

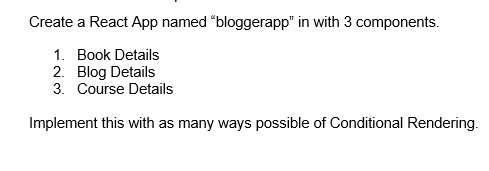
1. There are several ways to perform conditional rendering in React, including:

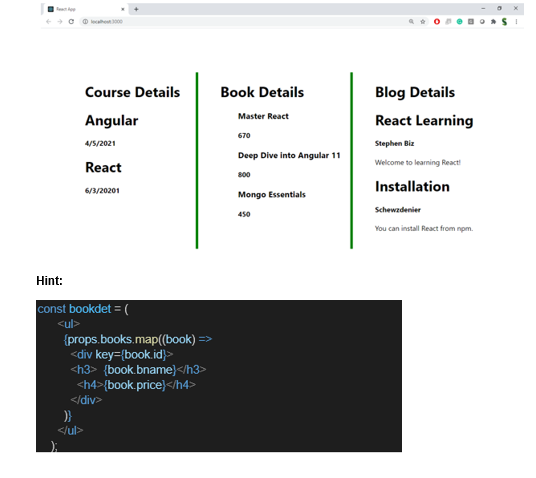
* Using if statements: We can use if statements to conditionally return different components or JSX based on a condition. This is a straightforward approach for simple conditional logic.
* Ternary operator: The ternary operator (condition ? true : false) provides a concise way to conditionally render two different components or elements within JSX.
* Logical && operator: The logical && operator can be used for a short-circuit evaluation. If the condition on the left is true, the element on the right will be rendered. This is useful for conditionally rendering a single element.
* Switch statements: For more complex conditional logic with multiple conditions, a switch statement can be a cleaner alternative to nested if statements.

1. To render multiple components in React, we can either:

* Return an array of elements: We can return an array of components or JSX elements from a render method. Each element in the array must have a unique key prop to help React identify which items have changed, are added, or are removed.
* Use React.Fragment: A React.Fragment allows us to group a list of children without adding extra nodes to the DOM. This is a more common and cleaner way to render multiple components than using an array.

1. A list component in React is a component that is responsible for rendering a collection of data, such as an array of objects or strings. These components typically iterate over the data using a method like map() and render a child component for each item in the collection. A key aspect of a list component is the use of unique key props for each rendered item to optimize React's rendering process and prevent unexpected behavior.
2. Keys are a special string attribute that we need to include when creating lists of elements in React. They are used to uniquely identify each element in a list. When a list is re-rendered, React uses the keys to determine which items have changed, are added, or are removed. This helps React efficiently update the UI and improves performance by preventing unnecessary re-rendering of elements that have not changed. The key should be a stable and unique identifier for each item.
3. When rendering a list of items, it is a good practice to extract the list item into its own component. The key prop should be applied to the top-level element within the map() call, not on the individual component itself. For instance, if we have a ListItem component, the key should be placed on the ListItem when we are mapping over the data. This ensures that the key is associated with the component's identity as a whole, allowing React to properly track and update the list.
4. In React, the map() function is the most common way to create lists of elements. It is a standard JavaScript array method that iterates over an array and returns a new array with the results of a function call on each element. In React, we typically use map() to iterate over an array of data and return a new array of JSX elements. This new array of JSX elements is then rendered to the UI. The map() function is often used in combination with a key prop to help React manage the list efficiently.







**Solution**

**BookDetails.js**

//map + && rendering

import React from 'react';

const BookDetails = ({ books }) => {

return (

<div className="section">

<h1>Book Details</h1>

<ul>

{books.length > 0 && books.map((book) => (

<div key={book.id}>

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

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

</div>

))}

</ul>

</div>

);

};

export default BookDetails;

**BlogDetails.js**

//if-else rendering

import React from 'react';

const BlogDetails = ({ blogs }) => {

if (blogs.length === 0) return <h3>No Blogs Available</h3>;

return (

<div className="section">

<h1>Blog Details</h1>

{blogs.map((blog, index) => (

<div key={index}>

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

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

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

</div>

))}

</div>

);

};

export default BlogDetails;

**CourseDetails.js**

//? : ternary rendering

import React from 'react';

const CourseDetails = ({ courses }) => {

return (

<div className="section">

<h1>Course Details</h1>

{

courses.length

? courses.map((course, i) => (

<div key={i}>

<h3>{course.name}</h3>

<h4>{course.date}</h4>

</div>

))

: <p>No Courses Available</p>

}

</div>

);

};

export default CourseDetails;

**data.js**

export const books = [

{ id: 101, bname: 'Master the world of React', price: 870 },

{ id: 102, bname: 'Deep Dive into Angular 11', price: 1000 },

{ id: 103, bname: 'Mongo Essentials', price: 550 },

];

export const blogs = [

{ title: 'React Learning', author: 'Stephen Biz', content: 'Welcome! Let\'s learn React' },

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

];

export const courses = [

{ name: 'Angular', date: '30/07/2025' },

{ name: 'React', date: '01/08/2025' },

];

**App.js**

import logo from './logo.svg';

import React from 'react';

import './App.css';

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

import BookDetails from './components/BookDetails';

import BlogDetails from './components/BlogDetails';

import CourseDetails from './components/CourseDetails';

function App() {

// Conditional Element Variable

let bookSection = books.length > 0 ? <BookDetails books={books} /> : null;

return (

<div className="App">

<div className="container">

<div className="column"><CourseDetails courses={courses} /></div>

<div className="column"><BlogDetails blogs={blogs} /></div>

<div className="column">{bookSection}</div>

</div>

</div>

);

}

export default App;

**App.css**

.App {

text-align: center;

font-family: Arial, sans-serif;

}

.container {

display: flex;

justify-content: center;

gap: 40px;

margin-top: 50px;

}

.column {

border-left: 5px solid green;

padding: 20px;

}

**Output**

