## **Objectives**

* Explain various ways of conditional rendering
* Explain how to render multiple components
* Define list component
* Explain about keys in React applications
* Explain how to extract components with keys
* Explain React Map, map() function

In this hands-on lab, you will learn how to:

* Implement conditional rendering in React applications

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **60 minutes.**

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.



**Hint:**







Code:

import React, { useState } from 'react';

// --- Mock Data ---

// Data for the three different components, as shown in the assignment.

const coursesData = [

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

{ id: 2, name: 'React', date: '6/3/2021' }, // Corrected the year from the screenshot

];

const booksData = [

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

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

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

];

const blogsData = [

{ id: 201, title: 'React Learning', author: 'Stephen Biz', content: 'Welcome to learning React!' },

{ id: 202, title: 'Installation', author: 'Schewzdenier', content: 'You can install React from npm.' },

];

// --- Reusable Components ---

// A styled heading for each section.

const SectionHeading = ({ children }) => (

<h1 className="text-2xl font-bold text-gray-800 mb-4">{children}</h1>

);

// A styled container for each of the three main columns.

const DetailColumn = ({ children, className }) => (

<div className={`flex-1 p-6 ${className}`}>

{children}

</div>

);

// --- List Rendering Components ---

// 1. CourseDetails Component: Renders a list of courses.

function CourseDetails({ courses }) {

return (

<div>

<SectionHeading>Course Details</SectionHeading>

{/\* We use .map() to iterate over the array and render each item. \*/}

{/\* The `key` prop is essential for React to efficiently update lists. \*/}

{courses.map(course => (

<div key={course.id} className="mb-4">

<h2 className="text-xl font-semibold text-gray-700">{course.name}</h2>

<p className="text-gray-500">{course.date}</p>

</div>

))}

</div>

);

}

// 2. BookDetails Component: Renders a list of books.

function BookDetails({ books }) {

return (

<div>

<SectionHeading>Book Details</SectionHeading>

{books.map(book => (

<div key={book.id} className="mb-4">

<h3 className="text-lg font-medium text-gray-700">{book.bname}</h3>

<p className="text-gray-500">Price: {book.price}</p>

</div>

))}

</div>

);

}

// 3. BlogDetails Component: Renders a list of blog posts.

function BlogDetails({ blogs }) {

return (

<div>

<SectionHeading>Blog Details</SectionHeading>

{blogs.map(blog => (

<div key={blog.id} className="mb-6">

<h2 className="text-xl font-semibold text-blue-700">{blog.title}</h2>

<p className="text-sm text-gray-500 mb-1">by {blog.author}</p>

<p className="text-gray-600">{blog.content}</p>

</div>

))}

</div>

);

}

// --- Main App Component ---

// This component controls the state and demonstrates conditional rendering.

export default function App() {

// State variables to control the visibility of each section.

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

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

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

return (

<div className="bg-gray-50 min-h-screen font-sans p-4 sm:p-8">

<div className="max-w-6xl mx-auto bg-white rounded-xl shadow-lg">

{/\* --- Controls for Conditional Rendering --- \*/}

<div className="p-6 border-b-2 border-gray-200">

<h1 className="text-xl font-bold mb-3">Conditional Rendering Controls</h1>

<p className="text-gray-600 mb-4">Use these checkboxes to show or hide each section using the logical <code>&&</code> operator.</p>

<div className="flex flex-wrap gap-x-6 gap-y-2">

<label className="flex items-center space-x-2 cursor-pointer">

<input type="checkbox" checked={showCourses} onChange={() => setShowCourses(!showCourses)} className="h-5 w-5 rounded text-blue-600 focus:ring-blue-500"/>

<span>Show Courses</span>

</label>

<label className="flex items-center space-x-2 cursor-pointer">

<input type="checkbox" checked={showBooks} onChange={() => setShowBooks(!showBooks)} className="h-5 w-5 rounded text-green-600 focus:ring-green-500"/>

<span>Show Books</span>

</label>

<label className="flex items-center space-x-2 cursor-pointer">

<input type="checkbox" checked={showBlogs} onChange={() => setShowBlogs(!showBlogs)} className="h-5 w-5 rounded text-purple-600 focus:ring-purple-500"/>

<span>Show Blogs</span>

</label>

</div>

</div>

{/\* --- Main Content Display --- \*/}

<div className="flex flex-col md:flex-row">

{/\* This is the core of the conditional rendering implementation.

The logical AND (&&) operator is used here. If the condition

(e.g., `showCourses`) is true, the component to the right of `&&`

will be rendered. If it's false, nothing is rendered.

\*/}

{showCourses && (

<DetailColumn className="border-r-2 border-green-500">

<CourseDetails courses={coursesData} />

</DetailColumn>

)}

{showBooks && (

<DetailColumn className={showBlogs ? "border-r-2 border-green-500" : ""}>

<BookDetails books={booksData} />

</DetailColumn>

)}

{showBlogs && (

<DetailColumn>

<BlogDetails blogs={blogsData} />

</DetailColumn>

)}

</div>

{/\* This message shows if all sections are hidden \*/}

{!showCourses && !showBooks && !showBlogs && (

<div className="p-10 text-center text-gray-500">

<p className="text-xl">All sections are hidden. Use the controls above to display them.</p>

</div>

)}

</div>

</div>

);

}

Output:









