Advanced JavaScript Exam – CP1295

Noah Forward – 20175630

# Question 1

"use strict";

*// There are two basic objectives*

*// (1) Use regular expression to provide validation for Access Code*

*// (2) User range testing to provide validation for the two Keys*

**var** valid\_keys = false;

**var** valid\_access\_code = false;

$(document).ready(() **=>** {

    setStatus();

    $("#access\_code\_id").focus();

    $("#validate\_id").click(() **=>** {

        validateAll();

    });

});

**const** validateAll = () **=>** {

**var** access\_code\_input = $("#access\_code\_id").val();

    valid\_access\_code = testAccessCode(access\_code\_input);

    if (valid\_access\_code) {

**var** key\_1\_input = $("#key\_1\_id").val();

**var** key\_2\_input = $("#key\_2\_id").val();

        valid\_keys = testKeys(key\_1\_input, key\_2\_input);

    }

    setStatus();

}

**const** testAccessCode = (testCode) **=>** {

**var** testString = testCode.trim();

    if (testString == "")

        return false;

*//=(1)====================================================*

*// REGULAR EXPRESSION component*

*// Add in a regular expression to test for*

*// exactly three letters Upper and Lower Case are allowed*

*// Use the space provided below.*

*// Solution key used 4 lines of code*

*// -------------------------------------------------------*

**const** pattern = /^[A-Za-z]{3}$/;

    if (pattern.test(testString)) {

        return true;

    } else {

        return false;

    }

*//=(1)====================================================*

}

**const** testKeys = (key1, key2) **=>** {

    if (key1 == "" || key2 == "")

        return false;

    if (isNaN(key1) || isNaN(key2))

        return false;

**var** k1 = parseInt(key1);

**var** k2 = parseInt(key2);

*// =(2)====================================================*

*// RANGE TESTING component*

*// Data has been validated for blank and number testing*

*// and has been converted to itegers*

*// Your goal is to ensure that k1 and k1 are both in the*

*// range of 00 to 99 (inclusive)*

*// You are not to use Regular Expressions testing for this*

*// checkpoint(2) as it was it was already used for checkpont (1)*

*// Fill in the required code in the space provided.*

*// Solution key used 5 lines of code.*

*// --------------------------------------------------------*

    if ((k1 > 0 && k1 < 100) && (k2 > 0 && k2 < 100)) {

        if (k1 + k2 == 100) {

            return true;

        }

    }

*//=(2)========================================================*

}

*// NO MODIFICATONS below this point*

**const** setStatus = () **=>** {

    if (valid\_access\_code) {

        $("#key\_1\_id").removeClass("lock\_keys").prop("readonly", false);

        $("#key\_2\_id").removeClass("lock\_keys").prop("readonly", false);

        $("#access\_code\_id").next().text("Valid Access Code");

        if (valid\_keys) {

            $("#display\_id").removeClass("invalid\_access\_code");

            $("#display\_id").removeClass("invalid\_keys");

            $("#display\_id").addClass("valid\_keys");

            $("#key\_1\_id").next().text("KEYS PAIR is VALID !!!");

        }

        else {

            $("#display\_id").removeClass("invalid\_access\_code");

            $("#display\_id").removeClass("valid\_keys");

            $("#display\_id").addClass("invalid\_keys");

            $("#key\_1\_id").next().text("Invalid Key Pair");

        }

    }

    else {

        $("#display\_id").removeClass("invalid\_keys");

        $("#display\_id").removeClass("valid\_keys");

        $("#display\_id").addClass("invalid\_access\_code");

        $("#access\_code\_id").next().text("Invalid Access Code");

        $("#key\_1\_id").addClass("lock\_keys").prop("readonly", true);

        $("#key\_2\_id").addClass("lock\_keys").prop("readonly", true);

        $("#key\_1\_id").val("");

        $("#key\_2\_id").val("");

    }

}

# Question 2

"use strict";

*// Goal of the problem is to correct 2 problems in train.js*

*// (1) add in a setCookie call that will generate a location cookie*

*// (2) complete the setCookie Function*

*//*

$(document).ready(() **=>** {

    $("#train\_form").submit(event **=>** {

        event.preventDefault();

        try {

**var** selRadioLocation = $("[name=location]:radio:checked").val();

            setCookie("location", selRadioLocation, 1); *// 1 day persistent cookie*

*// =(1)===============================================================*

*// setCookie code is missing (a) Add in missing code where indicated*

**var** selRadioSpeed = $("[name=speed]:radio:checked").val();

*//(a)*

            setCookie("speed", selRadioSpeed, 1)

            $("#message").val("CK GEN");

        }

        catch (error) {

            event.preventDefault();

            $("#message").val("NO CK GEN");

        }

    });

});

**const** setCookie = (name, value, days) **=>** {

*// =(2) =================================================*

*// Complete the required cookie code to complete*

*// the creation of the cookie.*

*// factor in all of the supplied attributes name, value, days*

*// Solution Key used 5 lines of code*

*// -(2)--------------------------------------------------*

**let** new\_cookie = name + "=" + encodeURIComponent(value);

    if (days > 0) {

        new\_cookie += "; max\_age=" + days \* 24 \* 60 \* 60;

    }

    new\_cookie += "; path=/";

    document.cookie = new\_cookie

*// ========================================================*

}

# Question 3

"use strict";

*// Focus of this problem is to convert from an*

*// iterative array solution to an array map solution*

*// (1) Goal is to remove the existing Iterative Array Code*

*//     and replace it with a MAP REDUCE function*

**var** arrayA = [];

**var** arrayB = [];

$(document).ready(() **=>** {

    $("#process\_id").click(event **=>** {

        process\_d(arrayB);

    });

    loadData();

});

**const** loadData = () **=>** {

    arrayA = ["Bars", "Chips", "Cookies", "Batteries", "Detergent"];

    arrayB = [2.95, 3.95, 5.95, 9.95, 12.95];

**const** ul = document.createElement("ul");

    $("#data\_id").append(ul);

    for (**let** i = 0; i < arrayA.length; i++) {

**const** name = arrayA[i];

**const** price = arrayB[i];

**const** name\_price = String(`${name} ${price}`);

**const** li = document.createElement("li");

**const** text = document.createTextNode(name\_price);

        li.appendChild(text);

        ul.appendChild(li);

    }

}

**const** process\_d = (numbers) **=>** {

*//=(1)=========================================================*

*//comment out the following Array Approach and replace it with*

*// an Array Based Map Reduce function*

*// The array that you will work with is numbers.*

*// No conversion required.*

*// Solution Code required 2 lines: 1 to calculate 1 to print*

*// ------------------------------------------------------------*

*// var result = 0;*

*// for (let i = 0; i < numbers.length; i++) {*

*//     result += numbers[i];*

*// }*

**let** result = numbers.reduce((acc, curr) **=>** {

        return acc + curr;

    })

    $("#results\_id").val(result);

*//=(1)=========================================================*

}

# Question 4

"use strict";

*// Conversion from Global Variables based problem*

*// to Class Object based problem*

*// There are five checkpoints for this problem (1) ... (5)*

*// =(1)======== COMMENT OUT THE FOLLOWING 3 LINES =======*

*//                                 (a) (b) (c)*

*// Converting from use of local Variables*

*// to Class Variables begins with this required deletion*

*// ------------------------------------------------------*

*// var item\_name; // (a)*

*// var item\_wholesale; //(b)*

*// var item\_markup; //(c)*

*// =(1)==================================================*

$(document).ready(() **=>** {

    loadDataIntoClass();

    preDisplay();

    $("#calculate\_selling\_id").click(event **=>** {

        displaySellingPrice();

    });

});

*// =(2)=================================================================*

*// Complete the class 'StoreItem'*

*// set all three attributes of the attributes to null*

*// you will require (1) item\_name (2) item\_wholesale (3) item\_markup*

*// 3 lines of code for the constructor 2 lines of code for selling price*

*// ----------------------------------------------------------------------*

**class** StoreItem {

**constructor**() {

        this.item\_name = null;

        this.item\_wholesale = null;

        this.item\_markup = null;

    }

*// you will require a get sellingPrice method*

*// formula is selling price = item\_wholesale + item\_markup*

**get** sellingPrice() {

        return this.item\_wholesale + this.item\_markup;

    }

}

*// =(2)===============================================================*

**const** loadDataIntoClass = () **=>** {

*// =(3)===============================================================*

*// comment out the three lines marked as COMMENT OUT*

*// local variable are not used in the solution (a) (b) (c)*

*// item\_name = "Cookies"; // (a) Comment out*

*// item\_wholesale = 5.95; // (b) Comment out*

*// item\_markup = 1.11; // (c) Comment out*

*// Add in the correspond 3 lines (d) (e) (f) to populate your new class 'storeItem'*

*// with (d) item\_name = "cookies", (e) item\_wholesale = 5.95, (f) item\_markup = 1.11*

*// -------------------------------------------------------------------------------*

    storeItem.item\_name = "cookies"; *// (a)*

    storeItem.item\_wholesale = 5.95 *// (b)*

    storeItem.item\_markup = 1.11 *// (c)*

*// =(3)===============================================================*

}

**const** preDisplay = () **=>** {

*// =(4)====================================================================*

*// Comment out the following 3 lines marked as (a) (b) (c)*

*// display is no longer based on local variables*

*// -------------------------------------------------------------------------*

*// $("#name\_id").val(item\_name); // (a) comment out this line*

*// $("#wholesale\_id").val(item\_wholesale); // (b) comment out this line*

*// $("#markup\_id").val(item\_markup); // (c) comment out this line*

*// create 3 lines of code to populate the form display variables based on*

*// the 3 class attributes (d) (e) (f)*

*// --------------------------------------------------------------------------------*

    $("#name\_id").val(storeItem.item\_name); *//(d)*

    $("#wholesale\_id").val(storeItem.item\_wholesale); *//(e)*

    $("#markup\_id").val(storeItem.item\_markup); *//(f)*

*// =(4)============================================================================*

}

**const** displaySellingPrice = () **=>** {

*// =(5)====================================================================*

*// Comment out the following 2 lines marked as comment out (a) (b)*

*// The calculations are not done locally and will a class method*

*// ----------------------------------------------------------------------------*

*// var item\_selling = item\_wholesale + item\_markup; // (a) comment out this line*

*// $("#selling\_id").val(`$ ${item\_selling.toFixed(2)}`); // (b) comment out this line*

*// create 1 line of code (c) to display the selling price using the class function*

*// the class function involved is storeItem.sellingPrice*

*// -----------------------------------------------------------------------------*

    $("#selling\_id").val(`$ ${storeItem.sellingPrice.toFixed(2)}`); *// (c)*

*// =(5)============================================================================*

}

**var** storeItem = new StoreItem();

# Question 5

"use strict";

**var** current\_guess = null;

*// There are 4 Checkpoints (1) ... (4)*

*// =(1)============================================*

*//      Comment out the following line*

*//  The secret\_number will be placed in an enclosure*

*//  go to the end of the document for checkpoint (2)*

*// const secret\_number = 123; // comment this line out*

*// =(1)============================================*

**var** cnaSecretNumber = null;

$(document).ready(() **=>** {

    activateEnclosure();

    $("#guess\_id").focus();

    $("#check\_guess\_btn").click(() **=>** {

        if (validateGuess()) {

            $("#display\_id").text("Congratulations !!! you WIN !!!");

        }

        else {

            $("#display\_id").text("Try Again !!!");

        }

    });

});

*// =(3)===================================================*

*//    Create an enclosure with one attribute*

*//    and one method.*

*//    Attribute is secretNumber = 123*

*//    hardcode the number - There is no set method*

*//    provide a method call getSecret that will return*

*//    the secretNumber.*

*//*

*//    Complete the enclosure in the space provided*

*//    within generateSecretVar*

*//    Solution key used 5 lines*

*// -----------------------------------------------------------*

**const** generateSecretVar = () **=>** {

**let** secret\_number = 123;

**const** getSecret = () **=>** {

        return secret\_number;

    }

    return {getSecret};

};

*// =(3)===================================================*

*// =(4)===================================================*

*// Fill in the one line that will activate the enclosure*

*// in the space provide within activateEnclosure function*

*// ---------------------------------------------------------*

**const** activateEnclosure = () **=>** {

    cnaSecretNumber = generateSecretVar();

}

*// =(4)===================================================*

**const** validateGuess = () **=>** {

**const** guessObj = $("#guess\_id");

**var** guessText = guessObj.val();

    if (guessText == "" || *//(1)*

        isNaN(guessText) || *//(2)*

        !Number.isInteger(parseFloat(guessText))) *//(3)*

    {

        guessObj.next().text("Invalid Guess - Enter 00 to 99");

        current\_guess = null;

        return false;

    }

    current\_guess = parseInt(guessText);

    if (current\_guess != lookUpSecretNumber()) {

        current\_guess = null;

        guessObj.next().text("Incorrect Guess - Not the Secret Number");

        return false;

    }

    guessObj.next().text("\*");

    return true;

}

*// =(2)============================================================================*

*// Switch from the Variable Version over to use Class Version*

*// Comment out the (a) line*

*// remove the comments from the (b) line - this will switch to use your class function*

*// ----------------------------------------------------------------------------------*

**const** lookUpSecretNumber = () **=>** {

*// return secret\_number; // (a) comment this line OUT to disable use of*

*//     Local Vars in the solution*

     return cnaSecretNumber.getSecret();  *// (b) remove the comment form this line*

*//     to use closure function*

}

*// =(2)============================================================================*

*//=========================== END OF CODE ================================*