Contents

1 Learning Objectives

JS Objects and functions, Event-driven programming, DOM API, DOM Style Object, dynamic management of HTML elements, JS events, using Bootstrap [1] , HTML Forms, and JS Regular Expression.

2 Special Note

Take the time to read the entire assignment twice before panicking or jumping into writing code. You should write down the steps to be performed (that is, what needs to be done) before you start coding. Subsequent assignments might be based on this one.

Also, remember that this is an individual assignment. You are asked to write your own code. Any similarity detected in any assignment(s) submitted by another student(s) will result in serious repercussions. Again, you have to write your own code. You are not allowed to share code with, or write code for, someone else. If you do so, you will be seriously penalized. If you have any questions, ask me for help/clarifications.

3 Problem Statement

This is an individual assignment. You are required to design and implement the HTML form that is depicted in Figure 1 and validate its fields using JavaScript and regular expression.

You must use Bootstrap [1] to design and implement the web interface and the web form. However, you **must use** the DOM Style Object [2] to change the initial style of the form's elements in case of errors (invalid inputs). Please refer to Section 4.1 to learn more about the technical requirements.

4 Implementation Guide

This section provides you with details about what needs to be done and the steps to be performed.

4.1 Technical Requirements

You must absolutely use follow the following requirements:

- Use HTML/HTML5 to design and implement the interface and the Web form.
- Use the DOM API [3]to access, update, and style the from elements and fields.
- Use JavaScript only, unauthorized use of external code or libraries is strictly forbidden. You are asked and expected to write your own code.
- Write your own custom CSS to style the interface as shown in the provided screenshots.
- **Proper error and exception handling** (see Chapter 04 that is posted under Slides B set on the course webpage).

4.2 Files to Create and Initial Steps

The following are the steps to be done at first:

1. Create the following three files: **1)** an HTML files named *form.html*; **2)** a JavaScript file named *form.js*; and **3)** a CSS file named *style.css*.
2. Write the HTML code to implement the required HTML form.
3. Write the CSS code to design and style the alert boxes.
4. Once your HTML and CSS code is completely implemented, write the required JavaScript code.

NOTE: you need to have both CSS and HTML code completed before writing any JavaScript code.

4.3 The Student Registration Form to Implement

The registration form to be implemented is depicted in Figure 1.

4.3.1 Instructions

- Your application must show an error message using an alert box. The number of errors must be shown as well as a detailed message for each and every invalid field. The background color must be pink. This alert box must be displayed using the DOM API [3]and the style object [2]. See Figure 4. *Hint*: the ***display*** property can be used to show/hide and HTML element.
- The background of the invalid fields must be change to pink. You must use the DOM API [3]and the style object [2]. See Figure 3.
- Hints must be displayed underneath each and every invalid field. See Figure 3.
- The validation process must be triggered only and only upon submitting the web form.
- Upon submitting the form, a function named *validateFrom()* must be called. This function does not receive any parameter. It validates the form's elements one by one using regular expression. You also must validate the length of the fields. Example: password must be 6 to 8 characters long.
- If all the fields are valid, an alert box must be shown. The background color must be light green. See Figure 2.

4.3.2 Fields Validation

You are **not allowed** to use the *required* nor the *pattern* HTML attribute that were introduced in HTML5. You must write the necessary regular expression patterns according to the provided format to validate the following fields:

1. **Student ID:** as provided, must start with three letters followed by a dot, followed by three digits followed by a dot then followed by 1 to four digits.
2. **Firstname:** Only letters. Maximum 15 letters.

3. **Lastname:** Only letters. Maximum 15 letters.

4. **Username:** minimum 6 and maximum 10 letters long. Only letters.

5. **Password:** minimum 6 and maximum 8 characters. Must contain digits, letters and at least two special characters such as @, #,$, , etc.

6. **Gender:** must be male or female (other than the default value which is unspecified).

7. **Phone number:** must be one opening parenthesis followed by three digits followed by a closing parenthesis followed by three digits followed by a dash then followed by four digits (e.g., (514)-999-9999).

8. **Email address:** must be a valid email address.

9. **Date of birth:** format: mm/dd/yyyy (two digits, /, two digits, /, 4 digits)

10. **Postal code:** must be a valid Canadian postal code (e.g., A1A1A1 or A1A 1A1).

11. **Street:** not empty (letters and digits).

12. **City:** not empty, only letters.

13. **Province:** not empty, only letters.

You can use JavaScript to check whether if a field is empty or not.

4.4 Functions to Implement

You must absolutely implement the following functions:

1. ***validateForm()***: this function does not receive any parameters and must be called upon submitting the form. *Hint*: you need to use the proper JS event that triggers the call to this function. This function validates each and every field defined in the form using JS and regex according to the specified data format (See Section 4.3.2).

2. ***changeLablelColor(fieldID)***: this function receives a field label's id as parameter and changes its text color to red.

3. ***resetLabelColor(fieldID)***: this function receives a field label's id as parameter and resets its text color to black.

4. ***showErrors(errMessages)***: this function receives an array of strings as parameter containing the error messages to be displayed. It also shows the error alert box. Note that the alert box must be initially hidden.

5. ***showSuccess()***: this function does not receive any parameter. It shows the success alert box. *Hint*: Showing an alert box means changing the value of its display property (see [2] for more details).

5 Evaluation criteria and Instructions

Please carefully read the following points. You will be judged based on:

- You should create the following files: form.html, form.js, and style.css
- Failing to respect the provided instructions will result in a major grade deduction. You should also use the name of files, objects, and functions exactly as provided.
- You must document your code.
- Completeness, correctness, and functionality of the implementation.
- Compliance of the implementation with the stated problem as well as simplicity and appropriateness of your implementation.
- Documentation.
- Programming style, code readability, naming conventions, and clarity.
- User-friendliness of the web interface.

6 What to Submit

- Compress all your files (.html, .css, .js) and upload them to LÉA.

7 Screenshots

References

[1] Bootstrap Project, "Bootstrap CSS Framework." http://getbootstrap.com//, 3434. [Online; accessed 1-Oct-2016].

[2] W3 Schools, "DOM style object." http://www.w3schools.com/jsref/dom_obj_style.asp, 2016. [Online; accessed 18-Oct-2016].

[3] W3 Schools, "DOM API." http://www.w3schools.com/js/js_htmldom_document.asp, 2016. [Online; accessed 18-Oct-2016].

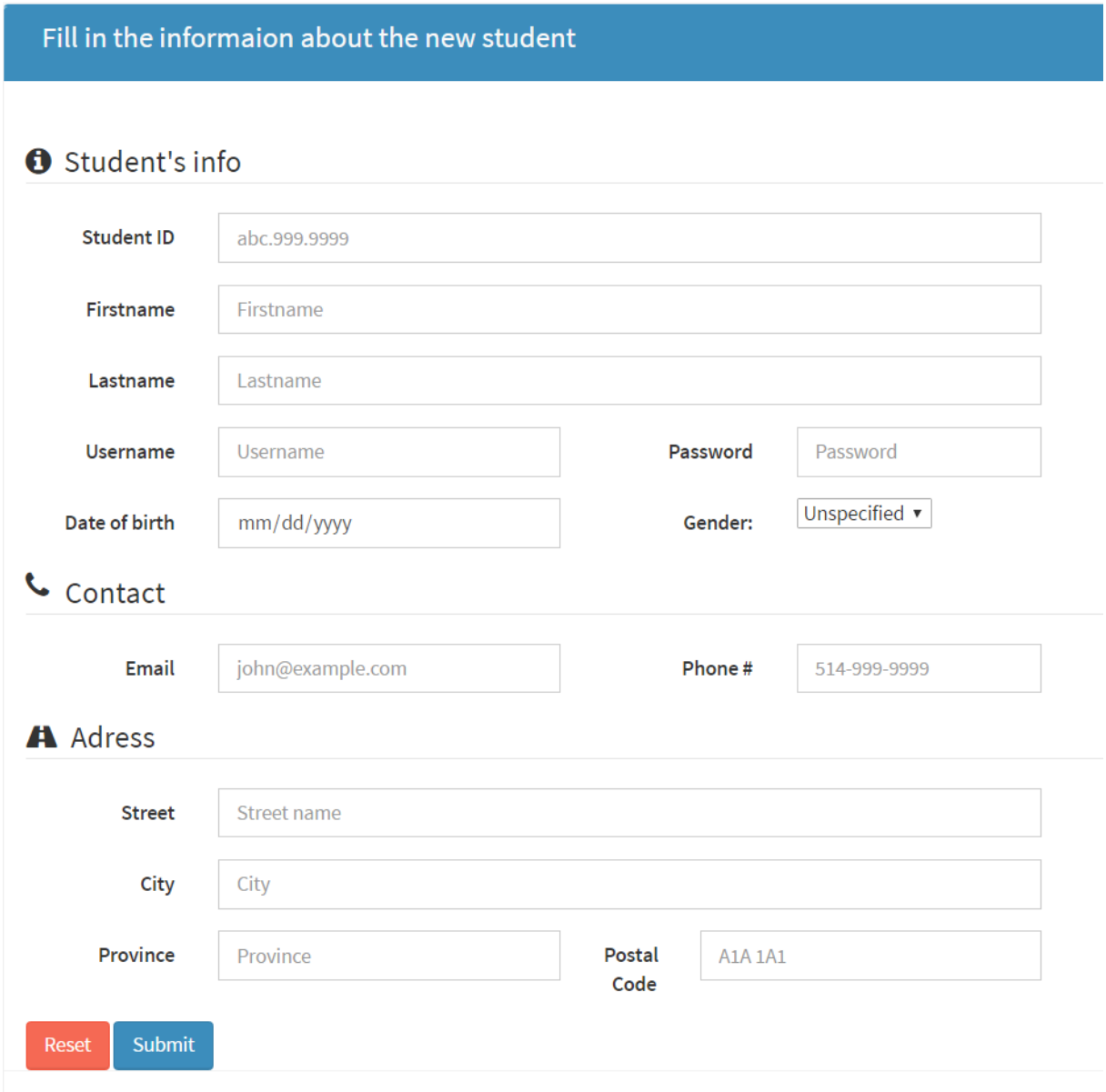

Figure 1: Student registration form

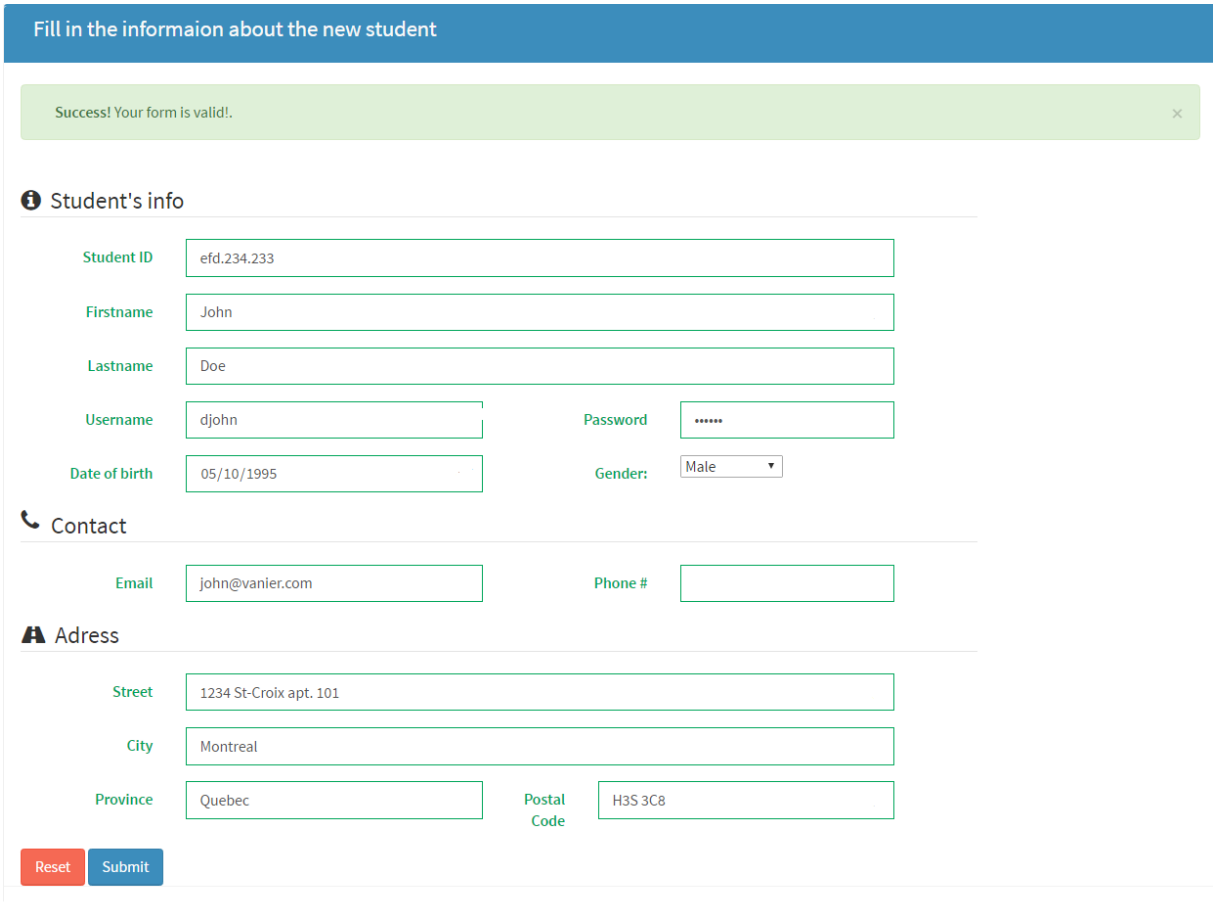

Figure 2: Form after successful validation

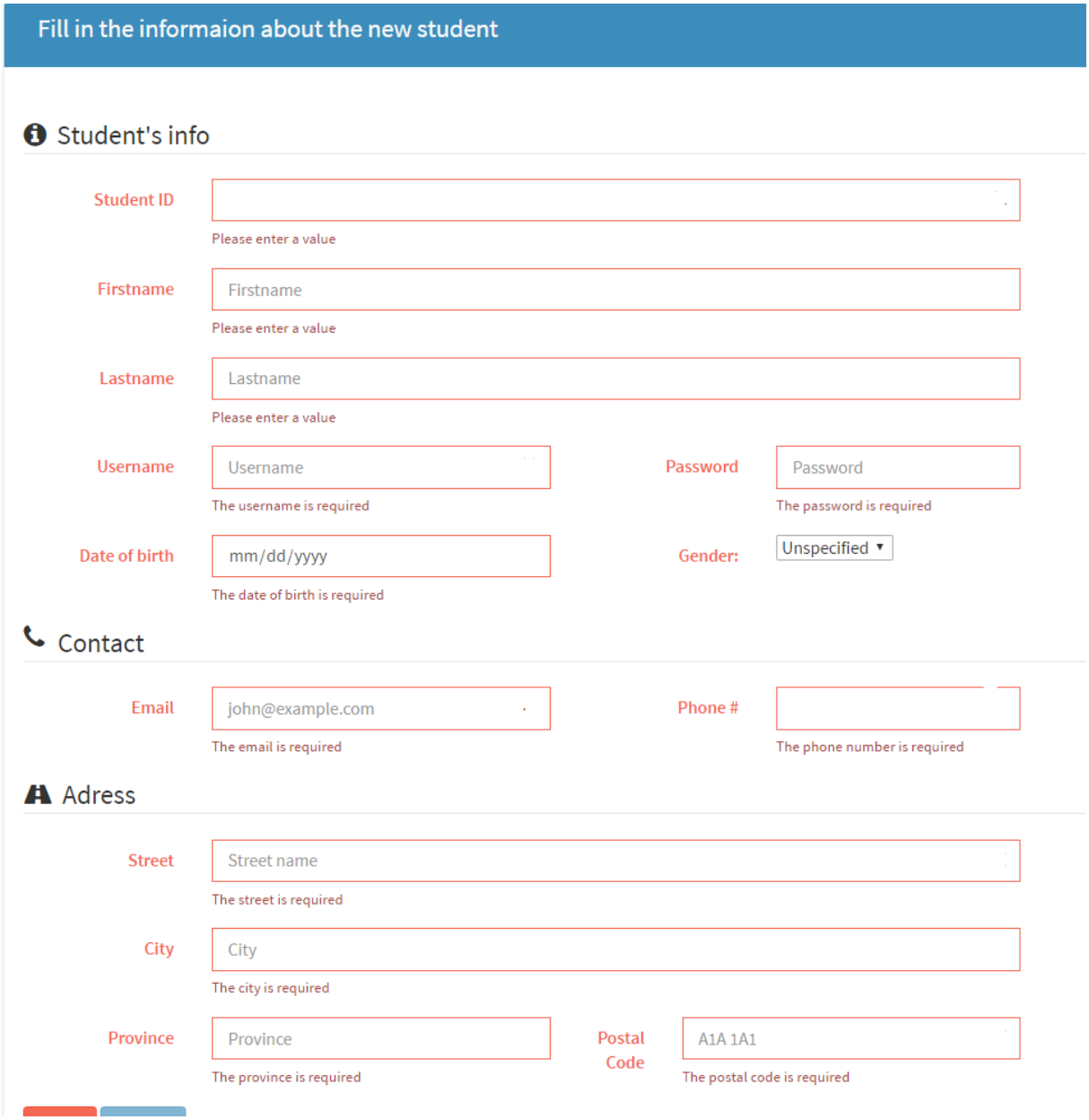

Figure 3: Form with errors

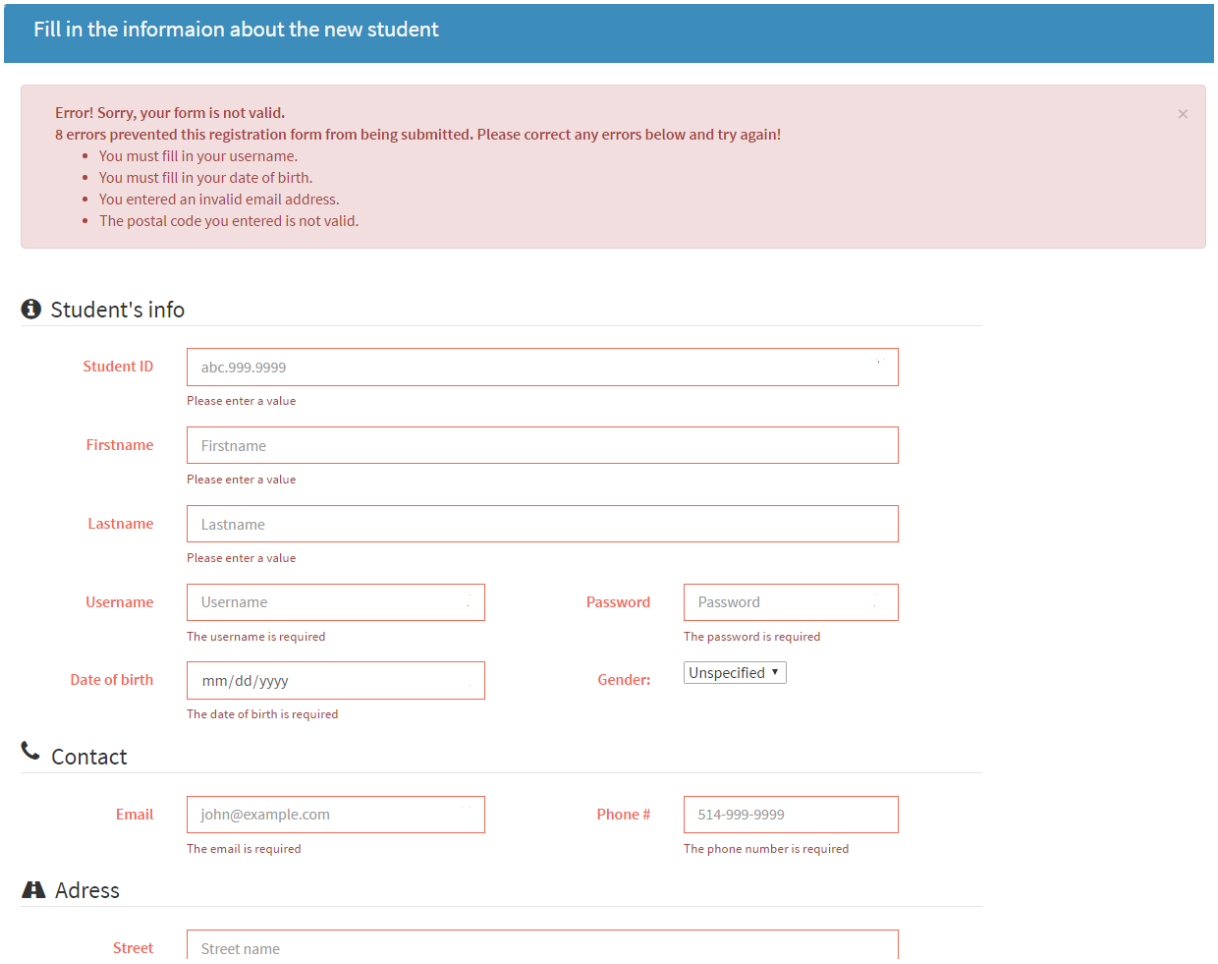

Figure 4: The error messages to be displayed