# Lists , Hooks , Localstorage , Api Project

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

A-1: In React, you can render a list of items using the .map()function, Using a for loop, Using forEach loop etc.

Important to use keys when rendering a list:

- To provide a unique identity to each list element from lists.
- It is useful when a user alters the list.
- It is used to identify which items have changed, updated, or deleted from the Lists.

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

A-2: Keys: keys are special string attributes you add to elements when rendering lists. They act as unique identifiers that help React efficiently manage and update components in the Virtual DOM.

- Without keys, React re-renders the entire list, making updates slower and less efficient.
- This affects performance, especially with large lists.
- React may misinterpret changes and reuse the wrong components.

# Hooks (useState, useEffect, useReducer, useMemo, useRef, useCallback)

Q:1 : What are React hooks? How do useState() and useEffect() hooks work in functional components?

A-1 : Hooks: Hooks allow us to "hook" into React features such as state and lifecycle methods. Hooks allow function components to have access to state and other React features. Because of this, class components are generally no longer needed.

- useState(): It is used How do useState() and useEffect hooks work in functional components. State refers to data that can change over time and trigger re-renders of components.
- useEffect(): It is used To handle the side effects in functional components. The side effect is any operation that affects something outside a component such as data fetching, interacting with the browser DOM.

Q:2 :What problems did hooks solve in React development? Why are hooks considered an important addition to React?

A-2 : Problems that solve in react development:

- Reusing logic was complicated.
- Complex components became hard to maintain.
- Confusing Lifecycle Methods.

→ hooks considered an important addition to React because they simplify how developers manage state, side effects, and reusable logic in functional components. Before hooks, functional components were limited to presenting UI without state, while class components were necessary for more complex logic. Hooks eliminate that divide, bringing several important benefits.

# Hooks (useState, useEffect, useReducer, useMemo, useRef, useCallback)

Q:3 : What is useReducer ? How we use in react app?

A-3 : The useReducer hook is a React hook used for state management in functional components. It's an alternative to useState.

→ UseReducer is used as an alternative to useState for managing complex state logic.

- Reducer function - a function that specifies how state changes in response to actions.
- Initial state - the starting state.

Q:4 : What is the purpose of useCallback & useMemo Hooks?

A-4 : useCallback hook : useCallback memoizes a function, so it doesn't get recreated on every render unless its dependencies change.

- The useCallback hook is used to memoize the functions to prevent unnecessary re-renders.

→ useMemo : The useMemo is similar to useCallback hook as it accepts a function and a list of dependencies but it returns the memoized value returned by the passed function. It recalculated the value only when one of its dependencies change.

# Hooks (useState, useEffect, useReducer, useMemo, useRef, useCallback)

Q:5 : What's the Difference between the useCallback & useMemo Hooks?

A-5 : useCallback :

- Returns a memorized callback.
- It returns referential equality between renders for function.
- It returns the function when the dependencies changes.

→ useMemo :

- Returns a memorized value.
- It returns referential equality between renders for values.
- It calls the function when the value changes and returns result.

# Hooks (useState, useEffect, useReducer, useMemo, useRef, useCallback)

Q:6 : What is useRef ? How to work in react app?

A-6 : useRef : The useRef Hook allows you to persist values between renders.

→ It can be used to store a mutable value that does not cause a re-render when updated.

→ It can be used to access a DOM element directly.

How it works : useRef(null) creates a reference.

→ The Ref is assigned to the <input> element.