React – 3

01.Lists and Keys

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

To render a list of items in React, you typically use the .map() function to iterate over an array and return JSX elements.

Keys are significant in React because they aid in determining whether items in a list have been changed, updated, or removed.

This process helps React to optimize the rendering by recycling existing DOM elements.

02) What are keys in React, and what happens if you do not provide a unique key?

A key is a unique identifier.

It is used to identify which items have changed, updated, or deleted from the Lists.

It is useful when we dynamically created components or when the users alter the lists. if you don't provide a unique key when rendering a list of components using .map(),

React will display a warning in the console, and you may encounter performance issues or unexpected UI behavior.

02.Lifecycle Methods (Class Components)

01) What are lifecycle methods in React class components? Describe the phases of a component’s lifecycle.

In React class components, lifecycle methods are special functions that are called at different stages of a component's existence, like when it's first rendered (mounted), when its props or state change (updated), or when it's removed from the DOM (unmounted), allowing developers to perform specific actions at each phase of the component's lifecycle.

The React component lifecycle is typically divided into three phases: Mounting, Updating, and Unmounting.

Examples of lifecycle methods:

Mounting phase:

constructor: Called when a component is initialized.

static getDerivedStateFromProps: Allows updating state based on initial props.

componentDidMount: Called after a component is rendered to the DOM for the first time.

Updating phase:

shouldComponentUpdate: Can be used to optimize re-renders by deciding if a component needs to update based on prop or state changes.

componentDidUpdate: Called after a component updates.

Unmounting phase:

componentWillUnmount: Called right before a component is removed from the DOM.

02) Explain the purpose of componentDidMount(), componentDidUpdate(), and componentWillUnmount().

componentDidMount():

After the component is mounted to the DOM.

Common uses:

Fetching data from an API and updating the component's state

Setting up event listeners

Performing DOM manipulations that require the element to be fully rendered

componentDidUpdate():

After a component updates its state or props

Common uses:

Updating external APIs based on state changes

Performing calculations based on updated props or state

Optimizing rendering by checking if significant changes have occurred

componentWillUnmount():

Right before a component is removed from the DOM

Common uses:

Clearing timers

Removing event listeners

Unsubscribing from external data sources

03. Hooks (useState, useEffect, useReducer, useMemo, useRef, useCallback)

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

Hook is a simple function

Hook is a default module of react js

Hooks are called while adding some features | use effects | fetch data | call api | stored memory | destructured of data etc.

Hook is add some features inside of react js that is use effects | fetch data | call api | stored memory | destructured of data etc

Hook is also used to take data from users there e called useRef

Note :hook is added a new features that can be only support in function components.

React js 16.8.3 is no more support hooks in class based components

Hook is not supported class based components

Types of hooks

a) useState()

b) useEffect()

c) useRef()

d) useMemo()

e) useFetch()

f) Fetch()

g) useReducer()

h) custom hook

usestate()

useState : useState is a type of hooks

useState is used to stored all type of data ex: number | boolean | bigIntger | array | objects

useState is used to inisilised data in form of array const[data,setData]=useState(0)

useState is destructured of data const[data,setData]=useState(0)

useState is used to update data one components to another components

useState is provides immutable data

useState is stored data in objects

useEffect()

useEffect(): useEffect is a types of hook

this hook is used to add some effects in our react js applications

this hook is add to fetch data | timers | add effects etc

useEffect(()=>{},[dependency])

dependency is pass in useEffect for ...

a) no dependency b) pass dependency [] c) [props,state]

a) no dependency : render data again and again

b) [] blank array render a data one times

c) [props,state] render data one times and we can even update data also

02) What problems did hooks solve in React development? Why are hooks considered an important addition to React?

React Hooks solves the problems of sharing the stateful logic between components in a more modular and reusable way than class component.

Hooks were added to React in version 16.8. Hooks allow function components to have access to state and other React features. Because of this, class components are generally no longer needed.

Hooks generally replace class components, there are no plans to remove classes from React.

03) What is useReducer ? How we use in react app?

The useReducer hook in React is like a more powerful version of useState.

It helps manage complex state logic in a more organized way.

How it Works:

You have a state (data you want to manage).

You define a reducer function that tells how to change the state based on an action.

You use useReducer(reducer, initialState), which returns:

The current state.

A dispatch function to trigger actions and update the state.

04) What is the purpose of useCallback & useMemo Hooks?

useCallback → Remembers a function so it doesn't get recreated every time.

Use case: When passing functions to child components to prevent unnecessary re-renders.

useMemo → Remembers a computed value so it doesn't get recalculated every time.

Use case: When performing expensive calculations to improve performance.

05) What's the Difference between the useCallback & useMemo Hooks?

useCallback hook should be used when we want to memoize a callback function, and. we can use useMemo to memoize the result of a function to avoid expensive computation.

useEffect is used to produce side effects to some state changes

06) What is useRef ? How to work in react app?

xios : axios is a libraries i.e used to Post | get | put | delete | update data from any URL

useRef():This is a type of hook this is used to take reference from user input

This hook is used to add data via useRef()

The useRef hook takes an initial value of any type as argument and returns an object with a single current property.

const ref = useRef(initialValue); React will set the initialValue you pass to the useRef hook as the value of the current property of the returned ref object.

04.Context API

01) What is the Context API in React? How is it used to manage global state across multiple components?

The Context API in React is a feature that allows you to manage the global state of your application without the need to pass data through multiple levels of components using props.

It provides a way to share data and functionality across different components, regardless of where they are located in the component tree.

02) Explain how createContext() and useContext() are used in React for sharing state.

createContext() creates a context object (MyContext) that holds a default value. MyContext.

Provider passes down the context value to its child components.

useContext(MyContext) allows components like ChildComponent to access the context value.