

CP1295
Advanced JavaScript
Final Exam 2024 v4
(30%)
PRACTICAL COMPONENT



Time: 3 Hours

The quiz consists of

Part (A) 35 multiple choice questions worth 1 marks each 35 marks total and

Part (B) Solve program problems 5 problems 7 marks each 35 Marks

Total marks for test: 70 Marks.

Test is worth 30% of term grade.

The test password will be released during the test.

Must Complete the Multiple-Choice section FIRST, then start the PRACTICAL component. (Drop Box)

Practical component will require use of Visual Studio. (or Equivalent).

Contents

P1. Programming Problem 1	3
i. P1-Problem Description	3
ii. P1-Starter Code	4
iii. P1-Solution Screen Shots	6
iiii. P1-Submission Requirements.....	8
P2. Programming Problem 2	9
i. P2-Problem Description	9
ii. P2-Starter Code	10
iii. P2-Solution Screen Shots	11
iiii. P2-Submission Requirements.....	13
P3. Programming Problem 3	14
i. P3-Problem Description	14
ii. P3-Starter Code	15
iii. P3-Solution Screen Shots	16
iiii. P3-Submission Requirements.....	17
P4. Programming Problem 4	18
i. P4-Problem Description	18
ii. P4-Starter Code	19
iii. P4-Solution Screen Shots	22
iiii. P4-Submission Requirements.....	23
P5. Programming Problem 5	24
i. P5-Problem Description	24
Goal is to convert from a variable use of 'secretNumber' to creation of an enclosure called cnaSecretNumber that has a variable called secret_number that could only be accessed by 'cnaSecretNumber.getSecret'.....	24
ii. P5-Starter Code	25
iii. P5-Solution Screen Shots	27
iiii. P5-Submission Requirements.....	28
Once finished the text, upload up to D2L Drop Box.....	28
Wait for Test Proctor to verify that test is uploaded and readable.....	28
E. SCOPE RULES	29

P1. Programming Problem 1

i. P1-Problem Description

Based on Lecture 09 Chapter 13

Key Points

- Regular Expressions
- Range Testing
 - *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*

The starter Screen Shot and Solution Screen shot will describe the operational details of the P1 model.

All code except the key points (1) and (2) are completed and are not part of this question's objectives.

ii. P1-Starter Code

pageP1.js

All files are available from the drop box. (pageP1.html, pageP1.css, pageP1.js.

There are only TWO areas that require your attention.

Area (1). Regular expression section.

Create the code here. The solution used 4 lines of code.

Area (2). Range testing section. The numbers are valid (00 to 99) and both add up to 100, then return a TRUE otherwise return a FALSE. Basic testing for "" and NaN have been already done. The solution used 5 lines of code.

pageP1.js

```
"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
    // =====
    
    //=(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);

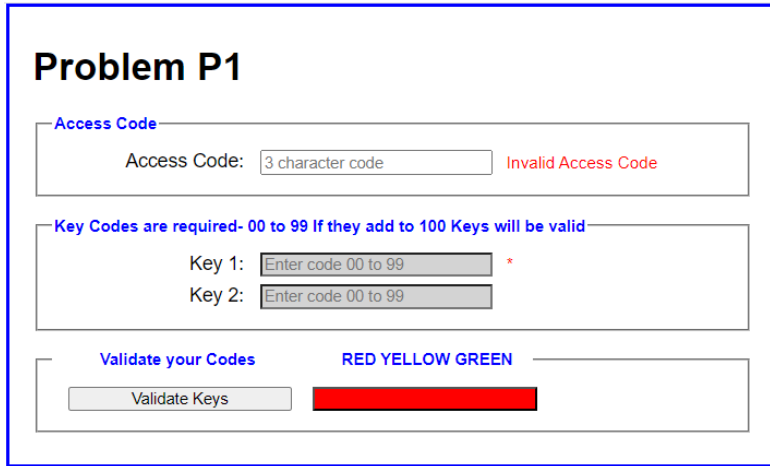
  // =(2)=====
  // RANGE TESTING component
  // Data has been validated for blank and number testing
  // and has been converted to itegers
  // your goal is to ensure that k1 and k1 are both in the
  // range of 00 to 99 (inclusive)
  // You are not to use Regular Expressions testing for this
  // checkpoint(2) as it was it was already used for checkpoint (1)
  // Fill in the required code in the space provided.
  // Solution key used 5 lines of code.
  // -----
  //=(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("");
  }
}

```

iii. P1-Solution Screen Shots

Initial Screen Shot



The screenshot shows a web form titled "Problem P1". It contains three main sections: 1. "Access Code" section with a label "Access Code:", a text input field containing "3 character code", and a red error message "Invalid Access Code". 2. "Key Codes" section with a blue instruction "Key Codes are required- 00 to 99 If they add to 100 Keys will be valid". It has two rows: "Key 1:" and "Key 2:", each followed by a disabled text input field containing "Enter code 00 to 99" and a red asterisk. 3. "Validate your Codes" section with a blue label "Validate your Codes", a blue text "RED YELLOW GREEN", a "Validate Keys" button, and a solid red rectangular bar.

Operational Notes

(1) The color Bar is RED. The Keys are Disabled.

First Enter the correct access Code.
Enter 3 Letters (upper or lowercase)

This is your regular expression challenge. (Task 1)

And press **Validate Keys to proceed to part 2.**

In the code. Locate the sections of code that requires your attention to resolve the challenges.

Look for the RED RECTANGLE that will require your attention.

Problem P1

Access Code

Access Code: Valid Access Code

Key Codes are required- 00 to 99 If they add to 100 Keys will be valid

Key 1: Invalid Key Pair

Key 2:

Validate your Codes RED YELLOW GREEN

Operational Notes

(2) Once the Access Code has been accepted the color bar changes to YELLOW

Next step is to enter the two keys.

The range of 00 to 99. This is your Range Testing. (Task 2)

The two number must add up to 100 to complete the validation test.

Onto the final screen shot.

In the code. Locate the section of code that requires your attention to resolve the Task 2 challenge.

Look for the RED RECTANGLE that will require your attention.

Problem P1

Access Code

Access Code: Valid Access Code

Key Codes are required- 00 to 99 If they add to 100 Keys will be valid

Key 1: KEYS PAIR is VALID !!!

Key 2:

Validate your Codes RED YELLOW GREEN

All is good.

Access Code and Keys are good.

Green BAR is displayed.

iiii. P1-Submission Requirements

Create a word document (if you have not already started one).

Page 1 – Your name, student number, and course number (CP1295)

Page 2 – Copy your solution code **pageP1.js** in its entirety into the word document.

Copy the code as TEXT. Do not use screen shots. The code will be copied for grading purposes onto a test computer.

Be sure that you modified only the code in the Red Rectangle(s).

P2. Programming Problem 2

i. P2-Problem Description

Key Points

- Based on Lecture 10 Chapter 14
- Using Node.JS
- Using Cookies
 - *Goal of the problem is to correct 2 cookie related problems in train.js. Do Not Make any modifications to Station.js*
 - *(1) add in a setCookie call that will generate a location cookie. One line of code required*
 - *(2) complete the setCookie Function. Solution used 5 lines of code.*

This question requires the use of 'node.js'.

(1) use command prompt

(2) navigate to location of P2 code

(3) enter 'npx http-server'

Check for confirmation that http-server is running.

You are ready to work on the 2 required tasks for this problem.

ii. P2-Starter Code

train.js

```

"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)=====
            // setCookie code is missing (a) Add in missing code where indicated
            var selRadioSpeed = $("[name=speed]:radio:checked").val();
            //(a)
            

            $("#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)=====
}

```

iii. P2-Solution Screen Shots

Be sure that you start the node.js.

Be sure that you see port 8080 as shown below.

Use comment prompt and verify that you are in the correct directory before you issue the command

npx http-server

```
H:\CP1295-Solution\P2>npx http-server
```

And verify that your server has started.

```
Available on:
http://10.200.94.12:8080
http://192.168.56.1:8080
http://192.168.63.1:8080
http://192.168.111.1:8080
http://127.0.0.1:8080
Hit CTRL-C to stop the server
```

TRAIN OPERATIONS V1

Press CTRL + SHIFT + DEL to Clear History between code changes

CLICK on one of the links below to use port 8080

Train Location

Location:

☒ St. John's

☐ Gander

☐ Corner Brook

Train Speed

Speed:

☒ Stopped

☐ Slow

☐ Fast

Transmit Location and Speed Data

[TRAIN](#)

[Train Station](#)

Notice the TRAIN link at the foot of the screen, this will take this screen into port 8080 so that Node.JS can process the cookies.

Select a location and speed then click on **Send Data to Station**.

This should generate the cookie.

Note the name of the cookies that are required.

(1) location

(2) speed

Name	Value
location	St.%20John's
speed	Stopped

Your diagnostics should verify the generation of the cookie. Be sure to use refresh as required.

(part 2) **Click on Train Station**

At the Train Station

CNA Train Station V1

Press CTRL + SHIFT + DEL to Clear History between code changes

CLICK on one of the links below to use port 8080 as required

Train Information

Location:

Speed:

Train Data Update

Check Train Status

Train
STATION

Click on **Check Train Status** to request the form to read and display the cookie.

Station code is working code. Do Not Modify the Station Code in any way.

CNA Train Station V1

Press CTRL + SHIFT + DEL to Clear History between code changes

CLICK on one of the links below to use port 8080 as required

Train Information

Location:

Speed:

Train Data Update

Check Train Status

Train
STATION

If the cookie data is displayed, then you have completed the task.

iiii. P2-Submission Requirements

Only the Train code is part of the question requirements.

Create a word document (if you have not already started one).

Page 1 – Your name, student number, and course number (CP1295)

Page 3 – Copy your solution **train.js** in its entirety into the word document.

Copy the code as TEXT. Do not use screen shots. The code will be copied for grading purposes onto a test computer.

Be sure that you modified only the code in the Red Rectangle(s).

P3. Programming Problem 3

i. P3-Problem Description

Based on Lecture 11 Chapter 15

Key Point

- Array MAP reduce function
 - *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. Solution used 2 Lines of code.*

ii. P3-Starter Code

pageP3.js

```

"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 Solution used 2 lines of code

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];
  }
  $("#results_id").val(result);
  //=(1)=====
}

```

iii. P3-Solution Screen Shots

Map Reduction Problem

Process Selection

Total Price:

- Bars 2.95
- Chips 3.95
- Cookies 5.95
- Batteries 9.95
- Detergent 12.95

Process

The Solution Screen shot is based on using the Array May Reduce line of code.

Whereas the Initial Screen shot is based on using the Iterative Array in an addition loop.

The two screens should be identical.

There is only 1 button to press to calculate the sum of the array of numbers and display the result in the text box.

iiii. P3-Submission Requirements

Only the pageP3 code is part of the question requirements.

Create a word document (if you have not already started one).

Page 1 – Your name, student number, and course number (CP1295)

Page 4 – Copy your solution **pageP3.js** in its entirety into the word document.

Copy the code as TEXT. Do not use screen shots. The code will be copied for grading purposes onto a test computer.

Be sure that you modified only the code in the Red Rectangle(s).

P4. Programming Problem 4

i. P4-Problem Description

Based on Lecture 12 Chapter 16

Key Points

- Classic Class will be required.

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

- (1) commenting out the local variables
- (2) composing the StoreItem Class
 - 3 attributes
 - 1 function to calculate sellingPrice
 - 5 lines of code
- (3a) commenting out attribute equivalent to setting of the attributes
- (3b) adding in the setting of the class attributes (directly to attributes)
- (4a) commenting out the display lines based on the attributes
- (4b) add in the display based on class attributes
- (5a,b) commenting out the calculation and display of the calculation base on attributes
- (5c) add in display based on class function sellingPrice

ii. P4-Starter Code

pageP4.js

```

"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() {

    }

    // you will require a get sellingPrice method
    // formula is selling price = item_wholesale + item_markup

    get sellingPrice() {

    }

}
// =(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
    // -----

    // (a)
    // (b)
    // (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)
    // -----

    //(d)
    //(e)
    //(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
    // -----
    // (c)

    //
    // =(5)=====
}

var storeItem = new StoreItem();
```

iii. P4-Solution Screen Shots

Classical Class Problem

Convert from using variables to using a Class Object

Data Elements

Name:

Wholesale:

Markup:

Process

Starting screen shot uses variables to store the attributes name, wholesale, markup.

The mathematics is to simply add the two numbers to generate the selling price.

This is triggered by pressing the **Calculate Setting Price button**.

The requirements are to convert from a variable based program to a **Class Object** based version using Class code.

iiii. P4-Submission Requirements

Only the pageP4 code is part of the question requirements.

Create a word document (if you have not already started one).

Page 1 – Your name, student number, and course number (CP1295)

Page 5 – Copy your solution **pageP4.js** in its entirety into the word document.

Copy the code as TEXT. Do not use screen shots. The code will be copied for grading purposes onto a test computer.

Be sure that you modified only the code in the Red Rectangle(s).

P5. Programming Problem 5

i. P5-Problem Description

Based on Lecture 14 Chapter 15

Key Points

- Use of Enclosure to replace existing variable use

Goal is to convert from a variable use of 'secretNumber' to creation of an enclosure called cnaSecretNumber that has a variable called secret_number that could only be accessed by 'cnaSecretNumber.getSecret'.

ii. P5-Starter Code

pageP5.js

```

"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 = () => {
    
};

// =(3)=====

// =(4)=====
//   Fill in the one line that will activate the enclosure
//   in the space provide within activateEnclosure function
// -----
const activateEnclosure = () => {
    
}

// =(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 from this line
  //                                     //      To use the closure function
}

// =(2)=====
//===== END OF CODE =====

```

iii. P5-Solution Screen Shots

Problem P5 - Create a Closure

Enter Guess Zone

Your Guess: *

Validation Section

Congratulations !!! you WIN !!!

Simple guessing game.

Solution keeps the secret number in a closure.

Enter a guess and press **Check Guess** button to access the get method to access the secret number and perform an equality test.

The screen shots are the same for the starter code and the solution.

iiii. P5-Submission Requirements

Only the pageP5 code is part of the question requirements.

Create a word document (if you have not already started one).

Page 1 – Your name, student number, and course number (CP1295)

Page 6 – Copy your solution **pageP4.js** in its entirety into the word document.

Copy the code as TEXT. Do not use screen shots. The code will be copied for grading purposes onto a test computer.

Be sure that you modified only the code in the Red Rectangle(s).

Once finished the text, upload up to D2L Drop Box.

Wait for Test Proctor to verify that test is uploaded and readable.

E. SCOPE RULES

Inclusions

- (1) Code must be based on code demonstrated in this course or its pre-requisite course(s)
 - Course Text Book
 - Course Notes
 - Course Handouts
- (2) DOM element selection techniques:
 - i. `document.querySelector(sel)`
 - ii. `document.querySelectorAll(sel)`
 - iii. for jQuery `$()`
- (3) jQuery can be used to add existing elements created by `document.createElement`
 - i. but cannot be used to create elements such as “li” and “ul”
- (4) All elements have to be created using `document.createElement()`

Exclusions

- (1) Code must follow the following exclusion rule(s)
 - a. Note: `getElementByTag`, `innerHTML`, `outerHTML` are not permitted in this test.
 - b. Use of jQuery for element processing is restricted
 - i. jQuery cannot be used to create elements (directly or indirectly)

Submission Rules

- (1) Word Document that contains the following
 - a. Your name and student number.
 - b. Shots (if any) as indicated in the instructions for each question.
 - c. JavaScript must be copied and pasted into Word Document as TEXT. DO NOT USE Screen shots of your java code. This will void the test as the test cannot be graded from screen shots of JavaScript code.

End of Test