• Question 1: How do you render a list of items in React? Why is it important to use keyswhen rendering lists?

```
const items = ['Apple', 'Banana', 'Cherry'];


function ItemList() {
  return (
    <ul>
      {items.map((item, index) => (
        <li key={index}>{item}</li>
      ))}
    </ul>
  );
}
```

Why are keys important when rendering lists?

Keys help React efficiently update and manage the components in a list. When you have a list of items, React uses keys to identify which items have changed, been added, or been removed. This way, it can minimize unnecessary re-renders and only update the parts of the DOM that have actually changed, which boosts performance.

Without keys, React may re-render the entire list or lose track of which elements have been modified, leading to inefficiencies or potential bugs in UI rendering.

A good key is typically a unique and stable value (like an ID) rather than an index. While using the index is fine in some cases, it can cause issues when the list order changes. A unique key prevents React from misidentifying elements when the list changes.

• Question 2: What are keys in React, and what happens if you do not provide a unique key?

What are keys in React?

In React, **keys** are unique identifiers assigned to elements in a list or collection of components. React uses these keys to keep track of which items have changed, been added, or removed, and to efficiently update the UI. They allow React to optimize the rendering process by only updating the necessary elements rather than re-rendering the entire list.

Keys are typically set on the root element of each item in a list. For example:

jsx

Copy

```
const items = ['Apple', 'Banana', 'Cherry'];


function ItemList() {
  return (
    <ul>
      {items.map((item, index) => (
        <li key={item}>{item}</li>  // `key` is set to the item value
      ))}
    </ul>
  );
}
```

In this case, key={item} uses the value of each fruit name as the unique identifier.

What happens if you do not provide a unique key?

If you don’t provide a unique key or omit it altogether, React will use the default algorithm for determining which elements to re-render. While this might not cause immediate issues, it can lead to:

1. **Poor performance**: Without keys, React has to re-render more components than necessary, especially when there are updates or changes in the list (such as adding or removing items). This can hurt performance, particularly in large lists.
2. **Incorrect UI updates**: In some cases, React may not be able to properly match elements in the list to their corresponding data. This could lead to rendering bugs where data is mismatched or where components don't update properly when the list changes.
3. **Unpredictable behavior**: When elements are reordered or removed, React may incorrectly reuse elements without properly resetting their state or properties, leading to unexpected results in the UI.