React

1. ReactJS-HOL
2. Define SPA and Its Benefits

SPA (Single Page Application) is a web application that interacts with the user by dynamically rewriting the current page, rather than loading entire new pages from a server.

Benefits of SPA:

Fast loading after initial load

Better user experience (no full page reload)

Efficient front-end routing

Works well with mobile apps

2. Define React and Identify Its Working

React is a JavaScript library developed by Facebook for building user interfaces. It is component-based and follows a declarative programming model.

How React Works:

Breaks UI into reusable components

Uses a virtual DOM for performance

Updates only the necessary parts of the UI

3. Differences Between SPA and MPA

| Feature | SPA | MPA |
| --- | --- | --- |
| Page Load | Loads a single HTML file | Loads a new HTML file per page |
| Speed | Faster after initial load | Slower, more page reloads |
| Navigation | Client-side routing (JS) | Server-side routing |
| SEO | Challenging without SSR | Better SEO |

4. Pros & Cons of SPA

Pros:

Faster user experience

Reduced server load

Rich interactivity

Cons:

SEO limitations

Initial loading can be slow

Can be complex to maintain

5. Explain About React

React allows developers to build web applications that can update and render efficiently in response to data changes. It emphasizes reusable components and one-way data binding.

6. Define Virtual DOM

Virtual DOM is a lightweight JavaScript representation of the actual DOM. React uses it to:

Detect changes efficiently

Batch and apply updates to the real DOM

Improve performance

7. Features of React

JSX (JavaScript XML)

Component-based architecture

Virtual DOM

One-way data binding

Fast rendering

Rich ecosystem with tools like React Router, Redux

App.js

import React from 'react';

function App() {

return (

<div>

<h1>Welcome to the first session of React</h1>

</div>

);

}

export default App;

Run the React App:

bash

CopyEdit

npm start

1. ReactJS-HOL

1. Explain React Components

React components are reusable, independent pieces of UI. They allow you to split the UI into smaller, manageable parts. Components can be either class-based or function-based.

2. Difference Between Components and JavaScript Functions

| Feature | JavaScript Functions | React Components |
| --- | --- | --- |
| Purpose | Perform logic/calculation | Return JSX to render UI |
| Return Value | Any JS data type | Must return JSX or null |
| Lifecycle Methods | Not available | Available (in class components) |
| Reusability in UI | Not inherently reusable as UI | Designed to be reusable UI pieces |

3. Types of Components

Class Component: Uses ES6 class syntax and supports lifecycle methods.

Function Component: Uses JavaScript functions and can use Hooks for state/lifecycle.

4. Explain Class Component

A class component extends React.Component and must include a render() method.

jsx

CopyEdit

import React, { Component } from 'react';

class Home extends Component {

render() {

return <h1>Welcome to the Home page</h1>;

}

}

5. Explain Function Component

A function component is a JavaScript function that returns JSX.

jsx

CopyEdit

function Home() {

return <h1>Welcome to the Home page</h1>;

}

6. Define Component Constructor

Used in class components to initialize state or bind methods:

jsx

CopyEdit

constructor(props) {

super(props);

this.state = { ... };

}

7. Define render() Function

The render() method is mandatory in class components and returns JSX to be displayed in the UI.

Home.js

import React, { Component } from 'react';

class Home extends Component {

render() {

return (

<div>

<h2>Welcome to the Home page of Student Management Portal</h2>

</div>

);

}

}

export default Home;

About.js

import React, { Component } from 'react';

class About extends Component {

render() {

return (

<div>

<h2>Welcome to the About page of the Student Management Portal</h2>

</div>

);

}

}

export default About;

Contact.js

import React, { Component } from 'react';

class Contact extends Component {

render() {

return (

<div>

<h2>Welcome to the Contact page of the Student Management Portal</h2>

</div>

);

}

}

export default Contact;

App.js

import React from 'react';

import Home from './Components/Home';

import About from './Components/About';

import Contact from './Components/Contact';

function App() {

return (

<div className="App">

<Home />

<About />

<Contact />

</div>

);

}

export default App;

1. ReactJS-HOL

1. Explain React Components

React components are independent, reusable UI elements. Each component can manage its own state and logic and can be either a class or function component.

2. Differences Between Components and JavaScript Functions

| Feature | JavaScript Functions | React Components |
| --- | --- | --- |
| Purpose | Perform logic or return data | Render UI using JSX |
| JSX Return | Not designed for JSX | Returns JSX for UI |
| Reusability in UI | Not inherently for UI reuse | Meant for UI reuse |
| State/Props Usage | Not applicable | Accepts props, can manage state |

3. Types of Components

Class Components – Use class syntax and have lifecycle methods.

Function Components – Use JavaScript functions, and can use Hooks like useState, useEffect.

4. Explain Class Component

A React component defined using a class and extending React.Component. Must include render() method.

5. Explain Function Component

A simpler, preferred way to define components using a plain function that returns JSX.

6. Define Component Constructor

Used in class components to initialize state and bind methods. Not required in function components.

7. Define render() Function

In class components, render() returns the JSX to be displayed. Not used in function components.

CalculateScore.js

import React from 'react';

import '../Stylesheets/mystyle.css';

function CalculateScore(props) {

const { name, school, total, goal } = props;

const average = total / goal;

return (

<div className="score-card">

<h2>Student Score Details</h2>

<p><strong>Name:</strong> {name}</p>

<p><strong>School:</strong> {school}</p>

<p><strong>Total Score:</strong> {total}</p>

<p><strong>Goal:</strong> {goal}</p>

<p><strong>Average Score:</strong> {average.toFixed(2)}</p>

</div>

);

}

export default CalculateScore;

mystyle.css

.score-card {

background-color: #f9f9f9;

border: 2px solid #4CAF50;

border-radius: 10px;

padding: 20px;

margin: 30px auto;

width: 60%;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

}

.score-card h2 {

color: #4CAF50;

text-align: center;

}

.score-card p {

font-size: 16px;

line-height: 1.6;

}

App.js

import React from 'react';

import './App.css';

import CalculateScore from './Components/CalculateScore';

function App() {

return (

<div className="App">

<CalculateScore name="Alice Smith" school="Greenwood High" total={450} goal={5} />

</div>

);

}

export default App;

1. ReactJS-HOL

1. Explain the Need and Benefits of Component Lifecycle

React components go through a lifecycle from creation to destruction. Understanding this allows developers to:

Perform operations at specific stages (e.g., fetching data after mount)

Improve performance and avoid memory leaks

Handle errors gracefully

2. Identify Various Lifecycle Hook Methods

| Phase | Lifecycle Method | Purpose |
| --- | --- | --- |
| Mounting | constructor() | Initialize state and bind methods |
|  | render() | Renders JSX to the DOM |
|  | componentDidMount() | Called once after first render |
| Updating | shouldComponentUpdate() | Decide whether to re-render |
|  | componentDidUpdate() | Runs after every update |
| Unmounting | componentWillUnmount() | Clean-up tasks before component is destroyed |
| Error Handling | componentDidCatch() | Catch JavaScript errors in child components |

3. Sequence of Steps in Rendering a Component

Constructor is called

Render is called

DOM is updated

componentDidMount() is called

If state changes, render runs again

On errors in child components, componentDidCatch() is called

Post.js

// src/Post.js

import React from 'react';

function Post({ title, body }) {

return (

<div className="post">

<h3>{title}</h3>

<p>{body}</p>

</div>

);

}

export default Post;

Posts.js

// src/Posts.js

import React, { Component } from 'react';

import Post from './Post';

class Posts extends Component {

constructor(props) {

super(props);

this.state = {

posts: [],

hasError: false

};

}

loadPosts = () => {

fetch('https://jsonplaceholder.typicode.com/posts')

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

.then(data => this.setState({ posts: data }))

.catch(error => {

console.error("Fetch Error:", error);

this.setState({ hasError: true });

});

};

componentDidMount() {

this.loadPosts();

}

componentDidCatch(error, info) {

alert("An error occurred in the Posts component.");

console.error("Error caught in component:", error, info);

}

render() {

const { posts, hasError } = this.state;

if (hasError) {

return <h2>Something went wrong. Please try again later.</h2>;

}

return (

<div>

<h2>Blog Posts</h2>

{posts.map(post => (

<Post key={post.id} title={post.title} body={post.body} />

))}

</div>

);

}

}

export default Posts;

App.js

// src/App.js

import React from 'react';

import './App.css';

import Posts from './Posts';

function App() {

return (

<div className="App">

<Posts />

</div>

);

}

export default App;

1. ReactJS-HOL

CohortDetails.module.css

/\* CohortDetails.module.css \*/

.box {

width: 300px;

display: inline-block;

margin: 10px;

padding: 10px 20px;

border: 1px solid black;

border-radius: 10px;

}

dt {

font-weight: 500;

}

CohortDetails.js

import React from 'react';

import styles from './CohortDetails.module.css';

const CohortDetails = ({ name, status, startDate, endDate }) => {

const titleColor = status.toLowerCase() === 'ongoing' ? 'green' : 'blue';

return (

<div className={styles.box}>

<h3 style={{ color: titleColor }}>{name}</h3>

<dl>

<dt>Status:</dt>

<dd>{status}</dd>

<dt>Start Date:</dt>

<dd>{startDate}</dd>

<dt>End Date:</dt>

<dd>{endDate}</dd>

</dl>

</div>

);

};

export default CohortDetails;

1. ReactJS-HOL

Trainer.js

// src/Trainer.js

class Trainer {

constructor(TrainerId, Name, Email, Phone, Technology, Skills) {

this.TrainerId = TrainerId;

this.Name = Name;

this.Email = Email;

this.Phone = Phone;

this.Technology = Technology;

this.Skills = Skills;

}

}

export default Trainer;

TrainersMock.js

// src/TrainersMock.js

import Trainer from './Trainer';

const trainers = [

new Trainer(1, 'John Doe', 'john.doe@example.com', '123-456-7890', 'React', ['JavaScript', 'React']),

new Trainer(2, 'Jane Smith', 'jane.smith@example.com', '234-567-8901', 'Node.js', ['JavaScript', 'Node.js']),

new Trainer(3, 'Emily Johnson', 'emily.johnson@example.com', '345-678-9012', 'Python', ['Django', 'Flask'])

];

export default trainers;

TrainersList.js

// src/TrainersList.js

import React from 'react';

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

const TrainersList = ({ trainers }) => {

return (

<div>

<h2>Trainers List</h2>

<ul>

{trainers.map(trainer => (

<li key={trainer.TrainerId}>

<Link to={`/trainer/${trainer.TrainerId}`}>{trainer.Name}</Link>

</li>

))}

</ul>

</div>

);

};

export default TrainersList;

Home.js

// src/Home.js

import React from 'react';

const Home = () => {

return (

<div>

<h2>Welcome to the Trainers App</h2>

<p>Select a trainer to view details.</p>

</div>

);

};

export default Home;

TrainerDetails.js

// src/TrainerDetails.js

import React from 'react';

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

import trainers from './TrainersMock';

const TrainerDetails = () => {

const { id } = useParams();

const trainer = trainers.find(t => t.TrainerId === parseInt(id));

return (

<div>

<h3>Trainer Details</h3>

{trainer ? (

<div>

<p><strong>Name:</strong> {trainer.Name}</p>

<p><strong>Email:</strong> {trainer.Email}</p>

<p><strong>Phone:</strong> {trainer.Phone}</p>

<p><strong>Technology:</strong> {trainer.Technology}</p>

<p><strong>Skills:</strong> {trainer.Skills.join(', ')}</p>

</div>

) : (

<p>Trainer not found.</p>

)}

</div>

);

};

export default TrainerDetails;

App.js

// src/App.js

import React from 'react';

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

import Home from './Home';

import TrainersList from './TrainersList';

import TrainerDetails from './TrainerDetails';

import trainers from './TrainersMock';

function App() {

return (

<Router>

<div>

<nav>

<Link to="/">Home</Link> |

<Link to="/trainers"> Trainers List</Link>

</nav>

<Routes>

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

<Route path="/trainers" element={<TrainersList trainers={trainers} />} />

<Route path="/trainer/:id" element={<TrainerDetails />} />

</Routes>

</div>

</Router>

);

}

export default App;

1. ReactJS-HOL

Cart.js

// src/Cart.js

import React from 'react';

class Cart extends React.Component {

render() {

const { Itemname, Price } = this.props;

return (

<div>

<h3>{Itemname}</h3>

<p>Price: ${Price}</p>

</div>

);

}

}

export default Cart;

OnlineShopping.js

// src/OnlineShopping.js

import React from 'react';

import Cart from './Cart';

class OnlineShopping extends React.Component {

// Initialize the state with an array of cart items

constructor(props) {

super(props);

this.state = {

items: [

{ Itemname: 'Laptop', Price: 799.99 },

{ Itemname: 'Smartphone', Price: 499.99 },

{ Itemname: 'Headphones', Price: 199.99 },

{ Itemname: 'Mouse', Price: 29.99 },

{ Itemname: 'Keyboard', Price: 49.99 }

]

};

}

render() {

return (

<div>

<h1>Online Shopping</h1>

<div>

{this.state.items.map((item, index) => (

<Cart key={index} Itemname={item.Itemname} Price={item.Price} />

))}

</div>

</div>

);

}

}

export default OnlineShopping;

index.js

// src/index.js

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import OnlineShopping from './OnlineShopping'; // Import the OnlineShopping component

ReactDOM.render(

<React.StrictMode>

<OnlineShopping /> {/\* Render the OnlineShopping component \*/}

</React.StrictMode>,

document.getElementById('root')

);

1. ReactJS-HOL

CountPeople

// src/CountPeople.js

import React from 'react';

class CountPeople extends React.Component {

// Step 1: Define the constructor and initialize the state

constructor(props) {

super(props);

this.state = {

entryCount: 0,

exitCount: 0

};

// Binding methods to the component context

this.updateEntry = this.updateEntry.bind(this);

this.updateExit = this.updateExit.bind(this);

}

// Step 2: Create the method to increment the entry count

updateEntry() {

this.setState({

entryCount: this.state.entryCount + 1

});

}

// Step 3: Create the method to increment the exit count

updateExit() {

this.setState({

exitCount: this.state.exitCount + 1

});

}

render() {

return (

<div>

<h1>Mall People Counter</h1>

<div>

<p>People Entered: {this.state.entryCount}</p>

<button onClick={this.updateEntry}>Login</button>

</div>

<div>

<p>People Exited: {this.state.exitCount}</p>

<button onClick={this.updateExit}>Exit</button>

</div>

</div>

);

}

}

export default CountPeople;

App.js

// src/App.js

import React from 'react';

import './App.css';

import CountPeople from './CountPeople';

function App() {

return (

<div className="App">

<CountPeople />

</div>

);

}

export default App;