Grade 11 Assignment #7 Java Script

Password Validator

We are going to write a program to make sure that a password is complex so that it cannot be easily guessed. Our password validator will check multiple aspects of a potential password and will report back to the user if their password meets a set of rules.

The rules for our password validator are:

- Has at least one uppercase letter
- Has at least one lowercase letter
- Is at least 8 characters long
- Has at least one special character

We can utilize functions to check for each of these requirements, then create a function to validate a user's password.

Tasks:

There are 20 Tasks to complete for this assignment:

1. Using NotePad++, create the code needed for the program called Password Validator.

   This program will be split into 5 functions:

   - A function that verifies the password has an uppercase letter.
   - A function that verifies the password has a lowercase letter.
   - A function that verifies the password is at least 8 characters

- A function that verifies the password has a special character.
- A function that reports to the user if their password is complex enough, and if not, tells them what they are missing.

2. Let's begin with the last function so that you can see your results as you progress through this project.

   Begin by declaring a function named isPasswordValid that takes one parameter named input.

3. The isPasswordValid function will run each function we make in the upcoming steps.

   Now, let's build the first condition:

   - A function that verifies the password has an uppercase letter.

   Declare a function named hasUppercase that takes one parameter named input.

4. The purpose of hasUppercase is to determine whether a password has an uppercase letter or not. If so, it should return true.

   Inside the hasUppercase function, you'll need to check every letter of the password to see if it is an uppercase letter. In a future step, you'll pass in the password to this function as the input parameter.

To accomplish this rule, write a for loop that iterates through each letter of the input parameter inside the hasUppercase function.

5. Inside the for loop, write an if statement that checks if each input letter is equal to the uppercase version of itself. You can utilize the JavaScript function toUpperCase to transform a letter to its uppercase version.

   If the input does have an uppercase letter, return true. If not, don't return anything.

6. The hasUppercase function can now check if an uppercase letter is present. Now you need to pass it a password.

   In the isPasswordValid function, write an if statement. Inside the if statement's condition (inside its parentheses), call the hasUppercase function and pass it input as the parameter.

   If hasUppercase returns true, then use console.log to print that the password is valid.

7. You passed in input to the hasUppercase function. input here is the parameter to the isPasswordValid function. Therefore you need to call the isPasswordValid with a password.

   On the last line of the program, call the isPasswordValid function, and pass in a string as a password.

8. You should see that the password is valid log to the console. Now try a password without a capital letter. Right now, nothing happens.

   Let's build a condition that logs to the console that the password needs a capital letter.

   In the isPasswordValid function, write another if statement. In the if statement's condition (inside of its parentheses), call the hasUppercase function, but this time check if calling it is false with the ! operator.

   If the function call is false, then log to the console that the password needs a capital letter.

   The ! operator will turn false into true. In this case, if the password does not have an uppercase letter, the function will not return anything. That is, the function will return undefined. undefined in JavaScript is *falsely*, or in other words, undefined evaluates to false.

   The other values in JavaScript that evaluate to false are: 0, -0, undefined, false, null, '', NaN.

   Conversely, all other values returned will evaluate to true, so they are referred to as *truthy*.

   If the hasUppercase function returns undefined, that will evaluate to false, therefore we can make that condition true with the ! operator, which enables you to perform logic when a condition evaluates to false.

9. Quick Check. Make sure everything is working before moving on.

   Great work so far! At this point you have laid the groundwork for the whole program.

Right now, if you try a password with no capital letters, you should see a message that says you need one, and if you try a password with capital letters, you should see a message that your password is valid.

Now you can add the rest of the conditions, following the same flow:

1. Write a function that fulfills one of the rules.
2. Add in if condition in the isPasswordValid function that makes sure it returns true in order to tell the user their password is valid.
3. Add another if statement in the isPasswordValid function that tells the user what they are missing if the function returns undefined, or false.

10. The next rule to build is:

- A function that verifies the password has a lowercase letter.

Declare a function named hasLowercase that has one parameter named input. This function should be similar to the hasUppercase function, but this time utilize the JavaScript function toLowerCase to verify the input has a lowercase letter.

Then, write the two if conditions in the isPasswordValid function to log the correct message according to whether the hasLowercase function returns true or not.

11. Repeat the same process as Step 9, but this time with the next rule:

- A function that verifies the password is at least 8 characters

Declare a function named isLongEnough that has one parameter named input, and return true if the password is greater or equal to 8 characters. You can utilize the .length property to check for this.

Then, write two if conditions in the isPasswordValid function to log the results of the function.

12. Now for the last rule:

    - A function that verifies the password has a special character.

    Declare a function named hasSpecialCharacter that has one parameter named input.

    In order to check if the input has a special character, you'll need to create a list of special characters.

    Declare a variable named specialCharacters and set its value to an array. Each item in the array should be one special character.

13. Inside the hasSpecialCharacter function, you'll need to see if any letter in the input matches any letter in the specialCharacters array.

    To do this, write a for loop that loops through each letter in the input variable.

    Inside the for loop's block, write another for loop that loops over each letter in the specialCharacters array.

14. Now, you'll need to check if any letter in the input equals any character in the specialCharacters array.

    Write an if statement in the inner for loop's block that checks if input[i] is equal to specialCharacters[j]. If so, return true.

15. In the isPasswordValid function, add the hasSpecialCharacter condition in two if statements, just like the other functions.

    One if statement should check if the hasSpecialCharacter returns a false value. If so, log to the console that the user is missing a special character.

    Then, if hasSpecialCharacter returns true, add hasSpecialCharacter to the if statement that logs the password is valid.

16. Quick Check. Nice work, but before you go on to the next step, try out multiple passwords and check to make sure the output matches what you expect.

17. However, you are not finished yet. You need to make your program user friendly by printing-out user friendly messages to the screen welcoming the user and asking them to try out the program while informing them the point of the program and what the output on the screen they see means.

18. It would be helpful to describe to other developers what this small Java program does. Write some comments that describes what this program does.

    a. Use multi-line comments to (/* comment in the middle of */):
        i. Write one at the top of the code (before the main part of the code) that gives a quick intro/description the assignment.
        ii. Below that comment, write one that summarizes the program.
    b. Use a single line comment to (//then comment):
        i. Create 5 comments anywhere you deem necessary or important in the code. Remember the comment is to highlight or explain what is going on or what is being done in a particular way or used and why.
        ii. Ideas for your comments above would be to explain: the different functions, for loops, if statements, and variable declarations or the printed out statements to the screen.
        iii. Identify the last line of code in every function or statement (anything that ends with a curly bracket { }) by writing “End of BLAH-BLAH”.
        iv. Identify the end of the program.

19. Once your program is complete, make sure to test it using the console (do not submit a program that does not work).

20. Lastly, upload (drag and drop) your assignment to the portal. Look for your name under the Assignment 7 webpage.