1. Explain various ways of conditional rendering in React

Conditional rendering in React means showing different UI elements based on certain conditions (like user login status or props).

Ways to do conditional rendering:

a) Using if-else statement:

if (isLoggedIn) {

return <UserGreeting />;

} else {

return <GuestGreeting />;

}

b) Using ternary operator:

return (

<div>

{isLoggedIn ? <UserGreeting /> : <GuestGreeting />}

</div>

);

c) Using logical AND (&&):

return (

<div>

{hasMessages && <p>You have new messages!</p>}

</div>

);

d) Using switch-case inside a function:

function getGreeting(status) {

switch (status) {

case 'admin': return <AdminGreeting />;

case 'user': return <UserGreeting />;

default: return <GuestGreeting />;

}

}

2. Explain how to render multiple components

You can render multiple components by grouping them in a parent element (like a div, section, or React.Fragment).

Example:

function App() {

return (

<div>

<Header />

<MainContent />

<Footer />

</div>

);

}

Or using a Fragment to avoid extra HTML:

<>

<Header />

<MainContent />

<Footer />

</>

3. Define List Component

A List component is used to display a list of items (like users, tasks, or products) in React. We often use the map() function to loop over data.

Example:

function NameList(props) {

const names = props.names;

return (

<ul>

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

</ul>

);

}

<NameList names={['Alice', 'Bob', 'Charlie']} />

4. Explain about keys in React applications

Keys are unique identifiers used by React to track list items and improve performance during re-rendering.

Why keys are important:

* Helps React identify which items changed, added, or removed.
* Prevents bugs in dynamic lists.

Good example with key:

const users = ['Alice', 'Bob'];

return (

<ul>

{users.map(user => <li key={user}>{user}</li>)}

</ul>

);

Avoid using index as key if the list order can change.

5. Explain how to extract components with keys

When rendering lists, we can extract a component and still use keys properly by passing the key to the component from the list.

Example:

function UserItem({ name }) {

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

}

function UserList() {

const users = ['Alice', 'Bob'];

return (

<ul>

{users.map(user => (

<UserItem key={user} name={user} />

))}

</ul>

);

}

The key is still passed to <UserItem />, so React can track each item efficiently.

6. Explain React map and map() function

The map() function is a built-in JavaScript array method used in React to loop over items and return elements.

Syntax:

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

return <Component key={index} />;

});

React usage example:

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

return (

<ul>

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

</ul>

);

Uses:

* To generate JSX from data arrays.
* To render lists dynamically.
* Keeps code clean and reusable.