1. Sample Question 1

Sample answer content for question 1.

2. Sample Question 2

Sample answer content for question 2.

3. Sample Question 3

Sample answer content for question 3.

4. Sample Question 4

Sample answer content for question 4.

5. Sample Question 5

Sample answer content for question 5.

6. Sample Question 6

Sample answer content for question 6.

7. Sample Question 7

Sample answer content for question 7.

8. Sample Question 8

Sample answer content for question 8.

9. Sample Question 9

Sample answer content for question 9.

10. Sample Question 10

Sample answer content for question 10.

11. Sample Question 11

Sample answer content for question 11.

12. Sample Question 12

Sample answer content for question 12.

13. Sample Question 13

Sample answer content for question 13.

14. Sample Question 14

Sample answer content for question 14.

15. Sample Question 15

Sample answer content for question 15.

16. Sample Question 16

Sample answer content for question 16.

17. Sample Question 17

Sample answer content for question 17.

18. Sample Question 18

Sample answer content for question 18.

19. Sample Question 19

Sample answer content for question 19.

20. Sample Question 20

Sample answer content for question 20.

21. Sample Question 21

Sample answer content for question 21.

22. Sample Question 22

Sample answer content for question 22.

23. Sample Question 23

Sample answer content for question 23.

24. Sample Question 24

Sample answer content for question 24.

25. Sample Question 25

Sample answer content for question 25.

26. Sample Question 26

Sample answer content for question 26.

27. Sample Question 27

Sample answer content for question 27.

28. Sample Question 28

Sample answer content for question 28.

29. Sample Question 29

Sample answer content for question 29.

30. Sample Question 30

Sample answer content for question 30.

31. Sample Question 31

Sample answer content for question 31.

32. Sample Question 32

Sample answer content for question 32.

33. Sample Question 33

Sample answer content for question 33.

34. Sample Question 34

Sample answer content for question 34.

35. Sample Question 35

Sample answer content for question 35.

36. Sample Question 36

Sample answer content for question 36.

37. Sample Question 37

Sample answer content for question 37.

38. Sample Question 38

Sample answer content for question 38.

39. Sample Question 39

Sample answer content for question 39.

40. Sample Question 40

Sample answer content for question 40.

41. Sample Question 41

Sample answer content for question 41.

42. Sample Question 42

Sample answer content for question 42.

43. Sample Question 43

Sample answer content for question 43.

44. Sample Question 44

Sample answer content for question 44.

45. Sample Question 45

Sample answer content for question 45.

46. What is reconciliation in React and how does React decide what to update?

Reconciliation is the process React uses to update the DOM efficiently by comparing the new Virtual DOM tree with the previous one.

- React performs a diffing algorithm that compares elements by type and key.

- If elements are the same type, React updates the changed props.

- If elements differ, React destroys the old tree and creates a new one.

- Keys in lists help React match items between renders.

- This process minimizes direct DOM manipulations, improving performance.

47. Explain controlled components vs uncontrolled components.

- Controlled Components:

  Form inputs where React state controls the input value using `value` and `onChange`.

  Pros: Single source of truth, validation, instant UI updates.

  Example: `const [name, setName] = useState(''); <input value={name} onChange={e => setName(e.target.value)} />`

- Uncontrolled Components:

  Form inputs that manage their own state internally. Use `ref` to access values.

  Pros: Simpler, less code.

  Example: `const nameRef = useRef(); <input ref={nameRef} />`

48. What is React.memo and when should you use it?

React.memo is a Higher-Order Component that memoizes the rendered output of a functional component.

Use it to avoid unnecessary re-renders when props don't change.

Usage: `const MyComponent = React.memo(function MyComponent(props) { ... })`

49. What are custom hooks and why use them?

Custom hooks are functions that use React hooks to encapsulate reusable stateful logic.

Benefits:

- Code reuse and modularity
- Clean components

Example:

```
function useWindowWidth() {
  const [width, setWidth] = useState(window.innerWidth);
  useEffect(() => {
```

```
    const handleResize = () => setWidth(window.innerWidth);
    window.addEventListener('resize', handleResize);
    return () => window.removeEventListener('resize', handleResize);
  }, []);
  return width;
}`
```

50. How would you handle real-time updates in React (like live scores)?

Options:

- WebSockets: Persistent connection for push-based updates.
- SSE (Server-Sent Events): One-way real-time updates.
- Polling: Periodically fetch data.

React Example using WebSocket:

```
`useEffect(() => {
  const ws = new WebSocket('wss://example.com/socket');
  ws.onmessage = (event) => {
    setScores(JSON.parse(event.data));
  };
  return () => ws.close();
}, []);`
```