








Tema 5. Plugins jQuery (jQuery UI)
Alejandro Amat Reina y Pablo Matías Garramone Ramírez

ÍNDICE

| | |
|--|-----------|
| 1. OBJETIVOS | 2 |
| 2. PLUG-INS | 3 |
| CÓMO USAR UN PLUGIN | 3 |
| 3. LA LIBRERÍA JQUERY UI | 4 |
| INCLUIR LA LIBRERÍA EN NUESTRA PÁGINA | 4 |
| TEMAS CSS | 5 |
| INCLUIR EL TEMA EN NUESTRA PÁGINA | 6 |
| 4. EFECTOS EN JQUERY UI | 7 |
| NUEVOS EFECTOS VISUALES | 7 |
| MEJORAS EN EL MÉTODO .ANIMATE () | 8 |
| 5. UTILIZAR WIDGETS DE JQUERY UI | 10 |
| BOTONES DE JQUERY UI (WIDGET BUTTON) | 10 |
| CAMBIAR LAS PROPIEDADES DE UN BOTÓN | 12 |
| MANEJADORES DE EVENTOS PARA LOS BOTONES | 12 |
| CUADROS DE DIÁLOGO DE JQUERY UI (DIALOG) | 13 |
| CAMBIAR EL ASPECTO DE LOS DIÁLOGOS | 14 |
| PERSONALIZACIÓN DE LOS DIÁLOGOS | 15 |
| CAMBIAR LAS PROPIEDADES DE UN DIÁLOGO | 17 |
| TOOLTIPS DE JQUERY UI (TOOLTIP) | 17 |
| 6. INTERACCIONES EN JQUERY UI (DRAG-AND-DROP) | 19 |
| EL MÉTODO .DRAGGABLE() | 19 |
| EVENTOS QUE SE PRODUCEN AL ARRASTRAR LOS ELEMENTOS | 20 |
| ACCIONES DEL MÉTODO .DRAGGABLE () | 21 |
| EL MÉTODO .DROPPABLE () | 22 |
| EVENTOS QUE SE PRODUCEN EN LOS ELEMENTOS DE DESTINO | 22 |
| ACCIONES DEL MÉTODO .DROPPABLE () | 23 |
| 7. EJERCICIOS | 26 |
| EJERCICIO 1 | 26 |
| EJERCICIO 2 | 26 |
| EJERCICIO 3 | 26 |
| EJERCICIO 4 | 29 |
| EJERCICIO 5 | 29 |
| EJERCICIO 6 | 29 |
| EJERCICIO 7 | 30 |
| 8. ÍNDICE DE EJEMPLOS Y TABLAS | 31 |
| EJEMPLOS | 31 |
| TABLAS | 31 |

1. Objetivos

En este cuarto tema se pretenden conseguir los siguientes objetivos:

-  Conocer la existencia de plugins que amplían las capacidades de jQuery.
-  Incluir el plugin jQuery UI en nuestra aplicación web.
-  Aplicar temas de jQuery UI a nuestra aplicación web.
-  Utilizar los widgets de jQuery UI.
-  Utilizar efectos de jQuery UI.

2. Plug-ins

A lo largo de las cuatro sesiones anteriores hemos examinado los componentes que forman el núcleo de jQuery (a excepción de la parte relacionada con las peticiones Ajax que veremos en el siguiente tema). Haciendo esto, hemos ilustrado muchas de las formas en las que la librería nos ayuda a realizar gran cantidad de tareas. Sin embargo, tan potente como la librería en sí misma, es su elegante arquitectura de plugins. Esta permite a los desarrolladores ampliar las capacidades de jQuery, con lo que se convierte en una librería con muchas más posibilidades.

La creciente comunidad de jQuery ha creado cientos de plugins, desde pequeños selectores hasta web widgets¹ que pueden ser utilizados a gran escala.

En la página web de jQuery podemos encontrar un gran repositorio de plugins disponibles en la url <http://plugins.jquery.com/>. Además, en muchos de ellos podemos encontrar enlaces a tutoriales de uso, código fuente, ejemplos, etc.

Cómo usar un plugin

El uso de un plugin jQuery es muy sencillo. Simplemente, necesitamos obtener el código del plugin, vincularlo a nuestra página html e invocar a las nuevas funcionalidades desde nuestros scripts.

Dado que los plugins pueden utilizar la propia librería jQuery, siempre los incluiremos después de esta.

En este tema, veremos el plugin jQuery UI por su relevancia dentro de los plugins jQuery y por la cantidad de mejoras a la interfaz de usuario que nos aporta.

¹ Componente gráfico genérico que puede ser incorporado a una página web para enriquecer su interfaz de usuario.

3. La librería jQuery UI

Mientras que otros plugins suelen realizar una única tarea, jQuery UI nos aporta una gran cantidad de nuevas funcionalidades. De hecho, podríamos hablar de la librería jQuery UI más que del plugin jQuery UI, ya que recopila una gran cantidad de plugins relacionados con la interfaz de usuario.

El equipo de jQuery UI ha creado una serie de componentes de interacción y widgets de interfaz de usuario que nos ayudarán a crear aplicaciones Web como si se trataran de aplicaciones de escritorio.

Estos componentes incluyen métodos para arrastrar y soltar (drag & drop), ordenar, seleccionar y cambiar el tamaño de los elementos, etc. Además, encontramos widgets como botones, diálogos, tooltips, componentes de fecha, barras de progreso, pestañas, etc.

La librería es demasiado extensa como para poder estudiarla por completo en un solo tema, de hecho haría falta un curso entero dedicado a este tema, pero, afortunadamente, una de las principales características del proyecto es la coherencia en el desarrollo de los componentes, de forma que todos funcionan de manera parecida, así que veremos algunos ejemplos de uso y esto nos servirá como base para poder utilizar el resto de la librería cuando sea necesario.

Incluir la librería en nuestra página

Para descargar la librería nos dirigiremos a la página <http://jqueryui.com>, dentro de esta, pulsaremos el enlace *stable* que nos descargará un zip con la última versión estable de la librería. Dentro de este, podremos encontrar: los fuentes, la documentación y los ejemplos de la librería. En concreto, dentro del directorio *ui* encontraremos los fuentes divididos en diferentes ficheros, por si queremos utilizar sólo alguna cosa concreta de la librería. Si lo que queremos es incluir la librería completa, cogeremos el fichero `jquery-ui.js`. Para la versión de producción tenemos una versión compactada dentro del directorio *minified*. Para utilizar esta versión compactada cogeremos el fichero `jquery-ui.min.js`.

Al igual que ocurre con jQuery, para incluir la librería en nuestra página, podemos elegir entre dos posibilidades:

- ✚ Hacer una copia del fichero de la librería en un directorio de nuestro sitio web y vincularlo con nuestros html.

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

- Incluir la librería por medio de un CDN, por ejemplo el de jQuery:

```
<script src="http://code.jquery.com/ui/1.12.0/jquery-ui.min.js" type="text/javascript"> </script>
```

Temas CSS

Una de las ventajas que nos ofrece la librería, son los temas css. Estos nos van a permitir cambiar la apariencia de nuestra aplicación web de una forma muy sencilla.

Para utilizar un tema de jQuery UI tenemos tres opciones:

- Crear nuestro propio css con todas las reglas de estilo que incluyen los distintos temas (demasiado costoso, ya que existen multitud de reglas).

- Descargar uno de los temas predefinidos existentes en la página web. Para ello, iremos a la url <http://jqueryui.com/themeroller/>. En la solapa Gallery del ThemeRoller, encontraremos una serie de temas predefinidos y los ejemplos de cómo se verían los distintos widgets de la librería con el tema que tengamos seleccionado. Sólo tenemos que elegir el que más nos guste y pulsar el botón Download.

- Utilizar la herramienta ThemeRoller de esta misma página para diseñar nuestro propio tema a partir de uno existente. Si seleccionamos la solapa Gallery del themeRoller y elegimos uno de los temas predefinidos, podemos modificar los estilos más importantes pulsando el botón Edit. Esto nos llevará a la solapa Roll Your Own. Aquí, podremos modificar las propiedades. Una vez tengamos el tema como nos interese, podemos descargarlo pulsando el botón Download Theme.



Incluir el tema en nuestra página

Una vez descargado el tema predefinido o personalizado que hayamos elegido, tendremos que vincularlo a nuestra página web. De esta forma, los estilos se aplicarán automáticamente a los widgets de jQuery UI que utilicemos.

Cuando utilizamos un tema personalizado no tenemos más remedio que copiarlo a nuestro directorio de estilos y vincularlo a la página, pero si utilizamos un tema predefinido, al igual que ocurría con la librería, tenemos la posibilidad de utilizar un CDN para incluirlo en nuestra página. Por ejemplo, para incluir el tema `black-tie`, usando el CDN de jQuery usaremos el siguiente código:

```
<link rel="stylesheet" type="text/css"
href="http://code.jquery.com/ui/1.12.0/themes/black-tie/jquery-ui.min.css" />
```

En la url <http://code.jquery.com/>, podemos encontrar todas las librerías, temas y plugins incluidos en este CDN.

4. Efectos en jQuery UI

Como vimos en el tema 4, jQuery soporta el uso de efectos visuales básicos, incluida la gestión de la opacidad y la altura de los elementos, así como la animación de otras propiedades CSS a través del método `.animate()`.

La librería jQuery UI nos amplía estas capacidades con las siguientes mejoras:

- 🚦 Nuevos efectos visuales.
- 🚦 Mejoras en el método `.animate()`.
- 🚦 La capacidad de utilizar clases CSS para producir efectos.

Nuevos efectos visuales

Además de la `.slideUp()`, `.slideDown()`, `.slideToggle()`, `.fadeIn()`, `.fadeOut()`, `.fadeTo()`, `.show()`, `.hide()`, y los efectos `toggle()` disponibles en jQuery, jQuery UI nos ofrece una variedad de nuevos efectos visuales. Todos estos efectos se pueden utilizar mediante una llamada al nuevo método `.effect()`.

La definición de este nuevo método es la siguiente:

```
$(selector).effect(effectName, options, duration, callback)
```

El único parámetro obligatorio es el nombre del efecto. Podemos seleccionar uno de los siguientes efectos: "blind", "bounce", "shake", "clip", "drop", "explode", "fold", "highlight", "puff", "pulsate", "scale", "size" y "slide". Cada uno de estos efectos puede recibir un objeto `options` (segundo parámetro del método `.effect()`) para cambiar su comportamiento por defecto (ver las opciones de cada efecto en la referencia del método <http://api.jqueryui.com/effect/>).

Además, como ocurre con los efectos básicos de jQuery, también podemos indicar una duración y una función que se ejecutará una vez el efecto haya terminado.

```
<!DOCTYPE html>
<html>
  <head>
    <title>método .effect()</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css"
href="http://code.jquery.com/ui/1.12.0/themes/black-tie/jquery-ui.min.css" />
    <script src="http://code.jquery.com/jquery-3.1.0.min.js"
      type="text/javascript"></script>
    <script src="http://code.jquery.com/ui/1.12.0/jquery-ui.min.js"
      type="text/javascript"></script>
    <script type="text/javascript">
      $(function()
      {
        $("#enlace").button(
```



```

    {
      label: "Destruye nave",
    }).click(function(evento)
    {
      evento.preventDefault();
      $("#img1").effect ("explode", { mode : "hide" }, 1000, function()
      {
        $("#enlace").button("option", "label", "Nave destruida");
      });
    });
  });
</script>
</head>
<body>
<a href="" id="enlace">enlace</a><br />

</body>
</html>

```

Ejemplo 1: [metodoEffect.html](#)

En el ejemplo, podemos ver como se oculta la imagen de la nave utilizando un efecto de "explode" y se cambia el texto del botón una vez ha finalizado el efecto.

Mejoras en el método .animate()

El método .animate() de jQuery, el cual nos permite realizar animaciones visuales cambiando las propiedades css de los elementos, tiene una serie de limitaciones en las propiedades css que puede utilizar, por ejemplo, no permite realizar una animación que vaya modificando las propiedades relacionadas con el color (background-color, border-color, etc.). Gracias a jQuery UI estas limitaciones desaparecen y podemos animar el color de los elementos como se muestra en el siguiente ejemplo.

```

<!DOCTYPE html>
<html>
  <head>
    <title>método .animate() con colores</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css"
href="http://code.jquery.com/ui/1.12.0/themes/black-tie/jquery-ui.min.css" />
    <script src="http://code.jquery.com/jquery-3.1.0.min.js"
      type="text/javascript"></script>
    <script src="http://code.jquery.com/ui/1.12.0/jquery-ui.min.js"
      type="text/javascript"></script>
    <script type="text/javascript">
      $(function()
      {
        $("#enlace").button(
        {
          label: "Cambia color",
        }).click(function(evento)
        {
          evento.preventDefault();

```

```
        $("p").animate (  
        {  
            "background-color" : "black",  
            "color" : "white"  
        }, 2000, function()  
        {  
            $("#enlace").button("option", "label", "Color cambiado");  
        });  
    });  
});  
</script>  
</head>  
<body>  
    <a href="" id="enlace">enlace</a><br />  
    <p> Párrafo 1 </p>  
    <p> Párrafo 2 </p>  
</body>  
</html>  
Ejemplo 2: animarColores.html
```

5. Utilizar Widgets de jQuery UI

Generalmente, el código JavaScript para la creación de cualquiera de los widgets es muy simple y directo:

```
$(selector).tipoWidget();
```

Cada tipo de widget puede requerir una estructura de marcado específica, por ejemplo, el widget `dialog` requiere un `<div>` que contendrá todo aquello que se va a mostrar en el diálogo.

La siguiente tabla ilustra el código necesario para inicializar algunos de los widgets de jQuery UI.

| Widget | Llamada al método |
|----------------------------|---|
| Botón (Button) | <code>\$('#myButton').button();</code> |
| Solapas (Tabs) | <code>\$('#tabs').tabs();</code> |
| Control fecha (Datepicker) | <code>\$('#date').datepicker();</code> |
| Diálogo (Dialog) | <code>\$('#myDialog').dialog();</code> |
| Barra de progreso | <code>\$('#itemLoader').progressbar();</code> |

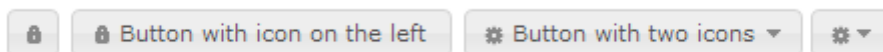
Tabla 1: Código de inicialización de algunos widgets

Los widgets también aceptan un objeto de configuración con las muchas opciones disponibles para el control de los mismos.

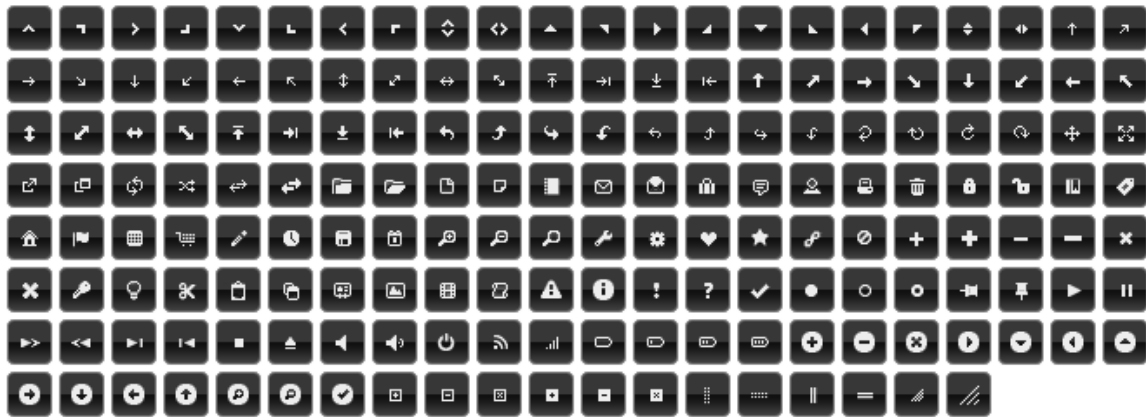
Botones de jQuery UI (widget Button)

Podemos hacer que un elemento html se visualice y comporte como un botón utilizando el método `.button()`. Este método recibirá un parámetro (`options`) que definirá cómo será el aspecto y comportamiento del botón.

Dentro de un botón podemos encontrar un texto, un icono, un texto y un icono, dos iconos (primario y secundario) o un texto y dos iconos.



Los iconos que podemos introducir en los botones están predefinidos en el fichero css del tema que estemos utilizando. Este fichero contiene un conjunto de clases css para acceder a los iconos localizados en el directorio de imágenes del tema. Por ejemplo, para el tema `black-tie` los iconos tienen el siguiente aspecto:



Para saber qué regla de estilo se corresponde con cada uno de estos iconos podemos ir al Theme Roller y, en la sección de iconos del tema, al situarnos sobre un icono, aparecerá un tooltip con su nombre de clase.

Dentro del objeto `options` que recibe el método `.button()`, podemos encontrar las opciones que se indican en la siguiente tabla.

| Opción | Función |
|-------------------------------|---|
| <code>options.disabled</code> | Cuando se establece a <code>true</code> el botón permanecerá inactivo. Los eventos <code>mouseover</code> no tendrán efecto, pero los clics sobre el botón continuarán teniéndose en cuenta. |
| <code>options.label</code> | Corresponde al texto mostrado en el botón. Si no se especifica, el contenido del elemento <code>html</code> será usado como texto del botón. |
| <code>options.icons</code> | Asocia iconos con el botón. Puede haber uno antes del texto (<code>primary</code>) y otro después (<code>secondary</code>). Los valores de las propiedades <code>primary</code> y <code>secondary</code> del objeto <code>icons</code> serán los nombres de las clases <code>css</code> definidas en el fichero del tema. |
| <code>options.text</code> | Indica si el texto del botón debe ser mostrado. Cuando vale <code>false</code> el texto no se mostrará. En este caso, al menos un icono debería estar presente. |

Tabla 2: Opciones del método `button`

```
<!DOCTYPE html>
<html>
  <head>
    <title>método .button()</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css"
href="http://code.jquery.com/ui/1.12.0/themes/black-tie/jquery-ui.min.css" />
    <script src="http://code.jquery.com/jquery-3.1.0.min.js"
      type="text/javascript"></script>
    <script src="http://code.jquery.com/ui/1.12.0/jquery-ui.min.js"
      type="text/javascript"></script>
    <script type="text/javascript">
      $(function()
      {
        $("#enlace").button(
        {
          icons:
          {
            primary: "ui-icon-triangle-1-s",
            secondary: "ui-icon-close-thick"
          },
          text: true
        });
      });
    </script>
  </head>
</html>
```

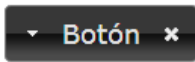
```

    });
  </script>
</head>
<body>
  <a href="" id="enlace">Botón</a>
</body>
</html>

```

Ejemplo 3: [metodoButton.html](#)

En el ejemplo anterior se mostrará un botón como este:



Como vemos, este botón contendrá el texto y los dos iconos indicados. Como no se ha introducido la propiedad `label` se toma el texto del enlace.

Cambiar las propiedades de un botón

También podemos utilizar el método `.button()` para cambiar alguna de las propiedades de un botón. Por ejemplo, podemos deshabilitarlo, cambiarle el texto, etc. Para hacer esto, le pasaremos al botón un string que especificará la acción que queremos realizar. La siguiente tabla muestra las acciones que le podemos pasar a este método.

| Acción | Función |
|---|--|
| <code>button("disable")</code> | Desactiva el botón |
| <code>button("enable")</code> | Activa el botón |
| <code>button("refresh")</code> | Refresca la visualización de un botón. |
| <code>button("option", param)</code> | Obtiene el valor de la opción especificada en <code>param</code> . La opción será una de las que podemos usar en el objeto <code>options</code> visto anteriormente. |
| <code>button("option", param, value)</code> | Cambia el valor de la opción especificada en <code>param</code> . |
| <code>button("destroy")</code> | Destruye el manejador del botón. El botón volverá a comportarse como el elemento <code>html</code> original. |

Tabla 3: Acciones del método `button`

Manejadores de eventos para los botones

jQuery UI no ha añadido nuevos eventos asociados a botones. De hecho, la gestión de las acciones del ratón corresponde a los eventos existentes (`click`, `mouseover`, etc.), utilizados como de costumbre por jQuery con los atajos del método `.bind()` correspondientes o el método `.on()`.

En el siguiente ejemplo se creará un botón y se le cambiará el icono secundario y el texto al hacer click sobre él.

```

<!DOCTYPE html>
<html>
  <head>
    <title>cambiar icono de un botón</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css"

```

```

href="http://code.jquery.com/ui/1.12.0/themes/black-tie/jquery-ui.min.css" />
<script src="http://code.jquery.com/jquery-3.1.0.min.js"
      type="text/javascript"></script>
<script src="http://code.jquery.com/ui/1.12.0/jquery-ui.min.js"
      type="text/javascript"></script>
<script type="text/javascript">
$(function()
{
  $("#volumen").button (
  {
    icons : { secondary : "ui-icon-volume-off" }
  }).click (function (event)
  {
    if($(this).button("option", "icons").secondary=="ui-icon-volume-off")
    {
      $(this).button("option", "icons", {secondary:"ui-icon-volume-on"});
      $(this).button("option", "label", "Volumen On");
    }
    else
    {
      $(this).button("option", "icons", {secondary:"ui-icon-volume-off"});
      $(this).button ("option", "label", "Volumen Off");
    }
  });
});
</script>
</head>
<body>
  <span id="volumen">Volumen Off</span>
</body>
</html>

```

Ejemplo 4: [cambiarIconoButton.html](#)

Cuadros de diálogo de jQuery UI (Dialog)

Los cuadros de diálogo de jQuery UI son soluciones interesantes para presentar la información en una página HTML. Podemos utilizarlos, por ejemplo, para hacerle una pregunta al usuario. Tienen el comportamiento tradicional de los cuadros de diálogo de las aplicaciones de escritorio, podemos moverlos, cambiarles el tamaño, y por supuesto, cerrarlos.

Para crear un cuadro de dialogo debemos tener un elemento `<div>` dentro de nuestra página que será el que convirtamos al widget Dialog. Este `<div>` tendrá un atributo `title` que contendrá el título del cuadro de diálogo y sus contenidos serán los que aparezcan dentro del mismo.

Para convertir el `<div>` a un cuadro de diálogo de jQuery, utilizaremos el método `.dialog()`. El cuadro de dialogo más básico que podemos tener es el que se muestra en el siguiente ejemplo.

```

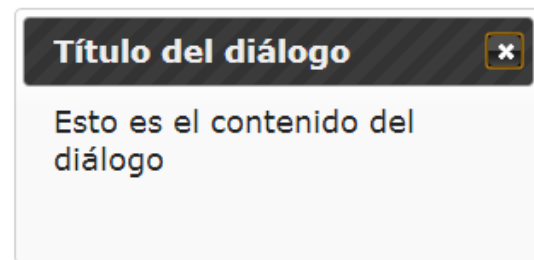
<!DOCTYPE html>
<html>
  <head>

```

```
<title>método .dialog()</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css"
href="http://code.jquery.com/ui/1.12.0/themes/black-tie/jquery-ui.min.css" />
<script src="http://code.jquery.com/jquery-3.1.0.min.js"
      type="text/javascript"></script>
<script src="http://code.jquery.com/ui/1.12.0/jquery-ui.min.js"
      type="text/javascript"></script>
<script type="text/javascript">
$(function()
{
    $("#dialogo").dialog();
});
</script>
</head>
<body>
<div id="dialogo"
    title="Título del diálogo">Esto es el contenido del diálogo</div>
</body>
</html>
```

[Ejemplo 5: dialogoBasico.html](#)

Si ejecutamos el ejemplo anterior, veremos que al cargar la página se muestra un cuadro de diálogo como el que aparece en la imagen. Este cuadro de diálogo ya incorpora las funcionalidades de desplazarlo, redimensionarlo y cerrarlo.



Por lo tanto, vemos que con una sola línea de código jQuery UI nos aporta una gran funcionalidad.

Cambiar el aspecto de los diálogos

Al usar el método .dialog() se cambia drásticamente la apariencia de los elementos HTML que contiene. Además, este método escanea el HTML y añade nuevas clases CSS a los elementos para darles el estilo apropiado.

Uno de los aspectos que podemos cambiar en los diálogos, que afectará a la funcionalidad del mismo, es la visualización del botón cerrar del diálogo. Este botón está asociado con un enlace (<a>) que tiene la clase CSS ui-dialog-titlebar-close. Para localizar este botón tendremos que acceder al elemento hermano del diálogo y buscar en él un elemento que tenga esta clase aplicada. Una vez localizado, podremos ocultarlo:

```
$("#div#dialog").dialog().prev().find(".ui-dialog-titlebar-close").hide();
```

Personalización de los diálogos

Como ya hemos comentado anteriormente, una de las grandes ventajas de la librería jQuery UI es su homogeneidad, gracias a la cual el funcionamiento de los distintos widgets es muy parecido, así pues, el método `.dialog()` puede recibir un objeto `options` con las opciones de configuración del cuadro de dialogo. La siguiente tabla nos muestra las distintas opciones de las que disponemos.

| Opción | Función |
|------------------------------------|---|
| <code>options.title</code> | Título de la ventana |
| <code>options.buttons</code> | Añade botones al cuadro de diálogo. Será una lista de objetos, donde el nombre de la propiedad será un string que representará el texto que aparece en el botón, y el valor de la propiedad será una función que será llamada cuando se pulse sobre el botón. |
| <code>options.position</code> | La posición del diálogo. Indicaremos las coordenadas en las que se ubica el diálogo. También podemos indicar una cadena en la que especificamos en qué posición queremos que se ubique. Por ejemplo: "left top": El dialogo se colocará en la esquina superior izquierda. "right bottom": El dialogo se colocará en la esquina inferior derecha. "top": El dialogo se colocará en la parte superior y centrado en anchura. "left": El dialogo se colocará en la parte izquierda y centrado en altura. "center": El dialogo se colocará centrado en anchura y altura. Por defecto la posición será "center". |
| <code>options.height</code> | El alto inicial (en píxeles) del diálogo. El valor por defecto es "auto" (el tamaño se ajustará automáticamente a los contenidos). |
| <code>options.width</code> | El ancho inicial (en píxeles) del diálogo. Por defecto es 300. |
| <code>options.maxHeight</code> | Máximo alto (en píxeles) al que el diálogo puede ser redimensionado. |
| <code>options.maxWidth</code> | Máximo ancho (en píxeles) al que el diálogo puede ser redimensionado. |
| <code>options.minHeight</code> | Mínimo alto (en píxeles) al que el diálogo puede ser redimensionado. Por defecto es 150. |
| <code>options.minWidth</code> | Mínimo ancho (en píxeles) al que el diálogo puede ser redimensionado. Por defecto es 150. |
| <code>options.show</code> | Efecto que se utilizará para mostrar el diálogo. Podemos elegir cualquiera de los que se han mencionado en el punto de efectos. Si se pasa false (valor por defecto) no se utilizará ningún efecto. |
| <code>options.hide</code> | Efecto que se utilizará para ocultar el diálogo. Podemos elegir cualquiera de los que se han mencionado en el punto de efectos. Si se pasa false (valor por defecto) no se utilizará ningún efecto. |
| <code>options.autoOpen</code> | Si vale true (valor por defecto), el dialogo se abre con la llamada al método <code>.dialog()</code> . Si vale false se mostrará cuando se llame al método <code>.open()</code> . |
| <code>options.draggable</code> | Si vale true (valor por defecto) el dialogo puede ser movido. |
| <code>options.resizable</code> | Si vale true (valor por defecto) el dialogo puede ser redimensionado. |
| <code>options.modal</code> | Si vale true el diálogo será modal. El resto de elementos de la página no estarán accesibles hasta que el diálogo se cierre. El valor por defecto es false. |
| <code>options.stack</code> | Si vale true (valor por defecto) el dialogo puede ser apilado y los usuarios podrán traerlo al frente haciendo click sobre él. Si vale false el diálogo será abierto y se mostrará sobre los anteriores que hubieran abiertos, pero los usuarios no podrán cambiar el orden de la pila. |
| <code>options.closeOnEscape</code> | Si vale true (valor por defecto) el dialogo se cerrará al pulsar la tecla escape. |

Tabla 4: Opciones de configuración de los cuadros de diálogo

Además de las opciones que acabamos de ver en la tabla anterior, también podemos configurar dentro del objeto `options` manejadores de evento que

gestionen el comportamiento del cuadro de diálogo. Los eventos de los cuales dispone un diálogo son los siguientes: focus, open, beforeclose, close, drag, dragStart, dragStop, resize, resizeStart, resizeStop.

Para asociar un manejador de evento con uno de estos eventos, sólo tenemos que indicar la función manejadora dentro del objeto options que pasamos:

```
$(selector).dialog(
{
...
  close: function(evento)
  {
    ...
  },
...
}
```

En el siguiente ejemplo se crea un diálogo con un elemento contenedor (<div>) que se crea al vuelo. El diálogo que se abre es modal con efectos de apertura y cierre. Además, se introducen dos botones con sus correspondientes manejadores y se añade un manejador para el evento close.

```
<!DOCTYPE html>
<html>
  <head>
    <title>método .dialog() con opciones</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css"
href="http://code.jquery.com/ui/1.12.0/themes/black-tie/jquery-ui.min.css" />
    <script src="http://code.jquery.com/jquery-3.1.0.min.js"
      type="text/javascript"></script>
    <script src="http://code.jquery.com/ui/1.12.0/jquery-ui.min.js"
      type="text/javascript"></script>
    <script type="text/javascript">
      $(function()
      {
        var $contenido = $('<div id="dialog" title="Dialog Title">¿Cuál es tu
respuesta?</div>');
        $("body").append($contenido);
        $("#dialog").dialog(
        {
          modal: true,
          show: {effect: "bounce", duration: 1000},
          hide: {effect: "explode", duration: 1000},
          title: "Dialogo con opciones",
          closeOnEscape: false,
          buttons:
          {
            "Sí": function()
            {
              $("body").append("<p>se ha pulsado Sí</p>");
              $(this).dialog( "close" );
            },
            "No": function ()
            {
              $("body").append("<p>se ha pulsado No</p>");
```

```

        $(this).dialog("close");
    }
},
close: function(evento)
{
    $contenido.remove();
    $("body").append($("<p>Adios</p>"));
}
);
});
</script>
</head>
<body>
</body>
</html>

```

Ejemplo 6: [dialogoOpciones.html](#)

Cambiar las propiedades de un diálogo

Al igual que ocurría con los botones, podemos utilizar el método `.dialog()` para cambiar alguna de las propiedades del diálogo, pero en este caso también podemos utilizar este método para realizar una serie de acciones como abrir el diálogo, cerrarlo, etc.

Para hacer esto, le pasaremos al diálogo un `string` que especificará la acción que queremos realizar. La siguiente tabla muestra las acciones que le podemos pasar a este método.

| Acción | Función |
|---|--|
| <code>dialog("open")</code> | Abre el cuadro de diálogo |
| <code>dialog("close")</code> | Cierra el cuadro de diálogo |
| <code>dialog("destroy")</code> | Destruye el manejador del diálogo. El diálogo volverá a comportarse como el elemento html original. |
| <code>dialog("disable")</code> | Desactiva el cuadro de diálogo |
| <code>dialog("enable")</code> | Activa el cuadro de diálogo |
| <code>dialog("isOpen")</code> | Devuelve true si el diálogo está abierto. Si se selecciona más de un diálogo con el selector, devolverá true si al menos uno está abierto. |
| <code>dialog("moveToTop")</code> | Pone el diálogo al frente de la pila de diálogos abiertos. |
| <code>dialog("option", param)</code> | Obtiene el valor de la opción especificada en param. La opción será una de las que podemos usar en el objeto options visto anteriormente. |
| <code>dialog("option", param, value)</code> | Cambia el valor de la opción especificada en param. |

Tabla 5: Acciones del método Dialog

Tooltips de jQuery UI (Tooltip)

El widget tooltip reemplaza los tooltips nativos dándoles el aspecto del tema que tengamos seleccionado y permitiéndonos varias personalizaciones:

- Mostrar otras cosas además del atributo title, por ejemplo, contenido extra obtenido mediante una petición Ajax.
- Personalizar la posición, por ejemplo, centrar el tooltip bajo los elementos.

- ✚ Añadir estilos extra para personalizar la apariencia en campos de error o aviso.

Para mostrar y ocultar el tooltip se utiliza un animación `fade`, aunque esto puede ser personalizado mediante las opciones `show` y `hide`.

Para convertir los tooltips nativos de un elemento en tooltips de jQuery UI, usaremos el método `.tooltip()`. Gracias a la delegación de eventos, podemos hacer que todos los tooltips nativos de nuestra página se sustituyan, simplemente, introduciendo la siguiente línea de código:

```
$(document).tooltip();
```

Al igual que en los widgets anteriores, el método `.tooltip()` puede recibir un objeto de opciones o realizar una determinada acción sobre el tooltip del elemento seleccionado. Para más información podéis consultar la documentación en la página <http://api.jqueryui.com/tooltip/>.

```
<!DOCTYPE html>
<html>
  <head>
    <title>método .tooltip()</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css"
href="http://code.jquery.com/ui/1.12.0/themes/black-tie/jquery-ui.min.css" />
    <script src="http://code.jquery.com/jquery-3.1.0.min.js"
      type="text/javascript"></script>
    <script src="http://code.jquery.com/ui/1.12.0/jquery-ui.min.js"
      type="text/javascript"></script>
    <script type="text/javascript">
      $(function()
      {
        $(document).tooltip();
      });
    </script>
  </head>
  <body>
    <a href="" title="Descripción enlace 1">enlace 1</a><br />
    <a href="" title="Descripción enlace 2">enlace 2</a><br />
  </body>
</html>
```

Ejemplo 7: [metodoTooltip.html](#)


6. Interacciones en jQuery UI (Drag-and-drop)


Drag-and-drop es una operación común en las páginas web para mover un objeto usando el ratón, la idea es arrastrar un elemento de la página y soltarlo sobre otro elemento. Por ejemplo, si la página muestra imágenes de artículos para comprar, los usuarios pueden arrastrar los artículos hasta el carro que simboliza todos los artículos a comprar.

jQuery UI puede gestionar estas operaciones, distinguiendo la operación de "arrastrar" (el movimiento de un objeto) y la operación de "soltar" (el depósito del elemento que se mueve). Para ello, jQuery UI ofrece el método `.draggable()` y el método `.droppable()`.

El método `.draggable()`

Este método gestiona los elementos de la página web que queremos poder arrastrar. Como siempre, este método tiene dos usos: un primer uso, que recibirá un objeto `options` y un segundo uso que nos permitirá cambiar algunas de estas propiedades o realizar una acción determinada.

```
 $(selector).draggable(options)
```

```
 $(selector).draggable("action", params)
```

El método de `.draggable(options)` declara que un elemento HTML se puede mover en la página HTML. El parámetro `options` es un objeto que especifica el comportamiento de los elementos que intervienen.

Existen multitud de opciones que podemos especificar para definir el comportamiento de esta operación, podemos consultar la especificación del método en la url <http://api.jqueryui.com/draggable/>. A continuación, veremos un resumen con las más importantes.




| Opción | Función |
|-----------------|---|
| options.helper | Crea y mueve una copia del elemento seleccionado. El valor "clone" indica que el elemento se duplica y que es el nuevo elemento el que se mueve, mientras que el original permanece en su posición. Con "original" (por defecto) se mueve el elemento inicial. Si se especifica una función, esta deberá crear y devolver un nuevo elemento que será el que se mueva. En cualquier caso, si se crea un nuevo elemento (por "clone" o utilizando una función), este se eliminará al final del movimiento. |
| options.cursor | Especifica el valor para la propiedad css "cursor" que se utilizará cuando los elementos se están moviendo. Esto hará que se muestre el cursor que indiquemos. Los posibles valores son: auto, crosshair, default, pointer, move, e-resize, ne-resize, nw-resize, n-resize, se-resize, sw-resize, s-resize, w-resize, text, wait, help. |
| options.opacity | La opacidad que tomará elemento durante el movimiento, por defecto es 1. |

| | |
|------------------------|--|
| options.revert | Indica si el elemento se mueve a su posición original al final del movimiento. Si vale true el elemento vuelve a su posición original. Si vale false (el valor predeterminado) el elemento queda donde fue soltado. Cuando se establece en "valid" el elemento vuelve si se ha soltado en un elemento que lo acepta y cuando se establece en "invalid" el elemento vuelve si se ha soltado en un elemento que no la acepta. |
| options.revertDuration | Duración (en milisegundos) del desplazamiento que hace que el elemento vuelva a su posición original (propiedad revert). Por defecto es 500 ms. |
| options.revertDuration | Duración (en milisegundos) del desplazamiento que hace que el elemento vuelva a su posición original (propiedad revert). Por defecto es 500 ms. |



Tabla 6: Opciones del método draggable

Eventos que se producen al arrastrar los elementos

Los eventos asociados con elementos móviles gestionan el comienzo y el final del desplazamiento, así como el propio desplazamiento. En concreto, durante la operación de arrastrar se lanzan tres tipos de eventos:

-  **start.** se produce cuando se inicia el desplazamiento (se ha pulsado con el ratón sobre el elemento y ha comenzado a moverse).
-  **Drag.** Se produce cuando el desplazamiento continúa después del primer movimiento.
-  **drop.** Se produce cuando se finaliza la operación de arrastrado (se suelta el botón del ratón)

Cada uno de los métodos asociados con estos eventos tiene dos parámetros:

-  **event** es el evento de ratón
-  **ui** es un objeto que contiene los atributos que se indican en la siguiente tabla.

| Atributo | Función |
|-------------|--|
| ui.helper | Objeto jQuery que representa el elemento que se está moviendo. Será el elemento sobre el que comenzó el arrastre o el indicado en la propiedad options.helper. |
| ui.position | Si el elemento que se está moviendo es el indicado en la propiedad options.helper, este atributo indica la posición (top, left) del elemento relativa a los bordes de la página. Si el elemento que se mueve es el original sobre el que se inició el arrastre, esta propiedad indica el desplazamiento desde la propiedad inicial del elemento. |
| ui.offset | Indica en todos los casos la posición (top, left) del elemento que se está moviendo relativa a los bordes de la página. |

Tabla 7: Atributos del objeto ui

jQuery UI nos permite gestionar estos eventos utilizando el método `.bind()`, o los atajos correspondientes.

```
<!DOCTYPE html>
<html>
  <head>
    <title>método .draggable()</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css"
href="http://code.jquery.com/ui/1.12.0/themes/black-tie/jquery-ui.min.css" />
    <script src="http://code.jquery.com/jquery-3.1.0.min.js">
```

```

        type="text/javascript"></script>
<script src="http://code.jquery.com/ui/1.12.0/jquery-ui.min.js"
        type="text/javascript"></script>
<script type="text/javascript">
$(function()
{
    $("#div1 span").draggable (
    {
        helper : "clone",
        revert: true,
        start : function (event, ui)
        {
            $("#start").text (ui.offset.top + ", " + ui.offset.left);
        },
        drag : function (event, ui)
        {
            $("#drag").text (ui.offset.top + ", " + ui.offset.left);
        },
        stop : function (event, ui)
        {
            $("#stop").text (ui.offset.top + ", " + ui.offset.left);
        }
    });
});
</script>
</head>
<body>
<div id="div1" style="border:solid 1px;background-color:gainsboro;">
    <span>Item 1 </span><br /><br />
    <span>Item 2 </span><br /><br />
    <span>Item 3 </span>
</div>
<p>Start : <span id="start"></span></p>
<p>Drag : <span id="drag"></span></p>
<p>Stop : <span id="stop"></span></p>
</body>
</html>

```

Ejemplo 8: [metodoDraggable.html](#)

En el ejemplo podemos observar como al arrastrar los ítems se crea una copia (helper:"clone") del elemento arrastrado, y cuando se suelta el elemento, vuelve a su posición original (revert: true). Además, a medida que se va arrastrando se va mostrando la posición en la que se encuentra.

Acciones del método .draggable()

Al igual que en el resto de la librería, el método .draggable() nos permite indicar acciones que podemos realizar sobre los elementos arrastrables.

| Acción | Función |
|----------------------|--|
| draggable("disable") | Desactiva la posibilidad de arrastrar los elementos seleccionados. Los elementos no se podrán mover hasta que se llame al método draggable pasándole la acción "enable". |
| draggable("enable") | Reactiva la posibilidad de arrastrar los elementos. |
| draggable("destroy") | Destruye el manejador del arrastrado. Los elementos dejarán de ser movibles. |

| | |
|--|---|
| <code>draggable("option", param)</code> | Obtiene el valor de la opción especificada en param. La opción será una de las que podemos usar en el objeto options visto anteriormente. |
| <code>draggable("option", param, value)</code> | Cambia el valor de la opción especificada en param. |

Tabla 8: Acciones del método draggable

El método .droppable()

El método `.droppable()` gestiona los elementos de la página HTML en los que se desea soltar un objeto arrastrado.

Cuando llamamos al método `.droppable()` pasándole un objeto `options`, estamos indicando que los elementos seleccionados pueden ser utilizados como destino de arrastre de otros elementos.

Este objeto `options` define, principalmente, los elementos que pueden ser soltados en estos elementos de destino y el comportamiento de los elementos cuando se suelta un elemento sobre ellos.

Al igual que ocurría en el método `.draggable()`, el objeto `options` de este método contiene un gran número de propiedades. Veremos a continuación un resumen de las mismas.



| Opción | Función |
|----------------------------------|---|
| <code>options.tolerance</code> | Indica cómo el objeto arrastrado debería situarse sobre el elemento de destino para ser aceptado. Los posibles valores son: "fit" (el elemento arrastrado debe estar totalmente dentro del destino), "intersect" (la mitad), "touch" (tocando) y "point" (El ratón ha entrado al elemento de destino). El valor por defecto es "intersect". |
| <code>options.accept</code> | Se trata de un selector que nos indica los elementos que son aceptados. |
| <code>options.hoverClass</code> | String que representa una o más clases css que serán añadidas al elemento de destino cuando un elemento aceptable se mueva sobre él. |
| <code>options.activeClass</code> | String que representa una o más clases css que serán añadidas al elemento de destino cuando un elemento aceptable este siendo arrastrado (no necesariamente sobre él). |

Tabla 9: Principales opciones del método .droppable()

Eventos que se producen en los elementos de destino

Los eventos asociados con elementos de destino del arrastre se utilizan para gestionar el inicio y el final del movimiento de un elemento aceptado y el depósito del propio elemento.

Cada uno de los métodos asociados con estos eventos tiene dos parámetros:

-  `event.` corresponde al evento de ratón.
-  `ui.` es un objeto que contiene las propiedades `{draggable, helper, offset}`:
 - `ui.draggable`: Objeto jQuery asociado con el elemento sobre el que se inició el arrastre (puede no coincidir con el elemento arrastrado,

en función del valor de la propiedad `helper` del método `.draggable()`.

- `ui.helper`: Objeto jQuery asociado con el elemento arrastrado (puede no coincidir con el elemento sobre el que se inició el arrastre, en función del valor de la propiedad `helper` del método `.draggable()`).
- `ui.offset`: En todos los casos, indica la posición (`top`, `left`) del elemento arrastrado relativa a los bordes de la página.

La siguiente tabla nos muestra los diferentes eventos que se producen en la operación de arrastrar y soltar en los elementos de destino. Podemos asociar manejadores de evento a través del objeto `options` del método `.droppable()`.

| Evento | Descripción |
|---------------------------------|--|
| <code>options.activate</code> | El evento se produce cuando comienza el arrastre de un elemento aceptable. |
| <code>options.deactivate</code> | El evento se produce cuando finaliza el arrastre de un elemento aceptable (se suelta el botón del ratón). |
| <code>options.over</code> | El evento se produce cuando el elemento arrastrado se encuentra sobre el elemento de destino (según la definición de la opción <code>tolerance</code>). |
| <code>options.out</code> | El evento se produce cuando el elemento arrastrado sale del elemento de destino. |
| <code>options.drop</code> | El evento se produce cuando se suelta un elemento aceptable sobre el elemento de destino. |

Tabla 10: Eventos del método `.droppable()`

Acciones del método `.droppable()`

Las acciones que podemos utilizar con el método `.droppable()` son exactamente las mismas que para el método `.draggable()`: “disable”, “enable”, “destroy”, (“option”, param) y (“option”, param, value).

En el siguiente ejemplo veremos una pequeña aplicación de carrito de la compra que funciona con arrastrar y soltar.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mini carrito de la compra</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css"
href="http://code.jquery.com/ui/1.12.0/themes/black-tie/jquery-ui.min.css" />
    <style type="text/css">
      .carro { border : transparent solid 2px; }
      .hover { border-color : red; }
      #libros {width:400px; height: 100px; border: solid thin black;}
      #libros img {margin: 10px;}
    </style>
    <script src="http://code.jquery.com/jquery-3.1.0.min.js"
      type="text/javascript"></script>
    <script src="http://code.jquery.com/ui/1.12.0/jquery-ui.min.js"
      type="text/javascript"></script>
    <script type="text/javascript">
```



```

$(function()
{
    $("div#libros img").draggable(
    {
        revert : "invalid"
    });
    $("div#carro img.carro").droppable(
    {
        hoverClass : "hover",
        drop : function (event, ui)
        {
            $("div#carro").append (ui.draggable);
            $(ui.draggable).css(
            { position:"relative", top:"0px", left:"0px", margin: "10px" })
            .addClass ("comprado");
        }
    });
    $("div#libros").droppable (
    {
        accept : ".comprado",
        drop : function (event, ui)
        {
            $("div#libros").append (ui.draggable);
            $(ui.draggable).css (
            { position:"relative", top:"0px", left:"0px", margin: "10px" })
            .removeClass ("comprado");
        }
    });
});
</script>
</head>
<body>
    <h2> Arrastra las libros al carro </h2>
    <div id="libros">
        
        
        
        
    </div>
    <hr />
    <h2> Carro: </h2>
    <div id="carro">
        
    </div>
</body>
</html>

```

[Ejemplo 9: carritoArrastrarYSoltar.html](#)

En el ejemplo las imágenes de los libros son `draggable`s y el carrito es `droppable`. Si lo probamos, vemos que podemos arrastrar los libros al carrito y estos se añadirán a la derecha del mismo (en el evento `drop` del carrito se añade el elemento arrastrado `ui.draggable`). Al añadir un libro al carrito, desaparece de la lista de libros a comprar (`revert:"invalid"`). Además, podemos devolver un libro

comprado a la lista de libros arrastrándolo hasta ella (la lista de libros también es droppable).

7. Ejercicios

Hemos hablado durante este tema de los plugins jQuery, y más concretamente, de jQuery UI. Realizaremos ahora una serie de modificaciones sobre nuestra aplicación para practicar los nuevos contenidos.

Ejercicio 1

Ya hemos comentado que para poder trabajar con la librería jQuery UI, debemos incluirla en nuestra página. En nuestro caso, vamos a incluirla utilizando el CDN de jQuery.

Además, hemos visto que la librería también nos proporciona una serie de temas CSS para darle a nuestra aplicación el aspecto que nos interese. En nuestro caso, vamos a utilizar un tema predefinido, en concreto, el tema llamado “black-tie”.

Añade al fichero `carro.html` el código necesario para utilizar la librería jQuery UI. Después, debes enlazar el CSS del tema `black-tie` utilizando el CDN de jQuery.

Ejercicio 2

Para probar los nuevos efectos de jQuery UI, vamos a cambiar el efecto que se utiliza para eliminar un artículo del carrito. En este caso, utilizaremos el método `.effect()` con el efecto `explode`.

Ejercicio 3

Desde el segundo tema venimos arrastrando un comportamiento algo incorrecto de nuestra aplicación. Cuando introducimos en el carrito el mismo ítem más de una vez aparecen varias copias del artículo. Quizá, sería más correcto llevar un control de la cantidad de artículos del mismo tipo que queremos comprar, de forma que cuando introducimos un ítem que ya existe en el carrito se incremente la cantidad y no se cree una copia del mismo.

Para implementar esta funcionalidad seguiremos los siguientes pasos:

- ✚ Al añadir el producto al carrito, además del enlace `delete` para eliminar el artículo del carrito, añadiremos también un `<input>` que guardará la cantidad de artículos que vamos a comprar, un enlace para incrementar la cantidad y otro enlace para decrementarla.
- ✚ Debemos crear los elementos utilizando este código:

- El enlace para eliminar los elementos:
`$('')`
- El input para la cantidad:
`$('<input class="cantidad" type="text" value="1" readonly="true" />')`
- El enlace para disminuir la cantidad:
`$('')`
- El enlace para incrementar la cantidad:
`$('')`

✚ Para añadir los elementos a la copia del ítem que vamos a introducir en el carro de la compra, debemos utilizar el método `.prepend()` de la siguiente forma:

```
.prepend($minus, $add, $delete, $cantidad)
```

Es importante que respetemos el orden indicado para que los elementos se muestren como se espera.

✚ Además, los tres enlaces creados (`delete`, `minus` y `add`) deben convertirse en botones de la librería jQuery UI. Para ello, utilizaremos el método `button` explicado anteriormente. Serán botones sin texto (propiedad `text` a `false`) y los iconos primarios que se mostrarán serán los siguientes:

- Para el botón `delete` `ui-icon-circle-close`.
- Para el botón `add` `ui-icon-circle-plus`.
- Para el botón `minus` `ui-icon-circle-minus`.

✚ En nuestro fichero de estilos (`carro.css`), tenemos dos reglas de estilo que ya no necesitamos, por lo tanto, las vamos a eliminar. Estas reglas son las siguientes:

```
.delete
{
    background:url(..img/delete.png);
}
.delete:hover
{
    background-image:url(..img/delete-hover.png);
}
```

Debemos buscar estas dos reglas dentro del fichero y comentarlas, de esta forma, cambiaremos la imagen que se mostraba en el enlace `delete`.

✚ Una vez los botones están creados, implementaremos las funcionalidades necesarias para que su comportamiento sea el esperado:

- Al añadir un artículo al carrito, tendremos que comprobar si el artículo ya existe, en cuyo caso, sólo incrementaremos la cantidad. Para la comprobación podemos utilizar el siguiente código:

```
if ($("#cart_items").children().is("#c"+$(this).attr("id")))
```

Si el método `.is()` devuelve `true` significa que el artículo ya existe en el carrito, en caso contrario, tendremos que añadirlo.
- Al eliminar un artículo del carrito, tenemos que actualizar el stock, el número de artículos comprados y el precio total, pero como la cantidad puede ser mayor que 1, tendremos que actualizar estos valores teniendo en cuenta la cantidad actual.
- Al igual que hacíamos con el enlace `delete`, debemos añadir los manejadores de evento correspondientes para los enlaces `add` y `minus`. Esto lo haremos en el método `document.ready` y utilizaremos el método `.on()` para que los manejadores se asocien con los elementos actuales y futuros. Evidentemente, el evento será `click`.
- Al pulsar el botón `add`, tendremos que hacer exactamente lo mismo que al hacer doble click sobre el artículo de la lista, por lo tanto, lo más sencillo es que desde este manejador, obtengamos el ítem correspondiente de la lista y ejecutemos el manejador de evento `dblclick` utilizando el método `.trigger()`.
- En el momento que el stock disponible de un artículo valga 0, ocultaremos el botón `add` para que no se pueda seguir incrementando la cantidad. Si el stock disponible vale 0 y se pulsa el botón `minus`, se volverá a mostrar el botón `add`.
- Al pulsar el botón `minus`, tendremos que comprobar si la cantidad actual es mayor que uno. Si es así, decrementaremos la cantidad y actualizaremos los valores de stock, número de artículos comprados y precio total. En caso contrario, habrá que eliminar el artículo del carro, por lo tanto, habrá que hacer lo mismo que al pulsar el botón `delete`, así que lo más sencillo será invocar a su manejador del evento `click` mediante el método `.trigger()`.

Ejercicio 4

Otra cosa fácilmente mejorable de nuestra aplicación es el aspecto de los botones de nuestra barra de navegación (`btn_comprar`, `btn_clear`, `btn_prev`, `btn_next`). Vamos a convertirlos en botones de la librería jQuery UI mediante el método `.button()`. Al igual que los botones `add`, `minus` y `delete`, serán botones sin texto (propiedad `text` a `false`) y los iconos primarios que se mostrarán serán los siguientes:

- ✚ Para el botón `btn_comprar` `ui-icon-cart`.
- ✚ Para el botón `btn_clear` `ui-icon-trash`.
- ✚ Para el botón `btn_prev` `ui-icon-circle-triangle-w`.
- ✚ Para el botón `btn_next` `ui-icon-circle-triangle-e`.

Ejercicio 5

Para mejorar el aspecto de nuestra página y hacer que funcione de una manera más homogénea (visualmente hablando), vamos a hacer que los tooltips de la página tomen la apariencia del tema de jQuery UI que estamos utilizando.

Ejercicio 6

En este ejercicio vamos a pedir confirmación al usuario antes de vaciar nuestro carro de la compra. Evidentemente, esto lo haremos en el manejador del evento click del botón vaciar.

Al pulsar este botón, se debe mostrar un dialogo de jQuery UI que muestre el siguiente mensaje: "¿Seguro que desea vaciar el carrito?". Crearemos un `<div>` que contendrá el mensaje y se lo añadiremos al `body`, después al cerrar el diálogo se eliminará el `<div>`.

El dialogo tendrá las siguientes características:

- ✚ Tendrá un botón Sí y otro botón No. Si se pulsa el botón No, se cerrará y se cancelará el vaciado. Si se pulsa el botón Sí, se cerrará y se vaciará el carrito.
- ✚ Debe de ser un Dialogo modal.
- ✚ La duración de los efectos de mostrado y ocultación será de 600 ms.
- ✚ El título será "Confirmar vaciado".
- ✚ No se mostrará el botón cerrar.



No se cerrará con la tecla escape.

(Ver el ejemplo 6)

Ejercicio 7

En este último ejercicio, implementaremos la posibilidad de añadir artículos al carrito arrastrándolos y soltándolos dentro del mismo. Para ello, haremos que los elementos de la clase `ítem` sean `draggables` y el elemento `cart_items` sea `droppable`.

Al arrastrar un ítem al carrito se hará una copia del mismo, que será la que se arrastre. Al soltar un artículo sobre el carrito, llamaremos al manejador del evento `dblclick` del ítem que se está añadiendo. Esto hará que se cree una copia del ítem en el carrito o se incremente la cantidad en el caso de que ya existiera. El elemento arrastrado volverá a su lugar de origen (`revert:true`).

El carrito debe aceptar elementos de la clase `.item`. Al comenzar el arrastre de un ítem se asociará al carrito la clase `drop-active` y al situar el ítem arrastrado sobre él se le asociará la clase `drop-hover`.

Con esto terminamos los contenidos del tema 5, en el siguiente tema hablaremos de la interacción con el servidor por medio de peticiones Ajax. En el aula virtual podéis encontrar un vídeo en el que se muestra cómo debe quedar la aplicación después de realizar todos los ejercicios del tema.

8. Índice de ejemplos y tablas

Ejemplos

| | |
|---|----|
| EJEMPLO 1: METODOEFFECT.HTML..... | 8 |
| EJEMPLO 2: ANIMARCOLORES.HTML | 9 |
| EJEMPLO 3: METODOBUTTON.HTML | 12 |
| EJEMPLO 4: CAMBIARICONOBUTTON.HTML | 13 |
| EJEMPLO 5: DIALOGOBASICO.HTML | 14 |
| EJEMPLO 6: DIALOGOOPCIONES.HTML | 17 |
| EJEMPLO 7: METODOTOOLTIP.HTML | 18 |
| EJEMPLO 8: METODODRAGGABLE.HTML | 21 |
| EJEMPLO 9: CARRITOARRASTRARYSOLTAR.HTML | 24 |

Tablas

| | |
|--|----|
| TABLA 1: CÓDIGO DE INICIALIZACIÓN DE ALGUNOS WIDGETS | 10 |
| TABLA 2: OPCIONES DEL MÉTODO BUTTON | 11 |
| TABLA 3: ACCIONES DEL MÉTODO BUTTON | 12 |
| TABLA 4: OPCIONES DE CONFIGURACIÓN DE LOS CUADROS DE DIÁLOGO | 15 |
| TABLA 5: ACCIONES DEL MÉTODO DIALOG | 17 |
| TABLA 6: OPCIONES DEL MÉTODO DRAGGABLE | 20 |
| TABLA 7: ATRIBUTOS DEL OBJETO UI | 20 |
| TABLA 8: ACCIONES DEL MÉTODO DRAGGABLE | 22 |
| TABLA 9: PRINCIPALES OPCIONES DEL MÉTODO .DROPPABLE() | 22 |
| TABLA 10: EVENTOS DEL MÉTODO .DROPPABLE() | 23 |