JavaScript Functions Assignment

Objectives

- Implement JavaScript functions with parameters and return values.
- Write JavaScript functions to solve real-world problems.

Instructions:

1. In your Sandbox, create a new JavaScript Console program called "JavaScriptFunctionsAssignment"
2. Edit the file based on the instructions below.
3. To submit the program, create a Share Link and email it to gorrm@kingphilip.org.

Part 1: Defining a Function

1. Create a function called **`part1()`** that prints the line "**`***Part 1 ***`**"
2. Call **`part1()`** from your **`start`** function. You may copy-and-paste your **`part1()`** function to use as a template for parts 2 through 4.
3. As you complete the rest of the assignment, add calls to each part's function in your **`start`** function.

Part 2: Parameters and Return Values

Windchill is defined by the weather authorities of the United States, Canada, and the United Kingdom by the following formula:

$$\boldsymbol{Wind\ Chill = 13.12 + 0.6215 \times T - 11.37 \times V^{0.16} + 0.3965 \times T \times V^{0.16}}$$

where $\boldsymbol{T}$ is the temperature in Celsius and $\boldsymbol{V}$ is the wind velocity in kilometers per hour.

1. Define a function in your program to calculate the wind chill for any temperature and velocity, like so:
   `windChill(temp, velocity);`
   Sample output from your function should resemble the following:
   `At -20C and 10 kph winds, it feels like -27.206987575396404 C`
2. In a second function entitled **`part1()`**, experiment with temperatures of $-20\ ℃$ and $-25\ ℃$ and integer wind velocities to find the wind chill closest to $-35\ ℃$. Show your calls to **`windChill()`** in your program, and report your results in a comment after your **`part3()`** function.

NOTE: For the roots, you may use the JavaScript Power function from the math toolbox, in the form **`Math.pow(base,exponent)`**. To find a square root, the exponent would be ½.

Part 3: Pythagorean Theorem

An algorithm to determine the length of the hypotenuse of a right triangle based on its legs is presented in *pseudocode* below, but it is not quite correct.

1. In a function entitled **`part3()`**, write a multi-line comment /* … */ describing any corrections you would make to the pseudocode algorithm.
2. Implement the algorithm as a JavaScript function (with documentation) that takes two numbers as arguments and returns the hypotenuse as a number.

```
COMMENT: Pythagorean example
pythagorean():
    side1 = 3
    side2 = 4
    long side = side1*side1 + side2*side1 ^ 0.5
    print (long side)
```

3. In your **part3()** function, use your **pythagorean()** function to determine the hypotenuse of a triangle with side lengths of 6 and 8.

Part 4: Extra Credit – Great Circle Distance

The Haversine formula is a method of estimating the distance between points on the earth's surface.

1. Read about the Haversine formula on the following web page: http://www.movable-type.co.uk/scripts/gis-faq-5.1.html
2. Write a JavaScript function to calculate great circle distances between two points on earth, expressed in latitude and longitude. The function should take four arguments, representing the latitude and longitude of two locations, and it should return the distance in kilometers. A template appears below.
3. Note that the angles in the formula[1] are expressed in radians, so you will need an auxiliary function to convert degrees to radians. The link above also describes a method for this conversion.
4. In a second function entitled **part4()**, use your function to calculate and print the great circle distances between the following cities. Note that the program should call the function for each *pair* of cities.
    - **Worcester, MA:** latitude 42.274470, longitude -71.804302 degrees[2]
    - **Beijing:** latitude 39.916345, longitude 116.397155 degrees
    - **Santiago de Chile:** latitude -33.469120, longitude -70.641997 degrees
5. Type the results of the three calculations in comment lines after your **part3()** function.
6. If you have covered Data Structures in the supplemental material you, may:
    a. Modify your function to accept two **lists** as arguments, where each list contains the latitude and longitude of a single city.
    b. Modify your function to contain two **objects** as arguments, where each object contains the latitude and longitude of a city, accessible by the words **latitude** and **longitude**.

Function Template:

```
/*
 * returns the distance in kilometers between
 * point A and point B given their latitude and longitude
 * arguments: latitude and longitude coordinates for two cities A and B
 *    latA, lonA, latB, lonB
 * returns: distance in kilometers between point A and point B
 */
function greatCircDist(latA, lonA, latB, lonB) {
     // Your code goes in here:
}
```

---

1 In fact, angles in most JavaScript functions (and functions in other programming languages) are expressed in radians.

2 By convention, latitudes in the northern hemisphere are positive and latitudes in the southern hemisphere are negative. Similarly, longitudes in the eastern hemisphere are positive and longitudes in the western hemisphere are negative.