Advanced Web Technologies

React - API

Prepared by: Eng. Nidal Aljuneidi

Install JSON Serve

Install it globally (so you can use it anywhere):

```
npm install -g json-server
```

Or locally (recommended for project-specific use)

```
npm install json-server --save-dev
```

Create a db.json File

```
{
  "posts": [
    { "id": 1, "title": "Hello World" },
    { "id": 2, "title": "React + JSON Server" }
  ]
}
```

Run JSON Serve

```
npx json-server --watch db.json --port 3001
```

Fetch Data in Reac

```
useEffect(() => {

  fetch('http://localhost:3001/posts')

    .then(res => res.json())

    .then(data => console.log(data));

}, []);
```

Add a Script to package.json

```
"scripts": {
  "start": "react-scripts start",
  "server": "json-server --watch db.json --port 3001"
}
```

Sample Dat

```
const posts = [
  { id: 1, title: "Intro to React" },
  { id: 2, title: "Using map() Function" },
  { id: 3, title: "Connecting with JSON Server" }
];
```

```
function PostList() {

  return (

    <div>

      <h2>Course Topics</h2>

      <ul>

        {posts.map(post => (

          <li key={post.id}>{post.title}</li>

        ))}

      </ul>

    </div>

  );

}
```

Create the Custom Hook

```
const useProducts = (url) => {
  const [products, setProducts] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await fetch(url);
        const result = await response.json();
        setProducts(result.products || []);
      } catch (error) {
        console.error("Error fetching products:", error);
      } finally {
        setLoading(false);
      }
    };

    fetchData();
  }, [url]);

  return { products, loading };
};

export default useProducts;
```

Hook in react is all about **clean code**, **reusability**, and **separation of concerns**

```jsx
const Post = () => {
  const { products, loading } = useProducts("https://dummyjson.com/products");

  return (
    <div>
      <h2>Welcome</h2>

      {loading ? (
        <p>Loading products...</p>
      ) :{ original Code}
};

export default Post;
```

Post Manage Example

```
npm install -g json-server
```

Create `db.json`

```
{
  "posts": [
    { "id": 1, "title": "First Post", "body": "Hello World!" },
    { "id": 2, "title": "Second Post", "body": "React is awesome!" }
  ]
}
```

Run JSON Server

```
json-server --watch db.json --port 3001
```

Project Structure

```
src/
├── hooks/
│   └── usePosts.js
├── components/
│   └── PostManager.js
├── App.js
```

Entry Point

```
import PostManager from "./components/PostManager";

function App() {
  return (
    <div className="App">
      <h1>React + JSON Server CRUD</h1>
      <PostManager />
    </div>
  );
}

export default App;
```

Custom Hook: usePosts.js

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

const API_URL = "http://localhost:3001/posts";

export function usePosts() {
  const [posts, setPosts] = useState([]);
  const [loading, setLoading] = useState(true);

  // 🔍 Fetch posts using async/await
  const fetchPosts = async () => {
    try {
      const res = await fetch(API_URL);
      const data = await res.json();
      setPosts(data);
    } catch (err) {
      console.error("Fetch error:", err);
    } finally {
      setLoading(false);
    }
  };

  useEffect(() => {
    fetchPosts();
  }, []);
```

Continue Custom Hook

```javascript
// ➕ Add post using .then()
  const addPost = (newPost) => {
    fetch(API_URL, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(newPost),
    })
      .then((res) => res.json())
      .then((data) => setPosts((prev) => [...prev, data]))
      .catch((err) => console.error("Add error:", err));
  };
```

UPdate Post.

```
const updatePost = async (id, updatedPost) => {
    try {
      const res = await fetch(`${API_URL}/${id}`, {
        method: "PUT",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(updatedPost),
      });
      const data = await res.json();
      setPosts((prev) =>
        prev.map((post) => (post.id === id ? data : post))
      );
    } catch (err) {
      console.error("Update error:", err);
    }
  };
```

INSTRUCTOR

```
  // ❌ Delete post using .then()
  const deletePost = (id) => {
    fetch(`${API_URL}/${id}`, { method: "DELETE" })
      .then(() => setPosts((prev) => prev.filter((post) => post.id !== id)))
      .catch((err) => console.error("Delete error:", err));
  };
 return { posts, loading, addPost, updatePost, deletePost };


}
```

Key	Purpose
`method: "POST"`	Tells the server you're sending data to *create* something new. Could also be `"GET"`, `"PUT"`, `"DELETE"`, etc.
`headers`	Provides metadata. Here we tell the server that we're sending **JSON data**.
`"content-type": "application/json"`	Required so the server knows to parse the incoming data as JSON.
`body: JSON.stringify(newPost)`	This is the **actual data** we're sending. Since the body must be a string, we use `JSON.stringify()` to convert the JavaScript object into JSON format.

setLoading

The `loading` state is used to track whether the app is currently fetching data. You typically use it to show the user **visual feedback**—like a spinner, message, or skeleton UI—while the data is being loaded.

React Router

A library for handling routing in React.

Lets you navigate between components without reloading the page.

Enables Single Page Applications (SPA) behavior.

Why Use React Router?

- ❑ Enables client-side navigation
- ❑ Prevents full page reloads
- ❑ Helps organize your app into multiple views
- ❑ Supports nested and dynamic routes

Installation

```
npm install react-router-dom
```

```
function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
      </Routes>
    </BrowserRouter>
  );
}
```

Pages

```jsx
// Home.jsx
export default function Home() {
  return <h1>Welcome to Home Page</h1>;
}

// About.jsx
export default function About() {
  return <h1>About Us</h1>;
}
```

Navigating with Links

```
import { Link } from 'react-router-dom';


<Link to="/">Home</Link>
<Link to="/about">About</Link>
```

Using useNavigate()

```
import { useNavigate } from 'react-router-dom';

const MyComponent = () => {
  const navigate = useNavigate();

  return <button onClick={() => navigate('/about')}>Go to About</button>;
};
```

Dynamic Routing

```
<Routes>
  <Route path="/user/:id" element={<User />} />
</Routes>
```

```
import { useParams } from 'react-router-dom';

function User() {
  const { id } = useParams();
  return <h2>User ID: {id}</h2>;
}
```

Nested Routes

```
<Routes>
  <Route path="/dashboard" element={<Dashboard />}>
    <Route path="profile" element={<Profile />} />
    <Route path="settings" element={<Settings />} />
  </Route>
</Routes>
```

404 - Not Found Route

```
<Route path="*" element={<NotFound />} />
```

Protected Routes (Auth)

```
function ProtectedRoute({ children }) {
  const isAuth = useAuth(); // Custom hook or context
  return isAuth ? children : <Navigate to="/login" />;
}
//usage
<Route path="/dashboard" element={<ProtectedRoute><Dashboard
/></ProtectedRoute>} />
```

Example

Install React Router

```
npm install react-router-dom
```

index.js

index.jx

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import { BrowserRouter } from 'react-router-dom';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>
);
```

App.jsx

```jsx
import { Routes, Route } from 'react-router-dom';
import Home from './pages/Home';
import About from './pages/About';
import Contact from './pages/Contact';
import Navbar from './components/Navbar';

function App() {
  return (
    <>
      <Navbar />
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/contact" element={<Contact />} />
      </Routes>
    </>
  );
}

export default App;
```

Home.jsx

```jsx
export default function Home() {
  return (
    <div>
      <h1>🏠 Home Page</h1>
      <p>Welcome to our React site!</p>
    </div>
  );
}
```

About.jsx

```
export default function About() {
  return (
    <div>
      <h1>ℹ️ About Page</h1>
      <p>This site is created with React Router.</p>
    </div>
  );
}
```

Contact.jsx

```jsx
export default function Contact() {
  return (
    <div>
      <h1>📞 Contact Page</h1>
      <p>Contact us at contact@example.com.</p>
    </div>
  );
}
```

Navbar.jsx (Navigation using Link)

```jsx
import { Link } from 'react-router-dom';

export default function Navbar() {
  return (
    <nav style={styles.nav}>
      <Link style={styles.nav.link} to="/">Home</Link>
      <Link style={styles.nav.link} to="/about">About</Link>
      <Link style={styles.nav.link} to="/contact">Contact</Link>
    </nav>
  );
}
```

```
const styles = {
  nav: {
    padding: '10px',
    backgroundColor: '#333',
    display: 'flex',
    justifyContent: 'center',
    gap: '20px'
    link: {
          color: 'white',
          textDecoration: 'none',
          fontSize: '18px',
    },

    }


};
```

Introduction to Icons & Emojis in React

Icons and emojis enhance UI/UX

React supports multiple icon libraries

Emojis can be added natively or via packages

Installing React Icons

```
npm install react-icons
```

Using React Icons

```
import { FaBeer } from "react-icons/fa";

function App() {
  return <h3>Cheers! <FaBeer /></h3>;
}

//Import icons from specific libraries
//Use as React components
```

popular React Icon Libraries

Library	`Prefix`	`Example`
Font Awesome	`fa`	`react-icons/fa`
Material Icons	`md`	`react-icons/md`
Bootstrap Icons	`bs`	`react-icons/bs`
Feather Icons	`fi`	`react-icons/fi`

Installing Font Awesome (Official)

```
npm install @fortawesome/fontawesome-svg-core

npm install @fortawesome/free-solid-svg-icons

npm install @fortawesome/react-fontawesome
```

Using Font Awesome in React

```
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { faCoffee } from "@fortawesome/free-solid-svg-icons";

function App() {
  return <FontAwesomeIcon icon={faCoffee} />;
}
```

//Use FontAwesomeIcon component

//Import icons individually

Emoji Libraries

```
emoji-dictionary

emoji-picker-react

react-emoji-render
```

```
npm install emoji-picker-react
```

```
import EmojiPicker from "emoji-picker-react";

function App() {
  return <EmojiPicker />;
}
```

Forms & Input Handling in React

Why Use Forms in Web Development?

- Collect user input (e.g., login, registration)
- Allow users to interact with your app
- Submit data to backend services
- Enable customization and preferences

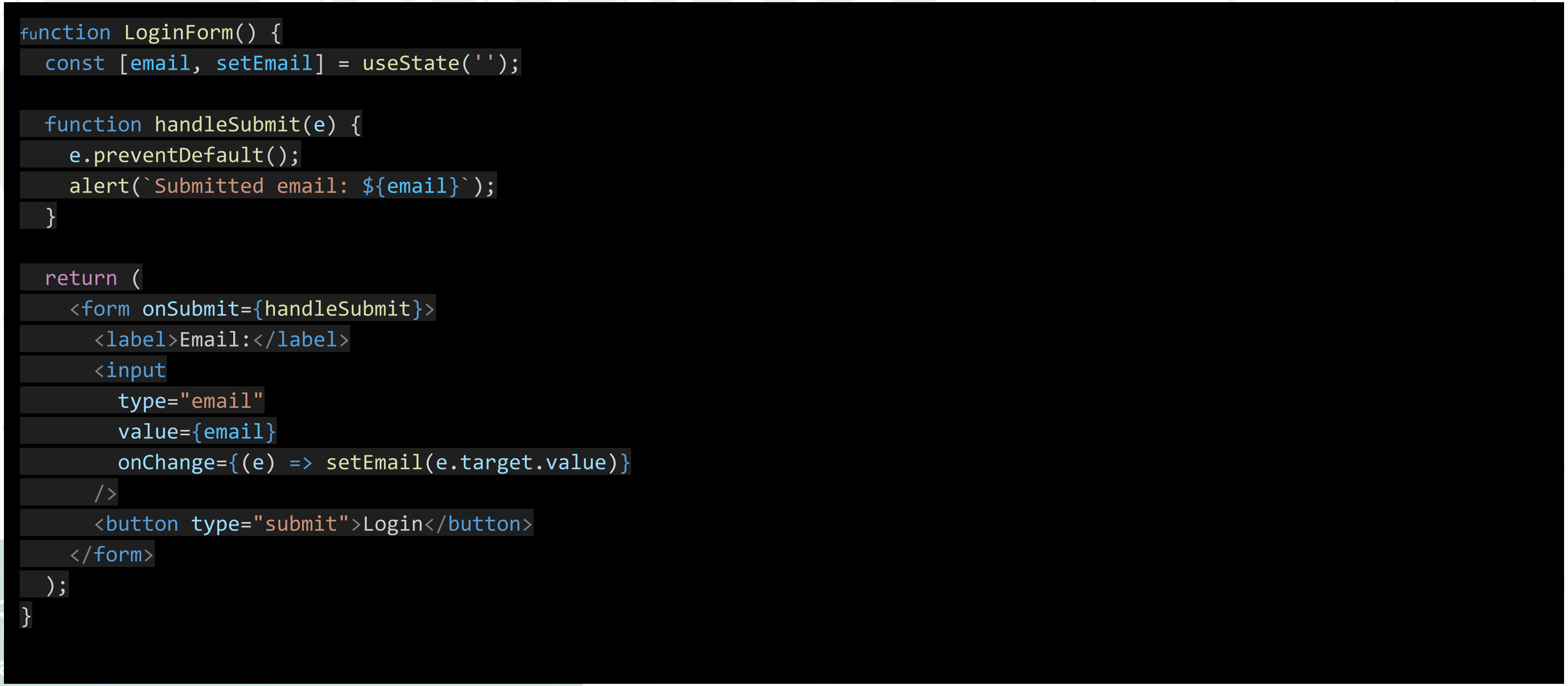
HTML vs. React Form Handling

HTML	REACT
Uses built-in form behavior	Controlled by React state
Values stored in DOM	Values stored in component state
Uncontrolled inputs	Mostly controlled components

Controlled Components

- The form element's value is controlled by React state
- `value` attribute is linked to state
- `onChange` updates the state

```
function TextInput({ value,
onChange }) {
  return (
    <input
      type="text"
      value={value}
      onChange={onChange}
    />
  );
}
```



```jsx
function LoginForm() {
  const [email, setEmail] = useState('');

  function handleSubmit(e) {
    e.preventDefault();
    alert(`Submitted email: ${email}`);
  }

  return (
    <form onSubmit={handleSubmit}>
      <label>Email:</label>
      <input
        type="email"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
      />
      <button type="submit">Login</button>
    </form>
  );
}
```

Handling Multiple Inputs

```
const [formData, setFormData] = useState({
  name: '',
  email: ''
});

function handleChange(e) {
  setFormData({
    ...formData,
    [e.target.name]: e.target.value
  });
}
```