Focus on Input Field using `useRef`:

- Write a React component where `useRef` is used to focus on an input field when a button is clicked.

Track Previous State with `useRef`:

- Create a React component that tracks the previous value of a state variable using `useRef` and displays both the current and previous values.

Persisting Values Across Renders:

- Implement a counter using `useRef` to store the count value, ensuring that it persists across renders but does not cause re-renders when updated.

Store Interval ID in `useRef`:

- Write a React component that starts a timer using `setInterval` and stores the interval ID in a `useRef`. The interval should be cleared when the component unmounts.

Use `useRef` to Access a Child Component Method:

- Create a parent component that uses `useRef` to call a method in a child component. The child component should have a method that prints a message to the console.

Conditionally Update `useRef` Value:

- Write a React component where a `useRef` value is updated only when certain conditions are met (e.g., a certain state value changes).

Use `useRef` for Performance Optimization:

- Create a component that uses `useRef` to avoid unnecessary re-renders of a value that doesn't need to trigger a render. For example, track mouse position without causing re-renders.

Handle Form Submission with `useRef`:

- Write a form component that uses `useRef` to store references to form elements and prevent re-renders on value changes.

Toggle Visibility with `useRef`:

- Write a React component that uses `useRef` to toggle the visibility of an element (e.g., show/hide a div when a button is clicked).

Control CSS Animation with `useRef`:

- Create a React component that uses `useRef` to trigger a CSS animation on a DOM element when a button is clicked.