What is React, and why would you use it?

React is a JavaScript library for building user interfaces, particularly for single-page applications and It allows developers to build reusable UI components manage the state of applications effectively. React is known for its virtual DOM.

**2. What are React components, and how are they classified?**

**Answer**:  
React components are reusable building blocks of a React application. They are classified as:

1. **Functional Components**: These are stateless components that use hooks for state and lifecycle management.
2. **Class Components**: These are stateful components that use lifecycle methods and this.state.

**3. Explain the difference between controlled and uncontrolled components.**

**Answer**:

* **Controlled Components**: Components where form data is handled by React state. The value of an input element is controlled via props.
* **Uncontrolled Components**: Components that store their state internally using the DOM instead of React state.

#### 4. ****What is the Virtual DOM, and how does it work?****

**Virtual DOM is a virtual symbol of the DOM**. But the main difference is that every time, with each change, the virtual DOM gets updated instead of the actual DOM .

#### ****5. How does React’s reconciliation process work?****

**Answer**:  
Reconciliation is the process by which React updates the DOM efficiently.

**6. What are hooks in React, and why are they useful?**

**Answer**:  
Hooks are functions that let you use React state and lifecycle features in functional components.  
Common hooks include:

* **useState**: For state management.
* **useEffect**: For handling side effects like data fetching or subscriptions.
* **useContext**: For consuming context values.

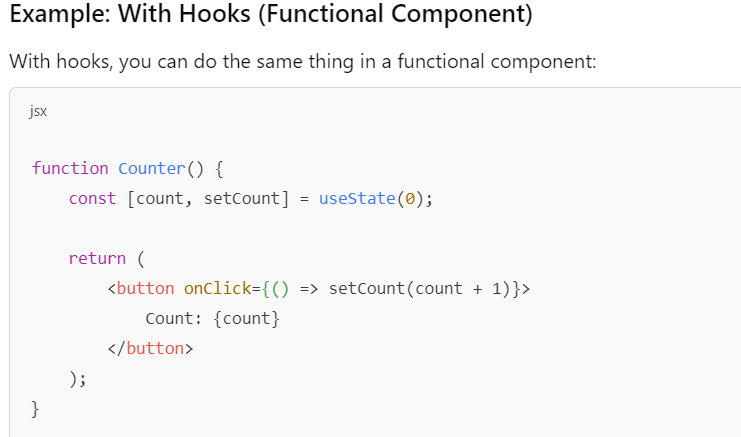
Hooks eliminate the need for class components and make code more reusable and easier to test.

 If you want to manage **state** in a functional component, you use the useState hook.

 If you want to run some code when a component renders or updates, you use the useEffect hook.

Without hook:





#### ****7. What is React Context API, and when should you use it?****

<https://www.geeksforgeeks.org/explain-new-context-api-in-react/>

**Answer**:  
The Context API allows you to share state across components without passing props through every level of the component tree. Use it when you need to manage global state, such as user authentication or theme data, without using a library like Redux.

**8. How would you optimize performance in a React application?**

**Answer**:

* Use **React.memo** to prevent unnecessary re-renders.
* Use **useCallback** and **useMemo** to memoize functions and values.
* Implement **lazy loading** for components using React.lazy and Suspense.
* Optimize large lists using libraries like react-window or react-virtualized.
* Use proper key values in lists for efficient reconciliation.

#### ****9. What is Redux, and how does it integrate with React?****

**Answer**:  
Redux is a state management library that provides a predictable state container for JavaScript apps. It integrates with React using the react-redux library, which provides Provider and connect or useSelector and useDispatch hooks for mapping state and dispatching actions

#### ****10. What are React portals, and when would you use them?****

**Answer**:  
React portals allow rendering children into a DOM node outside of the parent component hierarchy. They are used when a component needs to break out of its parent, such as modals, tooltips, or popups.

**11. How would you test React components?**

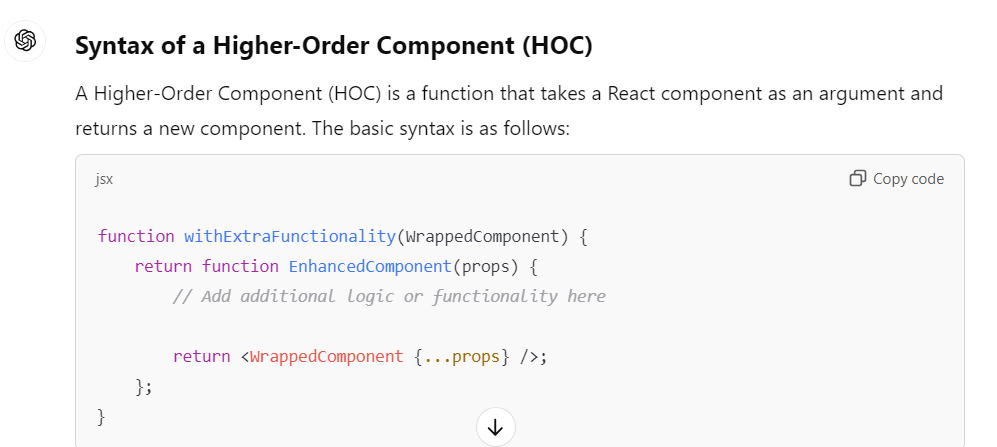
**Answer**:

* Use **Jest** for unit tests and mocking.
* Use **React Testing Library** to test components by interacting with the DOM.
* Write tests for:
  + Component rendering.
  + User interactions.
  + State changes.

#### ****12. What are higher-order components (HOCs)?****

A higher order component function accepts another function as an argument. The **map** function is the best example to understand this.

Example:

1. //Function Creation
2. function add (a, b) {
3. **return** a + b
4. }
5. function higherOrder(a, addReference) {
6. **return** addReference(a, 20)
7. }
8. //Function call
9. higherOrder(30, add) // 50
10. 

**13. Can you explain prop drilling and how to avoid it?**

**Answer**:  
Prop drilling occurs when you pass props through many nested components. You can avoid it by:

* Using **Context API**.
* Using a state management library like **Redux**.

**15. What is the difference between client-side routing and server-side routing?**

**Answer**:

* **Client-Side Routing**: Handled by React Router, it updates the URL and renders the appropriate component without a full page reload.
* **Server-Side Routing**: Involves sending a request to the server for every route change, returning a new HTML page.

## 16. Lifecycle of Components?

Mounting,updating,unmounting

1.mounting has 4 types

constructor() method is called before anything else, when the component is initiated,

The getDerivedStateFromProps() method is called right before rendering the element(s) in the DOM.

The render() method is required, and is the method that actually outputs the HTML to the DOM.

The componentDidMount() method is called after the component is rendered.

This is where you run statements that requires that the component is already placed in the DOM.

Updating()

1. getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()

## Unmounting:

The next phase in the lifecycle is when a component is removed from the DOM, or unmounting as React likes to call it.

React has only one built-in method that gets called when a component is unmounted:

* componentWillUnmount()