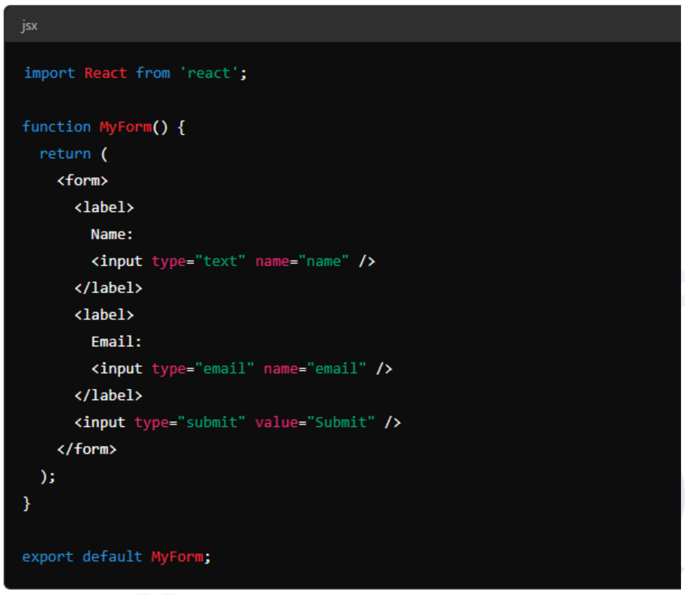
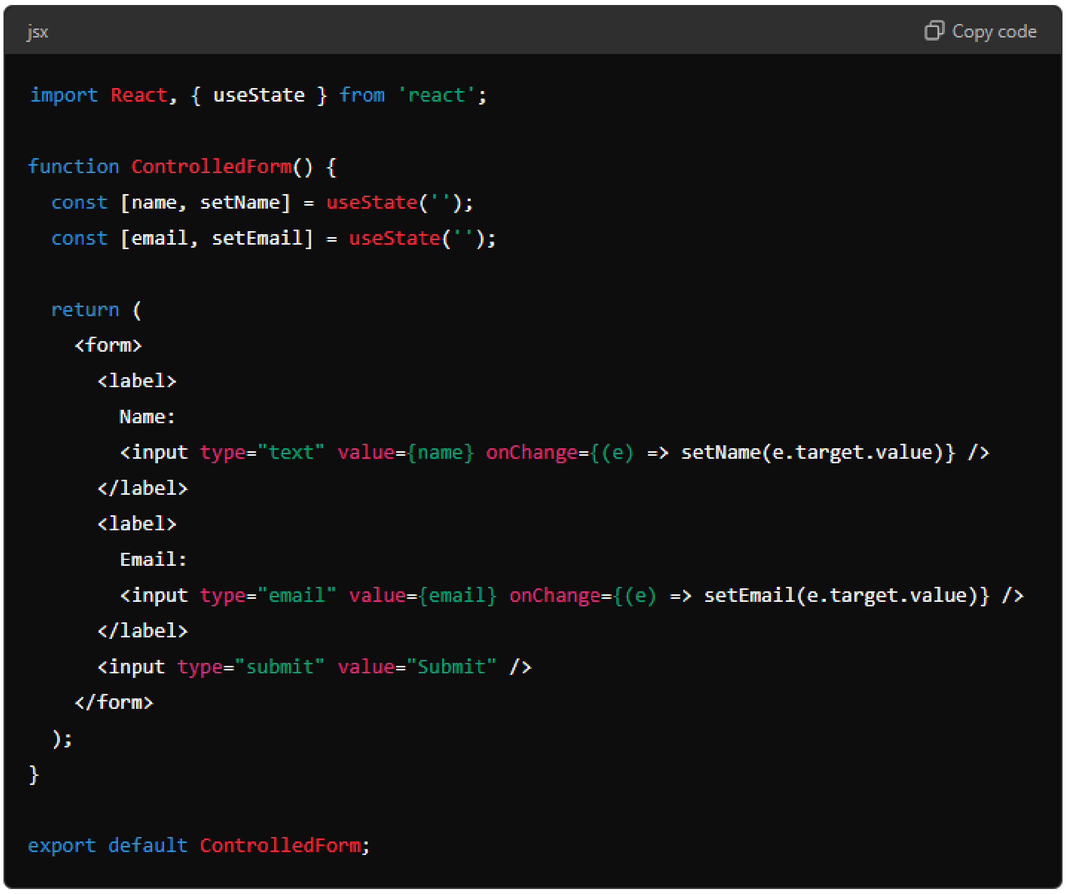
React Forms and Inputs

1. Forms and Inputs : In React, forms are used to collect user input. You can create forms using standard HTML form elements like <input>, <textarea>, and <select>.



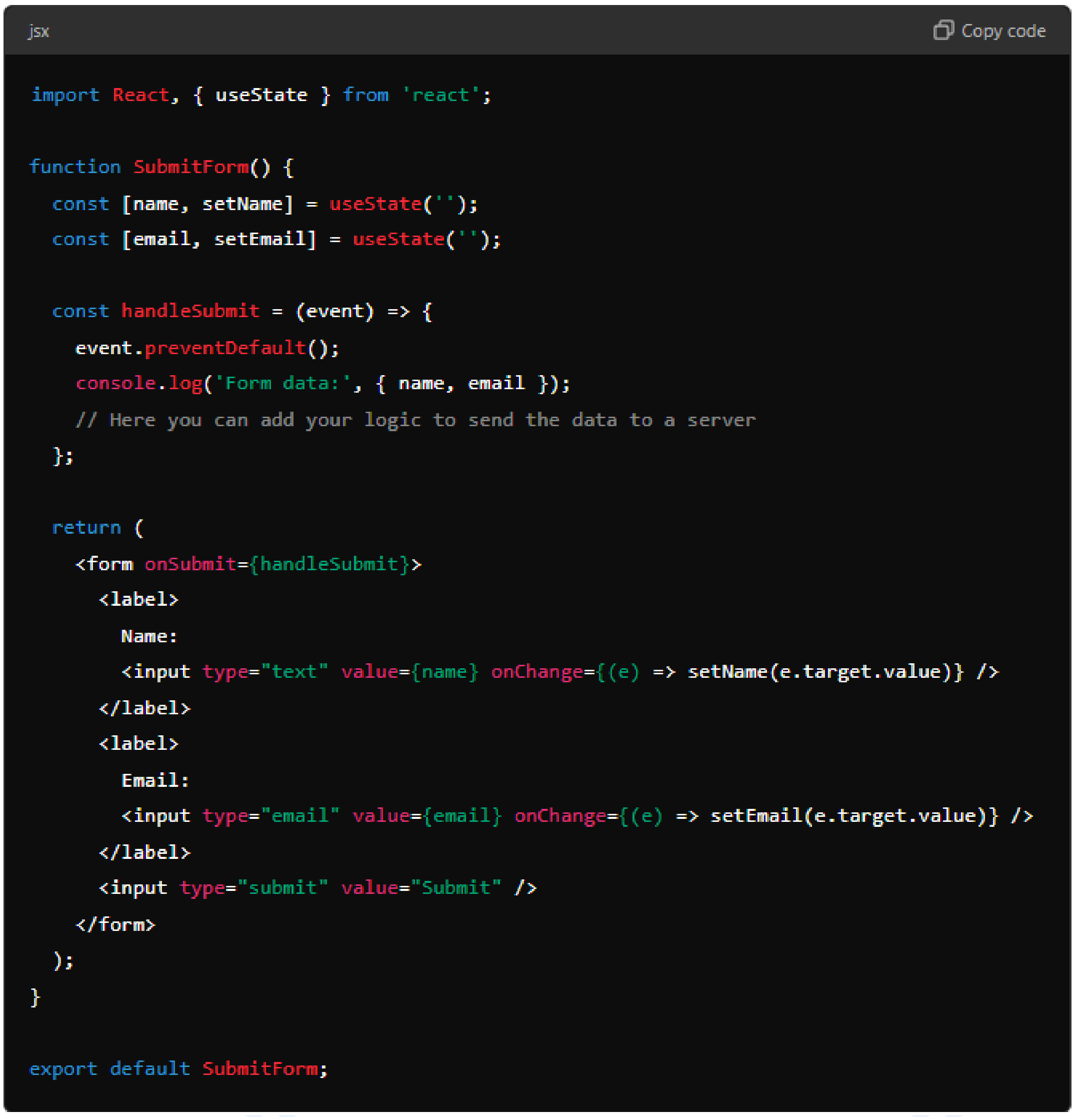
```jsx
import React from 'react';

function MyForm() {
  return (
    <form>
      <label>
        Name:
        <input type="text" name="name" />
      </label>
      <label>
        Email:
        <input type="email" name="email" />
      </label>
      <input type="submit" value="Submit" />
    </form>
  );
}

export default MyForm;
```

2. Controlled Components : In controlled components, the form data is handled by the component's state. This means the component that renders the form also controls what happens in that form on subsequent user input.



```jsx
import React, { useState } from 'react';

function ControlledForm() {
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');

  return (
    <form>
      <label>
        Name:
        <input type="text" value={name} onChange={(e) => setName(e.target.value)} />
      </label>
      <label>
        Email:
        <input type="email" value={email} onChange={(e) => setEmail(e.target.value)} />
      </label>
      <input type="submit" value="Submit" />
    </form>
  );
}

export default ControlledForm;
```

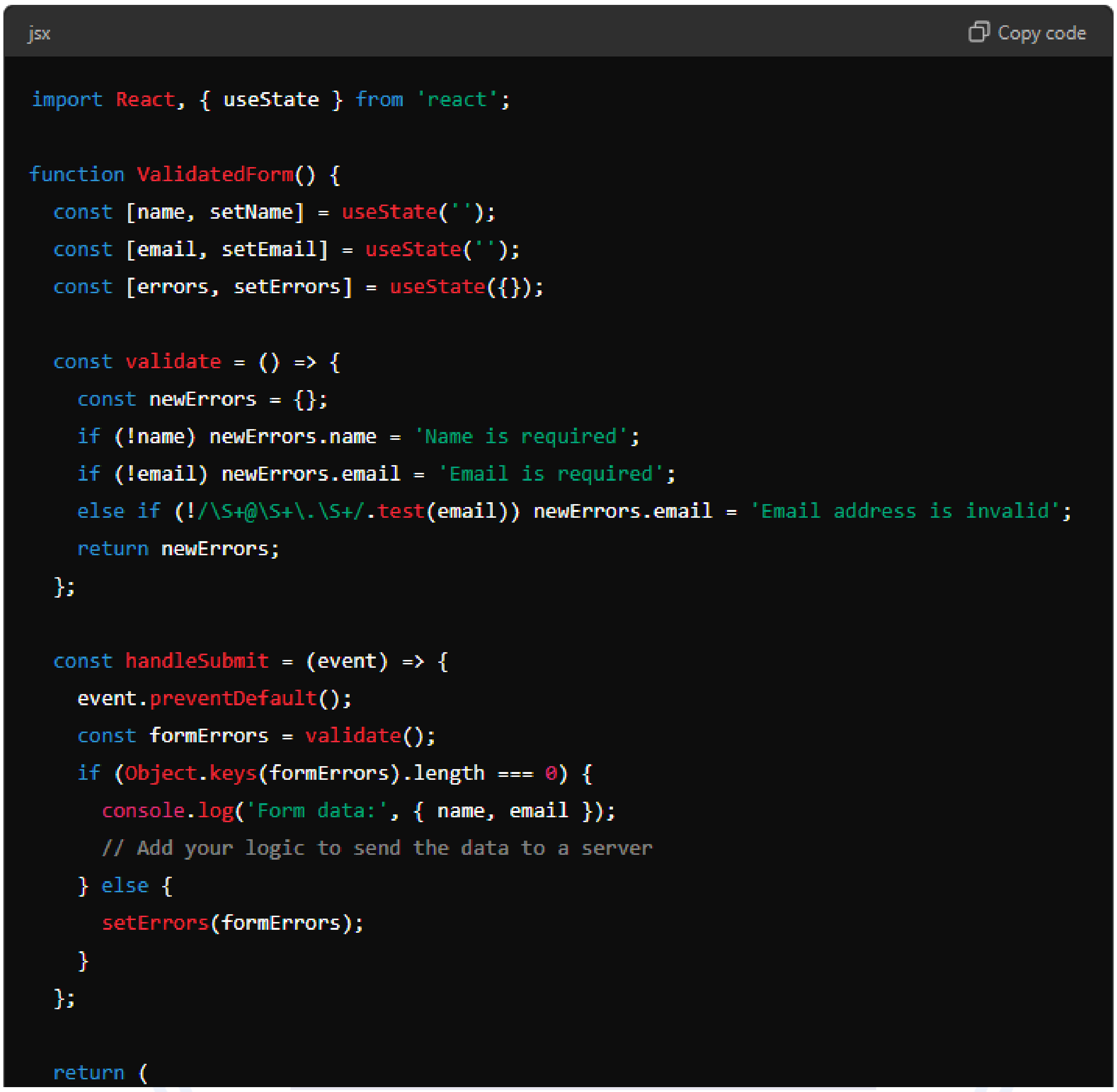
3. Handling Form Submissions: To handle form submissions, you can define a function to be called when the form is submitted. This function typically processes the form data or sends it to a server.



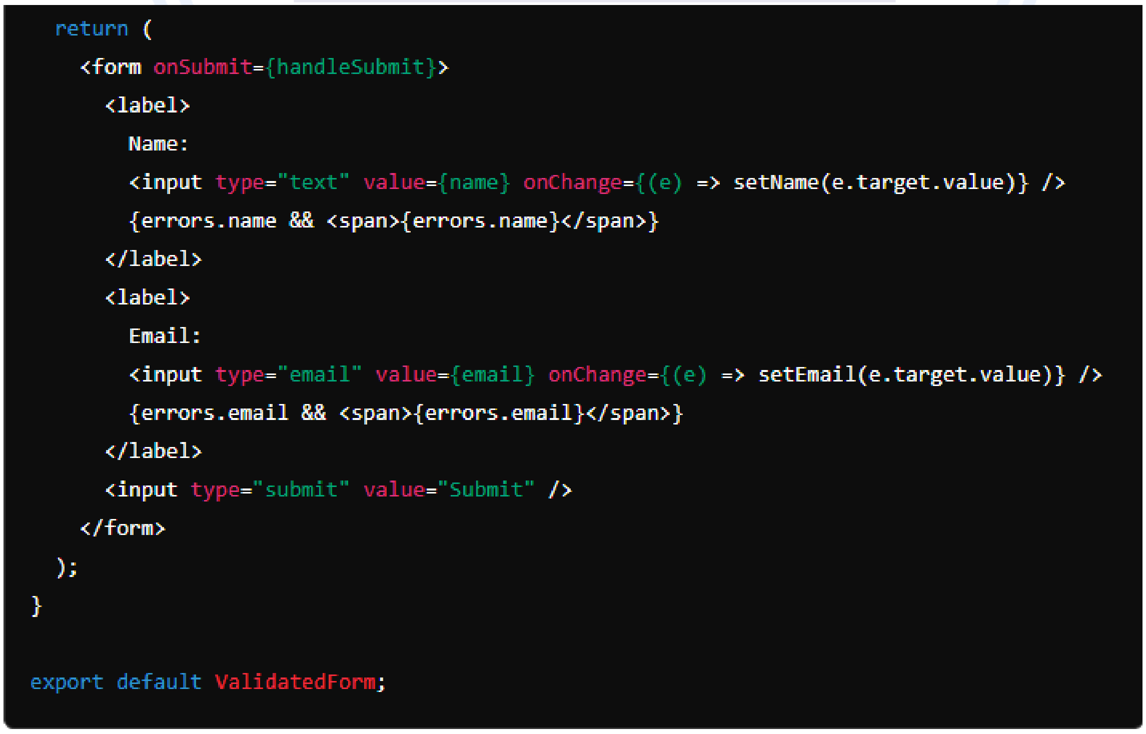
```jsx
import React, { useState } from 'react';

function SubmitForm() {
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');

  const handleSubmit = (event) => {
    event.preventDefault();
    console.log('Form data:', { name, email });
    // Here you can add your logic to send the data to a server
  };

  return (
    <form onSubmit={handleSubmit}>
      <label>
        Name:
        <input type="text" value={name} onChange={(e) => setName(e.target.value)} />
      </label>
      <label>
        Email:
        <input type="email" value={email} onChange={(e) => setEmail(e.target.value)} />
      </label>
      <input type="submit" value="Submit" />
    </form>
  );
}

export default SubmitForm;
```

4. Validation

Form validation ensures that the user input meets certain criteria before the form is submitted. Validation can be done using JavaScript.



```jsx
import React, { useState } from 'react';

function ValidatedForm() {
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');
  const [errors, setErrors] = useState({});

  const validate = () => {
    const newErrors = {};
    if (!name) newErrors.name = 'Name is required';
    if (!email) newErrors.email = 'Email is required';
    else if (!/\S+@\S+\.\S+/.test(email)) newErrors.email = 'Email address is invalid';
    return newErrors;
  };

  const handleSubmit = (event) => {
    event.preventDefault();
    const formErrors = validate();
    if (Object.keys(formErrors).length === 0) {
      console.log('Form data:', { name, email });
      // Add your logic to send the data to a server
    } else {
      setErrors(formErrors);
    }
  };

  return (
```



```jsx
  return (
    <form onSubmit={handleSubmit}>
      <label>
        Name:
        <input type="text" value={name} onChange={(e) => setName(e.target.value)} />
        {errors.name && <span>{errors.name}</span>}
      </label>
      <label>
        Email:
        <input type="email" value={email} onChange={(e) => setEmail(e.target.value)} />
        {errors.email && <span>{errors.email}</span>}
      </label>
      <input type="submit" value="Submit" />
    </form>
  );
}

export default ValidatedForm;
```

This example demonstrates how to add validation for required fields and a valid email format. The form will not submit if the validations fail, and error messages will be displayed to the user.