Lakeside View Global Vision
CEGEP JOHN ABBOTT COLLEGE

CONTINUING EDUCATION
FORMATION CONTINUE

Certificate Program in Internet Programming & Development (A.E.C. LEA.BN)

Course Outline

Program Name:	Internet Programming & Development	
Course Title:	JavaScript	
Course Number/Section:	420-PX4-AB	
Start Date:	December 8, 2016	End date: December 23, 2016
Schedule:	8:30 am to 2:30 pm	
Classroom:	BH-213	
Ponderation:	2 hours lecture + 2 hours laboratory + 3 hours homework	
Hours:	60 hours	
Credits:	2.33	
Competencies:	DC54 – Apply JavaScript in a web browser	
Pre-requisite(s):	(420-PC3-AB) Hyper-Text Markup Language (HTML & XML), (420-PD3-AB) Cascading Style Sheets (CSS)	
Semester:	Fall 2016	
Teacher:	Khattar Daou, M.S., Ph.D. Technical Sciences Microsoft Certified Trainer (MCT), Microsoft Office Specialist (MOS)	
Telephone:	N/A	
Email:	The instructor can be reached from the email tool within JAC Portal: Khattar.Daou@JohnAbbott.qc.ca	

Assignment 01

Deadline for submission: December 15th
Late submissions will not be accepted.

Student Name	Student ID

Case Problem 1 – to create a World clock

Jackson Electronics is a worldwide company that manufactures and sells quality equipment and components. The company has six corporate offices at different locations on the globe and employees of the company have to keep in constant communications with the different offices. David Lin maintains the corporate Web site and has come to you for help with a problem. He would like to augment the Web page that displays the location of the corporate offices to display the local time at each location. This will give employees information when they want to call or fax data from one office to another. To create this world clock, David needs to know how JavaScript's date object works with different time zones.

The Earth is divided into 24 time zones. Each time zone is referenced in comparison to the time kept in Greenwich, England, which is known as **standard time** or **Greenwich Mean Time** (**GMT**).
Coordinated Universal Time (UTC) is the basis for civil time in many places worldwide. Many devices for measuring and showing time use this 24-hour time scale, which is determined

using highly precise atomic clocks. Time zones around the world are expressed as positive or negative offsets from UTC. The hours, minutes, and seconds that UTC expresses is kept close to the mean solar time at the Earth's prime meridian (zero degrees longitude) located near Greenwich, England

UTC is commonly referred to as Greenwich Mean Time (GMT) when not counting the precise accuracy regarding fractions of a second. GMT is a historical term that has been widely used in many ways. GMT was first adopted as the world's time standard (http://www.timeanddate.com/time/gmt-utc-time.html) at the Washington Meridian Conference in 1884. GMT is no longer the basis for civil time but is now loosely interchanged with UTC to refer to time kept on the Greenwich meridian (longitude zero) (http://www.timeanddate.com/library/abbreviations/timezones/eu/gmt.html). Places such as the United Kingdom observe GMT during the non-daylight saving period.

You can determine how many minutes Greenwich time differs from your local time using the **getTimezoneOffset**() method. For example, if the **today** variable contains a date object and is run on a computer in Montreal, the expression

today. **getTimezoneOffset()**

returns the value 300 because Greenwich Mean Time is 300 minutes or five hours ahead of Montreal. For example, when it is noon (12:00 P.M.) in UTC/GMT, it will be 8:00 A.M. same day in Montreal. With this information you can determine the time in Greenwich by adding the offset value to your computer's local time. Since JavaScript measures time in milliseconds, you have to multiply the offset by the number of milliseconds in one minute.

You can determine the time anywhere in the world if you know Greenwich's time and the other location's offset from UTC/GMT. David has compiled a list of the six corporate offices and the time difference in minutes between each of those cities and UTC/GMT. The offices are:

- Office 1: Houston (-360)
- Office 2: London
- Office 3: New York
- Office 4: Seattle
- Office 5: Sydney (660)
- Office 6: Tokyo

The number in parentheses indicates the number of minutes the city is offset from UTC/GMT. A negative value indicates that the city is behind UTC/GMT time, while a positive value indicates that it is ahead of Greenwich. Tokyo, for example, is 540 minutes or 9 hours ahead of Greenwich.

David has already designed the contents of the world map Web page, but he needs your help in programming the time for the six offices (figure included in the data files).

Data files needed for this assignment

je.css | je_logo.jpg | map.jpg | worldTimetxt.html | timeZonestxt.js

To complete the following steps:

Create a folder with yourName_JS_A1. Organize your assignment to include folders for images, styles, and scripts

1. Open a text editor, open worldTimetxt.html document. Enter your **name** and the **date** in the head section as a <meta> tag and save the file as **worldTime.html**. Add the Document type declaration for HTML5 and the related tags in the <head> element.

2. Open timeZonextxt.js, enter your **name** and the **date** in the comment section, and save the file as **timeZones.js.**

3. Within the timeZones.js file, create a function named **addTime**(). The purpose of the addTime() function is to create a new date object by adding a specified number of milliseconds to an initial time value. The function has two parameters, named **oldTime** and **milliseconds**. The oldTime parameter stores a data object representing an initial time value. The milliseconds parameter stores the amount of time, in milliseconds, that should be added to the oldTime parameter. Add the following commands to the function:

    a. Create a date object named **newTime**, but do not specify a value for its date or time.

    b. Using the **getTime**() method, extract the number of milliseconds contained in the **oldTime** parameter and add this to the **milliSeconds** parameter. Store the sum in a variable named **newValue.**

    c. Using the **setTime**() method, set the time value of the **newTime** date object to the value of the **newValue** variable.

    d. Return the newTime date object from the function.

4. Below the addTime() function, create a function named **showTime**(). The purpose of this function is to return a text string showing the time in a 12-hour format. The function has a simple parameter named **time**, which contains the date and time that you want displayed. The showTime() function should return the following text string:

    hour: minute AM/PM

    where hour is the hour value in the 12-hour format, minute is the minute value, and AM/PM is either “AM” or “PM” depending on the time of day.

5. Save and validate the timeZones.js document.

6. Go to the worldTime.html file in your text editor. Above the closing </head> tag, insert an external script element to access the functions you created in the timeZones.js file and then insert an embedded script element.

7. Within the embedded script element, create a function named **worldClock**(). The purpose of this function is to calculate the time in different time zones. Within this function, do the following:
   a. Create a date object variable named **today** equal to the current data and time.

   b. Apply the **getTimezoneOffset**() method to the today variable to calculate the offset of your computer's clock from UTC/GMT in minutes. Change this value to milliseconds. Store the result in a variable named **offset**.

   c. Call the **addTime**() function using **today** as the first parameter value and **offset** as the second. Store the value returned by this function in a variable named u**niversalTime**. The universalTime variable represents the current data and time in Greenwich.

   d. Calculate the current data and time at Jackson Electronics' first office (Houston). To calculate this value, call the addTime() function with universalTime at the first parameter and the second parameter equal to the number of milliseconds that Houston is offset from universalTime. (Hint: Since Houston is 360 minutes behind Greenwich, the offset from universalTime is equal to (-360) * 60 * 1000). Store the date object returned by the addTime() function in a variable named **time1**or **houstonTime**. Repeat this step to create variables named time2 through time6 for the other five office locations using the offset values that David has indicated in the list above.

   e. The current times for the six office locations are to be displayed in input fields named **place1** through **place6** in the zones Web form. To display the value of the place1 field, call the **showTime**() function using the time1 variable as the parameter value. Repeat this step for the five remaining input fields.

8. Add an event handler attribute to the <body> tag to run the worldClock() function when the page is loaded by the browser and every second thereafter.

9. Save your changes and close the file.

10. Open worldTime.html in your browser. Verify that it shows the current time for the six office locations and that these times are correctly offset from Greenwich.

11. After successful CSS, HTML, and script validations, include all files and folder related to the Assignment1 in yourName_JS_A1 folder, and then compress the folder.

12. To compress a folder, right-click the folder, point to **Sent To**, and then select Compressed (Zipped) Folder.

13. Submit the zipped folder to your instructor by email at KDaou@DatscoTraining.com.

***Note**: This is a simplified example of what is a very complicated problem. Different countries apply time zones in different ways. For example, China spans several time zones but applies a uniform time throughout the country. Some countries also shift their time(s) twice a year during daylight saving time (otherwise known as summer time) while others do not apply daylight savings time at all. For example, the reported times in the case problem will be off by 1 hour during daylight saving time for the Seattle, Houston, and New York clocks. To create a truly accurate world clock, you would have to take into account all the various idiosyncrasies of global timekeeping.*

References:

1. **New Perspectives on JavaScript and AJAX, Comprehensive 2nd Edition** | Patrick Carey, Frank Canovatchel | ISBN: 978- 1439044032 © 2009 | Published by Course Technology
2. **Introduction to JavaScript Programming with XML and PHP** | Elizabeth Drake | ISBN: 9780133068306 © 2014 | Addison-Wesley Pearson Education
3. **Professional JavaScript for Web Developers, 3rd Edition** | Nicholas C. Zakas | ISBN: 978-1-1180-2669-4 © 2012 | Published by John Wiley & Sons, Inc.