React Interview Questions and Answers

1. Single Page Application (SPA) vs. Multi-Page Application (MPA)

SPA In single page web application, the reloading will not happen for every functionality we are performing. In single web application If we send the request to the server, we will get the response & again if we are sending any new request to the server, we will get the new response and the old response will get store it the browser cache. single page web application must faster in performs then MTA.

MPA In multi-page web application reloading will happen for every functionality we are performing In multi-page web application, every time both old and new request will go the server and will get new response. multi-page web application is better SEO then single page web application

2. What is React and its Specifications?

React: React is a popular JavaScript library developed by Meta (formerly Facebook) for building user interfaces

Specifications: Component based architecture, Declarative in nature , Virtual DOM.

3. Reconciliation (Important)

Reconciliation is the process of comparing the Virtual DOM with the real DOM to find differences. Only the changes are applied to the real DOM, making updates more efficient.

4. Actual DOM vs. Virtual DOM & Diffing Algorithm

Actual DOM: Slower updates as the entire DOM is re-rendered.

Virtual DOM: It is a light weight copy of real DOM which is used to optimize the update in DOM

Diffing Algorithm: It enables React to efficiently update the DOM by determining the minimum number of changes needed when the application state changes.

5. npm vs. npx

npm (Node Package Manager): Installs packages globally or locally.

npx (Node Package Executor): Runs packages without installing them globally.

6. JSX and Its Rules

JSX (JavaScript XML): JSX stands for JavaScript xml, and it is used to write html code in react.

Rules:

- Return a single parent element.
- Use curly braces {} for dynamic values.
- Each and every tag must be closed

7. Components and Their Types

- **Functional Components**: Stateless and use hooks.
- **Class Components**: Stateful and use lifecycle methods.

8. State and How to Update It (Important)

- **State**: State holds the information or data about that particular component. We could not able to transfer the state from one component to another components. State are mutable (we can change). Whenever there is any changes in the state value the components will re-render.
- **Updating State**: Use setState() in class components, useState() in functional components.

9. React Hooks and Built-in Hooks

- React Hooks are special functions that allow you to use React features (like state and lifecycle methods) within functional components
- Common hooks: useState, useEffect, useContext, useReducer, useRef.

10. Props, Prop Drilling, PropTypes, DefaultProps (Important)

- **Props**: Transfer the data from one component to another.Data flow from parent to Child. Immutable
- **Prop Drilling**: Passing props through multiple nested components.
- **PropTypes**: Type-checking for props.
- **DefaultProps**: Default values for props.

11. Ref and useRef Hook

- **Ref**: In react refs is an attribute or property that provides a way to directly access or interact with DOM element
- **useRef**: Preserves values across renders without causing re-renders.

What Is Ref ?

Refs is an attribute or property that provides a way to directly access or **interact with DOM element or react component.**

- In CBC we should use the method React.createRef() & It will return ref object with key current & value as null.
- If FBC to create a ref object we can use HOOK useRef() & it will return obj with key current and value undefined.

What are Keys in React?

Keys in React are special attributes used to identify the items in a list.

12. Controlled vs. Uncontrolled Components

- **Controlled**: Form data handled by react state . onchange(event).
- **Uncontrolled**: Form data handled by DOM itself . ref(mandator)

13. Component Lifecycle

- **Mounting, Updating, Unmounting** phases.
- Class components use lifecycle methods.

14. Lifecycle Methods in Class Components (Important)

Mounting Phases:

Constructor , static getderivedstatefromprops(props, state) , Render() , ComponentDidMount ()

Updating Phases:

Static get derived State from props, Should component update(),Render(),getSnapShotBeforeUpdate()-> props , state , component did update() , component DidUpdate()→ props,state, snapshot value.

UnMounting Phases:

componentwillunmount()

15. useEffect() Hook (Important)

- it’s a inbuild hook which is present inside the react package which is use to perform side effect . In the function based component
- common use case for useeffect data fetching manually updating dom, setting up the API .

- Syntax: useEffect(() => { /* effect */ }, [dependencies]);

What is Side Effect ?

In React, a side effect is an operation that occurs outside of a React component and interacts with the outside world . Side effects can include Data Fetching , Direct DOM Manipulations, Timers , Etc.

16. useContext Hook and Context API

- **useContext**: Consumes context values without prop drilling.
- In React, the Context API is a way to share data between components without having to pass props down through every level of the component tree, acting as a global state for specific data

17. React Routing and useRoutes Hook

- In React, routing refers to the mechanism that allows users to navigate between different pages or sections of a web application without a full page reload, typically using a library like React Router.
- useRoutes() defines routes as an array.

18. Redux (Important)

- Redux is a popular state Management librabrary for js application.
- It is used to manage the global state.
- Redux has Store which holds the state or data ,which can be updated by dispatching actions ,that are handled by Reducer.
- We have one Package tha is React-Redux package used to esablished the connection between React and Redux

- Actions, Reducers, Store, Dispatch.

Different Between Redux and Context API ?

There is no comparison between redux and context API . Therir use Casese are entrirely different Just becases they prevent props drilling ,that way we compare them otherwise , Redux is a state manage labary , while context API is a dependency injection mechanism.

19. useReducer Hook (Important)

- Manages complex state logic.
- const [count, dispatch] = useReducer(reducer, initialValue);
- It will take 2 argument i) reduces function ii) initial state value useReducer hook will return array of 2 element:- 1. State value 2. Dispatch function..

20. useMemo() and useCallback Hook

- **useMemo**: Memoizes values to optimize performance.
- **useCallback**: Memoizes functions to avoid re-renders.

21. Custom Hooks

- Reusable functions that use hooks.
- Custom hooks are designed to encapsulate reusable logic that can be shared among functional components, avoiding code duplication and making your application more maintainable.

22. Creating a React App

- npx create-react-app my-app

23. What is Webpack?

- A module bundler for optimizing and managing assets.

24. Higher-Order Components (HOC)

- A function that returns a new component.

25. Code Splitting in React

- Uses **React.lazy** and **Suspense** to load components dynamically.

26. forwardRef in React

- In React, forwardRef is a utility function that allows a parent component to access a DOM node or a child component's instance through a ref, enabling direct manipulation or interaction with the child's element.

27. Lists and Keys

- Keys help React identify which items changed.

28. Conditional Rendering

- We can render a component based on same component It have been done in 4 ways:

    1. If else

    2. Switch case

    3. Ternary operator

    4. && operator
    .

29. Performance Optimization in Large React Apps

- Memoization (useMemo, useCallback).
- Lazy loading.
- Virtualization (react-window).

Different between Library & Framework ?

Library: A library is like a toolset. You pick and choose what tools (functions or methods) you need and control how to use them. **You are in charge** of your program's flow, and the library just helps you with specific tasks.

Framework: A framework is like a pre-built structure or template. It provides a foundation and controls the flow of your program. **You follow its rules and fill in the gaps** to make your application.

Control: With a library, you are in control. With a framework, the framework is in control.