W2D5 – Quiz 2 (20 points) – Time: 45mins

**INSTRUCTIONS**: *Please type all your answers directly, as text, into this document. You do not have to use a Code Editor, but you are free to, if you prefer. If you do use a Code Editor to type/test-run your code answers, then you can simply copy and paste the code for each question into the space provided in this document.*

**When finished, please make sure to submit this document, containing your answers, in either Microsoft Word format or Adobe PDF format, only. Submit by uploading your file into the Assignment named, W2D5 – Quiz2, on Sakai.**

**Note: All submissions must be made to Sakai within the time given. After the Assignment Quiz closes on Sakai, any late submissions can be emailed. However, all late submissions will score a maximum of 80% of the points, if it scored is up to that. Thanks.**

1. (2 points) What is Hoisting, in JavaScript? Give an example with code, to illustrate/support your answer.
   1. (0.5 point) Hoisting:

Hoisting is the moving of all the declarations in a scope block/function/global at the top of the scope before code execution.

* 1. (1.5 points) Code example:

***console***.log(*x*); // prints undefined   
var *x* = 5;  
***console***.log(*x*); // print 5

1. (4.5 points) What are the Scopes that code (variables and functions) can exist in, in JavaScript (version ES6 and above)? For each one, give an example with code, to illustrate/support your answer.

JavaScript has two scopes - global and local scope within its environment of execution.

const *module* = (function () {  
let *x* = 0;  
***console***.log(*x*);  
});

x is a local scope

module is in the global scope

1. (2 points) What is a Closure, in JavaScript? Give an example with code, to illustrate/support your answer.
   1. (0.5 point) Closure:

A closure is a when an inner function uses variables from its enclosing environment and keeps the state of those variables called free variable.

* 1. (1.5 points) Code example:

function outerFunction() {  
let *outerVariable* = 1;  
function InnerFunction() {  
 alert(*outerVariable*);  
 }  
  
 InnerFunction();  
}  
outerFunction();

1. (2 points) Assume you have been tasked to develop a web application that manages the Students Registration data for a school. Write JavaScript code below, for enabling creation of objects of type named, Student, using a Constructor Function. The specification for the application states that a student should have the following data fields (properties):

StudentId (e.g. 000-61-0001)

FirstName (e.g. Bob)

LastName (e.g. Jones)

Cgpa (e.g. 3.51)

"use strict";  
  
function Student(*studentId*, *firstName*, *lastName*, *cgpa*) {  
 this.studentId = *studentId*;  
 this.firstName = *firstName*;  
 this.lastName = *lastName*;  
 this.cgpa = *cgpa*;  
}

1. (3 points) Using the Construction Function you have coded in Question 4 above, now write JavaScript code to create an array of students, using the following data:

Student 1:

StudentId: 000-61-0001

FirstName: Anabelle

LastName: Smith

Cgpa: 3.05

Student 2:

StudentId: 000-61-0002

FirstName: Bob

LastName: Jones

Cgpa: 2.96

Next, write code to add a method named, toCSVString, to the student objects, such that it returns the String representation of a student object, in Comma Separated Value (CSV) format.

For example: *s1*.*toCSVString()* should return the String, “000-61-0001, Anabelle, Smith, 3.05” (Note: where s1 refers to a student object).

const *arrayStudents* = [];  
  
const *student1* = new Student("000-61-0001", "Anabelle", "Smith", 3.05);  
const *student2* = new Student("000-61-0002", "Bob", "Jones", 2.96);  
  
*arrayStudents*.push(*student1*);  
*arrayStudents*.push(*student2*);  
  
Student.prototype.toCSVString = function () {  
 return `${this.studentId},${this.firstName},${this.lastName},${this.cgpa}`;  
}

1. (4.5 points) By applying the Module Pattern, write JavaScript code for a module named, myCalculator. In your module, implement code for the following two methods:
   1. doAddition: this method takes as parameters, two numbers and it returns the sum of the numbers.
   2. doMultiplication: this method takes as parameters, two numbers and it returns the product of the numbers.

Write code to use your module to call/test your implementation of the two methods and simply print-out each result to the console.

const myCalculator = function(){  
  
const doAddition = function(num1, num2){  
let sum =num1+num2;  
return sum ;  
console.log(sum);  
};  
const doMultiplication = function(num1, num2){  
  
return num1\*num2;  
};  
  
return{  
doAddition: doAddition,  
doMultiplication: doMultiplication,  
  
}  
}();  
  
console.log(myCalculator.doAddition(2,3));  
console.log(myCalculator.doMultiplication(2,3));

1. (2 points) Consider the HTML markup for a Web Form shown in the *Figure 1,* below.



*Figure 1.*

Write-out the full destination URL (including the query string/parameters) generated by the browser, when the form is submitted to the web server.

http://localhost/reg/?stid=61001&fnames=John+Lewis

//-- The End --//