COMP 484L Spring 2021 **Lab 1**

Due Thursday April 15th, 2021 23:55 – via submission to Canvas

Late submissions will be penalized by 10% per day (or part thereof), up to 10 days. After 10 days, the assignment will be given a 0%.

This assignment is to be worked on ***INDIVIDUALLY*** ! No team submissions allowed.

Purpose: You are going to learn basic form validation and event handling in JavaScript.

Requirements: To complete this project you will write and submit **one HTML file** containing embedded CSS (optional) and JavaScript (not optional). The file will be called `lab1.html`. This file contains a registration form, which **can be styled any way you like subject to the requirements below**.

This single submission file `lab1.html` 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).

No zip files will be accepted!
(Therefore: all content must be contained within this single HTML file.)

THIS TIME YOU *MAY* USE A WYSIWYG EDITOR! This editor can be used to generate the HTML markup of your document. *All JavaScript must be subsequently added to this file by hand* (i.e. *manually* editing the HTML file produced by the tool).

This lab will be a simple registration form. Every visible form input should be labeled with explanatory text (like "Username:" or "Enter Message Here…"). The form should have the following inputs:

- Username (a text field)
  - The user should enter **only letters** in this field, or the entry is invalid. You may optionally check this while the user is typing or when the field is blurred, but either way you ***must*** check this at form-submission time.
- Password (a password field, `<input type="password">`)
  - The user should enter a password containing a mix of upper-case and lower-case letters and numbers (at least one of each, or the entry is invalid). Other characters are allowed but optional. You may optionally check this while the user is typing or when the field is blurred, but either way you ***must*** check this at form-submission time.
- Student ID number (a text field)
  - The user should enter a 9 digit number. No letters are allowed or the entry is invalid. You may optionally check this while the user is typing or when the field is blurred, but either way you ***must*** check this at form-submission time.
- Message (a **textarea**)
  - The user may enter up to 25 **words** (note: *not characters*!).
  - The field should be disabled if 25 words are entered, such that no other words can be typed into the textarea. *This can be accomplished a few different ways.*

    - A text label next to the Message field should contain a number which counts down from 25 to 0 as the user enters words.
    - The string split() method can be used with a single space " " delimiter, or with a regular expression to capture all white space. This is up to you. Both are considered correct.
- Submit (a button input type, or a submit input type if you can get this to work **without actually submitting the form**)

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), then an easy-to-see message should be tastefully displayed at the bottom of the screen in its own div. Nothing else should be displayed in this div. While the user is filling out the form, this div should be empty (have no text). This message can say, for example: "**SUCCESS!**"
Again, the form should not actually be submitted!

FOR INVALID FORMS
If the form is invalid, the 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.* (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, also located in the <head> section, like the JavaScript code, see below….)

Everything invalid about the form entries should be displayed as error text.

NO FORM SUBMISSION!!!
The form on the page should *never* actually submit! Failure to prevent submission 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.

It is easy to avoid submission if you **do not** give the form a "real" submit button…
```
<input type="submit">
```

but rather just a generic button labeled with the text "submit," like this…
```
<input type="button" value="submit">
```

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

The JavaScript code should use the Event Listener model mentioned in class! The **only** line of code that is outside of a function should be…

```
window.addEventListener( "load", nameOfMyInitFunction );
```

The JavaScript code must use functions where appropriate (i.e. formSubmitted() or buttonClicked() for form submission). 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).

The JavaScript code must **not** make use of HTML Event attributes, such as onclick, onsubmit, etc. (In the lecture slides, this is referred to as the Inline Model, and it pre-dates the Event Listener Model.)

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 60+

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

30%: JavaScript code abides by spec

Cheating: This project 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 his code to another student also fails the assignment (you 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.