

Best Practices in **React** State Management:



DO

Lift state up to the nearest common ancestor when multiple components need access to the same state.

AVOID

Prop drilling (passing props through multiple layers of components unnecessarily).



[/mulukenm](#)

DO

```
import React, { useState } from 'react';

// Parent component where state is lifted
const ParentComponent = () => {
  const [count, setCount] = useState(0);

  const incrementCount = () => {
    setCount(prevCount => prevCount + 1);
  };

  return (
    <div>
      <h2>Parent Component</h2>
      <p>Count: {count}</p>
      {/* Pass state and setState function to ChildComponent */}
      <ChildComponent count={count} incrementCount={incrementCount} />
    </div>
  );
};

// Child component that receives state and setState function via props
const ChildComponent = ({ count, incrementCount }) => {
  return (
    <div>
      <h3>Child Component</h3>
      <p>Count from Parent: {count}</p>
      <button onClick={incrementCount}>Increment Count</button>
      {/* GrandchildComponent receives count and setState function via props */}
      <GrandchildComponent count={count} />
    </div>
  );
};

// Grandchild component that receives state directly via props
const GrandchildComponent = ({ count }) => {
  return (
    <div>
      <h4>Grandchild Component</h4>
      <p>Count from Parent via Child: {count}</p>
      {/* Render something based on count */}
    </div>
  );
};

export default ParentComponent;
```

**THIS IS BEST PRACTICE - LIFTING
STATE UP:**



[/mulukenm](#)

AVOID

```
import React, { useState } from 'react';

// Component where prop drilling occurs unnecessarily
const TopComponent = () => {
  const [count, setCount] = useState(0);

  const incrementCount = () => {
    setCount(prevCount => prevCount + 1);
  };

  return (
    <div>
      <h2>Top Component</h2>
      <p>Count: {count}</p>
      {/* Prop drilling through multiple layers */}
      <MiddleComponent count={count} incrementCount={incrementCount} />
    </div>
  );
};

// Middle component that passes props down to another component
const MiddleComponent = ({ count, incrementCount }) => {
  return (
    <div>
      <h3>Middle Component</h3>
      <p>Count from Top: {count}</p>
      {/* Prop drilling continues */}
      <BottomComponent count={count} incrementCount={incrementCount} />
    </div>
  );
};

// Bottom component that finally receives props and uses them
const BottomComponent = ({ count, incrementCount }) => {
  return (
    <div>
      <h4>Bottom Component</h4>
      <p>Count from Middle: {count}</p>
      <button onClick={incrementCount}>Increment Count</button>
      {/* More components could continue to receive props unnecessarily */}
    </div>
  );
};

export default TopComponent;
```

PITFALL TO AVOID - PROP DRILLING:



[/mulukenm](#)