**Superset-Id: 6364597  
Name: SRINJOY PAUL**

**Week-7 REACT HANDSON**

**OBJECTIVES:**

1. **List the features of ES6  
   Answer:** ES6 (ECMAScript 2015) introduced several powerful features to JavaScript. Key features include:

* let and const for block-scoped variable declarations
* Arrow functions (=>)
* Classes and class inheritance
* Template literals (`Hello, ${name}`)
* Default parameters
* Destructuring assignment
* Spread and rest operators (...)
* Promises for asynchronous operations
* Modules (import, export)
* Map and Set data structures
* Enhanced object literals
* Symbols
* Generators

1. **Explain JavaScript let  
   Answer:**The let keyword declares a block-scoped variable, meaning it is only accessible within the block {} in which it's defined.  
   let x = 10;

if (true) {

let x = 20;  
 console.log(x);

console.log(x);

* **let prevents redeclaration within the same scope.**
* **It is hoisted but not initialized, so accessing it before declaration results in a ReferenceError.**

1. **Identify the differences between var and let  
   Answer:**

**Feature var let**

|  |  |  |
| --- | --- | --- |
| Scope | Function-scoped | Block-scoped |
| Hoisting | Hoisted and initialized | Hoisted but not initialized |
| Redeclaration | Allowed | Not allowed in same scope |
| Temporal Dead Zone | No | Yes |

1. **Explain JavaScript const  
   Answer:**

**const declares block-scoped constants.**

**The variable must be initialized at the time of declaration.**

The binding cannot be reassigned, but if it's an object or array, its contents can still be modified.  
const PI = 3.14;  
const obj = { a: 1 };

obj.a = 2;

1. **Explain ES6 class fundamentals  
   Answer:**  
   ES6 introduces a more intuitive syntax for working with classes and OOP.

class Person {

constructor(name) {

this.name = name;

}

greet() {

return `Hello, ${this.name}`;

}

}

Key points:

* **constructor() is a special method for object creation.**
* **Methods do not require the function keyword.**
* **Syntax is cleaner and similar to other OOP languages.**

1. **Explain ES6 class inheritance  
   Answer:**ES6 supports inheritance using the extends and super keywords.

class Animal {

constructor(name) {

this.name = name;

}

speak() {

console.log(`${this.name} makes a sound.`);

}

}

class Dog extends Animal {

speak() {

console.log(`${this.name} barks.`);

}

}

const d = new Dog('Rex');

d.speak();

**extends creates a subclass.**

**super() calls the parent class constructor.**

1. **Define ES6 arrow functions  
   Answer:**Arrow functions provide a shorter syntax for function expressions and do not bind their own this.  
   const add = (a, b) => a + b;

console.log(add(2, 3)); // 5

**Key traits:**

* **No this, arguments, super, or new.target binding.**
* **Useful for callbacks and concise one-liners.**
* **Cannot be used as constructors.**

1. **Identify set(), map()  
   Answer:**Set is a collection of unique values.  
     
   const mySet = new Set([1, 2, 2, 3]);

console.log(mySet); // Set(3) {1, 2, 3}

Map is a collection of key-value pairs, where keys can be of any type.  
const myMap = new Map();

myMap.set('a', 1);

myMap.set(2, 'b');

console.log(myMap);

1. **Create a React Application named “cricketapp” with the following components:**
2. **ListofPlayers**

* **Declare an array with 11 players and store details of their names and scores using the map feature of ES6**
* **Filter the players with scores below 70 using arrow functions of ES6.**

1. **IndianPlayers**
   1. **Display the Odd Team Player and Even Team players using the Destructuring features of ES6**
   2. **Declare two arrays T20players and RanjiTrophy players and merge the two arrays and display them using the Merge feature of ES6**

**Answer:   
ListofPlayers.js**import React from "react";

function ListofPlayers({ players }) {

return (

<div>

<h1>List of Players</h1>

<ul>

{players.map((player, index) => (

<li key={index}>

Mr. {player.name} {player.score}

</li>

))}

</ul>

</div>

);

}

export default ListofPlayers;

**ScoreBelow70.js**

import React from "react";

function ScoreBelow70({ players }) {

const below70 = players.filter((p) => p.score < 70);

return (

<div>

<h1>List of Players having Scores Less than 70</h1>

<ul>

{below70.map((player, index) => (

<li key={index}>

Mr. {player.name} {player.score}

</li>

))}

</ul>

</div>

);

}

export default ScoreBelow70;  
**IndianPlayers.js**import React from "react";

function IndianPlayers({ IndianPlayers }) {

return (

<div>

<h1>List of Indian Players Merged:</h1>

<ul>

{IndianPlayers.map((player, index) => (

<li key={index}>Mr. {player}</li>

))}

</ul>

</div>

);

}

export default IndianPlayers;

**App.js**import React from "react";

import ListofPlayers from "./ListofPlayers";

import ScoreBelow70 from "./ScoreBelow70";

import IndianPlayers from "./IndianPlayers";

function App() {

const players = [

{ name: "Jack", score: 55 },

{ name: "Michael", score: 72 },

{ name: "John", score: 48 },

{ name: "Ann", score: 69 },

{ name: "Elisabeth", score: 75 },

{ name: "Sachin", score: 102 },

{ name: "Dhoni", score: 95 },

{ name: "Virat", score: 88 },

{ name: "Jadeja", score: 65 },

{ name: "Raina", score: 79 },

{ name: "Rohit", score: 82 }

];

const IndianTeam = [

"Sachin1",

"Dhoni2",

"Virat3",

"Rohit4",

"Yuvraj5",

"Raina6"

];

const oddPlayers = IndianTeam.filter((\_, i) => i % 2 === 0);

const evenPlayers = IndianTeam.filter((\_, i) => i % 2 !== 0);

const T20players = ["First Player", "Second Player", "Third Player"];

const RanjiPlayers = ["Fourth Player", "Fifth Player", "Sixth Player"];

const allIndianPlayers = [...T20players, ...RanjiPlayers];

if (flag) {

return (

<div>

<ListofPlayers players={players} />

<hr />

<ScoreBelow70 players={players} />

</div>

);

} else {

return (

<div>

<h1>Indian Team</h1>

<h2>Odd Players</h2>

<ul>

{oddPlayers.map((p, i) => (

<li key={i}>{p}</li>

))}

</ul>

<h2>Even Players</h2>

<ul>

{evenPlayers.map((p, i) => (

<li key={i}>{p}</li>

))}

</ul>

<hr />

<IndianPlayers IndianPlayers={allIndianPlayers} />

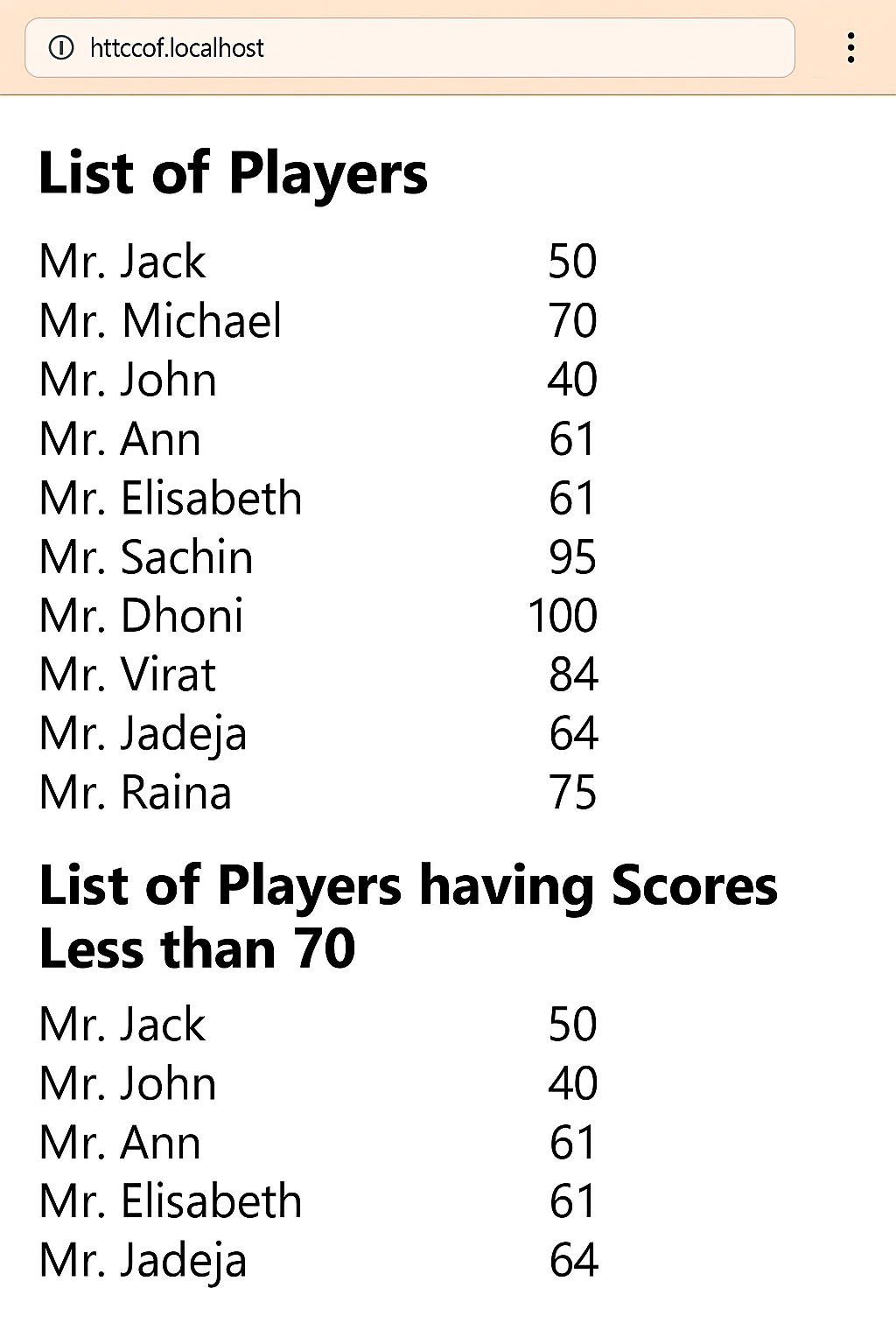
</div>

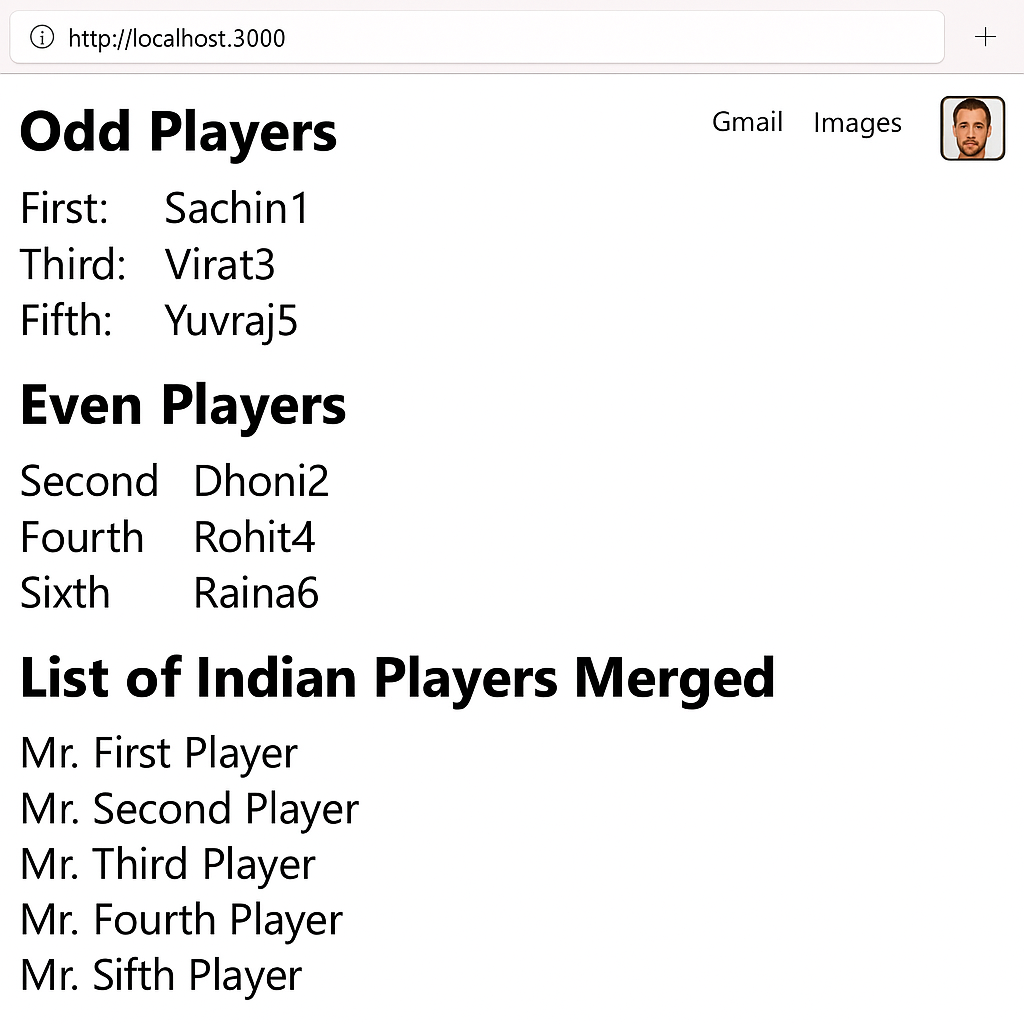
);

}

}

export default App;

**Output:  
**

****

**OBJECTIVES:  
1) Define JSX  
Answer:**JSX (JavaScript XML) is a syntax extension for JavaScript used in React to describe what the UI should look like. It allows you to write HTML-like code within JavaScript, making it easier to create and visualize UI structures. JSX is transpiled into regular JavaScript using tools like Babel.

**2) Explain about ECMA Script  
Answer:**ECMAScript (ES) is the standard specification on which JavaScript is based. It defines the core features of the language. Modern JavaScript introduces features such as **let/const,** arrow functions, classes, template literals, promises, modules, and more that are used extensively in React applications.

**3) Explain React.createElement()  
Answer:  
React.createElement()** is a method used by React to create a virtual DOM element. It's what JSX gets compiled into. It takes three arguments:  
**React.createElement(type, props, children)  
For example:  
React.createElement('h1', { className: 'title' }, 'Hello World')**

**4) Explain how to create React nodes with JSX  
Answer:**To create a React node with JSX, simply write the HTML-like syntax inside a React component. For example:  
const element = <h1>Hello, React!</h1>;

This element is a React node and can be rendered to the DOM.

**5) Define how to render JSX to DOM  
Answer:**

**We can render JSX to the DOM using ReactDOM.render() in a React application. Example:**const element = <h1>Welcome</h1>;

**ReactDOM.render(element, document.getElementById('root'));**

This renders the **element** inside the DOM element with id **"root"**.

**6) Explain how to use JavaScript expressions in JSX  
Answer:**We can include JavaScript expressions inside JSX using curly braces {}. Example:  
const name = "Srinjoy";

const element = <h1>Hello, {name}!</h1>;

Here, {name} is evaluated and inserted into the JSX.

**7) Explain how to use inline CSS in JSX  
Answer:**In JSX, inline CSS is written as a JavaScript object inside the style attribute. CSS property names use camelCase. Example:  
  
const styleObj = { color: 'blue', fontSize: '20px' };

const element = <h1 style={styleObj}>Styled Text</h1>;

**2)Create a React Application named “officespacerentalapp” which uses React JSX to create elements, attributes and renders DOM to display the page.**

**Create an element to display the heading of the page.**

**Attribute to display the image of the office space**

**Create an object of office to display the details like Name, Rent and Address.**

**Create a list of Object and loop through the office space item to display more data.**

**To apply Css, Display the color of the Rent in Red if it’s below 60000 and in Green if it’s above 60000.  
  
  
  
  
  
Answer:  
index.js**  
import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);  
  
**App.js:**

import React from 'react';

import './App.css';

function App() {

const element = "Office Space";

const jsxatt = <img src="office.jpg" width="25%" height="25%" alt="Office Space" />;

const officeList = [

{ Name: "DBS", Rent: 50000, Address: "Chennai" },

{ Name: "WeWork", Rent: 65000, Address: "Bangalore" },

{ Name: "Innov8", Rent: 45000, Address: "Hyderabad" },

{ Name: "Regus", Rent: 75000, Address: "Mumbai" }

];

return (

<div>

<h1>{element} , at Affordable Range</h1>

{officeList.map((item, index) => {

const colorClass = item.Rent <= 60000 ? 'textRed' : 'textGreen';

return (

<div key={index} className="office-card">

{jsxatt}

<h2>Name: {item.Name}</h2>

<h3 className={colorClass}>Rent: Rs. {item.Rent}</h3>

<h3>Address: {item.Address}</h3>

<hr />

</div>

);

})}

</div>

);

}

export default App;

**App.css:**.textRed {

color: red;

}

.textGreen {

color: green;

}

body {

font-family: Arial, sans-serif;

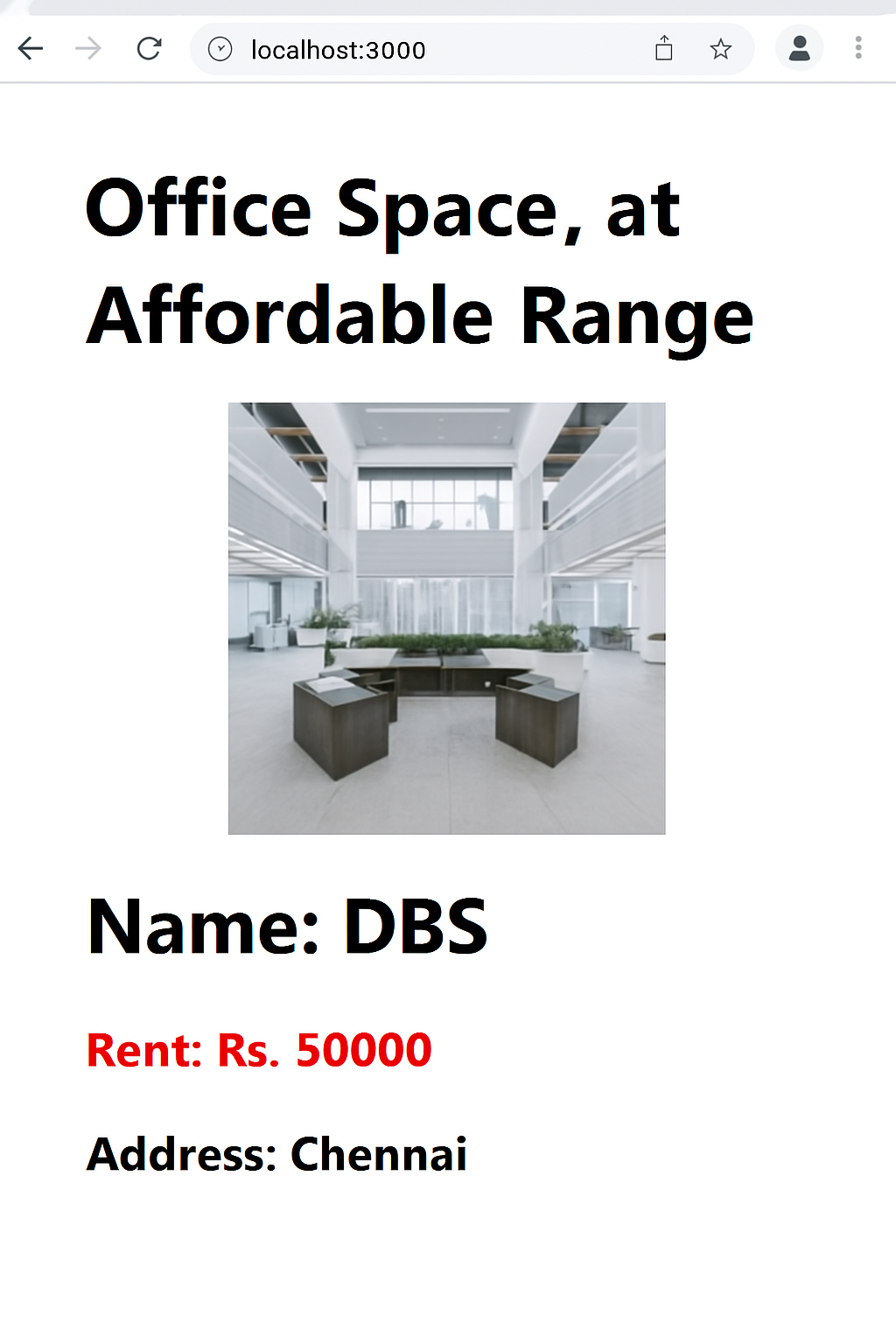
padding: 20px;

}

.office-card {

margin-bottom: 30px;

}

**Output:  
**

**OBJECTIVES:  
1) Explain React events  
Answer:**React events are the way to handle user interactions in a React application, such as clicks, form submissions, typing, mouse movements, etc. They are very similar to DOM events but follow a consistent and cross-browser React-specific system built on top of the standard DOM event system.  
Example:  
<button onClick={handleClick}>Click Me</button>

1. **Explain about event handlers  
   Answer:**Event handlers are functions that are triggered in response to user interactions. In React, event handlers are passed as props to elements using camelCase event names. They define what should happen when a specific event occurs.  
   Example:  
   function handleClick() {   
   alert("Button clicked!"); }  
   Used in JSX:  
   <button onClick={handleClick}>Click</button>
2. **Define Synthetic event  
   Answer:**A SyntheticEvent is a cross-browser wrapper around the browser’s native event. It is part of React’s event system and ensures that events work consistently across all browsers. React’s SyntheticEvent wraps the native event, providing the same interface and improving performance via event pooling.  
   Example:  
   function handleInputChange(e) {

console.log(e.target.value); // 'e' is a SyntheticEvent

}

1. **Identify React event naming convention  
   Answer:**In React, event names follow a camelCase naming convention instead of lowercase (as in HTML).

| **HTML** | **React** |
| --- | --- |
| onclick | onClick |
| onchange | onChange |
| onsubmit | onSubmit |
| onmouseover | onMouseOver |

For example:

<input onChange={handleChange} />

1. **Create a React Application “eventexamplesapp” to handle various events of the form elements in HTML.**
2. **Create “Increment” button to increase the value of the counter and “Decrement” button to decrease the value of the counter. The “Increase” button should invoke multiple methods.** 
   1. **To increment the value**
   2. **Say Hello followed by a static message.**
3. **Create a button “Say Welcome” which invokes the function which takes “welcome” as an argument.**
4. **Create a button which invokes synthetic event “OnPress” which display “I was clicked”**

**Create a “CurrencyConvertor” component which will convert the Indian Rupees to Euro when the Convert button is clicked.**

**Handle the Click event of the button to invoke the handleSubmit event and handle the conversion of the euro to rupees.**

**ANSWER:  
index.js**import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);  
  
**App.js**import React, { useState } from 'react';

import CurrencyConvertor from './CurrencyConvertor';

import './App.css';

function App() {

const [count, setCount] = useState(5);

const increment = () => {

setCount(prev => prev + 1);

sayHello();

};

const sayHello = () => {

alert("Hello Member1");

};

const decrement = () => {

setCount(prev => prev - 1);

};

const sayWelcome = (message) => {

alert(message);

};

const handleClick = () => {

alert("I was clicked");

};

return (

<div>

<p>{count}</p>

<button onClick={increment}>Increment</button>

<button onClick={decrement}>Decrement</button>

<br /><br />

<button onClick={() => sayWelcome("welcome")}>Say welcome</button>

<br /><br />

<button onClick={handleClick}>Click on me</button>

<br /><br />

<CurrencyConvertor />

</div>

);

}

export default App;

**CurrencyConvertor.js**import React, { useState } from 'react';

function CurrencyConvortor() {

const [amount, setAmount] = useState('');

const [currency, setCurrency] = useState('');

const handleSubmit = (e) => {

e.preventDefault();

if (currency.toLowerCase() === 'euro') {

const converted = amount \* 80;

alert(`Converting to Euro Amount is ${converted}`);

} else {

alert("Currency not supported.");

}

};

return (

<div>

<h2 className="currency-header">Currency Convertor!!!</h2>

<form onSubmit={handleSubmit}>

<label>Amount: </label>

<input

type="text"

value={amount}

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

/><br /><br />

<label>Currency: </label>

<input

type="text"

value={currency}

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

/><br /><br />

<button type="submit">Submit</button>

</form>

</div>

);

}

export default CurrencyConvertor;

**App.css**.currency-header {

color: green;

font-size: 24px;

font-weight: bold;

}

**Output:**






**OBJECTIVES:**

**1)Explain about conditional rendering in React  
Answer:**Conditional rendering in React means displaying different UI elements or components based on certain conditions (like state or props). React uses standard JavaScript control structures such as **if, ternary operators, or &&** to conditionally render elements.

Example using ternary:  
{isLoggedIn ? <p>Welcome!</p> : <p>Please log in.</p>}

Example using && operator:  
{hasNotifications && <p>You have new messages</p>}

**2) Define element variables  
Answer:**Element variables are variables used to store JSX elements. You can create them using **const, let,** or **var**, and then conditionally assign JSX to them before including them in the return statement of a component.

Example:  
let message;

if (isLoggedIn) {

message = <h1>Welcome back!</h1>;

} else {

message = <h1>Please sign in.</h1>;

}

return <div>{message}</div>;

**3) Explain how to prevent components from rendering  
Answer:**To prevent a component from rendering, you can return **null** from the component’s **render()** method (in class components) or from the function body (in function components). Returning **null** tells React to render nothing for that component.  
Example:  
function WarningBanner({ show }) {

if (!show) {

return null;

}

return <div className="warning">Warning!</div>;

}  
This is useful when you want to hide a component completely without removing it from the component tree or unmounting it manually.

1. **Create a React Application named “ticketbookingapp” where the guest user can browse the page where the flight details are displayed whereas the logged in user only can book tickets.**

**The Login and Logout buttons should accordingly display different pages. Once the user is logged in the User page should be displayed. When the user clicks on Logout, the Guest page should be displayed.**

**ANSWER:  
  
index.js:**import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import TicketBookingApp from './App';

ReactDOM.render(

<React.StrictMode>

<TicketBookingApp />

</React.StrictMode>,

document.getElementById('root')

);  
  
  
**index.css:**body {

font-family: Arial, sans-serif;

margin: 0;

padding: 20px;

background-color: #f5f5f5;

}

.app {

max-width: 800px;

margin: 0 auto;

background-color: white;

padding: 20px;

border-radius: 8px;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 20px;

padding-bottom: 10px;

border-bottom: 1px solid #eee;

}

button {

background-color: #4CAF50;

color: white;

border: none;

padding: 10px 15px;

border-radius: 4px;

cursor: pointer;

font-size: 14px;

}

button:hover {

background-color: #45a049;

}

ul {

list-style-type: none;

padding: 0;

}

li {

padding: 8px 0;

border-bottom: 1px solid #eee;

}

**App.js:**

import React, { useState } from 'react';

function LoginButton(props) {

return (

<button onClick={props.onClick}>

Login

</button>

);

}

function LogoutButton(props) {

return (

<button onClick={props.onClick}>

Logout

</button>

);

}

function UserGreeting() {

return (

<div>

<h2>Welcome back</h2>

<p>You can now book tickets for flights.</p>

<div>

<h3>Available Flights</h3>

<ul>

<li>Flight 101: New York to London - $499</li>

<li>Flight 202: Los Angeles to Tokyo - $899</li>

<li>Flight 303: Chicago to Paris - $699</li>

</ul>

<button>Book Tickets</button>

</div>

</div>

);

}

function GuestGreeting() {

return (

<div>

<h2>Please sign up.</h2>

<p>Guest users can browse flight details but cannot book tickets.</p>

<div>

<h3>Available Flights</h3>

<ul>

<li>Flight 101: New York to London - $499</li>

<li>Flight 202: Los Angeles to Tokyo - $899</li>

<li>Flight 303: Chicago to Paris - $699</li>

</ul>

</div>

</div>

);

}

function Greeting(props) {

const isLoggedIn = props.isLoggedIn;

if (isLoggedIn) {

return <UserGreeting />;

}

return <GuestGreeting />;

}

function TicketBookingApp() {

const [isLoggedIn, setIsLoggedIn] = useState(false);

const handleLoginClick = () => {

setIsLoggedIn(true);

};

const handleLogoutClick = () => {

setIsLoggedIn(false);

};

let button;

if (isLoggedIn) {

button = <LogoutButton onClick={handleLogoutClick} />;

} else {

button = <LoginButton onClick={handleLoginClick} />;

}

return (

<div className="app">

<header>

<h1>Ticket Booking App</h1>

{button}

</header>

<main>

<Greeting isLoggedIn={isLoggedIn} />

</main>

</div>

);

}

export default TicketBookingApp;  
  
**OUTPUT:** **OBJECTIVES:  
1) Explain various ways of conditional rendering  
Answer:**React allows multiple ways to render UI conditionally based on logic:  
1) Using if statements  
if (isLoggedIn) {

return <Dashboard />;

} else {

return <Login />;

}  
2) Using ternary operator  
{isLoggedIn ? <Dashboard /> : <Login />}  
  
  
3) Using logical AND (&&) operator  
{showMessage && <p>This is a message</p>}

4)Using element variables

let content;

if (isAdmin) {

content = <AdminPanel />;

} else {

content = <UserPanel />;

}

return <div>{content}</div>;

1. **Explain how to render multiple components  
   Answer:**We can render multiple components inside a parent component using JSX. Typically, they're wrapped in a container like <div> or <> (React fragment).Define list component  
   Example:  
   function App() {

return (

<>

<Header />

<Content />

<Footer />

</>

);

}

1. **Define list component  
   Answer:**A list component in React is used to display a collection of similar items. It iterates through an array and renders each item using a loop, often with the map() function.

Example:  
function NameList() {

const names = ['Alice', 'Bob', 'Charlie'];

return (

<ul>

{names.map(name => <li key={name}>{name}</li>)}

</ul>

);

}

1. **Explain about keys in React applications  
   Answer:**Keys are special string attributes assigned to list elements in React. They help React identify which items have changed, been added, or removed, optimizing re-rendering.

Example:  
const items = ['A', 'B', 'C'];

const list = items.map((item, index) => <li key={index}>{item}</li>);

1. **Explain how to extract components with keys  
   Answer:**When mapping over a list, it’s good practice to extract each item into its own component and assign the key at the parent level.  
   Example:  
   function ListItem(props) {

return <li>{props.value}</li>;

}

function NumberList(props) {

return (

<ul>

{props.numbers.map(num => (

<ListItem key={num.toString()} value={num} />

))}

</ul>

);

}

The **key**is passed to the **ListItem**during mapping in **NumberList**.

1. **Explain React Map, map() function  
   Answer:**In React, the **map()** function is commonly used to iterate over arrays and return a list of elements (usually JSX).

Example:

const fruits = ['Apple', 'Banana', 'Mango'];

const fruitList = fruits.map(fruit => <li key={fruit}>{fruit}</li>);

It’s a standard JavaScript array method used in React to dynamically generate components from data.

1. **Create a React App named “bloggerapp” in with 3 components.**

**1.Book Details**

**2.Blog Details**

**3.Course Details**

**Implement this with as many ways possible of Conditional Rendering.  
  
Answer:   
App.js:**import React, { useState } from 'react';

import './App.css';  
function BookDetails({ books }) {

return (

<div className="book-details">

<h1>Book Details</h1>

<ul>

{books.map((book) => (

<div key={book.id}>

<h3>{book.bname}</h3>

<h4>${book.price}</h4>

</div>

))}

</ul>

</div>

);

}

function BlogDetails({ blogs }) {

return (

<div className="blog-details">

<h1>Blog Details</h1>

{blogs.map((blog) => (

<div key={blog.id}>

<h2>{blog.title}</h2>

<h3>{blog.author}</h3>

<p>{blog.content}</p>

</div>

))}

</div>

);

}

function CourseDetails({ courses }) {

return (

<div className="course-details">

<h1>Course Details</h1>

{courses.map((course) => (

<div key={course.id}>

<h2>{course.name}</h2>

<p>{course.date}</p>

</div>

))}

</div>

);

}

function App() {

const books = [

{ id: 1, bname: "Master React", price: 670 },

{ id: 2, bname: "Deep Dive into Angular 11", price: 800 },

{ id: 3, bname: "Mongo Essentials", price: 450 }

];

const blogs = [

{ id: 1, title: "React Learning", author: "Stephen Biz", content: "Welcome to learning React!" },

{ id: 2, title: "Installation", author: "Schwarzenier", content: "You can install React from npm." }

];

const courses = [

{ id: 1, name: "Angular", date: "4/5/2021" },

{ id: 2, name: "React", date: "6/3/2021" }

];

const [activeTab, setActiveTab] = useState('books');

const [showAll, setShowAll] = useState(false);

const renderContentWithIfElse = () => {

if (activeTab === 'books') {

return <BookDetails books={books} />;

} else if (activeTab === 'blogs') {

return <BlogDetails blogs={blogs} />;

} else {

return <CourseDetails courses={courses} />;

}

};

const renderContentWithTernary = () => (

activeTab === 'books'

? <BookDetails books={books} />

: activeTab === 'blogs'

? <BlogDetails blogs={blogs} />

: <CourseDetails courses={courses} />

);

const renderContentWithLogicalAnd = () => (

<>

{activeTab === 'books' && <BookDetails books={books} />}

{activeTab === 'blogs' && <BlogDetails blogs={blogs} />}

{activeTab === 'courses' && <CourseDetails courses={courses} />}

</>

);

const renderContentWithSwitch = () => {

switch(activeTab) {

case 'books':

return <BookDetails books={books} />;

case 'blogs':

return <BlogDetails blogs={blogs} />;

case 'courses':

return <CourseDetails courses={courses} />;

default:

return <BookDetails books={books} />;

}

};

return (

<div className="app">

<h1>Blogger App</h1>

<div className="tabs">

<button onClick={() => setActiveTab('books')}>Books</button>

<button onClick={() => setActiveTab('blogs')}>Blogs</button>

<button onClick={() => setActiveTab('courses')}>Courses</button>

<button onClick={() => setShowAll(!showAll)}>

{showAll ? 'Show One' : 'Show All'}

</button>

</div>

<div className="content">

{showAll ? (

// Show all components when showAll is true

<>

<BookDetails books={books} />

<BlogDetails blogs={blogs} />

<CourseDetails courses={courses} />

</>

) : (

<div className="rendering-methods">

<h2>Conditional Rendering Methods:</h2>

<div className="method">

<h3>1. If-Else</h3>

{renderContentWithIfElse()}

</div>

<div className="method">

<h3>2. Ternary Operator</h3>

{renderContentWithTernary()}

</div>

<div className="method">

<h3>3. Logical &&</h3>

{renderContentWithLogicalAnd()}

</div>

<div className="method">

<h3>4. Switch Case</h3>

{renderContentWithSwitch()}

</div>

</div>

)}

</div>

</div>

);

}

export default App;  
  
**App.css:**body {

font-family: Arial, sans-serif;

margin: 0;

padding: 20px;

background-color: #f5f5f5;

}

.app {

max-width: 1000px;

margin: 0 auto;

background-color: white;

padding: 20px;

border-radius: 8px;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

h1 {

color: #333;

text-align: center;

}

.tabs {

display: flex;

justify-content: center;

gap: 10px;

margin-bottom: 20px;

}

button {

background-color: #4CAF50;

color: white;

border: none;

padding: 10px 15px;

border-radius: 4px;

cursor: pointer;

font-size: 14px;

}

button:hover {

background-color: #45a049;

}

.content {

margin-top: 20px;

}

.book-details, .blog-details, .course-details {

background-color: #f9f9f9;

padding: 15px;

border-radius: 5px;

margin-bottom: 20px;

}

.rendering-methods {

display: grid;

grid-template-columns: 1fr 1fr;

gap: 20px;

}

.method {

background-color: #f0f0f0;

padding: 15px;

border-radius: 5px;

}

.method h3 {

margin-top: 0;

color: #2c3e50;

}  
  
**Output:**