LAB INTRODUCTION

In this Lab you will apply all the techniques you have learned in class so far and build your first "Event Driven Form" website using only HTML, CSS and JavaScript. This will be a single web page that can be rendered in the browser directly (by just simply opening the file in a browser) or by placing the contents of the file in your apache2 web servers documentroot folder and loading the page on localhost. The purpose of this lab is to learn about basic form validation and event handling in JavaScript.

Submission Process

To complete this project you will write and **submit 1 file**: one HTML document with CSS and JavaScript. Submissions will be on canvas under Lab 2.

Late submissions will be penalized by 5% per day (or part thereof), up to 20 days. After 20 days, the assignment will be given a 0%.

Background:

As you have seen in class so far there are many interesting things you can do with JavaScript and event driven programming. In this lab we will focus on a adding event listeners to out DOM objects with the goal of making our website more user interactive and event driven.

Requirements

To complete this project you will write and submit one HTML file containing embedded CSS and JavaScript. The file will be called lab2.html. This file contains a registration form, **which can be styled any way you like subject to the requirements below.** This file must be an ASCII file (i.e. a plain text document with a .html extension). You may create it with any editor but you must ensure that it is a text file. In other words, it should not contain anything except ASCII characters (including HTML tags/CSS rules/JavaScript and content). **You may use an editor such as VS Code, Sublime Text, Atom, etc.**

This lab will be a simple registration form. Every visible form input should be labeled with explanatory text. The form should have the following inputs:

- First and last name (2 separate **text fields**)
    - The user should enter only letters in each field, with the first letter capitalized
- Student ID number (a **text field**)
    - The user should enter a 9 digit number
- Biography (a **textarea**)
    - The user may enter up to 25 words
- Calculator
    - The calculator should have a text field where **result** is displayed
        - If the user types in the text field, as long as the result is numeric, it can be used – (typing in or keying in a operator (+,-,/,*,=) in the text field clears the text fields existing contents but perform the operation)
        - If the user types non-numeric data in the text field, when the user presses any button, the text field should change to "ERROR: press C"
    - The calculator should have 16 buttons (of **button** input type)
        - 0-9 buttons, laid out any way you like
            - These start a new number with the corresponding digit value if no number has been entered yet, or append the corresponding digit value to a number
        - C button, which clears the **result** text field, and current calculated result, leaving the **result** text field blank

        - = button, which performs the last operation pressed and generates a new result in the **result** text field
        - +, -, /, and * buttons, which append the corresponding operation to the expression

The following examples demonstrate exactly how the calculator should behave. Do not deviate from this behavior, nor display any additional characters nor expressions in the result field.

- ➢ **EXAMPLE**
    - User presses 2 button (2 is displayed in result)
    - User presses + button
    - User presses 4 button (4 is displayed in result)
    - User presses = button (6 is displayed in result)
- ➢ **EXAMPLE**
    - **User presses C to clear display**
    - User presses 2 button (2 is displayed in result)
    - User presses 4 button (24 is displayed in result)
    - User presses / button
    - User presses 3 button (3 is displayed in result)
    - User presses = button (8 is displayed in result)
- ➢ **EXAMPLE**
    - **User presses C to clear display**
    - User presses 2 button (2 is displayed in result)
    - User presses 4 button (24 is displayed in result)
    - User presses / button
    - User presses 1 button (1 is displayed in result)
    - User presses 2 button (12 is displayed in result)
    - User presses = button (2 is displayed in result)
- ➢ **EXAMPLE**
    - **User presses C to clear display**
    - User types 24 in result field (24 is displayed in result)
    - User presses / button
    - User presses 1 button (1 is displayed in result)
    - User presses 2 button (12 is displayed in result)
    - User presses = button (2 is displayed in result)
- ➢ **EXAMPLE**
    - **User presses C to clear display**
    - User types 24h in result field (24h is displayed in result)
    - User presses / button (“ERROR: press C” is displayed in result)
        - User must press C to clear result

- A button labeled “Save” which transfers the current calculator **result** to a span labeled “Hours Worked This Week”
    - The span whose label is “Hours Worked This Week” is initialized to empty
        - Should read:

Hours Worked This Week:

    - If the user presses “Save” when the calculator result is either empty or not a number, a message should appear on the page (in differently colored error text) telling the user that this is not a valid number of hours worked. The span labeled “Hours Worked This Week” shall either remain empty (if previously empty) or be cleared/reset (if previously stored a value).
- Submit (a submit input type)

When the user presses the Submit button, the form is checked…

FOR VALID FORMS
If the form is valid (i.e. abides by the rules above, with no rules governing the calculator’s result at the time of submission), then an alert should pop up summarizing all the values the user entered (including current calculator result as well as the saved “Hours Worked This Week” value, if any), and stating that form submission was successful. Note: the user need not save an “Hours Worked This Week” for valid form submission (i.e. this span can remain blank). All other fields must be non-empty, except the calculator result which can be anything. I.e. you should be able to submit the form with an empty “Hours Worked This Week” span but all other fields must contain a value, additionally the result text field in the calculator should submit with whatever values are in there.

If the user did not save an “Hours Worked This Week,” then neither this heading nor any associated value should appear in the summary alert message.

FOR INVALID FORMS
If the form is invalid, no alert should pop up. Rather, for invalid forms, the form page’s background should change color (this can be accomplished with a div that encloses all other content, if you like), and error text should be added to the page (previously invisible/not displayed) that details all the current errors. The error text should stand out and be colored differently than any other elements on the page, much like the error message mentioned above for the “Hours Worked This Week” span. You may change the class attribute of the error text and body/div elements using JavaScript to accomplish this effect, where the classes are defined in embedded CSS.

NO SUBMISSION
The form on the page should not actually submit! In other words, submission should be prevented using the **prevent Default** method on the event object (in the form’s submission handler function). Failure to do this will result in a page-refresh when the submit button is pressed, and the functionality listed in the prior 2 sections (FOR VALID FORMS, FOR INVALID FORMS) will fail to work.

OTHER NOTES
The stylesheet and JavaScript should be embedded in the <head> section of the HTML document.

The JavaScript code must use functions where appropriate (i.e. formSubmitted() for form submission, equalButtonPressed() for the = button of the calculator, etc). These functions may be anonymous functions, as we will discuss in class. (In other words, they can be defined in- place with no function name in the call to addEventListener, or similar.)

The JavaScript code should have occasional explanatory comments where appropriate (on the order of once or twice per function).

Grade Breakdown:

10%: Submission instructions followed to the letter (1 html file submitted, named as stated above, with no contents except plain-text HTML as well as CSS and JavaScript in the head)

10%: JavaScript has proper formatting, proper use of functions, and comments as appropriate 20%: Web page renders properly in Mozilla Firefox 17+ as well as Internet Explorer 9+

30%: Web page contains all required content, behaving according to spec

30%: JavaScript code abides by spec

Cheating: This project (like all projects in this class) is an individual project. You can discuss this project with other students. You can explain what needs to be done and give suggestions on how to do it. You cannot share source code. If two projects are submitted which show significant similarity in source code then both students will receive an F on the assignment. Note a person who gives her/his code to another student also fails the class (as they are facilitating the dishonest actions of another).
Source code that is copied from websites without citation will also count as cheating, and the same consequences apply.