CP1295

Advanced JavaScript

Test #3 Practical Component

Value of Test 10%

This Section is worth 30 out of 60 marks.

Time: 90 Minutes

Scope Rules are on last page.

There are three questions.

For each question there are three tasks required. They will be one of the following:

(1) Fix a Line of code. (OR)

(2) Fill in one line of missing code

All areas of concern are indicated on this document with a RED rectangle.

On the test there are comments to direct you to each of the areas of concern.

Note: EACH identified problem can be corrected by ONE LINE of code.

You can download the starter code with the test.

A copy of each JavaScript starter code is on this test.

No modifications to any HTML or CSS documents permitted.

There are 10 marks per questions.

One of the three points that require correcting will be worth 4 points. The others will be 3 points.

Look for the "***" marker for the checkpoint item that is worth 4 points.

Submission Requirements.

Create a Word Document.

On Page 1 – include (1) Your Name (2) Your student number

When finished the test, upload your Word document to the server. DO not use any compression tools such as ZIP.

Question 1.

Opening Screen

Concept: Enter a valid secret code (3 uppercase letters) and you win a free movie pass.

Valid Data Test

Invalid Data Test. Blank data would display the same page.

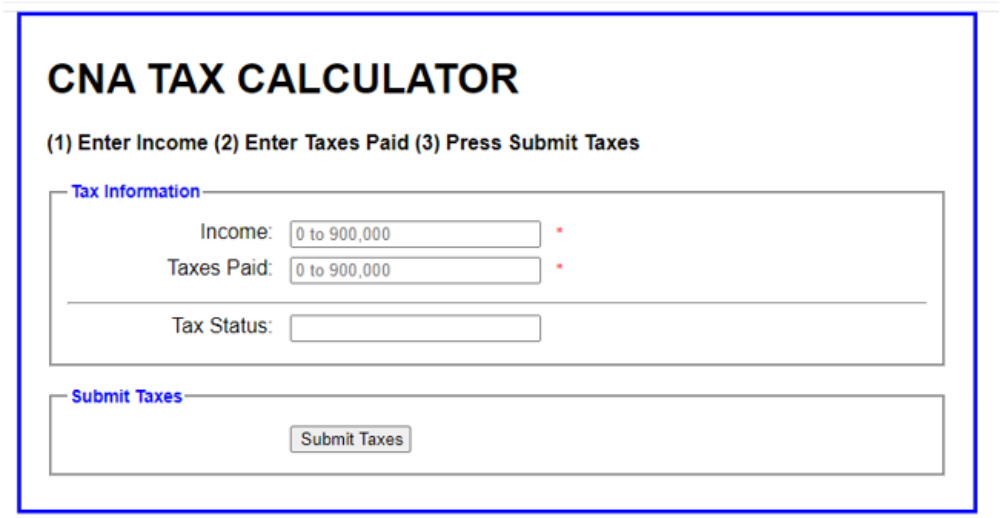
Starter Code

```
"use strict";

$(document).ready(() => {
    console.debug("document ready");
    $("#apply_id").click(event => {
        const passCode = $("#pass_id")
        //PLACE HOLDER                  // (1) Fill in the missing Line
    });
});

const check = (passCode) => {
    const itemCodePattern = /[0-9]/; // (2) Fix this line ***
    var testPassCode = passCode.val().trim();
    if (testPassCode == "") {
        passCode.next().text(`No Free Ticket`);
    } else if (!itemCodePattern.test(testPassCode)) {
        passCode.next().text(`No Free Ticket`);
    } else {
        alert("free ticket"); // fix this line Should  display free ticket
    }
}
```

Submission Requirements – Copy the JavaScript Code to the word document as TEST.

Question 2.

Opening Screen



Program Operation is as displayed.

The calculations are:

Tax is calculated at 50% of your income.

Tax for person with income of 6,000 is 3,000$. (Expensive place to live)

Taxes paid will be applied as a credit, such as 1000$

Tax Status

(1) The calculation Tax = Income / 2

(2) Amount Owing = Tax – Credit

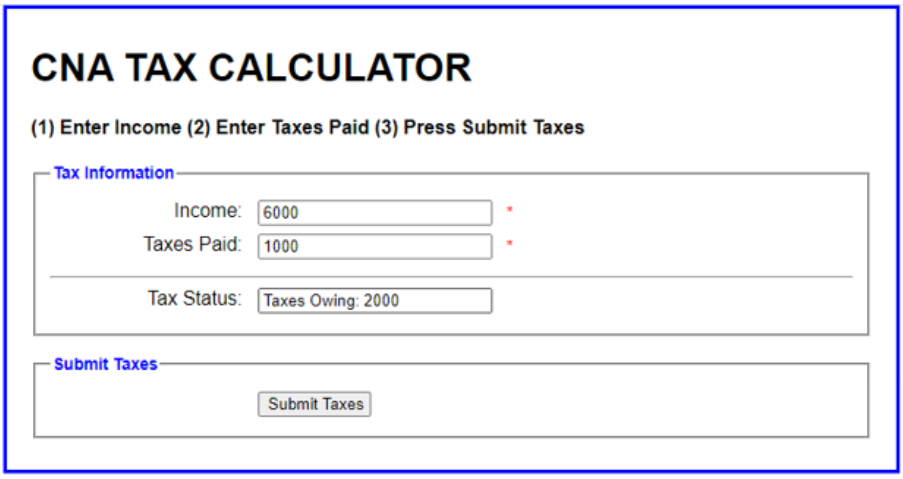
IF Amount Owing > 0 then you will see a message “Taxes Owing 2000”

If Amount Owing <=0 then you will see a message “Refund: 2000”

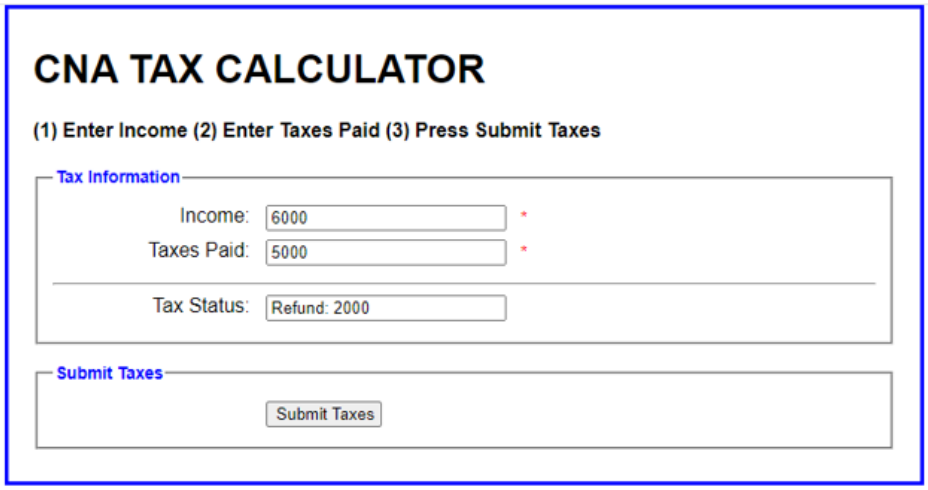
Income 6000, Taxes Paid 5000.

Taxes would be 3000. IF you overpaid by 2000. Message would indicate a Refund of 2000

Screen shot of Taxes Owing

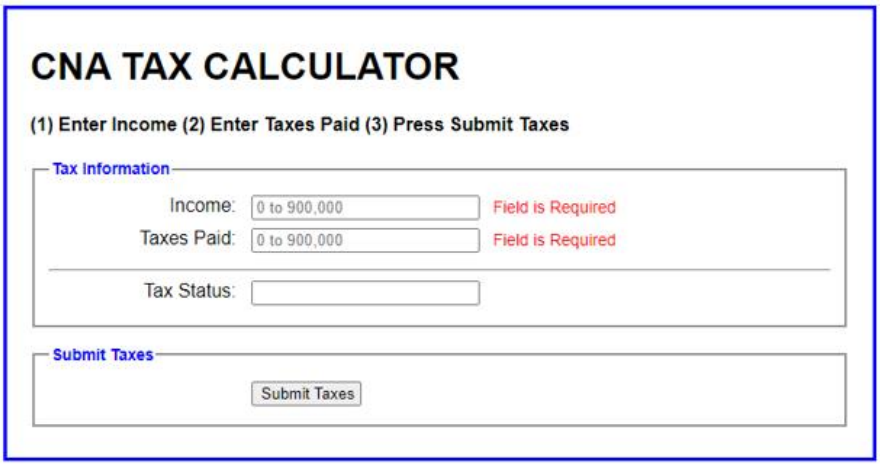


Screen Shots of someone who has overpaid their taxes and will get a refund.

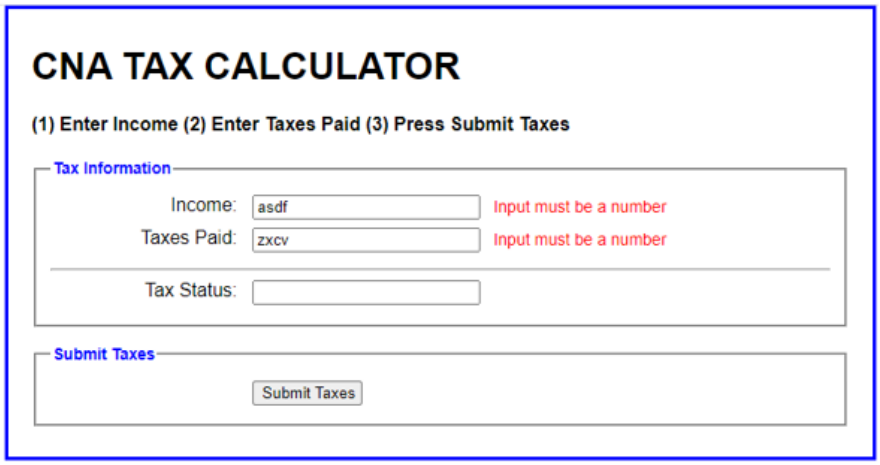


This Question has some basic error testing. In Code the error system is based on the Try/Catch system.

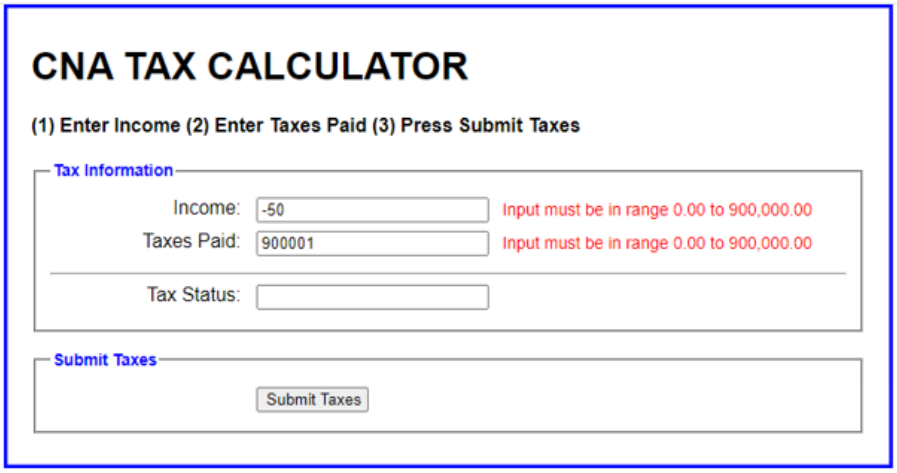
The blank field test. Notice the error messages in the span element.



Invalid Numbers



Range Testing. Valid Range is 0 to 900000



Starter Code

```
"use strict";
var income_val;
var taxesPaid_val;

$(document).ready(() => {
    console.debug("Document is Open");
    $("#submit").click(event => {    //(1) Button Problem
        const income = $("#income_id")
        try {
            income_val = testInput(income);
            income.next().text("*");
        }
        catch (error) {
            income.next().text(error.message);
        }

        const taxesPaid = $("#taxes_paid_id")
        try {
            taxesPaid_val = testInput(taxesPaid);
            taxesPaid.next().text("*");
        }
        catch (error) {
            taxesPaid.next().text(error.message);
        }
        calculateTax();
    });
});

const testInput = (testInput) => {
    var testNumber = testInput.val().trim();
    // blank test
    if (testNumber == "") {
        throw new Error();     // (2) Problem with Blank Messages
    }
    // Valid Number Test
    var testNumberFloat = parseFloat(testNumber)
    if (isNaN(testNumberFloat)) {
        throw new Error("Input must be a number");
    }
```

```
    // Range Test
    if (testNumberFloat < 0.0 || testNumberFloat > 900000.0) {
        throw new Error("Input must be in range 0.00 to 900,000.00");
    }
    // If you reached this point there are NO errors
    return testNumberFloat;
}

const calculateTax = () => {

    const taxStatus = $("#status_id");

    var taxAmount = income_val;(3) Problem with Tax Calculations***
    var creditAmount = taxesPaid_val;

    if ( !isNaN(taxAmount) && !isNaN(creditAmount)) {
        var taxesOwing = taxAmount - creditAmount;
        if (taxesOwing > 0.0)
            taxStatus.val(`Taxes Owing: ${taxesOwing}`);
         else{
            var refund = taxesOwing * -1;
            taxStatus.val(`Refund: ${refund}`);
         }
    }
    else
      taxStatus.val(``);
}
```

Problems are:

(1) The button on the form does not work.

(2) The Blank Error message does not work.

(3) The Tax Amount calculation has a problem

Question 3

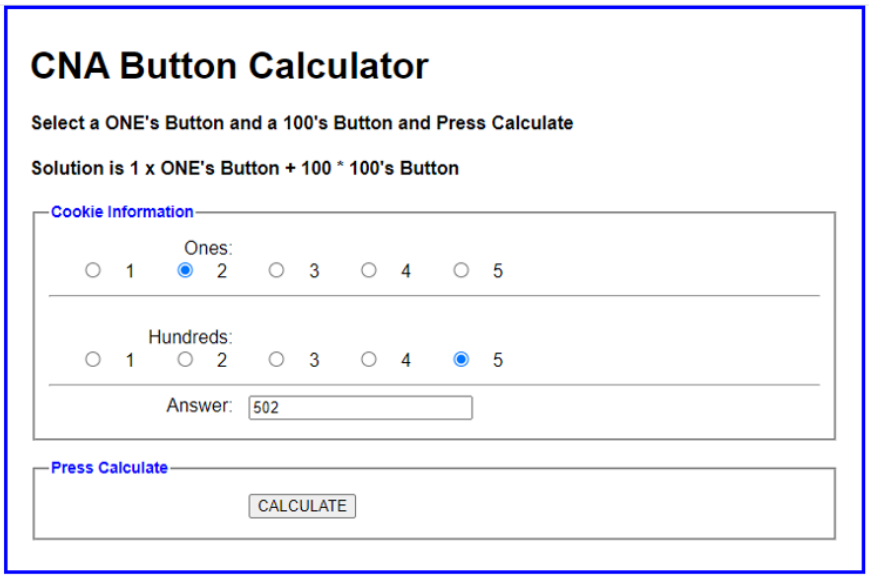
This is a calculator program that uses radio buttons.

Operation is to select a one's radio button and

Select a Hundreds radio button. 500 hundred form Hundreds added to 2 from the Ones.

There are only 5 buttons in each of the two groups. No values above 5.

Display shows in the Answer Box.



Starter Code

```
"use strict";

var ones_value = 0;
var hundreds_value = 0;
var answer = 0;

$(document).ready(() => {
    $("#calculate_id").click(event => {
        event.preventDefault();
        readButtons();
        calculateAndDisplay();
    });
});

const readButtons = () => {
    ones_value = parseInt($("input[name=ones_code]:checked").val());
   // place holder (1) Can't read the hundred's buttons ***
}

const calculateAndDisplay = () => {
   // place holder (2) Missing the Calculations
   // Place holder (3) Missing the Display refresh
}
```

Problems are:

(1) the hundreds buttons cannot be read

(2) the calculation equation is missing

(3) the results are not being displayed back to the form.

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. Two screen shots as indicated in the instructions.

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