SOFE2800 – Lab 2

Forms and JavaScript

This lab introduces the basics of **HTML forms** and **JavaScript** through building simple web pages. By completing hands-on activities, you will gain a clear understanding of how these technologies work together.

- **HTML Forms** allow us to collect user input, which is typically sent to a server for processing.
- **JavaScript** makes web pages more dynamic and interactive.

The lab consists of a series of activities and exercises designed to give you practical experience with these concepts.

Before attending the lab, be sure to complete **Prelab 2**. Also, remember that web pages may look slightly different across browsers. For consistency, this course recommends using **Google Chrome**.

Note: Review the debugging section from **Prelab 2** to assist you in this lab.

Create a portfolio showcasing your favorite character

Activity 1: Google Search

A full Google Search URL looks like this: https://www.google.com/search?q=Toronto

Here:

- https://www.google.com/search is the domain for Google Search.
- **q** is a variable that represents the search query.
- **Toronto** is the value assigned to the variable **q**.

This format corresponds to the GET request method.

In this activity, you will create an HTML form that sends a Google Search query.

Steps:

1. Create a new file and save it as **index.html**.
2. Add the basic **HTML** structure and set the page title.
3. Inside the `<body>` tag, create a form named **searchForm** with the action set to http://www.google.com/search. By default, the form will use the **GET** method, so submitting it will send a **GET** request to **Google Search**.
4. Inside the form, add a **label** with the text *Search Query*.
5. Add an `<input>` tag of type **text** with the name **q**. This creates a text box where the user can enter a query. The entered text will be assigned to the variable **q** and included in the request **URL**.
6. Add another `<input>` tag of type submit with the value *Search Google*. This creates a **button** that submits the form when clicked.

The completed HTML file should look like this:

```
14  <body>
15
16  <h2>Google Search</h2>
17
18      <form name="searchForm" action="https://www.google.com/search">
19        <label>Search Query </label>   <input type="text" id="searchText" name="q" value="">
20        <input type="submit" value="Search Google"> <br/><br/>
21      </form>
22  <hr/>
```

When opened in a browser, it will appear as follows:

Activity 2: Simple JS code

This activity builds on the previous one by adding JavaScript code.

1. Add a `<script>` tag to link the HTML file with a JavaScript file named **script.js**.
2. In the form, include the attribute **onsubmit="return validateSearchForm()"** so that the JavaScript function **validateSearchForm()** runs to check the form content before submission.

```
14  <body>
15
16  <script src="script.js"></script>
17  <h2>Google Search</h2>
18
19      <form name="searchForm" action="https://www.google.com/search"
20          onsubmit="return validateSearchForm()">
21        <label>Search Query </label>   <input type="text" id="searchText" name="q" value="">
22        <input type="submit" value="Search Google"> <br/><br/>
23      </form>
24  <hr/>
```

3. Create a new file in the same directory as your **HTML** file and name it **script.js**.
4. Define the **validateSearchForm** function:
   - Start by creating a variable **x** that stores the value of the **q** textbox. One way is to first reference the form with **document.forms["searchForm"]**, then access the textbox using **document.forms["searchForm"]["q"]**.
   - Check the value of **x**. If it is an empty string, display an alert box with the message *"The search query must be filled out"* and return **false** to prevent submission.

```
1   function validateSearchForm() {
2     let x = document.forms["searchForm"]["q"].value;
3     if (x == "") {
4       alert("The search query must be filled out");
5       return false;
6     }
7     else{
8       return true;
9     }
10  }
```

Exercise 1:

- Modify the second line in the **validateSearchForm** function so that the textbox is accessed by its **id** instead of its **name**.

Activity 3: Simple Calculator

In this activity, you will create a simple calculator using HTML forms and JavaScript to evaluate the results.

1. Append a new form to the HTML file named **simpleCalculator** with the following elements:
    - A label: **"Enter the Expression"**
    - An input of type **text**, named **operand1**, with an empty value.
    - A drop-down list named **operator** containing four options: +, -, *, and / (each with the corresponding value).
    - An input of type **text**, named **operand2**, with an empty value.
    - A button that triggers a JavaScript function named **calculate()** when clicked.
    - A read-only input of type **text**, named **result**, with an empty value (to display the outcome).

Since all calculations are performed on the client side, no form submission is required. Therefore, set the **onsubmit** attribute to "**return false;**".

```
17          <h2> Simple JS Calculator </h2>
18          <form name="simpleCalculator" onsubmit="return false;">
19              <label> Enter the experssion </label>
20              <input type="text" name="operand1" value="" />
21              <select name="operator">
22                  <option value="+">+</option>
23                  <option value="-">-</option>
24                  <option value="*">*</option>
25                  <option value="/">/</option>
26              </select>
27              <input type="text" name="operand2" value="" />
28              <button onclick="calculate()" > = </button>
29              <input class="readOnlyTextBox" type="text" name="result" value="" readonly /> <br/> <br/> <br/>
30          </form>
31          <hr/>
```

2. In **script.js**, add the function **calculate()**:
    - Retrieve the value from the **operand1** textbox, convert it to a number, and store it in the variable **op1**.
    - Retrieve the selected operator from the **operator** drop-down list and store it in the variable **oper**.
    - Retrieve the value from the **operand2** textbox, convert it to a number, and store it in the variable **op2**.

- Based on the operator (**oper**), compute the result and assign it to the variable **result**.
- Finally, display the value of **result** inside the **result** textbox.

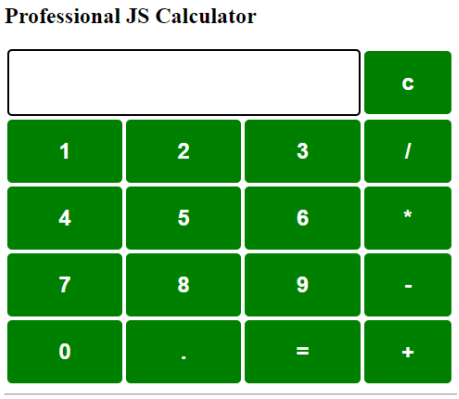
```
function calculate(){
    let op1 = Number(document.forms["simpleCalculator"]["operand1"].value);
    let oper = document.forms["simpleCalculator"]["operator"].value;
    let op2 = Number(document.forms["simpleCalculator"]["operand2"].value);
    let result=0;
    switch(oper){
        case '+':
            result=op1+op2;
            break;
        case '-':
            result=op1-op2;
            break;
        case '*':
            result=op1*op2;
            break;
        case '/':
            result=op1/op2;
            break;
        default:
            alert("Invalid operator");
            return;
    }
    document.forms["simpleCalculator"]["result"].value=result;
}
```

Exercise 2:

- Modify the **calculate()** function to validate the contents of the **operand1** and **operand2** textboxes before performing any calculations or displaying a result. The validation should handle the following cases:
  - When either textbox is left blank.
  - When a non-numeric value is entered.
  - When attempting to divide by zero.

Activity 4: Professional Calculator

Now it's time to create a professional calculator. It will look like this:



1. **Add the MathJS library**
   - Include the **math** JavaScript library to evaluate expressions.
   - You can copy the script tag from https://cdnjs.com/libraries/mathjs.
2. **Update the HTML file** to include a new form for the calculator:
   - Disable the form's submission feature.
   - Use a borderless table to align all form components neatly.
   - The first row should contain:
     - A read-only textbox spanning three columns to display the expression and result.
     - A **C** button to clear the textbox using the **clr()** function.
   - All other rows should have four buttons each:
     - Each button calls the **dis()** function, which appends the clicked key to the result textbox.
     - The **=** button calls the **solve()** function, which evaluates the expression and displays the result.
   - Assign the buttons to a class named **key** and the textbox to a class named **result**. These classes will be styled using a separate CSS file.

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/mathjs/13.1.1/math.js"
integrity="sha512-fMI/ndUubOcpzblrAaOFszYIE2iDKua+Cxg9dZp3cAkhsRWEHHbKtFCE1HEQ50viX502yj7nFEEcxq2SOTNrIg=="
crossorigin="anonymous" referrerpolicy="no-referrer"></script>
<h2>Professional JS Calculator</h2>
<form name="proCalculator" onsubmit="return false;">
    <table id="calculator">
        <tr>
            <td colspan="3"><input class="result" type="text" name="result" readonly></td>
            <td><input type="button" class="key" value="c" onclick="clr('proCalculator')" /> </td>
        </tr>
        <tr>
            <td><input type="button" class="key" value="1" onclick="dis('1','proCalculator')"> </td>
            <td><input type="button" class="key" value="2" onclick="dis('2','proCalculator')"> </td>
            <td><input type="button" class="key" value="3" onclick="dis('3','proCalculator')"> </td>
            <td><input type="button" class="key" value="/" onclick="dis('/','proCalculator')"> </td>
        </tr>
        <tr>
            <td><input type="button" class="key" value="4" onclick="dis('4','proCalculator')"> </td>
            <td><input type="button" class="key" value="5" onclick="dis('5','proCalculator')"> </td>
            <td><input type="button" class="key" value="6" onclick="dis('6','proCalculator')"> </td>
            <td><input type="button" class="key" value="*" onclick="dis('*','proCalculator')"> </td>
        </tr>
        <tr>
            <td><input type="button" class="key" value="7" onclick="dis('7','proCalculator')"> </td>
            <td><input type="button" class="key" value="8" onclick="dis('8','proCalculator')"> </td>
            <td><input type="button" class="key" value="9" onclick="dis('9','proCalculator')"> </td>
            <td><input type="button" class="key" value="-" onclick="dis('-','proCalculator')"> </td>
        </tr>
        <tr>
            <td><input type="button" class="key" value="0" onclick="dis('0','proCalculator')"> </td>
            <td><input type="button" class="key" value="." onclick="dis('.','proCalculator')"> </td>
            <!-- solve function call function solve to evaluate value -->
            <td><input type="button" class="key" value="=" onclick="solve('proCalculator')"> </td>
            <td><input type="button" class="key" value="+" onclick="dis('+','proCalculator')"> </td>
        </tr>
    </table>
</form>
<hr/>
```

3. **Create a CSS file** and link it to your HTML. Use it to format the **key** buttons and **result** textbox.

```css
.key{
    width: 100%;
    padding: 20px 40px;
    background-color: green;
    color: white;
    font-size: 24px;
    font-weight: bold;
    border: none;
    border-radius: 5px;
}
.result{
    padding: 20px 30px;
    font-size: 24px;
    font-weight: bold;
    border: 2px solid black;
    border-radius: 5px;
}
```

4. **Add the three functions to script.js:**

- **dis()**: Uses the += operator (line 36) to append the pressed key to the textbox.
- **clr()**: Clears the textbox content (line 39).
- **solve()**: Retrieves the textbox content, evaluates the expression, and writes the result back to the textbox (lines 42–44).

```
35  function dis(val,formName) {
36      document.forms[formName]["result"].value += val;
37  }
38  function clr(formName) {
39      document.forms[formName]["result"].value = ""
40  }
41  function solve(formName) {
42      let x= document.forms[formName]["result"].value
43      let y = math.evaluate(x)
44      document.forms[formName]["result"].value = y
45  }
```

Activity 5: Revisiting the Professional Calculator

Activity 4 contained many repeated lines that can be easily generated using JavaScript. In this activity, we will automate the creation of the calculator buttons using JS, which will be added to the HTML **body** tag so it executes and generates the necessary HTML at the end of the form.

- To avoid confusion with Activity 4, the new form is named **proCalculator2**.
- The setup is similar to Activity 4, but starting from the second row, the table rows are generated dynamically using JavaScript:
    - Define a **2D array** called **keys** that stores the value of each button. Each row in the array corresponds to a table row, and each column corresponds to a button within that row.

- Use **two nested loops**: the outer loop iterates over rows, and the inner loop iterates over columns. **keys[i][j]** gives the value of the button at row **i** and column **j**.
- The outer loop prints `<tr>`, the inner loop prints each `<td>` for the buttons, and the row is closed with `</tr>`.
  - If the button value is **"="**, the generated HTML will be:

    ```
    <td><input type="button" class="key" value="=" onclick=
    "solve('proCalculator2')"></td>
    ```

  - For any other button (e.g., **i=1**, **j=2**, value **"6"**), the generated HTML will be:

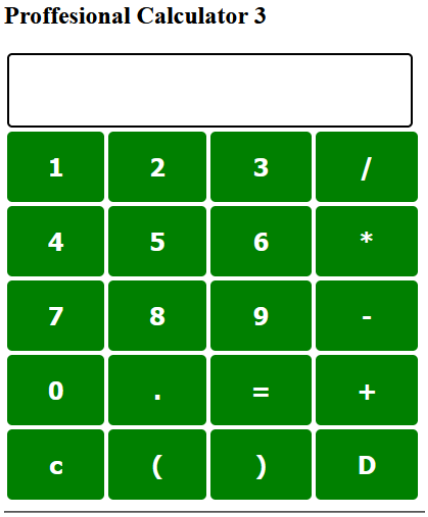
    ```
    <td><input type="button" class="key" value="6" onclick=
    "dis('6','proCalculator2')"></td>
    ```

- Since the form name is passed as the last argument to the JS functions, **no changes are needed in script.js**.

```
<h2>Professional JS Calculator (2) </h2>
<form name="proCalculator2" onsubmit="return false;">
    <table id="calculator">
        <tr>
            <td colspan="3"><input class="result" type="text"  name="result" readonly></td>
            <td><input type="button" class="key" value="c" onclick="clr('proCalculator2')" /> </td>
        </tr>
        <script>
            const keys = [['1', '2', '3', '/'],
                          ['4', '5', '6', '*'],
                          ['7', '8', '9', '-'],
                          ['0', '.', '=', '+']];
            ;
            for (let i = 0; i < 4; i++) {
                document.write("<tr>\n")
                for (let j = 0; j < 4; j++) {
                    if(keys[i][j]=='=') {
                        document.write("<!-- solve function call function solve to evaluate value -->\n")
                        document.write('<td><input type="button" class="key" value="', keys[i][j]);
                        document.write('" onclick="solve(\'proCalculator2\')"> </td>\n');
                    }
                    else{
                        document.write('<td><input type="button" class="key" value="', keys[i][j]);
                        document.write('" onclick="dis(\'', keys[i][j])
                        document.write('\',\'proCalculator2\')"> </td>\n');
                    }
                }
                document.write("</tr>\n")
            }
        </script>
    </table>
```

Exercise 3:

- Modify **Activity 5** to add three new keys: **(** , **)**, and **D**. The updated calculator should appear as shown below.



- **Note**: The **MathJS** library can handle parentheses, so no additional changes are needed for **(** and **)**. For the **D** key, update **script.js** to implement its functionality—**D** should delete the last character in the textbox.