useEffect

Question 1: Create a React app with a component called DocumentTitle, which takes a title prop. The component should update the document title with the provided title using the useEffect hook.

For example, a functional component called DocumentTitle. It takes a title prop and uses the useEffect hook to update the document title. The effect is triggered whenever the title prop changes. When the component is mounted, it sets the document title to the provided title. If the

title prop changes during the component's lifecycle, the effect will update the document title accordingly.

Question 2: Create a React app with a component called Timer, which displays a timer that increments every second. Use the useEffect hook to implement the timer functionality.

For example, a functional component called Timer. It uses the useEffect hook with an empty dependency array []. This means the effect will only run once when the component is mounted. Inside the effect, we set up a timer using setInterval to increment the seconds state every second. The effect also returns a cleanup function using return, which clears the interval when the component is unmounted, preventing memory leaks.

Question 3: Create a React app with a component called WindowWidth, which displays the current window width. Use the useEffect hook to update the width when the window is resized.

For example, a functional component called WindowWidth. It uses the useEffect hook with an empty dependency array [], so the effect runs only once when the component is mounted. Inside the effect, we add an event listener for the 'resize' event to the window object. When the window is resized, the handleResize function updates the windowWidth state with the new window width. The effect also returns a cleanup function using return, which removes the event listener when the component is unmounted to prevent unnecessary event handling.