Exercise 2 - Form Validation Exercise

Objective:

Create a form validation exercise to validate the nickname, email and password according to specific rules. The validation should show errors if the rules are not met and highlight the problematic input fields.

Resources:

- Weekend assignment HTML and CSS code.

Instructions:

1. **Understand the Existing HTML and CSS Code:**
   - Examine the given HTML structure, noting the IDs for the nickname, email and password inputs, as well as the error-message divs.
   - Understand how the CSS classes are applied to style the form and display errors.

2. **Initialize Form Elements:**
   - Retrieve the form and input elements using their corresponding IDs.

3. **Create the Show Error Function:**
   1. **Identify the Error Message Container:**
      - Determine the HTML element where the error message will be displayed. This is usually a div element associated with the input field, and it can be identified by concatenating the ID of the input field with 'Error', like this: `document.getElementById(input.id + 'Error')`.
   2. **Set the Error Message:**
      - Change the text content of the identified error message container to the specific error message you want to display. This message will describe what the error is or what the user needs to correct.

3. **Add an Error Class to the Input Field:**

   - Apply an `error` class to the input field that has been styled to indicate an error, such as changing the border color to red. This visually emphasizes the input fields that need to be corrected.

4. **Create a Validation Function for Form Submission:**

   - Add an event listener for the form's `submit` event that runs the validation function.
   - Use `e.preventDefault();` to prevent the default form submission, allowing custom validation.
   - Initialize a variable `isValid` to track the overall validation status.

     1. **Validate the nickname:**

        - Check if the nickname is at least 3 characters long.
        - If not, call the `showError` function with the `nicknameInput` and the error message: `"nickname must be at least 3 characters long."`, and set `isValid` to `false`.

     2. **Validate the Email:**

        - Use the provided regular expression to validate the email format.
        - If the email doesn't match the pattern, call the `showError` function with the `emailInput` and the error message: `"Please enter a valid email."`, and set `isValid` to `false`.

     3. **Validate the Password:**

        - Use the provided regular expression to check the password's complexity.
        - If the password doesn't meet the requirements, call the `showError` function with the `passwordInput` and the error message:
          ```
          "Password must be at least 8 characters long, containing lowercase, uppercase letters, numbers, and a special character."
          ```
          , and set `isValid` to `false`.

     4. **Show Success Alert:**

        - If `isValid` remains `true` after all validations, show a success alert using the provided example code.

1. **Remove Error Indicators on Input Change:**

    1. **Create an Array of Input Fields:**

        - Gather all the input fields you want to work with, including the nickname, email and password fields.

    2. **Iterate Over the Input Fields:**

        - Use a loop to go through each input field one by one. This allows you to apply the same logic to each field without repeating code.

    3. **Add an Input Event Listener:**

        - For each input field in the loop, attach an event listener that responds to the "input" event. This event triggers when the user changes the content of the field.

    4. **Remove the Error Class:**

        - Inside the event listener, remove the class that indicates an error from the input field. This class may be responsible for highlighting the field in red or displaying an error icon, and removing it will revert the field to its normal appearance.

    5. **Clear the Corresponding Error Message:**

        - Also inside the event listener, find the corresponding error message element using the ID of the input field (`document.getElementById(input.id + 'Error')`) and clear its content. This will remove any error message that was previously displayed.

Challenges and Tips:

- Experiment with different validation rules, explore the given code, and understand how it functions.
- Change the HTML and the CSS if needed.
- Utilize browser developer tools to examine HTML, CSS, and JavaScript during validation.
- Investigate `javascripttutorial.net` Javascript Form Validation for alternative approaches.
- Consider enhancing the exercise with additional validation or dynamic user feedback.

Evaluation Criteria:

- **Functionality:** Validation must work correctly for all fields, displaying proper error messages and styling.
- **Code Quality:** Code should be well-organized, efficient, following best practices, conventions, and design patterns.