React:

1. What is React or use of React or advantages and disadvantages of React ?
    - React is a widely used javascript library mostly used to develop single page applications.
    - Some of its advantages are:
        - Virtual DOM - The process of virtual dom improves the speed and performance of the application.
        - Reusable components - We can avoid writing the same code with the effective usage of reusing the same component.
        - Access to a wide range of external libraries like react-router.
        - Inclusion of Redux, another javascript library to manage states in an application.

    - Some of the disadvantages are:
        - Documentation has always been difficult to understand.
        - Dynamic technology - The updates let the users to unlearn and learn new concepts often.

2. What are **React Hooks** ?
    - React hooks allow you to hook into a state in any life cycle method within a function component.
    - Hooks can be used only in function components and not in class components.
    - Hooks should be initialized at the top of the function component and not inside any block. For eg: hooks should not be declared inside an if else condition or within an iteration.

3. What are **useState** hooks ?
    - Use state hooks return an array of two values: current state and function to update our state.
    - Whenever we update our state using an event listener or similar our entire component will be re-rendered with the updated state.
    - The state can be initialized in two ways:
        - Function version :
            - Using this we can avoid touching upon our initial state every time when the component re-renders.
            - This will run only during the first render.
            - We can use an arrow function with a return keyword which returns our initial state.

        - Hard coded version :
            - Here we will directly declare our initial state. For eg. counter.
            - By doing this the initial state computation will be declared every time when a component re-renders.

4. What are **useEffect** hooks ?
    - Use Effect hooks are used to handle the side effects, like network requests.
    - Use Effect can be used in 3 ways.
        - Use effect takes a function parameter and another parameter is an array which is optional.
        - If we use without the array parameter, the use effect will be rendered during all the phases-mounting, unmounting and updating.
        - If we use an empty array, then the use effect will apply only during the first render which is the mounting phase.
        - The next method is we can declare a state inside this array parameter, where the use effect will apply only when the declared state gets updated.
    - While using useState with an asynchronous function, we must have a clean up function which has to remove this asynchronous function during the unmounting phase of the component. Else it will throw a memory leak error, as our component will be unmounted, but the asynchronous function will still be processing. For eg: SetInterval must have clearInterval with a return keyword doing this clean up operation as a function.

5. What is **babel** ?
    - Babel acts as a javascript compiler as well as transpiler.
    - It converts the newer version of ECMAscripts into a standard version in current and older browsers.
    - For eg: let and const variables are converted into var, and spread operators will be converted in such a way to perform its primary actions.

6. What is **webpack** ?
    - Webpack is a module bundler.
    - Webpack helps to bundle and optimize different available file types in our application. It helps our code to be manageable and useful in debugging our code.
    - For eg., in Vanilla js, we will be writing separate js codes and specifying all the script tags separately which used to be a tedious process.
      Whereas we can visualize webpack as it bundles all these different js script files as a single file sending it to the browser.

7. What is **setState** ?
    - setState changes the component state and tells react that this component and its child has to be re-rendered with its updated state.
    - This is the primary method we use in the response of event listeners and network requests sent by the users in the browser.

8. What is **virtual DOM** ?
    - Virtual DOM is a lightweight representation of actual DOM.

- It is a node tree structure which stores all the elements as an object and its properties.
- It works in three steps:
    - When a component is updated the entire UI of the virtual DOM is re-rendered.
    - Now the updated DOM compares with the previous DOM and updates only the changes.
    - Once the updates are done, now the real DOM updates only the changes which have been made.

9. What are **React portals** ?
    - React portals allow you to render the child element into a DOM node which is outside the DOM hierarchy.
    - This will be mostly used while creating modals, loaders etc..,
    - Syntax for React portals - ReactDOM.createPortal(child, container);

10. What is **context API** ?
    - Context API is used to pass data through component trees without passing any props.
    - This can be used by creating a context as a separate file and using .provider we will be able to export value to another element.
    - In the component where we require these values, react provides a hook called useContext which gets the data from the required context provider.
    - Code for context API:

\\\\\\\ In context provider file: ///////

```
import { createContext } from "react";

export const testContext = createContext();

export const ContextProvider = ({children}) => {
    return (
                                    <testContext.Provider
value={100}>{children}</testContext.Provider>
    )
}
```

\\\\\\\ In index.js file ///////

Import { contextProvider } from "..";

```
<React.StrictMode>
      <BrowserRouter>
        <ContextProvider>
```

```
          <App />
        </ContextProvider>
      </BrowserRouter>
    </React.StrictMode>
```

Note: Context provider must enclose the other tags as <app> is our child here.

\\\\\\\\ In component where we want to use this context API: ///////

Import { useContext } from "react";

const value = useContext(newContext);

11. What are **lifecycle methods** in React ?
    - React lifecycle methods are the series of sequences that all the react component goes through from the birth of a component till its end.
    - Every react component goes through 3 phases:
        - Mounting
        - Updating
        - Unmounting

    - Mounting - Mounting is when React mounts or inserts a component in a DOM tree.
    - Updating - Updating is when React updates a component when there is a change in props or states.
    - Unmounting - Unmounting is when React removes a component from the DOM tree.
    - Within these phases two processes of phases happen, one is rendered and the other one is committed.
    - Within commit three methods happen in each phase which are componentDidMount, componentDidUpdate, componentWillUnmount.

12. How can you **memoize** components in React ?
    - Memoize in React is for the performance of the application, where it is used to speed up the rendering process.
    - So we can use useMemo() in order to achieve this.

13. What is **redux**?
    - Redux manages states. It stores all the states of the variables available in our application.

- It works in such a way that we dispatch our action from our component, this passes into the reducers.
- Reducer is a pure function which takes current state and action and returns the updated state.
- Now from this reducer it moves to the store which manages all the states available in our application.
- From this store it sends the updated props or states back to the component.

14. What is **immutability** ?
    - Immutability means something that cannot modify its value or state.
    - In JavaScript Strings, numbers are some examples for data types which are immutable.

15. What **Object.freeze()** does ?
    - Object.freeze freezes the object from using it.
    - Once we use this we can no longer make changes in these objects like adding or updating any properties.
    - Syntax: Object.freeze(Object_name);

16. Why is **immutability** required by redux ?
    - Immutability means that we cannot modify data.
    - So here, the reducers acts as a pure function, meaning, it takes one input and returns one output based on action.
    - So updating or modifying a data will lead to side effects which is not how the pure functions accept. So Immutability is required by redux.

17. How does redux use **shallow equality checking** ?
    - Shallow equality checking happens in the combineReducer function. It will return either an updated copy of the root state object or the current copy of the root state object.

18. How well does Redux “scale” in terms of performance and architecture?

19. How does Redux compare to the React Context API?
    - Context API is a hook provided by react called useContext which is used to pass data into components without the help of props.
    - Redux is a state management library used to manage all states in react application.
    - The way both Context API and redux works are completely different. Context API uses a .Provider to pass a required value into its children components in the DOM tree.
    - Whereas the Redux working process goes with actions, reduces and stores.
    - Redux avoids unnecessary re-rendering compared to context API.

20. What are some features of using webpack ?

21. What is **Tree shaking** ?
Webpack removes the dead code, that is the unused code from our application before final bundling. This process is called Tree shaking.

22. What is **routing** ?
- React routing is the standard library in react.
- It allows users to navigate between different components in a react application without refreshing the page.
- We can change the browser url.
- This is used in such a way that it takes a route tag and an attribute path where we have to provide the path where the component must get navigated.
- Npm- react router dom, elements used - routes and route. Route takes two attributes: path and element. And Link takes one attribute: to.

23. **Force render** a component ?
- .forceUpdate() method can be used to force re-render a component in react.

24. What is **reconciliation** ?
- It is the process through which react updates the browser DOM.
- —Explain how virtual dom works—

25. What is **useMemo**?
- useMemo hooks are used to improve our rendering performance.
- For eg, if we have a for loop which runs for larger value unnecessarily, means when our component re renders and if we don't want this function to run, then we can use useMemo.
- useMemo works similarly like useEffect, where it takes a function and a dependency which defines the state. Only if this state changes then this function will run.
- But, we cannot use this throughout our application as useMemo memoize our data, that is, it catches the mentioned data and updates only when there is a change. Which could increase our memory if you use useMemo all over our application.

26. What is **useCallback**?
- useCallback is similar to useMemo, but the only difference is useCallback returns a function whereas useMemo just returns a value of the function.

27. What is **useRef** ?
- Is used to get the reference of a particular HTML element.

28. What is **server side rendering**?

- Purpose - It is used in order to have a faster application.
- Process -
    - Traditional rendering:
        - Browser requests page
        - Browser request Js bundles
        - Browser requests API
        - Display content

    - Disadvantage of traditional rendering - To display content we have to wait till the entire request is completed. Which makes our application slow when we are working on larger applications.

    - Server side rendering:
        - Browser request page
        - Displays the static content using the HTML bundle.
        - Browser request other bundle

- We will be using an MVC pattern here to separate our server side files and client side files.
- Next.js is used to achieve SSR in react.

29. What is **JSX**?
    - Stands for Javascript XML file.
    - It allows us to write HTML code inside our JS file.
    - By doing this we don't have to write a set of codes like .getElement or .append.

30. What is **flux** ?
    - Flux is an architecture we follow to maintain our redux.
    - It is followed by actions, dispatcher (reducer file), store and views (Code for UI).

31. What is **React strict mode** ?
    - React strict mode is used to show potential errors in our application.
    - This will not be displayed in UI, it will show them as warnings.
    - This will be applicable only in development mode to find out the problems, and will not affect during production.

32. What is **lazy loading** ?
    - Lazy loading is the process of initializing the critical components first and non-critical components at a later point of time.
    - For eg Imagine you are working on a larger application and every time you update a state all the components are re-rendering. This process is time consuming. So we render the critical component first then jump into the non critical component.
    - React provides two methods - lazy and suspense to perform this operation.

33. What are **states** and **props** ?
    States are to manage data whereas props is the data we send from higher order components to lower order components.

34. What are controlled and uncontrolled components ?
    - Controlled components are inputs which are controlled by react. Here we will be using the useState hook to set our state.
    - Uncontrolled components are inputs which are controlled by DOM itself. Here we will be using a useRef hook which takes the reference value of the input.