Q1: What is Jest?

A1: Jest is a JavaScript testing framework developed by Facebook. It is commonly used for testing React applications but can be applied to any JavaScript project.

Q2: What do you install to get started with Jest and Why?

A2: To get started with Jest, you install the jest package using npm. Jest provides a test runner, assertion library, and mocking capabilities, making it a comprehensive solution for testing JavaScript applications.

Q3: What happens if you execute the following command from your Terminal: npm run test.

A3: Executing npm run test runs the test script defined in the "scripts" section of your package.json file. This script typically invokes Jest to discover and run tests in your project.

Q4: Describe a structure to create the unit test using Jest.

A4: The typical structure for Jest unit tests involves creating a "tests" folder or placing test files adjacent to the modules they test. Test files often have names ending in ".test.js" or ".spec.js."

Q5: What is Enzyme?

A5: Enzyme is a JavaScript testing utility for React that makes it easier to test React components' output and behavior. It provides shallow and full rendering, as well as utilities for interacting with rendered components.

Q6: How to install Enzyme?

A6: Enzyme is installed using npm. The three main packages needed are enzyme, enzyme-adapter-react-16 (or the appropriate version for your React version), and optionally enzyme-to-json for snapshot testing.

Q7: Describe the test process of shallow rendering with Enzyme.

A7: Shallow rendering in Enzyme involves rendering only the top-level component, not its children. This allows you to isolate the component under test. The process includes importing Enzyme, configuring the adapter, and using the shallow method to render the component.

Q8: What is Server-Side Rendering?

A8: Server-Side Rendering (SSR) is a technique where a React application is rendered on the server and then sent to the client as HTML. This can improve initial page load performance and is beneficial for SEO.

Q9: What are the three different components for declarative routing to add with the react-router package?

A9: The three main components for declarative routing in the react-router package are BrowserRouter, Route, and Link.

Q10: Describe the steps to set up routing with react-router.

A10: To set up routing with react-router, you install the package, wrap your application in a BrowserRouter, and define routes using the Route component. Links to navigate between routes are created with the Link component.

Q11: Describe the process to create an express server with react-router.

A11: To create an Express server with react-router, you configure the server to serve the React application and handle all routes. Use express.static to serve static files and ensure that all routes render the main React app.

Q12: What is the need for React Helmet?

A12: React Helmet is used to dynamically manage the document head, allowing you to change the title, meta tags, and other head elements in response to changes in your React components.

Q13: How to add head tags using React Helmet (No need to write the code; Describe the process).

A13: To add head tags using React Helmet, you import the Helmet component, place it within your React components, and use its properties to modify the document head dynamically.

Q14: How to add head tags using React Helmet? Describe the process with proper code.

A14: ```javascript

import React from 'react';

import { Helmet } from 'react-helmet';

const MyComponent = () => (

<div>

<Helmet>

<title>My Page Title</title>

<meta name="description" content="This is a description." />

</Helmet>

{/\* Rest of your component \*/}

</div>

);

```

Q15: What is the need for a context API?

A15: The context API in React is used to share state across components without explicitly passing props through every level of the component tree. It simplifies state management in larger applications.

Q16: Write two different components created by the context object.

A16: Two components created by the context object are the Provider component, which provides the context value to its descendants, and the Consumer component, which consumes the context value.

Q17: Discuss the process to create a Context for the lists and make it exportable so that the list data can be used everywhere.

A17: To create a context for lists, you use the createContext function, create a Provider component to wrap your application, and export the context so that it can be used in any component that needs access to the list data.

Q18: How can you get data conditionally from the Context for a smaller application but can be inefficient for larger applications?

A18: For a smaller application, you can get data conditionally from the Context by using the useContext hook. However, this approach can be inefficient for larger applications as it triggers re-renders for components consuming the context whenever the context value changes.

Q19: What are React Hooks?

A19: React Hooks are functions that enable functional components to use state and lifecycle features that were previously only available in class components. Common hooks include useState and useEffect.

Q20: Write three rules for hooks.

A20: Three rules for hooks are:

Hooks must be called at the top level of the functional component.

Hooks must be called in the same order in every render.

Only call hooks from React function components or custom hooks.

Q21: Note down any demerits of hooks.

A21: Some demerits of hooks include potential confusion around the order of hooks, the need to ensure they are called consistently, and the initial learning curve for developers transitioning from class components.

Q22: How to Use Higher-Order Components in React? Explain with example code.

A22: A higher-order component (HOC) is a function that takes a component and returns a new component with additional props. Example code:

javascript

Copy code

const withLogger = (WrappedComponent) => {

return class WithLogger extends React.Component {

componentDidMount() {

console.log('Component is mounted.');

}

render() {

return <WrappedComponent {...this.props} />;

}

};

};

const EnhancedComponent = withLogger(MyComponent);

Q23: Discuss useState and useEffect.

A23:

useState: A hook that allows functional components to manage state. It returns an array with the current state value and a function to update it. Example: const [count, setCount] = useState(0);

useEffect: A hook that performs side effects in functional components. It runs after every render and can be used for data fetching, subscriptions, and manual DOM manipulations.

Q24: What is a progressive web application (PWA)?

A24: A Progressive Web Application (PWA) is a type of web application that utilizes modern web technologies to deliver an app-like experience to users. PWAs are characterized by features such as offline support, push notifications, and the ability to install on a user's device.

Q25: Write down some benefits of PWA?

A25: Benefits of PWA include offline functionality, fast loading times, responsiveness, push notifications, and the ability to be installed on users' devices.

Q26: Explain the main features of styled components?

A26: Main features of styled components include the ability to write CSS directly within JavaScript using tagged template literals, dynamic styling based on props, and automatic generation of unique class names.

Q27: How do you apply style to a component using styled-components in React?

A27: Styled-components allow you to apply styles to a component by defining a styled component using the styled function and then using it as a regular React component. Example:

javascript

Copy code

import styled from 'styled-components';

const StyledButton = styled.button`

background-color: blue;

color: white;

`;

const MyComponent = () => (

<div>

<StyledButton>Styled Button</StyledButton>

</div>

);

Q28: What is the purpose of the fallback property in Suspense?

A28: The fallback property in Suspense is used to specify the content that should be displayed while waiting for asynchronous data to resolve. It helps create a better user experience during loading.

Q29: What are higher-order components (HOC)?

A29: Higher-order components (HOC) are functions that take a component and return a new component with additional props or behavior. They are a way to reuse component logic and apply cross-cutting concerns.

Q30: What is the React Suspense Error Boundary?

A30: React Suspense Error Boundary is a component that catches JavaScript errors anywhere in its component tree and logs those errors or displays a fallback UI. It works in conjunction with React Suspense to handle errors during the data fetching process.

Q31: What are the advantages of using React?

A31: Advantages of using React include:

Virtual DOM for efficient updates.

Component-based architecture for reusability.

Declarative syntax for easy code readability.

Large community and ecosystem.

Strong support for building single-page applications.

Q32: What are the limitations of React?

A32: Limitations of React include:

Steeper learning curve for beginners.

JSX might be challenging for some developers.

SEO optimization requires additional effort.

Large file sizes for complex applications.

Frequent updates and changes may lead to version compatibility issues.

Q33: What is JSX?

A33: JSX (JavaScript XML) is a syntax extension for JavaScript used with React. It allows you to write HTML elements and components in a syntax that looks similar to XML or HTML within JavaScript code.

Q34: What are the differences between functional and class components? Explain with example code.

A34:

Functional Component:

javascript

Copy code

const FunctionalComponent = () => {

return <div>Hello, I'm a functional component!</div>;

};

Class Component:

javascript

Copy code

class ClassComponent extends React.Component {

render() {

return <div>Hello, I'm a class component!</div>;

}

}

Functional components are simpler and used for stateless components, while class components can manage state and have lifecycle methods.

Q35: What is rendering in React?

A35: Rendering in React refers to the process of displaying React components on the screen. It involves converting React elements into the DOM elements that users can interact with.

Q36: How many times is render called in React?

A36: The render method in a React component is called whenever the component's state or props change. It is responsible for returning the JSX that represents the component's UI.

Q37: Differentiate between client-side and server-side rendering. Explain with example code.

A37:

Client-Side Rendering (CSR):

javascript

Copy code

// Client-side rendering example

useEffect(() => {

fetch('/api/data')

.then(response => response.json())

.then(data => setData(data));

}, []);

Server-Side Rendering (SSR):

javascript

Copy code

// Server-side rendering example (Next.js)

export async function getServerSideProps() {

const res = await fetch('https://api.example.com/data');

const data = await res.json();

return {

props: { data },

};

}

In CSR, data fetching and rendering occur on the client side, while SSR involves fetching data on the server before rendering.

Q38: How does SSR (server-side rendering) work with React? Explain the steps with proper code.

A38: Server-side rendering in React involves pre-rendering React components on the server and sending the HTML to the client. A popular framework for SSR with React is Next.js. Example steps:

Install Next.js: npm install next react react-dom

Create pages in the "pages" directory.

Use the getServerSideProps function for SSR:

javascript

Copy code

// pages/index.js

export async function getServerSideProps() {

const res = await fetch('https://api.example.com/data');

const data = await res.json();

return {

props: { data },

};

}

const HomePage = ({ data }) => (

<div>{data}</div>

);

export default HomePage;

Q39: What are the benefits of SSR?

A39: Benefits of Server-Side Rendering (SSR) include improved SEO, faster initial page loads, better performance on slow devices, and enhanced social media sharing.

Q40: What is an event handler, and how to use dynamic event handling in a React application? Explain with proper code.

A40: An event handler in React is a function that handles events, such as clicks or keypresses. Dynamic event handling involves passing a function as a prop to a component. Example:

javascript

Copy code

import React from 'react';

const MyButton = ({ onClickHandler }) => (

<button onClick={onClickHandler}>Click me!</button>

);

const App = () => {

const handleClick = () => {

console.log('Button clicked!');

};

return <MyButton onClickHandler={handleClick} />;

};