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

Conditional rendering lets you show different UI elements based on logic or data. The main ways are:

* **Ternary Operator:**

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

* **Logical AND (&&):**  
  Shows the element only if the condition is true.

{error && <p>Error occurred.</p>}

* **if/else Statement:**  
  Used before the return, especially when rendering is complex.

**let** content;

**if** (isLoading) {

content = <Spinner />;

} **else** {

content = <MainPage />;

}

**return** <div>{content}</div>;

* **Switch Statement:**  
  For multiple cases.

**switch** (status) {

**case** 'success': **return** <Success />;

**case** 'error': **return** <Error />;

**default**: **return** <Loading />;

}

**2. Rendering Multiple Components**

You can render several components together by placing them side by side in JSX:

**function** App() {

**return** (

<div>

<Header />

<MainContent />

<Footer />

</div>

);

}

Or, using an array:

**return** [<A key="a" />, <B key="b" />, <C key="c" />];

**3. Define List Component**

A **List Component** is a React component designed to render a series of items, usually using an array and the map() function.

Example:

**function** NameList({ names }) {

**return** (

<ul>

{names.map((name, idx) => (

<li key={idx}>{name}</li>

))}

</ul>

);

}

**4. About Keys in React**

**Keys** are unique values given to elements when creating lists in React. They help React identify which items have changed, been added, or removed, improving performance and preventing unexpected bugs.

Example:

**const** items = [{ id: 1, label: "A" }, { id: 2, label: "B" }];

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

**5. How to Extract Components with Keys**

When you split a list-rendering section into a separate component, pass the key prop when you call that component, not inside its definition.

**Correct:**

**function** Item({ value }) {

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

}

**function** ItemList({ items }) {

**return** (

<ul>

{items.map(item =>

<Item key={item.id} value={item.label} />

)}

</ul>

);

}

**6. React Map, map() Function**

The map() function is a standard JavaScript array method. In React, it’s often used to transform arrays of data into arrays of React elements for rendering lists.

**Example:**

**const** colors = ["Red", "Blue", "Green"];

**const** colorItems = colors.map(color => <li key={color}>{color}</li>);

**return** <ul>{colorItems}</ul>;

* Each entry in the array becomes a <li>.
* Always assign a unique key to each top-level element in the list.