**React 13**

### ****1. Various Ways of Conditional Rendering in React****

React allows several methods to conditionally render elements or components:

#### ****Using**** if-else ****(inside render/function):****

if (isLoggedIn) {

return <Dashboard />;

} else {

return <LoginForm />;

}

#### ****Using Ternary Operator (****? :****)****:

{isLoggedIn ? <Dashboard /> : <LoginForm />}

#### ****Using Logical AND (****&&****)****:

{hasPermission && <SettingsPanel />}

#### ****Using Switch Statement (inside function)****:

switch(userType) {

case 'admin': return <AdminPanel />;

case 'guest': return <GuestView />;

default: return <DefaultScreen />;

}

#### ****Using IIFE (Immediately Invoked Function Expression)****:

{(() => {

if (error) return <Error />;

if (loading) return <Loading />;

return <Content />;

})()}

### ****2. How to Render Multiple Components****

We can return multiple components in several ways:

#### Using a Wrapper Element (<div> or <section>):

return (

<div>

<Header />

<Content />

<Footer />

</div>

);

#### Using React Fragments (<>...</>):

return (

<>

<Header />

<Content />

<Footer />

</>

);

Fragments are preferred when you don’t want extra nodes in the DOM.

### ****3. Define List Component****

A **List Component** displays a list of items (often using map()).

#### Example:

function ItemList({ items }) {

return (

<ul>

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

<li key={index}>{item}</li>

))}

</ul>

);

}

List components are reusable and useful for displaying dynamic data arrays.

### ****4. Keys in React Applications****

**Keys** help React **identify which items have changed, added, or removed**.

A key must be **unique among siblings**.

Keys improve performance during re-renders.

#### Example:

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

Avoid using array index as key unless items are static and don’t change.

### ****5. Extract Components with Keys****

You can break a list item into its own component and pass key to it:

#### Step 1: Create child component

function ListItem({ value }) {

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

}

#### Step 2: Use it in a list with keys

function ItemList({ items }) {

return (

<ul>

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

</ul>

);

}

The key must be passed on the element returned from map() — not inside the child component.

### ****6. Explain React Map and**** map() ****Function****

map() is a **JavaScript array function** used in React to **render lists dynamically**.

It transforms each element and returns a new array (usually JSX elements).

#### Example:

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

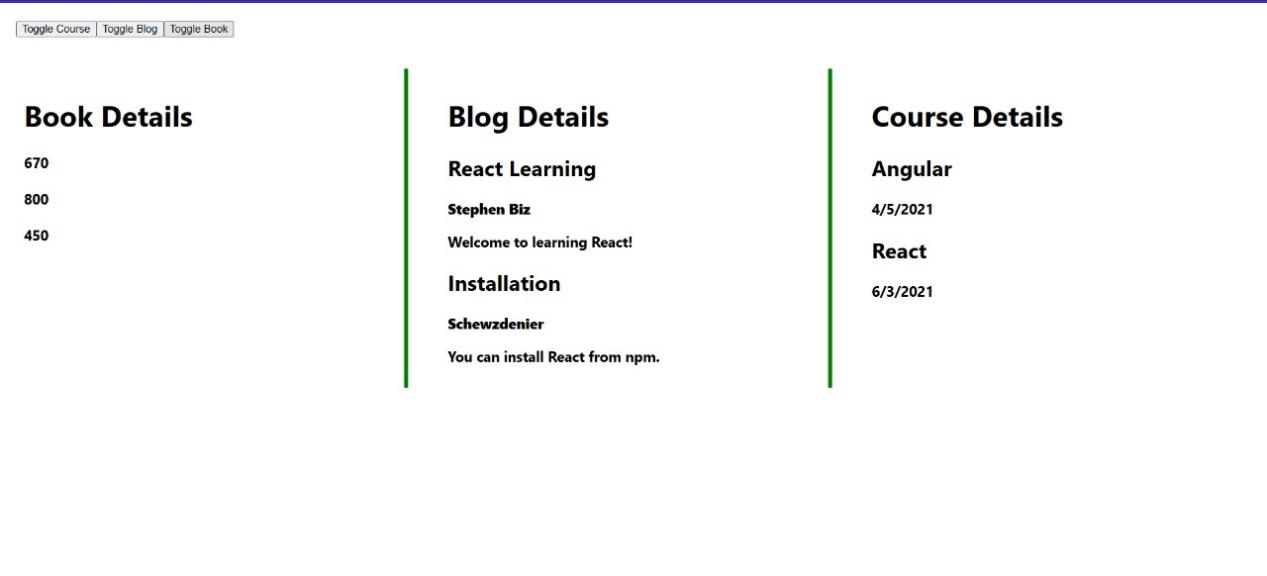
const fruitList = fruits.map((fruit, index) => (

<li key={index}>{fruit}</li>

));

React doesn’t have its own special map() — it just uses JavaScript’s native map().

**Output for this lab:**

****