1. What is React?

Ans🡪React is an open-source front-end JavaScript library that is used for building user interfaces, especially for single-page applications.

Fetures: It uses VirtualDOM instead of RealDOM considering that Real- DOM manipulations are expensive.a  
Supports server-side rendering.  
Follows Unidirectional data flow or data binding.  
Uses reusable/composable UI components to develop the view.

1. What is JSX?

Ans🡪JSX is a XML-like syntax extension to ECMAScript (the acronym stands for JavaScript XML). Basically it just provides syntactic sugar for the React.createElement() function, giving us expressiveness of JavaScript along with HTML like template syntax.

1. What is Conditional Rendering?

Ans🡪In React, you can create distinct components that encapsulate behavior you need. Then, you can render only some of them, depending on the state of your application.

Conditional rendering in React works the same way conditions work in JavaScript.

1. What is Lifecycle Diagram / Phases ( Hooks )

Ans🡪A React component undergoes three phases in its lifecycle: mounting, updating, and unmounting.

**The mounting phase** is when a new component is created and inserted into the DOM or, in other words, when the life of a component begins. This can only happen once, and is often called “initial render.”

**The updating phase** is when the component updates or re-renders. This reaction is triggered when the props are updated or when the state is updated. This phase can occur multiple times, which is kind of the point of React.

The last phase within a component's lifecycle is the **unmounting phase**, when the component is removed from the DOM.

1. Explain how lists work in React?

Ans🡪 We will use the map() function for traversing the list element, and for updates, we enclosed them between curly braces {}. Finally, we assign the array elements to listItems. Now, include this new list inside <ul> </ul> elements and render it to the DOM.

1. What is Virtual DOM & How does it Work?

Ans🡪 React uses Virtual DOM exists which is like a lightweight copy of the actual DOM(a virtual representation of the DOM). So for every object that exists in the original DOM, there is an object for that in React Virtual DOM. It is exactly the same, but it does not have the power to directly change the layout of the document. Manipulating DOM is slow, but manipulating Virtual DOM is fast as nothing gets drawn on the screen. So each time there is a change in the state of our application, the virtual DOM gets updated first instead of the real DOM.

When anything new is added to the application, a virtual DOM is created and it is represented as a tree. Each element in the application is a node in this tree. So, whenever there is a change in the state of any element, a new Virtual DOM tree is created. This new Virtual DOM tree is then compared with the previous Virtual DOM tree and make a note of the changes. After this, it finds the best possible ways to make these changes to the real DOM. Now only the updated elements will get rendered on the page again.

This process of comparing the current Virtual DOM tree with the previous one is known as ‘diffing’.

This entire process of transforming changes to the real DOM is called Reconciliation.

1. How does Class Components work?

Ans🡪 Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML via a render() function.

The component has to include the extends React.Component statement, this statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.

If your component has a constructor function, the props should always be passed to the constructor and also to the React.Component via the super() method.

class Car extends React.Component {

constructor(props) {

super(props);

}

render() {

return <h2>I am a {this.props.model}!</h2>;

}

}

1. What are keys in React?

Ans🡪 A “key” is a special string attribute you need to include when creating lists of elements in React. Keys are used in React to identify which items in the list are changed, updated, or deleted. In other words, we can say that keys are used to give an identity to the elements in the lists.

By this, we did not mean that the keys should be globally unique. All the elements in a particular array should have unique keys. That is, two different arrays can have the same set of keys.

1. What is memoisation in react?
2. What is the difference between react.memo and react.useMemo?
3. What are the parameters that react memo takes?
4. What are the different ways to apply useEffect?
5. How does Routing work with react?
6. What is SSR and CSR?
7. What are the lifecycle methods in class components?
8. What are dependencies in useEffect?
9. What is a pure component?
10. What is useCallback?
11. What are useRefs? What are some usecases?
12. What are callback refs?
13. How does Context API work? What does it solve?
14. What does useReducer do?
15. When do you use useReducer vs useState?
16. How do you use Profiler?
17. Can you create a tree structure and explain how the state management will be designed for a game like tic tac toe?
18. What is dispatcher?
19. What is flux architecture?
20. What is redux?
21. Can you create your own redux and explain all the different entities?
22. What are reducers?
23. What are actions?
24. What are side effects?
25. What all does a store do?
26. What are combine reducers?
27. What are thanks? Why do you need them?
28. Write how you would write an api request in redux thunks?
29. What does lazy loading mean?
30. What does webpack do?
31. What does babel do?
32. Can you write react without jsx?

**REDUX**

1. What is Redux?
2. What are actions?
3. What are action creators?
4. What are dispatchers?
5. What are reducers?
6. What are pure functions?
7. Why do we spread the state or return a new object in reducers?
8. Why do dispatchers take only actions which are objects
9. What are types in actions why do we need them?
10. What happens when you pass a function into a dispatcher?
11. Where do dispatchers come from?
12. What are the properties of a store?
13. Can you create your own redux and create the following methods? dispatchers constructor getState subscribe
14. Explain redux to a 5 year old?
15. What is the difference between Context API and Redux?
16. What are redux thunks?
17. Why do we need to use network requests in redux thunks?
18. What are middlewares?
19. Can you write your own middleware instead of thunks? what do thunks do?
20. Explain with a diagram how state gets updated with redux?
21. What is useSelector?
22. What is the compareFn that you pass in a useSelector?
23. What is useDispatch?
24. What is the use of Provider?
25. what is redux tool kit? How does it make redux better?