Manual de jQuery UI

Miguel Angel Alvarez
Jaime Peña Tresancos

desarrolloweb.com

desarrolloweb.com/manuales/manual-jqueryui.html

Introducción: Manual de jQueryUI

Manual que explica cómo utilizar jQueryUI, una librería Javascript para la creación de interfaces de usuario enriquecias del lado del cliente, basado en jQuery.

Encuentras este manual online en:
http://desarrolloweb.com/manuales/manual-jqueryui.html

Autores del manual

Las siguientes personas han participado como autores escribiendo artículos de este manual.

Miguel Angel Alvarez

Miguel es fundador de DesarrolloWeb.com y la plataforma de formación online EscuelaIT. Comenzó en el mundo del desarrollo web en el año 1997, transformando su hobby en su trabajo.

Jaime Peña Tresancos

Escritor. Colaborador habitual de revistas de tecnología y experto en nuevas tecnologías y programas Microsoft

Introducción a jQueryUI

Qué es jQueryUI y cuáles son los pasos que debes realizar para comenzar a usar estas librerías para la creación de interfaces de usuario enriquecidas y compatibles con todos los navegadores.

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.

Ahora 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.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *18/05/2010*
Disponible online en *http://desarrolloweb.com/articulos/primeros-paso-jquery-ui.html*

Videotutorial de introducción a jQueryUI

Videotutorial en el que mostraremos qué se debe hacer para comenzar a usar las librerías para la creación de interfaces de usuario enriquecidas jQueryUI.

Uno de los proyectos más interesantes que están basados en el framework Javascript jQuery es jQueryUI, unas librerías para la creación de interfaces de usuario avanzadas que nos ofrecen los propios creadores de jQuery. En este videotutorial mostraremos hasta qué punto es sencillo dar los primeros pasos con estas librerías y comenzar a utilizarlas en páginas web.

Las librerías jQueryUI nos ofrecen una serie de componentes, también llamados widgets, que están listos para usar e implementan muchas de las necesidades más típicas a la hora de construir una página web, como seleccionadores de fecha, sistemas de autocompletado, pestañas, acordeón, etc. Pero además contiene algunas herramientas que permiten asignar funcionalidades avanzadas a elementos de la página, como arrastrar y soltar. Todo ello se ofrece por medio de plugins jQuery que resulta extremadamente sencillos de usar.

No hace falta dar más introducciones porque, a estas alturas, el que quiera aprender jQuery ya debe conocer el Manual de jQuery de DesarrolloWeb.com, así como los videotutoriales de

jQuery ya publicados en este sitio. Por su parte, el que quiera aprender jQueryUI quizás ya conozca el Manual de jQueryUI que venimos publicando también desde hace tiempo. De hecho, este vídeo sirve para dar un soporte adicional a los estudiantes que están dando los primeros pasos en jQueryUI ayudados por este último manual.

Normalmente, cuando publicamos un videotutorial, acompañamos el mismo con diversas explicaciones para que los interesados puedan conocer de antemano qué es lo que vamos a trata. Sin embargo, en esta ocasión está un poco de más, dado que el guión del vídeo es directamente el que podemos encontrar en el artículo Primeros pasos con jQuery UI. Así pues, en el vídeo iremos recorriendo todos los puntos tratados en ese artículo y que nos pueden dar una idea de la facilidad de uso de jQueryUI (creo que incluso sin saber jQuery cualquier desarrollador que vea este vídeo sería capaz de implementar las funcionalidades principales de estas librerías) y de la cantidad de componentes que pone a disposición de los creadores de aplicaciones web.

El videotutorial mostrará entonces cómo descargar jQueryUI, cómo incluir las librerías en una página web y seguidamente cómo implementar algunos de los componentes más esenciales. En concreto se mostrará cómo hacer un selector de fechas (también llamado datepicker), como implementar una funcionalidad de arrastrar y cómo crear una interfaz de acordeón. Todo ello, para el asombro quizás de los más nóveles, con únicamente una línea de código Javascript!!

> **Nota:** Para no decepcionar a nadie, cabe aclarar que, si bien se utilizará una línea de código por cada componente que vamos a probar en este vídeo, en realidad hay que colocar dos líneas adicionales de código jQuery para poder iniciar las funcionalidades solo cuando el navegador esté listo para ejecutarlas. Por supuesto, también habrá que incluir las librerías correspondientes, que es donde realmente está toda la infraestructura que permitirá la creación de estas interfaces casi sin esfuerzo.

Para vídeos siguientes, o para leer en el Manual de jQueryUI dejamos todas las opciones de configuración de los widgets. Es decir, nosotros en este videotutorial mostraremos simplemente como inicializar los componentes con las opciones por defecto, pero es bueno saber que cada uno de ellos tiene una cantidad importante de opciones de configuración que nos ayudarán a personalizarlos todavía más, aparte de eventos y métodos que nos permitirán ir un poco más allá cuando estamos definiendo la interacción con el usuario.

Así pues, os dejamos con el vídeo que espero sea ameno para todos y sirva para quitarse el miedo a usar unas de las librerías más atractivas que nos podemos encontrar en estos momentos para la creación de páginas enriquecidas del lado del cliente.

> Para ver este vídeo es necesario visitar el artículo original en:
> http://desarrolloweb.com/articulos/videotutorial-introduccion-jqueryui.html
>
> Este artículo es obra de *Miguel Angel Alvarez*
> Fue publicado por primera vez en *07/10/2011*
> Disponible online en *http://desarrolloweb.com/articulos/videotutorial-introduccion-jqueryui.html*

Interacción

Cómo implementar comportamientos complejos para la interacción con el usuario, como el arrastrar y soltar, que son típicos en interfaces avanzadas, a través de los plugins que nos ofrece jQueryUI.

Funcionalidad de arrastrar con jQueryUI

Cómo implementar el comportamiento arrastrar a través de jQueryUI y los elementos draggable.

Las aplicaciones de escritorio permiten arrastrar y soltar y es un comportamiento que los usuarios hacen con naturalidad. Desgraciadamente, las aplicaciones web no disponen de esa posibilidad de manera nativa, por lo que se tiene que implementar a través de complejas instrucciones Javascript.

Lo cierto es que sería difícil hacer, desde cero, un comportamiento de arrastrar y soltar en Javascript, que además funcione bien en todos los navegadores. Por suerte existen diversas librerías que implementan APIs para poder hacer posible esa comunicación usuario-aplicación y que cualquiera de nosotros podría utilizar sin necesidad de grandes complicaciones.

jQueryUI dispone de un método que vamos a mostrar llamado draggable() que permite aplicar a elementos de la página la funcionalidad de arrastrar, y en este artículo del Manual de jQueryUI demostraremos hasta qué punto es sencillo utilizarlo.

Plugin draggable: sencillez por encima de todo

Con draggable() ocurre lo que muchas veces deja con la boca abierta a más de uno, pues resulta tan sencillo de utilizar que no paro de maravillarme. Para hacer que cualquier elemento de la página se pueda arrastrar a cualquier lugar de la página, simplemente tendremos que invocar dicho método sobre el elemento que queramos.

Por ejemplo tenemos cualquier tag en la página:

```
<h1>Probando el comportamiento draggable</h1>
```

Es una etiqueta normal y corriente. Ni siquiera necesita que le indiquemos estilos CSS para que el comportamiento se pueda implementar. Y ahora invocamos al método draggable de jQueryUI.

```
$("h1").draggable();
```

Con eso conseguimos que todos los encabezamientos H1 de la página se puedan arrastrar a otro lugar.

Es así de sencillo, ya tenemos un titular que se puede arrastrar y por supuesto, podríamos hacerlo con cualquier otro elemento de la página y sirve para tanto elementos block como inline.

Por ejemplo, en este código:

```
Este texto es de <span id="d2">prueba</span> para arrastrar
```

Podríamos hacer que la palabra prueba se pudiera arrastrar, con la siguiente sentencia:

```
$("#d2").draggable();
```

Puedes ver estos ejemplos en una página aparte.

Opciones disponibles para la configuración de draggable

Además del comportamiento predeterminado, el método draggable permite recibir diversas opciones para personalizar un poco más la forma en la que se implementa. Existen multitud de opciones y lógicamente unas son más útiles que otras. Veremos a continuación algunas que hemos querido destacar, pero recordar siempre que en la documentación de las librerías encontraréis muchas otras opciones.

Definir el cursor del ratón durante el arrastre:

Esto lo conseguimos con la opción "cursor". Es tan sencillo como indicar el nombre que CSS da al cursor que queremos que se utilice.

```
$("#arrastrame").draggable({
   cursor: 'move'
});
```

Así se define que, durante el arrastre, se muestre el cursor del ratón que es como una cruz con puntas de flecha en cada uno de los extremos, que sería el adecuado para simbolizar que se está moviendo ese elemento.

Podemos ver un ejemplo en una página aparte.

Retrasar el arrastre en el tiempo:

Si lo deseamos, podemos conseguir que pase una pequeña fracción de tiempo antes que el arrastre comience. Sería como un pequeño retardo antes de que el elemento empiece a acompañar al ratón durante la acción de arrastrar y sirve para que no se produzcan arrastres no deseados cuando se hace clic sobre un elemento arrastrable.

```
$("#arrastrame").draggable({
   delay: 500
});
```

Puedes ver un ejemplo en funcionamiento.

Distancia para comenzar a arrastrar:

Esta opción es parecida a la anterior, pero en vez de jugar con el tiempo arrastrando, tiene en cuenta el espacio en píxeles a partir del cual el elemento se moverá.

```
$("#arrastrame").draggable({
   distance: 50
});
```

Acoplar a rejilla:

Una interesante opción que permite arrastrar, pero dentro de una rejilla. Por medio de esta opción se deben indicar dos valores en un array que servirían para especificar la anchura y altura de la rejilla donde se podría acoplar el elemento durante el arrastre.

```
$("#arrastrame").draggable({
   grid: [200, 50]
});
```

Así el arrastre se produciría siempre en una rejilla de 200 píxeles de ancho x 50 de alto. El arrastre va situando el elemento en cada uno de los espacios de esa rejilla imaginaria. Es decir, la rejilla no aparece en la página (a no ser que nosotros hagamos un fondo u otra cosa manualmente para que se muestre), sino que simplemente se utilizará para definir las posiciones válidas para situar ese elemento arrastrado.

Podemos ver un ejemplo en funcionamiento con la opción grid.

Eventos en el comportamiento de arrastrar

Claro que implementar un comportamiento de arrastrar así, sin más, no sirve de mucho, de modo que jQueryUI pone a nuestra disposición varias utilidades como los eventos, que nos permiten hacer cosas cuando el usuario realiza acciones que tienen que ver con arrastrar los elementos.

Existen 4 eventos distintos relacionados con la capacidad de arrastrar:

- **Evento create:** Que se produce cuando se aplica la funcionalidad de arrastre a un elemento.
- **Evento start:** Que ocurre cuando comienza un arrastre.

- **Evento drag:** Se producen múltiples eventos drag cuando el ratón se está moviendo durante el proceso de arrastrar.
- **Evento stop:** Que se produce cuando se suelta el elemento, es decir, en el momento que se deja de arrastrar.

El código asociado a los eventos se define tal como estamos acostumbrados a hacer en jQuery. O bien al crear el elemento:

```
$("#arrastrame").draggable({
   start: function(){
      $("#log").html("Se ha producido el evento start");
   }
});
```

O bien en cualquier otro momento con el método bind:

```
$("#arrastrame").bind("drag", function(evento, ui){
   $("#log").html("Se ha producido el evento darg");
});
```

Nota: Ten en cuenta que si utilizas bind() para definir un evento "stop" tienes que utilizar como nombre del evento "dragstop".

```
$("#arrastrame").bind("dragstop", function(evento, ui){
   $("#log").html("Se ha producido el evento stop");
})
```

Pero no fíjate que, si creas el evento al invocar a draggable(), tienes que utilizar el nombre original "stop".

```
$("#arrastrame").draggable({
   stop: function(){
      $("#log").html("Se ha producido el evento stop");
   }
});
```

Ahora vamos el código completo de un ejemplo donde se han puesto en marcha esos códigos de definición de eventos.

Por un lado tenemos dos elementos HTML:

```
<div id="arrastrame" class="ui-corner-all ui-state-highlight">Arrastra suavemente esta capa!</div>
<div id="log"></div>
```

> **Nota:** Las clases aplicadas al primer DIV "ui-corner-all" y "ui-state-highlight" son de las existentes en el Framework CSS de jQueryUI.

Por otro lado el siguiente Javascript:

```
$(document).ready(function(){
   $("#arrastrame").draggable({
      start: function(evento, ui){
         $("#log").html('<span style="color: #090;">Se ha producido el evento start</span>' + "<br>" + $("#log").html());
      }
   });
   $("#arrastrame").bind("drag", function(evento, ui){
      $("#log").html('<span style="color: #009;">Se ha producido el evento darg</span>' + "<br>" + $("#log").html());
   })
   $("#arrastrame").bind("dragstop", function(evento, ui){
      $("#log").html('<span style="color: #900;">Se ha producido el evento stop</span>' + "<br>" + $("#log").html());
   })
})
```

El ejemplo simplemente define los tres eventos relatados anteriormente y cuando se producen va escribiendo mensajes en la capa con id="log". Comprobarás que al arrastrar se ejecuta un evento start, seguido de tantos eventos log como píxeles movamos el elemento arrastrado. Finalmente, al soltar el botón del ratón y dejar de arrastrar se produce el evento stop.

Puedes ver el ejemplo en marcha, pero ten en cuenta que tendrás que arrastrar con suavidad para que no se te llene la pantalla de mensajes por el evento drag que se ejecuta muchas veces.

Como has podido ver en el código anterior, cada función que implementa un evento recibe dos parámetros. El primero es el objeto evento del navegador y el segundo es un objeto relacionado con el elemento draggable, del que podemos obtener algunos datos.

Utilizando un nombre de parámetro llamado ui, los datos a los que accedemos serán los siguientes:

- **ui.helper:** es el objeto jQuery que se ha transformado en elemento draggable.
- **ui.position:** es un objeto que da la posición relativa del elemento draggable, con respecto a su contenedor. Tiene dos propiedades {top, left} para acceder a sus dos coordenadas.
- **ui.offset:** es un objeto que da la posición, pero en este caso la absoluta con respecto al comienzo de la página.

Métodos adicionales de los elementos draggable

Los elementos que hemos convertido en draggable tienen una serie de métodos adicionales implementados, que podemos invocar cuando se desee. Para ello tenemos que invocar al mismo método draggable(), pero indicando entre paréntesis lo que queremos hacer.

Por ejemplo, el método "destroy" elimina la funcionalidad de arrastrar de un elemento

draggable.

```
$("#elementodraggable").draggable("destroy");
```

El método "disable" aplica el estilo desactivado en el elemento draggable y momentáneamente no permite moverlo.

```
$("#arrastrame").draggable("disable");
```

Si queremos volver a activar la funcionalidad de draggable sobre un elemento que se ha desactivado previamente, tenemos que invocar al método "enable".

El método "option" permite asignar nuevos valores a las opciones del elemento draggable, una vez ha sido creado. Por ejemplo, vamos a hacer que el elemento vuelva a su posición original después de soltarlo.

```
$("#arrastrame").draggable("option", "revert", true);
```

Por último, el método "widget" nos sirve para devolver el elemento que se marcó como draggable.

```
elementoDraggable = $("#arrastrame").draggable("widget");
```

> **Nota:** En el ejemplo dedicado a los eventos pudiste encontrar diversos enlaces que hacen uso de los métodos relatados aquí.

Conclusión

Si te habías imaginado alguna vez una interfaz web con elementos que se pudieran arrastrar pero la has descartado por ser algo muy difícil de conseguir, con jQueryUI puedes replantearte ese objetivo. Si a tu cliente se le ha ocurrido una aplicación en la que haya que arrastrar elementos, no te preocupes! ya no hace falta imaginar argumentos para quitarle la idea de la cabeza, con jQueryUI lo tienes muy fácil.

Como hemos visto, en pocos segundos puedes crear elementos que se pueden arrastrar y existen multitud de maneras de controlarlos y de realizar distintas operaciones cuando se arrastran. En el próximo artículo del Manual de jQueryUI seguiremos tratando otras cosas relacionadas con los elementos draggable y encontrarás mucha más información y ejemplos para seguir aprendiendo.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *28/09/2011*
Disponible online en *http://desarrolloweb.com/articulos/funcionalidad-arrastrar-jqueryui.html*

Preguntas y respuestas con el elemento draggable de jQueryUI

Algunas cuestiones que nos parece útil explicar sobre el plugin draggable de jQueryUI, como modificar sus estilos, averiguar su posición o restringir su localización a un contenedor.

En el artículo anterior ya ofrecimos una bastante completa guía de uso del plugin draggable de jQueryUI, para implementar elementos que se puedan arrastrar en la página. Lo cierto es que el plugin draggable es tan completo que nos quedaron bastantes cosas por ver y para ello hemos creado un ejemplo que nos sirve para responder a varias de las cuestiones más típicas que existen sobre la funcionalidad de arrastrar elementos.

Hemos orientado este artículo como si fuera una lista de preguntas y respuestas. Primero nos plantearemos todas ellas de manera independiente y para terminar crearemos un ejemplo global que las pondrá en práctica lo aprendido.

¿Cómo puedo definir estilos CSS para los elementos draggables?

Cuando se convierte un elemento para que se pueda arrastrar, invocando el plugin draggable, jQueryUI coloca automáticamente una clase CSS llamada "ui-draggable". Podemos usar esa clase para aplicar los estilos que queramos a los elementos que se puedan arrastrar.

```
.ui-draggable{
   background: #feb;
}
```

¿Existe algún atajo para definir un estilo especial a utilizar en el momento en el que se está arrastrando un elemento?

Cuando un elemento draggable está siendo arrastrado, jQueryUI coloca de manera automática una clase CSS llamada "ui-draggable-dragging" que podríamos utilizar para definir estilos que se aplicarían sobre el elemento únicamente cuando está siendo arrastrado.

```
.ui-draggable-dragging{
   background: #def;
}
```

>

¿Existe alguna manera de colocar un elemento draggable dentro de un contenedor y que, al moverlo, si se sobrepasa el contenedor, que salgan unas barras de

desplazamiento?

Cualquier elemento draggable está realmente en un contenedor, aunque sea el propio documento global. Puedes colocarlo en un contenedor simplemente anidando ese elemento arrastrable en el código HTML sobre ese contenedor. Las barras de desplazamiento cuando el elemento sale del contenedor se consiguen si en el contenedor colocas el atributo overflow: auto.

El código HTML sería:

```
<div id="contenedor">
   <div id="arrastrame">Arrastra esta capa!</div>
</div>
```

El código CSS del contenedor:

```
#contenedor{
   width: 500px;
   height: 300px;
   border: 1px solid #ccc;
   overflow: auto;
}
```

¿Puedo conseguir que el elemento arrastrable no sobrepase nunca los límites marcados por el contenedor?

La opción containment de los elementos draggable permite colocar varias posibilidades para definir qué elementos deben trabajar como contenedores. Además hace que la posición del elemento arrastrable se restrinja a las que permita el espacio disponible en el contenedor. Un valor típico sería 'parent', que indica que la etiqueta padre del elemento arrastrable debe trabajar como contenedor. Pero también admite otros valores como 'document', 'window', un selector o incluso unas coordenadas en un array como [100, 100, 300, 300], para que se restrinja al espacio definido por los dos puntos [x1, y1, x2, y2].

```
$("#arrastrame").draggable({
   containment: 'parent'
})
```

¿En un evento, como puedo obtener una referencia al objeto jQuery que está siendo arrastrado?

Puedes definir acciones a realizar cuando arrastras un elemento, por medio del evento drag u otros eventos como start o stop. Dentro del código de las funciones definidas para esos eventos puedes acceder al objeto jQuery por medio de la variable this. Pero recuerda que para invocar métodos del framework debes convertirla en un objeto jQuery con la función dólar.

```
$("#arrastrame").draggable({
   drag: function(event, ui){
      $(this).html("ui-state-highlight");
   }
})
```

Además tienes acceso a un helper, con una referencia al objeto que está siendo arrastrado por medio del segundo parámetro recibido por la función que ejecuta el evento y la propiedad helper.

```
$("#arrastrame").draggable({
   drag: function(event, ui){
      ui.helper.html("Estoy siendo arrastrado");
   }
})
```

En la función que define el evento drag has recibido ui como segundo parámetro. Por medio de ui.helper tienes el objeto jQuery que se está arrastrando. A través de ese objeto puedes invocar métodos de jQuery para hacer cosas con ese elemento, como:

```
ui.helper.html("Estoy siendo arrastrado");
```

Eso hace que se cambie el HTML del elemento arrastrado para colocar el texto "Estoy siendo arrastrado".

¿Cómo puedo saber la posición de un elemento arrastrable?

En el mencionado parámetro ui, que se recibe en la función que implementa un evento, aparte de helper, existen otras dos propiedades que te servirán para acceder a la posición de ese elemento. Ambas propiedades son objetos con atributos top y left, que podemos acceder para saber la posición del elemento.

Para obtener la posición relativa al contenedor donde está situado el elemento draggable se utiliza la propiedad "position".

```
$("#arrastrame").draggable({
   drag: function(event, ui){
      $("#posx").text(ui.position.left);
      $("#posy").text(ui.position.top);
   }
})
```

Para acceder a la posición absoluta, cuyo origen de coordenadas sería la esquina superior derecha del espacio disponible para el documento HTML, se utiliza la propiedad "offset".

```
$("#arrastrame").draggable({
   drag: function(event, ui){
      $("#offsetx").text(ui.offset.left);
      $("#offsety").text(ui.offset.top);
   }
})
```

¿Puedo hacer que el elemento arrastrable se pueda soltar en un lugar determinado y hacer cosas cuando esto ocurra?

Las posibilidades de implementar scripts con los elementos draggable son tantas como desees. Pero en el caso de arrastrar y soltar en un lugar determinado ten en cuenta que en jQueryUI dispone de un plugin adicional llamado Droppable. Por medio de los elementos droppable podemos definir contenedores donde es posible soltar elementos draggable y existe un evento específico para hacer cosas cuando se produzca la acción de soltar. Es decir, el elemento droppable ya está preparado con la funcionalidad de detectar cuándo otro elemento ha sido arrastrado y soltado encima de él.

Ejemplo que ilustra algunas preguntas y respuestas sobre el elemento draggable

Ahora vamos a ver un ejemplo donde hemos puesto en marcha los conocimientos adquiridos al responder las anteriores preguntas.

Podemos ver directamente el código fuente, pues con lo comentado anteriormente espero que el lector pueda identificar las partes del código que responden las preguntas.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"
>
<html lang="es">
<head>
<title>Probando el comportamiento draggable de jQueryUI</title>
   <link type="text/css" href="css/blitzer/jquery-ui-1.8.16.custom.css" rel="Stylesheet" id="linkestilo" />
   <style type="text/css">
   body{
      font-family: sans-serif;
   }
   h1{
      padding: 10px;
      font-size: 30px;
   }
   #contenedor{
      width: 500px;
      height: 300px;
      border: 1px solid #ccc;
      overflow: auto;
      float: left;
   }
   #arrastrame{
      position: relative;
```

```
      top: 10px;
      left: 20px;
      width: 160px;
      padding: 7px;
      font-size: 10px;
      font-weight: bold;
      text-align: center;
   }
   #posicion{
      font-size: 12px;
      margin-left: 520px;
      line-height: 150%;
   }
   .ui-draggable{
      background: #feb;
   }
   .ui-draggable-dragging{
      background: #def;
   }
   </style>
   <script type="text/javascript" src="js/jquery-1.6.2.min.js"></script>
   <script type="text/javascript" src="js/jquery-ui-1.8.16.custom.min.js"></script>
   <script type="text/javascript">
      $(document).ready(function(){
         $("#arrastrame").draggable({
            containment: 'parent',
            drag: function(event, ui){
               $("#posx").text(ui.position.left);
               $("#posy").text(ui.position.top);

               $("#offsetx").text(ui.offset.left);
               $("#offsety").text(ui.offset.top);
            }
         })
      })
   </script>
</head>
<body>
   <h1 class="ui-state-default ui-corner-all">Probando la funcionalidad "draggable"</h1>
   <div id="contenedor">
      <div id="arrastrame" class="ui-corner-all ui-state-highlight">Arrastra esta capa!</div>
   </div>
   <div id="posicion">
      Coordenadas del elemento relativas al contenedor:
      <br>
      X: <span id="posx"></span>
      <br>
      Y: <span id="posy"></span>
      <br>
      <br>
      Coordenadas del elemento absolutas a la página:
      <br>
      X: <span id="offsetx"></span>
      <br>
      Y: <span id="offsety"></span>
```

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

El ejemplo se puede ver en una página aparte.

Conclusión

Seguramente haya muchas cosas en el aire todavía sobre los elementos arrastrables que podemos crear con jQuery UI. No obstante, aquí hemos podido encontrar respuestas a lo que podrían ser las dudas más típicas.

Solo queda recomendar echar un vistazo a la documentación, donde podremos encontrar bastantes más ejemplos y una descripción completa de opciones de configuración, eventos y métodos que permiten los elementos con la funcionalidad de arrastrar en este framework.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *04/10/2011*
Disponible online en *http://desarrolloweb.com/articulos/preguntas-respuestas-draggable-jqueryui.html*

Implementar la acción de soltar, de drag & drop, con droppable de jQueryUI

Cómo utilizar las librerías jQueryUI y el plugin droppable para realizar cosas cuando un usuario suelte un elemento sobre otro, en una operación de arrastrar y soltar (drag & drop).

En la operación de arrastrar y soltar participan dos elementos: El primero es el elemento que arrastras y el segundo es el elemento donde sueltas el primero. Por ello, las librerías jQueryUI mantienen dos clases distintas para realizar un ejercicio completo de arrastrar y soltar. Por un lado tenemos el plugin draggable, que se aplica sobre los elementos que quieres que se arrastren, y por otro el plugin droppable, que se aplica sobre aquellos elementos que quieres que se enteren cuando les sueltas algo encima.

En los artículos anteriores del Manual de jQueryUI estuvimos hablando en profundidad sobre los elementos draggable, de modo que ahora vamos a ver la otra parte que nos queda, es decir, los elementos droppable. Aunque sean dos clases distintas, lo cierto es que si has podido entender bien como funciona draggable, no tendrás ningún problema en entender los detalles sobre droppable.

Usando el plugin droppable

Quizás nos sorprendió lo sencillo que era conseguir que un elemento se pudiera arrastrar con jQueryUI, así que ahora no nos sorprenderá tanto la facilidad de detectar que se ha soltado en un lugar determinado. Para ello usamos este sencillo código.

```
$("#soltable").droppable();
```

Con esto conseguimos que el elemento con id="soltable" detecte cuando algún elemento draggable se suelte encima de él. Claro que para hacer algo cuando arrastren un elemento hasta él, no nos servirá con invocar al plugin droppable, sino que tendremos que realizar alguna otra acción y para ello entran en juego los eventos.

Si queremos hacer algo cuando se suelte un elemento encima de un droppable, tendremos que definir el evento "drop", de esta manera.

```
$("#soltable").droppable({
   drop: function( evento, ui ) {
      $(this).html("Lo soltaste!!!");
   }
});
```

Ese evento "drop" provocaría que, al soltar un elemento arrastrable encima del dropable, se cambie el texto que hay escrito en el elemento droppable.

Se puede ver un ejemplo de draggable que se puede soltar en un droppable.

Opciones de personalización de un elemento droppable

Existen diversas opciones de configuración de los elementos droppable que se pueden asignar, tanto en el momento de invocar el plugin, como después de haber sido creado el droppable por medio del método "option" del plugin.

Al invocar el plugin indicamos las opciones de configuración tal como estamos acostumbrados a hacer en jQuery, con notación de objeto y por medio de una serie de propiedades. Por ejemplo, podemos configurar la tolerancia con la que un elemento se acepta un elemento soltado con la propiedad de configuración "tolerance".

```
$("#soltable").droppable({
   tolerance: "fit"
});
```

Con esto conseguimos que el elemento que soltamos, sólo se considere soltado en el objetivo cuando está completamente dentro del mismo.

> **Nota:** La configuración por defecto de tolerance es "intersect", que significa que el elemento también se puede soltar en cualquier intersección, con tal que el 50% del área del elemento soltado esté dentro del elemento que lo recibe. Consultar otras posibilidades de configuración en la documentación de jQueryUI.

Interesante es también la opción "accept" o "scope" (ambas sirven para lo mismo). Con ellas indicamos el rango de elementos aceptaría recibir un droppable, por medio de un selector. Se tiene que configurar como valor a alguna de esas dos propiedades, o las dos, el selector o selectores que sí se aceptarían.

> **Nota:** En el caso de la opción "accept" también te permite indicar una función, que se ejecutaría -una vez por cada elemento de manera independiente- para saber si se acepta un elemento o no. La función debería devolver true para saber si ese elemento está o no permitido soltar y recibe como parámetro el elemento que se está soltando.

```
$("#soltable").droppable({
   accept: ".tarea"
});
```

Ese elemento solo aceptaría que se le soltasen elementos con el class "tarea". Para ser más exactos, sólo esos elementos de clase "tarea" provocarían que se lanzase el evento drop.

En la propia documentación de jQueryUI encontraréis otra serie de opciones interesantes.

Eventos en los elementos droppable

Podemos definir varios eventos distintos relacionados con los elementos droppable. Son los siguientes:

- **Evento create**: ocurre cuando se crea el elemento droppable.
- **Evento activate**: se produce cuando un elemento draggable que se podría aceptar por el droppable empieza a arrastrarse.
- **Evento deactivate**: ocurre cuando un elemento draggable que se podría hacer aceptado se deja de arrastrar.
- **Evento over**: ocurre cuando un elemento draggable que fuese le aceptable entra en un área de tolerancia donde se podría soltar sobre el droppable.
- **Evento out**: lo mismo que over pero cuando sale de la tolerancia.
- **Evento drop**: cuando se suelta un elemento draggable sobre un droppable, siempre y cuando el draggable sea aceptado por el droppable y se suelte dentro del área definida por la tolerancia.

Así se definiría un evento en el momento de creación del droppable.

```
$("#soltable").droppable({
   out: function( evento, ui ) {
      $(this).html("Ahora saliste...");
   }
});
```

Ese evento produciría que se cambiase el texto del droppable por "Ahora saliste", cuando un

elemento draggable, que sea aceptable, se salga del área de tolerancia.

También podemos definir eventos una vez se ha creado el droppable con el método bind() de jQuery.

```
$( "#soltable" ).bind( "drop", function(evento, ui) {
   //código del evento
});
```

Nota: al igual que el pluggin draggable, las funciones que se ejecutan a raíz de los eventos de droppable reciben dos parámetros. El primero es el propio objeto evento del navegador y el segundo es un objeto que tiene varias propiedades. Nosotros lo estamos recibiendo siempre con el nombre ui. En ese caso tendrá estas propiedades:

- **ui.draggable**: el objeto jQuery que corresponde con el elemento draggable que está provocando el evento.
- **ui.helper**: el helper del elemento draggable, otro objeto jQuery del elemento que se arrastra.
- **ui.position**: la posición del elemento draggable, relativa a donde esté contenido.
- **ui.offset**: la posición absoluta del elemento draggable.

Como se puede imaginar el lector, el evento clave a la hora de gestionar la utilidad de arrastrar y soltar es "drop". Por ejemplo, aquí tenemos un script para la definición de un evento "drop" que hace que, al soltar un elemento en el droppable, el elemento soltado cambie su color de fondo.

```
drop: function( event, ui ) {
   ui.draggable.css("background-color", "#ddd");
}
```

Si queremos hacer alguna cosa cuando el elemento se retira del área donde se podría soltar, tenemos que utilizar el evento "out". Pero ojo, que ese evento "out" se ejecuta siempre que se sale del área soltable, independientemente si antes lo habíamos soltado allí o no. Luego haremos un ejemplo para ver este asunto.

Métodos de los elementos "soltable"

Para acabar esta introducción sobre los componentes droppable de jQueryUI nos queda por comentar que existe una serie de métodos que se pueden invocar sobre ellos, para hacer cosas diversas, como habilitarlos o inhabilitarlos, cambiar sus opciones de configuración, etc.

Todos esos métodos se invocan como si estuviéramos invocando al propio plugin droppable, pero pasando como parámetro una cadena con el nombre del método a ejecutar, tal como estamos acostumbrados a hacer con diversos otros componentes jQueryUI, incluido el plugin draggable.

Por ejemplo, tenemos el método "disable" que sirve para inhabilitar un elemento droppable y se ejecutaría de la siguiente manera:

```
$("#elementodroppable").droppable("disable")
```

Otro ejemplo interesante es el método option, que sirve para definir opciones de configuración del elemento Lo ponemos en ejecución al pasar la palabra "option" y luego tenemos que indicar otro parámetro con el nombre de la opción al que queremos acceder, junto con un tercer parámetro con el nuevo valor de la opción, si es que deseábamos cambiarlo.

```
$("#soltable").droppable("option", "activeClass", "sueltaaqui");
```

Con eso conseguiríamos que, al iniciar el arrastre sobre un elemento que pueda soltarse en el elemento "#soltable", se aplique la clase "sueltaaqui". Esa clase CSS se pone en el elemento donde se puede soltar aquello que se está arrastrando, para que el usuario perciba dónde podría culminar su acción de drag&drop.

Si quisiéramos en algún momento saber qué clase se definió como "activeClass", podríamos hacerlo con:

```
$("#soltable").droppable("option", "activeClass")
```

Conclusión

Hasta el momento hemos ofrecido una introducción bastante completa sobre los componentes Droppable de jQueryUI, pero lo cierto es que las opciones de uso y configuración de los elementos "soltable" son bastante grandes. En cualquier caso, recuerda siempre que dispones de mucha más información en la propia documentación de jQuery UI.

Con lo visto hasta ahora ya debes tener una idea más o menos clara sobre cómo hacer sistemas de arrastrar y soltar medianamente complejos. No obstante, en el capítulo siguiente hacemos un nuevo ejemplo un poco más elaborado que seguramente te servirá para despejar algunas dudas que puedas tener todavía.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *14/12/2011*
Disponible online en *http://desarrolloweb.com/articulos/droppable-jqueryui.html*

Uso de los componentes disponibles en jQueryUI

Los componentes o widgets de jQueryUI permiten crear interfaces de usuario avanzadas y útiles para la creación de aplicaciones web enriquecidas. A continuación explicaremos cómo utilizar y exprimir las posibilidades de las librerías instalando componentes listos para usar.

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.

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.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *18/05/2010*
Disponible online en *http://desarrolloweb.com/articulos/componente-datepicker-jquery-ui.html*

jQuery UI Datepicker. Manual de uso simple

En el presente artículo hablaremos de qué es y qué proporciona el jQuery User Interface Datepicker.

En el presente artículo hablaremos de:

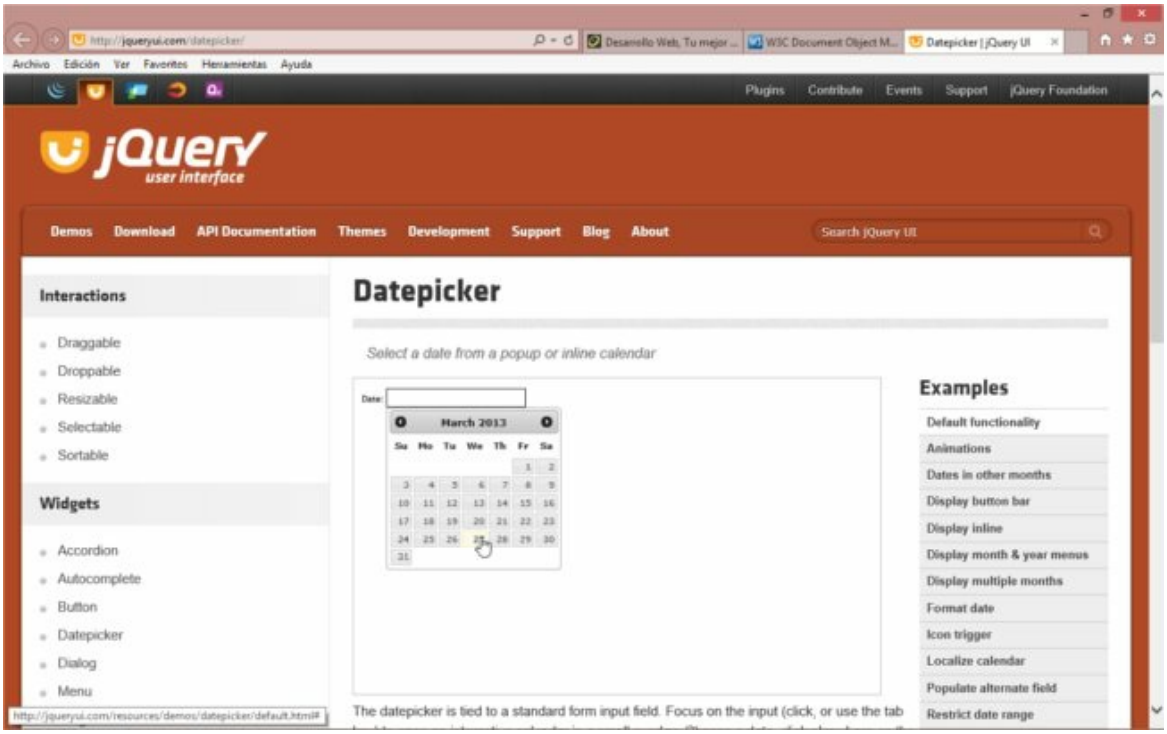
- Qué es y qué proporciona el *jQuery User Interface Datepicker*
- Las opciones básicas de presentación y adaptación idiomática de un calendario
- La gestión de las opciones de un *Datepicker*
- El uso de estilos y los estilos prediseñados
- Extracción de la información de una fecha seleccionada (*onSelect*)
- Asociación con una entrada de textos, por ejemplo de un formulario

¿Qué es el jQuery User Interface Datepicker?

Dentro del *jQuery User Interface* encontramos toda una serie de elementos de programación de la interfaz de usuario preprogramados y listos para ser integrados en nuestros proyectos HTML. Se trata de una amplia biblioteca *JavaScript* que abarca desde efectos dinámicos, hasta menús, calendarios, etc.

En concreto, el componente *Datepicker* nos proporciona un calendario totalmente personalizable, en el que podremos realizar selecciones de fechas y asociarlo a elementos HTML, como entradas de formularios.

El lugar de Internet de referencia y descarga es: jqueryui.com/datepicker



Las capacidades que nos proporcionará las iremos desgranando en éste y en un posterior artículo, a vuelapluma son:

- Navegación ágil mediante clics de ratón
- Extracción de fechas seleccionadas
- Asociación directa y automatizada con entradas de texto
- Opciones de internacionalización
- Enlace con menús de días y años
- Control de fechas seleccionables
- Ser mostrado y ocultado a demanda
- Personalización de su apariencia mediante estilos prediseñados o personales

Cómo mostrar un *Datepicker*

Comenzaremos por las tareas de inicialización y el cómo mostrar un calendario en nuestra página web; no será muy apasionante, dado que su funcionalidad será reducida. Se limitará a poder navegar en él como lo haríamos en el calendario de nuestro sistema operativo, para consultar fechas.

Observemos el Listado 1, que pasaremos a comentar con detalle.

######### Listado 1: Uso básico de un componente *Datepicker*

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Datepicker - Uso básico</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
```

```
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script>
$(function () {
$("#datepicker").datepicker();
});
</script>
</head>

<body>
Fecha:
<div id="datepicker"></div>
</body>
</html>
```

Si repasamos el código, nos encontramos, por orden secuencial:

- La referencia a la hoja de estilos del *User Interface de jQuery: jquery-ui.css* Debe ser la primera de las bibliotecas *JavaScript* referenciadas
- La referencia a la biblioteca *jQuery* general: *jquery-x.x.x.js*
- La referencia a la biblioteca *User Interface de jQuery: jquery-ui.js*
- Una función de inicialización del *Datepicker: $("#datepicker").datepicker();* Que, como vemos, toma el elemento al que va asociado y la función sin parámetros, sin más.
- La etiqueta –elemento- HTML al que va asociado el *Datepicker* en el cuerpo del documento

Opciones de localización idiomática

Se tratará de modificar en un archivo *JavaScript* complementario las opciones del *Datepicker* para adaptar a nuestro idioma los nombres de los meses y días, el formato de fecha, el día de comienzo de la semana y similares.

En el Listado 2 programamos un *Datepicker* similar al del Listado 1, pero con las adaptaciones necesarias para el idioma y usos del español.

######### Listado 2: Opciones de localización idiomática

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Datepicker - Opciones de localización</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script src="jquery.ui.datepicker-es.js"></script>

<script>
$(function () {
$.datepicker.setDefaults($.datepicker.regional["es"]);
$("#datepicker").datepicker({
firstDay: 1
});
});
</script>
</head>

<body>
Fecha:
<div id="datepicker"></div>
</body>
</html>
```

Ahora el orden de declaración de los archivos será:

- Fijamos la opción *fisrtDay* igual a 1, esto es, que la semana comience el lunes –realmente lo ponemos como algo ilustrativo de cómo programar opciones personalizadas, dado que ya se incluye en la biblioteca de localización-.
- La referencia a la hoja de estilos del *User Interface de jQuery*
- La referencia a la biblioteca *jQuery* general
- La referencia a la biblioteca *User Interface de jQuery*
- La referencia a la biblioteca de localización idiomática del *User Interface: jquery.ui.datepicker-es.js* Que debe ir tras de la anterior, dado que contiene redefiniciones de opciones –propiedades- del apartado *Datepicker del jQuery User Interface*

El archivo *jquery.ui.datepicker-es.js* ha de obtenerse de Internet, por ejemplo de la dirección URL siguiente: http://code.google.com/p/jquery-ui/source/browse/branches/labs/datepicker2/ui/i18n/jquery.ui.datepicker-es.js?r=3875

Nosotros hemos notado que dicho archivo tenía inconvenientes con los caracteres especiales trabajando con la especificación UTF-8 activada, por lo cual procedimos a modificarlo, tal cual se muestra en el Listado 3. Cada uno podrá tomarlo como base para su personalización.

######### Listado 3: Archivo *jquery.ui.datepicker-es.js* ajustado para UTF-8, que ha sido el que hemos empleado en todos nuestros ejemplos

```
jQuery(function ($) {
$.datepicker.regional['es'] = {
closeText: 'Cerrar',
prevText: ' 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']);
});
```

Personalización de opciones -propiedades-

Lo primero que deberemos conocer son las opciones –y no está de más, también los métodos- disponibles para la gestión de un *Datepicker*. La documentación oficial completa se puede obtener partiendo de la dirección URL siguiente: api.jqueryui.com/datepicker

Normalmente nos interesará modificar las opciones de presentación del Datepicker, accediendo a ellas en el momento de su creación.

Así, en la propia función de definición del *Datepicker* tendremos que modificar las opciones –propiedades si lo prefiere- por defecto.

Veámoslo con un ejemplo, codificado en el Listado 3, en el que cambiaremos los nombres de los meses y los nombres cortos de los días, para mimetizar un calendario revolucionario francés –nótese que no cumple las normas de comienzo del año en primavera, ni la semana de diez días, por ejemplo-.

######### Listado 4: Personalización de opciones. Mimetización de un calendario revolucionario francés

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Datepicker - Revolucionario francés</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>

<script>
$(function () {
$("#datepicker").datepicker({
firstDay: 1,
monthNames: ['Nivôse', 'Pluviôse', 'Ventôse',
```

```
'Germinal', 'Floréal', 'Prairial',
'Messidor', 'Thermidor', 'Fructidor',
'Vendémiaire', 'Brumaire', 'Frimaire'],
dayNamesMin: ['sep', 'pri', 'duo', 'tri', 'qua', 'qui', 'sex']
});
});
</script>
</head>

<body>
Fecha:
<div id="datepicker"></div>
</body>
</html>
```

Lo esencial de las modificaciones es bien simple:

- Como ya se ha explicado, fijamos *firstDay* igual a 1, esto es, la semana comienza en el día que correspondería a nuestro lunes –el 0 es el domingo y el 6 el sábado-.
- Definimos el contenido de *monthNames* con la matriz con las cadenas de nombres de los meses del año.
- Definimos el contenido de *dayNamesMin* con la matriz con las cadenas de nombres cortos de los días de la semana. Obsérvese que se comienza con el que correspondería al domingo y se finaliza en el que correspondería al sábado.
- En todo caso, el modo de hacerlo es: Propiedad: Valor de la propiedad.

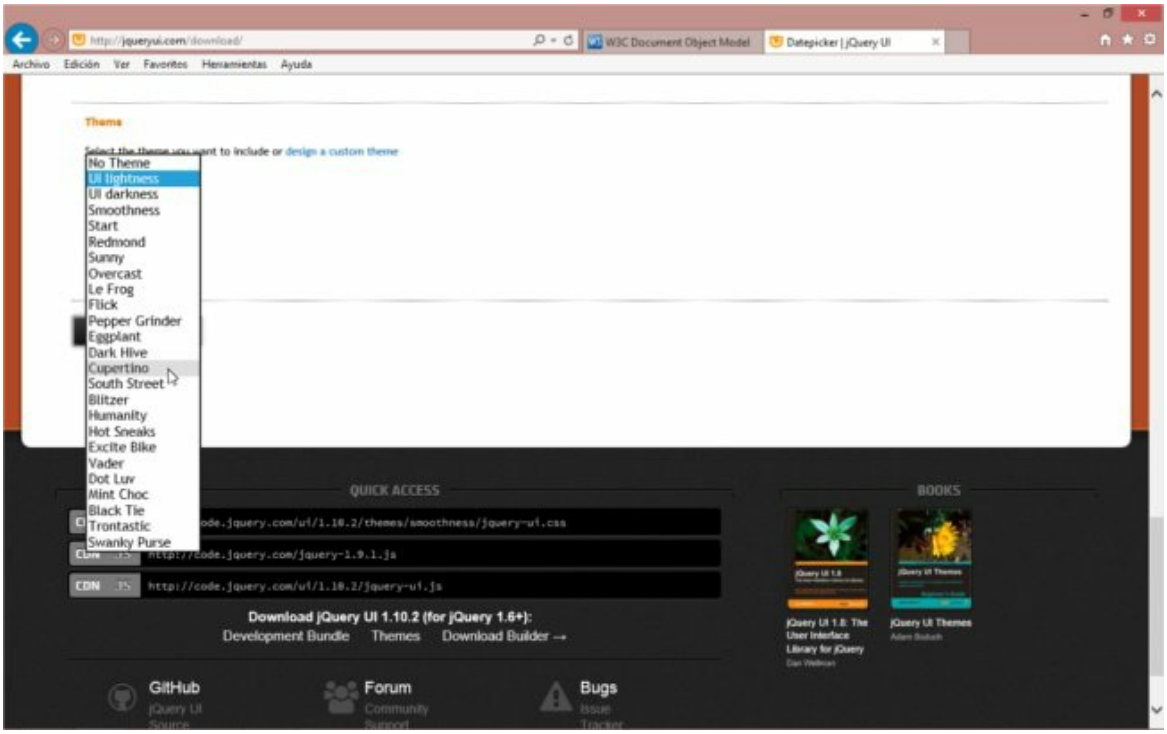
> **Nota:** Este tipo de redefiniciones es, esencialmente, lo que contienen los archivos de localización que hemos comentado en el apartado Opciones de localización idiomática

Jugando con los estilos

Hasta ahora hemos trabajado con referencias al *User Interface* alojado en el foro de *jQuery;* de esa manera disponemos de un modo muy cómodo y relativamente eficiente de crear los Datepicker, pero no de personalizar sus estilos, dado que se trabaja con uno predeterminado –una CSS referenciada concreta, dada por el administrador del *jQuery User Interface*-.

Sin embargo, el propio *jQuery User Interface* dispone de numerosos estilos prediseñados y listos para ser utilizados a demanda; pero para ello deberemos descargarlos y trabajar con ellos y referenciarlos en local –en el propio servidor-.

Podremos descargar el *jQuery User Interface* completo, que incluye *Datepicker*, la dirección URL es: jqueryui.com/download



En ese proceso de descarga, en la parte inferior nos encontraremos con el apartado "Theme", desplegando la lista podemos seleccionar el estilo del Datepicker que se descargará. En realidad, según lo que hayamos seleccionado previamente, es el estilo –el tema- de los componentes del *jQuery User Interface* que descargaremos.

De ahí es de donde se extraen los correspondientes CSS y directorios images particulares.

> **Nota:** En nuestro ejemplo hemos renombrado el archivo CSS descargado con el fin de poder trabajar con diferentes estilos, añadiéndole el nombre de estilo y cargándolo en local –véase la referencia correspondiente en el listado-. Trabajando en local, también deberemos copiar en cada caso el correspondiente directorio "images" y si queremos poder variar los estilos, darle un nombre particular para cada uno de los estilos –por ejemplo images_cupertino- y cambiar las referencias en el archivo CSS correspondiente.

######### Listado 5: Un calendario basado en un *Datepicker* en el que se ha utilizado el estilo *Cupertino*

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Datepicker - Estilo Cupertino</title>
<link rel="stylesheet" href="jquery-ui-1.10.2.cupertino.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script src="jquery.ui.datepicker-es.js"></script>

<script>
$(function () {
$.datepicker.setDefaults($.datepicker.regional["es"]);
$("#datepicker").datepicker({
firstDay: 1
});
});
</script>
</head>

<body>
Fecha:
<div id="datepicker"></div>
</body>
</html>
```

Calendario con un estilo *Cupertino*

Calendario con un estilo *Sunny*

Para cambiar todo el estilo por otro predefinido en el foro de *User Interface de jQuery* bastará descargar nuevamente el código y cambiar la hoja de estilo (y el directorio images) por la nueva; véase la segunda de las anteriores figuras, en la que se ha incluido la correspondiente al estilo *Sunny* en vez de la del estilo *Cupertino*.

Si lo que deseamos es retocar parcialmente un estilo predefinido, acudimos a modificar los estilos de la hoja de estilos –archivo CSS- descargado, bien con otro archivo CSS complementario y referenciado posteriormente en el archivo HTML –para respetar el orden de prelación de la cascada de estilos-, bien mediante estilos inline en el propio documento HTML.

Fijémonos en el Listado 5; allí programamos un calendario en el que alteraremos el tamaño de la fuente y su estilo, por ello estableceremos un apartado *style* que sobrecargará los estilos equivalentes de la hoja de estilos del *User Interface de jQuery* –archivo CSS referenciado al comienzo-.

El aspecto final se observa en la figura adjunta, compárese el resultado con el calendario original de estilo *Cupertino*.

######### Listado 6: Un calendario modificado, basado en un *Datepicker* en el que se ha utilizado el estilo *Cupertino* y algunos estilos personalizados

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
```

```
<title>jQuery UI Datepicker - Estilo Cupertino</title>
<link rel="stylesheet" href="jquery-ui-1.10.2.cupertino.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script src="jquery.ui.datepicker-es.js"></script>

<script>
$(function () {
$.datepicker.setDefaults($.datepicker.regional["es"]);
$("#datepicker").datepicker({
firstDay: 1
});
});
</script>

<style>
.ui-widget {
font-size: 1.5em;
font-style: italic;
}
</style>
</head>

<body>
Fecha:
<div id="datepicker"></div>
</body>
</html>
```

Calendario con un estilo *Cupertino* modificado con CSS

Naturalmente, el retoque de estilos requiere un conocimiento importante del archivo fuente del *jQuery User Interface*, con una buena dosis de paciencia y ensayo y error, pero que en ocasiones merecerá la pena para ir un paso más allá en nuestros trabajos.

Selección de una fecha

Hasta el momento hemos trabajado con la presentación del *Datepicker*, nos ocuparemos ahora de cómo extraer la información de la fecha seleccionada al hacer clic sobre una en concreto.

Fijémonos en el Listado 7:

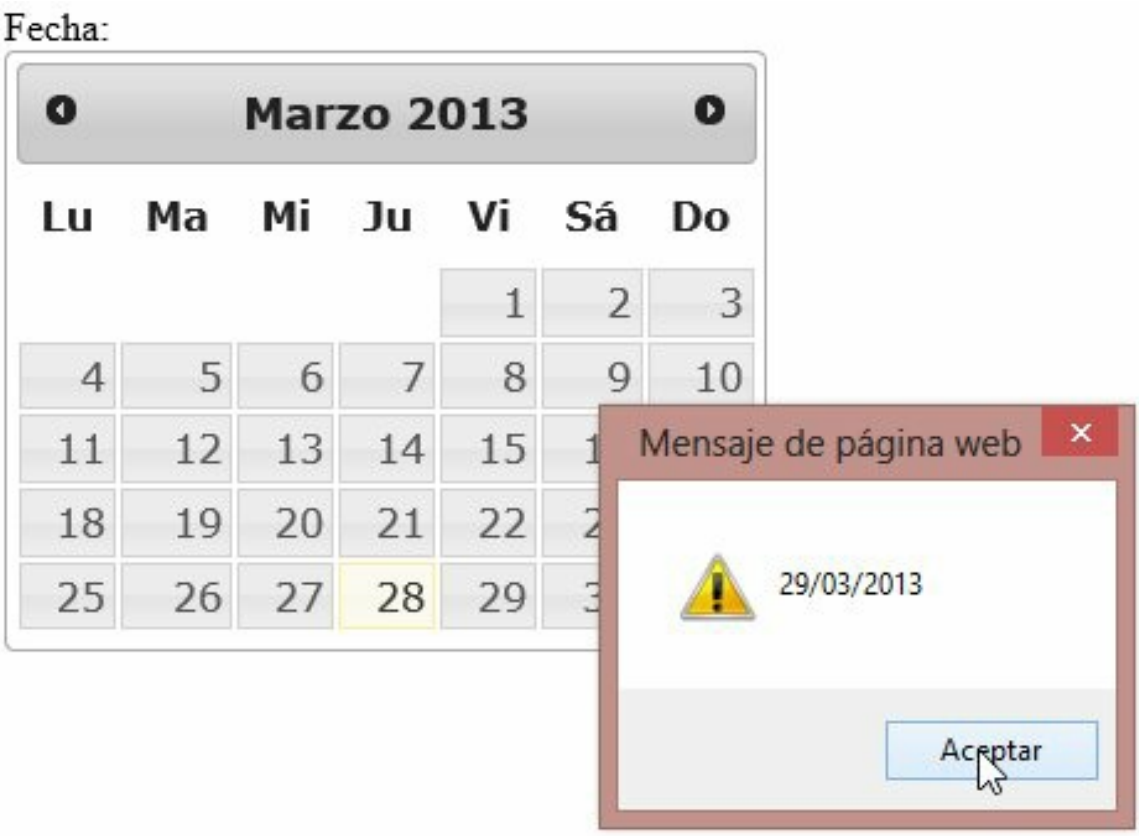
- Lo fundamental es la presencia del método de respuesta *onSelect*, que se activa cada vez que se hace clic sobre una fecha del *Datepicker*
- Se captura el evento y devuelve la fecha seleccionada, de manera que podremos acceder a ella de forma sencilla, como por ejemplo:

```
onSelect: function (date) {
alert(date)
}
```

- Nótese que no es preciso realizar ninguna tarea de conversión de datos, ya se nos devuelve la cadena formateada, conforme a las normas establecidas en las opciones de localización idiomática establecidas –de haberlas-

######### Listado 7: Extracción de información de la fecha seleccionada de un *Datepicker*. Uso de *onSelect*

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Datepicker - onSelect</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script src="jquery.ui.datepicker-es.js"></script>

<script>
$(function () {
$.datepicker.setDefaults($.datepicker.regional["es"]);
$("#datepicker").datepicker({
firstDay: 1,
onSelect: function (date) {
alert(date)
},
});
});
</script>
</head>

<body>
Fecha:
<div id="datepicker"></div>
</body>
</html>
```



Asociación a una entrada de textos

Una de las aplicaciones más directas de un Datepicker será su asociación a una entrada de textos –input type=”text”-, normalmente en el interior de un formulario.

En el Listado 8 recogemos esto mismo en su forma más simple. Bastará hacer clic sobre la entrada de textos y se desplegará el calendario y al seleccionar una fecha, esta quedará mostrada en el control de textos en el formato concordante con el correspondiente formato de fechas del estilo de localización idiomática –de haberse incluido tal, véase el apartado correspondiente-.

######### Listado 8: Asociación de un *Datepicker* a una entrada de textos

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Datepicker - Entrada de texto</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script src="jquery.ui.datepicker-es.js"></script>

<script>
$(function () {
$.datepicker.setDefaults($.datepicker.regional["es"]);
$("#datepicker").datepicker({
```

```
firstDay: 1
});
});
</script>
</head>

<body>
Fecha:
<input type="text" id="datepicker" />
</body>
</html>
```

Nótese que no hay nada que comentar acerca de la programación *JavaScript del Datepicker* y que el Listado 8 es idéntico al Listado 2, cuando simplemente mostrábamos el *Datepicker*. Solamente se diferencia en que hemos cambiado el elemento HTML al que va asociado, ahora un input type=”text”:

```
<body>
Fecha:
<input type="text" id="datepicker" />
</body>
```

Es el propio *Datepicker* el que se encargará de insertar la cadena de fecha en la entrada de texto, de forma automática y formateada al estilo de las opciones de localización idiomática establecidas –de haberlas-.

Para salir sin selección, hágase clic fuera del área comprendida por el *Datepicker*.

Cuando ya hay una fecha en el área de textos, el Datepicker se abrirá mostrando esa fecha, no

la actual.

Conclusiones

Hasta ahora hemos mostrado el uso más común del *jQuery User Interface Datepicker* y con ello lo que podría ser lo estándar de su utilidad, pero aún hay más, que presentaremos en un posterior artículo de uso más avanzado.

Esperamos que todo lo expuesto os haya servido de ayuda. Hasta nuestro próximo artículo, felices horas de programación.

Este artículo es obra de *Jaime Peña Tresancos*
Fue publicado por primera vez en *08/05/2013*
Disponible online en *http://desarrolloweb.com/articulos/jquery-ui-datepicker.html*

jQuery UI Datepicker. Manual de uso avanzado

Nos adentramos en las opciones avanzadas de jQuery User Interface Datepicker.

En nuestro anterior artículo jQuery UI Datepicker. Manual de uso simple comentábamos los aspectos básicos de la implementación y el uso del *jQuery User Interface Datepicker*, cuya última documentación completa –API de desarrollo- y ejemplos de uso, se pueden obtener en las direcciones URL: api.jqueryui.com/1.8/datepicker y jqueryui.com/datepicker

En el presente artículo hablaremos de:

- Ocultarlo, deshabilitarlo y reponerlo a sus estados iniciales
- Mostrar la semana del año, la barra de botones y los menús de meses y años
- Fijar la fecha de visión y la fecha por defecto
- Mostrarlo como una caja de diálogo
- Restringir un rango de fechas seleccionable
- Seleccionar rangos de fechas

Ocultarlo y volver a mostrarlo

En ocasiones nos interesará mostrar y ocultar un *Datepicker* bajo demanda. Para ello

disponemos de los métodos correspondientes *show y hide*. La sintaxis de llamada es muy simple:

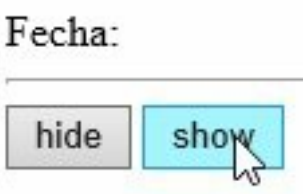
- $(referencia).datepicker().show()
- $(referencia).datepicker().hide()

Veamos el Listado 2, en el programamos un *Datepicker* asociado a un elemento *div* y dos botones de comando que llaman a las funciones *show y hide*.

######### Listado 1: Uso de los métodos de ocultación y mostrar un Datepicker

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Datepicker - Hide & Show</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script src="jquery.ui.datepicker-es.js"></script>

<script>
$(function () {
$.datepicker.setDefaults($.datepicker.regional["es"]);
$("#datepicker").datepicker({
firstDay: 1
});
});
$(function () {
$("#off").click(function () {
$("#datepicker").datepicker().hide();
});
});
$(function () {
$("#on").click(function () {
$("#datepicker").datepicker().show();
});
});
</script>
</head>

<body>
Fecha:
<div id="datepicker"></div>
<hr />
<input type="button" id="off" value="hide" />
<input type="button" id="on" value="show" />
</body>
</html>
```

Una vez pulsado el botón **"hide"** el calendario desaparecerá.



Para volver a reaparecer al pulsar sobre el botón **"show"**.

Deshabilitarlo y volver a habilitarlo

Otra opción que puede ser de interés es deshabilitar el *Datepicker*, de manera que deje de ser funcional, aunque siga estando presente.

El método a utilizar es *"disabled"* que, según su único parámetro, lo habilita o deshabilitará. La sintaxis es:

- $(referencia).datepicker("option", "disabled", true)
- $(referencia).datepicker("option", "disabled", false)

Veamos el Listado 2, en el programamos un *Datepicker* asociado a un elemento *div* y dos botones de comando que llaman a las funciones de deshabilitación y habilitación.

######### Listado 2: Uso de funciones de habilitar y deshabilitar un *Datepicker*

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Datepicker - Enable & Disable</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script src="jquery.ui.datepicker-es.js"></script>

<script>
```

```
$(function () {
$.datepicker.setDefaults($.datepicker.regional["es"]);
$("#datepicker").datepicker({
firstDay: 1
});
});
$(function () {
$("#disable").click(function () {
$("#datepicker").datepicker("option", "disabled", true);
});
});
$(function () {
$("#enable").click(function () {
$("#datepicker").datepicker("option", "disabled", false);
});
});
</script>
</head>

<body>
Fecha:
<div id="datepicker"></div>
<hr />
<input type="button" id="disable" value="disable" />
<input type="button" id="enable" value="enable" />
</body>
</html>
```

Cuando se pulse el botón **"disable"** el calendario seguirá presente, pero dejará de ser

funcional; no será posible navegar en él ni seleccionar una fecha.

Mostrar la semana del año

En ocasiones nos interesará conocer en qué semana del año nos encontramos y el *Datepicker* nos permite tener un acceso directo a dicho valor.

Bastará activar la opción *"showWeek"*, dándole un valor *"true"* en la inicialización, por ejemplo, con lo que se mostrará, a la izquierda de los días de la semana, el número de la semana del mes que representan.

En el Listado 3 se recoge un ejemplo de ello, hay que indicar a modo informativo que los días de la semana y las semanas en sí se computan conforme al estándar ISO 8601.

######### Listado 3: Uso de un calendario que muestra el número representativo de la semana del año

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Datepicker - Mostrar la semana del año ISO 8601</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script src="jquery.ui.datepicker-es.js"></script>

<script>
$(function () {
$.datepicker.setDefaults($.datepicker.regional["es"]);
$("#datepicker").datepicker({
showWeek: true,
firstDay: 1
});
});
</script>
</head>

<body>
<p>
Fecha:
<input type="text" id="datepicker" />
</p>
</body>
</html>
```

Mostrar la barra de botones

Aunque muy discreta, *Datepicker* dispone de una botonera para posicionarnos en la fecha actual y para cerrarse, sin más; como vemos dos funciones bastante útiles que seguramente nos interesará activar.

Para ello bastará incluir la opción *"showButtonPanel"* e igualarla a *"true"*, por ejemplo durante la inicialización, como se muestra en el Listado 4.

Recuérdese que para cerrarlo bastaría con hacer clic sobre cualquier área fuera del *Datepicker*, pero para ir a la fecha actual deberíamos navegar manualmente por el calendario y es por ello por lo que realmente tiene valor la botonera.

######### Listado 4: Activación de la botonera predefinida del Datepicker

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Datepicker - Mostrar la barra de botones</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script src="jquery.ui.datepicker-es.js"></script>

<script>
$(function () {
$.datepicker.setDefaults($.datepicker.regional["es"]);
$("#datepicker").datepicker({
showButtonPanel: true
```

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

<body>
<p>
Fecha:
<input type="text" id="datepicker" />
</p>
</body>
</html>
```

Fijar una fecha de visión

Por defecto, el *Datepicker* se muestra inicialmente centrado en la fecha actual, pero puede interesarnos proponer fechas iniciales diferentes.

Disponemos del método *"setDate"* y de la opción *"defaultDate"* que, respectivamente, nos permitirán fijar fechas diferentes o desplazamientos relativos, expresados en días, meses y años, respecto de la fecha actual, tanto hacia el futuro como hacia el pasado y fijar la fecha que se mostrará por defecto al iniciar el *Datepicker*.

Respecto a *"setDate"* su función es fijar la fecha a mostrar en el *Datepicker*, de manera que se muestre como la predeterminada. Su uso se ilustra en el Listado 5, en el cual se ha asociado a un botón de comando que, al ser pulsado, nos desplazará la fecha del *Datepicker* tres meses

más allá de la fecha actual.

Como mencionábamos, se puede pasar una fecha fija o un desplazamiento respecto a la fecha actual, expresada en días, meses y/o años, tanto retrospectivamente como hacia el futuro. La sintaxis es muy sencilla:

- Una fecha fija en el formato dd/mm/aaaa. Por ejemplo: 30/08/2014
- Un desplazamiento relativo en el que se muestra un número seguido de y para años, de m para meses y d para días; si se antepone el signo + serán fechas futuras y si se antepone el signo – serán fechas en el pasado. Por ejemplo: +3y
- Pueden acumularse desplazamientos relativos, separados por comas. Por ejemplo: +3m, -20d

######### Listado 5: Uso de la función *setDate* para desplazar la fecha de presentación del *Datepicker*

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Datepicker - setDate</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script src="jquery.ui.datepicker-es.js"></script>

<script>
$(function () {
$.datepicker.setDefaults($.datepicker.regional["es"]);
$("#datepicker").datepicker({
firstDay: 1
});
});
$(function () {
$("#setDate").click(function () {
$("#datepicker").datepicker("setDate", "+3m");
});
});
</script>
</head>

<body>
Fecha:
<div id="datepicker"></div>
<hr />
<input type="button" id="setDate" value="setDate" />
</body>
</html>
```

Al pulsar el botón **setDate**, el calendario actualizará la fecha que muestra a la fijada en la función, en nuestro caso tres meses más tarde que la fecha actual.

Por otra parte, la opción *"defaultDate"* fijará la fecha por defecto que será mostrada por el *Datepicker* como predeterminada al abrirse.

Como antes, se puede pasar una fecha fija o un desplazamiento respecto a la fecha actual, expresada en días, meses y/o años, tanto retrospectivamente como hacia el futuro.

En el Listado 6 se muestra un ejemplo de uso, en el que el *Datepicker* se iniciará con una fecha que es dos meses posterior a la fecha actual.

######### Listado 6: Uso de la opción defaultDate para fijar la fecha por defecto de un *Datepicker*

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Datepicker - defaultDate</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script src="jquery.ui.datepicker-es.js"></script>

<script>
$(function () {
$.datepicker.setDefaults($.datepicker.regional["es"]);
$("#datepicker").datepicker({
firstDay: 1,
defaultDate: "+2m"
});
});
</script>
</head>

<body>
Fecha:
<div id="datepicker"></div>
</body>
</html>
```

Mostrar el Datepicker como una caja de diálogo

Los *Datepicker* aparecen asociados a elementos en los documentos HTML y, como tales, se muestran pegados a tales elementos, como las entradas de texto que estamos empleando. Pero cabe la posibilidad de mostrarlos como cajas de diálogo independientes, flotantes en sobre el documento. Cierto es que con limitaciones, ya que no se puede fijar o modificar su posición.

Si por ejemplo, queremos una entrada de textos para insertar fechas y a la vez tener la posibilidad de seleccionarlas mediante un *Datepicker* que se muestre mediante la pulsación de un botón, los pasos a seguir en su programación serían:

- En la parte del documento HTML creamos dos elementos:
  - Una entrada de textos, que alojará la fecha seleccionada en el *Datepicker*
  - Un botón de comando que hará visible el *Datepicker* como un diálogo

- La función de creación del diálogo que contendrá el *Datepicker* debe ser similar a:

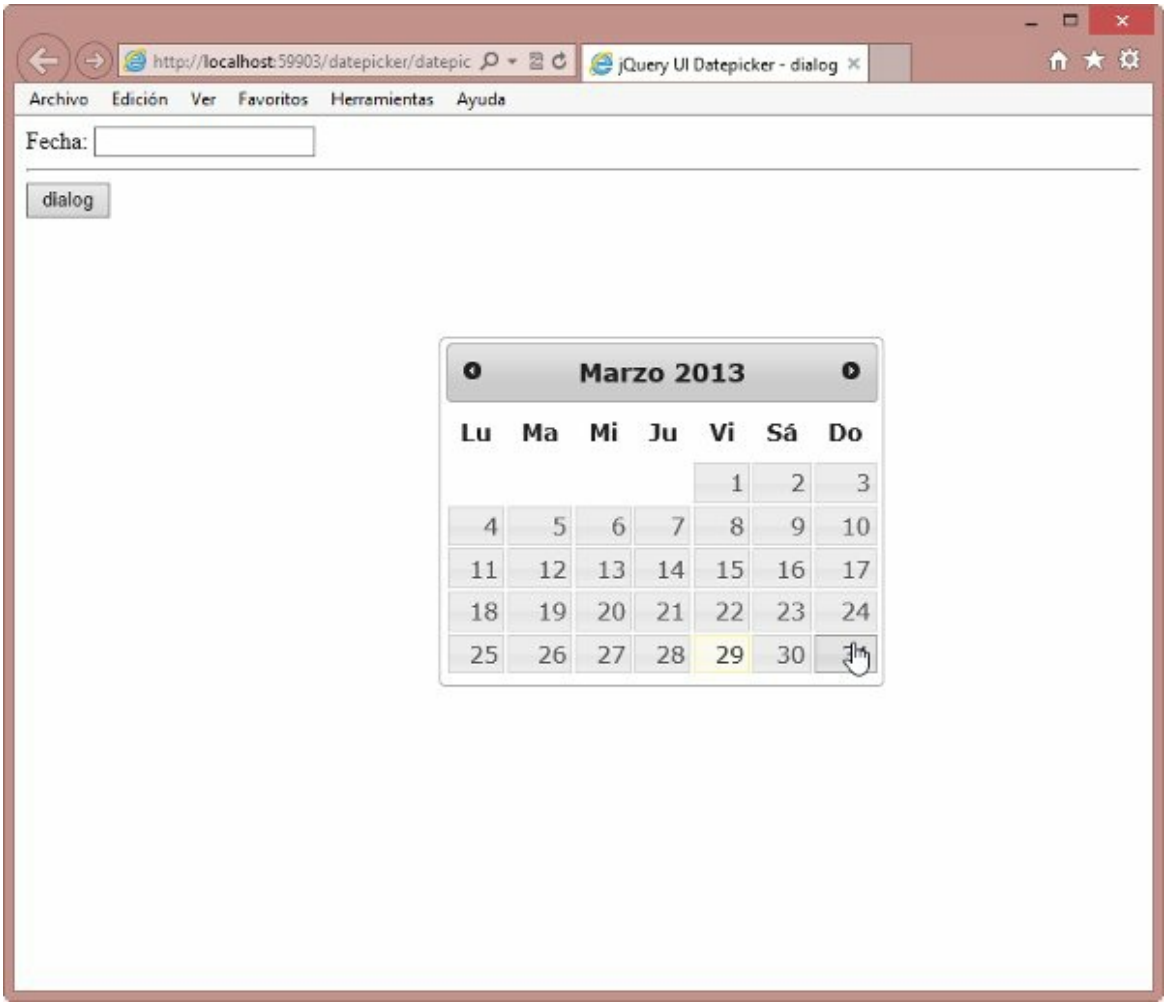
```
$(function () {
$("#dialog").click(function () {
$("#datepicker").datepicker("dialog", "", updateDate);
function updateDate(date) {
$("#datepicker").val(date);
}
});
});
```

- El método "dialog" llama a la función de respuesta personal, en nuestro caso *"updateDate"*
- En esa función devuelve como parámetro un objeto tipo *date*

- Se utiliza la función "val" para convertir el objeto date en una cadena de caracteres de formato fecha que se pasa al *Datepicker*

Listado 7: Uso de un Datepicker que se mostrará a demanda mediante como caja de diálogo

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Datepicker - dialog</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script src="jquery.ui.datepicker-es.js"></script>

<script>
$(function () {
$("#dialog").click(function () {
$("#datepicker").datepicker("dialog", "", updateDate);
function updateDate(date) {
$("#datepicker").val(date);
}
});
});
</script>
</head>

<body>
Fecha:
<input type="text" id="datepicker" />
<hr />
<input type="button" id="dialog" value="dialog" />
</body>
</html>
```



Mostrar menús de meses y años y la opción yearRange

Para una mejor navegación por el *Datepicker*, éste tiene la posibilidad de darnos accesos rápidos a los diferentes meses y años, utilizando las opciones *"changeMonth" y "changeYear"*.

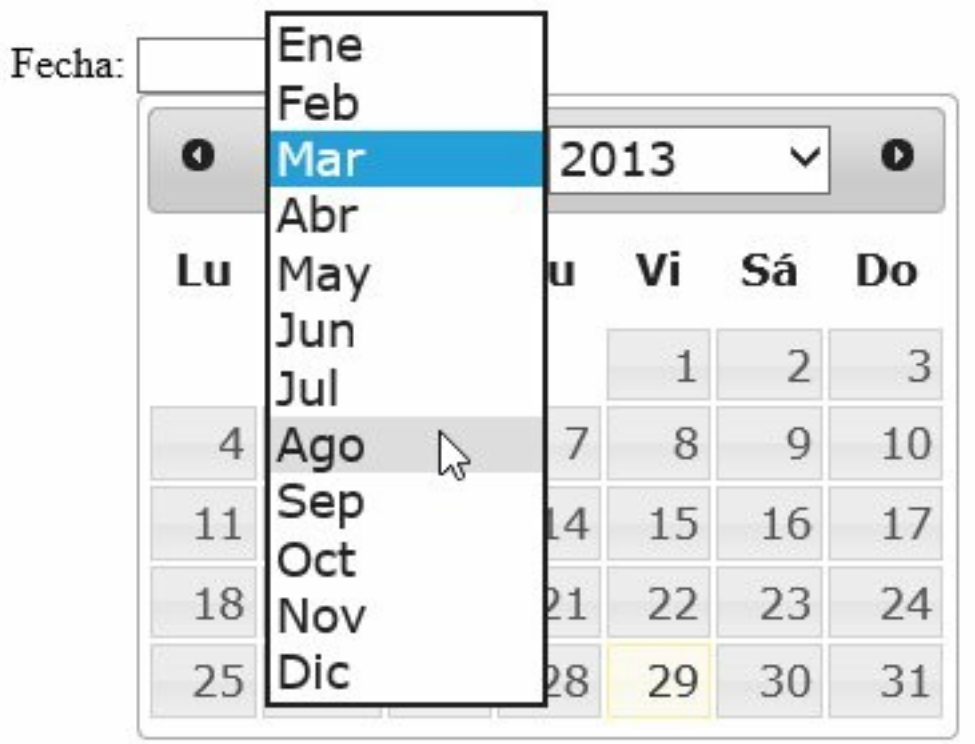
Aparecerán en la parte superior del *Datepicker* un par de listas desplegables con las entradas de selección de los meses del año y los años, como se observa en la figura adjunta.

En el Listado 8 se muestra un ejemplo de codificación de dichas opciones en la inicialización del *Datepicker*. Bastará poner sus valores a "true".

######### Listado 8: Opción de Datepicker para mostrar los menús de meses y años

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Datepicker - Mostrar menús de meses y años</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script src="jquery.ui.datepicker-es.js"></script>
```

```
<script>
$(function () {
$.datepicker.setDefaults($.datepicker.regional["es"]);
$("#datepicker").datepicker({
changeMonth: true,
changeYear: true
});
});
</script>
</head>

<body>
<p>
Fecha:
<input type="text" id="datepicker" />
</p>
</body>
</html>
```



Un problema con el código anterior es que sólo se nos mostrará un rango predeterminado de años, que puede no abarcar –o exceder- el que nos interese. Por ello, normalmente nos convendrá utilizar a la vez la opción *"yearRange"* que permite fijar rangos de años a ser seleccionables.

Así, en el Listado 9 hemos fijado un rango histórico que vaya de 1900 a 2013.

######### Listado 9: Opción de *Datepicker* para mostrar los menús de meses y años, con acotación del rango de años mediante la opción *"yearRange"*

```
<script>
$(function () {
$.datepicker.setDefaults($.datepicker.regional["es"]);
$("#datepicker").datepicker({
changeMonth: true,
changeYear: true,
yearRange: "1900:2013"
});
});
</script>
```

Restringir el rango de fechas seleccionable

Imaginemos que deseamos acotar las fechas de selección de reservas a un evento, podremos programar el *Datepicker* de tal forma que sólo permita un rango a nuestro criterio mediante las opciones *minDate y maxDate*.

Cómo hemos discutido anteriormente, los valores pasados pueden ser fechas fijas o desplazamientos relativos en días, meses y/o años, tanto retrospectivamente como hacia el futuro. Por ejemplo si fijamos una venta anticipada para una fecha fija puede interesar fijar fechas fijas de *minDate y maxDate;* si lo hacemos para venta de entradas al cine para un día cualquiera, puede interesarnos fijar una fecha relativa de *minDate* igual a -7d, una semana antes, y *maxDate* -1d, el día anterior.

En el Listado 10 se muestra cómo fijar un rango de fechas relativas a la actual, con un desplazamiento de 20 días hacia atrás y de 2 meses menos diez días hacia adelante.

En el Listado 11 se muestra un ejemplo de fijación de rango de fechas fijas, durante la inicialización del *Datepicker*.

######### Listado 10: Ejemplo de fijación de rango de fechas seleccionable en un *Datepicker*

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Datepicker - Restringir el rango de fechas</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script src="jquery.ui.datepicker-es.js"></script>

<script>
$(function () {
$.datepicker.setDefaults($.datepicker.regional["es"]);
$("#datepicker").datepicker({
minDate: "-20D",
maxDate: "+2M, -10D"
});
```

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

<body>
<p>
Fecha:
<input type="text" id="datepicker" />
</p>
</body>
</html>
```

######### Listado 11: Ejemplo de fijación de rango de fechas fijas

```
<script>
$(function () {
$.datepicker.setDefaults($.datepicker.regional["es"]);
$("#datepicker").datepicker({
minDate: "15/04/2013",
maxDate: "15/05/2013"
});
});
</script>
```

Seleccionar un rango de fechas

Otra aplicación muy usual de un calendario es la selección de rangos de fechas, por ejemplo para un viaje.

Para ello utilizaremos dos entradas de textos con dos *Datepicker enlazados*, de manera que la *"minDate"* del segundo sea la fecha seleccionada en el primero –de haberla- y la *maxDate* del primero la seleccionada en el segundo –de haberla-.

Todas esas asignaciones se codifican en las respectivas funciones *onClose* de los *Datepicker*, tal como se recogen en el Listado 12.

######### Listado 12: Modo de seleccionar un rango de fechas mediante dos *Datepicker* asociados

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Datepicker - Seleccionar un rango de fechas</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script src="jquery.ui.datepicker-es.js"></script>

<script>
$(function () {
$("#from").datepicker({
onClose: function (selectedDate) {
$("#to").datepicker("option", "minDate", selectedDate);
}
});
$("#to").datepicker({
onClose: function (selectedDate) {
$("#from").datepicker("option", "maxDate", selectedDate);
}
});
});
</script>

</head>

<body>
<p>
Desde:
<input type="text" id="from" name="from" />
Hasta:
<input type="text" id="to" name="to" />
</p>
</body>
</html>
```

Conclusiones

Con el presente artículo damos por finalizado el estudio del componente *Datepicker del jQuery User Interface*. Hemos repasado la práctica totalidad de sus funciones y opciones y hemos visto ejemplos de cómo utilizarlas en nuestras aplicaciones.

Esperamos que lo aquí expuesto les haya servido de ayuda en su trabajo y no dejen de preguntar cualquier duda que les pudiese surgir. Hasta la próxima, tengan unas provechosas sesiones de programación.

Este artículo es obra de *Jaime Peña Tresancos*
Fue publicado por primera vez en *16/05/2013*
Disponible online en *http://desarrolloweb.com/articulos/jquery-ui-datepicker-avanzado.html*

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.

Este artículo es obra de *Miguel Angel Alvarez*

Fue publicado por primera vez en *19/05/2010*
Disponible online en *http://desarrolloweb.com/articulos/componente-dialog-jquery-ui.html*

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.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *19/05/2010*
Disponible online en *http://desarrolloweb.com/articulos/caja-modal-jquery-ui.html*

Plugin Button de jQuery UI

Un plugin jQuery que nos permite implementar botones de funcionalidades mejoradas gracias a las librerías jQuery UI de Javascript y CSS.

En este artículo veremos las generalidades sobre un interesante plugin para crear botones con configuraciones avanzadas, así como aspecto y posibilidades mejoradas, con respecto a los botones normales utilizados en formularios HTML. Se trata del plugin Button de jQuery UI, del que veremos varios ejemplos para aprender a utilizarlo.

El plugin Button de jQuery incluye un método para convertir cualquier elemento HTML, como un enlace o una división, en un botón mejorado. Por este proceso el elemento se estilizará como si fuera un botón, aplicando esquinas redondeadas y otros estilos del tema de jQuery UI que hayamos seleccionado en la descarga de las librerías, y además se activará como elemento "caliente", que cambiará al pasar el ratón por encima, para indicar al usuario que se puede hacer clic sobre él.

Por medio de diversas configuraciones podemos conseguir además que el botón incluya algunos iconos y que tenga algunos comportamientos específicos de otros elementos, como botones de radio o checkboxes. Se trata por tanto de un componente de gran versatilidad que se puede utilizar para diversos propósitos en aplicaciones web enriquecidas del lado del cliente.

> **Nota:** entendemos que al menos ya conoces algo las librerías jQuery UI, que contienen diversos plugins de jQuery para crear interfaces de usuario y que ya comenzamos a explicar en el artículo Primeros pasos con jQuery UI.

Crear un botón jQuery UI

Para crear un botón, podemos utilizar en principio cualquier tipo de elemento HTML, como puede ser un enlace, una división o un span.

```
<a href="#">Mi botón</a>
<a href="#">Otro botón</a>
```

Ahora eso son enlaces absolutamente normales y para convertirlos en botones basta con invocar al plugin button, de la siguiente manera:

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

> **Nota:** para que el método button() funcione se tiene que haber cargado previamente jQuery y las librerías jQuery UI, tal como se indicó en el artículo Empezar a usar jQuery UI en una página web.

Con la anterior instrucción estamos convirtiendo en botones todos los enlaces (etiquetas A) de la página. Como se puede ver, se ha invocado al plugin button, por medio de una llamada al método button(), sobre un conjunto de elementos que obtenemos por medio del selector de jQuery "a" (todos los enlaces).

Aplicar un evento clic al botón

Una de las cosas que con toda certeza querremos hacer en algún momento con los botones es asignarle eventos click o similares. Para ello el propio plugin del botón no implementa ningún tipo de evento especial, pero podemos asignarle cualquier evento estándar Javascript, como a cualquier otro elemento de la página. Para ello utilizamos el mismo procedimiento que ya conocemos para asignar eventos por medio de jQuery.

Por ejemplo, tememos estos dos elementos HTML, que comparten una misma clase CSS:

```
<div class="boton">Este botón lo construyo con un DIV</div>
<br><br>
<div class="boton">Otro DIV</div>
```

Ahora podemos hacer que esos elementos DIV se conviertan en botones, con una llamada al plugin jQuery.

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

Como se puede comprobar, a través del selector ".boton" accedemos a todos los elementos de la página que tengan la clase (class de CSS) "boton" y luego con el método button() los convertimos en botones. A continuación, podríamos asignarle un evento clic, como a cualquier elemento de la página, por medio de jQuery y el método click():

```
$(".boton").click(function(){
   alert("Me has pulsado bien!!");
});
```

Ahora, cuando se pulse cualquiera de esos dos elementos se mostrará un mensaje en una caja de alerta.

Los botones vistos hasta el momento se puede encontrar en funcionamiento en una página aparte.

Hasta ahora hemos visto cómo crear un botón básico, pero el plugin button tiene muchas opciones de configuración avanzadas que conoceremos el próximo artículo.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *24/08/2010*
Disponible online en *http://desarrolloweb.com/articulos/plugin-button-jquery-ui.html*

Configuración avanzada de button jQuery UI

Algunas configuraciones posibles para mejorar las funcionalidades del plugin Button de las librerías jQuery UI.

En el pasado artículo estuvimos introduciendo el plugin button de las librerías jQuery UI, que implementa la posibilidad de convertir en botones diversos elementos de una página web. Ahora continuaremos mostrando algunas configuraciones avanzadas de los botones que nos permitirán aumentar sus funcionalidades y la utilidad de los mismos en las páginas web.

Comenzaremos por la creación de botones indicando las opciones de configuración por medio de un objeto de options que enviamos al método button() para invocar al plugin. Luego veremos cómo se puede usar algún método específico de este plugin para cambiar sobre la marcha la configuración de un botón.

Options en botones de jQuery UI

Cuando creamos un botón podemos indicar un objeto de opciones de configuración para implementarlo según nuestras necesidades específicas. Las opciones de configuración básicas permiten definir cosas como el texto que tendrá escrito, si el botón estará activo o no, si se verá el texto o no, o los iconos que pondremos en el botón.

> **Nota:** Los objetos de configuración ya los conocemos y los hemos usado a lo largo del Taller de jQuery para configurar plugins en el momento de invocarlos. Permiten definir las opciones que deseamos en el plugin y tomar las opciones por defecto de aquellas configuraciones que no hayamos indicado.

Lo más cómodo es ver este paso con un ejemplo. Partimos de este elemento HTML que luego convertiremos en botón.

```
<span class="botonoptions">Botón configurado</span>
```

Ahora podemos ver el código Javascript que nos permitirá convertirlo en botón, enviando diversas opciones de configuración en un objeto "options" al método button() con el que se invoca el plugin:

```
$(".botonoptions").button({
   icons: {
```

```
      primary: 'ui-icon ui-icon-refresh'
   },
   label: "Actualizar",
   disabled: true
})
```

Esto construirá un botón con un icono primario (luego explicaremos esto), el texto "Actualizar" como etiqueta y que estará desactivado. Recomiendo ver la lista completa de opciones de configuración en la propia documentación de jQuery UI para el plugin button.

Iconos en botones jQuery UI

Como hemos visto, los botones en jQuery UI tienen la posibilidad de incluir iconos, lo que resulta un detalle bastante interesante. En jQuery UI tenemos definidos hasta 176 iconos de 16x16 en una especie de framework CSS incluido en las propias librerías. Todos esos iconos los podemos utilizar para decorar botones u otras interfaces de la página y para ello tenemos que utilizar tan solo unos nombres de clases.

La lista de botones disponibles, junto con el nombre de la clase de cada uno, se puede ver en el ThemeRoller de jQuery UI. En la parte de abajo podremos encontrar todos los iconos disponibles y si nos situamos con el ratón sobre uno de ellos, aparecerá el nombre de la clase de ese icono.

En los botones de jQuery UI podemos utilizar dos iconos, uno que aparecerá antes y otro después del texto del botón (si es que hemos colocado texto en el botón). Luego, para definir los dos iconos que se van a utilizar, tenemos que indicarlos en la llamada al método que invoca el plugin, por medio de un objeto "icons" al que le indicamos dos propiedades "primary" (para el icono que aparecerá antes del texto) y "secondary" (para el icono que habrá después del texto).

Tenemos un nuevo ejemplo para crear un botón con dos iconos. De momento veamos el HTML utilizado para generarlo.

```
<div id="botoniconos">Botón con iconos</div>
```

Ahora el jQuery para invocar al plugin Button y convertir el elemento en un botón con dos iconos.

```
$("#botoniconos").button({
   icons: {
      primary: 'ui-icon ui-icon-scissors',
      secondary: 'ui-icon ui-icon-mail-closed'
   }
});
```

Métodos para realizar acciones con el botón

Como otros plugins de jQuery UI, el plugin de button tiene algunos métodos para realizar acciones especiales con el botón. En la documentación del plugin podremos ver la lista de métodos que contiene y nosotros vamos a hacer un ejemplo aquí con uno de ellos.

El método que vamos a ver es el que nos permite cambiar en línea las propiedades un botón, es decir, las opciones de configuración de ese botón que se indicaron por el objeto de options.

Para invocar ese método tenemos que llamar al método del plugin enviando como primer parámetro el string "option" y como segundo parámetro un nuevo objeto de options.

Veamos el siguiente código que implementa un evento clic sobre el botón con iconos del ejemplo anterior.

```
$("#botoniconos").click(function(){
   $(this).button("option", {
      icons: {
         primary: 'ui-icon ui-icon-refresh'
      }
   });
})
```

Como se puede ver, cuando se haga clic sobre el botón se invocará al método button() sobre el botón en el que se hizo clic. Pero en este caso no es una simple llamada al plugin para generar un botón, pues le estamos pasando parámetros distintos al método. En el primer parámetro de la llamada a button() señalamos que se desea cambiar las "option" y en el segundo indicamos un nuevo juego de opciones en un objeto. En este caso en concreto simplemente se define un nuevo icono como icono primario, con lo que cambiarán los iconos que se muestran en el botón.

Podemos ver en una página aparte los botones que hemos realizado en los ejemplos de este artículo.

Para una práctica más avanzada con botones leer el taller Array de plugins Button de jQuery UI.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *24/08/2010*
Disponible online en *http://desarrolloweb.com/articulos/configuaracion-avanzada-button-jquery-ui.html*

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.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *25/08/2010*
Disponible online en *http://desarrolloweb.com/articulos/array-plugins-button-jquery-ui.html*

Plugin autocomplete de jQuery UI

Como usar el plugin jQuery UI autocomplete, para crear un campo de texto que ofrece sugerencias para autocompletar la información escrita, según se escribe.

En el presente artículo vamos a mostrar un modo de crear un campo de texto con función de autocompletar, una utilidad dinámica muy de agradecer, que ayudará a los usuarios a teclear menos y escribir información más precisa. Sin duda habréis visto en innumerables ocasiones esa utilidad, en diversos sitios. Por ejemplo, Google Maps tiene un campo autocompletar que se activa para sugerirnos direcciones del mundo a medida que vamos escribiendo cualquier cosa.

A continuación mostraremos cómo utilizar un script ya creado para hacer dicho campo de autocompletar. Es decir, no explicaremos cómo hacerlo por nosotros mismos, sino cómo utilizar un desarrollo ya listo, que nos ahorrará mucho trabajo. Utilizaremos el plugin autocomplete de las librerías jQuery UI, que la verdad es un componente bastante potente que podremos configurar perfectamente para adaptarlo a cualquier necesidad.

Nota: Si deseas aprender jQuery UI te recomendamos que estudies previamente un poco de jQuery y que leas la serie de artículos sobre los primeros pasos con jQuery UI.

Ahora veremos como en una corta serie de pasos elementales, podremos tener nuestro propio componente de autocompletado.

Paso 1: incluir las librerías jQuery y jQuery UI

Como ya se comentó en el artículo Pasos para utilizar jQuery UI, parte 2, primero tenemos que incluir los scripts de jQuery y jQuery UI, así como la hoja de estilos de el tema de personalización del componente que estemos utilizando.

Todos los scripts y la hoja de estilos los obtenemos cuando descargamos nuestro paquete de jQuery UI. El código para incluirlo todo será algo como esto.

```
<link type="text/css" href="css/ui-darkness/jquery-ui-1.8.6.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-1.8.6.custom.min.js"></script>
```

> **Nota:** Lógicamente, tendrás que comprobar que las rutas a los archivos están correctas, tal como tengas la estructura de directorios en tu web. Además, muy probablemente habrás descargado jQuery UI con otro tema CSS, por lo que la ruta al archivo CSS seguramente tendrás que cambiarla.

Paso 2: colocar un campo de texto normal

En algún lugar del cuerpo de tu página tendrás que colocar una etiqueta INPUT para crear un campo de texto, que luego convertiremos en un componente autocompletar.

```
<input type="text" size="20" id="autoc1">
```

El identificador (atributo ID) será el que tú desees. Simplemente acuérdate de él, porque luego tendrás que utilizarlo en tu script.

Paso 3: generar un array con las opciones de autocompletado

Ahora vamos a crear un array Javascript con diversos valores, que utilizaremos en el campo de autocompletar.

Este array se utiliza cuando queremos que las opciones para autocompletado del campo estén en "local", es decir, escritas en la propia página que tiene el script. Sin embargo, esta es sólo una opción de configuración y la hemos elegido ahora por ser la más simple. Más adelante explicaremos cómo obtener esas opciones de forma remota.

Vamos a crear en este caso un array con las provincias de España, que tendría esta forma.

```
var arrayValoresAutocompletar = [
   "Álava",
   "Albacete",
   "Alicante",
   "..."
];
```

Paso 4: invocar al plugin autocomplete

En este paso hacemos la tarea de convertir el campo de texto normal en un campo de tipo autocomplete, que sugiera opciones según se escribe texto en él. Veremos lo sencillo que es, gracias a jQuery UI.

```
$("#autoc1").autocomplete({
   source: arrayValoresAutocompletar
})
```

Con esta sentencia invocamos al plugin sobre el campo de texto, al que hemos accedido por su identificador. Además, debemos indicar como opción en el plugin el lugar desde donde se deben obtener las distintas opciones para autocompletar.

Ejemplo puesto en marcha

Eso es todo! Quizás te sorprenda, como a mí, lo fácil que ha sido poner este plugin en marcha.

Si lo deseas, puedes ver el campo autocompletar que hemos creado. Para ver la funcionalidad en marcha debes escribir cualquier letra en el campo o un conjunto de letras como "alm" y te mostrará diferentes sugerencias de provincias de España para que elijas la que desees.

Todo el código completo para el campo autocompletar

Ahora puedes ver el código completo de esta página realizada para poner en marcha el plugin autocomplete.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"
>
<html lang="en">
<head>
<title>Probando autocomplete</title>
   <link type="text/css" href="css/ui-darkness/jquery-ui-1.8.6.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-1.8.6.custom.min.js"></script>
   <script type="text/javascript">
      $(document).ready(function(){
         var arrayValoresAutocompletar = [
            "Álava",
            "Albacete",
            "Alicante",
            "Almería",
            "Asturias",
            "Avila",
            "Badajoz",
            "Barcelona",
            "Burgos",
            "Cáceres",
            "Cádiz",
            "Cantabria",
            "Castellón",
```

```
            "Ceuta",
            "Ciudad Real",
            "Córdoba",
            "Cuenca",
            "Gerona",
            "Granada",
            "Guadalajara",
            "Guipúzcoa",
            "Huelva",
            "Huesca",
            "Islas Baleares",
            "Jaén",
            "La Coruña",
            "La Rioja",
            "Las Palmas",
            "León",
            "Lérida",
            "Lugo",
            "Madrid",
            "Málaga",
            "Melilla",
            "Murcia",
            "Navarra",
            "Orense",
            "Palencia",
            "Pontevedra",
            "Salamanca",
            "Segovia",
            "Sevilla",
            "Soria",
            "Tarragona",
            "Tenerife",
            "Teruel",
            "Toledo",
            "Valencia",
            "Valladolid",
            "Vizcaya",
            "Zamora",
            "Zaragoza"
        ];
        $("#autoc1").autocomplete({
            source: arrayValoresAutocompletar
        })
    })
  </script>
</head>
<body>
<h1>Probando el plugin autocomplete de jQuery UI</h1>

  <form>
    Escribe una provincia de España: <input type="text" size="20" id="autoc1">
    <br>
    * Pon al menos una letra para que salgan opciones.
    <br>
    * Puedes escribir algo como "ma" para que salgan sugerencias como "Madrid", "Salamanca", etc.
```

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

Este modo de trabajar con las opciones del autocomplete por medio de un array enviado al propio plugin es interesante cuando las posibilidades del campo son limitadas, pero no resulta optimo cuando las opciones que podrían colocarse son muchas, por ejemplo más de 100. En el artículo siguiente mostraremos cómo realizar este campo de autocompletado de manera que las opciones sugeridas se carguen desde un script remoto.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *08/12/2010*
Disponible online en *http://desarrolloweb.com/articulos/plugin-autocomplete-jquery-ui.html*

Campo de autocompletado en jQuery UI con carga remota de opciones

Cómo hacer un campo de autocompletar que permita conectarse con una URL para obtener las opciones sugeridas a través de JSON.

En un artículo anterior estuvimos analizando y haciendo una primera prueba con el plugin Autocomplete de jQuery UI, en la que aprendimos a utilizar ese plugin para sugerir valores de autocompletado que estaban en un array Javascript.

Obtener las sugerencias de autocomplete desde un array definido en el cliente no tiene nada de malo y es una opción perfectamente válida, pero no es eficiente en el caso que puedan existir muchas sugerencias de autocompletado.

En el caso del artículo anterior teníamos un campo donde debíamos escribir las provincias de España. Como las posibilidades eran sólo unas cuantas decenas, obtener las posibilidades de autocompletar desde un array Javascript es perfectamente válido. Pero pensamos en una aplicación como Google Maps, donde se escriben sugerencias para autocompletar direcciones o lugares del mundo... podríamos tener miles de poblaciones en solo país y millones de direcciones del mundo entero! Todo eso sería imposible de guardar en un array en el cliente, por lo que se debe hacer algo diferente.

La solución es tan sencilla como invocar un script remoto que nos devuelva las sugerencias para autocompletar el campo. A ese script se le pasará lo que el usuario haya escrito en el campo de autocompletado y se encargará de buscar palabras o términos que tengan similitud con aquello que se escribió. Las sugerencias para autocompletar el campo se deben devolver en formato JSON.

El ejemplo de campo autocompletar que hemos realizado para probar esta posibilidad se puede ver en una página aparte.

Configurar el campo autocompletar para la búsqueda remota de sugerencias

Para indicar a nuestro campo autocompetar que busque las sugerencias a través de un script remoto tenemos que indicar en el parámetro "source" de configuración del plugin la URL donde está el script que nos debe devolver las posibilidades de autocompletado.

```
$("#autoc2").autocomplete({
   minLength: 3,
   source: "buscar-autocompletar.php"
})
```

En este caso se utiliza un script PHP para buscar las sugerencias, que está en el archivo "buscar-autocompletar.php". Como no se especifica directorio, se supone que este archivo PHP está en la misma carpeta que la página desde donde se invoca.

Además, como se puede ver en el código anterior, se ha indicado otro parámetro de configuración del plugin llamado "minLength", que sirve para definir la longitud mínima que debe tener el texto escrito en el campo para que la función de autocompletar se active y se invoque al script para obtener sugerencias. Esto se hace por razones de rendimiento, para que el campo no se active hasta que el usuario haya escrito un número x de caracteres, de modo que no nos salgan demasiadas sugerencias ni se realicen accesos innecesarios al script que las genera.

Cuando se invoca al script para buscar las opciones para el campo autocomplete, jQuery envía automáticamente por el método GET aquello que se haya escrito en el campo de texto. Por ejemplo, si en el campo el usuario escribió las letras "pro", se invocaría al script PHP de la siguiente manera:

```
http://www.dominio.com/buscar-autocompletar3.php?term=pro
```

Formato JSON para las sugerencias de autocompletar

Como hemos dicho, el script de búsqueda debe devolver las opciones para el campo autocomplete en notación JSON. Ahora podemos ver cómo es el formato JSON que espera el campo de autocompletado para mostrar las sugerencias según se escribe.

```
[
   {"value":"valor 1","label":"Elemento 1"},
   {"value":"valor 2","label":"Elemento 2"},
   {"value":"otro valor cualquiera","label":"Elemento Lo que sea"}
]
```

Como se puede ver, en el JSON tenemos que especificar un array de diversos objetos con propiedades "label" y "value".

- La propiedad "label": sirve para definir el texto que se mostrará como sugerencia de autocompletado.
- La propiedad "value": sirve para definir el texto que se escribirá en el campo de

autocompletar cuando se seleccione esta opción.

Ahora que ya conocemos cómo utilizar un campo autocomplete para extraer datos de una página remota, podemos aprender cómo generar esos valores de autocompletado a través de una base de datos con PHP.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *20/12/2010*
Disponible online en *http://desarrolloweb.com/articulos/campo-autocompletado-carga-remota-jquery.html*

Accordion de jQueryUI

El plugin accordion de jQueryUI nos permite mostrar contenidos de una manera dinámica. Explicamos cómo utilizarlo.

A lo largo de varios artículos estamos explicando muchos de los componentes, también llamados "widgets", que jQueryUI dispone para los interesados en crear interfaces de usuario enriquecidas.

En este caso vamos a tratar un componente interesante, que seguro puede aportar mucho dinamismo a los contenidos de nuestra página web. Se trata del "Accordion", o acordeón en español, que permite mostrar diversos contenidos clasificados en secciones, de manera que se mantenga abierta una de esas secciones y se cierre automáticamente solo al abrir otra.

Nota: Si te interesa conocer jQueryUI desde el principio, recomendamos comenzar con la lectura del artículo Primeros pasos con jQuery UI.

Si conocemos un poco jQueryUI ya habremos podido experimentar lo sencillo que resulta, con el mínimo código, crear componentes de interfaces de usuario potentes y versátiles. El plugin Accordion no es una excepción, sino al contrario, resulta todavía más impresionante ver lo sencillo que es crear una interfaz muy atractiva con prácticamente nada de código.

Lo primero que podemos hacer es ver en este enlace cómo es un accordion y el código que hemos utilizado para generarlo:

```
$("#accordion").accordion();
```

Estructura HTML para generar los elementos del acordeón

Claro que esa línea de código es el Javascript necesario para iniciar el accordion, pero como te imaginarás, hará falta algo de HTML. Esta sería la estructura básica del HTML que

utilizaríamos para generar las secciones del accordion.

```
<div id="accordion">
   <h3><a href="#">Sección 1</a></h3>
   <div>Contenido 1...</div>
   <h3><a href="#">Sección 2</a></h3>
   <div>Contenido 2...</div>
   <h3><a href="#">Sección n</a></h3>
   <div>Contenido n</div>
</div>
```

Como ves, es un código HTML de lo más normal. En principio, según señalan en la documentación, deben generarse los titulares de las secciones en encabezamientos H3, pero durante las pruebas hemos visto que el widget por defecto intenta hacer el acordeón incluso con otros encabezados que podamos poner. No obstante, si queremos utilizar otro encabezado para las secciones podemos indicarlo al llamar al plugin en las "options", como veremos más adelante en este artículo.

Luego, el contenido de cada sección se coloca en una capa DIV, que puede tener cualquier texto con etiquetas HTML diversas, imágenes, etc.

Invocar el plugin Accordion

Una vez tenemos nuestro HTML ya solo nos queda invocar el plugin Accordion de jQueryUI, que se hace con la línea de código que hemos visto antes. Claro que siempre debemos llamarlo cuando el navegador esté listo para recibir acciones y para ello utilizamos el conocido evento "ready" del "document".

```
$(document).ready(function() {
   $( "#accordion" ).accordion();
});
```

> **Nota:** Por supuesto, tenemos que haber cargado previamente jQuery y jQueryUI, en una descarga que incluya el widget Accordion. Todo eso se explicó anteriormente.

Aquí podemos ver el acordeón generado con las opciones por defecto.

Opciones del plugin Accordion

El comportamiento que conseguimos con las opciones predeterminadas será suficiente en la mayoría de los casos, no obstante, existen algunas alternativas que podemos definir al invocar el componente. La lista completa la podemos ver en la documentación del Accordion de jQueryUI, aunque realmente no hay muchas que modifiquen de gran manera el acordeón.

Las opciones se expresan mediante un objeto de "options" que se pasa al invocar al plugin. Por

ejemplo, aquí estaríamos llamando al accordion con tres opciones:

```
$( "#accordion" ).accordion({
   'header': 'h2',
   'fillSpace': true,
   'active': 1
});
```

- **'header': 'h2** sirve para definir que se utilicen encabezados HTML de tipo H2 para los títulos de las secciones.
- **'fillSpace': true** sirve para que se ajuste a la altura del contenedor donde está colocado el acordeón, si es que ese contenedor tenía una altura definida.
- **'active': 1** hace que se muestre activa la segunda sección del acordeón (la primera sería la sección 0, así que la 1 es la segunda).

Podemos ver un Accordion generado utilizando esas tres opciones.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *04/07/2011*
Disponible online en *http://desarrolloweb.com/articulos/accordion-jqueryui.html*

Librería CSS de jQueryUI

jQueryUI dispone de un framework CSS que nos facilitará la creación de estilos para crear nuestras propias interfaces de usuario.

Framework CSS de jQueryUI

jQueryUI ofrece un framework CSS con estilos que facilitan la creación de interfaces de usuario y que se pueden utilizar en sitios web y en widgets o componentes basados en jQueryUI.

Para quienes están explorando las posibilidades de jQueryUI, que empezamos a explicar en el artículo sobre los primeros pasos en jQueryUI, les debe quedar claro que son unas librerías Javascript pensadas para la creación de componentes para interfaces de usuario en aplicaciones web enriquecidas del lado del cliente. En resumen, son una serie de componentes o widgets para hacer nuestros sitios más dinámicos y agradables de usar.

Sin embargo, hay una parte de este proyecto que nos llama la atención por ofrecer algo distinto. No son scripts para crear dinamismos o utilidades para la comunicación página-usuario, sino un paquete de estilos que nos pueden servir para decorar de una manera atractiva páginas web. Se trata de un framework CSS con estilos para definir el aspecto de interfaces de usuario en páginas web, que nos puede ayudar a aplicar un diseño más agradable a una página donde estemos usando jQuery UI.

> **Nota:** Para no llevarnos a engaño, queremos remarcar que el framework CSS del que vamos a hablar no tiene las funcionalidades básicas de otras librerías CSS ya comentadas como Blueprint o 960.gs, que ofrecen utilidades para la maquetación con CSS. En este caso de jQueryUI simplemente tendremos clases para crear estilos que podrían ser utilizados en crear interfaces de usuario.

Si bien es cierto que podemos utilizar los estilos CSS que jQueryUI nos ofrece para el diseño de cualquier elemento de un sitio web, están más pensandos para usarlos en la creación de widgets o componentes para páginas web basados en la arquitectura Javascript de jQueryUI.

Las clases defindas por jQueryUI y que forman el framework CSS del que estamos hablando están disponibles en cualquier descarga de jQuery UI y definidos por el tema concreto que hayamos seleccionado al hacer el download. Contienen estilos para definir muchas cosas diferentes, como esquinas redondeadas, elementos predeterminados, activos, desactivados, resaltados, con error, titulares, contenidos, distintos niveles de jerarquía, sombras, etc. Además, disponen de una pequeña biblioteca de 167 iconos, que quizás no parezcan muchos

pero que tienen todos los motivos típicos que se podrían usar en interfaces de usuario.

Como los estilos CSS de jQuery UI cambian según el tema que hemos elegido en la descarga, este framework CSS es altamente configurable en aspecto, proporcionando temas gráficos de bastante calidad y muy diversos, que se adaptarán sin duda a cualquier tipo de proyecto. jQuery UI tiene una galería de hasta 24 temas distintos y además dispone de una utilidad genial, que ellos llaman ThemeRoller, para personalizar cualquier estilo y crear temas nuevos fácilmente.

Antes de pasar a explicar cómo usar estos temas gráficos queremos resaltar la ventaja principal de usar estos temas gráficos en Interfaces de usuario basadas en jQueryUI. Resulta que estas librerías Javascript están pensadas para que sean fácilmente extensibles por otros programadores, de modo que cualquier pueda crear nuevos widgets basados en las funcionalidades de jQueryUI. Pues bien, al utilizar los estilos CSS que jQueryUI propone, nos estaremos asegurando que los componentes que creemos puedan intercambiar los temas gráficos diversos que las librerías han creado. De este modo, podríamos crear un componente en un sitio con fondo oscuro, que luego utilizaríamos también en sitios con fondos claros, simplemente cambiando el theme por otro de los que proponen en jQueryUI. Asimismo, otros desarrolladores podrán utilizar tus componentes y aplicarles los estilos que estimen oportunos en sus páginas. De este modo, nuestros componentes serán mucho más reutilizables, no solo por sus funcionalidades Javascript, sino también porque se puedan adaptar sin ningún esfuerzo a cualquier interfaz gráfica que queramos.

Clases del framework CSS de jQuery UI

Para explicar las clases del framework CSS de jQueryUI queremos antes que nada dejaros un link a la descripción que nos ofrecen en la propia documentación de jQueryUI. Allí podremos encontrar el listado completo de clases y las situaciones en las que se deben utilizar cada una de ellas.

Están divididas en diversos paquetes:

######### Layout Helpers:

Con diversas clases que nos sirven para ayudarnos a posicionar y destacar elementos dentro de la plantilla de un sitio web. Podemos ocultar elementos de diversas formas, solucionar el problema de maquetacion con float, o problemas de superposición de elementos cuando se utilizan iframes.

######### Widget Containers:

Con algunas clases que nos sirven para crear contenedores versátiles donde situar los componentes de interfaces de usuario. Existe un contenedor principal, en el que anidaremos luego contenedores con aspecto de titulares o de cuerpo.

######### Interaction States:

Estados de interacción con el usuario, que simbolizan el estado de los elementos, como predeterminado, con el ratón encima, con el foco de la aplicación o activos.

######### Interaction Cues:

Otra serie de estados, pero que esta vez nos ofrecen señales indicadoras para definir visualmente de qué manera un visitante puede interactuar con los distintos elementos que componen un widget. Contiene clases CSS que podemos usar para simbolizar estados como resaltado, error, desactivado o de jerarquía.

######### Icons:

Interesante base de datos de iconos minimalistas, de 16x16, que podemos utilizar para cualquier cosa que se nos ocurra. Los iconos son bastante generalistas, como abrir, cerrar, recargar, redimensionar, play, stop, subir, bajar, carpeta abierta o cerrada, símbolos mas, menos, etc.

######### Misc Visuals:

Algunos efectos visuales adicionales, como bordes de esquinas redondeadas o sombras.

Utilizar las clases CSS en nuestros proyectos

Para completar esta introducción al framework CSS de jQuery UI, ya solo nos queda dar un ejemplo sobre cómo se pueden aplicar estos estilos en las páginas web. Para quien sepa un poquito de CSS la verdad es que no debería significar un gran problema usar este framework, puesto que simplemente se necesita reconocer la clase que se desea utilizar y aplicarla dentro del atributo class del tag del elemento al que queremos aplicar el estilo.

Por ejemplo, queremos aplicar bordes redondeados a un elemento. Pues simplemente utilizamos la clase ui-corner-all sobre dicho elemento. Que queremos crear un aspecto de elemento desactivado, pues aplicamos la clase ui-state-disabled sobre la etiqueta de tal elemento. Es así de simple.

> **Nota:** Quizás sobre decir que necesitamos tener cargado el framework jQueryUI para que estos estilos funcionen, lo que incluye haber colocado el link con el CSS del tema gráfico que queramos aplicar. Todo esto se vio en el artículo Primeros Pasos con jQueryUI.

Para mostrar un ejemplo de uso de este framework CSS hemos creado un ejemplo de uso de las clases más importantes que nos ofrece.

En este ejemplo aplicamos diversos estilos CSS sacados del framework por medio de sus clases. Veamos ahora algunos de los estilos aplicados.

```
<h1 class="ui-state-default">Probando el framework CSS de jQueryUI</h1>
```

Esto hace que el titular H1 tenga el estilo predeterminado para los elementos. Lo único es que debemos aplicarle un padding para que quede bonito, que podría colocarse en nuestra hoja de

estilos general.

```
h1{
   padding: 10px;
}
```

Ese padding somos nosotros los que tenemos que definirlo, acorde con el aspecto de nuestro sitio, así como cualquier otro estilo personalizado que queramos aplicar a los elementos H1 para el sitio web que se está creando.

Podemos aplicar si lo deseamos varias clases CSS a un mismo elemento, lo que nos puede servir para aplicar dos o más estilos distintos.

```
<h2 class="ui-state-focus ui-state-disabled">Esto es un titulo H2 que tiene el foco y está desactivado</h2>
```

Este elemento H2 tendría el aspecto que tienen los elementos con el foco de la aplicación y además el estilo de estar desactivado.

```
<h2 class="ui-state-highlight ui-corner-all">Esto es un titulo H2 resaltado y con esquinas redondeadas</h2>
```

Este es otro ejemplo de estilo aplicado a través de dos clases, en el que decimos que un elemento tiene que aparecer resaltado y además con las esquinas redondeadas.

Los componentes que creemos pueden requerir el uso de más de una etiqueta, porque dispongan de varios contenedores a los que se aplica estilos de manera separada. En la creación de widgets tendremos una etiqueta general para el contenedor global del widget y luego dentro un contendor para el títular del widget y otro para el cuerpo.

```
<div class="ui-widget">
   <h3 class="ui-corner-top ui-widget-header">Cabecera de un Widget</h3>
   <div class="ui-corner-bottom ui-widget-content">Contenido del widget.</div>
</div>
```

Con ui-widget definimos el estilo del contenedor principal del componente y luego con ui-widget-header se define el estilo para su encabezado y con ui-widget-content el estilo para su cuerpo. Además, en este ejemplo se están aplicando bordes redondeados a algunas de las esquinas del componente, con ui-corner-top (esquinas redondeadas en la parte de arriba) y ui-corner-bottom (esquinas redondeadas abajo).

Otro caso interesante son los iconos, que tienen que crearse usando dos clases. La primera ui-icon que se utiliza para los estilos comunes a todos los iconos y la segunda que nos sirve para definir el tipo de icono concreto que se va a utilizar, como ui-icon-scissors, ui-icon-star, etc.

```
<span class="ui-icon ui-icon-star"></span>
```

```
<span class="ui-icon ui-icon-scissors"></span>
<span class="ui-icon ui-icon-folder-open"></span>
```

Eso son tres iconos distintos de entre los más de 160 disponibles. Consultar el ThemeRoller de jQueryUI para ver todos los iconos que nos permiten.

Si lo deseas, podrías meter el icono dentro de una caja con bordes redondeados, para dar un aspecto más de botón.

```
<div class="ui-state-default ui-corner-all cajaicono"><span class="ui-icon ui-icon-star"></span></div>
```

En este caso, aparte de las clases del framework he aplicado una clase creada por mi llamada "cajaicono" que me sirve para definir cosas como el tamaño de mi caja, márgenes, padding y cosas de ese estilo.

```
.cajaicono{
   width: 20px;
   padding: 4px 0 4px 4px;
   float: left;
   margin-right: 5px;
}
```

Ahora, si te fijas en el código del ejemplo, podrás ver que después de los botones colocados en cajas he colocado una capa vacía con la clase ui-helper-clearfix.

```
<div class="ui-helper-clearfix"></div>
```

Esto simplemente lo hago porque los botones en cajas están flotando a la derecha y los siguientes contenidos quiero que no sean afectados por esos elementos flotantes. Esta clase la verdad es que es un comodín muy interesante para solucionar ese problema y otros peores en los que el fondo de un contenedor no acompaña los elementos flotantes que pueda tener contenidos.

La belleza del intecambio de estilos CSS para sustituir la temática del diseño

Como comentaba al principio, los estilos del framework CSS de jQueryUI dependen del tema que se haya escogido. Por ello, utilizar todas estas clases tiene la ventaja de poder sustituir el "Theme" y con ello cambiar radicalmente el aspecto de nuestras interfaces de usuario.

Para ilustrar esta ventaja, en el ejemplo de uso de las clases CSS de jQuery UI, he creado un par de enlaces (abajo del todo) que sirven para intercambiar la hoja de estilo que se está utilizando. Con ello podremos ver de qué manera cambia el sitio web con solo pulsar el enlace.

Nota: El intercambio de la hoja de estilos se hace con la técnica mostrada en el artículo

intercambiar la hoja de estilos con jQuery.

Espero que con este pequeño detalle podamos entender lo fácil que es diseñar interfaces de usuario en jQueryUI y la ventaja de usar las clases del framework CSS, de modo que los widgets que creemos se adapten a cualquier temática visual que se pueda escoger en un sitio web.

Conclusión

Hemos visto que jQueryUI contiene un framework CSS bastante interesante. No es un framework CSS normal, porque no te sirve para maquetar con CSS una página, pero sí contiene muchas clases muy útiles, que pueden guiar al usuario de una manera más visual y estéticamente agradable entre las funcionalidades que hayamos implementado en una página.

Además, el lector habrá podido entender que los objetivos de las clases CSS que propone jQueryUI pasan también por ofrecer una estructura de diseño que se puede adaptar estéticamente a cualquier proyecto. A través de la descarga de temas gráficos del ThemeRoller, cualquier desarrollador puede seleccionar diversas declaraciones de estilos, de modo que los componentes se adapten fácilmente al aspecto de la página web que esté creando.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *26/09/2011*
Disponible online en *http://desarrolloweb.com/articulos/framework-css-jqueryui.html*

Taller de jQuery UI

Artículos prácticos que utilizan las librerías jQueryUI para la creación de páginas con interfaces de usuario avanzadas en Javascript.

Juego del ahorcado Javascript con jQuery UI y Canvas del HTML 5

Aplicación experimental del juego del ahorcado en Javascript, con jQuery y las librerías jQuery UI y uso del elemento Canvas del HTML 5.

En este artículo voy a presentar una aplicación experimental que he realizado, basada en el popular Juego del ahorcado. Hay personas que disfrutan montando maquetas de trenes, pero yo soy de los que se entretienen con la programación. Prueba de ello es el siguiente script, que realicé hace meses, únicamente con la intención de probar algunos de los más novedosos juguetitos para desarrolladores, como son jQuery UI o el elemento Canvas.

A pesar que no voy a explicar paso por paso cómo se construyó esta aplicación, me he decidido a escribir este artículo por si a alguien le interesa conocer una posible implementación del juego del Ahorcado en Javascript. Además, nos servirá de práctica para las personas que están aprendiendo jQuery o el dibujo con el elemento Canvas.

Para empezar, dejo un enlace a la página del juego en funcionamiento, para que los interesados puedan ver el resultado final de esta práctica.

> **Nota:** Para ver el juego en funcionamiento tendrás que entrar con un navegador medianamente moderno. Por lo menos tiene que ser compatible con el elemento Canvas del HTML 5.

Tecnologías usadas en la aplicación del ahorcado

En este juego del ahorcado hemos usado algunas de las tecnologías más modernas para el desarrollo de webs, las nombramos a continuación.

Framework Javascript jQuery:

Usamos jQuery para simplificar las cosas durante la programación en Javascript. Puedes aprender en el Manual de jQuery.

Librería jQuery UI:

Las librerías jQuery UI nos sirven para crear diversos componentes de interfaz de usuario avanzados, como botones o cajas de diálogo. Podría hacerse sin estas librerías, igual que también sin el propio jQuery, pero la verdad es que simplifican nuestra vida bastante y nos ayudan a realizar un juego más atractivo visualmente y facilitan una mejor experiencia de usuario. En DesarrolloWeb.com también puedes aprender jQuery UI.

Canvas del HTML 5:

Utilizamos el componente CANVAS para pintar cada una de las partes del cuerpo del ahorcado que se colocan según la persona que juega va cometiendo errores. La propia estructura de la ahorca también está pintada con el elemento Canvas del HTML 5. También podríamos hacer este juego sin utilizar el Canvas, colocando simples imágenes generadas con algún editor gráfico, pero justamente lo que deseamos es hacer una aplicación donde podamos practicar con el elemento Canvas. Puedes aprender sobre esto en el Manual de Canvas del HTML 5.

Aparte de todo esto, por supuesto, utilizamos el lenguaje Javascript, que se puede aprender en la sección Javascript a fondo.

Explicación básica del script

El primer pedazo de código que podemos revisar es donde se escoge una palabra. Esto se hace cuando la página carga. Ese pedazo de código está comentado con el texto "PALABRAS" a partir de la línea de código 37. Primero se define un array con todas las letras posibles y luego se crea una variable "palabraEscogida" con una de esas letras seleccionada aleatoriamente. Además, todo seguido, se define un array con los aciertos.

Una vez hemos reparado en el detalle de cómo se escoge la palabra inicialmente para el juego, podemos ver el resto del código. Para ver cómo hemos hecho este script para el juego del ahorcado recomiendo leer la parte en la que se inicia la aplicación y luego ir revisando cada una de las funciones a medida que se van invocando desde el inicio.

Como cualquier aplicación o script jQuery, el código que se ejecuta al principio está dentro método ready del objeto document.

```
$(document).ready(function(){
   //código que inicia la aplicación
}
```

1. Creamos un array con las letras del abecedario y luego un bucle para recorrerlas.
2. Para cada letra, generamos un elemento SPAN que tiene el texto de dicha letra y convertimos ese SPAN en un componente botón de las librerías jquery UI. Además, en el elemento SPAN guardo un dato con el método data() para saber qué letra corresponde a ese elemento.
3. También para cada una de las letras, se define un evento click con jQuery. El clic en usa de esas letras significa que el usuario la ha elegido para ver si estaba en la palabra que debía adivinar. A partir de los clics en esas letras se van ejecutando las acciones para mostrar esa letra, en caso que estuviera en la palabra, o colocar un pedazo más del

ahorcado, en caso que fuera un error.
4. Si la letra escogida estaba en la palabra, entonces la meto en el array de aciertos y la escribo en la página. Además, si estaban todas las palabras acertadas, muestro un mensaje en una caja de diálogo con el componente dialog de jQuery UI.
5. En caso que la letra escogida no estuviese en la palabra, incremento una variable con el número de fallos y dibujo el ahorcado con los fallos actuales. Si se ha llegado al número de fallos máximo, se muestra un mensaje con otra caja de diálogo.
6. Dentro del evento clic definido para toda letra que se haya pulsado, como última acción, se desactiva el botón y se elimina el evento clic, para que no se permita volver a pulsar sobre la misma letra.
7. Ya fuera del código del evento, pero todavía dentro del bucle para recorrer todas las letras, se hace un append() para colocar ese botón en la página.
8. Se dibuja el ahorcado con la función dibujaAhorcado(), enviando el número de fallos, que inicialmente es cero.
9. Además, se escribe la palabra escogida, inicialmente ocultando todas las letras con guiones, pues inicialmente no hay ninguna letra en el array de aciertos.

Código completo del juego del ahorcado

En las líneas anteriores hemos visto un resumen de la funcionalidad principal del juego del ahorcado, pero claro que hay mucho más código para hacerlo funcionar. Además, que deberíamos haber hecho otras cosas como incluir los scripts jQuery y jQuery UI, así como la hoja de estilos.

Podemos ver a continuación las 300 líneas de código que hacen el juego completo.

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

<html>
<head>
   <title>Aplicaciçon de Ahorcado</title>
   <link type="text/css" href="css/cupertino/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>
function aleatorio(inferior,superior){
numPosibilidades = superior - inferior + 1
aleat = Math.random() * numPosibilidades
aleat = Math.floor(aleat)
return parseInt(inferior) + aleat
}

function esta(caracter, miarray){
//console.log("buscando ", caracter, " en ", miarray)
for(var j=0;j<miarray.length;j++){
if (caracter==miarray[j]){
      return true;
}else{
      //console.log("el caracter ", caracter, " no es el valor del array ",miarray[j] )
   }
```

```
}
return false;
}

function estanTodas(arrayAciertos, mipalabra){
   for(var i=0; i<mipalabra.length; i++){
      if(!esta(mipalabra.charAt(i),arrayAciertos))
         return false;
   }
   return true;
}

////////////////////////////////////////////////////////////////////////////
// PALABRAS
////////////////////////////////////////////////////////////////////////////
var palabras = ['ahorcado', 'lavadora', 'invierno', 'plastico', 'ordenador', 'colador', 'guantera', 'alimentador', 'calculos'];
var palabraEscogida = palabras[aleatorio(0,palabras.length-1)]
var aciertos = [];

//console.log(palabraEscogida);

function escribePalabra(palabra, arrayAciertos){
   //console.log("estoy en escribePalabra y arrat de aciertos es: " , arrayAciertos);
   var texto = '';
   for(var i=0; i<palabra.length; i++){
      texto += "<span>";
      var cActual = palabra.charAt(i);
      if(esta(cActual,arrayAciertos)){
         texto += cActual;
      }else{
         texto += '_';
      }
      texto += "</span>";
      //console.log(cActual)
   }
   $("#letras").html(texto);
}

////////////////////////////////////////////////////////////////////////////
//// inicio todo!!!
////////////////////////////////////////////////////////////////////////////
$(document).ready(function(){

   //creo los botones con las letras
   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();
      letraActual.click(function(){
         //traigo la letra pulsada
         var miletra = $(this).data("letra").toLowerCase()
```

```
         //miro si esa letra está en la palabra
         if(palabraEscogida.indexOf(miletra)!=-1){
            //si está, va para aciertos
            aciertos.push(miletra);
            escribePalabra(palabraEscogida, aciertos);
            //miro si ha ganado
            if(estanTodas(aciertos,palabraEscogida)){
               var caja = $('<div class="dialogletra" title="Has Ganado!!">Felicidades! has adivinado la palabra!!</div>');
               caja.dialog({
               modal: true,
               width: 600,
               buttons: {
                  "Ok": function(){
                     $(this).dialog("close");
                  }
               }
               });
            }
         }else{
            //no estaba
            numFallos++;
            dibujaAhorado(numFallos);
            //miro si se ha perdido
            if(numFallos==6){
               var caja = $('<div class="dialogletra" title="Has Perdido!!">Lo lamento!! la palabra era: ' + palabraEscogida + '</div>');
               caja.dialog({
               modal: true,
               width: 600,
               buttons: {
                  "Ok": function(){
                     $(this).dialog("close");
                  }
               }
               });
            }
         }
         //una vez pulsado el botón, lo desabilito y quito su evento click
         $(this).button("disable");
         $(this).unbind( "click" );

      })
      $("#botonesletras").append(letraActual);
   }

   //inicio el canvas
   dibujaAhorado(numFallos);

   //inicio las palabras
   escribePalabra(palabraEscogida, aciertos);

});

////////////////////////////////
//CANVAS
////////////////////////////////
```

```
function cargaContextoCanvas(idCanvas){
   var elemento = document.getElementById(idCanvas);
   if(elemento && elemento.getContext){
      var contexto = elemento.getContext('2d');
      if(contexto){
         return contexto;
      }
   }
   return false;
}
function borrarCanvas(contexto, anchura, altura){
   contexto.clearRect(0,0,anchura,anchura);
}
function dibujaHorca(ctx){
   ctx.fillStyle = '#462501';
   ctx.fillRect(64,9,26,237);
   ctx.fillRect(175,193,26,53);
   ctx.fillRect(64,193,136,15);
   ctx.fillRect(64,9,115,11);
   ctx.beginPath();
   ctx.moveTo(64,65);
   ctx.lineTo(64,80);
   ctx.lineTo(133,11);
   ctx.lineTo(118,11);
   ctx.fill();
}
function dibujaCabeza(ctx){
   var img = new Image();
   img.onload = function(){
      ctx.fillStyle = '#f2d666';
      ctx.drawImage(img,150,38);
      ctx.fillRect(172,12,4,28);
   }
   img.src = 'images/picture.jpg';
}
function dibujaCuerpo(ctx){
   ctx.beginPath();
   ctx.moveTo(171,82);
   ctx.lineTo(168,119);
   ctx.lineTo(162,147);
   ctx.lineTo(189,149);
   ctx.lineTo(185,111);
   ctx.lineTo(183,83);
   ctx.fill()
}
function dibujaBrazoIzq(ctx){
   ctx.beginPath();
   ctx.moveTo(173,102);
   ctx.lineTo(140,128);
   ctx.lineTo(155,133);
   ctx.lineTo(178,114);
   ctx.fill()
}
function dibujaBrazoDer(ctx){
   ctx.beginPath();
```

```
    ctx.moveTo(180,99);
    ctx.lineTo(222,121);
    ctx.lineTo(209,133);
    ctx.lineTo(183,110);
    ctx.fill()
}
function dibujaPiernaIzq(ctx){
    ctx.beginPath();
    ctx.moveTo(166,142);
    ctx.lineTo(139,175);
    ctx.lineTo(164,178);
    ctx.lineTo(175,144);
    ctx.fill()
}
function dibujaPiernaDer(ctx){
    ctx.beginPath();
    ctx.moveTo(178,145);
    ctx.lineTo(193,178);
    ctx.lineTo(212,170);
    ctx.lineTo(188,142);
    ctx.fill()
}
//////////////////////////////////////////////////////
// GESTION DE FALLOS
//////////////////////////////////////////////////////
var numFallos = 0;
function dibujaAhorado(numerrores){
    var contexto = cargaContextoCanvas('canvasahorcado');
    if(contexto){
        dibujaHorca(contexto);
        if(numFallos>0){
            dibujaCabeza(contexto)
        }
        contexto.fillStyle = '#1f3e18';
        if(numFallos>1){
            dibujaCuerpo(contexto)
        }
        if(numFallos>2){
            dibujaBrazoIzq(contexto)
        }
        if(numFallos>3){
            dibujaBrazoDer(contexto)
        }
        if(numFallos>4){
            dibujaPiernaIzq(contexto)
        }
        if(numFallos>5){
            dibujaPiernaDer(contexto)
        }

    }
}

</script>
```

```
<style type="text/css">
body{
   font-size: 0.7em;
   font-family: Trebuchet MS, verdana, arial, sans-serif;
}
.botonletra{
   font-size: 0.9em;
   margin: 2px;
   width: 30px;
   text-align: center;
}
.dialogletra{
   font-size: 3em;
   text-align: center;
   padding-top: 15px;
}
#botonesletras{
   width: 330px;
   clear: both;
}
#dibujoahorcado{
   margin-bottom: 20px;
}
#letras{
   font-size: 3em;
   text-align:center;
   width: 320px;
   clear: both;
   margin-bottom: 10px;
}
#letras span{
   width: 30px;
   text-align:center;
   padding-left: 5px;
}
</style>

</head>

<body>
<div id="dibujoahorcado">
   <canvas id="canvasahorcado" width="320" height="250">
   El Ahorcado sólo funciona en navegadores que soporten Canvas. Actualízate a Firefox o Chrome, por poner dos posibilidades.
   </canvas>
</div>
<div id="letras">
</div>
<div id="botonesletras"></div>
</body>
</html>
```

Eso es todo!! quedan bastantes cosas que explicar en el tintero, pero estoy seguro que se podrán entender siguiendo los comentarios del código y las referencias a manuales y artículos donde explicamos cosas como jQuery o el componente Canvas.

Para acabar, podemos ver el ejercicio en marcha en una página aparte.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *25/01/2011*
Disponible online en *http://desarrolloweb.com/articulos/juego-ahorcado-javascript-jquery-canvas.html*

Ejemplo más complejo de Drag & Drop con jQueryUI

Veamos casos un poco más elaborados de utilidades Drag & Drop en páginas web, por medio de jQueryUI.

En el Manual de jQueryUI hemos tratado ya bastantes cosas tanto de los elementos Draggable como de los elementos Droppable. Creo que para afianzar estos conocimientos nos vendría bien trabajar con algún ejemplo un poquito más elaborado.

A este ejemplo que vamos a realizar lo he llamado Azules y Rojos y consiste en diversas capas de colores azules y rojos y dos contenedores que aceptan soltar sobre ellos elementos de un tipo concreto. El contenedor rojo acepta que soltemos sobre él cajitas rojas y el azul solo acepta cajitas azules, llevando la cuenta de todas las cajitas que fueron depositadas en ellos.

Es bien simple, pero para darnos cuenta de lo que vamos a realizar os recomiendo parar un instante para ver el ejemplo en marcha.

Nuestro HTML

Comencemos por analizar el código HTML que tenemos en nuestro ejemplo.

```
<div class="azul arrastrable" style="top: 140px; left: 150px;"></div>
<div class="rojo arrastrable" style="top: 98px; left: 100px;"></div>
<div class="rojo arrastrable" style="top: 120px; left: 180px;"></div>
<div class="azul arrastrable" style="top: 120px; left: 90px;"></div>

<div id="sueltarojo" class="suelta">
   Suelta aquí elementos rojos
</div>
<div id="sueltaazul" class="suelta">
   Suelta aquí elementos azules
</div>
```

Como puedes ver tenemos varios elementos DIV con la clase "arrastrable", que son los elementos que vamos a poder arrastrar por la página. A estos elementos voy a llamartes cajitas. Adicionalmente, estas cajitas tienen la clase "azul" unas y otras la clase "rojo". Esas clases nos van a servir, no solo para decorarlos con CSS, sino para saber cuáles de esas cajitas se van a aceptar en los contenedores correspondientes. Bien sencillo.

Además tenemos un poco más abajo en el código HTML un par de contendores, también elementos DIV, que son los lugares donde voy a soltar las cajitas y solo aceptarán las del color que le corresponda.

Javascript para crear los elementos "draggables" (las cajitas)

Ahora que tenemos clara la noción de los elementos HTML que forman parte de nuestro ejemplo, podemos analizar el código Javascript que servirá para crear las funcionalidades dinámicas. Comenzamos viendo cómo hacer que las cajitas se puedan arrastrar.

```
$(".arrastrable").draggable();
```

Con esta sencilla llamada al método draggable() podemos arrastrar las cajitas, a las que les había puesto la clase "arrastrable". La verdad es que esta parte es bien simple y como veremos en breve, la complejidad se nos va a concentrar en los elementos "droppable".

No obstante, aquí vemos un código adicional que he creado simplemente para saber si un elemento fue o no soltado anteriormente sobre un contenedor que lo aceptase.

```
$(".arrastrable").data("soltado", false);
```

Ese método data(), como debemos saber, me sirve para almacenar cualquier tipo de dato dentro de un objeto jQuery. Insisto, no es que necesite hacer esto para convertir a las cajitas en arrastrables, es solo que la operativa de mi ejemplo va a necesitar saber si una cajita fue ya depositada, o no, en un contenedor.

Definición del comportamiento de soltar ("droppable" sobre los contenedores)

Ahora vamos a ver el Javascript para definir el comportamiento de soltar cajitas sobre los contenedores. Pero antes un detalle. Como había dicho, pretendo llevar la cuenta de las cajitas que fueron soltadas en cada contenedor. Para ello voy a crear con el método data() una variable para hacer de contador.

```
$(".suelta").data("numsoltar", 0);
```

Lo hago sobre los elementos de la clase "suelta", que es una clase que tienen tanto el contenedor rojo como el azul.

Ahora, sobre esta misma clase "suelta" voy a definir con jQueryUI que sean elementos "droppable", que es donde está la mayor complejidad de este ejercicio.

```
$(".suelta").droppable({
   drop: function( event, ui ) {
      if (!ui.draggable.data("soltado")){
         ui.draggable.data("soltado", true);
```

```
            var elem = $(this);
            elem.data("numsoltar", elem.data("numsoltar") + 1)
            elem.html("Llevo " + elem.data("numsoltar") + " elementos soltados");
         }
      },
      out: function( event, ui ) {
         if (ui.draggable.data("soltado")){
            ui.draggable.data("soltado", false);
            var elem = $(this);
            elem.data("numsoltar", elem.data("numsoltar") - 1);
            elem.html("Llevo " + elem.data("numsoltar") + " elementos soltados");
         }
      }
   });
```

Se trata de una llamada al método droppable(), pasando dos funciones para gestionar tanto el evento "drop" (que se ejecuta al soltar un "draggable"), como el evento "out" (que se ejecuta al sacar un "draggable"). Lo interesante está justamente en estos eventos.

El evento drop, que se activa al soltar una cajita, realiza una comprobación sobre la cajita que se está soltando. Para acceder al objeto jQuery de esa cajita utilizamos el parámetro "ui" que recibe la función del evento y su propiedad draggable. Es decir, ui.draggable es una referencia al objeto jQuery que estamos soltando encima del contenedor. La comprobación simplemente accede a la variable que habíamos almacenado en la cajita con el método data(), para saber si esta cajita se había soltado previamente o no.

Si no se había soltado, por medio del evento drop realizo todas las operaciones que deseo, que son:

- Marcar a la cajita como que ha sido soltada, almacenando otro valor con data().
- Acceder al elemento contenedor y guardar una referencia en la variable elem.
- Incrementar en 1 el número de cajitas soltadas sobre ese elemento, por medio del contador que habíamos creado con data().
- Cambiar el contenido del contenedor para decir que llevo tantos elementos soltados como los indicados por nuestro contador.

Por su parte, el evento out, que se activa al sacar una cajita que previamente fue soltada, hace un poco lo mismo pero en sentido contrario. Es decir, si la caja estaba soltada previamente y ahora la estamos sacando del contenedor, pues la marco como no soltada, resto 1 al contador y actualizo el texto del contenedor.

Hasta aquí todo bien ¿no? Pero ahora os podéis preguntar ¿no habíamos quedado en que el contenedor azul solo tenía que aceptar las cajitas azules y el rojo solamente las rojas? Efectivamente, pero por darle variedad a este ejemplo, he dejado esas dos modificaciones fuera de la creación de los "droppables".

```
//soltar solo elementos rojos
$("#sueltarojo").droppable("option", "accept", ".rojo");
//soltar solo elementos azules
```

```
$("#sueltaazul").droppable("option", "accept", ".azul");
```

Con este código conseguimos modificar propiedades de los elementos droppable, para que solo acepten los elementos de una clase. En realidad se hace simplemente marcando con el atrubito "accept" el valor del selector que identifique los elementos que sí deseamos aceptar. En este caso los elementos de la clase "rojo" se aceptan en el contenedor con id "sueltarojo" y los elementos de la clase "azul" se aceptan en el contenedor con id "sueltazul".

Insisto en que ese comportamiento lo podía haber definido al invocar al método droppable(), en el momento de creación del comportamiento de soltar. Sin embargo, también por medio del método "option" puedo hacer esta configuración después de haber creado el componente droppable.

Conclusión

La funcionalidad básica de este ejercicio, al menos la que tiene que ver con el Drag & Drop, ya la hemos visto toda. En el ejemplo en marcha podrás ver que también hay unos enlaces para poder hacer más cajitas, azules o rojas, en tiempo de ejecución de la página. Esas cajitas creadas sobre la marcha se posicionan en un lugar aleatorio de la página y las podemos arrastrar también sobre los elementos droppable.

El código para crear ese comportamiento en los es el siguiente:

```
//enlaces para crear nuevos elementos rojos y azules
$(".creaelemento").click(function(e){
   e.preventDefault();
   var posx = aleatorio(10, 500);
   var posy = aleatorio(80, 200);
   var nuevoElemento = $('<div class="' + $(this).attr("href") + ' arrastrable" style="top: ' + posy + 'px; left: ' + posx + 'px;"></div>');
   nuevoElemento.draggable();
   $(document.body).append(nuevoElemento);
})
```

No voy a explicar este código de momento, pues no tiene nada en especial que no hayamos visto o en el Manual de jQuery o en el Manual de jQueryUI.

Espero que la parte del comportamiento de arrastrar y soltar, que era la que nos interesaba, se haya podido entender perfectamente con este ejemplo.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *02/05/2012*
Disponible online en *http://desarrolloweb.com/articulos/ejemplo-drag-drop-jquery.html*

Elementos ordenables con "sortable" de jQueryUI

Cómo hacer elementos cuyos items se puedan ordenar, por medio del interactor sortable de jQueryUI.

Dentro de jQueryUI hay muchos componentes que son realmente sencillos de poner en marcha. Es el caso de los elementos sortable, que no son widgets precisamente, sino que están clasificados dentro de los interactions. En el Manual de jQueryUI ya hemos visto en marcha varios plugins para interacción, como los droppable o draggable.

En concreto el comportamiento que vamos a explorar ahora se llama sortable y lo traduciríamos por ordenable. Sirve para generar interacción con el cliente, de modo que pueda ordenar los items de, por ejemplo, una lista. Es muy sencillo de implementar, como muchos otros plugins de jQueryUI.

Veamos un ejemplo de lista HTML que luego convertiremos en un sortable.

```
<ul id="ordenable">
   <li>Elemento ordenable</li>
   <li>Otro que se ordena</li>
   <li>...</li>
</ul>
```

Para darle la funcionalidad de ordenación a los elementos de esta lista, podemos invocar a la interacción con una única línea de código.

```
$("#ordenable").sortable();
```

Ahora esos elementos de la lista se pueden ordenar perfectamente, arrastrando y soltando en otra posición. Podemos ver el ejemplo en marcha.

Claro que la lista se ve muy parecida a una lista normal y nadie se va a dar cuenta que esos elementos se pueden ordenar con una acción de arrastrar y soltar, pero para eso podemos asignar algunas propiedades especiales a los elementos ordenable y algunos estilos CSS. Todo esto lo veremos a continuación.

Colocar iconos de jQueryUI a los elementos de la lista

Comenzamos viendo cómo podemos colocar unos iconos en los elementos de la lista, con unas flechitas que indiquen que se pueden ordenar. Como vemos, dentro del HTML de cada LI, colocamos un SPAN con una clase determinada que nos permite que se convierta en un icono.

```
<ul id="ordenable">
<li><span class="ui-icon ui-icon-arrowthick-2-n-s"></span> Elemento ordenable</li>
<li><span class="ui-icon ui-icon-arrowthick-2-n-s"></span> Otro que se ordena</li>
<li><span class="ui-icon ui-icon-arrowthick-2-n-s"></span> ...</li>
</ul>
```

Nota: La clase ui-icon y ui-icon-arrowthick-2-n-s son dos clases pertenecientes a jQueryUI que formarían parte del framework CSS que viene incluido con estas librerías y que se explicó en el artículo Librería CSS de jQueryUI.

Configurar propiedades del elemento ordenable

También podemos hacer un pequeño cambio para que, al arrastrar un elemento de la lista para ordenarlo, se cree un espacio en blanco entre los elementos que se posicionaría si lo soltásemos en ese momento.

```
$("#ordenable").sortable({
   placeholder: "ui-state-highlight"
});
```

El valor que asignemos a placeholder es el correspondiente a la clase CSS que queremos que se aplique para estilizar el lugar donde se soltaría el elemento. En este caso utilizamos "ui-state-highlight", que es una de las clases del framework CSS de jQueryUI.

Además, en nuestro CSS podríamos aplicar estilos a esta clase, o cualquier otra que tengamos a bien utilizar.

```
.ui-state-highlight { height: 1.5em; line-height: 1.2em; }
```

Nota: Otra utilidad muy interesante que nos permite jQueryUI y que está disponible en cualquier download de las librerías, por formar parte del core, es invocar el método disableSelection() sobre un elemento ordenable. Este método, no documentado en el momento de escribir este artículo, sirve para evitar que el texto de un elemento se pueda seleccionar.

```
$("#ordenable").disableSelection();
```

Es muy útil debido a que a menudo, al intentar ordenar un elemento, podríamos acabar seleccionando el texto del elemento en vez de arrastrarlo a otra posición.

Podemos ver una lista un poco más elaborada, con los estilos y configuraciones vistos anteriormente en una página aparte.

Existen multitud de propiedades en los elementos sortable que nos sirven para hacer diversas cosas, es cuestión de informarse de ellos en la documentación de jQueryUI. Por ejemplo, si queremos que algunos de los elementos de la lista no se puedan ordenar, podemos utilizar la propiedad cancel, asignando el selector de los elementos no ordenables.

```
<ul id="ordenable">
   <li>Elemento ordenable</li>
   <li class="no">Otro que se ordena</li>
   <li>otro elemento</li>
</ul>

$("#ordenable").sortable({
   cancel: ".no"
});
```

Con esto conseguimos que los elementos de la lista que tienen la clase "no" estén inhabilitados para modificar su posición.

Enviar el nuevo orden por Ajax a otra página del servidor

Ahora la pregunta que muchos de vosotros os podéis hacer ¿Cómo puedo procesar el orden actual, para almacenarlo en una base de datos o algo parecido?.

Afortunadamente, jQueryUI tiene métodos para hacer lo que queramos una vez cambiado el orden de los elementos. Existe el evento "update" que sirve justamente para realizar cualquier tipo de acción cuando se altera el orden de los elementos que hemos convertido en sortable.

Ahora vamos a ver un ejemplo para enviar por Ajax al servidor el nuevo orden de los elementos. Ese orden lo enviaremos en una cadena por GET y ya en la programación del servidor podréis hacer lo que se os antoje, como almacenarlo en una base de datos, enviarlo por email o lo que sea.

Veamos el siguiente código:

```
$("#ordenable").sortable({
   placeholder: "ui-state-highlight",
   update: function(){
      var ordenElementos = $(this).sortable("toArray").toString();
      $.get("cambia_orden.php",{nuevo_orden: ordenElementos},function (respuesta){
         alert(respuesta);
      });
   }
});
```

Con el evento update definimos las acciones a ejecutar al alterarse el orden. En este ejemplo generamos un texto a partir de la lista, y luego ejecutamos un $.get() para enviarlo por Ajax a una página PHP. Cuando se reciban los datos que devuelve esa página PHP se mostrarán en una caja de alerta.

Esto se puede ver en marcha en una página aparte. Os sugerimos activar el Firebug para ver por consola la forma que tiene la solicitud Ajax que estamos ejecutando con $.get().

Conclusión

Hemos visto hasta qué punto es potente el sistema de ordenar elementos que implementa jQueryUI. Nosotros tenemos que definir muchos asuntos, como los estilos, comportamientos especiales, etc. pero el sistema es capaz de realizar la mayoría de las cosas que nos podamos imaginar.

Como has podido ver, no solo se trata de hacer un sistema que se pueda manejar desde el lado del cliente para poder ordenar los elementos, sino también una interfaz que pueda comunicar con el servidor para enviar a cualquier lugar las actualizaciones del orden y efectuar cualquier tipo de operación.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *03/05/2012*
Disponible online en *http://desarrolloweb.com/articulos/elementos-sortable-jqueryui.html*

jQuery UI Menu. Manual de uso

En el presente artículo hablaremos de qué es y qué proporciona el jQuery User Interface Menu.

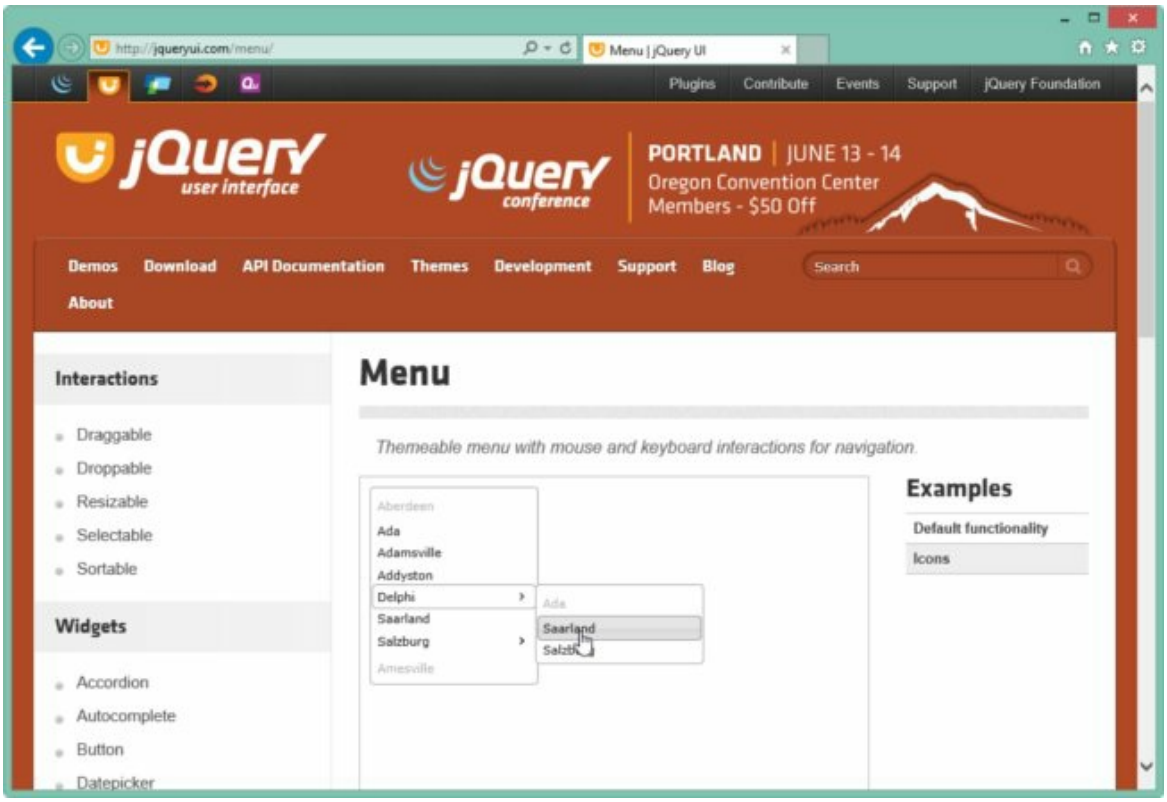
En el presente artículo hablaremos de:

- Qué es y qué proporciona el *jQuery User Interface Menu*
- Los componentes esenciales para su implementación en una página Web
- Su personalización mediante la aplicación de estilos prediseñados y otros redefinidos -sobrecargados-
- Cómo desactivarlo y reactivarlo
- Cómo adaptarlo para crear una disposición de menú horizontal

¿Qué es el jQuery User Interface Menu?

Dentro del *jQuery User Interface* encontramos toda una serie de elementos de programación de la interface de usuario preprogramados y listos para ser integrados en nuestros proyectos HTML. Se trata de una amplia biblioteca *JavaScript* que abarca desde efectos dinámicos, hasta menús, calendarios, etc.

El lugar de Internet de referencia y descarga del componente *Menu* es:
http://jqueryui.com/menu/



Las capacidades que nos proporcionará las iremos desgranando en los sucesivos epígrafes, a vuelapluma son:

- Creación de un menú a partir de una lista no ordenada
- Adaptación del menú con estilos prediseñados del *jQuery User Interface*
- Deshabilitar y rehabilitar el menú
- Personalización del menú mediante estilos
- Adaptación con estilos a un menú mostrado horizontalmente

Cuál es el punto de partida

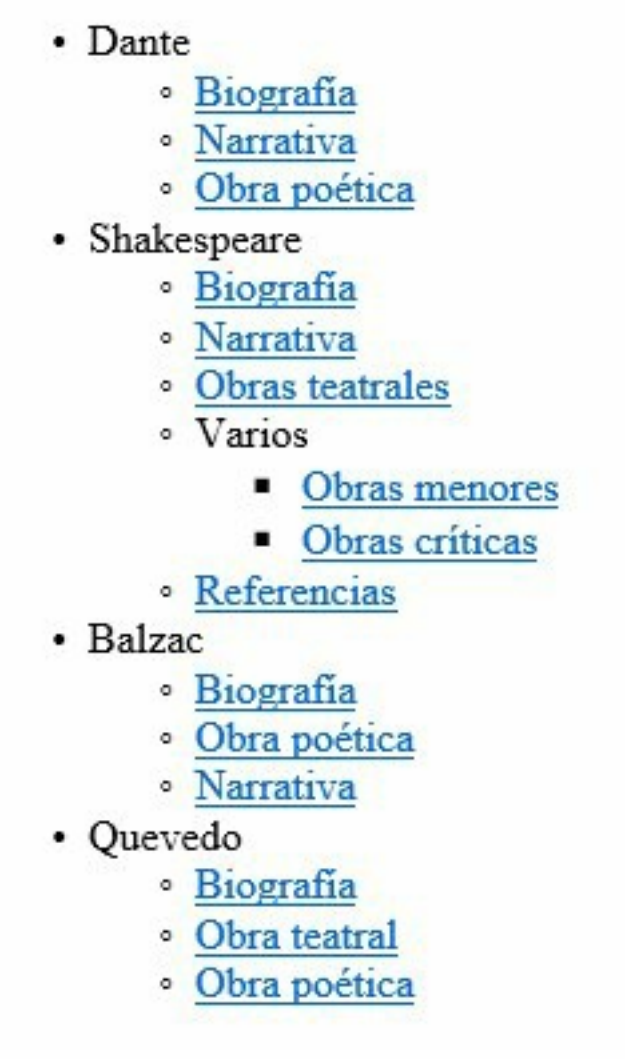
Comenzaremos por las tareas de inicialización y el cómo mostrar un menú en nuestra página Web; el punto de partida es una lista no ordenada, como por ejemplo la mostrada en el Listado 1 y en la figura adjunta.

Las opciones de menú serán finalmente aquellas entradas de las listas que contienen los hipervínculos asociados.

Listado 1: Lista no ordenada que nos servirá de base para el menú flotante.

```
<!DOCTYPE html>
<html>
<head>
<title>Menús flotantes, sólo esquema</title>
```

```
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
</head>
<body>

<div>
<ul>
<li>Dante
<ul>
<li><a href="#11">Biografía</a></li>
<li><a href="#12">Narrativa</a></li>
<li><a href="#13">Obra poética</a></li>
</ul>
</li>
<li>Shakespeare
<ul>
<li><a href="#21">Biografía</a></li>
<li><a href="#22">Narrativa</a></li>
<li><a href="#23">Obras teatrales</a></li>
<li>Varios
<ul>
<li><a href="#241">Obras menores</a></li>
<li><a href="#242">Obras críticas</a></li>
</ul>
</li>
<li><a href="#25">Referencias</a></li>
</ul>
</li>
<li>Balzac
<ul>
<li><a href="#31">Biografía</a></li>
<li><a href="#32">Obra poética</a></li>
<li><a href="#33">Narrativa</a></li>
</ul>
</li>
<li>Quevedo
<ul>
<li><a href="#31">Biografía</a></li>
<li><a href="#32">Obra teatral</a></li>
<li><a href="#33">Obra poética</a></li>
</ul>
</li>
</ul>
</div>

</body>
</html>
```



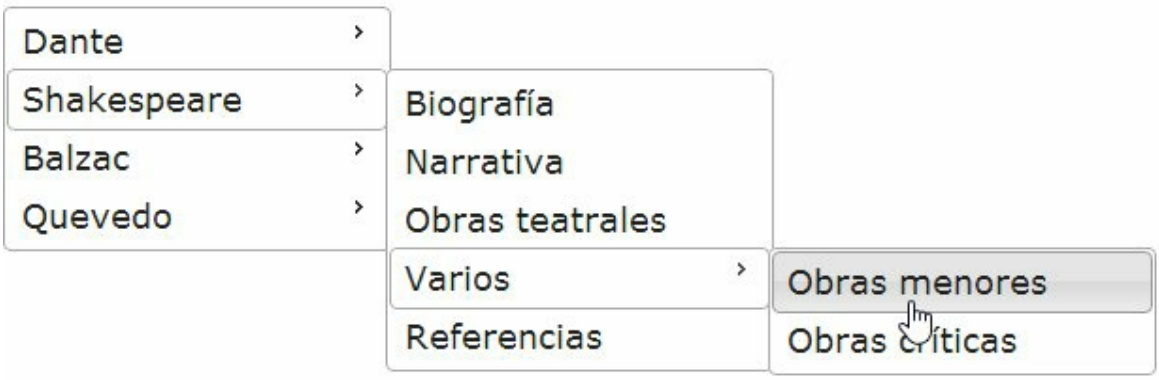
Cómo pasar del esquema al menú flotante

Observemos el Listado 2, que pasaremos a comentar con detalle.

Listado 2: Uso básico de un componente *Menu*

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Menu - Menú básico</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>

<style>
.ui-menu {
width: 200px;
}
</style>
```

```
<script>
$(function () {
$("#menu").menu();
});
</script>

</head>
<body>

<div style="position: absolute; top: 20px; left: 0px; z-index: 100">
<ul id="menu">
<li><a href="#">Dante</a>
<ul>
<li><a href="#11">Biografía</a></li>
<li><a href="#12">Narrativa</a></li>
<li><a href="#13">Obra poética</a></li>
</ul>
</li>
<li><a href="#">Shakespeare</a>
<ul>
<li><a href="#21">Biografía</a></li>
<li><a href="#22">Narrativa</a></li>
<li><a href="#23">Obras teatrales</a></li>
<li><a href="#">Varios</a>
<ul>
<li><a href="#241">Obras menores</a></li>
<li><a href="#242">Obras críticas</a></li>
</ul>
</li>
<li><a href="#25">Referencias</a></li>
</ul>
</li>
<li><a href="#">Balzac</a>
<ul>
<li><a href="#31">Biografía</a></li>
<li><a href="#32">Obra poética</a></li>
<li><a href="#33">Narrativa</a></li>
</ul>
</li>
<li><a href="#">Quevedo</a>
<ul>
<li><a href="#31">Biografía</a></li>
<li><a href="#32">Obra teatral</a></li>
<li><a href="#33">Obra poética</a></li>
</ul>
</li>
</ul>
</div>

</body>
</html>
```



Si repasamos el código nos encontramos, por orden secuencial:

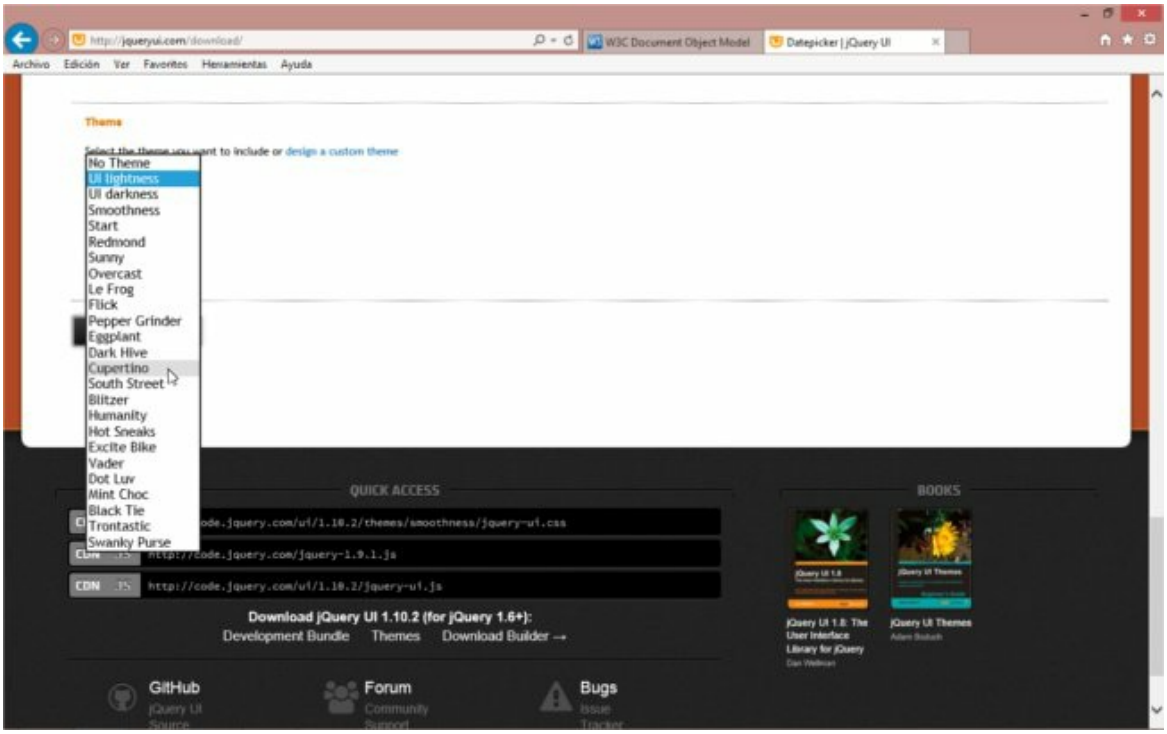
- Al comienzo del documento, en la etiqueta `<head>`, insertaremos las bibliotecas CSS y *jQuery* y los códigos correspondientes
- La referencia a la hoja de estilos del *User Interface de jQuery*: *jquery-ui.css* Debe ser la primera de las bibliotecas CSS referenciadas
- La referencia a la biblioteca *jQuery general jquery-x.x.x.js* Debe ser la primera de las bibliotecas *JavaScript* referenciadas
- La referencia a la biblioteca *User Interface de jQuery jquery-ui.js*
- Una función de inicialización del Menu *$("#menu").menu();* Que, como vemos, toma el elemento al que va asociado y la función sin parámetros, sin más
- En el cuerpo del documento, etiqueta `<body>` irá el resto de elementos HTML, entre ellos la definición del menú –lista no ordenada-
- En el esquema –lista no ordenada- hemos añadido referencias nulas -`<a href="#">`- a las cabeceras `<li>` principales, para que el puntero del ratón se muestre como un dedo apuntando y se muestre también una punta de flecha a su derecha indicando que se trata de una entrada de submenú
- Identificamos la lista numerada para poder asignarla a un componente *Menu*

Jugando con los estilos

Hasta ahora hemos trabajado con referencias al *User Interface* alojado en el foro de *jQuery;* de esa manera disponemos de un modo muy cómodo y relativamente eficiente de crear los *Datepicker*, pero no de personalizar sus estilos, dado que se trabaja con uno predeterminado –una CSS referenciada concreta, dada por el administrador del *jQuery User Interface*-.

Sin embargo el propio *jQuery User Interface* dispone de numerosos estilos prediseñados y listos para ser utilizados a demanda; pero para ello deberemos descargarlos y trabajar con ellos y referenciarlos en local –en el propio servidor-.

Podremos descargar el *jQuery User Interface* completo, que incluye *Menu*, la dirección URL es: http://jqueryui.com/download/



En ese proceso de descarga, en la parte inferior nos encontraremos con el apartado **Theme**, desplegando la lista podemos seleccionar el estilo del *Menu* que se descargará. En realidad, según lo que hayamos seleccionado previamente, es el estilo –el tema- de los componentes del *jQuery User Interface* que descargaremos.

De ahí es de donde se extraen los correspondientes CSS y directorios images particulares.

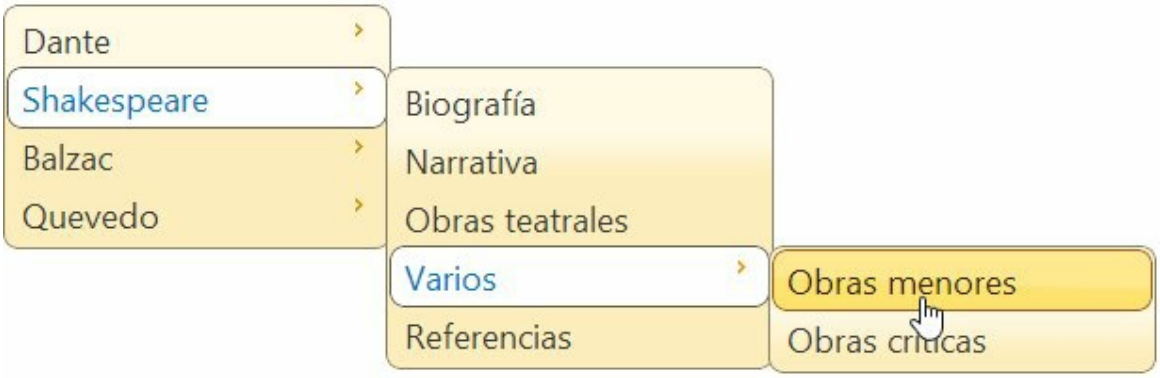
> **Nota:** En nuestro ejemplo hemos renombrado el archivo CSS descargado con el fin de poder trabajar con diferentes estilos, añadiéndole el nombre de estilo y cargándolo en local –véase la referencia correspondiente en el listado-. Trabajando en local, también deberemos copiar en cada caso el correspondiente directorio images y si queremos poder variar los estilos, darle un nombre particular para cada uno de los estilos –por ejemplo images_sunny- y cambiar las referencias en el archivo CSS correspondiente.

Listado 3: Un menú basado en un componente Menu, en el que se ha utilizado el estilo Sunny

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Menu - Menú básico con estilo personalizado</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>

<link rel="stylesheet" href="jquery-ui-1.10.2.sunny.css" />
```

```
<style>
.ui-menu {
width: 200px;
}
</style>

<script>
$(function () {
$("#menu").menu();
});
</script>

</head>
<body>

<div style="position: absolute; top: 20px; left: 0px; z-index: 100">
<ul id="menu">
<li><a href="#">Dante</a>
<ul>
<li><a href="#11">Biografía</a></li>
<li><a href="#12">Narrativa</a></li>
<li><a href="#13">Obra poética</a></li>
</ul>
</li>
<li><a href="#">Shakespeare</a>
<ul>
<li><a href="#21">Biografía</a></li>
<li><a href="#22">Narrativa</a></li>
<li><a href="#23">Obras teatrales</a></li>
<li><a href="#">Varios</a>
<ul>
<li><a href="#241">Obras menores</a></li>
<li><a href="#242">Obras críticas</a></li>
</ul>
</li>
<li><a href="#25">Referencias</a></li>
</ul>
</li>
<li><a href="#">Balzac</a>
<ul>
<li><a href="#31">Biografía</a></li>
<li><a href="#32">Obra poética</a></li>
<li><a href="#33">Narrativa</a></li>
</ul>
</li>
<li><a href="#">Quevedo</a>
<ul>
<li><a href="#31">Biografía</a></li>
<li><a href="#32">Obra teatral</a></li>
<li><a href="#33">Obra poética</a></li>
</ul>
</li>
</ul>
</div>
```

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



Para cambiar todo el estilo por otro predefinido en el foro de *User Interface de jQuery* bastará descargar nuevamente el código y cambiar la hoja de estilo (y el directorio *images*) por la nueva.

Si lo que deseamos es retocar parcialmente un estilo predefinido, acudimos a modificar los estilos de la hoja de estilos –archivo CSS- descargado, bien con otro archivo CSS complementario y referenciado posteriormente en el archivo HTML –para respetar el orden de prelación de la cascada de estilos-, bien mediante estilos *inline* en el propio documento HTML.

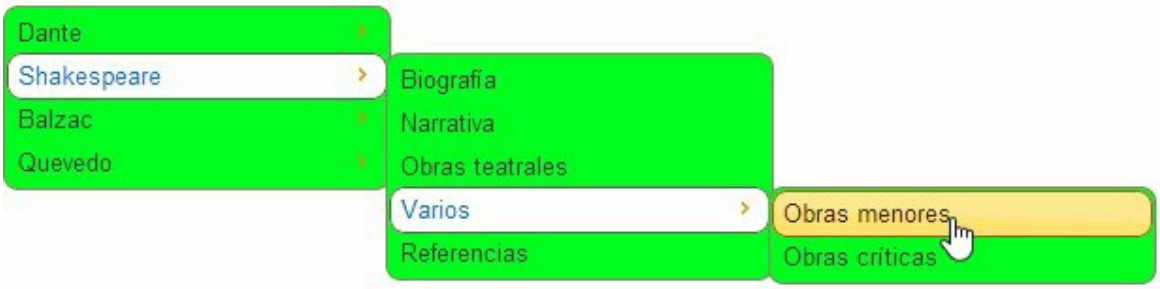
Fijémonos en el Listado 5; allí programamos un menú en el que alteraremos el tamaño de la fuente y su estilo, por ello estableceremos un apartado *style* que sobrecargará los estilos equivalentes de la hoja de estilos del *User Interface de jQuery* –archivo CSS referenciado al comienzo-.

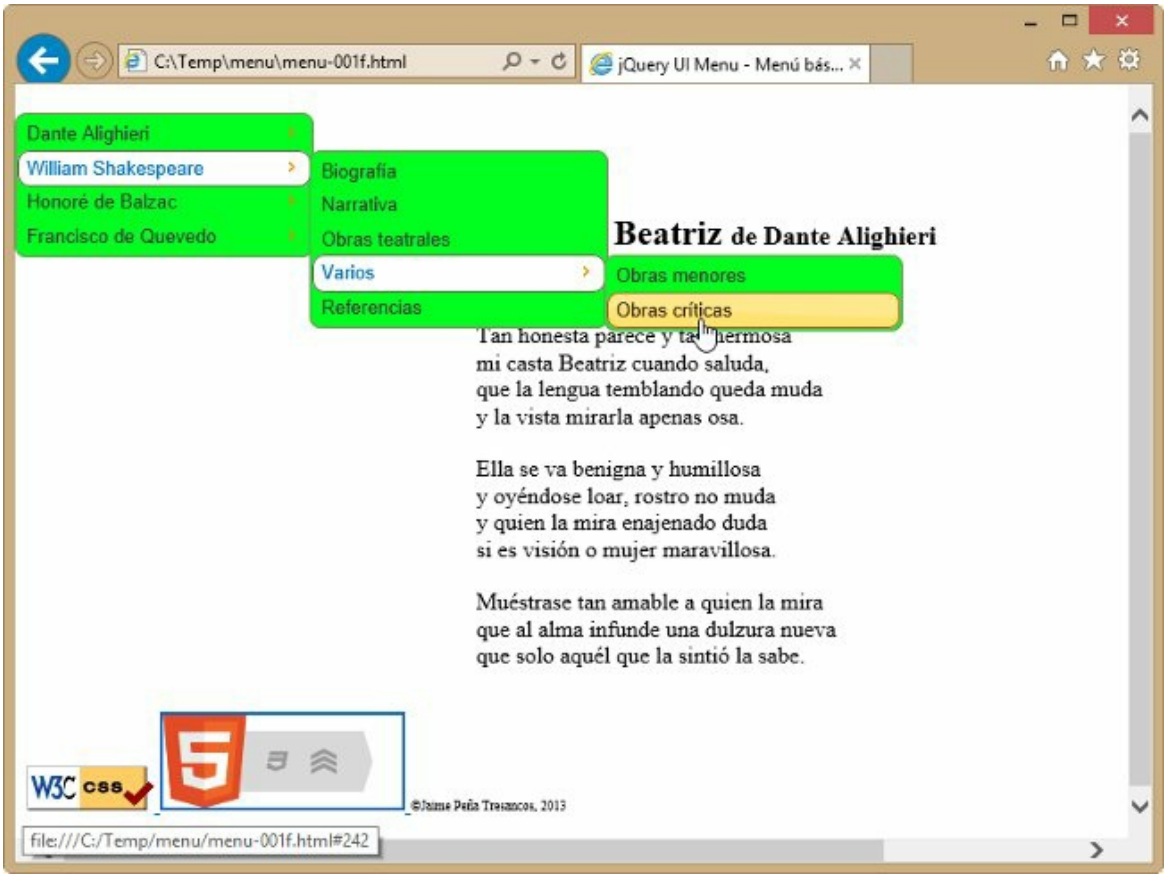
El aspecto final se observa en la figura adjunta, compárese el resultado con el menú original de estilo *Sunny*.

Listado 4: Un menú modificado, basado en un componente *Menu* en el que se ha utilizado el estilo *Sunny* y algunos estilos personalizados

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Menu - Menú básico con estilo personalizado y fuente y fondo personalizado</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>

<link rel="stylesheet" href="jquery-ui-1.10.2.sunny.css" />

<style>
.ui-menu {
width: 200px;
font: 13px Arial;
```

```
background: #00ff21;
}
</style>

<script>
$(function () {
$("#menu").menu();
});
</script>

</head>
<body>

...

</body>
</html>
```





Naturalmente, el retoque de estilos requiere un conocimiento importante del archivo fuente del *jQuery User Interface*, sino una buena dosis de paciencia y ensayo y error, pero que en ocasiones merecerá la pena para ir un paso más allá en nuestros trabajos.

Menú desactivado y reactivado

Otra opción que puede ser de interés es desactivar el componente *Menu*, de manera que deje de ser funcional, aunque siga estando presente.

El método a utilizar es *disabled* que, según su único parámetro, lo activará o desactivará, la sintaxis es:

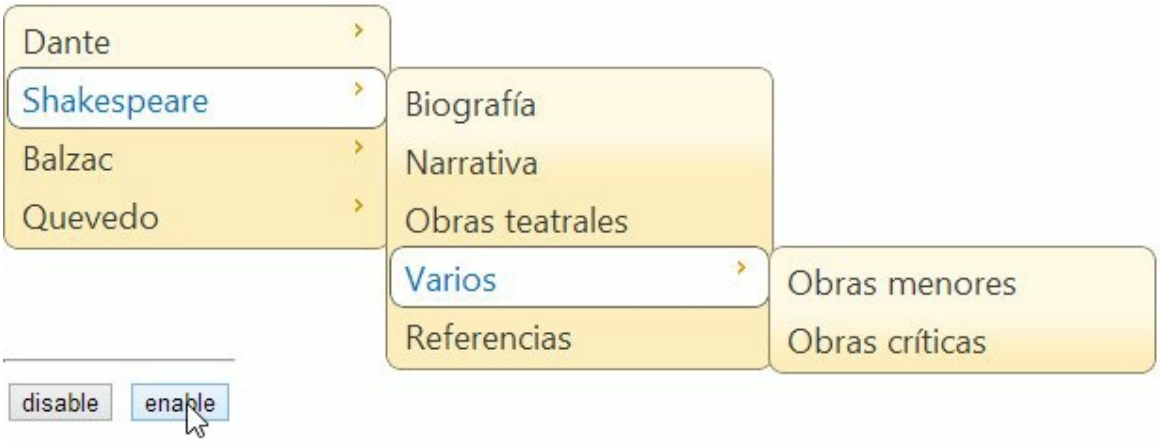
- $(referencia).menu("option", "disabled", true)
- $(referencia).menu("option", "disabled", false)

Veamos el listado que sigue, en el programamos un componente *Menu* y dos botones de comando que llaman a las funciones de desactivación y reactivación.

Listado 5: Componente *Menu* asociado a botones de comando para su desactivación y reactivación.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Menu - Menú básico con desactivación y reactivación</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>

<link rel="stylesheet" href="jquery-ui-1.10.2.sunny.css" />

<style>
.ui-menu {
width: 200px;
}
</style>

<script>
$(function () {
$("#menu").menu();
});

$(function () {
$("#disable").click(function () {
$("#menu").menu("option", "disabled", true);
});
});
$(function () {
$("#enable").click(function () {
$("#menu").menu("option", "disabled", false);
});
```

```
});
</script>

</head>
<body>
...
<div style="position: absolute; top: 200px; left: 0px">
<hr />
<input type="button" id="disable" value="disable" />
<input type="button" id="enable" value="enable" />
</div>
</body>
</html>
```



En la primera imágen vemos el menú desactivado, mientras que en la segunda aparecerá el menú reactivado.

Menú con orientación horizontal

Son varias las referencias encontradas en la Web para convertir un componente *Menu* en un menú horizontal, todas pasan por la sobrecarga de estilos.

En el Listado 6 hemos adaptado de forma fidedigna la versión recogida en la dirección URL: http://knowledgebase.bridge-delivery.com/jquery-ui-menu-horizontal-and-vertical/

Obsérvese que el punto clave es la inclusión de los estilos:

```
float:left;
```

También es crítico el modo de inicialización:

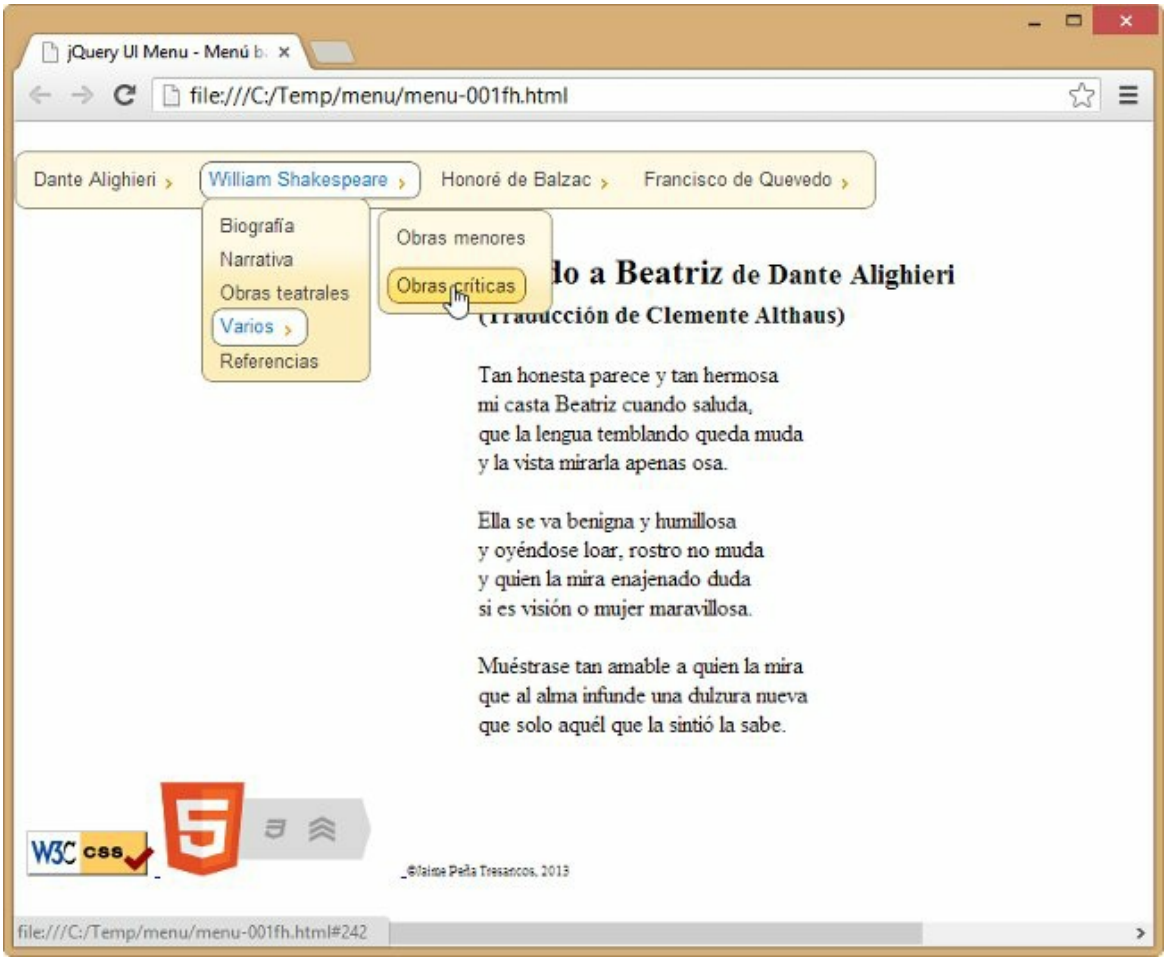
```
$(referencia).menu({position: {at: "left bottom"}});
```

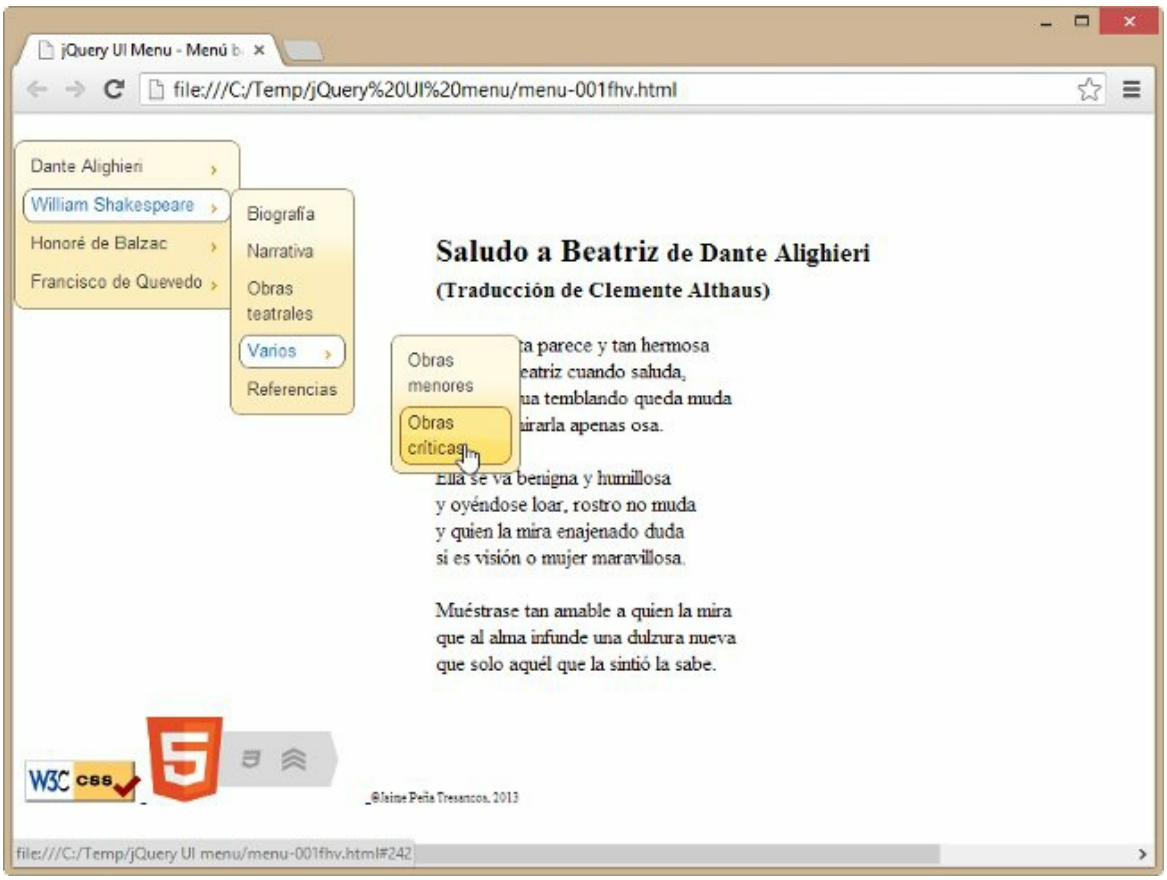
Si se eliminan los estilos *left* y si se inicializa como en los restantes ejemplos, tendríamos el menú vertical (;-)) –véanse las figuras adjuntas-. El resto de los estilos sobrecargados es complementario.

Listado 6: Adaptación de un componente *Menu* a una disposición de menú horizontal mediante la sobrecarga de estilos

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Menu - Menú básico con estilo personalizado y horizontal</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>

<link rel="stylesheet" href="jquery-ui-1.10.2.sunny.css" />

<style>
.ui-menu {
overflow: hidden;
font: 13px Arial;
}
.ui-menu .ui-menu {
overflow: visible !important;
}
.ui-menu > li {
float: left;
display: block;
width: auto !important;
}
.ui-menu ul li {
display:block;
float:none;
}
.ui-menu ul li ul {
left:120px !important;
width:100%;
}
.ui-menu ul li ul li {
width:auto;
}
```

```
.ui-menu ul li ul li a {
float:left;
}
.ui-menu > li {
margin: 5px 5px !important;
padding: 0 0 !important;
}
.ui-menu > li > a {
float: left;
display: block;
clear: both;
overflow: hidden;
}
.ui-menu .ui-menu-icon {
margin-top: 0.3em !important;
}
.ui-menu .ui-menu .ui-menu li {
float: left;
display: block;
}
</style>

<script>
$(function () {
$("#menu").menu({position: {at: "left bottom"}});;
});

</script>

</head>
<body>
...
</body>
</html>
```


jQuery UI Menu - Menú b
file:///C:/Temp/menu/menu-001fh.html
Dante Alighieri
William Shakespeare
Honoré de Balzac
Francisco de Quevedo
Biografía
Narrativa
Obras teatrales
Varios
Referencias
Obras menores
Obras críticas
Tan honesta parece y tan hermosa
mi casta Beatriz cuando saluda,
que la lengua temblando queda muda
y la vista mirarla apenas osa.
Ella se va benigna y humillosa
y oyéndose loar, rostro no muda
y quien la mira enajenado duda
si es visión o mujer maravillosa.
Muéstrase tan amable a quien la mira
que al alma infunde una dulzura nueva
que solo aquél que la sintió la sabe.
©Jaime Peña Tresancos, 2013
file:///C:/Temp/menu/menu-001fh.html#242


jQuery UI Menu - Menú b
file:///C:/Temp/jQuery%20UI%20menu/menu-001fhv.html
Dante Alighieri
William Shakespeare
Honoré de Balzac
Francisco de Quevedo
Biografía
Narrativa
Obras teatrales
Varios
Referencias
Obras menores
Obras críticas
Saludo a Beatriz de Dante Alighieri
(Traducción de Clemente Althaus)
Ella se va benigna y humillosa
y oyéndose loar, rostro no muda
y quien la mira enajenado duda
si es visión o mujer maravillosa.
Muéstrase tan amable a quien la mira
que al alma infunde una dulzura nueva
que solo aquél que la sintió la sabe.
©Jaime Peña Tresancos, 2013
file:///C:/Temp/jQuery UI menu/menu-001fhv.html#242

Aquí veremos primero el menú horizontal y en segundo lugar el menú vertical

Conclusiones

En el presente artículo hemos repasado las diversas capacidades del componente Menu del *jQuery User Interface*. También disponemos de ejemplos detallados de cómo utilizarlas en nuestras aplicaciones.

Esperamos que todo lo expuesto les haya servido de ayuda. Hasta nuestro próximo artículo, felices horas de programación.

Este artículo es obra de *Jaime Peña Tresancos*
Fue publicado por primera vez en *25/06/2013*
Disponible online en *http://desarrolloweb.com/articulos/jquery-user-interfaz-menu.html*

jQuery UI Dialog. Manual de uso

En el presente artículo explicaremos qué es y qué proporciona el jQuery User Interface Dialog.

En el presente artículo hablaremos de:

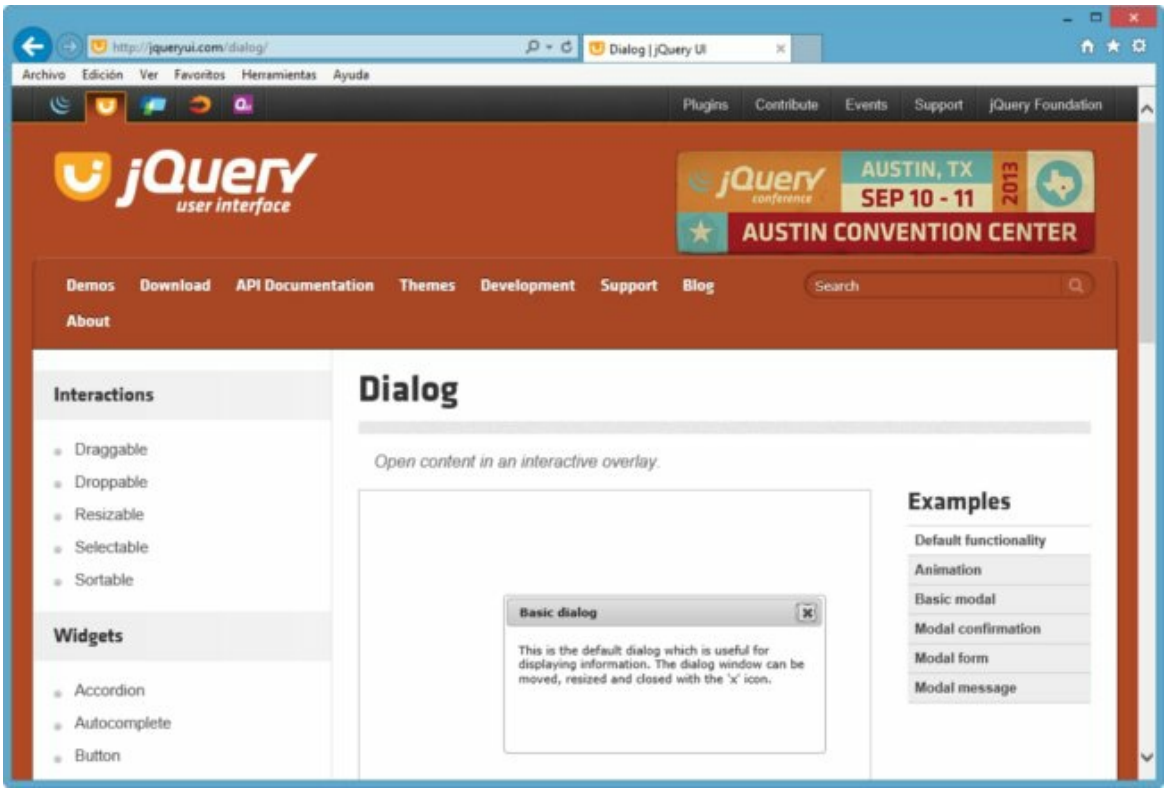
- Qué es y qué proporciona el *jQuery User Interface Dialog*
- Los componentes esenciales para su implementación en una página web
- Apertura de un diálogo bajo demanda
- Implementación de diálogos *modales y amodales*
- Aplicación de estilos personalizados
- Inclusión de botones de comando de cierre y demás
- Cómo fijar sus dimensiones personalizadas
- Uso del diálogo como un formulario simple
- Efectos visuales durante su presentación y eliminación

¿Qué es el *jQuery User Interface Dialog*?

Dentro del *jQuery User Interface* encontramos toda una serie de elementos de programación de la interfaz de usuario preprogramados y listos para ser integrados en nuestros proyectos

HTML. Se trata de una amplia biblioteca *JavaScript* que abarca desde efectos dinámicos, hasta menús, calendarios, diálogos, etc.

El lugar de Internet de referencia y descarga del componente *Dialog*, que es de lo que trataremos, es: jqueryui.com/dialog



Las capacidades que nos proporcionará las iremos desgranando en los sucesivos epígrafes, a vuelapluma son:

- Una caja de diálogo básica amodal redimensionable
- La posibilidad de convertirla en modal y con apertura a demanda
- La adaptación personal a estilos predefinidos o más personalizados
- La conversión completa a una caja de entrada de usuario, es decir, a un formulario

Un primer diálogo básico

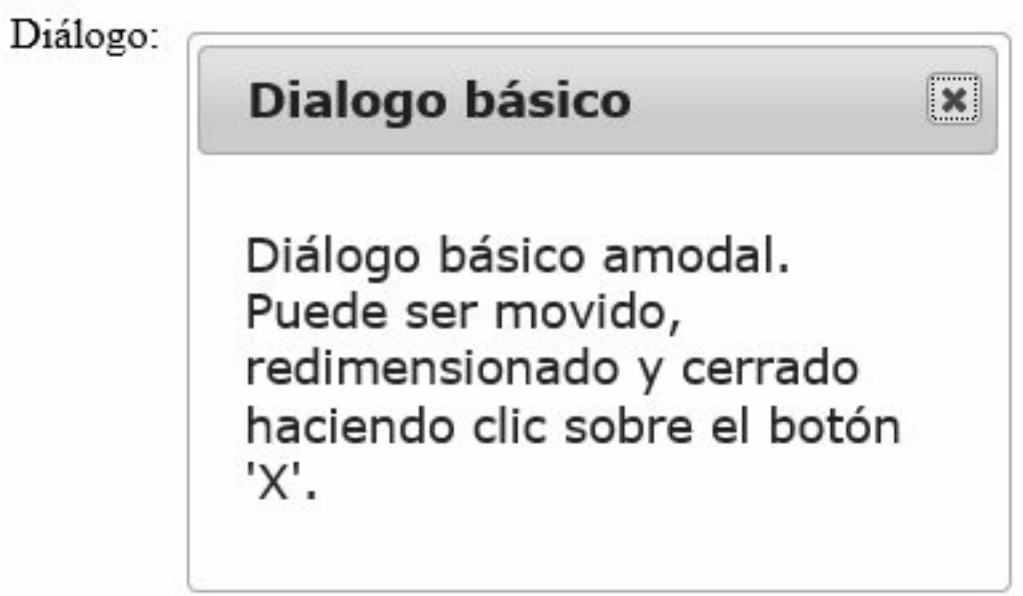
Nuestro primer ejemplo no será muy ambicioso, se trata de ilustrar un simple diálogo que se abrirá al ser cargada la página, con un mensaje en su interior y ninguna funcionalidad adicional.

El código es el del Listado 1 y se comentará seguidamente.

######### Listado 1: Código de un diálogo simple

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
```

```
<title>jQuery UI Dialog - Uso básico</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script>
$(function () {
$("#dialog").dialog();
});
</script>
</head>

<body>
Diálogo:
<div id="dialog" title="Dialogo básico">
<p>Diálogo básico amodal. Puede ser movido, redimensionado y cerrado haciendo clic sobre el botón 'X'.</p>
</div>
</body>
</html>
```



Si repasamos el código nos encontramos, por orden secuencial:

- Al comienzo del documento, en la etiqueta `<head>`, insertaremos las bibliotecas CSS y *jQuery* y los códigos correspondientes
- La referencia a la hoja de estilos del *User Interface de jQuery:jquery-ui.css* Debe ser la primera de las bibliotecas CSS referenciadas
- La referencia a la biblioteca *jQuery* general: jquery-x.x.x.js Debe ser la primera de las bibliotecas JavaScript referenciadas
- La referencia a la biblioteca *User Interface de jQuery: jquery-ui.js*

- Una función de inicialización del *Dialog*:

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

Que, como vemos, toma el elemento al que va asociado y la función sin parámetros, sin más

- En el cuerpo del documento, etiqueta `<body>` irá el resto de elementos HTML, entre ellos la definición del diálogo:

```
<div id="dialog" title="Dialogo básico">
```

Diálogos modales y amodales. Apertura a demanda

La presentación de diálogos puede ser de dos tipos:

- Modal: Capturará el foco de entrada, de manera que deberemos atender a lo requerido en el diálogo antes de poder acceder a otros elementos de la ventana principal. Normalmente es el comportamiento que desearemos.
- Amodal: No capturará el foco de entrada en modo exclusivo, de forma que podremos acceder a, por ejemplo, la ventana principal y desatender lo requerido en el diálogo. Adviértase que, sorprendentemente, es la opción por defecto en los componentes *Dialog del User Interface de jQuery*.

En el Listado 2 vemos que para pasar el diálogo a modal, sin más, hemos de especificar dicha propiedad en su construcción:

```
modal: true;
```

Por otra parte, para abrir un diálogo a demanda, que será lo que habitualmente deseemos, hemos de especificar:

- Que no se abra automáticamente, así pondremos su propiedad:

```
autoOpen: false
```

- Programaremos, por ejemplo, un botón de comando para su apertura en el código HTML:

```
<button id="abrir">Abrir diálogo</button>
```

- Asociaremos el botón:

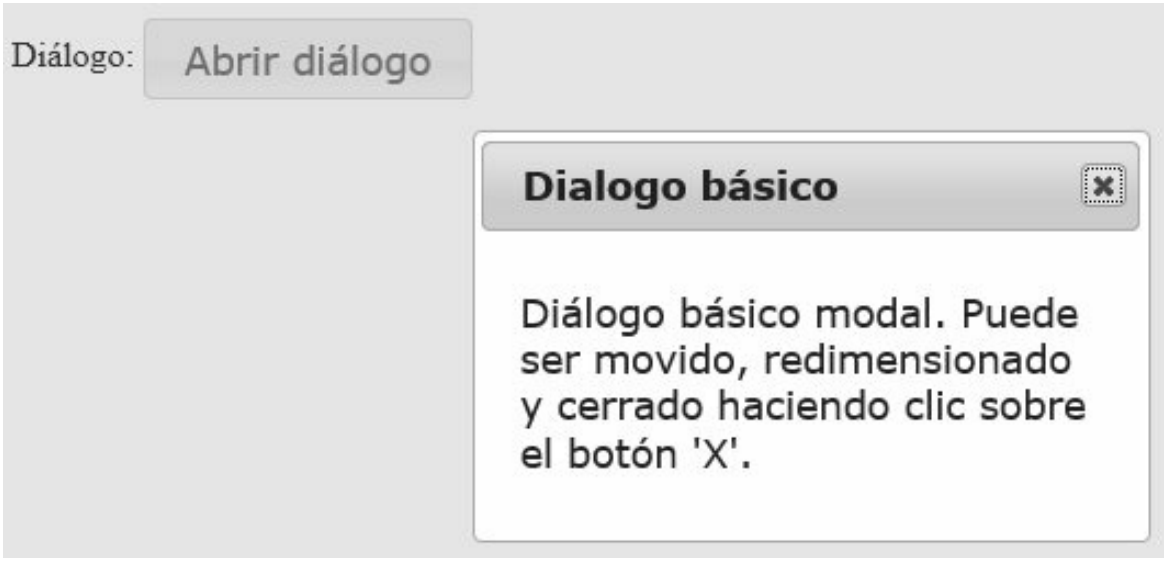
```
$("#abrir")
```

- Programaremos su apertura al hacer clic sobre dicho botón:

```
.click(function () { $("#dialog").dialog("open"); });
```

######### Listado 2: Código de un diálogo modal simple

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Dialog - Dialogo modal</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script>
$(function () {
$("#dialog").dialog({
autoOpen: false,
modal: true
});
$("#abrir")
.button()
.click(function () {
$("#dialog").dialog("open");
});
});
</script>
</head>

<body>
Diálogo:
<div id="dialog" title="Dialogo básico">
<p>Diálogo básico modal. Puede ser movido, redimensionado y cerrado haciendo clic sobre el botón 'X'.</p>
</div>
<button id="abrir">Abrir diálogo</button>
</body>

</html>
```



Añadir un botón de cierre

Aunque el cierre de un diálogo se puede siempre realizar haciendo clic sobre el botón 'X' de cierre genérico de una ventana, más elegante es mostrar un botón para tal función.

Para ello utilizamos dos funcionalidades, añadir botones y cerrar ventanas mediante código *JavaScript:*

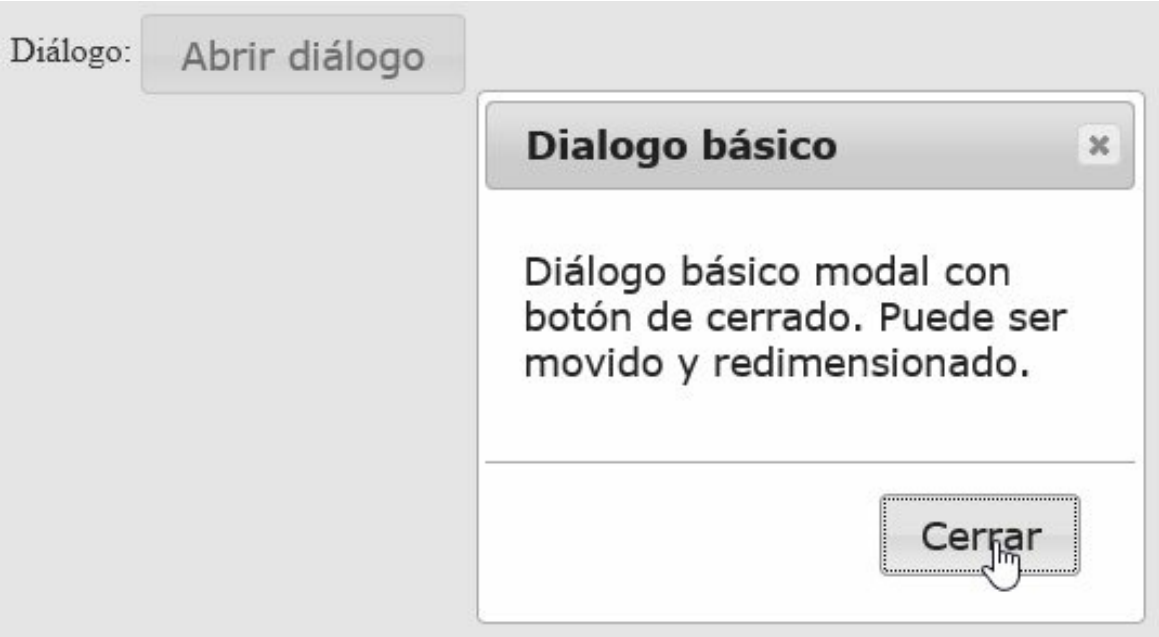
- Mediante la propiedad *buttons* creamos un botón con una determinada etiqueta y una función de respuesta al ser pulsado. Podríamos crear así varios, como veremos en el apartado **Diálogo para formulario simple**, al final del artículo.
- Para cerrar la ventana, hacemos una llamada a la función *close*, con el código:

```
$(this).dialog("close");
```

######### Listado 3: Un diálogo simple con un botón de comando para su cierre

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Dialog - Dialogo modal con botón de cerrado</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script>
$(function () {
$("#dialog").dialog({
autoOpen: false,
modal: true,
buttons: {
"Cerrar": function () {
```

```
$(this).dialog("close");
}
}
});
$("#abrir")
.button()
.click(function () {
$("#dialog").dialog("open");
});
});
</script>
</head>

<body>
Diálogo:
<div id="dialog" title="Dialogo básico">
<p>Diálogo básico modal con botón de cerrado. Puede ser movido y redimensionado.</p>
</div>
<button id="abrir">Abrir diálogo</button>
</body>

</html>
```



Diálogo con dimensiones personalizadas

Hasta el momento, los diálogos mostrados lo son con dimensiones predefinidas por el sistema, veremos cómo definir nosotros mismos sus dimensiones.

Hemos optado por una modificación al vuelo, en el momento de crear el diálogo, no en su definición, previo a su apertura, como puede apreciarse en el Listado 4.

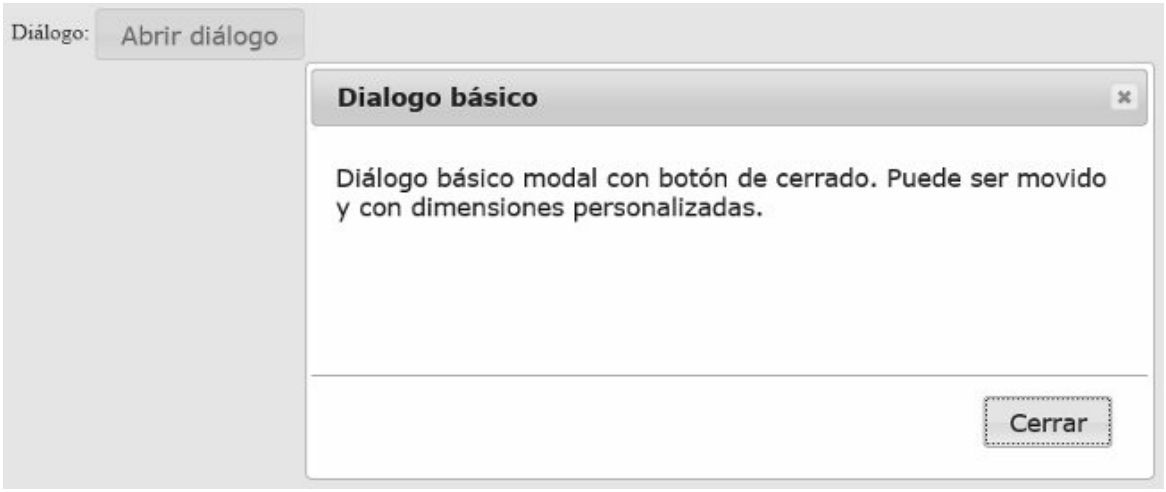
Las propiedades a fijar son *width y height*, como se muestra en la porción de código que sigue:

```
.click(function () {
$("#dialog").dialog("option", "width", 600);
$("#dialog").dialog("option", "height", 300);
$("#dialog").dialog("open");
});
```

En el Listado 4 se muestra el ejemplo completo de un diálogo modal, con dimensiones fijas y un botón para su cierre.

######### Listado 4: Diálogo modal simple, con botón de cierre y dimensiones fijas

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Dialog - Dialogo modal con dimensiones personalizadas</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>
<script>
$(function () {
$("#dialog").dialog({
autoOpen: false,
modal: true,
buttons: {
"Cerrar": function () {
$(this).dialog("close");
}
}
});
$("#abrir")
.button()
.click(function () {
$("#dialog").dialog("option", "width", 600);
$("#dialog").dialog("option", "height", 300);
$("#dialog").dialog("open");
});
});
</script>
</head>

<body>
Diálogo:
<div id="dialog" title="Dialogo básico">
<p>Diálogo básico modal con botón de cerrado. Puede ser movido y con dimensiones personalizadas.</p>
</div>
<button id="abrir">Abrir diálogo</button>
</body>

</html>
```



Para hacerlo no redimensionable, de dimensiones estáticas, bastará poner a *false* la propiedad *resizable*, como por ejemplo en la porción de código que sigue:

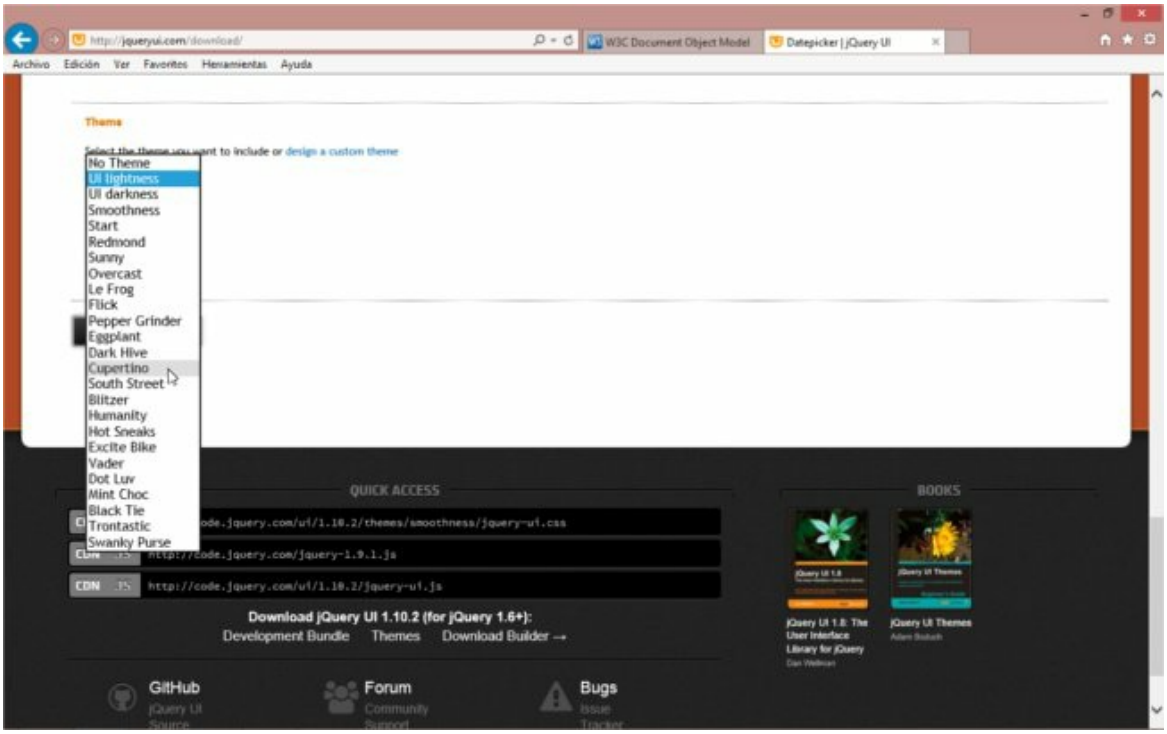
```
.click(function () {
$("#dialog").dialog("option", "width", 600);
$("#dialog").dialog("option", "height", 300);
$("#dialog").dialog("option", "resizable", false);
$("#dialog").dialog("open");
});
```

Jugando con los estilos

Hasta ahora hemos trabajado con referencias al *User Interface* alojado en el foro de *jQuery;* de esa manera disponemos de un modo muy cómodo y relativamente eficiente de crear los *Dialog*, pero no de personalizar sus estilos, dado que se trabaja con uno predeterminado –una CSS referenciada concreta, dada por el administrador del *jQuery User Interface*-.

Sin embargo el propio *jQuery User Interface* dispone de numerosos estilos prediseñados y listos para ser utilizados a demanda; pero para ello deberemos descargarlos y trabajar con ellos y referenciarlos en local –en el propio servidor-.

Podremos descargar el *jQuery User Interface* completo, que incluye Dialog, la dirección URL es: jqueryui.com/download



En ese proceso de descarga, en la parte inferior nos encontraremos con el apartado **Theme**, desplegando la lista podemos seleccionar el estilo del *Dialog* que se descargará. En realidad, según lo que hayamos seleccionado previamente, es el estilo –el tema- de los componentes del *jQuery User Interface* que descargaremos.

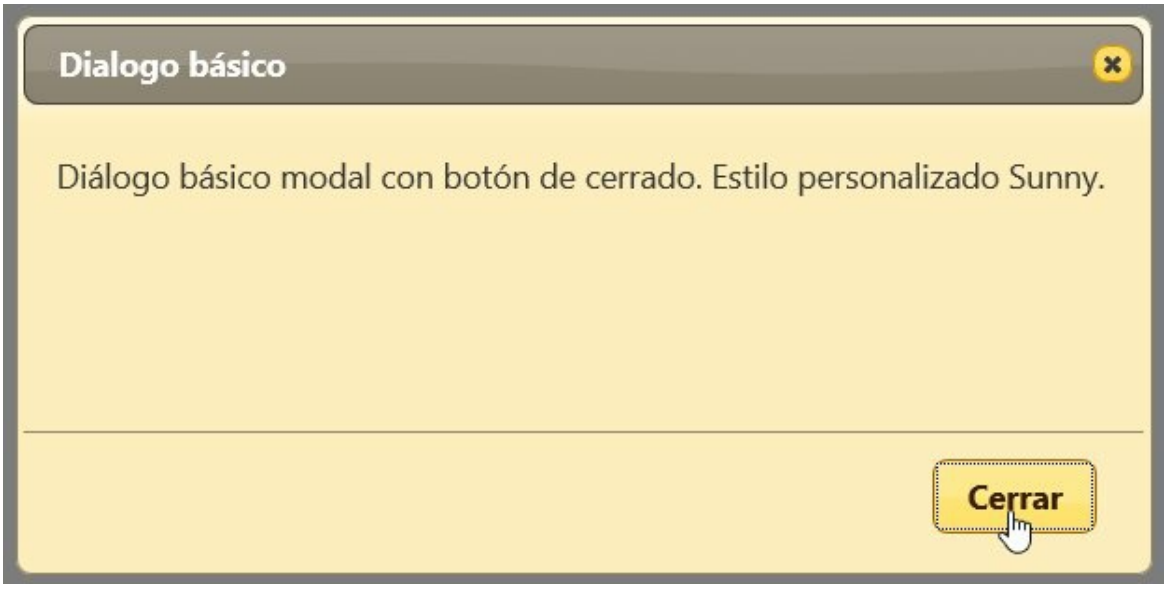
De ahí es de donde se extraen los correspondientes CSS y directorios *images* particulares.

> **Nota:** En nuestro ejemplo hemos renombrado el archivo CSS descargado con el fin de poder trabajar con diferentes estilos, añadiéndole el nombre de estilo y cargándolo en local –véase la referencia correspondiente en el listado-. Trabajando en local, también deberemos copiar en cada caso el correspondiente directorio *images* y si queremos poder variar los estilos, darle un nombre particular para cada uno de los estilos –por ejemplo _images*sunny*- y cambiar las referencias en el archivo CSS correspondiente.

######### Listado 5: Diálogo modal con un estilo personalizado *Sunny*

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Dialog - Dialogo modal con estilo personalizado</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>

<link rel="stylesheet" href="jquery-ui-1.10.2.sunny.css" />

<script>
```

```
$(function () {
$("#dialog").dialog({
autoOpen: false,
modal: true,
buttons: {
"Cerrar": function () {
$(this).dialog("close");
}
}
});
$("#abrir")
.button()
.click(function () {
$("#dialog").dialog("option", "width", 600);
$("#dialog").dialog("option", "height", 300);
$("#dialog").dialog("option", "resizable", false);
$("#dialog").dialog("open");
});
});
</script>
</head>

<body>
Diálogo:
<div id="dialog" title="Dialogo básico">
<p>Diálogo básico modal con botón de cerrado. Estilo personalizado Sunny.</p>
</div>

<button id="abrir">Abrir diálogo</button>
</body>

</html>
```



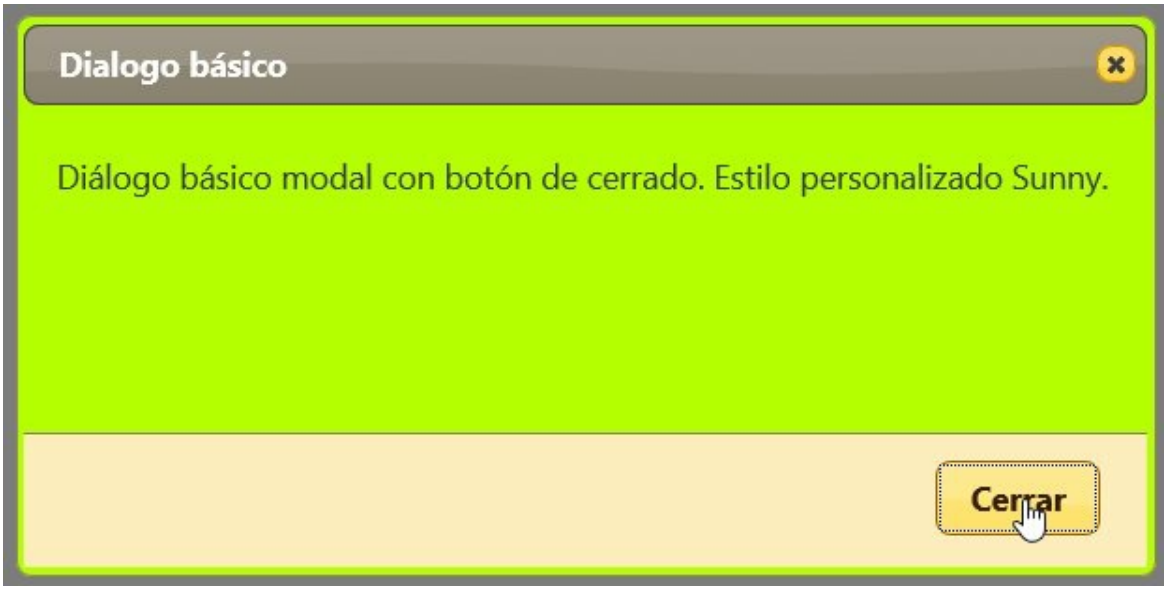
Para cambiar todo el estilo por otro predefinido en el foro de *User Interface de jQuery* bastará descargar nuevamente el código y cambiar la hoja de estilo (y el directorio *images*) por la nueva.

Si lo que deseamos es retocar parcialmente un estilo predefinido, acudimos a modificar los estilos de la hoja de estilos –archivo CSS- descargado, bien con otro archivo CSS complementario y referenciado posteriormente en el archivo HTML –para respetar el orden de prelación de la cascada de estilos-, bien mediante estilos *inline* en el propio documento HTML.

Fijémonos en el pequeño código de estilo en el que alteraremos el color de fondo de la caja de diálogo, lo que hacemos es establecer un apartado *style* que sobrecargará los estilos equivalentes de la hoja de estilos del *User Interface de jQuery* –archivo CSS referenciado al comienzo-.

El aspecto final se observa en la figura adjunta, compárese el resultado con el diálogo original de estilo *Sunny*.

```
<style>
.ui-dialog {
background: #b6ff00;
}
</style>
```



Dialogo para formulario simple

Una de las aplicaciones más interesantes de los diálogos es su uso como formulario de entrada.

En nuestro ejemplo final ilustraremos una pequeña aplicación en la que tendremos dos datos a mostrar en la pantalla principal de la aplicación: un nombre y una fecha. Mediante un botón de comando llamaremos a un formulario de entrada de datos que recogerá dichos valores en una entrada de texto editable y en un componente "Datepicker" para selección de fechas, respectivamente.

Para comenzar, codificaremos la entrada de textos para el caso del nombre, como se recoge en el Listado 6. Los puntos esenciales son:

- Todo el diseño del diálogo en sí, sus componentes, se realiza en la etiqueta `<div>` correspondiente:

```
<div id="dialog" title="Dialogo básico">
```

- En el diálogo se programa la caja de textos de entrada de usuario:

```
Nombre:<input type="text" id="el_nombre" value="" />
```

- Al abrir el diálogo, se toma el valor del nombre de la ventana principal y se le pasa al componente de entrada de textos del diálogo:

```
el_nombre.value = nombre.value;
```

- Obsérvese cómo se programan los botones de comando dentro del componente *Dialog*. Bajo la entrada de la propiedad *buttons* se van dando los nombres y las funciones de respuesta de cada uno de ellos. En caso de pulsar el botón **Aceptar** se toma el valor de la entrada de textos y se pone en el lugar correspondiente de la ventana principal:

```
nombre.value = el_nombre.value;
```

- En caso de pulsar el botón **Cerrar** no se recogerá nada de lo introducido en el diálogo.
- Con la pulsación de cualesquiera de ambos botones, se cerrará el diálogo:

```
$(this).dialog("close");
```

######### Listado 6: Aplicación con una caja de diálogo para recogida de entrada de textos

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Dialog - Dialogo para formulario simple</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>

<link rel="stylesheet" href="jquery-ui-1.10.2.sunny.css" />

<script>
$(function () {
$("#dialog").dialog({
```

```
autoOpen: false,
modal: true,
buttons: {
"Aceptar": function () {
nombre.value = el_nombre.value;
$(this).dialog("close");
},
"Cerrar": function () {
$(this).dialog("close");
}
}
});
$("#abrir")
.button()
.click(function () {
el_nombre.value = nombre.value;
$("#dialog").dialog("option", "width", 600);
$("#dialog").dialog("option", "height", 300);
$("#dialog").dialog("option", "resizable", false);
$("#dialog").dialog("open");
});
});
</script>
</head>

<body>
Diálogo:
<div id="dialog" title="Dialogo básico">
Nombre:<input type="text" id="el_nombre" value="" />
</div>

<input type="text" id="nombre" value="" />
<button id="abrir">Abrir diálogo</button>
</body>
</html>
```

En el Listado 7 extendemos el ejemplo para contemplar la entrada de fechas, con el componente *Datepicker*. Para conocer más acerca del componente *jQuery UI Datepicker* puede acudir a nuestros anteriores artículos:

- jQuery UI Datepicker. Manual de uso simple
- jQuery UI Datepicker. Manual de uso avanzado

######### Listado 7: Ejemplo de aplicación con un diálogo para entrada de textos y fechas, mediante el componente jQuery UI Datepicker

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>jQuery UI Dialog - Dialogo para formulario simple</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>

<script src="jquery.ui.datepicker-es.js"></script>

<link rel="stylesheet" href="jquery-ui-1.10.2.sunny.css" />

<script>
$(function () {
$(function () {
$.datepicker.setDefaults($.datepicker.regional["es"]);
$("#datepicker").datepicker({
firstDay: 1
});
});
$("#dialog").dialog({
autoOpen: false,
modal: true,
buttons: {
```

```
"Aceptar": function () {
var texto1 = document.getElementById('nombre').firstChild;
var newText1 = document.createTextNode(el_nombre.value);
texto1.parentNode.replaceChild(newText1, texto1);
var texto2 = document.getElementById('fecha').firstChild;
var newText2 = document.createTextNode(datepicker.value);
texto2.parentNode.replaceChild(newText2, texto2);
$(this).dialog("close");
},
"Cerrar": function () {
$(this).dialog("close");
}
}
});
$("#abrir")
.button()
.click(function () {
el_nombre.value = document.getElementById('nombre').firstChild.nodeValue;
datepicker.value = document.getElementById('fecha').firstChild.nodeValue;
$("#dialog").dialog("option", "width", 600);
$("#dialog").dialog("option", "height", 300);
$("#dialog").dialog("option", "resizable", false);
$("#dialog").dialog("open");
});
});
</script>
</head>

<body>
<div id="dialog" title="Formulario simple">
Nombre:<input type="text" id="el_nombre" value="" /><br />
Fecha:<input type="text" id="datepicker" />
</div>

Nombre: <strong id="nombre">Julio</strong><br />
Fecha: <strong id="fecha">11/07/1975</strong><br />
<hr />
<button id="abrir">Abrir diálogo</button>
</body>
</html>
```

Efectos *show y hide*

Por último tratamos los posibles efectos visuales durante su presentación y eliminación.

Se llevan a cabo mediante las propiedades *show y hide*, que toman dos valores:

- El nombre del efecto –prediseñado-
- El tiempo de duración en milisegundos

En el Listado 8 se puede ver cómo implementarlas en el código de creación del diálogo. El resto es lo mismo que el Listado 7.

Para una lista de todos los posibles efectos y su significado consúltese la dirección URL: api.jqueryui.com/category/effects

######### Listado 8: Implementación de efectos **show** y **hide**

```
<script>
$(function () {
$(function () {
$.datepicker.setDefaults($.datepicker.regional["es"]);
$("#datepicker").datepicker({
firstDay: 1
});
});
$("#dialog").dialog({
autoOpen: false,
modal: true,
show: {
effect: "blind",
duration: 1000
```

```
},
hide: {
effect: "explode",
duration: 1000
},
buttons: {
"Aceptar": function () {
var texto1 = document.getElementById('nombre').firstChild;
var newText1 = document.createTextNode(el_nombre.value);
texto1.parentNode.replaceChild(newText1, texto1);
var texto2 = document.getElementById('fecha').firstChild;
var newText2 = document.createTextNode(datepicker.value);
texto2.parentNode.replaceChild(newText2, texto2);
$(this).dialog("close");
},
"Cerrar": function () {
$(this).dialog("close");
}
}
});
$("#abrir")
.button()
.click(function () {
el_nombre.value = document.getElementById('nombre').firstChild.nodeValue;
datepicker.value = document.getElementById('fecha').firstChild.nodeValue;
$("#dialog").dialog("option", "width", 600);
$("#dialog").dialog("option", "height", 300);
$("#dialog").dialog("option", "resizable", false);
$("#dialog").dialog("open");
});
});
</script>
```

Conclusiones

En el presente artículo hemos repasado muchas de las funcionalidades del componente *jQuery User Interface Dialog*. Con ellas creemos se está en condiciones de implementar elegantes diálogos y formularios en sus páginas web con un mínimo de codificación.

Esperamos que todo lo expuesto les haya servido de ayuda. Hasta nuestro próximo artículo, felices horas de programación.

Este artículo es obra de *Jaime Peña Tresancos*
Fue publicado por primera vez en *09/08/2013*
Disponible online en *http://desarrolloweb.com/articulos/jquery-ui-dialog.html*