**PropTypes**

By Traltb@fe.edu.vn

**PropTypes** là một công cụ mạnh mẽ trong React giúp xác thực các loại dữ liệu mà component của bạn nhận vào dưới dạng props. Điều này giúp dễ dàng phát hiện lỗi khi props không đúng loại hoặc không được truyền đúng cách.

Các bước sử dụng:

Bước 1: Cài đặt ProTypes

npm install prop-types

Bước 2: Sau khi cài đặt, bạn cần import PropTypes vào file React component của mình.

import PropTypes from 'prop-types';

Bước 3: Tạo component sử dụng PropTypes cho các props trong component của mình. Điều này giúp React xác định kiểu dữ liệu mà props nhận vào.

**Ví dụ 1: Tạo 1 ứng dụng kiểm tra thông tin người dùng**

Bước 1: Tạo ứng dụng react:

npx create-react-app proptype-exercise

cd proptype-exercise

Bước 2: Cài đặt prop-types

npm install prop-types

Bước 3: Tạo component sử dụng PropTypes

Tạo component UserProfile.js như sau:

import React from "react";

import PropTypes from "prop-types";

// Component UserProfile

const UserProfile = ({ name, age }) => {

  // Kiểm tra nếu `name` không được truyền hoặc rỗng

  if (!name || typeof name !== "string") {

    return (

      <div>

        <h3>Thông Tin Người Dùng</h3>

        <p style={{ color: "red" }}>

          Tên không hợp lệ hoặc không được cung cấp!

        </p>

      </div>

    );

  }

  // Kiểm tra nếu `age` không được nhập hoặc không phải là một số hợp lệ

  if (!age) {

    return (

      <div>

        <h3>Thông Tin Người Dùng</h3>

        <p>Tên: {name}</p>

        <p style={{ color: "red" }}>Tuổi không được để trống!</p>

      </div>

    );

  } else if (isNaN(age)) {

    return (

      <div>

        <h3>Thông Tin Người Dùng</h3>

        <p>Tên: {name}</p>

        <p style={{ color: "red" }}>Tuổi phải là một số hợp lệ!</p>

      </div>

    );

  }

  return (

    <div>

      <h3>Thông Tin Người Dùng</h3>

      <p>Tên: {name}</p>

      <p>Tuổi: {age}</p>

    </div>

  );

};

// Xác định PropTypes cho UserProfile

UserProfile.propTypes = {

  name: PropTypes.string.isRequired, // 'name' phải là một chuỗi và là bắt buộc

  age: PropTypes.oneOfType([PropTypes.string, PropTypes.number]).isRequired, // 'age' có thể là chuỗi hoặc số và là bắt buộc

};

export default UserProfile;

Bước 4: Sử dụng UserProfile trong App:

import React from "react";

import UserProfile from "./components/UserProfile";

const App = () => {

  return (

    <div className="App">

      <h1>Ứng Dụng React</h1>

      {/\* Trường hợp hợp lệ \*/}

      <UserProfile name="Nguyễn Văn A" age={25} />

      {/\* Trường hợp name không hợp lệ \*/}

      <UserProfile name="" age={25} />

      {/\* Trường hợp tuổi không hợp lệ \*/}

      <UserProfile name="Nguyễn Văn B" age="twenty five" />

      {/\* Trường hợp không nhập tuổi \*/}

      <UserProfile name="Nguyễn Văn C" age={null} />

    </div>

  );

};

export default App;

Kết quả:

A screenshot of a computer

AI-generated content may be incorrect.

**Ví dụ 2: Cải tiến ví dụ trên, sử dụng Form**

Bước 1: Cài đặt thêm react bootstrap

npm install react-bootstrap bootstrap

Bước 2: import Boostrap vào file index.js

import 'bootstrap/dist/css/bootstrap.min.css';

Bước 3: Tạo component UserProfile2.js

import React, { useState } from "react";

import PropTypes from "prop-types";

import { Form, Button, Container } from "react-bootstrap";

// Component UserProfile

const UserProfile2 = ({ name, age, onSubmit }) => {

  const [formData, setFormData] = useState({

    name: name,

    age: age,

  });

  const [errors, setErrors] = useState({});

  // Hàm xử lý thay đổi giá trị input

  const handleChange = (e) => {

    const { name, value } = e.target;

    setFormData({

      ...formData,

      [name]: value,

    });

  };

  // Kiểm tra lỗi dữ liệu trước khi submit

  const validateForm = () => {

    const newErrors = {};

    // Kiểm tra tuổi phải từ 18 đến 100

    const age = parseInt(formData.age, 10);

    if (!formData.name) {

      newErrors.name = "Tên là bắt buộc";

    }

    if (!formData.age) {

      newErrors.age = "Tuổi không được để trống!";

    } else if (isNaN(age)) {

      newErrors.age = "Tuổi phải là một số hợp lệ!";

    } else if (age < 18 || age > 100) {

      newErrors.age = "Tuổi phải nằm trong khoảng từ 18 đến 100!";

    }

    setErrors(newErrors);

    return Object.keys(newErrors).length === 0;

  };

  // Hàm submit form

  const handleSubmit = (e) => {

    e.preventDefault();

    if (validateForm()) {

      onSubmit(formData);

    }

  };

  return (

    <Container>

      <h3>Thông Tin Người Dùng</h3>

      <Form onSubmit={handleSubmit}>

        {/\* Trường tên \*/}

        <Form.Group controlId="formName">

          <Form.Label>Tên</Form.Label>

          <Form.Control

            type="text"

            name="name"

            value={formData.name}

            onChange={handleChange}

            isInvalid={!!errors.name}

          />

          <Form.Control.Feedback type="invalid">

            {errors.name}

          </Form.Control.Feedback>

        </Form.Group>

        {/\* Trường tuổi \*/}

        <Form.Group controlId="formAge">

          <Form.Label>Tuổi</Form.Label>

          <Form.Control

            type="text"

            name="age"

            value={formData.age}

            onChange={handleChange}

            isInvalid={!!errors.age}

          />

          <Form.Control.Feedback type="invalid">

            {errors.age}

          </Form.Control.Feedback>

        </Form.Group>

        {/\* Nút submit \*/}

        <Button variant="primary" type="submit">

          Gửi

        </Button>

      </Form>

    </Container>

  );

};

// Xác định PropTypes cho UserProfile

UserProfile2.propTypes = {

  name: PropTypes.string.isRequired, // 'name' là chuỗi bắt buộc

  age: PropTypes.oneOfType([PropTypes.string, PropTypes.number]), // 'age' có thể là chuỗi hoặc số

  onSubmit: PropTypes.func.isRequired, // Hàm onSubmit để xử lý khi người dùng submit form

};

export default UserProfile2;

Bước 4: Sử dụng UserProfile2.js trong App.js

import React from "react";

import UserProfile2 from "./components/UserProfile2";

const App = () => {

  const handleFormSubmit = (formData) => {

    console.log("Dữ liệu đã gửi:", formData);

  };

  return (

    <div className="App">

      <h1>Ứng Dụng React</h1>

      <UserProfile2 name="Nguyễn Văn A" age={25} onSubmit={handleFormSubmit} />

      <UserProfile2

        name="Nguyễn Văn B"

        age="twenty five"

        onSubmit={handleFormSubmit}

      />

      <UserProfile2 name="" age={30} onSubmit={handleFormSubmit} />

    </div>

  );

};

export default App;

A screenshot of a chat

AI-generated content may be incorrect.

**Ví dụ 3:**

Tạo component MyForm.js có nội dung như sau:

import React, { useState, useReducer } from "react";

import { Button, Form, Container, Alert } from "react-bootstrap";

import PropTypes from "prop-types";

// Reducer để quản lý trạng thái form

const initialState = {

  name: "",

  email: "",

  password: "",

  isSubmitted: false,

};

const formReducer = (state, action) => {

  switch (action.type) {

    case "SET\_FIELD":

      return { ...state, [action.field]: action.value };

    case "SUBMIT":

      return { ...state, isSubmitted: true };

    default:

      return state;

  }

};

// Component Form

const MyForm = ({ title, onSubmit }) => {

  const [state, dispatch] = useReducer(formReducer, initialState);

  const [errors, setErrors] = useState({});

  const [showAlert, setShowAlert] = useState(false); // Biến để kiểm soát việc hiển thị alert

  // Hàm xử lý thay đổi giá trị input

  const handleChange = (e) => {

    const { name, value } = e.target;

    dispatch({ type: "SET\_FIELD", field: name, value });

  };

  // Hàm kiểm tra lỗi trước khi submit

  const handleValidation = () => {

    const newErrors = {};

    if (!state.name) newErrors.name = "Tên không được để trống!";

    if (!state.email) newErrors.email = "Email không được để trống!";

    if (!state.password) newErrors.password = "Mật khẩu không được để trống!";

    // Nếu có lỗi, hiển thị alert

    if (Object.keys(newErrors).length > 0) {

      setShowAlert(true);

    } else {

      setShowAlert(false);

    }

    setErrors(newErrors);

    return Object.keys(newErrors).length === 0;

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    if (handleValidation()) {

      dispatch({ type: "SUBMIT" });

      onSubmit(state);

    }

  };

  return (

    <Container>

      <h3>{title}</h3>

      {/\* Hiển thị Alert nếu có lỗi \*/}

      {showAlert && (

        <Alert variant="danger">

          <strong>Lỗi:</strong> Vui lòng điền đầy đủ thông tin.

        </Alert>

      )}

      <Form onSubmit={handleSubmit}>

        <Form.Group controlId="formName">

          <Form.Label>Tên</Form.Label>

          <Form.Control

            type="text"

            name="name"

            value={state.name}

            onChange={handleChange}

            isInvalid={!!errors.name}

          />

          <Form.Control.Feedback type="invalid">

            {errors.name}

          </Form.Control.Feedback>

        </Form.Group>

        <Form.Group controlId="formEmail">

          <Form.Label>Email</Form.Label>

          <Form.Control

            type="email"

            name="email"

            value={state.email}

            onChange={handleChange}

            isInvalid={!!errors.email}

          />

          <Form.Control.Feedback type="invalid">

            {errors.email}

          </Form.Control.Feedback>

        </Form.Group>

        <Form.Group controlId="formPassword">

          <Form.Label>Mật khẩu</Form.Label>

          <Form.Control

            type="password"

            name="password"

            value={state.password}

            onChange={handleChange}

            isInvalid={!!errors.password}

          />

          <Form.Control.Feedback type="invalid">

            {errors.password}

          </Form.Control.Feedback>

        </Form.Group>

        <Button variant="primary" type="submit">

          Submit

        </Button>

      </Form>

    </Container>

  );

};

// Xác định PropTypes cho MyForm

MyForm.propTypes = {

  title: PropTypes.string.isRequired, // Tiêu đề phải là một chuỗi

  onSubmit: PropTypes.func.isRequired, // Hàm onSubmit phải là một function

};

export default MyForm;

**Bước 2: Sử dụng MyForm trong App.js**

import React from "react";

import MyForm from "././components/MyForm"; // Giả sử MyForm được lưu trong thư mục cùng với App.js

const App = () => {

  const handleFormSubmit = (formData) => {

    console.log("Dữ liệu đã gửi:", formData);

  };

  return (

    <div className="App">

      <h1>Ứng Dụng React</h1>

      <MyForm title="Đăng Ký Người Dùng" onSubmit={handleFormSubmit} />

    </div>

  );

};

export default App;

A red and white rectangular object

AI-generated content may be incorrect.

Giải thích:

 **useState và useReducer:**

* useState được sử dụng để lưu trữ các lỗi nhập liệu.
* useReducer được dùng để quản lý trạng thái của form, bao gồm các trường như tên, email, mật khẩu và trạng thái form đã được submit.

 **PropTypes:**

* Trong component MyForm, chúng ta xác định các PropTypes. Prop title phải là một chuỗi và prop onSubmit phải là một hàm.
* Sử dụng PropTypes giúp bảo vệ tính chính xác của dữ liệu truyền vào và dễ dàng phát hiện lỗi khi chạy ứng dụng.

 **Form và Xử Lý Validation:**

* Chúng ta tạo form với các trường nhập liệu như Tên, Email, và Mật khẩu.
* Alert sẽ hiển thị khi có lỗi và người dùng chưa điền đầy đủ thông tin. showAlert là biến trạng thái để kiểm soát việc hiển thị Alert
* Các lỗi sẽ được kiểm tra trong hàm handleValidation và hiển thị thông qua Form.Control.Feedback.
* Trong hàm handleValidation, nếu có lỗi, chúng ta sẽ kích hoạt setShowAlert(true) để hiển thị thông báo lỗi. Nếu không có lỗi, setShowAlert(false) sẽ ẩn thông báo lỗi.

 **Submit Form:**

* Khi người dùng bấm nút submit, hàm handleSubmit sẽ kiểm tra tính hợp lệ của form và nếu form hợp lệ, nó sẽ gọi onSubmit với dữ liệu form.

**Ví dụ 4: Tạo Form sử dụng PropTypes để validate dữ liệu:**

* Tên không được để trống, chứa 3-50 kí tự
* Tuổi không được để trống, từ 18-100 tuổi
* Email không được để trống, đúng định dạng
* Số điện thoại: từ 10-15 chữ số
* Phải đồng ý với điều khoản

A red and blue lines on a white background

AI-generated content may be incorrect.

**Exercise 19: Demo about Validation in component**