

**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](mailto:KDaou@DatscoTraining.com)

## Assignment 01

**Deadline for submission: Class 5 (as per schedule)**

Late submissions will not be accepted.

Student Name	Student ID
<input type="text"/>	<input type="text"/>

### 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)**. 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 New York, the expression



today. **getTimezoneOffset()**

returns the value 300 because Greenwich Mean Time is 300 minutes or five hours ahead of New York. 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. Thus, the code to calculate the number of hours using this function would be

today. **getTimezoneOffset() \* 60 \* 1000**

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

- Office 1: Houston (-360)
- Office 2: London (0)
- Office 3: New York (-300)
- Office 4: Seattle (-480)
- Office 5: Sydney (660)
- Office 6: Tokyo (540)

The number in parentheses indicates the number of minutes the city is offset from GMT. A negative value indicates that the city is behind Greenwich Mean 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 | logo.jpg | map.jpg | worldtxt.html | zonetxt.js

#### *To complete the following steps:*

Create a folder with yourName\_JS\_AJAX\_Assign\_01. Organize your assignment to include folders for images, styles, and scripts

1. Open a text editor, open worldtxt.html document. Enter your **name** and the **date** in the head section as a <Meta> tag and save the file as **world.html**. Add the Document type declaration for XHTML 1.1 and the related tags in the <head> element.
2. Open zonetxt.js, enter your **name** and the **date** in the comment section, and save the file as **zones.js**.
3. Within the zones.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 zones.js document.
6. Go to the world.html file in your text editor. Above the closing </head> tag, insert an external script element to access the functions you created in the zones.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 date and time.
  - b. Apply the getTimezoneOffset() method to the today variable to calculate the offset of your computer’s clock from GMT in minutes. Change this value to milliseconds by multiplying the value by 60 and then by 1000. 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 GMT. The GMT variable represents the current date and time in Greenwich.
  - d. Calculate the current date and time at Jackson Electronics’ first office (Houston). To calculate this value, call the addTime() function with GMT at the first parameter and the second parameter equal to the number of milliseconds that Houston is offset from GMT. (Hint: Since Houston is 360 minutes behind



Greenwich, the offset from GMT is equal to  $(-360) * 60 * 1000$ ). Store the date object returned by the `addTime()` function in a variable named **time1**. 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 `world.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, XHTML, and script validations, include all files and folder related to the Assignment1 in yourName\_JS\_AJAX\_Assign\_01 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 using the following Email address:  
[KDaou@DatscoTraining.com](mailto: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. **Beginning JavaScript with DOM Scripting and Ajax 2<sup>nd</sup> Edition** | Russ Ferguson , Christian Heilmann | ISBN: 978-1430250920 © 2013 | Published by APress
2. **New Perspectives on JavaScript, Comprehensive** | Patrick Carey, Frank Canovatchel | ISBN: 978-0-619-26797-1 © 2006 | Published by Course Technology
3. **Beginning JavaScript, 3rd Edition** | Paul Wilton, Jeremy McPeak | ISBN: 978-0-470-05151-1 ©2007 | Published by John Wiley & Sons

