Short Answer:

1. What are hooks in React? Why do we use them?

   Hooks are functions inside of function components that allows interaction with state.

2. Explain the following hooks (what it does, what's the syntax). What are their class component equivalents? 4. useContext

   1. useState

   We call useState inside a function component to add some local state to it. It will return a pair: [currentState , updateState]

   useState: initialization will be triggered only once

   updateState: like setState() in class component updateState(newState)

   updateState(currentState=>newState)

   2. useEffect

   It is similar to componentDidUpdate/ componentDidMount/ componentWillUnmount. It detects changes of the component and can be invoked right after.

   3. useRef

   returns a mutable ref object whose current property is initialized to the passed argument (initialValue). The returned object will persist for the full lifetime of the component.

   4. useContext

   useContext accepts a context object (the value returned from React.createContext) and returns the current context value for that context

3. Explain how the useReducer hook works. What is its syntax, and what are some usecases?

   useReducer is usually preferable to useState when you have complex state logic that involves multiple sub-values or when the next state depends on the previous one.

const [state, dispatch] = useReducer(reducer, initialArg, init);

4. What are guard routes? Why do we use them? What are some use-cases?

   Route guards can tell the router whether or not it should allow navigation to a requested route. They make this decision by looking for a true or false return value from a class which implements the given guard interface.

5. What are custom hooks? What is their naming convention? Why do we use them? What are some use-cases? .

   The main reason to write a custom hook is to enhance the code's reusability A custom Hook is a JavaScript function whose name starts with 'use' and may call other Hooks. Every time you use a custom Hook, all state and effects inside of it are fully isolate.

6. What is Redux? Why do we use redux? Explain its structure.

   Redux is an open-source JavaScript library for managing and centralizing application state.

7. Explain the following hooks (what it does, what's the syntax).

   1. useSelector

      useSelector is approximately equivalent to the mapStateToProps argument to connect conceptually. The selector will be run whenever the function component renders. useSelector() will also subscribe to the Redux store, and run your selector whenever an action is dispatched.

      Const {prop}=useSelector((state)=>({stateProperties:state}))

   2. useDispatch

      const [state, dispatch] = useReducer(reducer, { count: 0, number: 1 });
      an alternative of useState.
      Define a reducer first, and call the dispatch function () => dispatch({ type: 'increment' })

   3. useStore

This hook returns a reference to the same Redux store that was passed in to the <Provider> component.

const store = useStore();

const something = store.getState();

8. What is React.memo? How does it differ from useMemo? Explain in terms of syntax and how they work.

   React.memo can boost the performance of a react component by wrapping around the component if it renders the same result given the same props

9. Explain useCallback (what it does, what's the syntax). What are some use-cases?

   useCallback returns a memoized callback function. It will only rerender once its dependencies update. The useCallback and useMemo Hooks are similar. The main difference is that useMemo returns a memoized value and useCallback returns a memoized function.