React Coding Questions Cheat Sheet

1. Counter Component

```
function Counter() {
  const [count, setCount] = React.useState(0);

  return (
    <>
      <p>{count}</p>
      <button onClick={() => setCount(count + 1)}></button>
      <button onClick={() => setCount(count - 1)}></button>
      <button onClick={() => setCount(0)}> Reset</button>
    </>
  );
}
```

2. Controlled Input (Two-way binding)

```
function InputForm() {
  const [text, setText] = React.useState("");

  return (
    <input
      value={text}
      onChange={(e) => setText(e.target.value)}
      placeholder="Type here"
    />
  );
}
```

3. Todo List (Add/Delete)

```
function TodoList() {
  const [todos, setTodos] = React.useState([]);
  const [task, setTask] = React.useState("");

  const addTodo = () => {
    if (task.trim()) {
      setTodos([...todos, task]);
      setTask("");
    }
  };

  const removeTodo = (index) => {
    setTodos(todos.filter((_, i) => i !== index));
  };

  return (
```

```
    <>
      <input value={task} onChange={(e) => setTask(e.target.value)} />
      <button onClick={addTodo}>Add</button>
      <ul>
        {todos.map((todo, i) => (
          <li key={i}>
            {todo} <button onClick={() => removeTodo(i)}></button>
          </li>
        ))}
      </ul>
    </>
  );
}
```

4. Debounce Input

```
function DebouncedInput() {
  const [text, setText] = React.useState("");
  const [debounced, setDebounced] = React.useState("");

  React.useEffect(() => {
    const timer = setTimeout(() => setDebounced(text), 500);
    return () => clearTimeout(timer);
  }, [text]);

  return (
    <>
      <input onChange={(e) => setText(e.target.value)} />
      <p>Debounced: {debounced}</p>
    </>
  );
}
```

5. useEffect on Prop Change

```
function WatchProp({ value }) {
  React.useEffect(() => {
    console.log("Value changed:", value);
  }, [value]);

  return <p>{value}</p>;
}
```

6. React.memo & useCallback

```
const Button = React.memo(({ onClick }) => {
  console.log("Button rendered");
  return <button onClick={onClick}>Click</button>;
});

function Parent() {
```

```
  const [count, setCount] = React.useState(0);
  const handleClick = React.useCallback(() => {
    console.log("Clicked");
  }, []);

  return (
    <>
      <p>{count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
      <Button onClick={handleClick} />
    </>
  );
}
```

7. Cleanup useEffect

```
useEffect(() => {
  const interval = setInterval(() => {
    console.log("Running...");
  }, 1000);

  return () => {
    clearInterval(interval);
    console.log("Cleaned up!");
  };
}, []);
```

8. Fetch API with Loading/Error

```
function FetchData() {
  const [data, setData] = React.useState(null);
  const [loading, setLoading] = React.useState(true);
  const [error, setError] = React.useState("");

  React.useEffect(() => {
    fetch("https://jsonplaceholder.typicode.com/posts/1")
      .then((res) => res.json())
      .then(setData)
      .catch(() => setError("Failed to load"))
      .finally(() => setLoading(false));
  }, []);

  if (loading) return <p>Loading...</p>;
  if (error) return <p>{error}</p>;
  return <div>{data.title}</div>;
}
```

9. Conditional Rendering

```
{isLoggedIn ? <Dashboard /> : <Login />}
```

10. Toggle Dark/Light Mode

```
function ThemeToggler() {
  const [dark, setDark] = React.useState(false);
  return (
    <div className={dark ? "dark" : "light"}>
      <button onClick={() => setDark(!dark)}>Toggle</button>
    </div>
  );
}
```