CS108 - Software Systems Lab

Lab 4 - JavaScript

Student: Aditya Sanapala, 23b0912@iitb.ac.in
Lecturer: Kameswari Chebrolu, chebrolu@cse.iitb.ac.in

Problem 1: JavaScript as a Programming Language

In this activity, you need to find the largest prime number which is less than or equal to the number which is received as input.
The HTML code is already provided to you in the file `basic.html`. You need to write JavaScript code in the same file, and submit. (If there is no prime number less than or equal to the given number, just print 0) (Also, your final result string should be stored in the paragraph with `id="result"`, see the HTML Code for your reference!)

The website before giving an input looks as in `1a.png`. Suppose we give 10 as the value of n, then the website will look like as in `1b.png`. (after clicking the button)

Please make sure that the format of the result is as shown in the images.

Problem 2: Ram Mandir Pran Pratistha

Let's talk about 22 January 2024. Suppose there were n priests present in Ayodhya on the occasion of *Ram Mandir Pran Pratishtha*. Everyone has brought a gift for this auspicious event (A single person has brought a single gift). The idea is that everybody will get a gift which has been brought by someone else. You need to find out the number of ways in which this is possible. (Looks similar to JEE stuff, doesn't it?)

Write your code in the `script.html` file provided (JavaScript code should also be written in that file only). Also, your final result string should be stored in the paragraph with `id="result"`, see the HTML Code for your reference.

A simple look of the website is shown in `2a.png`. Suppose, if you type 3 in the space provided and click on the `Calculate` button, then the look will be as shown as in `2b.png`.

In the case of 3 priests (A, B, C), the possible matchings are (A gives B, B gives C, C gives A) and (A gives C, C gives B, B gives A). So, the $result = 2$ message will be printed.

Problem 3: User Registration Validation

You have to create a user registration form with an input validation using JavaScript. Handle errors and provide appropriate feedback to users.

Begin by completing the `index.html` file with a user registration form. Include the following fields with appropriate labels and input boxes/button with `id` strictly the same as follows:

(a) Full Name, `id="fullName"`

(b) Email Address, `id="email"`

(c) Password, `id="password"`

(d) Confirm Password, `id="confirmPassword"`

(e) Submit Button, `id="submit"`

The submit button on clicking should call the function `validateForm()` from the `script.js` file.

Note: If the same ids are not followed, you will face problem(s) in evaluation.

Use the appropriate `type` attribute for each input box. The full name input box will have the type `text`, the email address input box will have the type `email`, and the password and confirm password input boxes will have the type `password`. The submit button will have the type `button`. Follow this link to look at possible input field `type` attributes.

Note: An empty `<div>` container with `id="feedback"` is provided in `index.html`. Do not modify it. Use that container in your JavaScript file to give back the error messages/feedback.

In `style.css`, just make sure that every component is centered and set the background color of the body to azure, as shown in `3a.png`.

Complete the `script.js` file. Most of the instructions are written as comments in that file. You must complete the `validateName, validateEmail, validatePassword, ConfirmPassword`, and `validateForm` functions. These functions don't return anything.

Whenever you have to give feedback about a successful entry or an error message, refer to the instructions in the `How to give your feedback/error message to the HTML page from a JavaScript file?`. Now, read the content below to learn more about the functions in the `script.js` file.

When `validateForm()` is called, the `try` block has code that runs in the following order:

(a) Checks if all fields are non-empty/filled or not. If not, throw the error message `Error: All fields are required.`

(b) Calls the `validateName()` function

(c) Calls the `validateEmail()` function

(d) Calls the `validatePassword()` function

(e) Calls the `ConfirmPassword()` function

(f) Gives the feedback `Registration successful!` in green color to the HTML page

The `catch` block contains the code that gives feedback to the HTML page about the error message it catches. The error message, if caught, will either be `Error: All fields` `are required.` or those thrown by any one of the `validateName()`, `validateEmail()`, `validatePassword()` or `ConfirmPassword()` functions.

Note: The `try` block will reach step (f) successfully only if none of the steps (a) - (e) throw an error message.

About the functions:

(a) The `validateName()` function checks if the name is entered or not. If not, throw the error `Error: Full name is required.`

(b) The `validateEmail()` function checks if the email entered is valid or not. An email is valid iff:

   (i) it contains exactly 1 @

   (ii) there should be atleast one character to the left of @. If yes, those characters should only be a-z or 0-9

   (iii) to the right of @ there should be exactly one dot

   (iv) between the *symbol*64 and the dot, there should be at least one character. If yes, those characters should be only from a-z

   (v) to the right of the dot, there should be exactly 3 lowercase English characters

   This check is conveniently possible by regex. Use the `test()` function of JavaScript to make it happen. Follow this link to learn more about it. If the email is invalid, throw the error message `Error: Invalid Email Address.`

(c) The `validatePassword()` function checks if the password entered is at least 8 characters long. If not, throw the error message `Error: Password must be at least 8 characters.`

(d) The `ConfirmPassword()` function checks if the re-entered password matches or not. If not, throw the error message `Error: Passwords do not match.`

How to give your feedback/error message to the HTML page from a JavaScript file?

Using the `innerHTML` element, we can use the `<div> "feedback"` container of HTML to give the error messages/responses back. Follow this link to learn more about the `innerHTML` element. When all fields are validated successfully, then on clicking the `Submit` button, the message `Registration successful!` should be printed in green color. If any one of the fields is invalidated, then on clicking `Submit`, an error message should appear in red. We can use the "span" tag along with the `innerHTML` element to adjust the color of the feedback. Follow this link to learn more about the `span` tag.

An example of a successful entry is showed in `3b.png`. An example for an error message is shown in `3c.png`.

Problem 4: Let’s Manipulate

In this activity we will use Javascript to manipulate HTML files provided.

There is one `sample.html` file provided to you, which has its script linked to `dom.js`. You need to write your code in `dom.js`, to manipulate the HTML webpage as follows:

(a) For all images (with `<img>` tag), change the source of the image to be `timepass.png`

(b) Delete all the `<h1>` heading tags. (Remember, no other headings should get deleted.)

(c) For all paragraphs (with `<p>` tags), change the content of the paragraphs to be `Enough of JavaScript, let’s stop.`

(d) Change all the `<h2>` content to make it uppercase. So, suppose you have a `<h2>` heading saying `Don’t Stop`, then it should be converted to `DON’T STOP`

(e) For all `<div>` containers with element `id="div1"`, add a heading (`<h3>`) with no text

All these changes will happen when you click the `Change` button. The `Change` button in your HTML file will call a function in your JavaScript code. So, you need to edit that function in `dom.js`.

Once you click the `Evaluate` button, the results will be shown after a few seconds, so please wait.