Lists and Keys

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

Ans : In React, you usually render a list of items by mapping over an array and returning JSX for each element.

Example:

```
function ItemList() {
  const items = ["Apple", "Banana", "Orange"];


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

- map() loops through the array.
- Each li element is returned with a **key prop**.

🔑 **Why keys are important?**

- React uses keys to **track changes** in lists (additions, deletions, reordering).
- Without keys, React might **re-render the entire list unnecessarily** instead of just updating changed items.

- This improves **performance** and ensures React updates the **right elements**.

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

- **Ans :** A key is a special string attribute used by React to identify each element in a list.
- Keys should be unique and stable (not change between renders).
- Example: IDs from a database are the best keys.

Example:

```
const users = [
  { id: 1, name: "Raj" },
  { id: 2, name: "Priya" }
];


function UserList() {
  return (
    <ul>
      {users.map(user => (
        <li key={user.id}>{user.name}</li>  {/* unique id as key */}
      ))}
    </ul>
  );
}
```

If you don’t provide a unique key:

1. React falls back to using **array index** as the key (not ideal).
2. Problems may occur if list items are reordered or removed:
    - Wrong items may be updated.
    - Unnecessary re-renders happen.
    - Component state (like input values inside a list item) may reset unexpectedly.