**[EXERCISE] Creating an Email Form Component in React with Validation**

**Task Description:**

Your task is to create an email form component in React. The form should allow the user to enter an email address and then validate its correctness. If the email is invalid (e.g., it does not contain the @ symbol), an appropriate error message should be displayed.

**Starting Points:**

Initially, you have a prepared skeleton of the form with only the input field and its surrounding div element. Your task is to expand this component with the necessary functionality.

**Starter Code:**

Here is the initial code for the EmailForm component. Start from here:

1. import React from 'react';
3. function EmailForm() {
4. return (
5. <div>
6. <form>
7. <input
8. type="email"
9. placeholder="Enter email"
10. />
11. </form>
12. </div>
13. );
14. }
16. export default EmailForm;

**Tasks to Complete:**

**1. Adding State Management with Hooks**

* **a.** Import the useState Hook from React.
* **b.** Add an email state to the component to store the value entered by the user.
* **c.** Add an error state to store error messages.

**2. Handling Input Field Changes**

* **a.** Create a validateEmail function that updates the email state based on the value entered by the user.
* **b.** In the same function, check if the entered email contains the @ symbol. If not, set the error state to an appropriate message, e.g., "Invalid email address." Otherwise, clear the error state.
* **c.** Assign the validateEmail function to the onChange event of the input field.

**3. Displaying Error Messages**

* **a.** Below the input field, add an element that will display the error message if the error state is not empty.
* **b.** Style this error message with a red color to make it stand out on the page.

**4. Adding a Submit Button and Handling Click Events**

* **a.** Add a Submit button below the input field.
* **b.** Create a function to handle the click event of the Submit button. This function should:
  + Check if the current email is valid (i.e., there are no errors).
  + If the email is valid, display an alert with the message "Email has been sent!"
  + If the email contains errors, prevent the form from submitting and display the appropriate error messages.

**5. Extending Validation**

* **a.** Add an additional check to ensure that the email address also contains a dot (.) after the @ symbol. If not, set an appropriate error message, e.g., "Email must contain a dot after @."
* **b.** Update the validateEmail function to include this new validation criterion.

**6. Styling the Component**

* **a.** Add basic styling to the form to make it more aesthetically pleasing (e.g., adding margins, changing the button color).
* **b.** You can use CSS or inline styles based on your preference.

**7. (Optional) Refactoring the Code**

* **a.** Instead of using two separate Hooks (email and error), try using a single Hook that will store an object containing both values.
* **b.** Transform the validateEmail function to work with the new state structure.

**Tips:**

* **Importing Hooks:** Remember to import all necessary Hooks from React at the beginning of the file.
  1. javascript
  2. Skopiuj kodimport React, { useState } from 'react';
* **State Management:** Use useState to manage the values entered by the user and the error messages.
* **Event Handling:** Ensure that the functions handling events (onChange, onSubmit) are correctly assigned to the respective elements.
* **Validation:** Strive to write clear and efficient validation conditions that can be easily extended in the future.
* **Styling:** Ensure that the form is readable and user-friendly.
* **Testing:** Regularly test your component to ensure that all functionalities work correctly after each change.

**Expected Outcome:**

After completing all tasks, you should have a functional email form with basic validation. The form should:

* Allow the user to enter an email address.
* Check if the email contains the @ symbol and a dot (.) after the @.
* Display appropriate error messages if the email is invalid.
* Allow the form to be submitted only when the email is valid.
* Be aesthetically designed thanks to the added styling.

**Sample Solution (Scroll Down):**

Ensure that you understand each step and try to create solutions independently before referring to the provided code.

1. import React, { useState } from 'react';
3. function EmailForm() {
4. const [email, setEmail] = useState('');
5. const [error, setError] = useState('');
7. const validateEmail = (e) => {
8. const value = e.target.value;
9. setEmail(value);
11. // Basic validation
12. if (!value.includes('@')) {
13. setError('Invalid email address.');
14. } else if (!value.split('@')[1].includes('.')) {
15. setError('Email must contain a dot after @.');
16. } else {
17. setError('');
18. }
19. };
21. const handleSubmit = (e) => {
22. e.preventDefault();
23. if (error === '' && email !== '') {
24. alert('Email has been sent!');
25. // You can add further logic here, e.g., sending data to a server
26. } else {
27. setError('Please enter a valid email address before submitting.');
28. }
29. };
31. return (
32. <div style={styles.container}>
33. <form onSubmit={handleSubmit} style={styles.form}>
34. <input
35. type="email"
36. value={email}
37. onChange={validateEmail}
38. placeholder="Enter email"
39. style={styles.input}
40. />
41. {error && <p style={styles.error}>{error}</p>}
42. <button type="submit" style={styles.button}>
43. Submit
44. </button>
45. </form>
46. </div>
47. );
48. }
50. const styles = {
51. container: {
52. margin: '50px',
53. fontFamily: 'Arial, sans-serif',
54. },
55. form: {
56. display: 'flex',
57. flexDirection: 'column',
58. maxWidth: '300px',
59. },
60. input: {
61. padding: '10px',
62. marginBottom: '10px',
63. fontSize: '16px',
64. },
65. error: {
66. color: 'red',
67. marginBottom: '10px',
68. },
69. button: {
70. padding: '10px',
71. backgroundColor: '#4CAF50',
72. color: 'white',
73. border: 'none',
74. fontSize: '16px',
75. cursor: 'pointer',
76. },
77. };
79. export default EmailForm;

**Option 7: Refactoring the Code**

1. javascript
2. Skopiuj kodimport React, { useState } from 'react';
4. function EmailForm() {
5. const [state, setState] = useState({
6. email: '',
7. error: '',
8. });
10. const validateEmail = (e) => {
11. const value = e.target.value;
12. let error = '';
14. if (!value.includes('@')) {
15. error = 'Invalid email address.';
16. } else if (!value.split('@')[1].includes('.')) {
17. error = 'Email must contain a dot after @.';
18. }
20. setState({
21. email: value,
22. error: error,
23. });
24. };
26. const handleSubmit = (e) => {
27. e.preventDefault();
28. if (state.error === '' && state.email !== '') {
29. alert('Email has been sent!');
30. } else {
31. setState((prevState) => ({
32. ...prevState,
33. error: 'Please enter a valid email address before submitting.',
34. }));
35. }
36. };
38. return (
39. <div style={styles.container}>
40. <form onSubmit={handleSubmit} style={styles.form}>
41. <input
42. type="email"
43. value={state.email}
44. onChange={validateEmail}
45. placeholder="Enter email"
46. style={styles.input}
47. />
48. {state.error && <p style={styles.error}>{state.error}</p>}
49. <button type="submit" style={styles.button}>
50. Submit
51. </button>
52. </form>
53. </div>
54. );
55. }
57. const styles = {
58. container: {
59. margin: '50px',
60. fontFamily: 'Arial, sans-serif',
61. },
62. form: {
63. display: 'flex',
64. flexDirection: 'column',
65. maxWidth: '300px',
66. },
67. input: {
68. padding: '10px',
69. marginBottom: '10px',
70. fontSize: '16px',
71. },
72. error: {
73. color: 'red',
74. marginBottom: '10px',
75. },
76. button: {
77. padding: '10px',
78. backgroundColor: '#4CAF50',
79. color: 'white',
80. border: 'none',
81. fontSize: '16px',
82. cursor: 'pointer',
83. },
84. };
86. export default EmailForm;