veremos como reacciona

6 jQuery Plugins

Existen una gran variedad, pero una gran variedad de plugins para jQuery que realmente nos facilita el crear muchas animaciones y que nuestros elementos sean interactivos y muy agradables al usuario y de una manera muy fácil.

En este capítulo veremos un par de ellos, como por ejemplo crear una galería fotográfica y un famoso “carrusel” que en la mayoría de las páginas web podemos encontrar de estos.

6.1 Step Carousel

Este tipo de plugin nos permite tener una serie de paneles que pueden contener cualquier tipo de etiquetas HTML (como texto, imágenes, etc) y esta serie de paneles tendrán una transición pensando en un carrusel, estos podrán ir avanzando automáticamente o con botones (siguiente, atrás o de todas las opciones).

Para empezar hay que incluir el plugin, lo descargamos desde la página oficial, pero puede buscarlo en los recursos de este manual, el archivo se llama “carousel.js”

```
<script type="text/javascript" src="js/jquery-7.js"></script>  
<script type="text/javascript" src="js/plugins/carousel.js"></script>
```

Es importante tener siempre la versión de jQuery más actualizada, en este caso estoy usando la versión 1.7 (con versiones más viejas este plugin no funciona en su totalidad). Una vez incluido el archivo JavaScript del plugin hay que seguir las especificaciones que nos dice el creador, o sea hay que crear la estructura HTML en la cual fue hecho y posteriormente configurar el plugin a nuestro código HTML. Así es como funcionan la mayoría de los plugins de jQuery; incluimos la librería de jQuery y un archivo .js con el plugin, creamos la estructura HTML que se te pide (Siempre viene en el sitio web del creador) y posteriormente se configura el plugin.

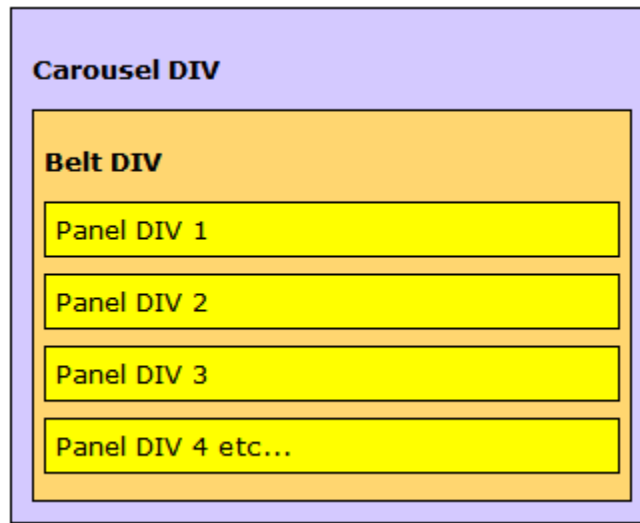
6.1.1 Estructura HTML

Tal vez aún no entienden muy voy lo que vamos a hacer, pero con una facilidad verán lo que vamos a crear. Esta parte también incluye CSS, y muchos plugins como estos exigen algunas propiedades CSS para que funcione correctamente.

Primero que nada vamos a crear la siguiente estructura HTML:

```
<div id="mygallery" class="stepcarousel">  
  <div class="belt">  
  
    <div class="panel">  
        
    </div>  
    <div class="panel">  
        
    </div>  
  </div>  
</div>
```

Esta serie de divs son fáciles de entender, y las explicare de manera simple con el siguiente grafico:



Vamos a tener un div principal que será nuestro “visor”, nuestra ventana para ver el Panel actual; es decir, nosotros tenemos muchos paneles en nuestro div Belt (el contenedor) pero nosotros vamos a ver un Panel a la vez, pero en realidad van a existir muchos, lo que pasara es que el div Belt se ira desplazando para que podamos ver los demás paneles, etc.

6.1.2 CSS

Para que todo lo mencionado anteriormente pueda ser posible, necesitamos configurar las características de los elementos HTML :

```
<style type="text/css">
  .stepcarousel{
    position: relative; /*NO TOCAR*/
    overflow: scroll; /*NO TOCAR*/
    width: 1000px; /*Altura de nuestro "visor"*/
    height: 720px; /*Ancho de nuestro "visor"*/
    border: 10px solid black;
  }
  .stepcarousel .belt{
    position: absolute; /*NO TOCAR*/
    left: 0;
    top: 0;
  }
  .stepcarousel .panel{
    float: left; /*NO TOCAR*/
    overflow: hidden; /*Oculta todo lo que sobre sale del panel*/
    margin: 10px; /*Margen del panel*/
    width: 980px; /*Altura del panel */
    height: 700px;
  }
</style>
```

Ya tenemos casi todo listo, solo falta la configuración.

6.1.3 Configuración del Plugin

```
<script type="text/javascript">
  stepcarousel.setup({
    galleryid: 'mygallery', //id del carrusel
    beltclass: 'belt', //clase que contiene los paneles
    panelclass: 'panel', //clase del <div> que es el panel
    autostep: {
      enable:true,
      moveby:1,
      pause:3000
    }, //inicio automatico de 1 en 1
    panelbehavior: {
      speed:500,
      wraparound:true,
      wrapbehavior:'slide',
      persist:true
    },
    defaultbuttons: {
      enable: true,
      moveby: 1,
      leftnav: ['images/left_arrow.png',-80,80],
      rightnav: ['images/right_arrow.png',-40,80]
    },
    contenttype: ['inline'] //['inline'] o ['ajax','path_to_external_file']
  });
</script>
```

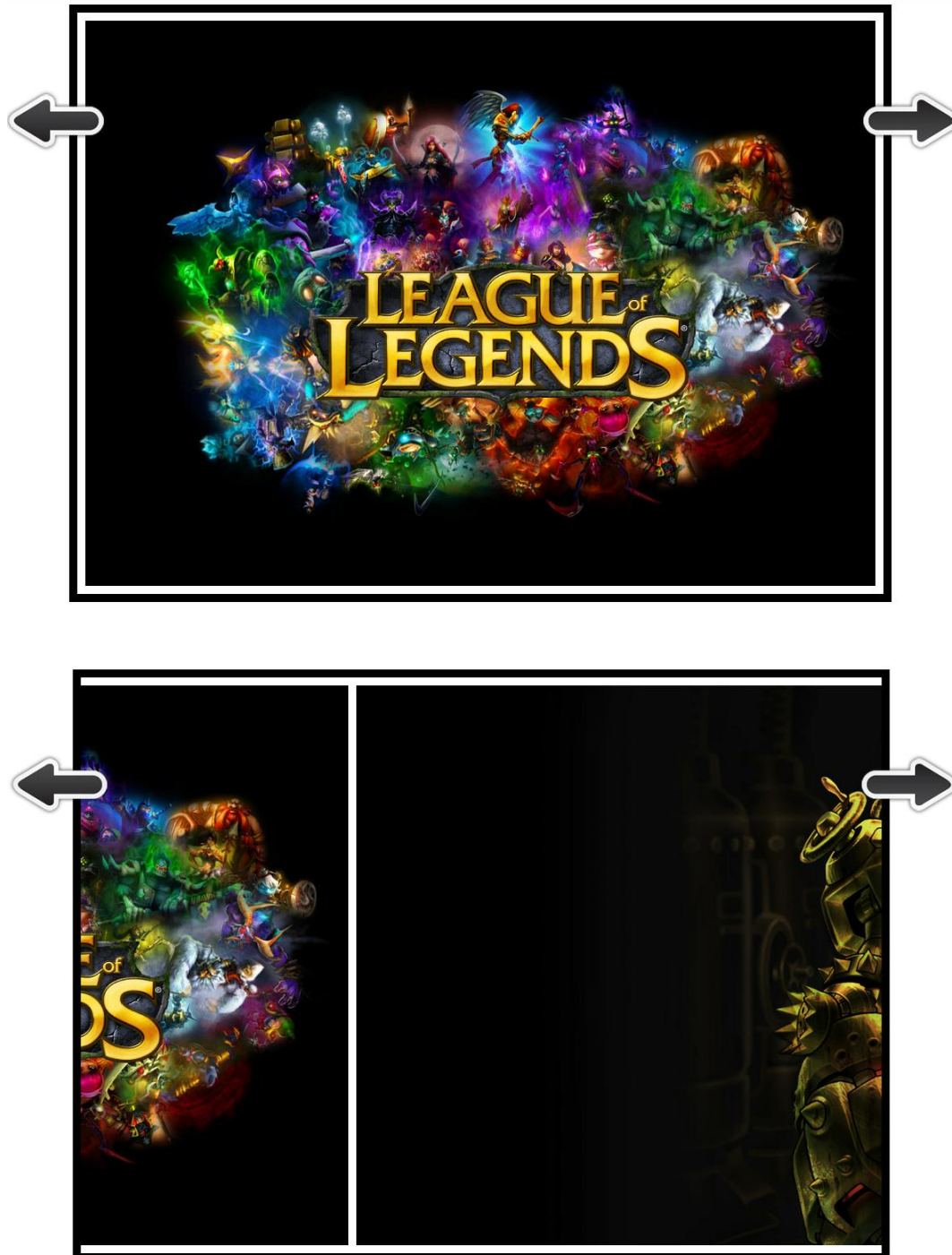
Las primeras 3 propiedades simplemente estamos indicando las clases y el id de nuestros elementos HTML en los que va a crear el carrusel. Seguido tenemos la propiedad “autoStep” que contiene a su vez mas propiedades que básicamente indica que iniciara la transición de paneles automáticamente de 1 en 1 cada 3 segundos.

Seguido sigue el comportamiento en “panelbehavior”: indicamos el tiempo que durara la transición de cada panel, “wrapAround” decide si al llegar al último panel se devolverá al primero, etc. También tenemos los botones de navegación “leftnav” y “rightnav”, simplemente le decimos la ruta de la imagen que representara las flechas de navegación, y posteriormente indicamos la posición, hay que hacer pruebas para comprender el “plano cartesiano” que se maneja.

Ya a grandes rasgos los nombres de las propiedades hablan por si solas, solo para terminar veremos el ultimo “contentType”, aquí decidimos de que forma vamos a cargar el contenido en nuestro carrusel, la forma que lo estamos haciendo es ‘inline’, significa que los “paneles” están dentro de la misma pagina donde esta el carrusel, esto significa que si tenemos mucho contenido esto se va a cargar desde un principio al iniciar la pagina, en cambio si usamos AJAX, se ira cargando conforme se valla solicitando el panel, si no se quieren ver mas que el primer panel, no se cargaran los demás (ahorra ancho de banda y muchas cosas de esas). En fin para usar “AJAX” no tenemos mas que ponerle en vez de [‘inline’] -> [‘ajax’,‘contenido.html’] y listo, en contenido.html

debe de estar todo el código HTML con los paneles (de la misma forma que lo hicimos arriba, la única diferencia es no ponerlo en la pagina principal, si no nomas en 'contenido.html').

Si hemos hecho todo conforme hemos visto el resultado será el siguiente (NOTA: en los paneles y los elementos hay que poner las rutas de las imágenes que queramos, igual con las flechas):



Momento de una transición.

6.2 Galería fotográfica con “Glisse”

El Plugin anterior en realidad no es una galería fotográfica en si, sirve para muchas cosas y principalmente sirve para el inicio de un sitio web, dar a conocer información útil, noticias, etc. Glisse es un Plugin de jQuery que nos permite crear galerías fotográficas realmente atractivas, claro si tenemos los conocimientos y el talento de manejar CSS y darle un aspecto muy atractivo, pero jQuery nos ayuda a crear efectos y esa galería en pantalla completa o en modo normal como según se desee.

Ejemplo de galería con Glisse



Momento de la transición a otra fotografía

El proceso es similar al anterior, hay que bajar los archivos JavaScript, pero en este caso incluye archivos CSS (como con jquery UI), veamos.

6.2.1 Estructura HTML

Para manejar este plugin es muy pero muy simple, en todos los elementos que nuestro sitio tenga (que obviamente son para nuestra galería fotográfica) solo hay que seguir con los atributos que se nos piden:

```
<div id="content">
  <ul class="gallery">
    <li></li>
  </ul>
</div>
```

Primero que nada tenemos la clase, esta clase la usaremos para aplicar el plugin (lo haremos mas adelante), seguido tenemos el titulo de la imagen, esto aparecerá al momento de iniciar la presentación de las fotos. El atributo rel='group1' va a ser igual en todas las imágenes, ya que por lo regular vamos a dar siguiente para visualizar las siguientes imágenes, esto es para hacer grupos y todos los elementos que tengan ese grupo aparecerán en la misma galería.

El atributo src indica la imagen que se vera (en pequeño), osea debe de ser un thumbnail (la imagen en miniatura) y en **data-glisse-big** debe de ir el enlace de la imagen verdadera de tamaño grande, ya que esta será la que visualizaremos al dar clic en el thumbnail.

NOTA: el id "content" y la clase "gallery" son opcionales, estos vienen exclusivamente en este ejemplo por los estilos de pagina que se usaran, se pueden modificar al gusto

6.2.2 CSS

Como comente anteriormente, este plugin cuenta con un archivo CSS y un archivo JavaScript, estos los puedes encontrar aquí <http://glisse.victorcoulon.fr/>, de igual manera vendrán incluidos en este manual.

Primero que nada debemos de incluir el archivo CSS antes de los nuestros:

```
<link rel="stylesheet" href="css/glisse.css"/>
```

Posteriormente debemos de agregar los estilos de pagina con los estilos de nuestra galería, dado que este plugin usa CSS3 les recomiendo que usemos los estilos de pagina que el creador del plugin nos proporciona, si conocemos muy bien CSS3 podemos crear (o CSS normal) los estilos del visor de la presentación. Por lo tanto podemos descargar esos estilos en el siguiente enlace <http://glisse.victorcoulon.fr/example-1/css/app.css?1> (en los recursos de este manual vendrán los estilos de igual manera)

En mi caso este archivo css es el siguiente:

```
<link rel="stylesheet" href="css/glisse.css"/>
<link rel="stylesheet" href="css/cap6_2.css"/>
```

6.2.3 Configuración

Para configurarlo primero que nada tenemos que estar seguros que ya tenemos nuestras imágenes, con los 2 estilos de paginas requeridos, el plugin Glisse y con la librería jQuery mas actualizada (ojo, es necesaria la versión 1.6 o superior)

Entonces ya solo resta iniciar la configuración:

```
<script src="js/jquery-7.js"></script>
<script type="text/javascript" src="js/plugins/glisse.js"></script>
<script>
    $(document).ready(function () {
        $('.photos').glisse({
            changeSpeed: 550,
            speed: 500,
            effect: 'roll',
            fullscreen: false
        });
    });
</script>
```

El selector “photos” es la clase que tiene todas y cada una de las imágenes que tendrán esta característica de galería fotográfica. El Plugin es muy simple, solo lo único laborioso es trabajar con CSS.

Opciones del Plugin:

- **changeSpeed** – duración de la transición entre 2 fotografías
- **speed** – duración de abrir/cerrar una fotografía
- **fullscreen** – pantalla completa
- **effect** – efecto de transición

Efectos validos:

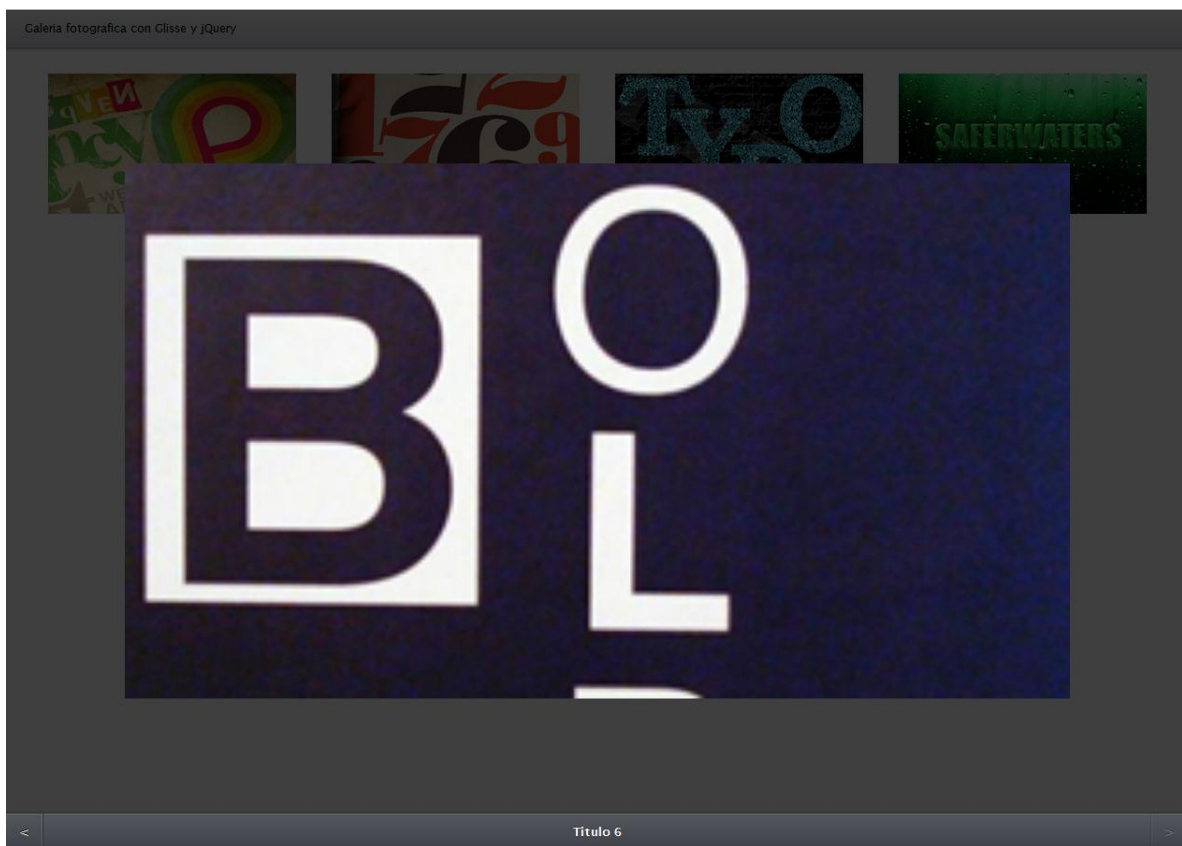
- bounce
- fadeBig
- fade
- roll
- rotate
- flipX
- flipY

En resumen tenemos:


```

<html>
<head>
<title>Galeria Fotografica :: Capitulo 6 parte 2</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<link rel="stylesheet" href="css/glisse.css"/>
<link rel="stylesheet" href="css/cap6_2.css"/>
<script src="js/jquery-7.js"></script>
<script type="text/javascript" src="js/plugins/glisse.js"></script>
<script>
    $(document).ready(function () {
        $('.photos').glisse({
            changeSpeed: 550,
            speed: 500,
            effect: 'roll',
            fullscreen: false
        });
    });
</script>
</head>
<body>
<div id="navbar">
    <p>Galeria fotografica con Glisse y jQuery</p>
</div>
<div id="content">
<ul class="gallery">
<li></li>
<li></li>
<li></li>
<li></li>
<li></li>
<li></li>
</ul>
</div>
</body>
</html>

```



Trabajos citados

Drive, D. (s.f.). Obtenido de <http://www.dynamicdrive.com/dynamicindex4/stepcarousel.htm>

jQuery. (s.f.). Obtenido de <http://archive.plugins.jquery.com/project/stepcarouselviewer>

Murphey, R. (s.f.). *Fundamentos de jQuery*. Obtenido de <http://librojquery.com/>

UI, j. (s.f.). Obtenido de <http://jqueryui.com/demos/>

W3Schools. (s.f.). Obtenido de http://www.w3schools.com/jquery/jquery_events.asp