DOCUMENTATION FOR VALIDATION OF A REGISTRATION FORM

MARIA BOBY | REG NO 2447130

This is a registration form with java script validations.

1. **Event Listener for DOMContentLoaded:**

```
document.addEventListener('DOMContentLoaded', function () {
    ...
});
```

Ensures that the script runs only after the HTML document has been completely loaded and parsed. This prevents errors that might occur if the script tries to access elements that are not yet available in the DOM.

2. **Retrieving Form and Input Elements:**

```
const form = document.getElementById('registrationForm');
const nameInput = document.getElementById('name');
const emailInput = document.getElementById('email');
const passwordInput = document.getElementById('password');
const confirmPasswordInput =
document.getElementById('confirmPassword');
const dobInput = document.getElementById('dob');
const submitBtn = document.getElementById('submitBtn');
```

Retrieves references to the form and its input elements by their IDs. This allows the script to easily manipulate and validate these elements.

3. **Validation Functions:**

3.1. `validateName()`:

```
function validateName() {
    const nameValue = nameInput.value.trim();
    const isValid = /^[A-Za-z\s]{3,}$/.test(nameValue);
    displayValidation(nameInput, isValid, 'nameError');
```

```
return isValid }
```

Uses a regular expression to ensure that the name contains at least 3 alphabetic characters and spaces. This ensures the input meets basic naming conventions. It helps maintain data quality and consistency.

3.2. validateEmail():

```
function validateEmail() {

   const emailValue = emailInput.value.trim();

   const isValid = /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(emailValue);

   displayValidation(emailInput, isValid, 'emailError');

   return isValid;

}
```

Uses a regular expression to validate the email format. Ensures that users provide a valid email address, which is crucial for communication and verification.

3.3. validatePassword():

```
function validatePassword() {

   const passwordValue = passwordInput.value.trim();

   const isValid = /^(?=.*[A-Za-z])(?=.*\d)[A-Za-
z\d]{8,}$/.test(passwordValue);

   displayValidation(passwordInput, isValid, 'passwordError');

   validateConfirmPassword();

   return isValid;

}
```

Ensures the password is at least 8 characters long and contains both letters and numbers. This improves security by enforcing a basic level of password strength. Calls validateConfirmPassword() to ensure that the confirmation password matches.

3.4. validateConfirmPassword():

```
function validateConfirmPassword() {
```

```
    const passwordValue = passwordInput.value.trim();

    const confirmPasswordValue = confirmPasswordInput.value.trim();

    const isValid = passwordValue === confirmPasswordValue;

    displayValidation(confirmPasswordInput, isValid,
'confirmPasswordError');

    return isValid;

}
```

Confirms that the password and its confirmation match. This prevents users from accidentally entering different passwords and ensures consistency.

3.5. validateDob():

```
function validateDob() {

    const dobValue = dobInput.value.trim();

    const dobDate = new Date(dobValue);

    const age = calculateAge(dobDate);

    const isValid = age >= 18;

    displayValidation(dobInput, isValid, 'dobError');

    return isValid;

}
```

Validates that the user is at least 18 years old. This is often a legal requirement for registration on many platforms.

3.6. calculateAge(dob):

```
function calculateAge(dob) {

    const diff = Date.now() - dob.getTime();

    const ageDate = new Date(diff);

    return Math.abs(ageDate.getUTCFullYear() - 1970);

}
```

Calculates age based on the date of birth. This is used to ensure that the user meets the minimum age requirement.

4. **Display Validation Function:**

```
function displayValidation(input, isValid, errorElementId) {
    const errorElement = document.getElementById(errorElementId);
    if (isValid) {
        input.classList.remove('invalid');
        input.classList.add('valid');
        errorElement.classList.add('hidden');
    } else {
        input.classList.remove('valid');
        input.classList.add('invalid');
        errorElement.classList.remove('hidden');
    }
}
```

Updates the appearance of input fields and displays or hides error messages based on the validation result. Provides immediate feedback to users about the validity of their inputs, improving the user experience.

Real-time Validation:

```
nameInput.addEventListener('input', validateName);
emailInput.addEventListener('input', validateEmail);
passwordInput.addEventListener('input', validatePassword);
confirmPasswordInput.addEventListener('input', validateConfirmPassword);
dobInput.addEventListener('input', validateDob);
```

Adds event listeners to input fields for real-time validation. This ensures users receive immediate feedback and can correct errors as they type.

Final Form Validation on Submit:

```
form.addEventListener('submit', function (event) {
    event.preventDefault();
    const isNameValid = validateName();
    const isEmailValid = validateEmail();
    const isPasswordValid = validatePassword();
    const isConfirmPasswordValid = validateConfirmPassword();
    const isDobValid = validateDob();
    if (isNameValid && isEmailValid && isPasswordValid &&
isConfirmPasswordValid && isDobValid) {
        alert('Registration Successful');
        form.submit(); // Uncomment this line to submit the form
    } else {
        alert('Please fix the errors before submitting the form.');
    }
});
```

Handles the form submission:

- Prevents default submission to validate the form first.
- Calls all validation functions to ensure all fields are correct.
- Displays a success message if the form is valid and submits the form (commented out for testing purposes).
- Alerts the user to fix errors if validation fails.