Índice

Contenido

Práctica 4. JQuery

Introducción

En esta práctica vamos a reimplementar la versión de gestión CRUD (Create-Read-Update-Delete) realizada en la práctica anterior aprovechando el framework JQuery, simplificando la implementación anterior basada en javascript crudo. Partiremos de la versión tew-gestioneitorv2_0.

Esta sesión se estructurará en dos pasos correspondientes a sendos proyectos:

1. Proyecto CRUD con JQuery en lógica de negocio de la gestión de alumnos.
2. Proyecto CRUD con JQuery con los campos Ciudad y Dirección en JQuery.

Gestión de eventos en JQuery

En JQuery hay que tener en cuenta dos aspectos respecto a la gestión de eventos:

- Uso de ready
- Evento por defecto y propagación hacia arriba en DOM

Uso de ready

El evento ready() se dispara después de que al recargar una página todo el árbol DOM ha sido cargado[1]. Permite definir las funciones que se ejecutarán una vez esté cargado el árbol DOM. Estas funciones y manejadores de eventos se deben incluir en el ámbito de la función jQuery ready():

```
$(function() {
// INICIALIZACIONES
 //DEF. FUNCIONES
//DEF. MANEJADORES DE EVENTOS
});
```

Pueden definirse tantas veces como se desea en diferentes puntos de código. Se comporta como un spacename (JQuery).

Las inicializaciones se ejecutarán cada vez que se cargue el árbol DOM, las funciones serán invocadas desde los manejadores de eventos o desde otras funciones y los manejadores cuando se disparen los eventos oportunos. Aquellos eventos que tengan como comportamiento por defecto del navegador recargar la página obligarán a JQuery a invocar a ready() una vez se haya ejecutado el manejador para el evento en cuestión (submit por ejemplo), y consecuentemente se habrán ejecutado de nuevo todas las inicializaciones de ready(). Esto unas veces puede interesarnos y otras no. En el trascurso de esta clase veremos en qué situaciones nos interesa y en cuáles no.

[1] Se diferencia del evento load() en que load() espera porque finalice la carga tanto de imágenes como de enlaces y ready() sólo de los elementos del árbol DOM.

Evento por defecto y propagación hacia arriba en DOM

Cuando se captura un evento en JQuery, la función manejadora debe retornar un valor booleano (en caso de que no se retorne nada, sea asume true):

- En caso de retornar true, el evento provocará la acción por defecto y será propagada hacia arriba en el árbol DOM desde el elemento que ha disparado el evento en cuestión. Por ejemplo un "submit" tiene como efecto enviar los elementos del formulario, resetar el formulario y recargar la página con el resultado del action asociada al formulario.
- En el caso de retornar false, una vez ejecutado el manejador, se ejecutarán los métodos asociados al evento gestionado:
    - Event.preventBydefault(), anulando la acción por defecto asociada al evento.
    - Event.stopPropagation(), anula la propagación del efecto hacia arriba en el árbol DOM.

  La mayoría de las veces sólo es necesario uno de los métodos.

Paso1: Adaptación a JQuery de la gestión CRUD de alumnos

1. Lo primero que deberás hacer es descargar del CV el proyecto 2.0 para esta sesión (tew-gestioneitorv2_0.zip).
2. Descomprime el proyecto en tu directorio work e impórtalo en eclipse (File/Import) y selecciona la opción del desplegable "General/Existing projects into Workspace".
3. A continuación, vamos a definir la estructura básica de JQuery. Inserta en la siguiente estructura todo el código suministrado en (javascript/functions-crudAlumno.java).

```
$(function(){

//INICIALIZACIÓN
….
Inicializar();
//FUNCIONES

//EVENTOS

});
```

4. Para poder usar JQuery debemos importar la librería JQuery en un servidor externo o disponer de ella en nuestro servidor. Optaremos por importarla de la dirección de google. Añade en la cabecera de crudAlumno.html la importación (antes de la importación de tus funciones javasvascript functions-crudalumno.js)

```
<!--  Enlace al framework javascript JQuery -->
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>
```

5. Lo primero que debemos recordar es que ahora nuestro método de inicialización Inicializar() debe ser invocado en la sección //INICIALIZACIÓN en vez de <body onload="Inicializar()">, además de las inicializaciones que ya había.
6. Vamos a aprovechar las capacidades de JQuery simplificando el acceso a algunos elementos de HTML. Vamos a ir repasando las diferentes funciones y manejadores de eventos que te hemos suministrado en functions-crudAlumnos.java.
7. Adapta los objetos creados como literales en las funciones Add y Edit,

```
{
    ID        : document.getElementById("txtID").value,
    IDUser    : document.getElementById("txtIDUser").value,
    Nombre    : document.getElementById("txtNombre").value,
……….
```

sustituyéndolo por:

```
{
    ID        : $("#txtID").val(),
    IDUser    : $("#txtIDUser").val(),
    Nombre    : $("#txtNombre").val(),
……….. .
```

8. En la función List(), para modificar la tabla "tblList" podemos usar el selector JQuery $("#tblList"):
   a. Sustituye el uso de innerHTML por html() para crear la cabecera (thead)
      `$("#tblList").html()`
   b. Para modificar el cuerpo (tbody) e ir añadiendo las filas debes emplear:
      `$("#tblList tbody").append(El código de cada fila)`
9. Si has realizado correctamente los pasos puedes probar el listado y ver que se visualiza el listado correctamente.
10. El manejador del evento "submit" del formulario "frmCRUDAlumno" vamos a definirlo como (sustituye el que te suminitramos, "botonFormulario" y borra el onclick del formulario en la página HTML crudAlumno.html):

```
$("#frmCRUDAlumnos").bind("submit",function(event){
    if(operation == "A")
            Add();
    else
            Edit();
….
});
```

   a. Si realizamos la llamada a la función List() a continuación, hay que evitar el evento por defecto para que no se pinte dos veces (el evento por defecto para submit es cargar la pagina otra vez y enviar el formulario y reiniciarlo a campos vacíos). Debemos elegir:
      - Hacer la llamada a List() y a event.preventDefault()
      - Dejar que se haga la llamada a List() en la sección de inicializaciones y no incluirlo aquí.
   b. En el caso de la primera opción, el formulario no se va a iniciar a vacío cada vez que pulsemos submit (tendremos que hacerlo nosotros por código):
      `$("#frmCRUDAlumnos")[0].reset();`
   c. Revisa qué pasa con el atributo readonly de txtID si no haces lo siguiente:
      `$("#txtID").removeAttr("readonly");`

11. Según lo anterior añade después del if el siguiente código:

```
//Listamos la tabla del web storage sobre la tabla con los datos actualizados
List();
//Prevenimos el evento por defecto para el evento submit (que consiste en enviar
los datos del formulario y recargar la página
event.preventDefault();
//Limpiamos el formulario,  Hay que recurrir al elemento 0 para aplicar reset.
```

```
$("#frmCRUDAlumnos")[0].reset();
//Restaturamos el atributo readonly para que se pueda escribir
$("#txtID").removeAttr("readonly");
```

12. Para los eventos onclick de las imágenes edit.png y delete.png puedes definir los manejadores del evento sobre la clase de estilo btnEdit y btnDelete usando la notación JQuery respectivamente sustituyendo el "onclick=…" de cada img por:

```
$("#tblList").on("click", ".btnEdit", function(event)
$("#tblList").on("click", ".btnDelete", function(event)
```

13. También debes modificar el acceso a los campos del formulario frmCRUDAlumno en el manejador del evento "click" para la clase btnEdit:

```
$("#txtID").val(alumno.ID);
$("#txtIDUser").val(alumno.IDUser);
$("#txtNombre").val(alumno.Nombre);
$("#txtApellidos").val(alumno.Apellidos);
$("#txtEmail").val(alumno.Email);
$("#txtID").attr("readonly","readonly");
$("#txtNombre").focus();
```

14. Otro aspecto que debe modificarse es el acceso al atributo "Alt" en los manejadores de "click" para edit.png y delete.png:

```
parseInt($(this).attr("alt").replace("Editar", ""));
```

Ten en cuenta que $(this) es el objeto que dispara el evento.

15. En ambos manejadores el uso de event.preventDefault() es innecesario ya que el evento click() no afecta a ready().

16. En el manejador del evento "click" para la clase .btnDelete, después de actualizar el array tbAlumnos en el web storage, es necesario repintarla sobre la vista html mediante la llamada a la función List().
17. Una vez que compruebes que todo funciona correctamente prueba a sustituir las especificación de los manejadores de .btnEdit y .bntDelete:

```
$(".btnEdit").bind("click", function(){ …. });
$(".btnDelete").bind("click", function(){ …. });
```

Deberás comprobar que el evento click en las dos acciones: editar y borrar sólo funciona la primera vez, se pierde a partir de la segunda vez que pulsamos los iconos de editar o borrar. Esto es debido a que al cambiar la estructura HTML de la tabla tblList, JQuery no sabe enlazar los eventos con los nuevos elementos de la tabla. La solución que hemos tomado en el punto 12 consiste en emplear un selector que parta del padre de los elementos que disparan los eventos (<img>): `$("#tblList").on("click", ".btnDelete", function(event){});`, de esta forma JQuery es capaz de reconocer los elementos que dispararán el evento "click". Deja las cosas como estaban.

Paso2: Adaptación a JQuery del geoposicionamiento de alumnos

18. Vamos a añadir los campos Ciudad y Direccion a la tabla "tbAlumnos" del web storage. Esto implicará:

a. Añade los campos Ciudad y Direccion en los objetos creados como literales en las funciones Add yEdit,
b. Añade los campos Ciudad y Direccion en la modificación de los campos del formulario frmCRUDAlumno en el manejador de .bntEdit.
c. c. Añade las columnas Ciudad y Direccion al crear la tabla "tblList" en la función List() (thead y tbody).

Entregables

A continuación se detalla los que será el producto de esta sesión.

1. Proyecto web estático en el que añade a la gestión CRUD basada en JQuery sin los campos de posicionamiento. (tew-gestioneitorv2_5).
2. Proyecto web estático en el que añade a la gestión CRUD con toda la lógica de negocio basada en JQuery además de los campos Ciudad y Dirección (tew-gestioneitorv2_8).