**What is Memoization in React ?**

Memoization in React is an optimization technique used to improve performance by caching the results of expensive function calls and returning the cached result when the same inputs occur again. This prevents unnecessary re-renders and computations, leading to a more responsive application.

React provides several tools for memoization:

* **React.memo:**A higher-order component that memoizes a functional component. It performs a shallow comparison of the props and re-renders the component only if the props have changed.
* **useMemo:**A hook that memoizes a value. It takes a function and an array of dependencies as arguments. The function is only executed when the dependencies change, and the result is cached and returned on subsequent renders.

**What is Context API?**

Context API is used to pass global variables anywhere in the code without the prop drilling. It helps when there is a need for sharing state between a lot of nested components. It is light in weight and easier to use, to create a context just need to call React.createContext(). No need to install other dependencies or third-party libraries like redux for state management.

**Why is Context API used?**

Context API solves the problem of prop drilling in React. Prop Drilling occurs when data is to be passed between multiple layers before finally sending it to the required component. This makes the application slower. This problem is solved by Context API as it creates global variables to be used throughout the application without any middle components involved. It is also easier to use than React Redux

**Working**

To work with Context API we need React.createContext. It has two properties Provider and Consumer. The Provider acts as a parent it passes the state to its children whereas the Consumer uses the state that has been passed.

**Benefits of Context API over React Redux**

* In Redux we have to manipulate or update multiple files to add even a single feature but in Context it can be done in much lesser lines of code
* One way data binding in React is maintained using Context whereas Redux violates it.
* Multiple stores/contexts can be created using Context whereas Redux creates just a single store

**Error Boundaries in React**

Error boundaries are special **React components** that catch JavaScript errors in their child component tree, log those errors, and display a fallback UI instead of crashing the entire application.

**How Error Boundaries Work**

React components normally do not catch errors in event handlers, asynchronous code, or within themselves. However, **error boundaries catch errors occurring in:**

* **Render phase**
* **Lifecycle methods**
* **Constructor of child components**

**Error boundaries only work for class components** (Functional components must use hooks like useErrorBoundary from external libraries).  
**They do not catch errors in event handlers, asynchronous code, or inside themselves.**  
**You can have multiple error boundaries to isolate failures to specific parts of the UI.**

**Use them to prevent app crashes and provide a better user experience.**

**Custom hooks in React**

Custom hooks in React are reusable functions that encapsulate logic using built-in React hooks (such as useState, useEffect, etc.). They help in abstracting component logic and keeping components clean and readable.

**Why Use Custom Hooks?**

1. **Code Reusability** - Avoids duplication of logic across components.
2. **Separation of Concerns** - Keeps UI and logic separate.
3. **Cleaner Components** - Reduces clutter in functional components.
4. **Stateful Logic Sharing** - Allows sharing logic between components without using higher-order components (HOCs) or render props.

**Best Practices for Custom Hooks**

1. **Always start the function name with use** to ensure React recognizes it as a hook.
2. **Follow the Rules of Hooks** (only call hooks at the top level, never inside loops, conditions, or nested functions).
3. **Keep hooks small and focused** on a single responsibility.
4. **Return only necessary values** (avoid exposing unnecessary internal states).
5. **Use existing hooks within custom hooks** to leverage React’s built-in features.

JWT (JSON Web Token) is a compact, URL-safe token format used for securely transmitting information between parties as a JSON object. It is commonly used for authentication and authorization in web applications.

**Advantages of JWT**

* **Stateless**: No need to store session data on the server.
* **Compact**: Can be easily transmitted in URLs, headers, or cookies.
* **Secure**: Supports various signing algorithms, including HMAC and RSA.

**Common Use Cases**

* API authentication (e.g., OAuth 2.0)
* Single Sign-On (SSO)
* Secure communication between microservices

Styled-components is a popular library for styling React components using tagged template literals. It allows you to write CSS directly within your JavaScript file, keeping styles scoped to specific components.

**Key Features**

1. **Dynamic Props** – Use props to modify styles dynamically.
2. **Scoped Styles** – Avoid global CSS conflicts.
3. **Theming Support** – Works with ThemeProvider for global theming.
4. **Nesting & Pseudo-selectors** – Supports CSS nesting and pseudo-classes.