**Hands-on: 17. ReactJS-HOL**

**Introduction**

Modern web applications often need to fetch, display, or submit data to external servers. REST APIs (Representational State Transfer Application Programming Interfaces) are widely used for this purpose. React makes it easy to consume REST APIs using built-in browser features like fetch() or third-party libraries like Axios. This section explains how to call and handle REST API data in a React application.

**1. How to Consume REST APIs from React Applications**

1. Using the fetch() API

The fetch() method is a native JavaScript function used to make HTTP requests.

* Example: GET request using fetch

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

function UserList() {

const [users, setUsers] = useState([]);

useEffect(() => {

fetch('https://jsonplaceholder.typicode.com/users') // sample API

.then(response => response.json())

.then(data => setUsers(data))

.catch(error => console.error('Error:', error));

}, []);

return (

<ul>

{users.map(user => <li key={user.id}>{user.name}</li>)}

</ul>

);

}

* Explanation:
* useEffect() is used to make the API call when the component mounts.
* fetch() sends a GET request.
* The response is converted to JSON and stored in state (setUsers).
* The result is rendered using map().

2. Using Axios (Third-party Library)

Axios is a promise-based HTTP client that simplifies API calls.

npm install axios

* Example with Axios:

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

import axios from 'axios';

function PostList() {

const [posts, setPosts] = useState([]);

useEffect(() => {

axios.get('https://jsonplaceholder.typicode.com/posts')

.then(response => setPosts(response.data))

.catch(error => console.log(error));

}, []);

return (

<div>

{posts.slice(0, 5).map(post => (

<div key={post.id}>

<h3>{post.title}</h3>

<p>{post.body}</p>

</div>

))}

</div>

);

}

3. Handling POST Requests

Use fetch() or axios.post() to send data to a server.

function createPost() {

fetch('https://jsonplaceholder.typicode.com/posts', {

method: 'POST',

headers: { 'Content-type': 'application/json' },

body: JSON.stringify({ title: 'New Post', body: 'This is content' })

})

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

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

}

Or with Axios:

axios.post('https://jsonplaceholder.typicode.com/posts', {

title: 'New Post',

body: 'This is content'

});

* Best Practices
* Use useEffect to avoid infinite loops when making API calls.
* Always handle errors using .catch() or try...catch.
* Show loading states and fallback UIs while waiting for API responses.
* Use async/await for cleaner asynchronous code.

**Conclusion**

Consuming REST APIs in React is a vital skill for building dynamic, data-driven applications. Whether using the native fetch() method or the popular Axios library, React provides the tools and flexibility to integrate external data sources easily. Proper handling of API calls—including state updates, error management, and loading indicators—ensures a responsive and reliable user experience.