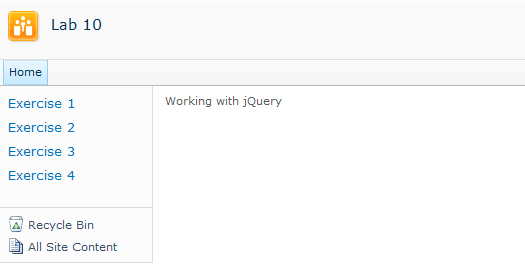
## Working with jQuery

**Lab Time**: 45 minutes

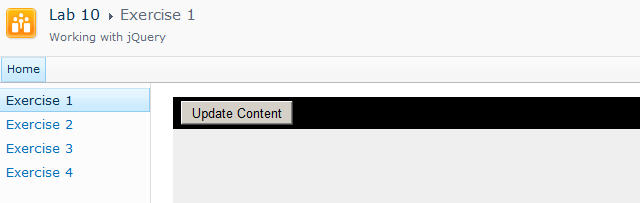
### Exercise 1: Getting Started with jQuery

In this exercise you will move through the basic steps of linking a Web page to the core jQuery library and using jQuery to register an event handler. You will also get experiences calling the jQuery function to create jQuery objects as well as calling some of the convenient methods exposed by jQuery objects.

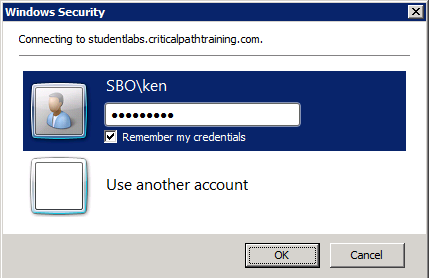
1. In the browser, navigate to the Brand Camp labs site collection and open the site at **http://[[COLLAB-SITE]]**. Next, navigate to the child site for **Lab 10** by using the global navigation menu of the top-level site. You should see four links in the Quick Launch Bar to navigate to the pages you will be working on through the exercises in this lab.



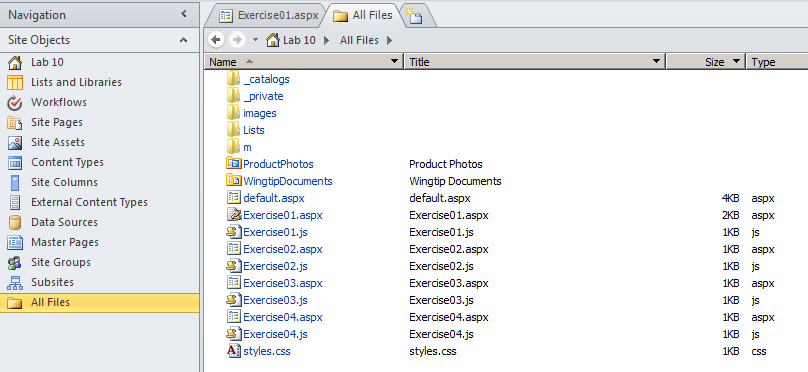
1. In the browser, navigate to the page for **Exercise 1** by clicking the navigation link in the Quick launch. The page contains a command button that currently has no behavior behind it. Your first task in Exercise 1 is to write an event handler and register it to execute when the user clicks the button.



1. Launch SharePoint Designer 2010 and open the site at **http://[[COLLAB-SITE]]/Lab10**. If you are prompted to log in, enter the site collection owner as **[[AD\_DOMAIN]]\[[USERNAME]]**. For example, if you were supplied the user account login credentials for Ken Sanchez with a user account name **ken** and a domain name **SBO,** then theaccount name for logging in should be entered as **SBO\ken**. The password for all accounts should be set to **Password1**.



1. Once the site has opened in SharePoint Designer, click the **All Files** link at the bottom of the **Site Objects** section. You should be able to see the files for pages named **Exercise01.aspx**, **Exercise02.aspx**, **Exercise03.aspx** and **Exercise04.aspx**. For example, the page named **Exercise01.aspx** has an associated JavaScript file named **Exercise01.js**.



At this point, you should have the **Lab 10** site open in both the browser as well as the SharePoint Designer. Leave both of these open as you will be moving back and forth between them for all the remaining exercises of this lab.

1. In SharePoint Designer, open the page named **Exercise01.aspx** in advanced edit mode. Locate the placeholder for **PlaceHolderAdditionalPageHead**. You can observe that thisplaceholder contains a link to a style sheet named **styles.css** as well as a script link to **Exercise01.js**. In a later step you will add a second script link to reference the core jQuery library.

<asp:Content ContentPlaceHolderId="PlaceHolderAdditionalPageHead" runat="server">

<link rel="stylesheet" type="text/css" href="styles.css" />

<!-- TO DO: Add script link to core jQuery library here -->

<script src="Exercise01.js" type="text/javascript" ></script>

</asp:Content>

1. Examine the placeholder for **PlaceHolderMain**. You should observe that there is HTML content inside **PlaceHolderMain** that defines an input element of type button with an **id** of **cmdAddContent** as well as an empty div element with an **id** of **Div1**.

<asp:Content ContentPlaceHolderId="PlaceHolderMain" runat="server">

<div id="MainContentDiv" >

<div id="Toolbar" >

<input id="cmdMyHandler" type="button" value="Update Content" />

</div>

<div id="ContentAreaDiv" >

<div id="Div1"/>

</div>

</div>

</asp:Content>

There are several different ways in which you can link to the appropriate JavaScript files that contain the core jQuery library. For example, you can link to the jQuery library from Content Delivery Networks (CDNs) provided by Microsoft or Google using simple script links containing the following publically accessible URLs.

<script src="**http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.5.1.js** " type="text/javascript" ></script>

<script src="**http://ajax.googleapis.com/ajax/libs/jquery/1.5.1/jquery.min.js**" type="text/javascript" ></script>

It is also possible to copy the JavaScript file for the jQuery library into a SharePoint environment and then link to it from that location. This is the approach that will be used in this lab. The Collaboration Site Collection you are using in this lab already contains a copy of the JavaScript file for core jQuery library that is located at the following path, which is relative to the top-level site.

**/js/jquery-1.5.1.min.js**

In the next step you will add a dynamic script link to **Exercise01.aspx** to reference this JavaScript file. Once you have done this, you can begin using jQuery in the JavaScript code behind the page.

1. Inside for **PlaceHolderAdditionalPageHead** in **Exercise01.aspx**, add an instance of the **SharePoint:ScriptLink** control. Add the **runat** attribute with a value of **server**. Add a **Defer** attribute with a value of **false** to ensure the JavaScript file is downloaded synchronously. Add a **Name** attribute with a value of **~sitecollection/js/jquery-1.5.1.min.js**. The **Name** attribute of the **SharePoint:ScriptLink** control is used to provide the URL to the JavaScript file. Note that when SharePoint processes the page with this control, it replaces the dynamic token of **~sitecollection** with the base URL of the top-level site for the current site collection.

<asp:Content ContentPlaceHolderId="PlaceHolderAdditionalPageHead" runat="server">

<link rel="stylesheet" type="text/css" href="styles.css" />

<!-- script link to core jQuery library -->

<SharePoint:ScriptLink

runat="server"

Defer="false"

Name="~sitecollection/js/jquery-1.5.1.min.js" />

<script src="Exercise01.js" type="text/javascript" ></script>

</asp:Content>

Note that the page **Exercise01.aspx** contains a **Register** directive at the top of the page with **tagprefix** attribute of **SharePoint**. You should take note that the **SharePoint:ScriptLink** control would not work correctly and would cause the hosting page to fail if this **Register** directive were not present in the page.

1. After you have added the **SharePoint:ScriptLink** control, save your changes to **Exercise01.aspx**. You can now close **Exercise01.aspx** because you will not be required to make any more modifications to this file.
2. In SharePoint Designer, open **Exercise1.js**. Note that this JavaScript file will initially be empty. Over the next few steps, you will add JavaScript code to **Exercise1.js** to provide **Exercise01.aspx** with some custom behavior.
3. Add the jQuery document ready event handler to the top of **Exercise1.js**.

$(function () {

});

1. Add a new function named **myHandler** just below the document ready event handler. This function will be used as an event handler so you should define it with a single parameter named **event**.

$(function () {

});

function myHandler(event) {

}

1. Now you will implement the document ready event handler. This will register the **myHandler** function as an event handler for the **click** event of the command button with the id of **cmdMyHandler**. Accomplish this by calling the jQuery function and passing a selector of **#cmdMyHandler** to return a jQuery object which wraps the command button. Call the **click** method on the jQuery object that is returned, and pass the **myHandler** function as a parameter.

$(function () {

$("#cmdMyHandler").click(myHandler);

});

function myHandler(event) {

}

1. At this point, your code will register the **myHandler** function as an event handler that fires when a user clicks the button. Implement **myHandler** to write a text value of **Hello jQuery** into the div element with an id if **Div1**. Accomplish this by calling the jQuery function and passing a selector of **#Div1** toreturn a jQuery object which wraps the div element. Call the **html** method on the jQuery object that is returned.

$(function () {

$("#cmdMyHandler").click(myHandler);

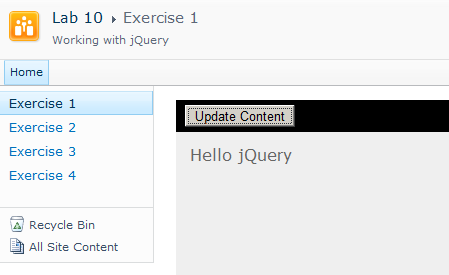
});

function myHandler(event) {

$("#Div1").html("Hello jQuery");

}

1. Now save your work by saving **Exercise01.js** in the SharePoint Designer.
2. Return to the browser and refresh the page named **Exercise01.aspx**. Note that you must first refresh the page to test your work and to ensure that the JavaScript you just wrote is downloaded into the browser. Once the page has refreshed, click the command button. You should be able to see your message added to the page contents.



1. As a final step, use jQuery to add some CSS styling to **Div1**. More specifically, make the font color white and the background color black. Add a padding value of 4px and a margin of 10px. Give the div element a fixed width of 240px. Make the text inside the div element align to the center and make the font a size of 18px. You can accomplish this using the call to the **css** method shown in the following code listing. When you are done, save your work by saving **Exercise01.js**.

$(function () {

$("#cmdMyHandler").click(myHandler);

});

function myHandler(event) {

$("#Div1")

.html("Hello jQuery")

.css({

"color": "white",

"backgroundColor": "black",

"padding": "4px",

"margin": "10px",

"width": "240px",

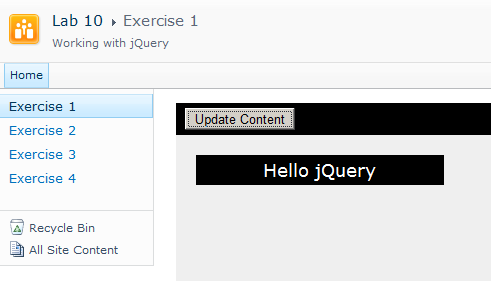
"textAlign": "center",

"fontSize": "18px"

});

}

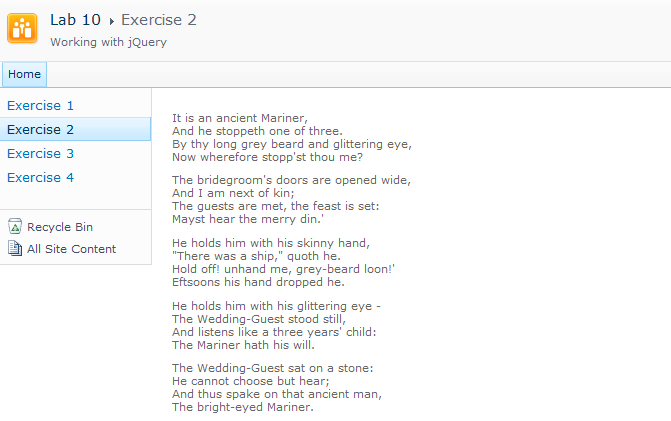
1. Return to the browser and refresh the page named **Exercise01.aspx**. Click the command button. You should be able to see your message added to the page contents.



### Exercise 2: Working with jQuery Selectors

In this exercise you will add jQuery code to a page to add custom formatting to the paragraphs inside.

1. In the browser, navigate to the page for **Exercise 2** by clicking the navigation link in the Quick launch. The page contains text with a large collection of paragraphs from a popular poem.



1. In SharePoint Designer, open **Exercise02.aspx** in advanced edit mode. Locate the placeholder for **PlaceHolderAdditionalPageHead**. You can observe that thisplaceholder contains a link to a style sheet named **styles.css** as well as script links to the jQuery library and **Exercise02.js**. This means the page has been preconfigured for you to write jQuery code in **Exercise02.js**.
2. Examine the contents of **PlaceHolderMain**. You should see that there is an outer div element with an **id** value of **MainContentArea**. Inside this div element, there are quite a few **p** elements with the content for a popular poem you might have read before.

<asp:Content ContentPlaceHolderId="PlaceHolderMain" runat="server">

<div id="MainContentArea" >

<p>It is an ancient Mariner,<br />

And he stoppeth one of three.<br />

By thy long grey beard and glittering eye,<br />

Now wherefore stopp'st thou me?</p>

<p>The bridegroom's doors are opened wide,<br />

And I am next of kin;<br />

The guests are met, the feast is set:<br />

Mayst hear the merry din. </p>

<p>He holds him with his skinny hand,<br />

"There was a ship" quoth he.<br />

"Hold off! unhand me, grey-beard loon!"<br />

Eftsoons his hand dropped he.</p>

<p>He holds him with his glittering eye -<br />

The Wedding-Guest stood still,<br />

And listens like a three years' child:<br />

The Mariner hath his will.</p>

</div>

</asp:Content>

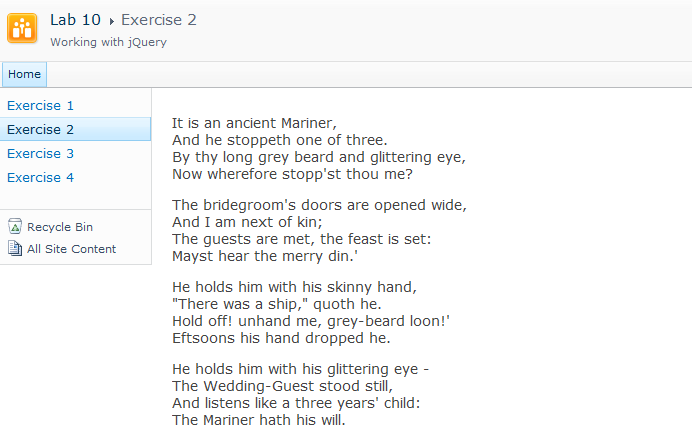
1. Open **Exercise02.js**.
2. At the top of **Exercise02.js**, add a document ready event handler and implement it to add CSS style rules to every **p** element that exists within the div element with the **id** of **MainContentArea**. In particular, style the **p** elements with a font color of **#444** and a font size of **14px**.

$(function () {

$("#MainContentArea p").css({ "color": "#444", "font-size": "14px" });

});

1. Save your changes to **Exercise02.js**.
2. Return to the browser and refresh page **Exercise02.aspx**. Your page should show the paragraphs of the poem with a large font and different color.



1. Add more code to the document library event handler to give a different formatting to the odd paragraphs. In particular, style the odd **p** elements with a font color of **#888** and a left margin of **14px**.

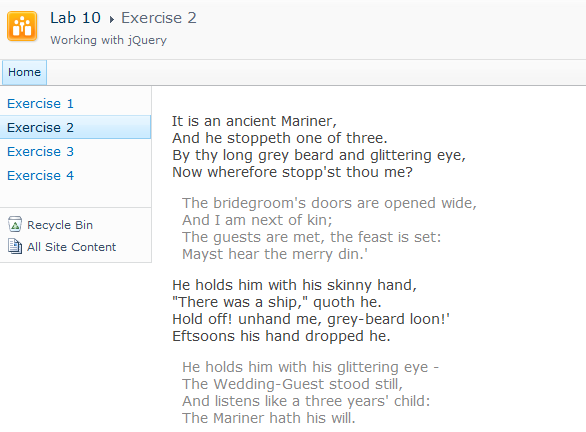
$(function () {

$("#MainContentArea p").css({ "color": "#444", "font-size": "14px" });

$("#MainContentArea p:odd").css({"color": "#888", "margin-left": "10px"});

});

1. Save your changes to **Exercise02.js**.
2. Return to the browser and refresh page **Exercise02.aspx**. Your page should show the paragraphs of the poem with a large font and different color.



1. Add more code to the document library event handler to give a different formatting to the first paragraph. In particular, style the first **p** element with a font color of **#000** and a left margin of **18px**.

$(function () {

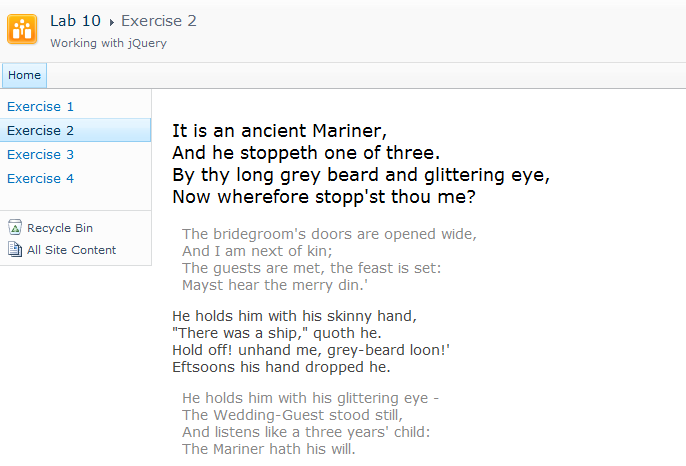
$("#MainContentArea p").css({ "color": "#444", "font-size": "14px" });

$("#MainContentArea p:odd").css({"color": "#888", "margin-left": "10px"});

$("#MainContentArea p:first").css({ "color": "#000", "font-size": "18px" });

});

1. Save your changes to **Exercise02.js**.
2. Return to the browser and refresh page **Exercise02.aspx**. Your page should show the paragraphs of the poem with a large font and different color.



1. As a final task, you will add an event handler that allows a user to change the format of a paragraph by clicking on it. The main idea is that a user can click on a paragraph after reading it to change its formatting so the text appears crossed out. Begin by adding a new function named **CrossItOut** and then add a line to the bottom of the document ready event handler to register this function to handle the **click** event on each **p** element inside the **MainContentArea** div.

$(function () {

$("#MainContentArea p").css({ "color": "#444", "font-size": "14px" });

$("#MainContentArea p:odd").css({"color": "#888", "margin-left": "10px"});

$("#MainContentArea p:first").css({ "color": "#000", "font-size": "18px" });

$("#MainContentArea p").click(CrossItOut);

});

function CrossItOut(event) {

}

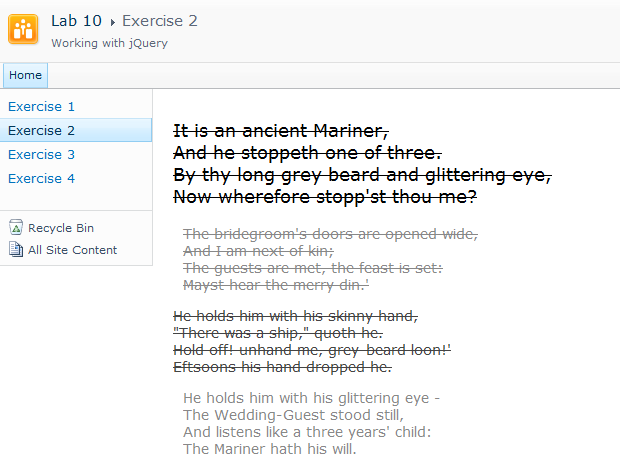
1. Implement the **CrossItOut** function so that it formats the paragraph that was clicked with the CSS attribute named **text-decoration** which has a value of **line-through**. Note you can access the HTML element with the current **p** element using the **this** keyword. If you pass the **this** keyword to the jQuery function, it will return a jQuery object that allows you to call the **css** method.

function CrossItOut(event) {

$(this).css({"text-decoration": "line-through"});

}

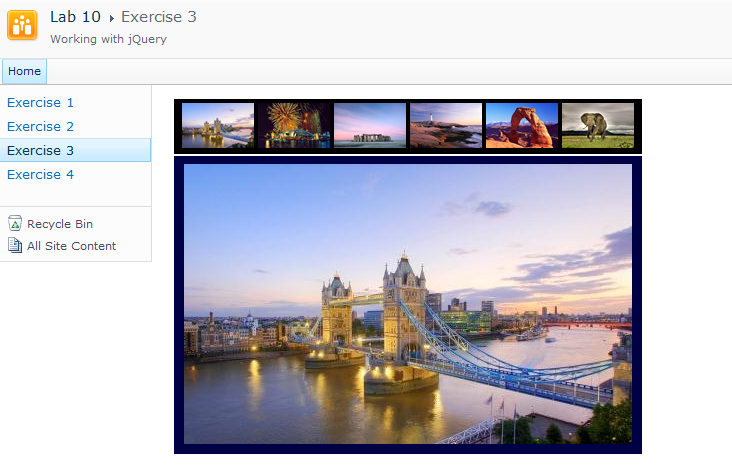
1. Save your changes to **Exercise02.js**.
2. Return to the browser and refresh page **Exercise02.aspx**. Now you should be able to click on a paragraph to give it strikethrough formatting.



### Exercise 3: Creating an Image Toolbar

In this exercise you will use jQuery to register event handlers on a set of **img** elements, creating a toolbar that responds to mouseover events.

1. In the browser, navigate to the page for **Exercise 3** by clicking the navigation link in the Quick launch. The page contains a toolbar with a small set of images. Below there is a larger image. In this exercise, you will add jQuery code to register event handlers that allow the user to hover the mouse over a small image in the toolbar to display it the larger view below.



1. In SharePoint Designer, open **Exercise03.aspx** in advanced edit mode. Locate the placeholder for **PlaceHolderAdditionalPageHead**. You can observe that thisplaceholder contains a link to a style sheet named **styles.css** as well as script links to the jQuery library and **Exercise03.js**. This means the page has already been preconfigured for you to write jQuery code in **Exercise03.js**.
2. Examine the contents of **PlaceHolderMain**. You should see that there is a div element with an **id** of **ImageToolbar** which contain a set of **img** elements with **src** attribute that reference JPG files in the **images** folder of the current site. Below the **ImageToolbar** div there is another div element with an **id** of **PictureAreaDiv** which contains a **img** element with an id of **picture**.

<asp:Content ContentPlaceHolderId="PlaceHolderMain" runat="server">

<div id="ImageToolbar" >

<img src="images\LondonBridge.jpg" alt="LondonBridge" />

<img src="images\Sidney.jpg" alt="Sidney" />

<img src="images\Stonehenge.jpg" alt="Stonehenge" />

<img src="images\Lighthouse.jpg" alt="Lighthouse" />

<img src="images\Rock.jpg" alt="Rock" />

<img src="images\Elephant.jpg" alt="Elephant" />

</div>

<div id="PictureAreaDiv" >

<img id="picture" src="images\LondonBridge.jpg" alt="picture" />

</div>

</asp:Content>

1. Open **Exercise03.js**.
2. At the top of **Exercise03.js**, add a document ready event handler and an event handler function named **onMouseOver**. Implement the document ready event handler to register **onMouseOver** as an event handler. It will handle all the **mouseover** events for every **img** element inside the **ImageToolbar** div.

$(function () {

$("img", "#ImageToolbar").mouseover(onMouseOver);

});

function onMouseOver(event) {

}

1. Implement **onMouseOver** to retrieve the **src** attribute from the **img** element that was clicked, and then assign this value to the **src** attribute of the **img** element with the id of **picture**.

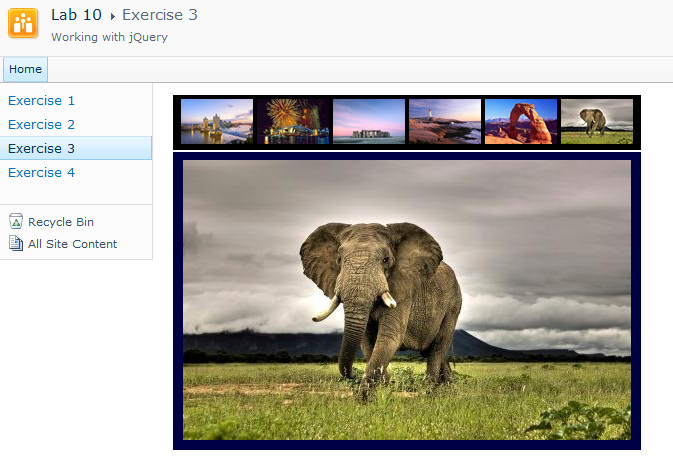
function onMouseOver(event) {

var path = $(this).attr('src');

$("#picture").attr('src', path);

}

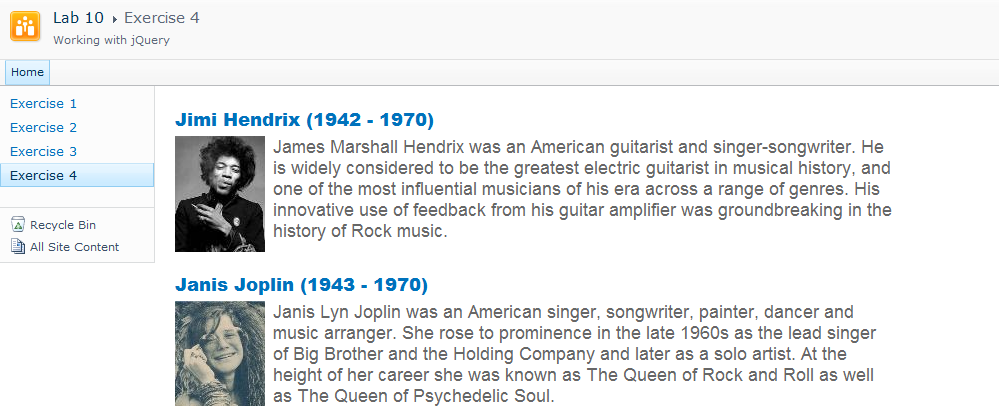
1. Save your changes to **Exercise03.js**.
2. Return to the browser and refresh page **Exercise03.aspx**. Now you should be able to hover the mouse over one of the smaller images in the image toolbar and see that image appear below in the larger img element.



### Exercise 4: Leveraging the jQuery UI Accordion

In this exercise you will use the jQuery UI library to create an advanced user interface experience using the Accordion component.

1. In the browser, navigate to the page for **Exercise 4** by clicking the navigation link in the Quick launch. The page contains a collection of div element with information and a photo of famous rock stars.



1. In SharePoint Designer, open **Exercise04.aspx** in advanced edit mode. Locate the placeholder for **PlaceHolderAdditionalPageHead**. You can observe that thisplaceholder contains a link to a style sheet named **styles.css** as well as a **SharePoint:CssRegistration** control which adds a link to a style sheet used by components in the jQuery UI library. There are also two **SharePoint:ScriptLink** controls which add script links to the code jQuery library and the jQuery UI library. At the bottom there is a script link to **Exercise04.js.** This means the page has already been preconfigured for you to write jQuery UI code in **Exercise04.js**.

<asp:Content ContentPlaceHolderId="PlaceHolderAdditionalPageHead" runat="server">

<link rel="stylesheet" type="text/css" href="styles.css" />

<SharePoint:CssRegistration

name="<% $SPUrl:~SiteCollection/css/ui-lightness/jquery-ui-1.8.13.custom.css%>"

After="corev4.css"

runat="server"/>

<SharePoint:ScriptLink

runat="server"

Defer="false"

Name="~sitecollection/js/jquery-1.5.1.min.js" />

<SharePoint:ScriptLink

runat="server"

Defer="false"

Name="~sitecollection/js/jquery-ui-1.8.13.custom.min.js" />

<script src="Exercise04.js" type="text/javascript" ></script>

</asp:Content>

1. Examine the contents of **PlaceHolderMain**. You should see that there is a div element with an **id** of **ImageToolbar** which contain a set of **img** elements with **src** attribute that reference JPG files in the **images** folder of the current site. Below the **ImageToolbar** div there is another div element with an **id** of **PictureAreaDiv** which contains a **img** element with an id of **picture**.

<asp:Content ContentPlaceHolderId="PlaceHolderMain" runat="server">

<div id="Rockstars" >

<div>

<h3><a href="#">Jimi Hendrix (1942 - 1970)</a></h3>

<div>

<div style="float:left; margin-right:4px;">

<img src="images/jimi\_hendrix.jpg" />

</div>

<p>James Marshall Hendrix was ....</p>

<div style="clear:both;" />

</div>

</div>

<div>

<h3><a href="#">Janis Joplin (1943 - 1970)</a></h3>

<div>

<div style="float:left; margin-right:4px;">

<img src="images/janis\_joplin.jpg" />

</div>

<p>Janis Lyn Joplin was...</p>

<div style="clear:both;" />

</div>

</div>

</div>

</asp:Content>

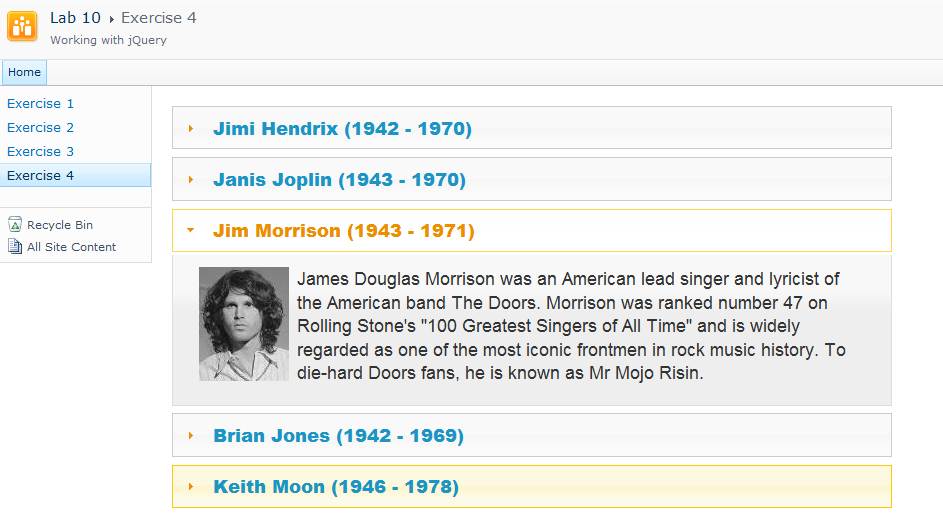
1. Open **Exercise04.js**.
2. At the top of **Exercise04.js**, add a document ready event handler that calls the jQuery function to retrieve a jQuery object that wraps the div with an id of **Rockstars**. Call the **accordion** method on the jQuery object, passing a new JavaScript object that has a header property with a value of **h3**.

$(function () {

$("#Rockstars").accordion({ header: "h3" });

});

1. Save your changes to **Exercise04.js**.
2. Return to the browser and refresh page **Exercise04.aspx**. The page content with the information about rock stars should now be displayed within an accordion component.



As a final task in this lab you will experiment by switching out different jQuery themes to see how it affects the accordion component. The page you have been working with is currently using a jQuery theme named **ui-lightness**. The main CSS file and images for the **ui-lightness** theme as well as three other jQuery themes named **redmond**, **cupertino** and **black-tie** have been copied into the top-level site of the Collaboration Site Collection at the following paths.

**/css/ui-lightness/jquery-ui-1.8.13.custom.css**

**/css/redmond/jquery-ui-1.8.13.custom.css**

**/css/cupertino/jquery-ui-1.8.13.custom.css**

**/css/black-time/jquery-ui-1.8.13.custom.css**

1. Return to **Exercise04.aspx** in SharePoint Designer and locate the **SharePoint:CssRegistration** control in **PlaceHolderAdditionalPageHead**. You should be able to see that the name attribute has a value that contains a path with the folder named ui-lightness.

<SharePoint:CssRegistration

name="<% $SPUrl:~SiteCollection/css/ui-lightness/jquery-ui-1.8.13.custom.css%>"

After="corev4.css"

runat="server"/>

1. Update the name attribute to use the **redmond** theme instead of the **ui-lightness** theme.

<SharePoint:CssRegistration

name="<% $SPUrl:~SiteCollection/css/redmond/jquery-ui-1.8.13.custom.css%>"

After="corev4.css"

runat="server"/>

1. Update the name attribute to use the **redmond** theme instead of the **ui-lightness** theme.
2. Save your changes to **Exercise04.js**.
3. Return to the browser and refresh page **Exercise04.aspx**. The accordion component should now display with different colors.
4. Experiment by changing the page to use the two other jQuery themes named **cupertino** and **black-tie**. Choose the theme that you like best.