**Fetching and caching data in React**

By traltb@fe.edu.vn

**Fetching (Lấy dữ liệu)**: Xem chi tiết nội dung ở slide: slot19\_Fetching and caching data in react

**Fetching dữ liệu** trong React có thể thực hiện bằng fetch() hoặc Axios, cả hai đều sử dụng Promises để xử lý dữ liệu bất đồng bộ.

**Ví dụ 1**: Sử dụng fetch() để lấy dữ liệu từ API và hiển thị nó trong một component React:

Sử dụng ứng dụng demo-json-server ở slot 18, tạo mới 1 component tên PostList.js trong thư mục src/components

File PostList.js:

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

const PostList = () => {

  const [data, setData] = useState(null);

  const [loading, setLoading] = useState(true);

  useEffect(() => {

    fetch("http://localhost:3000/posts")  // API GET Posts

      .then((response) => response.json())  // Chuyển dữ liệu về dạng JSON

      .then((data) => {

        setData(data);  // Lưu dữ liệu vào state

        setLoading(false);  // Đánh dấu việc tải xong

      })

      .catch((error) => console.error("Lỗi khi lấy dữ liệu:", error));

  }, []);  // Chạy 1 lần khi component được mount

  if (loading) {

    return <div>Đang tải...</div>;  // Hiển thị thông báo đang tải

  }

  return (

    <div>

      <h1>Danh sách bài viết</h1>

      <ul>

        {data.map((post) => (

          <li key={post.id}>

            <h3>{post.title}</h3>

            <p>{post.content}</p>

          </li>

        ))}

      </ul>

    </div>

  );

};

export default PostList;

Giải thích:

 **useEffect**: Được dùng để thực hiện tác vụ lấy dữ liệu khi component được render lần đầu.

 **fetch()**: Dùng để gửi yêu cầu HTTP và nhận về phản hồi từ API.

 **.then()**: Xử lý phản hồi từ API, chuyển đổi dữ liệu sang JSON.

 **.catch()**: Xử lý lỗi nếu có vấn đề khi lấy dữ liệu.

Cập nhật file App.js

import PostList from './components/PostList';

function App() {

  return (

    <div className="App">

      <PostList />

    </div>

  );

}

export default App;

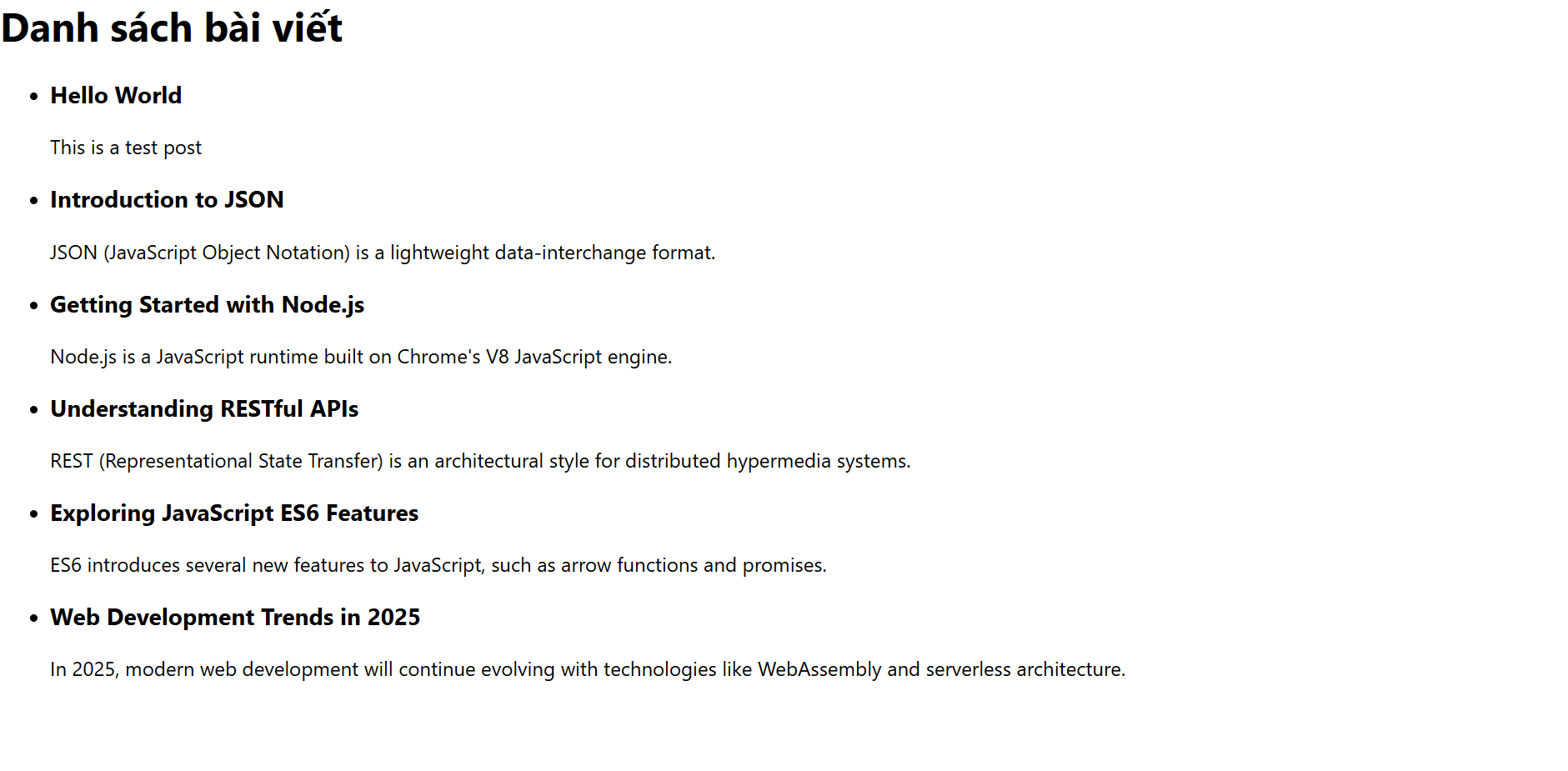
**Cách chạy ứng dụng:**

* Vào Terminal: Chọn Split Terminal, chạy server:

json-server --watch db.json

* Tại Terminal chính: chạy npm start

Kết quả chạy:



**Đọc thêm: Quản lý tác vụ bất đồng bộ với Promises**

JavaScript sử dụng **Promises** để xử lý các tác vụ bất đồng bộ. Một Promise đại diện cho một giá trị có thể có hoặc chưa có, có thể thành công hoặc thất bại.

Promise có ba trạng thái:

* **Pending**: Trạng thái ban đầu, khi promise chưa được thực hiện.
* **Fulfilled**: Khi tác vụ bất đồng bộ thành công.
* **Rejected**: Khi tác vụ bất đồng bộ thất bại.

const fetchData = () => {

    return new Promise((resolve, reject) => {

      setTimeout(() => {

        const data = "Dữ liệu đã được tải";

        const error = null;

        if (data) {

          resolve(data);  // Kết thúc Promise thành công với dữ liệu

        } else {

          reject(error);  // Kết thúc Promise thất bại

        }

      }, 2000);  // Giả lập thời gian tải dữ liệu

    });

  };

**Ví dụ 2**: Mặc dù Promise rất hữu ích, sử dụng **async/await** cung cấp cú pháp dễ đọc hơn và giúp xử lý mã bất đồng bộ một cách rõ ràng hơn thay vì sử dụng .then() và .catch() và tránh lồng các lời gọi then, catch này.

**async** dùng để đánh dấu một hàm là bất đồng bộ, và **await** dùng để tạm dừng hàm cho đến khi Promise được giải quyết hoặc bị từ chối.

 **async**: Đánh dấu hàm fetchData là bất đồng bộ.

 **await**: Chờ đợi kết quả của fetch() và response.json().

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

const PostList = () => {

  const [data, setData] = useState(null);

  const [loading, setLoading] = useState(true);

  // Sử dụng useEffect để thực hiện tác vụ fetching khi component mount

  useEffect(() => {

    // Định nghĩa hàm async để thực hiện fetch dữ liệu

    const fetchData = async () => {

      try {

        // Gửi yêu cầu fetch và chờ kết quả

        const response = await fetch("http://localhost:3000/posts");

        if (!response.ok) {

          throw new Error("Không thể lấy dữ liệu");

        }

        const data = await response.json();  // Chuyển dữ liệu về dạng JSON

        setData(data);  // Lưu dữ liệu vào state

        setLoading(false);  // Đánh dấu việc tải xong

      } catch (error) {

        console.error("Lỗi khi lấy dữ liệu:", error);

      }

    };

    fetchData();  // Gọi hàm fetchData khi component mount

  }, []);  // Chạy 1 lần khi component được mount

  if (loading) {

    return <div>Đang tải...</div>;  // Hiển thị thông báo đang tải

  }

  return (

    <div>

      <h1>Danh sách bài viết</h1>

      <ul>

        {data.map((post) => (

          <li key={post.id}>

            <h3>{post.title}</h3>

            <p>{post.content}</p>

          </li>

        ))}

      </ul>

    </div>

  );

};

export default PostList;

 **Hàm fetchData**:

* Thay đổi gọi fetch() ở ví dụ 1 sang một hàm bất đồng bộ fetchData với từ khóa async.
* await được sử dụng để đợi kết quả từ fetch(), giúp mã dễ đọc hơn và tránh việc lồng .then().

 **response.ok**:

* Trước khi tiếp tục với response.json(), kiểm tra xem phản hồi có thành công hay không (response.ok). Nếu không, chúng ta ném một lỗi.

 **Sử dụng try/catch**:

* try/catch giúp bắt và xử lý lỗi nếu có sự cố trong quá trình lấy dữ liệu, giúp mã dễ dàng quản lý lỗi hơn.

 **useEffect**:

* Hàm fetchData được gọi trong useEffect khi component được mount.

Ví dụ 3: Mặc dù fetch() là một công cụ mạnh mẽ, nhưng nó có thể khá dài dòng. **Axios** là một thư viện AJAX phổ biến giúp việc gửi các yêu cầu HTTP trở nên đơn giản hơn.

Axios giúp đơn giản hóa việc thực hiện các yêu cầu HTTP, tự động xử lý JSON và cung cấp các phương thức dễ sử dụng như .get(), .post(), .put(), .delete().

**Cài đặt axios:**

**Dừng server ở terminal chính: Ctrl+C**

Gõ lệnh sau để cài đặt axios: npm install axios

Thay đổi mã nguồn của file PostList.js như sau:

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

import axios from "axios";

const PostList = () => {

  const [data, setData] = useState(null);

  const [loading, setLoading] = useState(true);

  // Sử dụng useEffect để thực hiện tác vụ fetching khi component mount

  useEffect(() => {

    // Định nghĩa hàm async để thực hiện fetching dữ liệu

    const fetchData = async () => {

      try {

        // Gửi yêu cầu GET với axios và chờ kết quả

        const response = await axios.get("http://localhost:3000/posts");

        setData(response.data);  // Lưu dữ liệu vào state từ phản hồi

        setLoading(false);  // Đánh dấu việc tải xong

      } catch (error) {

        console.error("Lỗi khi lấy dữ liệu:", error);

      }

    };

    fetchData();  // Gọi hàm fetchData khi component mount

  }, []);  // Chạy 1 lần khi component được mount

  if (loading) {

    return <div>Đang tải...</div>;  // Hiển thị thông báo đang tải

  }

  return (

    <div>

      <h1>Danh sách bài viết</h1>

      <ul>

        {data.map((post) => (

          <li key={post.id}>

            <h3>{post.title}</h3>

            <p>{post.content}</p>

          </li>

        ))}

      </ul>

    </div>

  );

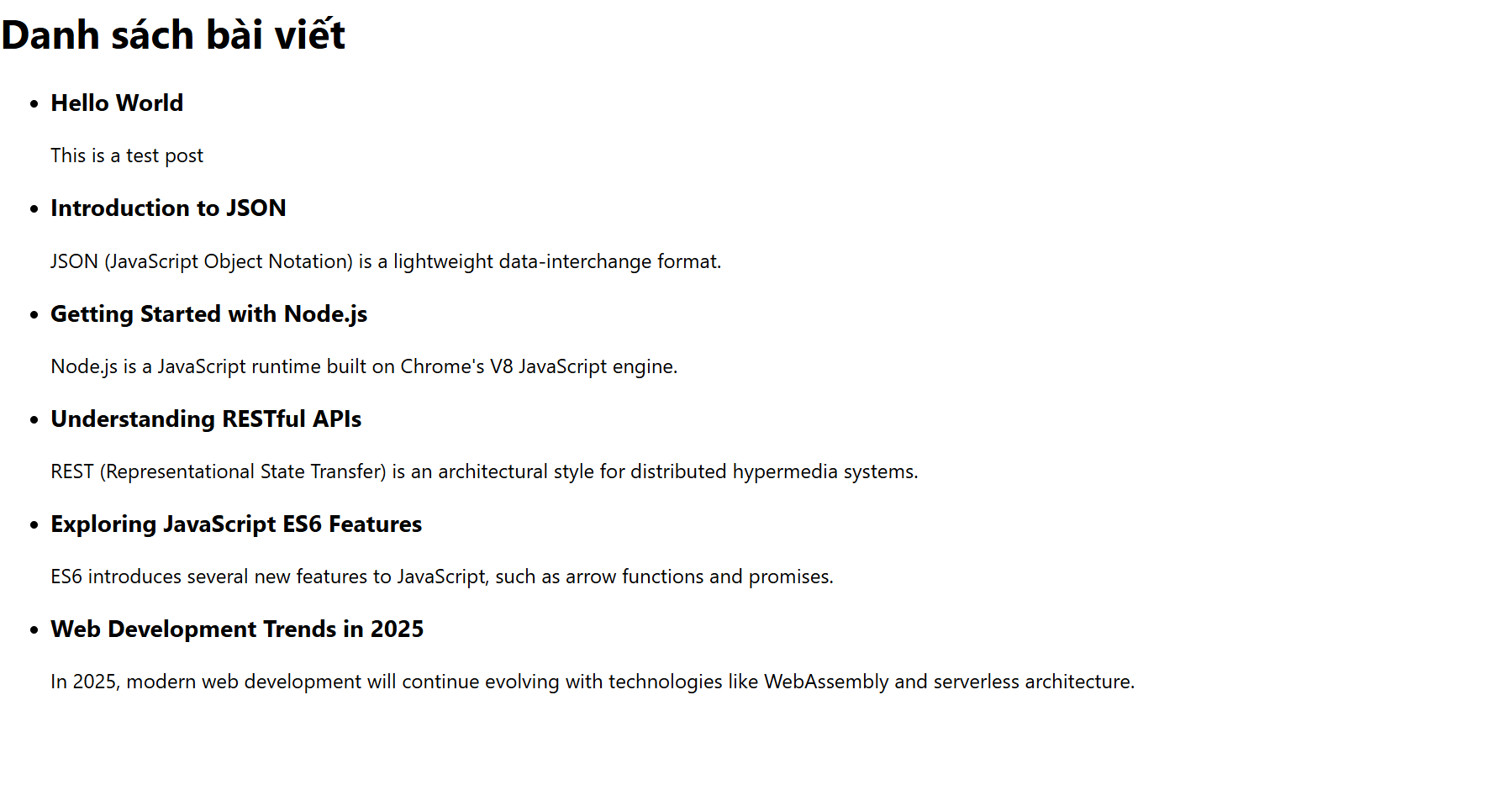
};

export default PostList;

 **Sử dụng axios.get()**: Thay vì sử dụng fetch(), bạn sử dụng axios.get() để gửi yêu cầu GET tới API. axios.get() trả về một Promise, và kết quả sẽ được chứa trong response.data.

 **Xử lý lỗi**: Tương tự như fetch(), bạn có thể sử dụng try/catch để xử lý lỗi nếu có vấn đề khi lấy dữ liệu từ API.

Kết quả tương tự như trên:



Ví dụ 4: Thực hiện thêm các tính năng CREATE, UPDATE, DELETE a post.

Cài đặt react router để cho phép điều hướng đến các trang:

npm install react-router-dom

Cập nhật lại file PostList.js:  
Thêm Thẻ Link để liên kết đến CreatePost.js

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

import axios from "axios";

import { Link } from "react-router-dom";  // Import Link từ react-router-dom

const PostList = () => {

  const [data, setData] = useState(null);

  const [loading, setLoading] = useState(true);

  // Use useEffect to fetch data when the component mounts

  useEffect(() => {

    const fetchData = async () => {

      try {

        const response = await axios.get("http://localhost:3000/posts");

        setData(response.data); // Save response data to state

        setLoading(false); // Mark loading as complete

      } catch (error) {

        console.error("Error fetching data:", error);

        setLoading(false); // Make sure to stop loading on error

      }

    };

    fetchData();

  }, []); // Empty array ensures this runs only once when the component mounts

  if (loading) {

    return <div>Đang tải...</div>; // Show loading message while data is being fetched

  }

  if (!data || data.length === 0) {

    return <div>Không có bài viết nào!</div>; // Show a message if no posts are available

  }

  return (

    <div>

      <h1>Danh sách bài viết</h1>

      <button>

        <Link to="/create">Tạo bài viết mới</Link>  {/\* Button Create dẫn đến trang CreatePost \*/}

      </button>

      <ul>

        {data.map((post) => (

          <li key={post.id}>

            <h3>{post.title}</h3>

            <p>{post.content}</p>

          </li>

        ))}

      </ul>

    </div>

  );

};

export default PostList;

Cập nhật file App.js:

import React from "react";

import { BrowserRouter as Router, Route, Routes } from "react-router-dom";

import PostList from "./components/PostList";

import CreatePost from "./components/CreatePost";

const App = () => {

  return (

    <Router>

      <div>

        <h1>Quản lý Bài Viết</h1>

        <Routes>

          <Route path="/" element={<PostList />} />

          <Route path="/create" element={<CreatePost />} />

        </Routes>

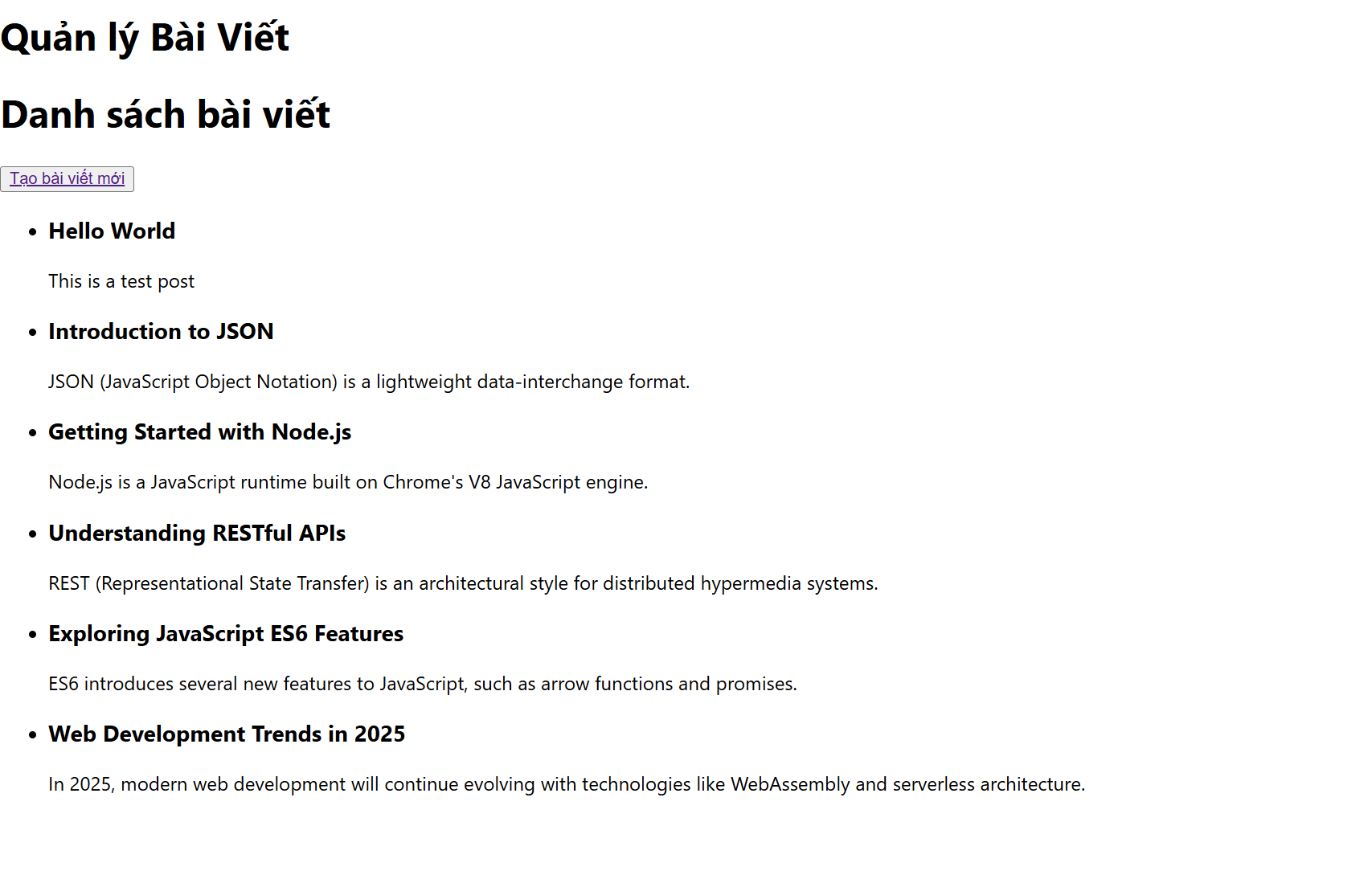
      </div>

    </Router>

  );

};

export default App;



File CreatePost.js:

Tạo mới 1 bài post, sau khi tạo xong, chuyển hướng đến PostList dùng useNavigate:

import React, { useState } from "react";

import axios from "axios";

import { useNavigate } from "react-router-dom";  // Import useNavigate

const CreatePost = () => {

  const [title, setTitle] = useState("");

  const [content, setContent] = useState("");

  const [status, setStatus] = useState("");

  const navigate = useNavigate();  // Khởi tạo useNavigate để chuyển hướng

  const handleSubmit = async (e) => {

    e.preventDefault();

    const newPost = {

      title,

      content,

    };

    try {

      // Gửi yêu cầu POST để tạo bài viết mới

      await axios.post("http://localhost:3000/posts", newPost);

      setStatus("Bài viết đã được tạo thành công!");

      setTitle("");  // Clear input fields

      setContent("");

      // Chuyển hướng đến trang ListPost sau khi tạo bài viết thành công

      navigate("/");  // Giả sử ListPost là trang trang chủ, thay đổi theo yêu cầu thực tế của bạn

    } catch (error) {

      setStatus("Có lỗi xảy ra khi tạo bài viết.");

      console.error("Lỗi khi tạo bài viết:", error);

    }

  };

  return (

    <div>

      <h1>Thêm bài viết mới</h1>

      <form onSubmit={handleSubmit}>

        <input

          type="text"

          placeholder="Tiêu đề"

          value={title}

          onChange={(e) => setTitle(e.target.value)}

        />

        <br />

        <br />

        <textarea

          placeholder="Nội dung"

          value={content}

          onChange={(e) => setContent(e.target.value)}

        />

        <br />

        <button type="submit">Tạo bài viết</button>

      </form>

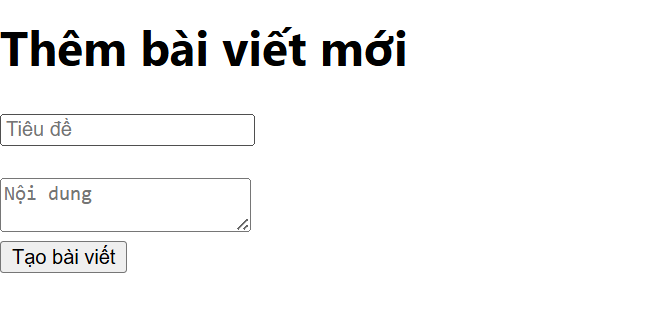
      <p>{status}</p>

    </div>

  );

};

export default CreatePost;



Sau khi thêm bài viết, bài viết mới sẽ được thêm vào danh sách PostList.

Tương tự: cho chức năng EditPost, DeletePost

Cập nhật lại file PostList.js để cho phép Edit và Delete Post:

**File PostList.js:**

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

import axios from "axios";

import { Link, useNavigate } from "react-router-dom";  // Thêm useNavigate để chuyển hướng sau khi xóa

const PostList = () => {

  const [data, setData] = useState(null);

  const [loading, setLoading] = useState(true);

  const navigate = useNavigate();  // Khởi tạo useNavigate để chuyển hướng sau khi xóa bài viết

  // Fetch dữ liệu khi component mount

  useEffect(() => {

    const fetchData = async () => {

      try {

        const response = await axios.get("http://localhost:3000/posts");

        setData(response.data); // Lưu dữ liệu vào state

        setLoading(false); // Đánh dấu việc tải xong

      } catch (error) {

        console.error("Lỗi khi lấy dữ liệu:", error);

        setLoading(false); // Dừng trạng thái tải nếu có lỗi

      }

    };

    fetchData();

  }, []); // Chạy 1 lần khi component được mount

  if (loading) {

    return <div>Đang tải...</div>; // Hiển thị thông báo đang tải

  }

  if (!data || data.length === 0) {

    return <div>Không có bài viết nào!</div>; // Hiển thị thông báo nếu không có bài viết

  }

  // Hàm để xóa bài viết

  const handleDelete = async (id) => {

    try {

      await axios.delete(`http://localhost:3000/posts/${id}`);

      setData(data.filter((post) => post.id !== id)); // Cập nhật danh sách bài viết sau khi xóa

    } catch (error) {

      console.error("Lỗi khi xóa bài viết:", error);

    }

  };

  return (

    <div>

      <h1>Danh sách bài viết</h1>

      <button>

        <Link to="/create">Tạo bài viết mới</Link>  {/\* Button Create dẫn đến trang CreatePost \*/}

      </button>

      <ul>

        {data.map((post) => (

          <li key={post.id}>

            <h3>{post.title}</h3>

            <p>{post.content}</p>

            <div>

              {/\* Link đến trang chỉnh sửa bài viết \*/}

              <Link to={`/edit/${post.id}`}>Chỉnh sửa</Link>

              <button onClick={() => handleDelete(post.id)}>Xóa</button>

            </div>

          </li>

        ))}

      </ul>

    </div>

  );

};

export default PostList;

**File EditPost.js:**

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

import axios from "axios";

import { useNavigate, useParams } from "react-router-dom";

const EditPost = () => {

  const [title, setTitle] = useState("");

  const [content, setContent] = useState("");

  const [status, setStatus] = useState("");

  const { id } = useParams(); // Lấy id từ URL

  const navigate = useNavigate();  // Điều hướng sau khi cập nhật

  useEffect(() => {

    const fetchPost = async () => {

      try {

        const response = await axios.get(`http://localhost:3000/posts/${id}`);

        if (response.data) {

          setTitle(response.data.title);

          setContent(response.data.content);

        } else {

          console.error(`Không tìm thấy bài viết với id ${id}`);

        }

      } catch (error) {

        console.error("Lỗi khi lấy bài viết:", error);

      }

    };

    fetchPost(); // Lấy dữ liệu bài viết cần chỉnh sửa

  }, [id]);

  const handleSubmit = async (e) => {

    e.preventDefault();

    const updatedPost = {

      title,

      content,

    };

    try {

      const response = await axios.put(`http://localhost:3000/posts/${id}`, updatedPost);

      if (response.status === 200) {

        setStatus("Bài viết đã được cập nhật!");

        setTitle("");  // Clear input fields

        setContent("");

        navigate("/");  // Quay lại trang danh sách bài viết sau khi cập nhật

      }

    } catch (error) {

      setStatus("Có lỗi xảy ra khi cập nhật bài viết.");

      console.error("Lỗi khi cập nhật bài viết:", error);

    }

  };

  return (

    <div>

      <h1>Chỉnh sửa bài viết</h1>

      <form onSubmit={handleSubmit}>

        <input

          type="text"

          placeholder="Tiêu đề"

          value={title}

          onChange={(e) => setTitle(e.target.value)}

        />

        <br />

        <br />

        <textarea

          placeholder="Nội dung"

          value={content}

          onChange={(e) => setContent(e.target.value)}

        />

        <br />

        <button type="submit">Cập nhật bài viết</button>

      </form>

      <p>{status}</p>

    </div>

  );

};

export default EditPost;

File DeletePost.js:  
import React from "react";

import axios from "axios";

import { useNavigate, useParams } from "react-router-dom";  // Sử dụng useParams để lấy id và useNavigate để điều hướng

const DeletePost = () => {

  const { id } = useParams();  // Lấy id bài viết từ URL

  const navigate = useNavigate();  // Khởi tạo useNavigate để điều hướng

  // Hàm để xóa bài viết

  const handleDelete = async () => {

    try {

      await axios.delete(`http://localhost:3000/posts/${id}`);

      // Chuyển hướng người dùng về trang danh sách bài viết sau khi xóa thành công

      navigate("/");

    } catch (error) {

      console.error("Lỗi khi xóa bài viết:", error);

    }

  };

  return (

    <div>

      <h1>Xác nhận xóa bài viết</h1>

      <p>Bạn chắc chắn muốn xóa bài viết này?</p>

      <button onClick={handleDelete}>Xóa</button>

      <button onClick={() => navigate("/")}>Hủy</button> {/\* Quay lại trang danh sách bài viết \*/}

    </div>

  );

};

export default DeletePost;

Cập nhật trong file App.js:

import React from "react";

import { BrowserRouter as Router, Route, Routes } from "react-router-dom";

import PostList from "./components/PostList";

import CreatePost from "./components/CreatePost";

import EditPost from "./components/EditPost";

import DeletePost from "./components/DeletePost";

const App = () => {

  return (

    <Router>

      <div>

        <h1>Quản lý Bài Viết</h1>

        <Routes>

          <Route path="/" element={<PostList />} />

          <Route path="/create" element={<CreatePost />} />

          <Route path="/edit/:id" element={<EditPost />} />

          <Route path="/delete/:id" element={<DeletePost />} />

        </Routes>

      </div>

    </Router>

  );

};

export default App;

**Ví dụ 5:**

Dựa vào các ví dụ trên, chỉnh sửa giao diện CRUD cho Post sử dụng react-bootstrap.

Sử dụng PropTypes để validate dữ liệu khi thực hiện Create, Edit Post, Delete Post thì cần hiển thị thông báo xác nhận Xóa.

**Ví dụ 6:**

Chỉnh sửa code ở ví dụ 5, yêu cầu trang home là trang Login:   
+ Login thành công, hiển thị thông báo “Login successfully with username:…. ” và điều hướng đến trang PostList

Ngược lại, hiển thị thông báo lỗi.

Yêu cầu có sử dụng PropTypes để validate form Login trong trường hợp user/name pass để trống.

=>Tham khảo thêm kĩ thuật Lazy loading giúp tải các phần của ứng dụng chỉ khi người dùng cần chúng, thay vì tải toàn bộ ứng dụng ngay khi mở trang