jQueryUI

Estimated time for completion: **30 minutes**

Overview:

In this lab you will add several effects from jQueryUI.

Goals:

- Use effect()
- Use dialog()
- Use datepicker()
- Use autocomplete()

Lab Notes:

The lab uses a movie review website as its base. Registered users can view movies and reviews. Reviewers can write reviews. Administrators can do most anything. The accounts (username and passwords are the same value) for the application are:

- **Registered Users:** user1, user2 and user3
- **Reviewers:** reviewer1, reviewer2 and reviewer3
- **Administrator:** admin

Use jQueryUI effects

On the movie index page you will enhance the movie details dialog with some effects. Also we'll allow the movie poster to be draggable.

Criteria:

- Use effect()
- Use draggable()

Steps:

1. On the movie index page we currently are using some built-in effects from jQuery (such as `slideDown()` and `slideUp()`), but for this part we'll use some of the more sophisticated effects to show and hide the movie details dialog.

a. Open the solution from `~/jQueryUI/before`.

b. Notice that jQueryUI (`jquery-ui-1.9.0.js`) is already in the `~/Scripts` folder (it's included in the MVC project template).

c. Open `~/Views/Shared/_Layout.cshtml` and at the bottom of the file locate where we're already including our JavaScript files (such as jQuery).

d. After the `<script>` for jQuery, add another `<script>` for jQueryUI.

```
<script src="~/Scripts/jQuery-1.8.2.js"></script>
<script src="~/Scripts/jquery-ui-1.9.0.js"></script>
<script src="~/Scripts/jquery.validate.js"></script>
<script src="~/Scripts/MyScript.js"></script>
```

e. We also need to include the CSS for jQueryUI. At the top of this file find the `<head>` element. Add a new `<link>` for the base jQueryUI CSS which is `"~/Content/themes/base/jquery.ui.all.css"`.

```
<link rel="stylesheet"
      href="~/Content/themes/cupertino/jquery.ui.all.css" />
```

2. Now we want to start using effects to the movie details dialog on the movie index page.

   a. Open `~/Script/MyScript.js`.

   b. For intellisense add another `/// <reference>` line at the top of the file to reference jQueryUI.

```
/// <reference path="jquery-1.8.2.js" />
/// <reference path="jquery-ui-1.9.0.js" />
/// <reference path="jquery.validate.js" />
```

   c. Inside of the `document.ready` event handler locate the code that loads the movie details dialog when the table row is clicked.

   d. Notice the call to `slideDown()` to display the movie details dialog. Remove the call to `slideDown()`.

   e. Add a call to `show()`.

   f. Also add a call to `effect()` and pass as `"slide"` as the argument.

   g. Notice the code to `append()` the `<button>` to the dialog. Inside the `<button>` `click` event is a call to `slideUp()`. Remove the call to `slideUp()`.

   h. Add a call to `effect()` and pass one of the other effects from jQueryUI (such as `"drop"` or `"explode"`).

```
$(".movies")
    .on("click", "tbody tr", function () {
        var me = $(this);
        var href = me.find("a").attr("href");
        if (href) {
            $("#movieDetails").load(href, function () {
                $(this)
                    .show().effect("slide")
                    .append(
                        $("<button>Close</button>")
                            .click(function () {
                                $("#movieDetails")
                                    .effect("drop");
                            })
                    );
            });
        }
    })
```

3. Run the application and browse to the movie listing page. Try out the new effects on the movie details dialog. Don't forget to Ctrl-F5 to load the updated JavaScript.
4. Just for giggles, make the movie poster image draggable.
   a. Inside of the `document.ready` event handler use jQuery to query for the element whose `id` is `"imgContainer"`.
   b. On the matched item call `draggable()`.

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

5. Run the application and visit the movie index. Mouse over a row in the table and then try to drag the movie poster around the screen.

Use jQueryUI dialog

Use jQueryUI's dialog for the movie details dialog instead of our home-grown approach.

Criteria:

- Use dialog()

Steps:

1. For the movie details dialog on the movie index page we'll use the jQueryUI dialog rather than the home-grown approach.
   a. Open `~/Script/MyScript.js`.
   b. Locate the `click` event handler for the table rows that displays the movie details dialog.
   c. Notice inside of the `click` event handler is a call to `load()` with a callback. Inside the load callback on the `$(this)` selector remove the calls to `show()`,

effect() and append(). In other words, keep the call to $(this) and remove everything else.

d. On the returned jQuery object from $(this), call dialog().

```
$(".movies")
    .on("click", "tbody tr", function () {
        var me = $(this);
        var href = me.find("a").attr("href");
        if (href) {
            $("#movieDetails").load(href, function () {
                $(this).dialog();
            });
        }
    })
```

2. Run the application and browse to the movie index page. Click a row in the table and see how the jQueryUI modal looks.

   Doesn't look very nice, does it? The reason is we still have the CSS for the home-grown dialog. Let's fix this now.

   a. Open ~/Content/Site.css.

   b. Locate the CSS rule for #movieDetails.

   c. Remove all the styles for this rule except for the display:none.

```
#movieDetails
{
    display:none;
}
```

3. Now run the application again and the movie details dialog should look better. Don't forget to Ctrl-F5 in the browser to reload the updated style sheet.

   Well, it is somewhat better, but the default width of the dialog is too narrow. We'll adjust that now.

   a. Back in ~/Scripts/MyScript.js.

   b. Pass a settings object to the dialog() method with a property called width and a value of "800". Also set a property called modal with a value of true.

```
$(".movies")
    .on("click", "tbody tr", function () {
        var me = $(this);
        var href = me.find("a").attr("href");
        if (href) {
            $("#movieDetails").load(href, function () {
                $(this).dialog({
                        width: "800", modal: true
```

```
                    });
            });
        }
    })
```

4. Run the application again and now the modal should look better. Also, notice it's now a modal dialog.
5. The one last thing to configure on the dialog is the title.
    a. Add the `"title"` property on the settings object passed to `dialog()`. Set the value to the name of the movie that was clicked in the table. The movie name is the text contents of the `<a>`.

```
$(".movies")
    .on("click", "tbody tr", function () {
        var me = $(this);
        var href = me.find("a").attr("href");
        if (href) {
            $("#movieDetails").load(href, function () {
                $(this)
                    .dialog({
                        width: "800", modal: true,
                        title: "Movie Details : " + me.find("a").text()
                    });
            });
```

6. Run the application and ensure the title of the modal now contains the movie name.

Use jQueryUI datepicker

Logged in as an admin you can edit almost any data in the movie review application. This includes the movie review information which contains a date. You will use jQueryUI's datepicker to enhance the movie review date field.

Criteria:

- Use datepicker()

Steps:

1. Run the application and login as an administrator. Notice the "Admin" menu. Click it and then click the "Reviews" menu. Click on "Edit" for any of the reviews. Notice the "review date" field – this is the field that we want to enhance with jQueryUI's datepicker. To identify the element it is wrapped in a `<div>` with a class of `"editor-date"`. We'll now add a datepicker to that input element.
    a. Open `~/Script/MyScript.js`.

   b. Inside of the `document.ready` event handler, use the jQuery function to query for `<input type="text">` elements that are descendants of an element with the `class` of `"editor-date"`.

   c. On the matched item call `datepicker()`.

```
$(".editor-date input[type=text]").datepicker();
```

2. Run the application and edit a review. When you click in the review date you should see the date picker effect.

Use jQueryUI autocomplete

You will add autocomplete to the movie search page. The search term input will provide movie title suggestions from the server.

Criteria:

- Use autocomplete()

Steps:

1. In this part you will use autocomplete to provide a dynamic suggestion list from the server. The page we will add this to is the movie search page. As the user types into the textbox they will be provided with the list of movies that match the search term. To support this, the server has an endpoint at `"/Movies/Home/SearchAutoComplete"` that returns the JSON data to populate the autocomplete.

   a. Open `~/Script/MyScript.js`.

   b. Inside of the `document.ready` event handler, use the jQuery function to query for `<input type="text">` elements that are descendants of an element whose `id` is `"movieSearch"`.

   c. On the matched item call `autocomplete()`. Pass a settings object as a parameter with a property called `source` and a value of `"/Movies/Home/SearchAutoComplete"`.

```
$("#movieSearch input[type=text]").autocomplete({
    source: "/Movies/Home/SearchAutoComplete"
});
```

2. Run the application and visit the movie search page. Type some text into the input and check that the suggestion list appears.

Download and use a jQueryUI theme

The default theme from jQueryUI is a grey styled color. Download a different theme and use it.

Criteria:

- Use a custom jQueryUI theme

Steps:

1. Download a custom theme now and use it in your application.

   If you are unable to visit the website then there are already a couple of custom jQueryUI themes already downloaded in the `~/Content/themes` directory.

   a. Go to http://jqueryui.com/themeroller/ and customize a theme.

   b. Download the theme and unzip it.

   c. In the unzipped directory locate the "css" subdirectory and inside of that you will find your theme directory. Copy this to `~/Content/themes`.

   d. Open `~/Views/Shared/_Layout.cshtml` and locate the styles included in the `<head>`.

   e. Change the `<link>` that references `"~/Content/themes/cupertino/jquery.ui.all.css"` and change it to your new theme.

2. Run the application and test your new theme.

Solutions:

The final solution for this lab is available in the `~/after` directory.