# Day-19 morning assessment

# 100 mcqs

1. What does ReactDOM.createRoot() return in React 18?

A root object with a .render() method

2.Which method is used to hydrate a server-rendered React application in React 18? hydrateRoot()

3. What is the main purpose of React’s virtual DOM?

To improve performance by reducing direct DOM manipulations

4. When React re-renders a component, it primarily compares:

The new virtual DOM with the old virtual DOM

5. What happens if a React component returns null?

Nothing is rendered for that component

6. Why is findDOMNode() discouraged in modern React?

It breaks encapsulation and strict mode

7. Which DOM API does React use internally for updates?

DOM diffing and patching

8. React batches state updates:

In both synchronous and asynchronous contexts in React 18

9. What is React’s "root" element typically in a standard CRA project?

<div id="root">

10. In React concurrent mode, rendering is:

Interruptible and prioritized

11. What does React’s reconciliation algorithm primarily optimize?

DOM updates by reusing existing nodes

1. What happens if multiple root.render() calls target the same container?

The last one wins, replacing previous renders

Section 2 — Props & Component Patterns (12 Questions)

1. In React, props are:

Immutable inputs passed from parent to child

14. Which of these is the correct way to set a default prop value in a functional component?

Both a and b are valid in modern React

15. When should you use children prop?

To pass React elements or content between component tags

16. What is "prop drilling"?

Passing props deeply through many layers of components unnecessarily

17. Which pattern helps avoid prop drilling?

Higher-order components (HOC)

Context API

Render props

18. What happens if a child component changes its prop value locally?

The local change won’t affect the parent or other components

19. Which is an example of a controlled component in React?

<input value={state} onChange={setState} />

20. In React, defaultProps for functional components:

Still work but destructuring with default values is preferred

21. What is a common downside of the render props pattern?

It causes "wrapper hell" with deeply nested functions

22. Which of the following can be passed via props?

Strings

Objects

Functions

23. If you need to pass data up from a child to a parent component, you should:

Pass a callback function from parent to child and call it in the child

1. Which React pattern allows sharing logic between components without HOCs or render props?

Custom hooks

1. In a functional component, what hook replaces this.state from class components?

useState

26. In class components, which method is called immediately after a component is inserted into the DOM?

componentDidMount

27. In React, state updates are:

Sometimes batched and asynchronous

28. Which lifecycle method is used to clean up subscriptions or timers in a class component?

componentWillUnmount

29. What is the primary difference between state and props?

State is mutable inside the component, props are immutable

30. What is the correct way to update state based on the previous state in functional components?

setState(prev => prev + 1)

31.In React 18, state updates triggered in a setTimeout are:

Batched automatically

32. In class components, which method is called after every render update?

componentDidUpdate

33. What will happen if you call setState with the same value as the current state?

React will skip rendering for performance optimization

34. Which hook is most suitable for managing complex state logic with multiple sub-values? a) useState

useReducer

35. In class components, which lifecycle method is called before a component is removed from the DOM? componentWillUnmount

1. If you need to preserve state between re-renders without causing re-renders when it changes, you should use:

useRef

Section 4 — Hooks Deep Dive (18 Questions)

1. What is the primary rule of hooks in React?

Hooks must be called at the top level of a functional component

38. What does the second argument to useEffect control?

Dependencies that determine when the effect runs

39. Which hook would you use to memoize a computed value?

useMemo

40. Which hook is used for persisting values between renders without triggering re-renders? a)

useRef

41. In useEffect, returning a function serves as:

A cleanup function for when the effect is re-run or the component unmounts

1. Which hook allows you to create a custom hook?

Any built-in hook can be used inside a custom hook

1. What will happen if you forget to provide a dependency array to useEffect?

It will run on every render

44. useLayoutEffect runs:

Before painting the screen, synchronously after DOM mutations

45. What is the main difference between useCallback and useMemo?

useCallback returns a memoized function, useMemo returns a memoized value

46. What is the initial value of a useRef() without arguments?

null

47. Which hook would you use to subscribe to an external data source?

useEffect

48. How can you optimize a component that renders a list of items to avoid unnecessary re-renders?

useMemo and useCallback for handlers and derived data

49. What happens if you update state inside useEffect without a dependency array?

Infinite render loop

50. Which hook would you use for form input management if the form state is complex?

useReducer

51. Can you use hooks inside regular JavaScript functions?

No, only inside functional components or custom hooks

52. What will useEffect(() => {}, []) do?

Run only once after the component mounts

53. How do you prevent useEffect from running on initial render but run on updates?

Check a ref flag inside useEffect

54. Which hook is used to manually trigger a re-render without changing state?

useRef with a dummy state update

useReducer with a dummy dispatch

Section 5 — JSX Advanced Usage (10 Questions)

55. In JSX, which attribute should you use instead of class for CSS classes?

className

56. What is the correct way to insert JavaScript expressions in JSX?

Using curly braces { }

57. What will happen if you return multiple JSX elements without a wrapper

It will throw a syntax error

58. Which JSX element can be used to return multiple elements without adding extra DOM nodes?

<React.Fragment> or <>...</>

59. In JSX, how should inline styles be passed?

As an object with camelCased property names

60. What will {false && <div>Hello</div>} render?

1. Nothing

61. JSX elements must have:

A single parent element

62. Which statement about JSX is true?

JSX compiles to React.createElement calls

63. How can you add a comment inside JSX?

{/\* comment \*/}

64. In JSX, tabIndex attribute is:

Written in camelCase

Section 6 — React Performance Optimizations (12 Questions)

65. Which higher-order component can you use to prevent unnecessary re-renders of functional components?

React.memo

66. In class components, which base class provides a shallow prop and state comparison to avoid re-renders?

PureComponent

67. What does useCallback help with in performance optimization?

Prevents re-creation of functions unless dependencies change

68. Which tool in React DevTools helps analyze component render performance?

Profiler

69. What is the main purpose of useMemo in optimization?

To cache expensive computations

70. What is code splitting in React primarily used for?

Reducing JavaScript bundle size by loading code on demand

71. Which library function is often used for lazy loading components in React?

React.lazy

72. When using React.memo, which scenario might still cause re-renders unnecessarily?

Props being new object or array references each render

73. What does shouldComponentUpdate return to skip a render?

false

74. Which technique helps avoid prop drilling and improves performance in deeply nested components?

Context API

75. What does tree shaking in React builds aim to do?

Remove unused code during bundling

76. How can you prevent a child component from re-rendering when a parent updates?

Wrap it in React.memo and ensure stable props

Section 7 — JavaScript ES6+ Advanced (24 Questions)

77. Which ES6 feature allows unpacking values from arrays or objects into distinct variables?

Destructuring

78. What will console.log([...new Set([1,2,2,3])]) output?

[1, 2, 3]

79. Which statement about let and const is correct?

Both are block-scoped

80. What will be logged?

console.log(0 == '0'); console.log(0 === '0');

true false

81. Which method can be used to merge two objects in ES6?

Object.assign or spread syntax {...obj1, ...obj2}

82. What will typeof NaN return?

"number"

83. Which operator allows default values in destructuring?

=

84. What will be logged?

let x = [1, 2, 3]; let y = x;

y.push(4); console.log(x);

[1, 2, 3, 4]

85. Which ES6 feature allows functions to have variable numbers of arguments?

Rest parameters

86. What will console.log([...'hello']) output?

['h', 'e', 'l', 'l', 'o']

87. Which array method returns a new array without mutating the original? a) push

slice

88. What is the output?

console.log([] + []); console.log([] + {}); console.log({} + []);

"" "[object Object]" "[object Object]"

89. Which keyword is used to create a class in ES6?

class

90. What is the output?

let a = 10; let b = a++; console.log(a, b);

ans:11 10

91. What will console.log('5' - 3) output?

2

92. Which of the following is NOT a JavaScript primitive type?

object

93. What does Object.freeze() do?

Prevents object properties from being changed or added

94. Which method checks if all elements in an array pass a test?

every()

95. What is the result of null == undefined?

true

96. Which statement about arrow functions is true?

They inherit this from their enclosing scope

97. What is the output?

console.log(typeof function(){});

"function"

98. What is the correct syntax for optional chaining in JavaScript?

obj?.prop

obj?.[prop]

99. Which operator is used for nullish coalescing in JavaScript?

??

100. What is the output?

console.log(1 < 2 < 3);

console.log(3 > 2 > 1);

true false