Taller de jQuery

Ejercicios prácticos para aprender jQuery y la programación aplicaciones web enriquecidas del lado del cliente. Básicamente se trata de una serie de plugins de jQuery explicados que nos sirven de repaso y como biblioteca de scripts.

Autores del manual

Este manual ha sido realizado por los siguientes colaboradores de DesarrolloWeb.com:

Miguel Angel Alvarez
Director de DesarrolloWeb.com
http://www.desarrolloweb.com
(16 capítulos)

Eduard Nebot
(1 capítulo)

Introducción al taller de jQuery

Notas a los lectores sobre el Taller de jQuery que publicamos en DesarrolloWeb.com, para aprender jQuery practicando con la realización de diversos plugins.

Una vez más, dedicamos un nuevo manual a una de las tecnologías más actuales en el mundo del desarrollo de páginas web, el framework Javascript jQuery. En este caso se trata de un manual práctico, para personas que ya ya tengan algunos conocimientos sobre el framework y que quieran seguir aprendiendo por la práctica y reforzando lo que ya saben.

Este prólogo sirve para explicar nuestras intenciones en este nuevo taller de jQuery y orientar a los lectores nuevos que quieran obtener referencias sobre aquellos conocimientos que se deben adquirir antes de poder entender todos los ejemplos. Si ya tienes cierto dominio sobre el framework Javascript y conoces bien los recursos que existen en DesarrolloWeb.com sobre él, puedes simplemente saltarte este capítulo.

Objetivos del Taller de jQuery

En este taller deseamos aportar diversos ejemplos útiles para el desarrollo de páginas web enriquecidas del cliente, con dos intenciones. Por un lado deseamos seguir formando a las personas que ya tienen algún conocimiento de jQuery, esta vez por la práctica, de modo que puedan aplicar toda la teoría explicada en el Manual de jQuery. Por otra parte queremos generar una pequeña biblioteca de scripts jQuery, en forma de plugins, que cualquiera podría adaptar e integrar en su página web para implementar funcionalidades habituales en las páginas web.

Así pues, en este taller encontrarás una serie de artículos prácticos sobre cómo hacer diversos plugins de jQuery, que te ayudarán a seguir aprendiendo jQuery, reforzar los conocimientos que tengas y conocer las maneras correctas de encarar la resolución de problemas de una manera óptima.

Además, tendrás acceso a diversos códigos que permitirán desarrollar pequeñas funcionalidades requeridas en páginas web. Nuestro objetivo no es hacer scripts muy completos, que cubran todas las necesidades que puedas llegar a tener en cualquier proyecto, sino mostrar resoluciones simples a problemas habituales, que podrías con poco esfuerzo adaptar a tu proyecto.

Dado que muchas de las funcionalidades más requeridas ya están implementadas en diversos plugins desarrollados por terceras personas, otro de los objetivos de este taller es explicar cómo utilizar esos plugins de jQuery publicados en Internet, que hayamos probado y podamos recomendar a los lectores.

Referencias para seguir este taller de jQuery

Este taller de jQuery ya da por sabidos la mayoría de los conocimientos de este framework, por lo que recomendamos especialmente leer el Manual de jQuery publicado en DesarrolloWeb.com. Sobre todo, sería interesante seguir los capítulos que van hasta la creación de Plugins en jQuery y también las reglas para el desarrollo de plugins.

Sobra decir también que se requieren unos conocimientos profundos de Javascript y unas nociones básicas sobre programación orientada a objetos.

Así que, sin más preámbulos, os dejamos con estas prácticas en jQuery que esperamos os parezcan ilustradoras y fáciles de asimilar y aplicar en vuestro día a día.

Artículo por Miguel Angel Alvarez

Vídeo: Caja de diálogo dinámica con Photoshop, CSS y jQuery

Videotutorial en el que creamos una interfaz de usuario dinámica, desde el diseño con Photoshop, su maquetación CSS y los dinamismos con Javascript y jQuery.

Esta práctica que vamos a mostrar en vídeo toca tantas partes del desarrollo de páginas que es difícil resumirla en pocas palabras. Podríamos decir, en líneas generales, que vamos a crear una interfaz de usuario parecida a una caja de diálogo, donde mostrar mensajes como respuesta a las acciones del usuario. También podríamos ser más concretos y decir que vamos a realizar una capa flotante con esquinas redondeadas y borde semitransparente, maquetada con CSS y dinámica por medio de Javascript y jQuery.

La verdad es que, la llamemos como la llamemos, lo más sencillo es que se acceda al ejemplo para ver exactamente lo que vamos a crear.

Se trata, como decimos, de una práctica donde vamos a tocar diversos programas, lenguajes y tecnologías, como son Photoshop, HTML, CSS, Javascript y jQuery. Por eso requerirá para entenderla por completo conocimientos de varios ámbitos del desarrollo de páginas web. No obstante, esperamos que sea útil a un gran número de personas, pues aunque no se dominen del todo las materias, al menos sirva para ilustrar cómo se puede trabajar para crear interfaces de usuario personalizadas, que respondan a las acciones de los visitantes.

El videotutorial está dividido en tres partes, para crear vídeos de menos de 10 minutos que poder subir a YouTube. Cada una de las partes trata sobre una etapa de desarrollo, como el diseño, la maquetación y la programación de dinamismos en el cliente. Son los siguientes:

Vídeo 1: Diseño con Photoshop de la caja de diálogo

En este primer videotutorial estamos creando simplemente una serie de imágenes para poder luego construir la caja de diálogo. Empezamos creando una selección a la que le suavizamos los bordes para crear el fondo de la capa y luego un efecto de trazo por fuera de la capa al que le bajamos la opacidad para hacerlo semitransparente. Esa caja de diálogo luego la rompemos y extraemos tres imágenes para poder maquetarla en HTML y CSS. Las imágenes serán archivos png de 24 bit, para que nos guarde esa transparencia por canal alpha.

El primer vídeo acaba cuando tenemos las tres imágenes, de igual tamaño, listas para ser utilizadas. Y recuerda que si no dominas Photoshop, puedes aprender a través del Manual y los talleres de Photoshop que tenemos publicados en DesarrolloWeb.com.

Vídeo 2: Maquetación de la capa flotante con HTML y CSS

La segunda parte de esta práctica es la creación de la capa flotante en una página web. Ayudados de las imágenes que hemos creado en el anterior artículo, se definen una serie de etiquetas DIV donde albergar el contenido de la capa. Además a cada DIV le aplicamos estilos con CSS, principalmente para asignarle tamaños y los fondos de imagen, para la capa de arriba, de en medio y debajo. La segunda parte del videotutorial acaba asignando con el atributo position: absolute, top y left un lugar para la capa flotante. Una vez que la capa está encima de otros elementos de la página, podemos apreciar que el borde es semitransparente.

Para entender los pasos de este vídeo bastará con conocer un poco de CSS.

Videotutorial 3: Última parte para animar con Javascript y jQuery

La tercera parte de este vídeo es un poco más complicada que las anteriores y requiere por tanto unos conocimientos un poco mayores, en relación a programación del lado del cliente con Javascript. Para las personas que conozcan un poco de jQuery les parecerá algo extremadamente sencillo, pero puede que a los demás no tanto. No obstante, el ejemplo realmente puede mostrar a todos lo fácil que es crear unas sencillas instrucciones en Javascript para definir la interacción del usuario con la página.

En concreto veremos cómo hacer que la capa inicialmente esté oculta y luego por medio de unos enlaces que se muestre la capa, o que se oculte. Para ello, en el vídeo se mostrará cómo, mediante el framework Javascript jQuery, se pueden definir eventos para un par de enlaces, que tengan el código necesario para realizar acciones cuando se hace clic sobre ellos.

Para las personas que no dominen estos pasos recomendamos aprender un poco de Javascript y si ya lo conocen, aprender jQuery, del que además tenemos varios vídeos interesantes para aprender a dar los primeros pasos con jQuery.

Nota: Hemos subido los ejemplos realizados, así como el PSD de Photoshop creado a la red social de código fuente Github: http://github.com/desarrollowebcom/Caja-de-di-logo-din-mica-con-Photoshop--CSS-y-jQuery/downloads

Artículo por Miguel Angel Alvarez

Plugin jQuery: textarea con cuenta de caracteres

Segundo ejemplo de plugin práctico en jQuery para hacer textarea que lleva la cuenta de los caracteres escritos por el usuario.

Este es un taller práctico sobre jQuery que esperamos sirva para que las personas puedan continuar aprendiendo la manera de crear sus propios plugins. Como ya sabemos, los plugins son una manera óptima de programar tus scripts jQuery, ya que permitirán solucionar sus necesidades y además crear código limpio y reutilizable.

En los dos artículos anteriores ya estuvimos hablando de los Plugins en jQuery y de las reglas fundamentales para desarrollarlos. También vimos un primer ejemplo de un plugin sencillo, que espero nos haya abierto las miras y dado una idea sobre las posibilidades de construcción de componentes para páginas web. En este artículo continuaremos ofreciendo ejemplos para reforzar lo aprendido y para que las personas puedan familiarizarse aun más con el modo de creación de plugins en jQuery.

El objetivo del ejemplo que ocupará este artículo es la creación de un plugin para conseguir que un campo textarea de formulario informe en todo momento de caracteres que ha escrito el usuario. Es decir, vamos a hacer un método del objeto jQuery que servirá para decirle a los campos de texto textarea que se expandan para convertirse en un textarea que cuente los caracteres en una capa de texto de al lado.

Para tener una idea exacta de nuestros objetivos podemos ver el ejemplo en marcha que vamos a desarrollar.

Entendamos el plugin textarea con contador de caracteres

Para hacer los textareas que cuenten caracteres nosotros queremos hacer algo como esto en jQuery.

```
$("textarea").cuentaCaracteres();
```

Con eso queremos conseguir que a todos los textareas del documento HTML les aparezca una información al lado con el número de caracteres que tenga el textarea escrito dentro. Esa cuenta de caracteres debe mostrarse nada más cargarse la página y actualizarse cuando se escriba algo dentro. Todo eso se automatizará, para que no tengamos que hacer nada, salvo la anterior llamada al plugin.

Entonces, dentro del plugin tenemos que hacer varias cosas.

1. Un bucle con each para recorrer todos los objetos que pueda haber en el objeto jQuery que reciba el método para activar este plugin. Este paso es igual en todos los plugins.
2. Dentro de ese bucle podemos iterar con todos los elementos que haya en el objeto jQuery, que vamos a suponer son textareas. Vamos a crear un nuevo elemento DIV sobre la macha y vamos a iniciarlo con el texto de la cuenta de caracteres actual del textarea. Ese elemento creado "on the fly" lo añadiremos al cuerpo de la página, justo después de la etiqueta del textarea.
3. Además, haremos un evento, para que cuando el usuario escriba algo en el textarea, el texto con la cuenta de caracteres se actualice automáticamente.

Estos tres pasos serían un resumen del funcionamiento del plugin, cuyo código completo podemos ver a continuación.

```
//creo el plugin cuentaCaracteres
jQuery.fn.cuentaCaracteres = function() {
   //para cada uno de los elementos del objeto jQuery
   this.each(function(){
      //creo una variable elem con el elemento actual, suponemos un textarea
      elem = $(this);
```

```
      //creo un elemento DIV sobre la marcha
      var contador = $('<div>Contador caracteres: ' + elem.attr("value").length + '</div>');
      //inserto el DIV después del elemento textarea
      elem.after(contador);
      //guardo una referencia al elemento DIV en los datos del objeto jQuery
      elem.data("campocontador", contador);

      //creo un evento keyup para este elemento actual
      elem.keyup(function(){
         //creo una variable elem con el elemento actual, suponemos un textarea
         var elem = $(this);
         //recupero el objeto que tiene el elemento DIV contador asociado al textarea
         var campocontador = elem.data("campocontador");
         //modifico el texto del contador, para actualizarlo con el número de caracteres escritos
         campocontador.text('Contador caracteres: ' + elem.attr("value").length);
      });
   });
   //siempre tengo que devolver this
   return this;
};
```

El código está comentado para que se pueda entender mejor. Quizás nos pueda llamar más la atención la línea donde se utiliza la función jQuery para generar sobre la marcha un objeto jQuery con el campo DIV con el que vamos a seguir la cuenta. Vemos que a través del método attr() accedemos al value del textarea y con la propiedad length a su longitud en caracteres.

```
var contador = $('<div>Contador caracteres: ' + elem.attr("value").length + '</div>');
```

Luego también puede que nos llame la atención el funcionamiento del método data(), que nos permite almacenar y recuperar datos que se guardarán en el propio objeto jQuery de cada textarea.

Así guardo una referencia al objeto con la capa contador en el textarea, en un dato llamado "campocontador".

```
elem.data("campocontador", contador);
```

Y con este otro código en el evento recupero esa capa, pues luego en el evento tengo que cambiar el contenido con la cuenta de caracteres actualizada.

```
var campocontador = elem.data("campocontador");
```

Una vez creado el plugin, convierto todos los textareas en textareas-contador de caracteres, con este código:

```
$(document).ready(function(){
   $("textarea").cuentaCaracteres();
})
```

Eso es todo, pero quizás se vea más claro si vemos el código completo del ejemplo.

```
<html>
<head>
<title>Creando plugins en jQuery</title>
   <script src="../jquery-1.4.1.min.js"></script>
<script>

//creo el plugin cuentaCaracteres
jQuery.fn.cuentaCaracteres = function() {
   //para cada uno de los elementos del objeto jQuery
   this.each(function(){
      //creo una variable elem con el elemento actual, suponemos un textarea
      elem = $(this);
      //creo un elemento DIV sobre la marcha
      var contador = $('<div>Contador caracteres: ' + elem.attr("value").length + '</div>');
      //inserto el DIV después del elemento textarea
      elem.after(contador);
      //guardo una referencia al elemento DIV en los datos del objeto jQuery
      elem.data("campocontador", contador);

      //creo un evento keyup para este elemento actual
      elem.keyup(function(){
         //creo una variable elem con el elemento actual, suponemos un textarea
         var elem = $(this);
```

```
            //recupero el objeto que tiene el elemento DIV contador asociado al textarea
            var campocontador = elem.data("campocontador");
            //modifico el texto del contador, para actualizarlo con el número de caracteres escritos
            campocontador.text('Contador caracteres: ' + elem.attr("value").length);
        });
    });
    //siempre tengo que devolver this
    return this;
};
$(document).ready(function(){
    $("textarea").cuentaCaracteres();
})
</script>

</head>
<body>
<form>
        <textarea rows=5 cols=30 id="mitextarea">hola</textarea>
        <br>
        <br>
        <textarea rows=5 cols=30 id="otrotextarea">Otra cuenta...</textarea>
    </form>
</body>
</html>
```

Este ejemplo se puede ver en una página aparte.

> **Nota:** Si quieres ver más ejemplos prácticos de creación de plugins te recomiendo que leas el Taller de JQuery.

Artículo por Miguel Angel Alvarez

Plugin jQuery para hacer un Tip simple

Ejemplo de pluging en jQuery que nos permitirá implementar una funcionalidad de tip sobre elementos de la página.

Llegado a este punto del Manual de jQuery tenemos la base suficiente para trabajar con ejemplos interesantes y prácticos. Además. para dominar el esquema de trabajo típico con este framework Javascript, seguramente nos venga bien seguir practicando en la realización de ejemplos que pongan en práctica todo lo visto hasta el momento.

En artículos anteriores ya explicamos los plugins en jQuery y en este artículo continuaremos trabajando con ellos, pues son la manera más adecuada de hacer componentes para páginas web, con un código limpio y ordenado, con la ventaja que los podremos reutilizar en diversos proyectos, nuestros o incluso de otros desarrolladores.

En esta ocasión vamos a realizar un ejemplo de tip sencillo con jQuery, osea, un sistema para crear elementos "calientes" que, al pasar el ratón sobre ellos, muestren un mensaje con una explicación contextual, lo que habitualmente se conoce como "tip". Lo cierto es que anteriormente en este manual ya explicamos una manera de hacer un tip simple, cuando vimos los eventos de ratón mouseenter y mouseleave, con lo que la práctica actual se basará en el artículo visto anteriormente.

Sin embargo, el modo en el que vamos a desarrollar el tip tiene una una diferencia fundamental, con respecto a las explicaciones anteriores y es que antes no se utilizaban los plugins porque no los habíamos explicado todavía. Con ello, el ejemplo que veremos ahora implementado por medio de un plugin, será mucho más interesante porque ahorraremos código y podremos portarlo a otras webs sin mayor dificultad.

El ejemplo que vamos a ver a continuación, para hacernos una idea de los objetivos, lo hemos publicado en una página aparte.

Esquema de funcionamiento del plugin de tip

Para poder seguir este ejemplo con mayor facilidad, vamos a explicar los pasos que debe realizar este plugin específicamente, sin repetir lo que ya se comentó en el artículo de Reglas para hacer plugins en jQuery.

- Para llamar al plugin, recordemos, invocamos un método con el nombre del plugin, sobre elementos de la página web que tenemos en un objeto jQuery. Ese objeto jQuery puede tener uno o varios elementos de la página que hemos obtenido con un selector. El método del plugin que ejecutamos sobre el objeto jQuery, recibirá un texto como parámetro, que será lo que aparezca en la capa del tip.
- El plugin se encargará como primer paso de generar la capa del tip e insertarla en el código del documento. En el momento de insertarla no se verá todavía, porque las capas del tip tendrán el estilo CSS display:none.
- Definiremos un evento mouseenter sobre el elemento "caliente", es decir sobre aquellos elementos sobre los que invocamos el plugin. Ese mouseenter se encargará de mostrar la capa del tip y colocarla en un lugar próximo a donde se encuentra la puntera del ratón.
- Definiremos un evento mouseleave, también sobre los elementos "calientes", que se encargará de ocultar la capa del tip.

Eso es todo. Así que veamos el código del Plugin.

```
jQuery.fn.creaTip = function(textoTip) {
   this.each(function(){
      elem = $(this);
      var miTip = $('<div class="tip">' + textoTip + '</div>');
      $(document.body).append(miTip);
      elem.data("capatip", miTip);

      elem.mouseenter(function(e){
         var miTip = $(this).data("capatip");
         miTip.css("left", e.pageX + 10);
         miTip.css("top", e.pageY + 5);
         miTip.show(500);
      });
      elem.mouseleave(function(e){
         var miTip = $(this).data("capatip");
         miTip.hide(500);
      });
   });

   return this;
};
```

Lo importante ahora en este código es fijarse en la manera de crear el plugin. Como vemos, estamos asignando una función a la propiedad jQuery.fn.creaTip. Con ello realmente estamos extendiendo las funcionalidades del framework Javascript para que a partir de ahora los objetos jQuery dispongan de un método nuevo llamado "creaTip", que recibe un parámetro que es el texto que tiene que aparecer en el tip.

Como en todos los plugins jQuery, tenemos que hacer un recorrido a los elementos que pueda haber en el objeto jQuery que reciba este método nuevo. Esto se hace con this.each() y en ese recorrido tenemos que hacer todas las acciones necesarias para dotar de la funcionalidad nueva a esos elementos.

Ahora veamos las siguientes líneas de código.

```
elem = $(this);
```

Con esa línea creamos una variable elem a la que asignamos el valor $(this), es decir, el objeto jQuery con el que estamos trabajando.

```
var miTip = $('<div class="tip">' + textoTip + '</div>');
```

Con esta línea generamos sobre la marcha un elemento nuevo, que es una capa con la class CSS "tip" que guardamos en la variable miTip. Como texto en el div colocamos el contenido de la variable "textoTip". Este nuevo elemento se ha creado simplemente, pero aun no lo hemos metido en la página. En este momento simplemente se creó una variable miTip con el elemento en memoria.

```
$(document.body).append(miTip);
```

Con esta línea metemos en el cuerpo de la página el elemento DIV generado previamente. El elemento se colocará en el código de la página al final del todo, antes de cerrar el BODY. Aunque se haya insertado un elemento en la página, no se verá todavía, puesto que la capa insertada tiene la clase CSS "tip" y luego veremos que a esta clase le hemos metido un display:none para que no aparezca hasta que se pase el ratón por encima del elemento con tip.

```
elem.data("capatip", miTip);
```

Con esta línea hacemos una cosa interesante. Con elem.data() estamos guardando un dato en el elemento "caliente". El dato tiene el nombre "capatip" y el valor es miTip (la capa del tip). Con este paso estamos almacenando la capa del tip en el elemento que lo tiene que mostrar, así tendremos cada tip asociado a su propio elemento.

Luego en el código se definen dos eventos sobre el elemento "caliente", para mostar y ocultar el tip cuando el puntero del ratón entre y salga de él. Esas funciones para procesar los eventos son como las que pudimos conocer en el artículo en el que vimos el tip simple sin definir el plugin. Pero hay un detalle que merece la pena comentar. Veamos el código del evento mouseenter

```
elem.mouseenter(function(e){
   var miTip = $(this).data("capatip");
   miTip.css("left", e.pageX + 10);
   miTip.css("top", e.pageY + 5);
   miTip.show(500);
});
```

Como se puede ver, lo primero que se hace es:

```
var miTip = $(this).data("capatip");
```

Con esto estamos accediendo al dato "capatip" que habíamos guardado en el elemento que tiene el tip. En ese dato, como habíamos comentado, contiene la capa del tip que debemos mostrar. Luego, simplemente hacemos las acciones necesarias para situar esa capa del tip en un lugar próximo al puntero del ratón y mostrarla en la página con el método show().

Después del recorrido con each() y antes de acabar la función del plugin, tenemos que devolver el propio objeto jQuery que recibió el método, con return this.

Una vez definido el plugin, para generar los tips tenemos que invocarlo sobre los elementos de la página que deseemos que muestren un tip. Recordar de hacer este paso sólo cuando la página esté lista para recibir acciones:

```
$(document).ready(function(){
   $("#elemento1").creaTip("todo bien...");
   $("#elemento2").creaTip("Otra prueba...");
})
```

Código completo de la página del tip jQuery

Quizás venga bien ver el código completo de este ejemplo de creación de plugins en jQuery, donde podremos ver todo el proceso completo de implementación y puesta en marcha del tip.

```
<html>
<head>
<title>Trabajando con plugins en jQuery - Tip simple</title>
<style type="text/css">
.tip{
background-color: #ffcc99;
padding: 10px;
display: none;
position: absolute;
    width: 200px;
}
</style>
<script src="../jquery-1.4.1.min.js"></script>
<script>

jQuery.fn.creaTip = function(textoTip) {
   this.each(function(){
      elem = $(this);
      var miTip = $('<div class="tip">' + textoTip + '</div>');
      $(document.body).append(miTip);
```

```
      elem.data("capatip", miTip);

      elem.mouseenter(function(e){
         var miTip = $(this).data("capatip");
         miTip.css("left", e.pageX + 10);
       miTip.css("top", e.pageY + 5);
         miTip.show(500);
      });
      elem.mouseleave(function(e){
         var miTip = $(this).data("capatip");
         miTip.hide(500);
      });
   });

   return this;
};


$(document).ready(function(){
$("#elemento1").creaTip("todo bien...");
$("#elemento2").creaTip("Otra prueba...");
})
</script>

</head>
<body>
<h1>Trabajando con plugins en jQuery - tip simple</h1>

<div id="elemento1" style="background-color: #ccccff; padding: 5px;">Pasa el ratón por encima de
esta capa...</div>
<p>
Este texto es para poner <a id="elemento2" href="#">este enlace que también tiene tip</a>.
</p>

</body>
</html>
```

El ejemplo se puede ver en marcha en una página aparte.

> **Referencia:** Puedes aprendiendo a partir de este ejemplo nuevas utilidades de los plugins de jQuery como las opciones de configuración. Con ello conseguirás un plugin para hacer tips en jQuery más dinámico y sobre todo más configurable. Si te interesa, explora el artículo Plugin Tip con opciones en jQuery.

Esperamos que con este ejemplo y los explicados anteriormente en este manual de desarrollo web .com se haya comprendido la manera de hacer plugins en jQuery. Ahora sería interesante que cada uno pueda practicar por su parte lo aprendido, para realizar algún otro plugin simple que os ayude a afianzar los conocimientos.

Artículo por Miguel Angel Alvarez

Plugin Tip con opciones en jQuery

Un ejemplo de plugin en jQuery para hacer un sistema de tip más avanzado, que permite configurarse por medio de unas opciones en el plugin.

Hace poco tiempo publicamos un artículo en DesarrolloWeb.com sobre un plugin para mostrar un tip con un mensaje que aparecería en una capa al pasar el ratón sobre un elemento caliente. Eso es lo que llamamos un tip y lo explicamos en el artículo Plugin jQuery para hacer un Tip simple.

Ahora vamos a hacer una modificación en ese plugin para implementar una serie de opciones, que nos permitirán configurar de una manera más versátil el comportamiento del plugin. Las opciones que vamos a implementar serán las siguientes:

- Velocidad de la animación de mostrar y ocultar el tip
- Animación a utilizar para mostrar el tip
- Animación a utilizar para ocultar el tip
- Clase CSS para la capa del tip

Todas esas opciones se definen, junto con los valores por defecto que van a tomar, al crear el código del plugin. En el anterior artículo ya explicamos de manera general cómo funciona el sistema de options en plugins, que vamos a utilizar a continuación.

Comenzamos por especificar, con notación de objeto, las opciones de configuración por defecto para el plugin:

```
var configuracion = {
   velocidad: 500,
   animacionMuestra: {width: "show"},
   animacionOculta: {opacity: "hide"},
   claseTip: "tip"
}
```

Ahora veamos el inicio del código del plugin, donde debemos observar que en la función que define el plugin se están recibiendo un par de parámetros. El primero es el texto del tip, que necesitamos para crear la capa del tip (Este parámetro ya aparecía en el código del plugin del artículo anterior). El segundo son las opciones específicas para configurar el plugin.

```
jQuery.fn.creaTip = function(textoTip, opciones) {
   //opciones por defecto
   var configuracion = {
      velocidad: 500,
      animacionMuestra: {width: "show"},
      animacionOculta: {opacity: "hide"},
      claseTip: "tip"
   }
   //extiendo las opciones por defecto con las opciones del parámetro.
   jQuery.extend(configuracion, opciones);

   this.each(function(){
      //código del plugin
   });
});
```

Método jQuery.extend()

Quizás en este código, lo que más nos llame la atención sea el lugar donde extiendo las opciones por defecto definidas en la variable "configuracion", con las opciones específicas para el plugin concreto, recibidas por medio del parámetro "opciones".

```
jQuery.extend(configuracion, opciones);
```

Esta sentencia es una llamada al método extend() que pertenece a jQuery. Esta función recibe cualquier número de parámetros, que son objetos, y mete las opciones de todos en el primero. Luego, después de la llamada a extend(), el objeto del primer parámetro tendrá sus propiedades más las propiedades del objeto del segundo parámetro. Si alguna de las opciones tenía el mismo nombre, al final el valor que prevalece es el que había en el segundo parámetro. Si tenemos dudas con respecto a este método, leer el artículo jQuery.extend().

Así, podemos ver cómo con extend() las propiedades por defecto del plugin se combinan con las que se envíen en las opciones. Luego, en el código del plugin, podremos acceder a las propiedades a través de la variable configuración, un punto y el nombre de propiedad que queramos acceder.

```
configuracion.velocidad
```

Código completo del plugin tip con opciones

Veamos todo el código de nuestro primer plugin en implementar el sistema de opciones:

```
jQuery.fn.creaTip = function(textoTip, opciones) {
```

```
   var configuracion = {
      velocidad: 500,
      animacionMuestra: {width: "show"},
      animacionOculta: {opacity: "hide"},
      claseTip: "tip"
   }
   jQuery.extend(configuracion, opciones);

   this.each(function(){
      elem = $(this);
      var miTip = $('<div class="' + configuracion.claseTip + '">' + textoTip + '</div>');
      $(document.body).append(miTip);
      elem.data("capatip", miTip);

      elem.mouseenter(function(e){
         var miTip = $(this).data("capatip");
         miTip.css({
            left: e.pageX + 10,
            top: e.pageY + 5
         });
         miTip.animate(configuracion.animacionMuestra, configuracion.velocidad);
      });
      elem.mouseleave(function(e){
         var miTip = $(this).data("capatip");
         miTip.animate(configuracion.animacionOculta, configuracion.velocidad);
      });
   });

   return this;
};
```

Invocar al plugin con o sin las opciones de configuración

Para acabar, vamos a invocar al plugin del tip con opciones, pero lo vamos a hacer de dos modos, uno con las opciones por defecto y otro con opciones específicas.

Así se llamaría al plugin con las opciones por defecto:

```
$("#elemento1").creaTip("todo bien...");
```

En realidad le estamos pasando un parámetro, pero no son las opciones, sino es el texto que tiene que aparecer en el tip. Como no se indican opciones, ya que no hay segundo parámetro, se toman todas las definidas por defecto en el plugin.

Las opciones, según se puede ver en el código del plugin, se deberían enviar en un segundo parámetro cuando se llama al plugin, tal como se puede ver a continuación:

```
$("#elemento2").creaTip("Otra prueba...", {
   velocidad: 1000,
   claseTip: "otroestilotip",
   animacionMuestra: {
      opacity: "show",
      padding: '25px',
      'font-size': '2em'
   },
   animacionOculta: {
      height: "hide",
      padding: '5px',
      'font-size': '1em'
   }
});
```

Ahora hemos indicado varias opciones específicas, que se tendrán en cuenta al crear el plugin con este segundo código.

Para acabar, dejamos un enlace para ver el ejemplo en funcionamiento.

Artículo por Miguel Angel Alvarez

Plugin jQuery Fortaleza de una clave

Creamos un plugin con jQuery para mostrar la fortaleza de una clave escrita en un campo input de formulario.

Con la intención de practicar con el framework Javascript jQuery vamos a tratar un nuevo ejemplo de desarrollo de un plugin que podrá venirnos bien en los formularios de registro de usuario. Se trata de crear una utilidad para los campos password donde se escriben las claves, que permita mostrar al visitante la fortaleza de la clave a medida que la va escribiendo. Es un dinamismo que seguramente habréis visto en infinidad de webs y que fácilmente podemos imitar, para ayudar a nuestros usuarios a elegir claves seguras.

Antes de comenzar conviene decir que para seguir las explicaciones de esta práctica con jQuery tenemos que tener unos conocimientos básicos sobre el framework, que podremos obtener con la lectura del Manual de jQuery. Además, será extremadamente recomendable haber leído y asimilado la parte donde explicamos la creación de plugins, pues vamos a dar por sabidos esos conocimientos.

Así pues, veamos cuáles son los objetivos de este artículo. Tendremos un formulario con un campo input password. Con jQuery mostraremos dinámicamente un mensaje al lado del campo con la fortaleza de la clave que haya escrita. A medida que el usuario cambie el contenido del campo, se actualizará el mensaje del lado, mostrando la fortaleza de la nueva clave.

Código HTML con el campo password

Podemos comenzar viendo el código HTML necesario para hacer nuestro ejemplo.

```
<form>
<p>
Clave:<br />
<input type="password" name="clave" id="clave">
</p>
</form>
```

Como podemos ver es extremadamente sencillo. Lo único que tenemos es el campo input de tipo password, al que le hemos colocado un identificador para poder acceder a él desde Javascript.

Código del plugin jQuery para mostrar la seguridad de la clave

El plugin que vamos a ver contendrá básicamente dos partes, la primera en la que generaremos dinámicamente un elemento HTML al lado del campo password, donde aparecerá el mensaje que nos indique la fortaleza de la clave actual. Para facilitarnos las cosas, almacenaremos una referencia a ese elemento "mensaje" de fortaleza, dentro del propio objeto jQuery del campo input.

La segunda parte será donde definamos el evento de teclado necesario para conseguir que el mensaje de fortaleza de la clave se actualice dinámicamente, a medida que el usuario cambie el contenido del campo password. Dentro de este evento recuperaremos el objeto jQuery del elemento mensaje y actualizaremos su contenido para mostrar cómo de fuerte es la clave.

Para decidir en nuestro script lo segura que sea la clave en este ejemplo simplemente contaremos el número de caracteres de la clave. A mayor número de caracteres, más segura será la clave. Por tanto, el objetivo de este artículo es mostrar cómo podemos hacer la parte de jQuery para crear los elementos y eventos para producir los comportamientos dinámicos en el cliente, no tanto el análisis de la cadena de texto que haya en el campo password para decidir si la clave es segura o no.

> **Nota:** Para las personas que deseen mejorar este plugin agregando una función más inteligente, que ofrezca la fortaleza de la contraseña de una manera más precisa, recomendamos leer el artículo Script para informar de la seguridad de una clave, con Javascript.

Veamos entonces el código de este plugin.

```
jQuery.fn.fortalezaClave = function(){
```

```
   $(this).each(function(){
      elem = $(this);
      //creo el elemento HTML para el mensaje
      msg = $('<span class="fortaleza">No segura</span>');
      //inserto el mensaje en la página, justo después del campo input password
      elem.after(msg)
      //almaceno la referencia del elemento del mensaje dentro del campo input
      elem.data("mensaje", msg);

      elem.keyup(function(e){
         elem = $(this);
         //recupero la referencia al elemento del mensaje
         msg = elem.data("mensaje")
         //miro la fortaleza
         //extraigo el atributo value del campo input password
         claveActual = elem.attr("value");
         var fortalezaActual = "";
         //saco la fortaleza en función de los caracteres que tenga la clave
         if (claveActual.length < 5){
            fortalezaActual = "No segura";
         }else{
            if(claveActual.length < 8){
               fortalezaActual = "Medianamente segura";
            }else{
               fortalezaActual = "Segura";
            }
         }
         //cambio el texto del elemento mensaje para mostrar la fortaleza actual
         msg.html(fortalezaActual);
      });
   });
   return this;
}
```

La funcionalidad ha quedado implementada en el plugin y hemos comentado el código para que se pueda entender mejor. Gracias a este plugin disponemos de un nuevo método en los objetos jQuery llamado fortalezaClave() que se puede invocar sobre los elementos password (en realidad valdría sobre cualquier campo input de formulario) para mostrar la seguridad de una clave.

Ahora, para poner en funcionamiento esta utilidad, tenemos que hacer una llamada al método del plugin fortalezaClave() sobre el elemento password que tengamos en el formulario.

```
//cuando la página esté lista, cargo la funcionalidad del plugin sobre el elemento password
$(document).ready(function(){
   $("#clave").fortalezaClave();
});
```

Eso es todo, ahora sólo queda poner el enlace al ejemplo en marcha.

Modificación para tratar el caso que no haya nada escrito en el campo clave

Este ejemplo se puede modificar mínimamente para tratar el caso de que no haya escrito nada en el campo password del formulario, para no liar al usuario mostrando que "la clave no es segura" cuando realmente no ha escrito nada todavía en el campo password.

Además, henos cambiado la estructura de control para resolver la seguridad, que ya empieza a resultar un poco complicada con tanto if anidado que tendríamos que poner, para utilizar la estructura de control switch-case, que nos vendría mucho mejor tal como está este ejemplo.

El código del plugin quedaría de esta manera.

```
jQuery.fn.fortalezaClave = function(){
   $(this).each(function(){
      elem = $(this);
      msg = $('<span class="fortaleza">Escribe la clave</span>');
      elem.after(msg)
      elem.data("mensaje", msg);
```

```
      elem.keyup(function(e){
         elem = $(this);
         msg = elem.data("mensaje")
         claveActual = elem.attr("value");
         var fortalezaActual = "";
         console.log(claveActual.length)
         switch (claveActual.length){
            case 0:
               fortalezaActual = "Escribe la clave";
               console.log(aki)
               break;
            case 1:
            case 2:
            case 3:
               fortalezaActual = "La clave debe contener al menos 4 caracteres";
               break;
            case 4:
            case 5:
               fortalezaActual = "Clave no segura";
               break;
            case 6:
            case 7:
               fortalezaActual = "Clave medianamente segura";
               break;
            default:
               fortalezaActual = "Clave segura";
         }
         msg.html(fortalezaActual);
      });
   });
   return this;
}
```

Esta segunda variante la podemos ver también en marcha en este enlace.

Esperamos que haya resultado interesante e ilustrador este plugin jQuery para mostrar la seguridad de una clave.

Artículo por Miguel Angel Alvarez

Plugin jQuery para añadir campos en un formulario

Creamos un plugin con jQuery que sirve para añadir campos en un formulario HTML bajo solicitud del cliente.

En este artículo vamos a publicar una de las utilidades más solicitadas en Javascript, que consiste en un formulario que puede crecer dinámicamente en número de campos. Es decir, un formulario HTML que tiene una opción para que el visitante pueda añadir nuevos campos, dejando todo el procesamiento y la modificación de la página web del lado del cliente.

La utilidad que estamos relatando la haremos con el framework Javascript jQuery, que nos facilitará sensiblemente las cosas gracias a la cantidad de librerías útiles que contiene, para hacer este y cualquier otro script del lado del cliente. Además, crearemos todo el código con la estructura de plugin jQuery, para que cualquier persona pueda reutilizar esta utilidad en sus páginas web.

En artículos anteriores del Manual de jQuery ya explicamos muchas cosas que debemos de conocer sobre los plugins, con lo que recomendamos la lectura para poder entender el código que vamos a ver enseguida. Concretamente, recomendamos la lectura de los artículos donde comenzamos a tratar los plugins en jQuery.

Código HTML del formulario

La idea de este plugin es muy simple, pues únicamente necesitamos un pequeño sistema para añadir elementos HTML en la página bajo demanda del cliente. Tendremos un enlace en el formulario con el texto "Más campos" y al pulsarlo, simplemente se inyectará el código HTML para mostrar un campo nuevo en el formulario.

Entonces tendremos el código de un formulario, que contendrá ese enlace "mágico" que inserta campos en el formulario. Es indiferente como realizamos el formulario, es decir, pondremos los campos que necesitemos, ahora bien, los campos que vamos a insertar serán todos del mismo tipo, para facilitar las cosas.

Veamos el código HTML del formulario que necesitaremos para crear esta funcionalidad.

```
<form>
<p>
Nombre:<br>
<input type="Text" name="nombre">
</p><p>
Edad:<br>
<input type="Text" name="Edad">
</p><p>
Compra:<br>
<input type="Text" name="compra1">
<p>
<a href="#" id="mascampos">Más campos</a>
</p><p>
<input type=submit value="enviar">
</p>
</form>
```

Como se puede ver, es un formulario bastante simple, pero como decía, cada desarrollador pondrá los campos que necesite. La única cosa que deberemos tener con certeza es el código del enlace que insertará los campos al hacer clic sobre él. Fijémonos que ese enlace tiene un identificador para luego acceder a él desde jQuery.

Plugin para añadir campos en formulario

Ahora podemos ver el código del plugin jQuery que nos ocupa.

```
jQuery.fn.generaNuevosCampos = function(etiqueta, nombreCampo, indice){
   $(this).each(function(){
      elem = $(this);
      elem.data("etiqueta",etiqueta);
      elem.data("nombreCampo",nombreCampo);
      elem.data("indice",indice);

      elem.click(function(e){
         e.preventDefault();
         elem = $(this);
         etiqueta = elem.data("etiqueta");
         nombreCampo = elem.data("nombreCampo");
         indice = elem.data("indice");
         texto_insertar = '<p>' + etiqueta + ' ' + indice + ':<br><input type="text" name="' +
nombreCampo + indice + '" /></p>';
         indice ++;
         elem.data("indice",indice);
         nuevo_campo = $(texto_insertar);
         elem.before(nuevo_campo);
      });
   });
   return this;
}
```

Tal como hemos desarrollado este plugin, para funcionar debe recibir tres parámetros. El primero de ellos, "etiqueta", sirve para etiquetar el nuevo campo que se va a crear, osea, contendrá un texto que pondremos al lado de los campos que se van a crear dinámicamente. El segundo parámetro, "nombreCampo", sirve para darle un nombre interno para ese campo, que colocaremos en el atributo name del elemento INPUT HTML. El parámetro "indice" sirve para modificar el nombre del campo creado incluyendo un índice, de modo que ningún campo creado tenga el mismo nombre, lo que podría ser un

problema en nuestro formulario. Este índice será numérico y se incrementará en cada campo que se inserte, de modo que sea distinto en cada campo nuevo generado.

Entonces, dentro del plugin tenemos que hacer un par de cosas. Primero almacenar en el elemento HTML del enlace todos los datos que necesitamos para crear los campos, es decir, todos los datos que se enviaron como parámetros al llamar al plugin. Luego tendremos que implementar un evento clic sobre el enlace, que se encargará de realizar todas las acciones para generar dinámicamente e insertar el campo en el formulario.

El evento "click" del enlace tendrá que recuperar todos los datos almacenados en el enlace y luego generar el HTML del campo a insertar. Se generan los elementos por medio de la función jQuery y luego se insertan mediante el método before() del elemento enlace, para que el nuevo campo se coloque justo antes del enlace.

Para llamar al plugin utilizaremos un código Javascript parecido al siguiente:

```
$(document).ready(function(){
   $("#mascampos").generaNuevosCampos("Compra", "compra", 2);
});
```

Para ver y entender mejor este código se puede acceder al ejemplo en marcha, donde podremos pulsar el enlace para generar nuevos campos. Podremos ver como los campos que se generen tienen sus nombres (atributos name del campo input) únicos, para que se puedan acceder correctamente cuando se envíe el formulario y se reciban los datos.

Ver el ejercicio en marcha en una página aparte.

Artículo por Miguel Angel Alvarez

Plugin checkbox personalizado con jQuery

Un plugin en jQuery para hacer un campo de formulario checkbox pero con un diseño distinto, totalmente personalizable por el desarrollador.

A veces los campos de formulario que tenemos disponibles en HTML son un poco "aburridos", por decirlo de alguna manera. Quiero decir que son siempre iguales para todas las páginas y existen pocas opciones para configurar su aspecto, sobre todo en el caso de los elementos checkbox. Como diseñadores caprichosos, nosotros podríamos desear que nuestros checkboxes tuvieran un aspecto o color determinado, que haga mejor combinación con otros elementos de nuestro layout. Y estas son cosas que podemos conseguir fácilmente con un poco de jQuery.

En este artículo pretendemos hacer un plugin para crear campos checkbox personalizados, con las mismas funcionalidades de los checkbox normales, pero que tengan un aspecto configurable por el desarrollador. Para ello utilizaremos el modelo de creación de plugins en jQuery, de modo que haremos todo el trabajo en un plugin que cualquier persona podría utilizar en su sitio web y configurar los checkbox según sus preferencias.

Para seguir las explicaciones de este artículo necesitaremos saber acerca de la creación de plugins en jQuery y en concreto vamos a practicar con dos cosas que hemos aprendido recientemente:

- Gestión de opciones en plugins jQuery
- Crear Funciones y variables dentro de plugins jQuery

Para apreciar con exactitud cómo serán algunos ejemplos de checkbox que vamos a realizar, podemos echar un vistazo al ejemplo en marcha.

Personalización del plugin por medio de objeto de opciones

Podemos comenzar por ver el principio de código del plugin, donde estamos definiendo las variables de configuración por defecto y las estamos extendiendo con las variables de configuración definidas al invocarlo.

```
jQuery.fn.checkboxPersonalizado = function(opciones) {
   //opciones de configuración por defecto
```

```
   var conf = {
      activo: true,
      colorTextos: {
         activo: "#00f",
         pasivo: "#669"
      },
      textos: {
         activo: "",
         pasivo: ""
      },
      imagen: {
         activo: 'images/thumb_up.png',
         pasivo: 'images/thumb_down.png'
      },
      cssElemento: {
         padding: "2px 2px 2px 24px",
         display: "block",
         margin: "2px",
         border: "1px solid #eee",
         cursor: "pointer"
      },
      cssAdicional: {

      },
      nameCheck: ""
   };
   //Las extiendo con las opciones recibidas al invocar el plugin
   jQuery.extend(conf, opciones);

   this.each(function(){

      //CÓDIGO DEL PLUGIN

   });
   return this;
};
```

Tal como se puede ver, se han definido varias variables para configurar el objeto, que se dispondrán en un objeto que tenemos en la variable "configuracion". Entre las variables de configuración tenemos una llamada "activo" con un valor boleano para decidir si el elemento checkbox estaría o no seleccionado desde el principio. Tenemos una variable "colorTextos", para definir el color del texto cuando el elemento está activo y pasivo. También tenemos otra serie de configuraciones para los estados de activo y pasivo (seleccionado o no seleccionado), como la imagen que se tiene que mostrar al lado del texto.

Ahora veamos el código del plugin, lo que iría dentro de this.each(). Recordemos que cada variable creada aquí es accesible dentro de todo el bloque de código definido por las llaves del this.each(). Así mismo, las funciones declaradas aquí son accesibles desde cualquier parte de este bloque.

```
//variables locales al plugin
var miCheck = $(this);
var activo = conf.activo
//el elemento checkbox interno pero no visible
var elementoCheck = $('<input type="checkbox" style="display: none;" />');
//el nombre del checkbox puede ser configurado desde options o con el propio texto del campo
if(conf.nameCheck==""){
   elementoCheck.attr("name", miCheck.text());
}else{
   elementoCheck.attr("name", conf.nameCheck);
}
//inserto el checkbox en la página
miCheck.before(elementoCheck);
//aplico estilos que vienen en la configuración
miCheck.css(conf.cssElemento);
miCheck.css(conf.cssAdicional);

//si el elemento estaba marcado para estar activo
if (activo){
   //lo activo
   activar();
```

```
}else{
   //lo desactivo
   desactivar();
}

//defino un evento para el elemento
miCheck.click(function(e){
   //compruevo la variable activo, definida dentro del plugin
   if(activo){
      desactivar();
   }else{
      activar();
   }
   activo = !activo;
});

//función local en el plugin para desactivar el checkbox
function desactivar(){
   //cambio los estilos para el elemento a los marcados como pasivos
   miCheck.css({
      background: "transparent url(" + conf.imagen.pasivo + ") no-repeat 3px",
      color: conf.colorTextos.pasivo
   });
   //si hay un texto específico para cuando estaba pasivo
   if (conf.textos.pasivo!=""){
      miCheck.text(conf.textos.pasivo)
   }
   //desmarcho el checkbox interno que es invisible, pero que se envía como elemento de formulario.
   elementoCheck.removeAttr("checked");
};

function activar(){
   miCheck.css({
      background: "transparent url(" + conf.imagen.activo + ") no-repeat 3px",
      color: conf.colorTextos.activo
   });
   if (conf.textos.activo!=""){
      miCheck.text(conf.textos.activo)
   }
   elementoCheck.attr("checked","1");
};
```

El código está convenientemente comentado para que se pueda entender mejor. Pero lo que queremos mostrar en este caso es que hemos creado dos funciones dentro del código del plugin: activar() y desactivar(). Esas dos funciones, al estar dentro del bloque this.each(), se pueden acceder desde cualquier parte del plugin y comparten el mismo ámbito de variables que el propio plugin, luego podremos acceder desde ellas a cualquier variable definida en el bloque this.each().

Para que quede un poco más clara la estructura completa del plugin, coloco a continuación su código completo:

```
jQuery.fn.checkboxPersonalizado = function(opciones) {
   //opciones de configuración por defecto
   var conf = {
      activo: true,
      colorTextos: {
         activo: "#00f",
         pasivo: "#669"
      },
      textos: {
         activo: "",
         pasivo: ""
      },
      imagen: {
         activo: 'images/thumb_up.png',
         pasivo: 'images/thumb_down.png'
      },
      cssElemento: {
         padding: "2px 2px 2px 24px",
         display: "block",
         margin: "2px",
         border: "1px solid #eee",
```

```
            cursor: "pointer"
         },
         cssAdicional: {

         },
         nameCheck: ""
      };
      //Las extiendo con las opciones recibidas al invocar el plugin
      jQuery.extend(conf, opciones);

      this.each(function(){
         var miCheck = $(this);
         var activo = conf.activo
         var elementoCheck = $('<input type="checkbox" style="display: none;" />');
         if(conf.nameCheck==""){
            elementoCheck.attr("name", miCheck.text());
         }else{
            elementoCheck.attr("name", conf.nameCheck);
         }
         miCheck.before(elementoCheck);
         miCheck.css(conf.cssElemento);
         miCheck.css(conf.cssAdicional);

         if (activo){
            activar();
         }else{
            desactivar();
         }
         miCheck.click(function(e){
            if(activo){
               desactivar();
            }else{
               activar();
            }
            activo = !activo;
         });

         function desactivar(){
            miCheck.css({
               background: "transparent url(" + conf.imagen.pasivo + ") no-repeat 3px",
               color: conf.colorTextos.pasivo
            });
            if (conf.textos.pasivo!=""){
               miCheck.text(conf.textos.pasivo)
            }
            elementoCheck.removeAttr("checked");
         };

         function activar(){
            miCheck.css({
               background: "transparent url(" + conf.imagen.activo + ") no-repeat 3px",
               color: conf.colorTextos.activo
            });
            if (conf.textos.activo!=""){
               miCheck.text(conf.textos.activo)
            }
            elementoCheck.attr("checked","1");
         };
      });
      return this;
};
```

Invocar al plugin checkbox personalizado con jQuery

Ya que hemos hecho un checkbox personalizado, por un objeto de options, vamos a mostrar cómo se pueden crear varios tipos de checkbox con este código. Veamos el siguiente HTML:

```
<span class="ch">Seleccionar</span>
<span class="ch">Me interesa</span>
```

```
<span class="ch">Oooo</span>
<br>
<br>
<span id="otro">otro suelto</span>
```

Se puede apreciar que tenemos simples elementos SPAN. Por un lado tenemos 3 SPAN con la clase "ch" y por otro lado otro SPAN suelto con identificador "otro". Ahora veamos cómo los convertiríamos en campos de formulario checkbox personalizados:

```
$(".ch").checkboxPersonalizado();
```

Así crearíamos 3 checkbox, en los 3 primeros SPAN que tenían la class "ch". Estos checkbox personalizados se crearían con las opciones por defecto.

```
$("#otro").checkboxPersonalizado({
   activo: false,
   colorTextos: {
      activo: "#f80",
      pasivo: "#98a"
   },
   imagen: {
      activo: 'images/weather_cloudy.png',
      pasivo: 'images/weather_rain.png'
   },
   textos: {
      activo: 'Buen tiempo :)',
      pasivo: 'Buena cara ;)'
   },
   cssAdicional: {
      border: "1px solid #dd5",
      width: "100px"
   },
   nameCheck: "buen_tiempo"
});
```

En este segundo caso de invocación al plugin estamos convirtiendo en un checkbox personalizado el último SPAN, que tenía identificador "otro". En este segundo caso estamos utilizando multitud de variables de configuración específicas, que harán que el checkbox tenga un aspecto radicalmente diferente a los anteriores.

Para acabar, se puede ver el ejemplo en funcionamiento en una página aparte.

Artículo por Miguel Angel Alvarez

Primeros pasos con jQuery UI

Guía para comenzar jQuery UI, una librería de componentes listos para usar en páginas dinámicas del cliente, creada a partir del framework Javascript jQuery.

jQuery UI es un complemento que permite implementar componentes diversos para generar interfaces de usuario en páginas web, además de otras funcionalidades básicas para crear aplicaciones web enriquecidas. Como su propio nombre indica, está basado en el popular framework Javascript y podemos encontrar links, explicaciones, así como demos y descargas a partir del sitio web oficial de jQuery.

En DesarrolloWeb.com desde hace meses venimos explicando con detalle todo lo que los desarrolladores debe saber para utilizar jQuery y así realizar páginas enriquecidas en el cliente. Todo lo tenemos en el Manual de jQuery y además recientemente hemos publicado un Taller de jQuery con ejemplos prácticos sobre la realización de plugins con este framework y la utilización de componentes de terceros. Entre los scripts jQuery listos para usar que existen en Internet debemos de comenzar por los proporcionados en jQuery UI, ya que es la distribución oficial de componentes para la creación de interfaces de usuario avanzadas.

Así que comenzaremos publicando una serie de pasos iniciales necesarios para comenzar a usar jQuery UI en nuestras páginas web. Más tarde comenzaremos a implementar funcionalidades y componentes avanzados que nos proporcionan

estas librerías. Esta guía está basada en las propias explicaciones que se encuentran en el sitio de jQuery UI.

Revisar los demos de jQuery UI

Podemos empezar por observar la cantidad de funcionalidades listas para usar, así como componentes (widgets) para implementar interfaces de usuario. Todo esto lo encontramos en la página de demos de jQuery UI.

Quizás lo más interesante en la página de demos son los denominados widgets, donde podremos encontrar ejemplos de uso de estas librerías para crear los típicos menús "acordeón", calendarios para seleccionar fechas, ventanas de diálogo, interfaces de pestañas, etc.

Escoger los componentes en la página de descarga

Para comenzar realmente a usar jQuery UI necesitamos descargar las librerías, pero cabe decir que este sistema es bastante amplio, por lo que igual nos interesa descargar sólo una parte, para que no resulte demasiado pesado para las personas que visiten nuestro sitio web. Lo bueno es que la página de descargas de jQuery UI tiene un sistema para poder seleccionar sólo aquellos componentes que deseamos utilizar en nuestro sitio, con lo que la descarga se puede optimizar para únicamente contener aquellas cosas que realmente van a ser necesarias en nuestro sitio.

Podemos acceder a la página de descargas en: http://jqueryui.com/download

Componentes:
Aquí podemos seleccionar/deseleccionar los checkbox de cada uno de los componentes que forman parte de jQuery UI. Algunos tienen dependencias con otros, por lo que observaremos que, al seleccionar algunos componentes, automáticamente se seleccionan sus dependencias.

Aparte de los componentes a descargar, tenemos un par de cosas adicionales que marcar antes de la descarga.

Temas (Themes):
Por una parte tenemos que seleccionar el tema o aspecto que van a tener las interfaces de usuario de jQuery UI. Tenemos decenas de temas ya creados por el propio equipo de las librerías, con aspectos que varían bastante, sobretodo cromáticamente. Además, existe un generador de temas, que podemos utilizar para personalizar aun más el aspecto de las interfaces de usuario. El objetivo es que los menús y demás de este sistema tengan un aspecto que se integre bien con el diseño de nuestra página web.

Versión:
También tendremos que escoger la versión de jQueryUI que queremos utilizar, sabiendo que cada versión está preparada para funcionar con una versión del framework. En el momento de escribir este artículo existen dos versiones de jQuery UI que podemos seleccionar. La versión más moderna es la 1.8.1, que funciona sobre jQuery 1.4+ y la versión 1.7.3, más antigua, que funciona sobre jQuery 1.3.2.

Por último podemos apretar el botón de download para descargar un zip con todo lo que necesitaremos para usar jQueryUI.

Contenido de la descarga de jQueryUI

Una vez descargado nuestro paquete de jQuery UI obtendremos un archivo comprimido con diversos directorios y archivos, que tiene las siguientes carpetas principales:

Carpeta "css": En esta carpeta se encuentra el CSS y las imágenes para generar el tema o los temas escogidos. Realmente no tenemos por qué tocar esta carpeta en principio para nada, pero contiene cosas que serán fundamentales para que todo se muestre como deseamos.

Carpeta "development-bundle": Esta carpeta contiene una serie de materiales útiles para los desarrolladores que van a utilizar estas librerías. Veremos aquí páginas de documentación, ejemplos de uso y otras cosas interesantes. Nada de lo que hay aquí es necesario en principio para hacer funcionar los componentes jQuery UI, pero son materiales que podrán venirnos bien para aprender.

Carpeta "js": Aquí veremos los scripts Javascript de jQuery y jQuery UI necesarios para que todos los componentes funcionen. Esta carpeta contiene el archivo Javascript con el código de los componentes que habíamos seleccionado al hacer la descarga, además del archivo Javascript con el código de la versión del framework que funciona bien con las librerías.

Con el zip de descarga podremos ya comenzar a implementar alguno de los componentes de jQueryUI, pero esto es algo que veremos en el próximo artículo, segunda parte de la iniciación a jQuery UI.

Artículo por Miguel Angel Alvarez

Primeros pasos con jQuery UI, parte 2

Continuamos con la guía para crear tu primer script jQuery UI, para crear interfaces de usuario avanzadas.

En el anterior artículo comenzamos a introducir las librerías jQuery UI y ofrecimos una guía inicial para descargarlas e identificar los archivos y carpetas que hay en el archivo comprimido que obtendremos. En este artículo pasaremos a la práctica, mostrando cómo hacer una página web que utilice uno de los componentes disponibles en jQuery UI, que es el Datepicker o seleccionador de fecha por medio de un calendario.

Empezar a usar jQuery UI en una página web

Ahora que ya tenemos todo lo que nos hace falta para hacer nuestro primer ejemplo con jQuery UI, vamos a ponernos manos a la obra. En este paso vamos a crear un archivo de ejemplo en el que usaremos las funcionalidades del calendario para seleccionar una fecha.

Todos los materiales de la descarga los debemos copiar en el directorio de nuestro sitio web. Así que tendremos que tener en cuenta el lugar donde hemos dejado las carpetas que descargamos, principalmente las que son necesarias para que todo funcione "css" y "js".

El ejemplo que voy a realizar lo he colocado en una carpeta donde tengo los subdirectorios descargados de jQuery UI "css" y "js". Si en vuestro caso la estructura de carpetas es diferente, deberéis tener en cuenta la ruta desde este archivo de ejemplo hacia el archivo donde se encuentran las librerías jQuery UI y la hoja de estilos necesaria para definir el aspecto de los componentes, según el tema que hayamos seleccionado.

Como primer paso incluiremos el código siguiente:

```
<link type="text/css" href="css/le-frog/jquery-ui-1.8.1.custom.css" rel="Stylesheet" />
<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
<script type="text/javascript" src="js/jquery-ui-1.8.1.custom.min.js"></script>
```

La primera línea corresponde a un link con la hoja de estilos que queremos utilizar para el look & feel de las interfaces de usuario. En mi caso estoy utilizando el tema que se llama "le-frog", que tiene un bonito color verde. La segunda línea corresponde con la inclusión del código Javascript del framework jQuery. Por último, la tercera es el código de los componentes descargados de jQuery UI.

> **Nota:** Estas tres líneas tendremos que editarlas para indicar correctamente las rutas para acceder al tema CSS que deseamos implementar, así como para acceder a los archivos con el código Javascript jQuery y jQueryUI.

Invocar los métodos para generar las interfaces de usuario

Ahora que ya tenemos incluidos los archivos para poder usar jQuery UI ya sólo nos queda crear el Javascript necesario para que los componentes se generen en la página, así como configurarlos para que se comporten como nosotros necesitemos. De momento vamos a ver un ejemplo sencillo, en el que realizaremos un campo de texto para escribir una fecha. Al pulsar el campo de texto se mostrará un calendario para poder seleccionar la fecha que queramos. Este comportamiento está implementado por el componente llamado "Datepicker".

Para conseguir esto necesitamos crear un campo de texto, con la correspondiente etiqueta INPUT, colocada en el lugar donde deseemos que ese campo aparezca.

```
<input type="text" name="fecha" id="campofecha">
```

Fijémonos en el atributo "id" de este campo, que necesitaremos utilizar ahora al hacer el código Javascript necesario para convertirlo en un campo "Datepicker", que es el siguiente:

```
$("#campofecha").datepicker();
```

Esto es una llamada al método datepicker() sobre el objeto jQuery seleccionado por el identificador campofecha. Con eso hacemos que ese campo de texto se convierta en un seleccionador de fechas. Eso es todo!! con esa línea de código conseguimos toda la funcionalidad buscada!!

Quizás pueda sorprender lo sencillo y cómodo de usar de jQuery UI, pero claro que además los componentes tendrán varias maneras de configurarse para adaptarse a las necesidades más particulares, pero eso es algo que veremos en próximos artículos.

De momento, antes de acabar, colocamos aquí el código de una página completa donde estamos usando el componente Datepicker de jQuery UI:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"
>
<html lang="es">
<head>
<title>Seleccionar fecha con jQuery UI</title>
   <link type="text/css" href="css/le-frog/jquery-ui-1.8.1.custom.css" rel="Stylesheet" />
   <script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
   <script type="text/javascript" src="js/jquery-ui-1.8.1.custom.min.js"></script>
<script>
$(document).ready(function(){
   $("#campofecha").datepicker();
})
</script>
</head>
<body>

<form>
   Fecha: <input type="text" name="fecha" id="campofecha">
</form>

</body>
</html>
```

Para los interesados, dejamos el link para que puedan ver el ejemplo en funcionamiento.

En el próximo artículo vamos a explicar cómo configurar este componente para alterar un poco su modo de funcionamiento y hacer que los meses, así como el formato de fecha, estén en español.

Artículo por Miguel Angel Alvarez

Componente Datepicker de jQuery UI

Un componente de interfaces de usuario en jQuery UI que sirve para mostrar un calendario con el que seleccionar una fecha de manera visual.

Una de las interfaces de usuario más solicitadas y también de las más útiles, es un calendario para seleccionar una fecha. Este calendario permite que las personas puedan escoger una fecha a golpe de ratón y de manera visual, sin tener que escribir la fecha. Sin lugar a dudas el visitante agradecerá la comodidad de seleccionar la fecha por medio de este componente, pero también existen otras ventajas, como no obligarle a conocer el formato de fecha que necesitamos para una correcta entrada de datos.

En el artículo anterior ya estuvimos explicando las generalidades de las jQuery UI, por lo que se recomendaría su lectura, ya que contiene algunas guías que serán necesarias para aprender a utilizar cualquier tipo de componente y que vamos a dar por sabidas en el presente texto. Conviene leer por tanto el artículo Primeros pasos con jQuerty UI y la continuación Empezar a usar jQuery UI.

Entender la documentación de los componentes jQuery UI

Todos los componentes que nos ofrecen en jQuery UI los podemos encontrar correctamente documentados en la página de jQuery UI, a través de la sección de demos y documentación. En las páginas de demos encontraremos varios casos de uso de los componentes y también la documentación para hacerlos funcionar, léase, las opciones de configuración que ponen a nuestra disposición, así como los métodos de los objetos, eventos, etc.

De paso que explicamos cómo utilizar este calendario para seleccionar fechas queremos explicar también las cosas que podremos encontrar en la documentación de los componentes en general, así cualquier persona podrá continuar a partir de aquí, ya sea pasando a usos más avanzados del calendario o de otros componentes de jQuery UI. Así pues, conviene echar un vistazo a la página del demo y documentación del Datepicker, para revisar las informaciones que nos encontramos.

En la documentación es especialmente ilustrador el apartado de ejemplos, que contiene varios casos de uso para cada componente y para distintas opciones de configuración habituales. Por ejemplo, para la documentación del calendario podremos ver versiones con distintos formatos, idiomas, mostrando el número de la semana en el año, cambios en la animación, ver más de un mes al mismo tiempo, restricción de fechas para seleccionar sólo un rango permitido, etc. Nosotros nos centraremos en mostrar cómo localizar el calendario para que esté escrito en español y las fechas también tengan el formato dd/mm/aa, además veremos cómo colocar algunas opciones de configuración para mostrar un icono al lado del campo de texto, que una vez pulsado salga el calendario y otras cosas.

Un poco más abajo de los demos de uso de cada componente encontraremos también la documentación que mostrará el listado completo de opciones de configuración, los eventos que permite el componente, funciones especiales o métodos que podemos invocar sobre el mismo, etc. Conviene leer con calma apartado "Overview", que tiene informaciones generales y luego utilizar la completa referencia de opciones de configuración del apartado "Options", así como los eventos y demás.

> **Nota:** Por supuesto que los ejemplos son útiles, pues nos mostrarán casos de uso sencillos sobre los componentes, que podremos utilizar para guiarnos en los primeros pasos con ese componente. Pero en el momento que queramos complicar un poco las cosas vamos a tener que revisar con calma la documentación, pues en ella encontraremos todas las referencias sobre cada componente al detalle.

Definir variables de configuración para el seleccionador de fechas

En el anterior artículo de primeros pasos en jQuery UI ya mostramos cómo cargar un componente de las librerías y vimos el ejemplo concreto con el Datepicker, así que algo ya sabes sobre generar este sistema. En concreto necesitábamos un campo de texto INPUT.

```
<input type="text" name="fecha" id="campofecha">
```

Luego invocábamos al Datepicker sobre el campo de texto, con una llamada al método datepicker() sobre el objeto jQuery del elemento campo de texto.

```
$("#campofecha").datepicker();
```

Pues bien, para definir variables de configuración de este componente, podemos enviarlas a partir de la llamada al método datepicker(), enviando en notación de objeto todas las preferencias de configuración que deseemos aplicarle.

```
$("#campofecha").datepicker({
   showOn: 'both',
   buttonImage: 'calendar.png',
   buttonImageOnly: true,
   changeYear: true,
   numberOfMonths: 2,
   onSelect: function(textoFecha, objDatepicker){
      $("#mensaje").html("<p>Has seleccionado: " + textoFecha + "</p>");
   }
});
```

Con este listado de variables estamos configurando diversas cosas del elemento para seleccionar fechas, las siguientes:

- showOn: define que el calendario se muestre al hacer focus en el campo de texto o al pulsar el botón para abrir el calendario.
- buttonImage: permite definir la URL de una imagen para utilizarla como botón de abrir calendario.
- buttonImageOnly: para que sólo aparezca la imagen como botón.

- changeYear: para que haya un campo select con el que cambiar el año actual del calendario.
- numberOfMonths: para que se vean varios meses a la vez en varios calendarios.
- onSelect: un evento que se produce cuando el usuario selecciona una fecha.

De todas las variables de configuración anteriores, en las que sólo estamos viendo unas pocas de las que permite el componente, es especialmente interesante el tema del evento onSelect. Al definir un evento podemos realizar acciones Javascript para realizar cosas cuando se produce. Imaginaros que al seleccionar una fecha queréis comprobar cualquier cosa, pues por medio del evento onSelect podríais realizar cualquier tipo de función para ello. Entre los eventos tenemos otros interesantes como onChangeMonthYear, beforeShow, onClose, etc.

Otra cosa importante es que todas las variables de configuración se pueden cargar o bien al instanciar el componente o bien una vez ya está creado el datepicker con el método datepicker(), pasando como primer parámetro la cadena "option", un segundo parámetro con el nombre de la variable a cambiar y un tercero con el valor.

```
$("#campofecha").datepicker( "option", "changeMonth", true );
```

Con esa sentencia hacemos que el calendario tenga un select para cambiar rápidamente el mes. (Mirar el ejemplo del final del artículo para más información)

Localización del calendario jQuery UI

Una de las cosas más importantes para nuestro calendario es que esté en español, no sólo por los nombres de mes o de la semana, sino también por el formato de la fecha escrito en el campo de texto una vez seleccionamos un día del calendario.

Para localizar el calendario tenemos que cargar una serie de datos en las opciones del calendario, como los nombres de los días de la semana, de los meses y el formato de nuestra fecha. Todos estos datos podríamos construirlos por nosotros mismos, pero en jQuery han puesto a nuestra disposición un listado de archivos de configuración por idiomas.

La lista de localizaciones disponibles está en http://jquery-ui.googlecode.com/svn/trunk/ui/i18n/

Aunque, como decíamos, si no encuentras tu localización, la puedes crear. Para crear la localización tendríamos que definir variables de configuración como monthNamesShort, dayNames, dayNamesShort, dayNamesMin, weekHeader, etc. El código sería como este, para el idioma español.

```
/* Inicialización en español para la extensión 'UI date picker' para jQuery. */
/* Traducido por Vester (xvester [en] gmail [punto] com). */
jQuery(function($){
   $.datepicker.regional['es'] = {
      closeText: 'Cerrar',
      prevText: '<Ant',
      nextText: 'Sig>',
      currentText: 'Hoy',
      monthNames: ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio', 'Julio', 'Agosto',
'Septiembre', 'Octubre', 'Noviembre', 'Diciembre'],
      monthNamesShort: ['Ene','Feb','Mar','Abr', 'May','Jun','Jul','Ago','Sep', 'Oct','Nov','Dic'],
      dayNames: ['Domingo', 'Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado'],
      dayNamesShort: ['Dom','Lun','Mar','Mié','Juv','Vie','Sáb'],
      dayNamesMin: ['Do','Lu','Ma','Mi','Ju','Vi','Sá'],
      weekHeader: 'Sm',
      dateFormat: 'dd/mm/yy',
      firstDay: 1,
      isRTL: false,
      showMonthAfterYear: false,
      yearSuffix: ''};
   $.datepicker.setDefaults($.datepicker.regional['es']);
});
```

Como se puede ver, se definen unos valores en un objeto $.datepicker.regional['es'] y luego se asigna a todos los calendarios de la página por medio de $.datepicker.setDefaults().

En general es código lo podemos colocar en cualquier sitio o bien copiarlo en un archivo a parte, como por ejemplo jquery.ui.datepicker-es.js e incluirlo con la etiqueta SCRIPT:

```
<script type="text/javascript" src="js/jquery.ui.datepicker-es.js"></script>
```

Ejemplo completo de calendario Datepicker en jQuery UI

Para acabar podemos ver un ejemplo de calendario que tiene varias de las cosas que hemos visto en este artículo.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"
>
<html lang="es">
<head>
<title>Seleccionar fecha con jQuery UI</title>
   <link type="text/css" href="css/le-frog/jquery-ui-1.8.1.custom.css" rel="Stylesheet" />
   <script type="text/javascript" src="../../jquery-1.4.2.min.js"></script>
   <script type="text/javascript" src="js/jquery-ui-1.8.1.custom.min.js"></script>
   <script type="text/javascript" src="js/jquery.ui.datepicker-es.js"></script>
<script>
$(document).ready(function(){
   $("#campofecha").datepicker({
      showOn: 'both',
      buttonImage: 'calendar.png',
      buttonImageOnly: true,
      changeYear: true,
      numberOfMonths: 2,
      onSelect: function(textoFecha, objDatepicker){
         $("#mensaje").html("<p>Has seleccionado: " + textoFecha + "</p>");
      }
   });
})
</script>
</head>
<body>

<form>
   Fecha: <input type="text" name="fecha" id="campofecha">
</form>

<div id="mensaje"></div>


<a href="#" id="cambiames">Mostrar formulario para cambiar mes</a>
<script>
$(document).ready(function(){
   $("#cambiames").click(function(){
      $("#campofecha").datepicker( "option", "changeMonth", true );
   });
});
</script>

</body>
</html>
```

Podemos ver el ejemplo en marcha en una página aparte.

Lo cierto es que, cuando investigamos un poco este Datepicker, podemos ver que es un componente magnífico, con todas las opciones de configuración y personalización que podamos necesitar. Hablar de todas ellas y presentar ejemplos sería material para un manual entero, por ello, para usos más complejos recomendamos explorar la documentación de jQuery UI, porque probablemente haya alguna cosa que nos haga falta saber y que no hemos explicado en este artículo.

Artículo por Miguel Angel Alvarez

Componente Dialog de jQuery UI

Plugin Dialog, el componente para crear cajas de diálogo dinámicas que ofrecen las librerías para creación de interfaces de usuario jQuery UI.

Estamos revisando los plugins más interesantes que ofrecen las librerías jQuery UI para crear interfaces de usuario. Sin lugar a dudas uno de los más importantes y versátiles de entre los componentes que nos podremos encontrar es Dialog, una herramienta completa para crear cajas de diálogo perfectamente configurables.

Esas cajas de diálogo sirven para mostrar a los usuarios mensajes emergentes en ventanas DHTML, es decir, por medio de una capa que se superpone al contenido de la página. Estas capas se pueden configurar de la forma con la que el desarrollador necesite y mostrar todo tipo de mensajes. Además por medio de los Javascript de la librería jQuery, sin que tengamos que hacer nada, el usuario podrá redimensionar las cajas de diálogo y arrastrarlas a otros puntos de la página.

Como podréis ver a lo largo de este artículo, es un componente que tiene un aspecto muy atractivo y que además gracias a su versatilidad lo podremos utilizar en multitud de ocasiones. Además tenemos disponible uno de los comportamientos dinámicos más demandados por las personas, las denominadas cajas de diálogo "modal box", esas que cuando están activas hacen que la página se oscurezca y se inhabilite, de modo que el usuario no pueda seguir haciendo cosas hasta que se cierre la ventana.

Nota: Para entender todo lo que vamos a ver será se requieren unos conocimientos previos sobre jQuery, y sobre las librerías jQuery UI. Sobre el framework en general podréis aprender con el Manual de jQuery y sobre jQuery UI sería importante haber leído ya los primeros pasos a jQuery UI.

Crear cajas de diálogo

Como paso previo necesitaremos haber incluido el código Javascript y CSS de las librerías jQuery UI, tal como se explica en el artículo Empezar a usar jQuery UI en una página web. Una vez lo tenemos, necesitaremos crear un DIV con el contenido que queramos mostrar en la caja de diálogo. Este DIV lo podremos colocar en el código HTML de la página y más adelante explicaremos cómo se podría crear sobre la marcha con código Javascript y jQuery. Así pues, tendremos el HTML siguiente escrito en algún lugar de la página, es indiferente dónde.

Esta es la caja de diálogo más básica, que se puede redimensionar y arrastrar a otra posición. Además, se puede cerrar con el icono del aspa "X" que aparece en el titular de la caja.

Como se puede ver, se le ha indicado un atributo id, para asignarle un nombre a este DIV. Además hemos utilizado también un atributo en el DIV no habitual, que es el title, cuyo valor se utilizará como título en la caja de diálogo.

Ahora, tendremos que realizar el código Javascript para invocar el método que hará que ese DIV se convierta en una caja de diálogo, gracias a jQuery UI, casi por arte de magia!

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

Como se puede ver, por medio del objeto jQuery que hemos seleccionado con "#dialogo", osea, el DIV anterior, invocamos el método dialog(), que hará que la caja de diálogo se muestre en la página, con el contenido indicado en el HTML del DIV.

Podemos verla en una página aparte.

El código fuente completo de este primer ejemplo es el siguiente:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"
>
<html lang="en">
<head>
<title>Caja de diálogo jQuery UI</title>
   <link type="text/css" href="css/dot-luv/jquery-ui-1.8.1.custom.css" rel="Stylesheet" />
   <script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
   <script type="text/javascript" src="js/jquery-ui-1.8.1.custom.min.js"></script>
</head>
<body>
   <h1>Caja de diálogo jQuery UI</h1>

<p>Cuando se terminen de cargar los scripts, aparecerá una caja de diálogo...</p>

<div id="dialogo" title="Título de la caja" style="display: none;">
   <p>Esta es la caja de diálogo más básica, que se puede redimensionar y arrastrar a otra posición.
Además, se puede cerrar con el icono del aspa "X" que aparece en el titular de la caja.</p>
```

```
</div>

<script>
$(document).ready(function(){
   $("#dialogo").dialog();
});
</script>

</body>
</html>
```

Espero que las personas que hayan seguido los cursos y talleres de jQuery en DesarrolloWeb.com no tengan ningún problema en entender el código anterior. Como se puede haber visto, para crear una caja de diálogo con jQuery UI es suficiente una línea de código.

Pero claro que, a medida que queramos configurar el comportamiento de estas cajas de diálogo y aprovechar sus posibilidades, tendremos que conocer más a fondo el componente y para eso nos vendrá muy bien la documentación, que se puede acceder en la URL:

http://jqueryui.com/demos/dialog/

En este artículo y algunos siguientes vamos a explorar muchas de las cosas que podremos hacer con las cajas de diálogo, pero lo cierto es que las posibilidades son enormes y su uso dependerá también de las necesidades con las que nos encontremos. Así que a lo largo de los próximos artículos vamos a ir implementando a las tareas más habituales que podréis necesitar, como hacer que las cajas de diálogo sólo se abran cuando nosotros queramos, o cuando el usuario pulse un enlace, que tengan ciertas dimensiones, que aparezcan en un lugar determinado y con una animación, que tengan botones para realizar acciones adicionales, que respondan a eventos, etc.

Crear el DIV de la caja de diálogo sobre la marcha con jQuery

Antes de acabar este primer artículo voy a tratar un caso interesante, que tiene más que ver con jQuery que con el propio plugin Dialog. Antes vimos cómo convertir un DIV escrito en el código HTML de la página en una caja de diálogo y ahora vamos a mostrar cómo podríamos generar ese DIV dinámicamente con instrucciones Javascript y jQuery, para abrir una caja de diálogo sin tener que haber ningún DIV en el código.

```
var caja2 = $('<div title="Segunda caja"><p>Esta es una segunda caja...</p></div>');
caja2.dialog();
```

Como se puede ver, en la primera línea creamos un objeto jQuery a partir de un pedazo de código HTML, que es una capa DIV. Ese elemento DIV que estamos creando lo almacenamos en la variable caja2 y luego invocamos a dialog() a través de ella.

El código completo de este ejemplo sería el siguiente:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"
>
<html lang="en">
<head>
<title>Caja de diálogo jQuery UI</title>
   <link type="text/css" href="css/dot-luv/jquery-ui-1.8.1.custom.css" rel="Stylesheet" />
   <script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
   <script type="text/javascript" src="js/jquery-ui-1.8.1.custom.min.js"></script>
   <style type="text/css">
   body{
      font-size: 0.75em;
      font-family: verdana, arial, helvetica;
   }
   </style>
</head>
<body>
   <h1>Caja de diálogo jQuery UI</h1>

<p>Cuando se terminen de cargar los scripts, aparecerá una caja de diálogo... pero esta vez la vamos
a crear de otra manera.</p>
```

```
<script>
$(document).ready(function(){
   var caja2 = $('<div title="Segunda caja creada"><p>Esta es una segunda caja de
diálogo...</p><p>En esta ocasión generé en tiempo de ejecución el DIV con Javascript y luego, sin
llegar a mostrar el DIV en la página, lo convierto en una caja de diálogo.</p></div>');
   caja2.dialog();
});
</script>

</body>
</html>
```

Podemos ver en una página aparte el ejemplo de esta segunda caja de diálogo.

En el siguiente artículo continuaremos explicando ejemplos de tipos de cajas de diálogo con jQuery UI.

Artículo por Miguel Angel Alvarez

Opciones del plugin Dialog de jQuery UI. Caja modal.

Cómo configurar las cajas de diálogo a través de un objeto de opciones que enviamos al método dialog(), de jQuery UI. Vemos también cómo hacer una caja modal.

En el artículo anterior comenzamos a explicar lo que son las cajas de diálogo y cómo jQuery UI ofrece un fantástico plugin para implementarlas. Además vimos un par de ejemplos iniciales que convendría tener presentes antes de continuar con el presente ejemplo.

Como vimos, existe un método llamado dialog() que tenemos que invocar para convertir un elemento DIV en una caja de diálogo. Lo que no habíamos visto todavía es que a este método podemos pasarle parámetros para configurar dicha caja, a partir de una serie de opciones. Todas las opciones tienen valores por defecto, con lo que en los anteriores ejemplos, simplemente estábamos abriendo una caja de diálogo configurada por defecto.

Las opciones de configuración las tenemos que enviar en notación de objeto, indicando una serie de atributos u opciones con sus diferentes valores. Podríamos ver un ejemplo de caja de diálogo configurada con diferentes valores que los estándar:

```
$("#dialogo").dialog({
   modal: true,
   title: "Caja con opciones",
   width: 550,
   minWidth: 400,
   maxWidth: 650,
   show: "fold",
   hide: "scale"
});
```

Como se está viendo en el código anterior, se está creando una caja de diálogo a partir de un elemento de la página que tiene el identificador id=dialogo. Además, al método dialog() le estoy pasando un objeto formado por distintas propiedades.

Podemos ver cómo sería la caja del anterior ejemplo en una página aparte.

Caja modal (modal box)

Una de los ejemplos más recurridos y de las preguntas que seguro solicitarán más personas es cómo hacer las cajas modales, lo que en inglés se llama "Modal box". Es una caja que al aparecer hace que toda la página se oscurezca, menos la propia caja de diálogo, para llamar la atención del usuario y no permitir que haga otras cosas, sino prestar atención al texto de la caja o a las acciones que solicite.

Esto se consigue con el primero de los atributos del objeto de opciones del código anterior.

modal: true

Otras options del cuadro de diálogo

Aparte de la propiedad para hacer cajas modales, hemos aplicado otra serie de atributos que comentamos a continuación:

title: "Caja con opciones"
La propiedad title sirve para cambiar el título de la caja de diálogo y se tiene en cuenta antes del contenido que pueda tener el atributo title del DIV con el que se ha hecho la caja.

width: 550
La propiedad width indica el ancho de la caja modal, en píxeles.

minWidth: 400
Es la anchura mínima permitida. Recordemos que el usuario puede redimensionar la caja, de modo que la anchura real de la misma podría variar. Si definimos el atributo minWidth nos aseguraremos que su anchura nunca baje de este valor en píxeles.

maxWidth: 650
De manera similar a minWidth, pero para definir una anchura máxima permitida.

show: "fold"
Con el atributo show podemos definir un efecto para que la caja de diálogo no se muestre de golpe sino con una transición suavizada. El valor de show que podemos seleccionar es una cadena de caracteres con alguno de los efectos posibles. Leer la nota sobre este tema.

hide: "scale"
Igual que el atributo show, pero sirve para definir la transición o efecto utilizado al cerrar la ventana de diálogo.

> **Nota:** Como se vio en anteriores ejemplos, las cajas de diálogo aparecen de golpe, pero nosotros podemos querer que aparezcan de manera suavizada con una transición o efecto dadas. En jQuery UI ya están creados una buena variedad de efectos, que se pueden encontrar entre los componentes de descarga de las librerías.
>
> Algunos efectos que podremos asignar a la animación de mostrar u ocultar el cuadro de diálogo son "explode", "fold", "scale", "clip", "slide", etc. Es conveniente comentar que estos efectos no están creados exclusivamente para animar las cajas, sino para animar muchos otros componentes de jQueryUI.
>
> Para poder activar cualquiera de estos efectos tenemos que haber descargado un paquete de jQuery UI que los contenga, es decir, haber seleccionado alguno de esos componentes de efectos cuando descargamos las librerías para interfaces de usuario.

Ahora podemos ver el código del ejemplo completo:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"
>
<html lang="en">
<head>
<title>Caja de diálogo jQuery UI</title>
   <link type="text/css" href="css/dot-luv/jquery-ui-1.8.1.custom.css" rel="Stylesheet" />
   <script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
   <script type="text/javascript" src="js/jquery-ui-1.8.1.custom.min.js"></script>
</head>
<body>
   <h1>Caja de diálogo jQuery UI</h1>

<p>Cuando se terminen de cargar los scripts, aparecerá una caja de diálogo...</p>

<div id="dialogo">
   <p>Esta es la caja de diálogo más básica, que se puede redimensionar y arrastrar a otra posición.
Además, se puede cerrar con el icono del aspa "X" que aparece en el titular de la caja.</p>
</div>

<script>
$(document).ready(function(){
   $("#dialogo").dialog({
      modal: true,
      title: "Caja con opciones",
```

```
      width: 550,
      minWidth: 400,
      maxWidth: 650,
      show: "fold",
      hide: "scale"
   });
});
</script>

</body>
</html>
```

Hay que señalar que en este ejemplo hemos visto sólo un pequeño conjunto de opciones disponibles en el componente Dialog. Recordar siempre consultar la documentación para una completa referencia. En el siguiente artículo explicaremos cómo crear una caja de diálogo y abrirla sólo cuando el usuario pulse un enlace.

Artículo por Miguel Angel Alvarez

Campos SELECT dinámicos con jQuery, Ajax, PHP y MySQL

Un script para generar dos campos de formulario SELECT dinámicos y combinados, con programación cliente Javascript y Ajax con jQuery y programación servidor por medio de PHP y MySQL.

La creación de los campos SELECT combinados es una de las tareas clásicas que algún desarrollador va a tener que solucionar algún día. Existen muchas maneras de llevarla a cabo, pero sin duda, cuando se utiliza Ajax el resultado de cara al usuario es mucho mejor. En este artículo queremos mostrar una manera de hacer que las opciones de un SELECT dependan de lo que se haya escogido en otro, lo que conocemos como "selects dinámicos" o "selects combinados".

Para ello vamos a relatar un caso práctico completo. Un sistema que nos permitirá escoger provincias en un select y luego una población en otro select, que dependerá de la provincia seleccionada en el promer select. Veremos todo el sistema de manera global, desde la programación del cliente con Javascript y Ajax, pasando por el desarrollo en el servidor por medio de PHP y una base de datos MySQL donde están todas las provincias y poblaciones. Además, utilizaremos el el framework Javascript jQuery, para facilitarnos la vida gracias a sus funciones para hacer Ajax y modificar el DOM de la página.

> **Nota:** Recordar que en DesarrolloWeb.com podréis aprender sobre todas las tecnologías utilizadas, como Javascript, Ajax, jQuery, PHP, MySQL...
> Así mismo, podemos consultar otros artículos que también tratan sobre hacer este tipo de interfaces de usuario, como Elementos de formulario select asociados con Javascript http://www.desarrolloweb.com/articulos/1281.php o bien Selects combinados con Ajax y PHP.

En el ejercicio utilizaremos en total 3 tablas: provincias, poblaciones y clientes. En este tutorial vamos a insertar y modificar la base de datos, para ello necesitaremos crear 3 documentos: insertar.php, modificar.php y buscar.php.

Para los documentos insertar.php y modificar.php iniciamos el documento poniendo la conexión a base de datos y el enlace con la librería jQuery.

> **Nota:** Si te interesa el código fuente de los archivos de este ejemplo te recomiendo que no copies y pegues el código que aparece en este artículo. Es mucho mejor que descargues todo el código para hacer los selects dinámicos de la la página de GitHub: http://github.com/desarrollowebcom/SELECT-dinamicos-jQuery-Ajax-PHP-MySQL/downloads

```
<?
function Conectarse(){
if (!($link=mysql_connect("localhost","root","")))
{echo "Error conectando a la base de datos.";
exit();
}
if (!mysql_select_db("desarrolloweb",$link))
{
echo "Error seleccionando la base de datos.";
```

```
exit();
}
return $link;
}
$link=Conectarse();

echo "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0
Transitional//EN\" \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">
<html xmlns=\"http://www.w3.org/1999/xhtml\">
<head><meta http-equiv=\"Content-Type\" content=\"text/html; charset=ISO-8859-1\" />
<script src=\"http://code.jquery.com/jquery-latest.js\"></script>
</head>

<body>";
?>
```

Y para su finalización :

Ahora creamos el formulario de insertar. Como podemos ver, en el select RE_Prov he puesto un atributo que lo utilizaremos para modificarlo. Si esta vacío al insertar no pasa nada.

```
<?
echo "<FORM action=\"".$_SERVER["PHP_SELF"]."\" method=\"POST\" style='margin:30px 5px 30px 5px;'>
<table width=\"500px\">
<tbody>
<tr>
<td>Nombre</td>
<td><input type=\"text\" name=\"nombre\" /></td>
</tr>
<tr>
<td>Provinvia</td>
<td><select name='RE_Prov' id=\"provincia\" poblacioattri=''>
<option value=''>Seleccione provincia</option>";
$B_BUSCAR= mysql_query ("SELECT * FROM provincias order by provincia asc",$link);
$R_BUSCAR=mysql_fetch_assoc($B_BUSCAR);
$C_BUSCAR=mysql_num_rows($B_BUSCAR);
$suma=0;
do{ ++$suma;
echo "<option value='".$R_BUSCAR['id']."'>".$R_BUSCAR['provincia']."</option>";
}while($R_BUSCAR=mysql_fetch_assoc($B_BUSCAR));
echo "</select> <span id='Buscando'></span></td>
</tr>
<tr>
<td>Población</td>
<td><select name='RE_pobla' id=\"poblacionList\">
<option value='' selected='selected'>Seleccionar población</option>
</select></td>
</tr>
<tr>
<td></td>
<td><input type=\"submit\" /></td>
</tr>
</tbody>
</table>
</form>";
?>
```

Este es el Script que utilizaremos para hacer la búsqueda a la base de datos.

```
<script>
jQuery('#provincia').change(function () {
var numero =document.getElementById("provincia").value;
var poblacio = jQuery(this).attr("poblacioattri");
var to=document.getElementById("Buscando");
to.innerHTML="buscando....";
jQuery.ajax({
type: "POST",
url: "busqueda.php",
data: 'idnumero='+numero+'&idpobla='+poblacio,
success: function(a) {
jQuery('#poblacionList').html(a);
```

```
var to=document.getElementById("Buscando");
to.innerHTML="";
}
});
})
.change();
</script>
```

Detalles del script:
Para saber el valor de la provincia: var numero =document.getElementById("provincia").value;
Cuando movemos este select escribimos en buscando.... esto está puesto aquí:
var to=document.getElementById("Buscando");
to.innerHTML="buscando....";

Enviamos la información deseada al documento busqueda.php en post y enviado estos datos:

data: 'idnumero='+numero+'&idpobla='+poblacio,
idnumero es el valor del select de la provincia y idpobla es el atributo pero que ahora esta vació.
Cuando se realice la busqueda lo mostraremos aquí:

```
jQuery('#poblacionList').html(a);
```

y el borraremos buscando.... así:

```
var to=document.getElementById("Buscando");
to.innerHTML="";
```

Ponemos lla inserción a la base de datos.

```
<?
if($_POST["nombre"]){
$nombre=addslashes($_POST["nombre"]);
$RE_Prov=addslashes($_POST["RE_Prov"]);
$RE_pobla=addslashes($_POST["RE_pobla"]);

$insertar=mysql_query("INSERT clientes (nombre,provincia,poblacion) VALUES
('$nombre','$RE_Prov','$RE_pobla')",$link);
if ($insertar==1){echo "<script>alert(\"Se ha insertado.\");</script>";}else{echo
"<script>alert(\"Error!!!!.\");</script>";}
}
?>
```

y cerramos :

```
</body>
</html>
?>
```

Ahora vamos a explicar que hay en buscar.php:
Primero ponemos la conexión y su búsqueda a la base de datos i mostramos los resultados.
Recordar que desde insertar.php y modificar.php se envían dos variables por POST que son $_POST["idnumero"] que es el valor de la provincia y después $_POST["idpobla"] que es el valor de la población.

```
<?
function Conectarse()
{
if (!($link=mysql_connect("localhost","root","")))
{
echo "Error conectando a la base de datos.";
exit();
}
if (!mysql_select_db("desarrolloweb",$link))
{
echo "Error seleccionando la base de datos.";
exit();
}
return $link;
}
$link=Conectarse();
```

```
$B_BUSCAR= mysql_query ("SELECT * FROM poblaciones where idprovincia='".$_POST["idnumero"]."' order
by poblacio asc",$link);
$R_BUSCAR=mysql_fetch_assoc($B_BUSCAR);
$C_BUSCAR=mysql_num_rows($B_BUSCAR);
if($C_BUSCAR){
do{
if($_POST["idpobla"]==$R_BUSCAR['id']){$TRUE=" selected='TRUE'";}else{$TRUE="";}
echo "<option value='".$R_BUSCAR['id']."' $TRUE>".htmlentities($R_BUSCAR['poblacio'])."</option>";

}while($R_BUSCAR=mysql_fetch_assoc($B_BUSCAR));
}else{
echo "<option value=''>".htmlentities("Seleccionar población")."</option>";
}
mysql_close($link);
?>
?>
```

Ahora vamos a crear el formulario de modificar, pero promero vamos a crear el listado con los resultados que hay en la tala de clientes y después iremos al formulario mediante un link.

```
<?
// Para hacerlo sencillo he puesto una variable que se llama ac y que si no existe se muestra el
listado esta variable se envía por GET.
if(!$_GET["ac"]){
echo "<h3>Listado</h3>
<table width=\"500px\" style='margin:30px 5px 30px 5px;'>";
$B_BUSCAR= mysql_query ("SELECT * FROM clientes",$link);
$R_BUSCAR=mysql_fetch_assoc($B_BUSCAR);
$C_BUSCAR=mysql_num_rows($B_BUSCAR);
$suma=0;
do{ ++$suma;
echo "<tr><td><a href='".$_SERVER["PHP_SELF"]."?ac=1&id=".$R_BUSCAR["id"]."'>$suma
".stripslashes($R_BUSCAR["nombre"])." (Modificar)</a></td></tr>";
}while($R_BUSCAR=mysql_fetch_assoc($B_BUSCAR));
echo "</table>";
}else{
// Ahora buscamos los datos seleccionados.


$B_cliente= mysql_query ("SELECT * FROM clientes where id='".$_GET["id"]."'",$link);
$R_cliente=mysql_fetch_assoc($B_cliente);
$C_cliente=mysql_num_rows($B_cliente);

// Recuperamos la información y la ponemos dentro de una variable.
$nombre=stripslashes($R_cliente["nombre"]);
$RE_Prov=stripslashes($R_cliente["provincia"]);
$RE_pobla=stripslashes($R_cliente["poblacion"]);

echo "
<h3>Modificar</h3>
<FORM action=\"".$_SERVER["PHP_SELF"]."\" method=\"POST\" style='margin:30px 5px 30px 5px;'>
<table width=\"500px\">
<tbody>
<tr>
<td>Nombre</td>
<td><input type=\"text\" name=\"nombre\" value='$nombre' /></td>
</tr>
<tr>
<td>Provinvia</td>";
// Ahora si que utilizamos el atributo poblacioattri donde ponemomos dentro la variavle de la
población actual
echo "
<td><select name='RE_Prov' id=\"provincia\" poblacioattri='$RE_pobla'>
<option value=''>Seleccione provincia</option>";
$B_BUSCAR= mysql_query ("SELECT * FROM provincias order by provincia asc",$link);
$R_BUSCAR=mysql_fetch_assoc($B_BUSCAR);
$C_BUSCAR=mysql_num_rows($B_BUSCAR);
$suma=0;
do{ ++$suma;
echo "<option value='".$R_BUSCAR['id']."'";if($RE_Prov==$R_BUSCAR['id']){echo "
```

```
selected='true'";}echo ">".$R_BUSCAR['provincia']."</option>";
}while($R_BUSCAR=mysql_fetch_assoc($B_BUSCAR));
echo "</select> <span id='Buscando'></span></td>
</tr>
<tr>
<td>Población</td>
<td><select name='RE_pobla' id=\"poblacionList\">
<option value='' selected='selected'>Seleccionar población</option>
</select></td>
</tr>
<tr>
<td></td>
<td><input type=\"hidden\" name=\"MOD_ID\" value=\"".$_GET["id"]."\" /><input type=\"submit\"
/></td>
</tr>
</tbody>
</table>
</form>";
}
?>
?>
```

Ponemos el script donde si que recuperamos el atributo poblacioattri y la forma de recuperarlo es así:

```
var poblacio = jQuery(this).attr("poblacioattri");

<script>
jQuery('#provincia').change(function () {
var numero =document.getElementById("provincia").value;
var poblacio = jQuery(this).attr("poblacioattri");
var to=document.getElementById("Buscando");
to.innerHTML="buscando....";
jQuery.ajax({
type: "POST",
url: "busqueda.php",
data: 'idnumero='+numero+'&idpobla='+poblacio,
success: function(a) {
jQuery('#poblacionList').html(a);
var to=document.getElementById("Buscando");
to.innerHTML="";
}
});
})
.change();
</script>
?>
```

Y modificamos:

```
if($_POST["nombre"]){
$nombre=addslashes($_POST["nombre"]);
$RE_Prov=addslashes($_POST["RE_Prov"]);
$RE_pobla=addslashes($_POST["RE_pobla"]);

$mod=mysql_query("UPDATE clientes SET nombre='$nombre',provincia='$RE_Prov',poblacion='$RE_pobla'
WHERE id=".$_POST["MOD_ID"]."",$link);
if ($mod==1){echo "<script>alert(\"Se ha Modificado.\");</script>";}else{echo
"<script>alert(\"Error!!!!.\");</script>";}
}
?>
```

y cerramos :

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

Artículo por Eduard Nebot

Menú desplegable en jQuery

Creamos un plugin jQuery para implementar una barra de navegación con un menú desplegable en varios submenús.

Cuando tratamos con interfaces de usuario en páginas web uno de los ejemplos más típicos es el menú desplegable, osea, una barra de navegación que muestra varios enlaces y que, al ponerse encima de esos enlaces, muestra un submenú con otros enlaces. Ese submenú que aparece automáticamente, también desaparece por si solo cuando el usuario le retira el puntero del mouse.

Si lo deseas, puedes ver el menú desplegable dinámico que vamos a realizar.

En diversos artículos de DesarrolloWeb.com hemos tratado sobre la realización de menús desplegables y en esta ocasión vamos a dedicarnos a realizar el sistema utilizando jQuery, por medio de un plugin que cualquier desarrollador pueda utilizar en su propia página web.

Claro que éste no es un problema trivial, puesto que tenemos que trabajar con varias cosas de aquí y de allá: CSS, Javascript y jQuery para realizar el plugin. Afortunadamente tenemos todas las referencias en DesarrolloWeb.com para aprender todo lo que vamos a necesitar.

Objeto para configurar las opciones del menú

Para realizar este ejercicio de la manera más versátil posible hemos implementado una notación para generar las opciones del menú en una variable Javascript de tipo objeto. Es decir, los menús y submenús se especificarán con Javascript y enviaremos ese objeto al plugin para generar el HTML necesario para crearlos en la página.

Así que esta es la variable donde definimos el contenido del navegador desplegable:

```
//un array por cada uno de los menús desplegables
var opciones_menu = [
   {
      texto: "Enlace 1",
      url: "http://www.desarrolloweb.com",
      enlaces: [
         {
            texto: "Enlace 1.1",
            url: "#Enlace1-1"
         },
         {
            texto: "Enlace 1.2",
            url: "#Enlace1-2"
         },
         {
            texto: "Enlace 1.3",
            url: "#Enlace1-3"
         }
      ]
   },

   {
      texto: "DesarrolloMultimedia.es",
      url: "http://www.desarrollomultimedia.es",
      enlaces: [
         {
            texto: "Enlace 2.1",
            url: "#Enlace2-1"
         },
         {
            texto: "Enlace 2.2",
            url: "#Enlace2-2"
         }
      ]
   }
];
```

Como se puede ver se ha definido un array donde tenemos cada uno de los enlaces principales, que a su vez se declaran con un objeto que tiene varias propiedades.

Para cada enlace principal tenemos los datos:

- texto: con una cadena para el texto del enlace
- url: con una cadena para la URL a la que dirigir el enlace
- enlaces: un array con cada uno de los enlaces del submenú asociado a este enlace principal. Los enlaces del submenú que colocamos en este array son otros objetos. Con los siguientes datos:
    - texto: el texto para el enlace del submenú
    - url: la URL a la que dirigir este enlace del submenú.

Una vez entendido ese formato para el objeto de los enlaces, podemos ver el código del plugin.

```
////////////////////////////////////////////////////////////////////////
//creación del plugin generaMenu.
//envío el menú de opciones como parámetro
////////////////////////////////////////////////////////////////////////
(function($) {

$.fn.generaMenu = function(menu) {
   this.each(function(){
      var retardo;
      var capaMenu = $(this);
      //creo e inserto la lista principal
      var listaPrincipal = $('<ul></ul>');
      capaMenu.append(listaPrincipal);
      //enlaces principales
      var arrayEnlaces = [];
      var arrayCapasSubmenu = [];
      var arrayLiMenuPrincipal = [];
      //recorro los elementos del menú
      jQuery.each(menu, function() {
         //ahora en this tengo cada uno de los elementos.
         var elementoPrincipal = $('<li></li>');
         listaPrincipal.append(elementoPrincipal);
         //creo el enlace e inserto
         var enlacePrincipal = $('<a href="' + this.url + '">' + this.texto + '</a>');
         elementoPrincipal.append(enlacePrincipal);

         var capaSubmenu = $('<div class="submenu"></div>');
         //guardo la capa submenu en el elemento enlaceprincipal
         enlacePrincipal.data("capaSubmenu",capaSubmenu);
         //creo una lista para poner los enlaces
         var subLista = $('<ul></ul>');
         //añado la lista a capaMenu
         capaSubmenu.append(subLista);
         //para cada elace asociado
         jQuery.each(this.enlaces, function() {
            //en this tengo cada uno de los enlaces
            //creo el elemento de la lista del submenú actual
            var subElemento = $('<li></li>');
            //meto el elemento de la lista en la lista
            subLista.append(subElemento);
            //creo el enlace
            var subEnlace = $('<a href="' + this.url + '">' + this.texto + '</a>');
            //cargo el enlace en la lista
            subElemento.append(subEnlace);

         });
         //inserto la capa del submenu en el cuerpo de la página
         $(document.body).append(capaSubmenu);


         ////////////////////////////////////////
         //EVENTOS
         ////////////////////////////////////////
```

```
//defino el evento mouseover para el enlace principal
enlacePrincipal.mouseover(function(e){
   var enlace = $(this);
   clearTimeout(retardo)
   ocultarTodosSubmenus();
   //recupero la capa de submenu asociada
   submenu = enlace.data("capaSubmenu");
   //la muestro
   submenu.css("display", "block");
});

//defino el evento para el enlace principal
enlacePrincipal.mouseout(function(e){
   var enlace = $(this);
   //recupero la capa de submenu asociada
   submenu = enlace.data("capaSubmenu");
   //la oculto
   clearTimeout(retardo);
   retardo = setTimeout("submenu.css('display', 'none');",1000)

});

//evento para las capa del submenu
capaSubmenu.mouseover(function(){
   clearTimeout(retardo);
})

//evento para ocultar las capa del submenu
capaSubmenu.mouseout(function(){
   clearTimeout(retardo);
   submenu = $(this);
   retardo = setTimeout("submenu.css('display', 'none');",1000)
})

//evento para cuando se redimensione la ventana
if(arrayEnlaces.length==0){
   //Este evento sólo lo quiero ejecutar una vez
   $(window).resize(function(){
      colocarCapasSubmenus();
   });
}

////////////////////////////////////////
//FUNCIONES PRIVADAS DEL PLUGIN
////////////////////////////////////////

//una función privada para ocultar todos los submenus
function ocultarTodosSubmenus(){
   $.each(arrayCapasSubmenu, function(){
      this.css("display", "none");
   });
}

//función para colocar las capas de submenús al lado de los enlaces
function colocarCapasSubmenus(){
   $.each(arrayCapasSubmenu, function(i){
      //coloco la capa en el lugar donde me interesa
      var posicionEnlace = arrayLiMenuPrincipal[i].offset();
      this.css({
         left: posicionEnlace.left,
         top: posicionEnlace.top + 28
      });
   });
}


//guardo el enlace y las capas de submenús y los elementos li en arrays
arrayEnlaces.push(enlacePrincipal);
arrayCapasSubmenu.push(capaSubmenu);
arrayLiMenuPrincipal.push(elementoPrincipal);
```

```
            //coloco inicialmente las capas de submenús
            colocarCapasSubmenus();
        });

    });

    return this;
};

})(jQuery);
```

Como se puede ver, gran parte del código es para hacer unos bucles por los que recorrer todo el array de enlaces que pasamos como parámetro al plugin y construir el HTML necesario para el menú.

Luego se crearon unos eventos para mostrar y ocultar los submenús al situarse encima de los enlaces.

Uno de los temas a destacar en este código es que las capas de los submenús se tienen que colocar al lado de los enlaces principales. Para ello, se utiliza la función colocarCapasSubmenus(), que recorre el array de capas del submenú y coloca una a una todas estas capas en una posición absoluta que se calcula en función de la posición del elemento LI donde está el enlace principal.

Nota: Aunque el plugin sea algo complejo, no vamos a ofrecer más explicaciones sobre por qué se ha hecho así, aparte de los propios comentarios del código. Si deseas encontrar alguna que otra explicación adicional, te recomiendo que leas los artículos Menú desplegable Cross-Browser 1 y la segunda parte del menú desplegable cross-browser, que utilizan exactamente la misma idea básica para ser construidos que este menú desplegable en jQuery.

Ejecutar el plugin para generar el menú desplegable

Ahora sólo nos queda invocar al plugin para generar el menú desplegable. Para eso necesitamos una capa donde mostrar el menú en el código HTML.

```
<div id="menu"></div>
```

Además, necesitamos el array con los enlaces del menú, que habíamos definido al principio del artículo en la variable "opciones_menu". Esa variable la tenemos que enviar al iniciar el plugin.

```
$("#menu").generaMenu(opciones_menu);
```

Con esto ya tenemos el menú funcionando en la página. Sólo nos faltaría un poco de CSS.

CSS para definir el aspecto del menú desplegable

Por último vamos a decorar el menú según nuestros intereses, para que quede acorde con el diseño de nuestra página.

Para definir los estilos de los enlaces del menú principal, lo podemos hacer a partir de la capa donde estamos mostrando el menú, a la que habíamos puesto en le código HTML un identificador id="menu". A través de ese identificador podremos definir no sólo el estilo de la propia capa, sino también el estilo de la lista UL que se generará para el menú y de los elementos LI o los enlaces.

```
#menu{
   background-color: #e5e5e5;
   padding: 10px;
   text-align: center;
   overflow: hidden;
}
#menu ul, .submenu ul{
   list-style: none;
   margin: 0;
   padding: 0;
}
#menu li{
   margin: 0;
   padding: 0 20px 0 10px;
```

```
   float: left;
}
```

También podemos estilizar cada una de las capas de los submenús, a las que en el código del plugin le pusimos la clase CSS "submenu". Ese nombre de clase es fijo en el código de nuestro plugin, por lo que una mejora sería que el plugin recibiera el nombre de esa clase en un array de opciones, para que cada usuario lo pudiera configurar.

Como de momento ese nombre de clase no se puede cambiar, tendríamos que aplicar los estilos de esta manera.

```
.submenu{
   display: none;
   position: absolute;
   padding: 4px 4px 2px;
   background-color: #e5e5e5;
}
.submenu li{
   padding: 7px 10px;
   margin: 4px 0;
   background-color: #ffc;
   width: 200px;
}
.submenu li:hover{
   background-color: #ff6;
}
```

Como se puede ver, no sólo se aplican estilos a la capa del submenú, sino también a las listas UL que contendrá, sus elementos LI, enlaces, etc.

Así que no hay mucho más que contar sobre este menú desplegable. Seguramente a los lectores se les ocurrirá varias mejoras, pero de momento está bien tal como nos ha quedado y no deseamos complicar aun más el ejercicio.

Si deseas ver el menú desplegable hasta el momento, puedes hacerlo desde este enlace al ejemplo en funcionamiento.

Artículo por Miguel Angel Alvarez

Plugin para seleccionar y deseleccionar varios checkbox con jQuery

Desarrollo de un plugin en jQuery para hacer un enlace con el que seleccionar y deseleccionar toda una lista de checkbox de una vez.

Este plugin que vamos a realizar incluye varias de las mejoras que venimos explicando en los tutoriales sobre plugins del Manual de jQuery. De hecho, lo creamos con dos motivos didácticos. Por un lado queremos demostrar cómo hacer una envoltura del código del plugin por medio de una función, para protegerlo y aislarlo de otros posibles códigos de la página. Por otro lado, este script es una simple práctica sobre cómo hacer plugins en jQuery, que servirá como un ejemplo más en el Taller de jQuery.

Lo que vamos a construir es un sistema para seleccionar y deseleccionar un grupo de checkbox por medio de un enlace. Al hacer clic en un enlace se seleccionarán de una vez todos los checkboxes del grupo y al hacer clic en otro se quitará la selección de todos, ambos casos sin importar cómo estaban previamente.

Es una utilidad sencilla de realizar, que suele verse en distintas páginas, donde hay formularios con un listado grande de checkbox. Todo ello lo haremos en formato de plugin para que se pueda utilizar en cualquier página web donde se desee.

Adicionalmente, a este ejemplo le hemos incorporado un sistema para cargar una opción, que sirve para indicar si deseamos o no que el grupo de checkboxes esté habilitado o no. Si está habilitado podremos seleccionarlos y deseleccionarlos al pulsar sobre los checkbox y sobre los enlaces de seleccionar/deseleccionar todos. Si no están habilitados, sólo podríamos cambiar sus valores por medio de los enlaces.

Podemos ver una página con el ejemplo en marcha, para hacernos una idea exacta de nuestros objetivos.

Instrucciones para marcar y desmarcar un chekbox en jQuery

La funcionalidad de seleccionar y deseleccionar un checkbox se consigue a través del atributo HTML "checked" de la etiqueta INPUT. Si tiene el atributo el checkbox estará marcado y si no tiene el atributo, aparecerá desmarcado.

Para marcar un checkbox utilizaríamos este código:

```
//siendo la variable campo un objeto jQuery que contiene uno o más checkboxes
campo.attr("checked","1");
```

Para quitar la marca de un checkbox se puede hacer quitando el atributo "checked":

```
//campo es uno o varios campos check en un objeto jQuery
campo.removeAttr("checked");
```

Instrucciones para habilitar deshabilitar un checkbox

Un checkbox, así como cualquier otro campo de formulario se puede habilitar o deshabilitar a través del atributo "disabled" del campo INPUT.

Este sería el código para deshabilitar el campo:

```
campo.attr('disabled', true);
```

Ahora podemos ver el código para habilitar el campo, que simplemente quita el atributo disabled.

```
campo.removeAttr('disabled');
```

En ambos casos se supone que la variable campo es un objeto jQuery que contiene uno o más campos INPUT de formulario.

Plugin para generar enlaces de marcar y desmarcar todos los campos checkbox

Ahora que ya sabemos cómo realizar esos pasos básicos para alterar los checkboxes, pasemos al código del plugin jQuery, que esperamos os resulte bastante sencillo.

Nuestra idea es que el plugin se invoque sobre un contenedor vacío, donde se generarán y se insertarán los enlaces para seleccionar y deseleccionar los checkbox.

Como se va a comprobar, el plugin recibe un par de parámetros, uno es un selector para acceder al grupo de checkboxes que debemos marcar/desmarcar y otro es una lista de opciones en formato de objeto para configurar el plugin.

```
(function($) {
   $.fn.seleccionarDeseleccionarTodo = function(selectorCheckboxes, opciones) {
      //opciones de configuración por defecto
      var configuracion = {
         habilitados: true
      };
      //Las extiendo con las opciones recibidas al invocar el plugin
      $.extend(configuracion, opciones);

      this.each(function(){
         //todos los checkboxes
         var camosCheck = $(selectorCheckboxes);
         //capa para los controles de seleccionar/deseleccionar checkboxes
         elem = $(this);

         //creo el enlace para seleccionar todo
         var enlaceSel = $('<a href="#">Seleccionar</a>');
         //meto el enlace en la página
         elem.append(enlaceSel);
         //un separador de los enlaces
         elem.append($("<span> | </span>"));
         //creo el enlace para deseleccionar todo
         var enlaceDesel = $('<a href="#">Deseleccionar</a>');
         //meto el enlace en la página
         elem.append(enlaceDesel);
```

```
        //miro si los campos deben estar habilitados
        if(configuracion.habilitados){
           habilitarCampos();
        }else{
           deshabilitarCampos();
        }


        ////////////////////////////////////////////////////////
        //funciones
        ////////////////////////////////////////////////////////
        function habilitarCampos(){
           camosCheck.removeAttr('disabled');
        }
        function deshabilitarCampos(){
           camosCheck.attr('disabled', true);
        }
        function seleccionarTodosCampos(){
           camosCheck.attr("checked","1");
        }
        function deseleccionarTodosCampos(){
           camosCheck.removeAttr("checked");
        }

        //evento para el enlace de deseleccionar todos los campos
        enlaceDesel.click(function(e){
           e.preventDefault();
           deseleccionarTodosCampos();
        });
        //añado un evento al enlace para seleccionar todos los campos
        enlaceSel.click(function(e){
           e.preventDefault();
           seleccionarTodosCampos();
        });

      });
      return this;
   };
})(jQuery);
```

El código anterior está comentado para que se puedan ir entendiendo los pasos realizados. Además, hemos separado el código en diversas funciones que esperamos que ayuden que sea más claro.

Ejemplo para poner en marcha el plugin de marcar/desmarcar checkbox

Para poner en marcha el plugin necesitamos un HTML con un contenedor vacío para los enlaces de marcar/desmarcar y varios checkbox. Algo como esto:

```
<form method="post" action="#">
   <div id="seleccionardeseleccionar"></div>
   <input type="checkbox" name="si" class="controlseleccionar"> jQuery
   <br>
   <input type="checkbox" name="si" class="controlseleccionar"> Eventos
   <br>
   <input type="checkbox" name="si" class="controlseleccionar"> Plugins
   <br>
   <input type="checkbox" name="si" class="controlseleccionar"> Otra cosa
</form>
```

Ahora, con un poco de jQuery invocamos el plugin:

```
$("#seleccionardeseleccionar").seleccionarDeseleccionarTodo(".controlseleccionar");
```

Como se puede ver, se invoca sobre el elemento DIV donde queremos ver los enlaces y luego se envía el selector para los campos checkbox, que es ".controlseleccionar", pues a todo el grupo de INPUT les pusimos esa clase CSS.

Podríamos también invocar el código del plugin enviando el objeto de opciones en un segundo parámetro:

```
$("#seleccionardeseleccionar").seleccionarDeseleccionarTodo(".controlseleccionar", {
```

```
   habilitados: false
});
```

Eso es todo. Ahora si lo deseamos, podemos acabar echando un vistazo al ejemplo en marcha.

Artículo por Miguel Angel Alvarez

Array de plugins Button de jQuery UI

Un taller jQuery para utilizar dos componentes de interfaces de usuario de las librerías jQuery UI, los plugins Button y Dialog.

Aquí tenemos un nuevo ejemplo de uso de las librerías jQuery UI, para crear interfaces de usuario en el framework Javascript jQuery. Es una sencilla práctica donde mostraremos el modo de trabajo con los plugins Button y Dialog, que ya habíamos empezado a explicar anteriormente en los artículos Plugin Button de jQuery UI y plugin Dialog de jQuery UI.

En este ejemplo generaremos un conjunto de botones con las letras del abecedario y los insertaremos en el código HTML de la página. Luego definiremos un evento clic, para cada uno de los botones, que lance una ventana de diálogo mostrando la letra pulsada. Esto se podría hacer con Javascript normal, con botones HTML y cajas de Alert, pero nosotros lo vamos a hacer utilizando los plugins de jQuery UI mencionados anteriormente, con lo que disfrutaremos de una interfaz de usuario muy mejorada.

Lo iremos viendo por partes, para que se pueda entender todo mejor, pero antes de comenzar, recomendamos ver el ejemplo en marcha para saber exactamente el objetivo de este ejercicio.

Partimos de una capa, con una etiqueta DIV inicialmente vacía, donde insertaremos los botones dinámicamente. Esa capa está en el código HTML de la página y le hemos puesto un identificador "botonesletras".

```
<div id="botonesletras"></div>
```

Bucle Javascript para recorrer las letras del abecedario

El primer paso es hacer un bucle sobre las letras del alfabeto y esto lo haremos generando un array con todas las letras y utilizando un bucle for para recorrerlo.

```
var letras = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'Ñ', 'O', 'P',
'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'];
for(i=0; i<letras.length; i++){
   //recorro todas las letras y hago los botones
}
```

Generar cada botón e insertarlo en la página

Ahora, dentro del bucle anterior, tenemos que generar cada uno de los botones y para ello crearemos dinámicamente elementos SPAN con la letra actual. Los convertiremos en botones y los insertaremos en la página.

```
var letras = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'Ñ', 'O', 'P',
'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'];
for(i=0; i<letras.length; i++){
   //creo el span de la letra
   letraActual = $('<span class="botonletra">' + letras[i] + '</span>');
   letraActual.data("letra",letras[i]);
   //lo convierto en un botón
   letraActual.button();
   //lo inserto en la página, en la capa con id botonesletras
   $("#botonesletras").append(letraActual);
}
```

Fijarse que todos los SPAN los creo con la clase "botonletra", pues luego los seleccionaré todos a la vez por medio de esa clase para crear el evento en todos ellos de una sola vez.

Luego almaceno en el propio elemento un dato, para poder recordar a qué letra pertenece este SPAN. Este paso lo hago con la intención de saber qué letra se ha pulsado cuando se defina el evento.

Convierto el SPAN en un botón con una llamada al método button(), que es la manera de invocar el plugin de Button de jQuery UI.

Acabo el bucle insertando la letra actual, convertida en botón, en el elemento "#botonesletras" (El DIV mencionado al principio del artículo).

Crear el evento click en los botones

Ahora creemos el evento click, para mostrar la tecla pulsada en una caja de diálogo.

El evento se crea con el método click(), invocado sobre los elementos de la clase "botonletra", que es la class de CSS que puse en los SPAN de cada botón.

```
$(".botonletra").click(function(){
   //creo una capa con la letra pulsada
   var caja = $('<div class="dialogletra" title="Has pulsado una letra">' + $(this).data("letra") +
'</div>');
   //la convierto en caja de diálogo modal box
   caja.dialog({
      modal: true,
      buttons: {
         "Ok": function(){
            $(this).dialog("close");
         }
      },
      show: "fold",
      hide: "scale"
   });
})
```

Como se habrá visto, con un único código se define el evento sobre todos los botones. En el código del evento se hacen un par de sencillos pasos para generar las cajas de diálogo. El primero para crear una capa y el segundo para convertir esa capa en una caja de diálogo Modal Box, por medio del plugin Dialog de jQuery UI.

Hay que fijarse que en la capa creada dinámicamente escribimos el valor de la letra pulsada, que recuperamos por medio del método data() del elemento que recibe el evento. La caja de diálogo mostrará el contenido de esa capa creada dinámicamente, como se explicó al ver el plugin Dialog de jQuery UI.

Código completo de la práctica con jQuery UI

Para acabar, mostramos el código completo de la página, donde se podrá ver también otras partes no comentadas ahora porque ya las hemos visto anteriormente en DesarrolloWeb.com, en los manuales de jQuery y jQuery UI.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
   <title>Probando el plugin button de jQuery UI</title>
   <link type="text/css" href="css/dot-luv/jquery-ui-1.8.1.custom.css" rel="Stylesheet" />
   <script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
   <script type="text/javascript" src="js/jquery-ui-1.8.1.custom.min.js"></script>
<script>
$(document).ready(function(){
   var letras = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'Ñ', 'O',
'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'];
   for(i=0; i<letras.length; i++){
      //creo el span de la letra
      letraActual = $('<span class="botonletra">' + letras[i] + '</span>');
      letraActual.data("letra",letras[i]);
      //lo convierto en un botón
      letraActual.button();
      //lo inserto en la página, en la capa con id botonesletras
      $("#botonesletras").append(letraActual);
```

```
   }
   $(".botonletra").click(function(){
      var caja = $('<div class="dialogletra" title="Has pulsado una letra">' + $(this).data("letra")
+ '</div>');
      caja.dialog({
         modal: true,
         buttons: {
            "Ok": function(){
               $(this).dialog("close");
            }
         },
         show: "fold",
         hide: "scale"
      });
   })
});
</script>
<style type="text/css">
body{
   font-size: 0.7em;
}
.botonletra{
   font-size: 0.8em;
   margin: 2px;
}
.dialogletra{
   font-size: 3em;
   text-align: center;
   padding-top: 15px;
}
</style>
</head>

<body>

<div id="botonesletras"></div>
</body>
</html>
```

Y finalizamos poniendo el enlace del ejemplo en funcionamiento.

Artículo por Miguel Angel Alvarez