

Microsoft® Virtual Labs

**Internet Explorer 8 –
Improved Programmability**

Microsoft®

Table of Contents

Internet Explorer 8- Improved Programmability	1
Exercise 1 Implement Native JSON Support and the Improved getElementById Function	2
Exercise 2 Optimize JavaScript using IE8 Developer Tools	4

Internet Explorer 8- Improved Programmability

Objectives

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

- Use IE8's Developer Tools to optimize your JavaScript
- Use Native JSON support and the improved getElementById function

Scenario

As a developer for Northwind Traders, you've been tasked with making updates to the product section of the company's website. Currently, the product browser uses AJAX calls to return data from the server. Data is formatted with JSON using an external library. Your task is to make use of IE8's native support JSON and JS profiler to streamline the product browser and make it more responsive for customers. In addition to using IE8's JSON and JS profiler, you will also use the improved "getElementById" function.

To save time, certain components have been pre-installed and pre-configured. These components include Internet Information Services 7 (IIS7) and Internet Explorer 8. You will be using Notepad for editing HTML and JavaScript.

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

30 Minutes

Computers used in this Lab




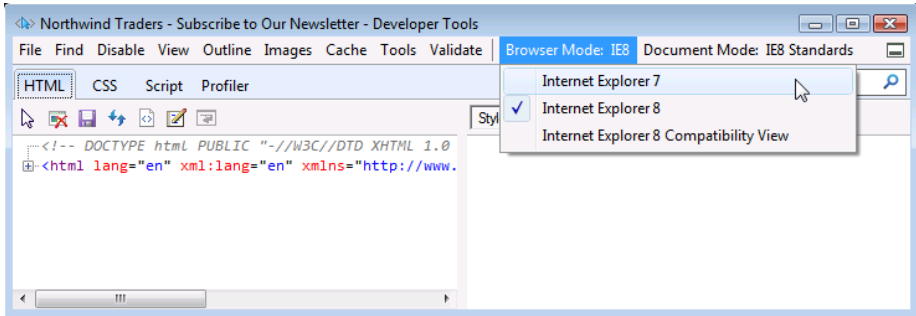

The password for the Administrator account on all computers in this lab is: P@ssword.


Exercise 1

Implement Native JSON Support and the Improved getElementByld Function

Scenario

In this exercise you will see an example of an email newsletter signup form for Northwind Traders. The user must enter their email address and click “Sign up” to join the list. After submitting their address an alert box notifies them of their submission. This alert box uses the improved getElementById function and you will see how it differs from its implementation in previous versions of Internet Explorer.

Tasks	Detailed Steps
<p>Complete the following tasks on:</p>  VistaIE8DT	<p>a. Open Newsletter.html in IE8. Open IE8 Developer Tools (Tools > Developer Tools or press F12). Click Browser Mode and select Internet Explorer 7. (see Figure below) Changing the Browser Mode to IE7 will demonstrate how getElementById functioned prior to IE8.</p> 
<p>1. Changing the Browser Mode</p> <p>2. Testing the Form</p>	<p>a. Once the Browser Mode has been changed enter an email address into the form. You may use any address that you wish. In this example test@test.com was used. Click “Sign up” and take note of what the alert box says. (see Figure below)</p>  <p>b. The alert box is giving a value of undefined due to a conflicting name and id which have the same value.</p>




Tasks	Detailed Steps
<p>3. Viewing the Code</p>	<p>a. As you can see in the code below there are two anchor tags which are causing this issue. The first anchor tag has a name attribute set and the second anchor tag has an id attribute set.</p> <pre data-bbox="506 300 1398 520"> <form> <p>Enter your e-mail ...</p> <label for="email_subscribe">E-mail: </label> <input type="text" id="email_subscribe" /> Sign up </form> </pre>
<p>4. Resetting the Browser Mode and Testing</p>	<p>a. Reopen the Developer Tools window if it is not already open. Change the Browser Mode back to IE8. Enter an email address (any address will do), and click “Sign up.” This time the email address you entered will show up in the alert box as IE8 has skipped over the name attribute and responded with the value for anchor tag with the id attribute. (see Figure below)</p> 

Exercise 2

Optimize JavaScript using IE8 Developer Tools

Scenario

In this exercise, you will use IE8’s developer tools to optimize existing JavaScript.

Tasks	Detailed Steps
<p>Complete the following tasks on:</p>  VistaIE8DT	<p>a. Open ProductBrowser.aspx by starting Internet Explorer 8 and typing “http://localhost/ProductBrowser.aspx” in the address bar. You will see two different buttons. Each will pull data from an external database and display the results; however, they are each coded slightly different. (see Figure below)</p> 
<p>2. Opening the JavaScript Profiler</p>	<p>a. Before clicking the buttons we need to start the JavaScript Profiler within the IE8 Developer Tools. Open Developer Tools (Tools > Developer Tools or press F12).</p> <p>b. With the Developer Tools window open click the Profiler tab which is located on the blue bar of the Developer Tools. (see Figure below)</p> 

Tasks	Detailed Steps																																																
<p>3. Using the JavaScript Profiler</p>	<p>a. Click the “Start Profiling” button. Note that the button text changes to “Stop Profiling.” Now we can run our scripts and test how long our scripts are taking to complete.</p> <p>b. Click the “Get Products (v1)” button five times pausing briefly between clicks. You will see the product list display 77 items. Now click “Get Products (v2)” button five times pausing briefly between clicks.</p> <p>c. Click the “Stop Profiling” button in the Developer Tools to stop the profiler. The results of the profiler will now be displayed. Sort the results by alphabetical order by clicking on the heading labeled “Function” twice (the first column of the table). Scroll to near the top of the list and look for the functions “buildProductsTable1” and “buildProductsTable2.” (see Figure below)</p> <table border="1" data-bbox="506 569 1360 940"> <thead> <tr> <th>Function</th> <th>Count</th> <th>Inclusive Time (ms)</th> <th>Exclusive Time (ms)</th> </tr> </thead> <tbody> <tr> <td>Array.concat</td> <td>434</td> <td>4.88</td> <td>4.88</td> </tr> <tr> <td>Array.push</td> <td>17,486</td> <td>23.44</td> <td>23.44</td> </tr> <tr> <td>Array.shift</td> <td>11,252</td> <td>8.79</td> <td>8.79</td> </tr> <tr> <td>Array.unshift</td> <td>11,252</td> <td>14.65</td> <td>14.65</td> </tr> <tr> <td>buildProductsTable1</td> <td>5</td> <td>454.12</td> <td>20.51</td> </tr> <tr> <td>buildProductsTable2</td> <td>5</td> <td>216.81</td> <td>9.77</td> </tr> <tr> <td>clean</td> <td>395</td> <td>151.37</td> <td>3.91</td> </tr> <tr> <td>complete</td> <td>10</td> <td>523.46</td> <td>0.00</td> </tr> <tr> <td>css</td> <td>41</td> <td>1.95</td> <td>0.98</td> </tr> <tr> <td>curCSS</td> <td>41</td> <td>0.98</td> <td>0.98</td> </tr> <tr> <td>data</td> <td>32,458</td> <td>408.22</td> <td>408.22</td> </tr> </tbody> </table>	Function	Count	Inclusive Time (ms)	Exclusive Time (ms)	Array.concat	434	4.88	4.88	Array.push	17,486	23.44	23.44	Array.shift	11,252	8.79	8.79	Array.unshift	11,252	14.65	14.65	buildProductsTable1	5	454.12	20.51	buildProductsTable2	5	216.81	9.77	clean	395	151.37	3.91	complete	10	523.46	0.00	css	41	1.95	0.98	curCSS	41	0.98	0.98	data	32,458	408.22	408.22
Function	Count	Inclusive Time (ms)	Exclusive Time (ms)																																														
Array.concat	434	4.88	4.88																																														
Array.push	17,486	23.44	23.44																																														
Array.shift	11,252	8.79	8.79																																														
Array.unshift	11,252	14.65	14.65																																														
buildProductsTable1	5	454.12	20.51																																														
buildProductsTable2	5	216.81	9.77																																														
clean	395	151.37	3.91																																														
complete	10	523.46	0.00																																														
css	41	1.95	0.98																																														
curCSS	41	0.98	0.98																																														
data	32,458	408.22	408.22																																														
<p>4. Interpreting the Results</p>	<p>a. The “Count” column, in this example, is how many times the function was called. Since you clicked each button five times, “5” should be displayed in both buildProductTable rows. The Inclusive Time (ms) is what we are most concerned with. As you can see the Inclusive Time for buildProductTable1 is more than double the amount of time than buildProductsTable2. You can see where if you had a much larger script this might play a huge role in how you can shorten the time it takes for your scripts to run. If you had a script that took longer than this example you could be waiting for several seconds. Using the Profiler can help you see where loading time is being spent so you can optimize your scripts.</p> <p>b. This concludes the IE8 Improved Programmability Lab.</p>																																																