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

Answer: A root instance that includes a .render() function.

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

Answer: hydrateRoot()

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

Answer: To speed things up by limiting direct DOM updates.

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

Answer: The latest virtual DOM to the previous one.

5. What happens if a React component returns null?

Answer: Nothing is added to the DOM for that component.

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

Answer: It can break component encapsulation and strict mode rules.

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

Answer: DOM diffing followed by patching.

8. React batches state updates:

Answer: In both synchronous and asynchronous contexts in React 18.

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

Answer:

10. In React concurrent mode, rendering is:

Answer: It can pause and prioritize work before completing rendering.

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

Answer: Reuse existing DOM nodes to minimize updates.

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

Answer: The latest call overwrites the previous render.

13. In React, props are:

Answer: Immutable values sent from parent to child.

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

Answer: Either using Component.defaultProps or via parameter defaults.

15. When should you use children prop?

Answer: To render nested JSX/content passed between tags.

16. What is "prop drilling"?

Answer: Passing props deeply through many layers of components unnecessarily.

17. Which pattern helps avoid prop drilling?

Answer: By using HOCs, Context API, or render props.

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

Answer: Only the local variable changes — the parent stays the same.

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

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

20. In React, defaultProps for functional components:

Answer: Still works, but default values in destructuring are now preferred.

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

Answer: It can cause nested function “wrapper hell”

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

Answer: Strings, objects, functions.

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

Answer: Give the child a callback from the parent and call it.

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

Answer: Using Custom hooks

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

Answer: useState

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

Answer: componentDidMount

27. In React, state updates are:

Answer: Sometimes batched and asynchronous.

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

Answer: componentWillUnmount

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

Answer: 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?

Answer: setState(prev => prev + 1)

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

Answer: Automatically batched

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

Answer: componentDidUpdate

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

Answer: React will skip rendering for performance optimization

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

Answer: useReducer

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

Answer: componentWillUnmount

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

Answer: useRef

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

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

38. What does the second argument to useEffect control?

Answer: Dependencies that determine when the effect runs

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

Answer: useMemo

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

Answer: useRef

41. In useEffect, returning a function serves as:

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

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

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

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

Answer: It will run on every render

44. useLayoutEffect runs:

Answer: Synchronous to DOM mutations, before painting.

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

Answer: useCallback returns a function, useMemo returns a value.

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

Answer: null

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

Answer: useEffect

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

Answer: Using useMemo/useCallback for handlers & derived data.

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

Answer: An infinite loop.

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

Answer: useReducer

51. Can you use hooks inside regular JavaScript functions?

Answer: b) No, only inside functional components or custom hooks.

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

Answer: Run only once after the component mounts

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

Answer: Use a useRef flag to detect first render.

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

Answer: Use useReducer or a dummy state with useRef.

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

Answer: className

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

Answer: curly braces { }

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

Answer: It will throw a syntax error.

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

Answer: <React.Fragment> or <>...</>

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

Answer: As an object with camelCased property names.

60. What will {false && Hello } render?

Answer: Nothing

61. JSX elements must have:

Answer: One parent container element.

62. Which statement about JSX is true?

Answer: JSX compiles to React.createElement calls

63. How can you add a comment inside JSX?

Answer: {/* comment */}

64. In JSX, tabIndex attribute is:

Answer: Written in camelCase

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

Answer: React.memo

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

Answer: PureComponent

67. What does useCallback help with in performance optimization?

Answer: Prevents re-creation of functions unless dependencies change.

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

Answer: Profiler

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

Answer: To cache expensive computations.

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

Answer: Reducing JavaScript bundle size by loading code on demand.

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

Answer: React.lazy

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

Answer: Props being new object or array references each render.

73. What does shouldComponentUpdate return to skip a render?

Answer: False

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

Answer: Context API

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

Answer: Remove unused code during bundling.

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

Answer: Wrap it in React.memo and ensure stable props.

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

Answer: Destructuring

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

Answer: [1, 2, 3]

79. Which statement about let and const is correct?

Answer: Both are block-scoped.

80. What will be logged?

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

Answer: True

False

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

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

82. What will typeof NaN return?

Answer: "number"

83. Which operator allows default values in destructuring?

Answer: =

84. What will be logged?

let x = [1, 2, 3];

let y = x;

y.push(4);

console.log(x);

Answer: [1, 2, 3, 4]

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

Answer: Rest parameters

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

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

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

Answer: slice

88. What is the output?

console.log([] + []);

console.log([] + {});

console.log({} + []);

Answer: "" "[object Object]" "[object Object]"

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

Answer: class

90. What is the output?

let a = 10;

let b = a++;

console.log(a, b);

Answer: 11 10

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

Answer: 2

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

Answer: object

93. What does Object.freeze() do?

Answer: Prevents object properties from being changed or added.

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

Answer: every()

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

Answer: True

96. Which statement about arrow functions is true?

Answer: They inherit this from their enclosing scope.

97. What is the output?

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

Answer: "function"

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

Answer: obj?.prop and obj?.[prop]

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

Answer:??

100. What is the output?

console.log(1 < 2 < 3);

console.log(3 > 2 > 1);

Answer: True

False