

Microsoft® Virtual Labs

Using New AJAX and Enhanced
Layout Standards Support in
Internet Explorer 8

Table of Contents

Using New AJAX and Enhanced Layout Standards Support in Internet Explorer 8	1
Exercise 1 Implement traditional browser functionality in AJAX-based applications	2
Exercise 2 Storing data in the DOM Store	4

Using New AJAX and Enhanced Layout Standards Support in Internet Explorer 8

Objectives

After completing this lab, you will be better able to:

- Restore traditional browser functionality to AJAX based applications
- Use IE8's DOM Storage and Connectivity events to store data offline and submit it to a live server once connectivity has been restored

Overview

This lab is intended for software developers who want to improve the usability of their AJAX-based applications. Developers will also learn about the DOM Storage and Connectivity events and will implement that will allow users to work both on and offline with web-based applications.

Scenario

As a developer for Northwind Traders, you've been tasked with adding new functionality to the web based customer database. Northwind Traders maintains a database of all the customers and distributors that is available to the sales staff via intranet. Currently, the customer database web pages make use of AJAX but lack traditional browser functionality such as back and forth buttons and URLs for specific pages. Furthermore, management would like for the sales staff to be able to add new customer information while they are traveling.

Northwind Traders' IT staff has deployed Internet Explorer 8 to all of the company's staff. Using IE8's new AJAX navigation features, you will return functionality to the browser's back and forward buttons. You will also restore functionality to the address bar so that staff members can send direct links to customer information to each other.

In addition to these updates, you will also make use of IE8's connectivity events to allow the sales staff to enter new customer information without having access to an Internet connection. The data will be stored in IE8's client-side DOM store and transmitted to the company's server once Internet connectivity is restored. This feature will be immensely helpful to sales staff that are flying to and from meetings and conferences.

To save time, certain components have been pre-installed and pre-configured. For the purposes of this lab, you will be developing locally and using the local copy of IIS for testing.

Estimated Time to Complete This Lab

35 Minutes

Computers used in this Lab



Win7DSK


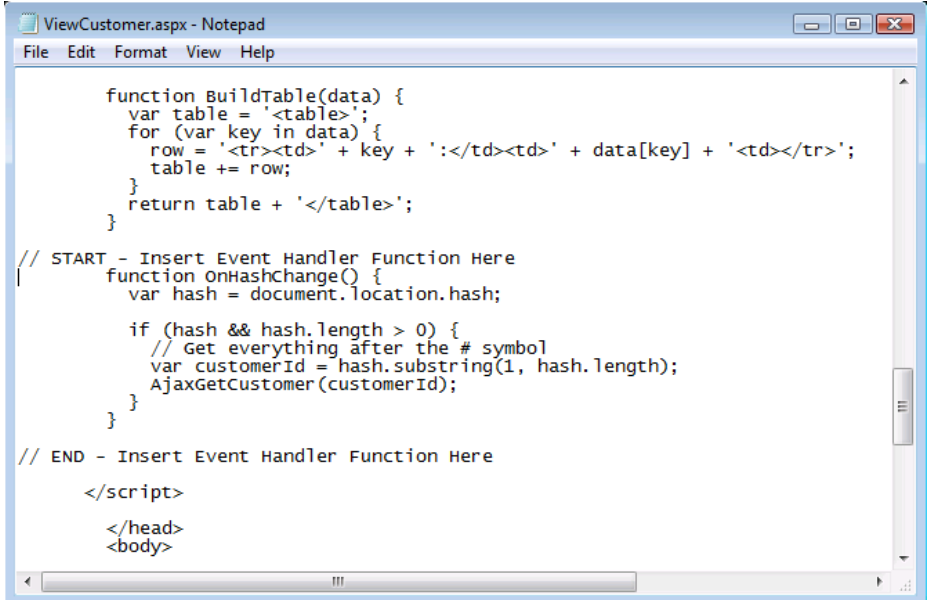
Log on to this machine using the account **Win7-PC\DEVLabUser** with the password: **DEVP@ssword**

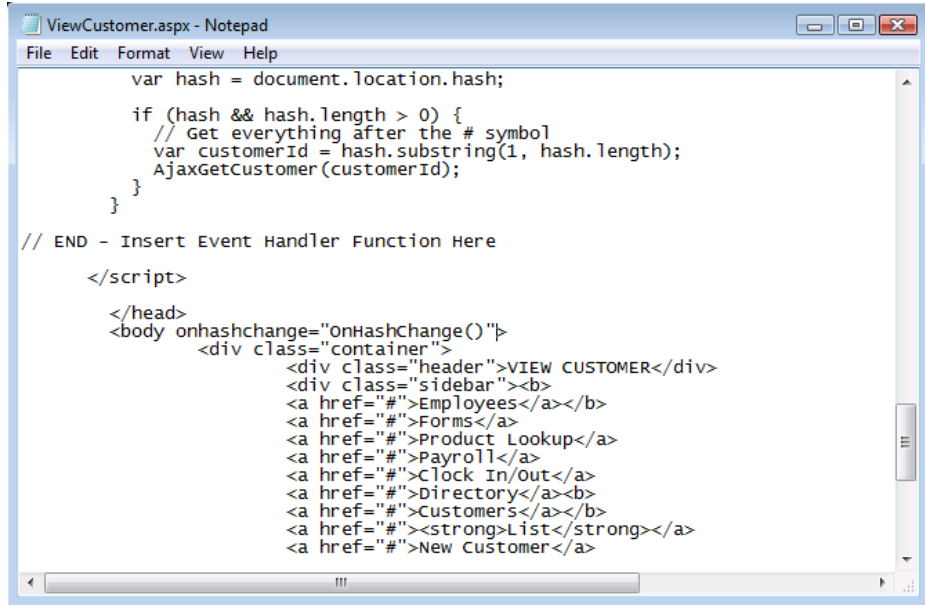
Exercise 1

Implement traditional browser functionality in AJAX-based applications

Scenario

In this exercise, you will use the Notepad application to restore traditional browser navigation functionality to an AJAX based application.

Tasks	Detailed Steps
<p>Complete the following tasks on:</p>  <p>1. Creating the Event Handler</p>	<p>a. An event handler for the onhashchange event needs to be created. Open ViewCustomer.aspx in notepad. Run Notepad as Administrator and open ViewCustomer.aspx located at C:\inetpub\wwwroot\DTLab. Below is an example implementation of how you can handle the onhashchange event. Insert the following function into ViewCustomers.aspx between the “Start” and “End” comments just above the <body> section. (see Figure)</p> <pre>function OnHashChange() { var hash = document.location.hash; if (hash && hash.length > 0) { // Get everything after the # symbol var customerId = hash.substring(1, hash.length); AjaxGetCustomer(customerId); } }</pre> 



Tasks	Detailed Steps
<p>2. Applying the Event Handler</p>	<p>a. Attach the event handler to the onhashchange event. You do this by adding an onhashchange event attribute to the body tag. For example, your body tag might look something like the following:</p> <pre data-bbox="506 304 1398 331"><body onhashchange="OnHashChange()"></pre> <p>b. Insert the above attribute into the body tag and save as ViewCustomers.aspx, overwriting the previous file (see Figure).</p>  <p>The screenshot shows a Notepad window with the following code:</p> <pre> var hash = document.location.hash; if (hash && hash.length > 0) { // Get everything after the # symbol var customerId = hash.substring(1, hash.length); AjaxGetCustomer(customerId); } // END - Insert Event Handler Function Here </script> </head> <body onhashchange="OnHashChange()"> <div class="container"> <div class="header">VIEW CUSTOMER</div> <div class="sidebar"> Employees Forms Product Lookup Payroll Clock In/Out Directory Customers List New customer </pre>
<p>3. Reviewing the Functionally</p>	<p>a. Return to ViewCustomers.aspx in IE8 and refresh the page. Now click the next button several times. Now click the back button and notice how it returns to the previous customer.</p>

Exercise 2

Storing data in the DOM Store

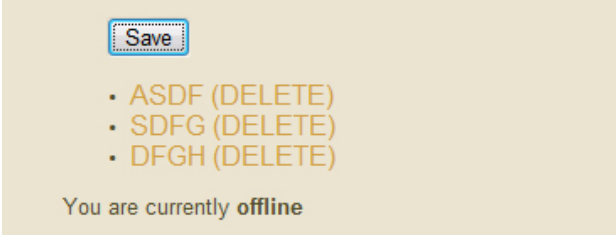
Scenario

In this exercise, you will learn about the DOM Store and how to store and retrieve data on the user's computer. You will use the Notepad application to create scripts that can access the client side DOM Storage.

Tasks	Detailed Steps
<p>Complete the following tasks on:</p>  <p>1. Creating the Event Handlers</p>	<p>a. Open NewCustomer.aspx by typing in "http://localhost/Newcustomer.aspx" and you will see the form which is being set up to use localStorage. (see Figure)</p>  <p>b. An event handler for the online and offline events need to be created. Below is an example implementation of how you can handle the online and offline events. This function is found in the scripts directory of the Lab Folder in the file localStorage.js.</p> <pre data-bbox="509 1503 1398 1879"> /* Event handler called when going * from offline to online */ function onLine() { \$("#status").text("online"); if (localStorage.customers) { var customers = JSON.parse(localStorage.customers); if (!isEmpty(customers)) { \$("#saveLocalStorage").show(); } } } </pre>

Tasks	Detailed Steps
	<pre> } /* Event handler called when going * from online to offline */ function offLine() { \$("#status").text("offline"); \$("#saveLocalStorage").hide(); } </pre>
<p>2. Applying the Event Handler</p>	<p>a. Attaching the event handler to the online and offline events. You do this by adding the online and offline event attributes to the body tag. For example your body tag might look something like the following (see Figure):</p> <pre> <body ononline="onLine();" onoffline="offLine();"> </pre> <pre> <meta http-equiv="X-UA-Compatible" content="IE=8" /> <script src="scripts/jquery-1.2.6.min.js" type="text/javascript"></script> <script src="scripts/localstorage.js" type="text/javascript"></script> </head> <body ononline="onLine();" onoffline="offLine();"> <h2> New Customer</h2> <p> Fill in the form below to create a new customer.</p> </pre> <p>b. Open NewCustomer.aspx in Notepad and modify the body tag to match the code example above.</p>
<p>3. Saving to localStorage</p>	<p>a. Before making the call to your web service to save your data, a check should be made to see if you are online or offline. If you are online submit the data as you normally would. If you are offline you will want to save it in the localStorage. Below is an example function to save your form data to localStorage. Keep in mind that localStorage properties and values must be strings. The following function is found in the scripts directory of the Lab Folder in the file localStorage.js.</p> <pre> /* Saves the formData to localStorage. * @param formData JSON Object of form properties and values */ function saveToLocalStorage(formData) { var customers = {}; // If we have any customers already in localStorage // we need to get those. Since they are stored as // strings we can use IE8's built in JSON methods // to parse it back into a JSON object for easy manipulation. if (localStorage.customers) { customers = JSON.parse(localStorage.customers); } // Use the Customer Id as the property name. customers[formData.CustomerID] = formData; // Because localStorage can only store strings, // we can use IE8's built in JSON methods to convert // our JSON object into a string. localStorage.customers = JSON.stringify(customers); } </pre> <p>Note: The sales staff now has a way to store data entered in web applications while offline but are unable to submit it to the server. You will implement IE8's Connectivity events to automatically submit the data in the client side DOM Store to the server upon reconnecting to an Internet connection. This will allow the sales staff to</p>

Tasks	Detailed Steps
	<p><i>perform their work at any location without regard for the presence of an Internet connection.</i></p>
<p>4. Saving localStorage to External Database</p>	<p>a. To save your localStorage data use the same process as you would if you were submitting form data. Below is an example implementation of taking the localStorage and submitting it to your web services to save to your external database. The following function is found in the scripts directory of the Lab Folder in the file localStorage.js.</p> <pre data-bbox="505 436 1398 1709"> /* Save the customers that are currently * in the localStorage to the database. */ function saveLocalStorage() { if (!localStorage.customers) return false; var customersObj = JSON.parse(localStorage.customers); if (isEmpty(customersObj)) return false; \$("#saveLocalStorage").hide(); // Send our customers to our web service as an array. var customers = new Array(); for (var customer in customersObj) { customers.push(customersObj[customer]); } // \$.ajax is a jQuery method \$.ajax({ type: "POST", url: "NewCustomer.aspx/SaveCustomers", // Using the built in IE8 JSON methods to // convert our object to a string data: "{customers:" + JSON.stringify(customers) + "}", contentType: "application/json; charset=utf-8", dataType: "json", success: function(msg) { var msg = "customers"; if (customers.length == 1) { msg = "customer"; } \$("#flash").text("Successfully saved " + msg); localStorage.clear(); updateList(); }, error: function(request, status, error) { \$("#flash").text("Error (" + status + "): " + error); } }); } </pre> <p>b. Now that everything has been put in place it is time to test our form. If NewCustomer.aspx is not already open in IE8, reopen it. Click the “File” drop down menu and select “Work Offline.” You will notice that below the form the status changes from “online” to “offline.”</p> <p>c. Now enter some text in the Customer Id field (This field is limited to five characters long). The Customer Id is the only required field so we do not need to</p>

Tasks	Detailed Steps
	<p>both with filling them out for this lab. Click the “Save” button and the Customer Id you entered will display below the save button with an “DELETE” next to it. The new customer is now saved in localStorage. To remove the customer from localStorage simply click the “DELETE” and in the alert box click “Ok.” (see Figure)</p>  <p>The screenshot shows a light beige background. At the top left is a blue button with the text 'Save'. Below it is a bulleted list of three customer IDs: 'ASDF (DELETE)', 'SDFG (DELETE)', and 'DFGH (DELETE)'. At the bottom of the screenshot, the text 'You are currently offline' is displayed.</p> <p>d. Go ahead and add another customer or two to build up a list of three or 4 new customers. Once that is completed switch the browser back to “online” mode by clicking the “Tools” drop down menu and deactivate “Work Offline.” You will notice that below the form the status changes from “offline” to “online” and a button appears below the customers saved in localStorage which says “Save Local Storage.” Click the “Save Local Storage” button. At the top of the form you are shown that the customers were successfully saved to the database.</p> <p>e. The process for saving customers when online works in the exact same way but saves the customers directly to the database. This concludes the AJAX and Enhanced Layout Standards Lab.</p>

Related Resources:

[Internet Explorer 9 Beta Download](#)