🔑 Keys in React – Why They're Important

In React, **keys** help identify which items have **changed, been added, or removed** when rendering lists. They make React updates **efficient and predictable**.

✅ Why Use Keys?

- Improve performance during rendering
- Help React avoid unnecessary re-renders
- Maintain component state correctly in dynamic lists

🧪 Example Without Keys (⚠️ Warning)

jsx Copy Edit

```jsx
const items = ['Apple', 'Banana', 'Orange']; function FruitList() { return ( <ul>
{items.map((fruit) => ( <li>{fruit}</li> // ⚠️ Missing key! ))} </ul> ); }
```

⚠️ React will show a warning:
"Each child in a list should have a unique 'key' prop."

✅ Correct Usage: Use `key` Prop

jsx Copy Edit

```jsx
{items.map((fruit, index) => ( <li key={index}>{fruit}</li> ))}
```

✅ This works, but avoid using `index` as key if the list can change.

🔐 Best Practice: Use Unique IDs

jsx Copy Edit

```
const users = [ { id: 101, name: 'Vivek' }, { id: 102, name: 'Priya' } ]; {users.map((user) => (
<li key={user.id}>{user.name}</li> ))}
```

⚠️ When NOT to use Index as Key

Avoid using index as key **if:**

- List can be reordered
- Items can be deleted or added dynamically

It can cause:

- Wrong component re-renders
- Input state bugs
- Unexpected behavior

🤖 Summary

🔸 Use...	✅ When
`unique ID`	Best option always
`index`	Only when list is static