**Objectives**

**Explain various ways of conditional rendering**

|  |  |  |
| --- | --- | --- |
| Method | Example | Use When... |
| Ternary Operator | {isLoggedIn ? <User /> : <Guest />} | Two clear conditions |
| Logical && Operator | {isAdmin && <AdminPanel />} | Only render if condition is true |
| if-else Statements | if (loggedIn) return <User />; else return <Guest />; | More complex logic before returning JSX |
| Element Variables | let view = isLoggedIn ? <User /> : <Guest />; | When you want to assign JSX to variables |
| Early Return with null | if (!show) return null; | Skip rendering component completely |

**Explain how to render multiple components**

You can render multiple components inside a single parent (like <div> or <> fragment):

Example:

function App() {

return (

<>

<Header />

<Content />

<Footer />

</>

);

}

Use fragments (<>...</>) or <div> to group multiple components.

**Define list component**

A List Component is used to dynamically render a list of items using .map().

Example:

function NameList() {

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

return (

<ul>

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

</ul>

);

}

It loops through the array and returns a list element for each.

**Explain about keys in React applications**

Keys are unique identifiers assigned to list elements to help React track changes efficiently.

Example:

{items.map(item => <li key={item.id}>{item.name}</li>)}

Why Important:

* Helps React know which items changed, added, or removed
* Improves performance in dynamic lists

**Explain how to extract components with keys**

When rendering a list, it's best practice to extract list items into their own components, and pass the key to the component itself.

Example:

function Item({ item }) {

return <li>{item.name}</li>;

}

function ItemList() {

const items = [{ id: 1, name: 'Book' }];

return items.map(item => <Item key={item.id} item={item} />);

}

This keeps code clean and maintainable.

**Explain React Map, map() function**

React commonly uses JavaScript’s .map() to loop through arrays and render elements.

Example:

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

return fruits.map((fruit, index) => <li key={index}>{fruit}</li>);

map() transforms each array item into a JSX element.

**Code :**

**--Data.js**

export const books = [

  { id: 101, bname: 'Master React', price: 670 },

  { id: 102, bname: 'Deep Dive into Angular 11', price: 800 },

  { id: 103, bname: 'Mongo Essentials', price: 450 }

];

export const blogs = [

  { title: 'React Learning', author: 'Stephen Biz', body: 'Welcome to learning React!' },

  { title: 'Installation', author: 'Schwezdenier', body: 'You can install React from npm.' }

];

export const courses = [

  { name: 'Angular', date: '4/5/2021' },

  { name: 'React', date: '6/3/20201' }

];

**--Book.js**

**import React from 'react';**

function Book({ books }) {

  return (

    <div className="st2">

      <h1>Book Details</h1>

      {books.map(book => (

        <div key={book.id}>

          <h3>{book.bname}</h3>

          <h4>{book.price}</h4>

        </div>

      ))}

    </div>

  );

}

export default Book;

**--Blog.js**

import React from 'react';

function Blog({ blogs }) {

  return (

    <div className="v1">

      <h1>Blog Details</h1>

      {blogs.map((blog, index) => (

        <div key={index}>

          <h3>{blog.title}</h3>

          <p><strong>{blog.author}</strong></p>

          <p>{blog.body}</p>

        </div>

      ))}

    </div>

  );

}

export default Blog;

**--Course.js**

import React from 'react';

function Course({ courses }) {

  return (

    <div className="mystyle1">

      <h1>Course Details</h1>

      {courses.map((course, index) => (

        <div key={index}>

          <h3>{course.name}</h3>

          <p>{course.date}</p>

        </div>

      ))}

    </div>

  );

}

export default Course;

**--App.js**

import React, { useState } from 'react';

import Book from './Book';

import Blog from './Blog';

import Course from './Course';

import { books, blogs, courses } from './Data';

function App() {

  const [showBooks, setShowBooks] = useState(true);

  const [showBlogs, setShowBlogs] = useState(true);

  const [showCourses, setShowCourses] = useState(true);

  // Element variable example

  let courseSection;

  if (showCourses) {

    courseSection = <Course courses={courses} />;

  }

  return (

    <div style={{ padding: "20px", fontFamily: "Arial" }}>

      <h1>Blogger App</h1>

      {/\* Toggle Buttons \*/}

      <div style={{ marginBottom: "20px" }}>

        <button onClick={() => setShowBooks(!showBooks)}>Toggle Book Details</button>{' '}

        <button onClick={() => setShowBlogs(!showBlogs)}>Toggle Blog Details</button>{' '}

        <button onClick={() => setShowCourses(!showCourses)}>Toggle Course Details</button>

      </div>

      {/\* Conditional rendering section \*/}

      <div style={{ display: 'flex', justifyContent: 'space-around' }}>

        {/\* AND (&&) Operator \*/}

        {showBooks && <Book books={books} />}

        {/\* Ternary Operator \*/}

        {showBlogs ? <Blog blogs={blogs} /> : <p>No blog details to show.</p>}

        {/\* Element variable \*/}

        {courseSection}

      </div>

    </div>

  );

}

export default App;

**Output :**









