Concordia University
School of Extended Learning
Centre for Continuing Education

CERTIFICATE IN WORDPRESS AND JAVASCRIPT

Course Title:	JavaScript and Ajax				
Course Number:	CEWP 339				
Course hours:	40	Course Weighting:	2-3-3	Number of Credits (Units):	N/C
Pre-requisite:	CEWP 329 – HTML5 and Cascading Style Sheets				

Khattar Daou, M.S., Ph.D. Technical Sciences
Microsoft Certified Trainer (MCT), Microsoft Office Specialist (MOS)

Enterprise Strategy Consultant
Datsco – Software Development, Training, & Consulting
6687 Hamilton Street
Montreal, Quebec
H4E 3C6
KDaou@DatscoTraining.com

Assignment 02

Deadline for submission: Class 10 (as per schedule)
Late submissions will not be accepted.

Student Name	Student ID

Case Problem 1 – to create a Travel expense report

DeLong Enterprises
Kay Ramirez is the payroll manager for DeLong Enterprises, a manufacturer of computer components. The company has been busy putting corporate information up on the company's Internet. Kay is heading a project to put all off the payroll-related forms and reports online. She has come to you for help in writing a program for the online travel expense form. The travel expense report form requires employees to itemize their various travel expenses for corporate trips. Kay would like scripts added to the form to ensure that all of the required data is entered in the correct format. If a required data field is left blank or if data is entered in an improper format, Kay would like the program to highlight the field in yellow and refuse submission of the form to the corporate Web server. Kay wants the form to support the following features:

- An employee must enter his or her last name, first name, social security number, an address for the reimbursement check, and a summary of the trip.

- The employee must enter the account ID number in the format **ACTdddddd**, the department ID number in the format **DEPTdddddd**, and the project ID number in the format **PROJdddddd**.
- For each day in which the employee has recorded an expense, the travel date must be entered.
- When the employee enters a travel, lodging, or meal expense, the subtotal of expenses for that day and the total cost of the trip should be automatically updated.
- Travel expenses must be entered as digits (either with or without the two-digit decimal place) and displayed to two digits.

A preview of the completed form (highlighting a few enormous entries) is attached as an image named DeLong Enterprises.

Data files needed for this assignment

Back.jpg | done.html | expense.css | exptxt.html | travelExp.js | links.jpg | logo.jpg

To complete the following steps:

Create a folder with yourName_courseNumber. Add a subfolder named Assignment_02. Organize your assignment to include folders for images, styles, and scripts

1. Use your text editor to open the exptxt.html file. Enter **your name** and the **date** in the comment section and save the file as **expense**.html Take some time to examine the contents of the file. Pay close attention to the field names and the organization of the form.
2. In the head section of the file, create a script element to link to an external JavaScript file named **travelExp.js**. Insert a function named **testLength**(). The purpose of the testLength() function is to test whether the user has entered any text in a required field. If no text has been entered, the function should highlight the field and return the value false. If any text has been entered, the function should remove any highlighting and return the value true. The function has a single parameter named "field" that represents the field object to be tested. To complete the function:
    a. Insert a conditional expression that tests whether the length of the field value is equal to zero.
    b. If the length is equal to 0, then:
        i. Change the background color of the field object to yellow and
        ii. Return the Boolean value false
    c. Otherwise:
        i. Change the background color of the field object to white and
        ii. Return the Boolean value true
3. Create a function named **testPattern**(). The purpose of this function is to compare the value of a field against a regular expression pattern. If the field's value does not match the regular expression, the function should highlight the field on the form and return the Boolean value false. If the field's value does match the regular expression, the function should remove any highlighting and return the Boolean value true. The function has two parameters: the **field** parameter representing the field object to be tested and **reg**, a regular expression literal containing the pattern used for testing. To complete the testPattern() function:

   a. Insert a conditional expression that employs the **test**() method to test whether the value of the field object matches the regular expression contained in the **reg** parameter.
   b. If the test() method returns the value false, then:
      i. Change the background color of the field object to yellow
      ii. Change the color of the field object to red, and
      iii. Return the Boolean value false
   c. Otherwise:
      i. Change the background color of the field object to white
      ii. Change the color of the field object to black, and
      iii. Return the Boolean value true
4. Create a function named **testDates**(). The purpose of this function is to check whether a travel date has been entered far a day in which the employee has recorded an expense. The function has no parameters. Add the following commands to the function:
   a. Create a variable named dateExists. The purpose of this varable is to record whether the employee recorded a traveling expense without entering the date. Set the initial value of this variable to true.
   b. Create a For loop that loops through integer values from 1 to 4. The purpose of this loop is to examine each of the four rows in the travel expense table. If the subtotal value in the row is greater than 0, there should be a date entered in the corresponding date field. The subtotal fields in the table are named **sub1** through **sub4**. The date fields in the table are named **date1** through **date4**.
   c. Within the For loop, insert a conditional statement that tests whether the value of $\text{sub}_{\textbf{count}}$ is not equal to “0.00”, where ***count*** is the value of the counter in the for loop. (Hint: Use the object reference document.expform.elements[“sub” + count].value, where count is the counter in the For loop.)
   d. If $\text{sub}_{\textbf{count}}$ is not equal to “0.00” (indicating that an expense has been entered in that row of the travel expense table), do the following:
      i. Call the **testLength**() function using the $\text{date}_{\textbf{count}}$ field as the parameter;
      ii. Store the value returned by the testLength() function in a variable named **rowDateExists**; and
      iii. If the value of the rowDateExists variable is false, set the value of the **dateExists** variable to false
   e. After the For loop, return the value of the dateExists variable.
5. Create a function named **validateForm**(). The purpose of this function is to validate the form before it can be submitted to the server by calling the **testPattern**(), **testLength**(), and **testDates**() functions you just created. The function has no parameters. Add the following commands to the function:
   a. Create a variable named **valid**. The purpose of this variable is to record whether the form is valid or not. Set the initial value of the valid variable to true.
   b. Call the testLength() function with the lname field object as the parameter. (Hint: Use the object reference document.expform.lname). If the value returned by the testLEngth() function is false, set the value of the valid variable to false. Repeat this step for the fname, address, and summary fields.
   c. Call the testPattern() function with the account field object for the field parameter. For the reg parameter insert a regular expression literal that matches a

text string containing only the text “ACT” followed by six digits. If the value returned by the testPattern() function is false, set the value of the valid variable to false.

   d. Call the testPattern() function with the department field for the field parameter. The reg parameter should contain a regular expression literal for a text string containing only the characters “DEPT” followed by six digits. If the value returned by the testPattern() function is false, set the value of the valid variable to false.
   e. Repeat the previous step for the project field, using a rgular expression that matches a text string containing only the characters “PROJ” followed by six digits.
   f. Call the testPattern() function for the ssn field (containing the social security number of the employee). The reg parameter should match either a nine-digit number or a text string in the form ddd-dd-dddd. If the value returned by the testPattern() function is false, set the value of the valid variable to false.
   g. Call the testDates() function. If the function returns the value false, set the value of the valid variable to false.
   h. Insert a conditional statement that tests whether the value of the valid variable is false. If it is false then:
      i. Display the following alert message: “Please fill out all required fields in the proper format.”
   i. Return the value of the valid variable.
6. Locate the form element and add an event handler to run and return the value of the **validateForm**() function whether the form is submitted.
7. Go back to the embedded script element and add a function named **calcRow**(). The purpose of this function is to return the subtotal of the expenses within a single row in the travel expense table. The function has a single parameter named **row**, which represents the number of the table row (from 1 to 4) to calculate from. Add the following commands to the function:
   a. Create a variable named **travel** equal to the numeric value of the travel$_{row}$ field, where *row* is the value of the row parameter. (Hint: Use the object reference document.expform.elements[“travel” + row].value.) Be sure to use the parseFloat() function to convert the field value to a number. In the same fashion, create a variable named **lodge** equal to the numeric value of the lodge$_{row}$ field and a variable named **meal** equal to the numeric value of the meal$_{row}$ field.
   b. Return the sum of the travel. Lodge and row variables.
8. Create a function named **calcTotal**(). The purpose of this function is to return the total of all expenses in the travel expense table by calling the **calcRow**() function for each row in the table. The function has no parameters. Add the following commands to the function:
   a. Create a variable named **totalExp** and set its initial value to 0.
   b. Insert a For loop with a counter that runs from 1 to 4.
   c. Within the For loop, increase the value of the totalExp variable by the value returned by the **calcRow**() function using the value of the counter as the value of the row parameter.
   d. Return the value of the **totalExp** varable.

9. Create a function named **update**(). The purpose of this function is to update the expense values displayed in the table and to verify that the employee has entered a valid expense amount. The function will be called whenever the employee exits from one of the 12 expense fields in the table. The function has a single parameter named **expense** that represents the field object that called the function. Add the following commands to the function:
    a. Create a variable named **numRegExp** that contains the regular expression literal /^\d*(\.\d{0,2})?$/. This pattern matches any text string that only contains a number with or without two decimal place accuracy.
    b. Insert a conditional statement that tests whether the value property of the expense parameter matches the **numRegExp** pattern.
    c. If the condition is met (meaning that the employee has entered a valid expense amount), then run the following commands:
        i. Display the value property of the expense parameter to two decimal places (Hint: Use the **toFoxed**() method; if your browser does not support this method, use the toFixed2() function from the travelExp.js file);
        ii. Insert a For loop with a counter that run s from 1 to 4. Within the For loop, set the value of the sub$_{\textbf{count}}$ field to the value returned by the calcRow() function, where count is the value of the For loop counter. Format the value to appear to two decimal places; and
        iii. Set the value of the total field equal to the value returned by the calcTotal() function. Display the total field value to two decimal places.
    d. If the condition is not met (meaning that the user has entered an invalid number), then:
        i. Display the alert message “Invalid currency value”;
        ii. Change the value property of the expense parameter to “0.00”; and
        iii. Return the focus to the expense object.
10. Locate the html tag for the **travel1** input field. Add an event handler that runs the **update**() function in response to the blur event. Use the “this” keyword for the expense object parameter. Add the same event handler to the **travel2** through **travel4** fields, the **lodge1** through **lodge4** fields, and the **meal1** through **meal4** fields.
11. Save your changes and close the file. Open the **expense**.html in your browser. Test the operation of the travel expense table, verifying that it automatically updates the travel expenses as you add new values to the table.
12. Test the form validation commands by attempting to submit the form under the following conditions:
    a. Without all of the required fields filled out;
    b. With invalid entries for the account, department, project, and social security number fields; and
    c. With travel expenses entered without corresponding dates in the travel expense table. The form should highlight the errors and alert you of the mistake.
13. After successful CSS, HTML, and script validations, include all files and folder related to the Assignment2 in yourName_courseNumber folder, and then compress the folder
14. To compress a folder, right-click the folder, point to Sent To, and then select Compressed (Zipped) Folder.

15. Submit the zipped folder to your instructor with using the following Email address: KDaou@DatscoTraining.com

References:

1. **Beginning JavaScript with DOM Scripting and Ajax 2nd Edition** | Russ Ferguson , Christian Heilmann | ISBN: 978-1430250920 © 2013 | Published by APress
2. **New Perspectives on JavaScript and AJAX, Comprehensive, 2nd Edition** | Patrick Carey, Frank Canovatchel | ISBN: 9781439044032 © 2010 | Published by Course Technology, Cengage Learning
3. **Professional JavaScript for Web Developers, 3rd Edition** | Nicholas C. Zakas | ISBN: 978-1-1180-2669-4 © 2012 | Published by John Wiley & Sons
4. **Beginning JavaScript, 3rd Edition** | Paul Wilton, Jeremy McPeak | ISBN: 978-0-470-05151-1 ©2007 | Published by John Wiley & Sons