Section 1 — React DOM & Rendering Internals (12 Questions)

1. What does ReactDOM.createRoot() return in React 18?
a) The virtual DOM object
b) A root object with a .render() method
c) The rendered HTML
d) A DOM node reference
Answer: b) A root object with a .render() method

2. Which method is used to hydrate a server-rendered React application in React 18?
a) hydrate()
b) hydrateRoot()
c) createHydrationRoot()
d) renderServer()
Answer: b) hydrateRoot()

3. What is the main purpose of React's virtual DOM?
a) To bypass the browser's DOM API entirely
b) To improve performance by reducing direct DOM manipulations
c) To store CSS styles
d) To compile JSX
Answer: b) To improve performance by reducing direct DOM manipulations

4. When React re-renders a component, it primarily compares:
a) The new real DOM with the old real DOM
b) The new virtual DOM with the old virtual DOM
c) Props with state
d) The server DOM with client DOM
Answer: b) The new virtual DOM with the old virtual DOM

5. What happens if a React component returns null?
a) An error is thrown
b) Nothing is rendered for that component
c) The previous render is reused
d) React renders an empty string in the DOM
Answer: b) Nothing is rendered for that component

6. Why is findDOMNode() discouraged in modern React?
a) It's too slow
b) It breaks encapsulation and strict mode
c) It's not compatible with ES6
d) It's only available in class components

Answer: b) It breaks encapsulation and strict mode

7. Which DOM API does React use internally for updates?
a) innerHTML
b) document.write
c) DOM diffing and patching
d) Full HTML replacement
Answer: c) DOM diffing and patching

8. React batches state updates:
a) Only inside event handlers in React 18
b) In both synchronous and asynchronous contexts in React 18
c) Only for hooks
d) Only in class components
Answer: b) In both synchronous and asynchronous contexts in React 18

9. What is React's "root" element typically in a standard CRA project?
a) <main>
b) <app>
c)
d) <section id="root">
Answer: c)

10. In React concurrent mode, rendering is:
a) Always synchronous
b) Interruptible and prioritized
c) Rendered twice
d) Always delayed until idle
Answer: b) Interruptible and prioritized

11. What does React's reconciliation algorithm primarily optimize?
a) Network requests
b) Component logic
c) DOM updates by reusing existing nodes
d) CSS rendering speed
Answer: c) DOM updates by reusing existing nodes

12. What happens if multiple root.render() calls target the same container?
a) They merge
b) The last one wins, replacing previous renders
c) They both render in sequence
d) React throws an error
Answer: b) The last one wins, replacing previous renders

Section 2 — Props & Component Patterns (12 Questions)

13. In React, props are:
a) Mutable values that control component behavior
b) Immutable inputs passed from parent to child
c) Global variables for the component tree
d) Used only in class components
Answer: b) 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?
a) Component.defaultProps = { propName: value }
b) { propName = value } in function parameters
c) Both a and b are valid in modern React
d) Use setState in useEffect
Answer: c) Both a and b are valid in modern React

15. When should you use children prop?
a) To pass primitive values only
b) To pass React elements or content between component tags
c) To pass event handlers
d) To manage internal state
Answer: b) To pass React elements or content between component tags

16. What is "prop drilling"?
a) Passing props deeply through many layers of components unnecessarily
b) Overwriting default props
c) Using props in multiple components
d) Converting props into state
Answer: a) Passing props deeply through many layers of components unnecessarily

17. Which pattern helps avoid prop drilling?
a) Higher-order components (HOC)
b) Context API
c) Render props
d) All of the above
Answer: d) All of the above

18. What happens if a child component changes its prop value locally?
a) The prop value changes for the parent too
b) The local change won’t affect the parent or other components
c) The entire app re-renders
d) React throws an error
Answer: b) The local change won’t affect the parent or other components

19. Which is an example of a controlled component in React?
a) <input value={state} onChange={setState} />
b) <input /> without any value prop
c) A class component with no state
d) Any component receiving props
Answer: a) <input value={state} onChange={setState} />

20. In React, defaultProps for functional components:
a) Are deprecated for TypeScript projects
b) Still work but destructuring with default values is preferred
c) Cannot be used in React 17+
d) Only work for class components
Answer: b) Still work but destructuring with default values is preferred

21. What is a common downside of the render props pattern?
a) It increases bundle size
b) It causes "wrapper hell" with deeply nested functions
c) It breaks React's lifecycle
d) It cannot work with hooks
Answer: b) It causes "wrapper hell" with deeply nested functions

22. Which of the following can be passed via props?
a) Strings
b) Objects
c) Functions
d) All of the above
Answer: d) All of the above

23. If you need to pass data up from a child to a parent component, you should:
a) Use props.children
b) Pass a callback function from parent to child and call it in the child
c) Use the defaultProps property
d) Modify the parent's state directly from child
Answer: b) Pass a callback function from parent to child and call it in the child

24. Which React pattern allows sharing logic between components without HOCs or render props?
a) Custom hooks
b) Context API
c) Portals
d) Prop drilling
Answer: a) Custom hooks

25. In a functional component, what hook replaces this.state from class components?
a) useEffect
b) useState
c) useReducer
d) useMemo
Answer: b) useState

26. In class components, which method is called immediately after a component is inserted into the DOM?
a) componentWillMount
b) componentDidMount
c) componentDidUpdate
d) shouldComponentUpdate
Answer: b) componentDidMount

27. In React, state updates are:
a) Always synchronous
b) Always asynchronous
c) Sometimes batched and asynchronous
d) Performed directly on the state object
Answer: c) Sometimes batched and asynchronous

28. Which lifecycle method is used to clean up subscriptions or timers in a class component?
a) componentWillUnmount
b) componentDidUnmount
c) componentWillMount
d) componentWillClean
Answer: a) componentWillUnmount

29. What is the primary difference between state and props?
a) State is mutable inside the component, props are immutable
b) Props are mutable, state is immutable
c) Both are mutable
d) Props can only store strings
Answer: a) 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?
a) setState(state + 1)
b) setState(prev => prev + 1)
c) state = state + 1
d) this.state = this.state + 1
Answer: b) setState(prev => prev + 1)

31. In React 18, state updates triggered in a setTimeout are:
a) Batched automatically
b) Not batched
c) Only batched in strict mode
d) Deprecated
Answer: a) Batched automatically

32. In class components, which method is called after every render update?
a) componentDidUpdate
b) componentDidMount
c) shouldComponentUpdate
d) getDerivedStateFromProps
Answer: a) componentDidUpdate

33. What will happen if you call setState with the same value as the current state?
a) React will always re-render
b) React will skip rendering for performance optimization
c) An error will be thrown
d) It will reset the component's lifecycle
Answer: b) React will skip rendering for performance optimization

34. Which hook is most suitable for managing complex state logic with multiple sub-values?
a) useState
b) useReducer
c) useEffect
d) useRef
Answer: b) useReducer

35. In class components, which lifecycle method is called before a component is removed from the DOM?
a) componentWillUnmount
b) componentDidUnmount
c) componentWillDelete
d) componentRemoved
Answer: a) componentWillUnmount

36. If you need to preserve state between re-renders without causing re-renders when it changes, you should use:
a) useState
b) useEffect
c) useRef
d) useCallback
Answer: c) useRef

Section 4 — Hooks Deep Dive (18 Questions)

37. What is the primary rule of hooks in React?
a) Hooks must only be called in class components
b) Hooks must be called at the top level of a functional component
c) Hooks can be called inside loops and conditions freely
d) Hooks must always return JSX
Answer: b) Hooks must be called at the top level of a functional component

38. What does the second argument to useEffect control?
a) The order of effects
b) Dependencies that determine when the effect runs
c) Cleanup logic
d) Component re-render count
Answer: b) Dependencies that determine when the effect runs

39. Which hook would you use to memoize a computed value?
a) useMemo
b) useCallback
c) useRef
d) useEffect
Answer: a) useMemo

40. Which hook is used for persisting values between renders without triggering re-renders?
a) useMemo
b) useState
c) useRef
d) useReducer
Answer: c) useRef

41. In useEffect, returning a function serves as:
a) A cleanup function for when the effect is re-run or the component unmounts
b) A new render trigger
c) An alternative to dependencies
d) A state reset
Answer: a) A cleanup function for when the effect is re-run or the component unmounts

42. Which hook allows you to create a custom hook?
a) Any built-in hook can be used inside a custom hook
b) Only useState and useEffect
c) useCustom
d) useReducer only
Answer: a) 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?
a) It will run only once
b) It will run on every render
c) It will never run
d) It will cause a syntax error
Answer: b) It will run on every render

44. useLayoutEffect runs:
a) Before painting the screen, synchronously after DOM mutations
b) After painting the screen, asynchronously
c) Only on initial mount
d) Only when triggered manually
Answer: a) Before painting the screen, synchronously after DOM mutations

45. What is the main difference between useCallback and useMemo?
a) useCallback returns a memoized value, useMemo returns a function
b) useCallback returns a memoized function, useMemo returns a memoized value
c) They are interchangeable
d) useMemo is faster than useCallback
Answer: b) useCallback returns a memoized function, useMemo returns a memoized value

46. What is the initial value of a useRef() without arguments?
a) undefined
b) null
c) 0
d) ""
Answer: b) null

47. Which hook would you use to subscribe to an external data source?
a) useState
b) useEffect
c) useReducer
d) useMemo
Answer: b) useEffect

48. How can you optimize a component that renders a list of items to avoid unnecessary re-renders?
a) useEffect
b) useMemo and useCallback for handlers and derived data
c) useReducer only
d) Avoid keys in list items
Answer: b) useMemo and useCallback for handlers and derived data

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

a) Nothing
b) Infinite render loop
c) It runs only once
d) It throws an error
Answer: b) Infinite render loop

50. Which hook would you use for form input management if the form state is complex?
a) useState
b) useReducer
c) useMemo
d) useLayoutEffect
Answer: b) useReducer

51. Can you use hooks inside regular JavaScript functions?
a) Yes, anywhere
b) No, only inside functional components or custom hooks
c) Only in async functions
d) Only in arrow functions
Answer: b) No, only inside functional components or custom hooks

52. What will useEffect(() => {}, []) do?
a) Never run
b) Run on every render
c) Run only once after the component mounts
d) Throw an error
Answer: c) Run only once after the component mounts

53. How do you prevent useEffect from running on initial render but run on updates?
a) Check a ref flag inside useEffect
b) Pass null as dependency array
c) Use useLayoutEffect instead
d) It’s not possible
Answer: a) Check a ref flag inside useEffect

54. Which hook is used to manually trigger a re-render without changing state?
a) useEffect
b) useRef with a dummy state update
c) useReducer with a dummy dispatch
d) Both b and c are valid
Answer: d) Both b and c are valid

Section 5 — JSX Advanced Usage (10 Questions)

55. In JSX, which attribute should you use instead of class for CSS classes?
a) css
b) className
c) styleClass
d) classAttr
Answer: b) className

56. What is the correct way to insert JavaScript expressions in JSX?
a) Using double quotes " "
b) Using curly braces { }
c) Using parentheses ( )
d) Using backticks ` `
Answer: b) Using curly braces { }

57. What will happen if you return multiple JSX elements without a wrapper?
a) It will work fine
b) It will throw a syntax error
c) It will wrap them automatically in a div
d) It will render only the first element
Answer: b) It will throw a syntax error

58. Which JSX element can be used to return multiple elements without adding extra DOM nodes?
a) <Wrapper>
b)
c) <React.Fragment> or <>...</>
d) <Multi>
Answer: c) <React.Fragment> or <>...</>

59. In JSX, how should inline styles be passed?
a) As a string like HTML
b) As an object with camelCased property names
c) As an object with kebab-case property names
d) Using CSS modules only
Answer: b) As an object with camelCased property names

60. What will {false && Hello } render?
a) Hello
b) Nothing
c) false
d) An empty string
Answer: b) Nothing

61. JSX elements must have:

a) A single parent element
b) Multiple sibling root elements
c) A key prop always
d) An enclosing function
Answer: a) A single parent element

62. Which statement about JSX is true?
a) JSX is a string representation of HTML
b) JSX compiles to React.createElement calls
c) JSX is required to use React
d) JSX cannot include JavaScript expressions
Answer: b) JSX compiles to React.createElement calls

63. How can you add a comment inside JSX?
a) <!-- comment -->
b) // comment
c) {/* comment */}
d) /** comment */
Answer: c) {/* comment */}

64. In JSX, tabIndex attribute is:
a) Written in lowercase
b) Written in camelCase
c) Not supported
d) Only works in HTML, not JSX
Answer: b) 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?
a) React.memo
b) React.preventRender
c) React.optimize
d) PureComponent
Answer: a) React.memo

66. In class components, which base class provides a shallow prop and state comparison to avoid re-renders?
a) Component
b) PureComponent
c) StatelessComponent

d) ShallowComponent
Answer: b) PureComponent

67. What does useCallback help with in performance optimization?
a) Prevents re-creation of functions unless dependencies change
b) Prevents state changes
c) Reduces bundle size
d) Avoids memory leaks
Answer: a) Prevents re-creation of functions unless dependencies change

68. Which tool in React DevTools helps analyze component render performance?
a) Network tab
b) Profiler
c) Console
d) Lighthouse
Answer: b) Profiler

69. What is the main purpose of useMemo in optimization?
a) To store mutable values
b) To cache expensive computations
c) To manage component state
d) To batch updates
Answer: b) To cache expensive computations

70. What is code splitting in React primarily used for?
a) Reducing JavaScript bundle size by loading code on demand
b) Splitting CSS into separate files
c) Breaking state into smaller pieces
d) Running code in parallel
Answer: a) Reducing JavaScript bundle size by loading code on demand

71. Which library function is often used for lazy loading components in React?
a) React.lazy
b) React.load
c) React.defer
d) React.split
Answer: a) React.lazy

72. When using React.memo, which scenario might still cause re-renders unnecessarily?
a) Props being primitive values
b) Props being new object or array references each render
c) Component having no props
d) Component having static content
Answer: b) Props being new object or array references each render

73. What does shouldComponentUpdate return to skip a render?
a) null
b) false
c) undefined
d) 0
Answer: b) false

74. Which technique helps avoid prop drilling and improves performance in deeply nested components?
a) Context API
b) Inline styles
c) Fragment usage
d) JSX comments
Answer: a) Context API

75. What does tree shaking in React builds aim to do?
a) Remove unused code during bundling
b) Optimize component rendering
c) Reduce API calls
d) Cache state
Answer: a) Remove unused code during bundling

76. How can you prevent a child component from re-rendering when a parent updates?
a) Wrap it in React.memo and ensure stable props
b) Always use useEffect
c) Call stopPropagation
d) Avoid using keys in lists
Answer: a) 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?
a) Spread syntax
b) Destructuring
c) Rest parameters
d) Template literals
Answer: b) Destructuring

78. What will console.log([...new Set([1,2,2,3])]) output?
a) [1, 2, 2, 3]
b) [1, 2, 3]
c) {1, 2, 3}

d) Error
Answer: b) [1, 2, 3]

79. Which statement about let and const is correct?
a) Both are function-scoped
b) Both are block-scoped
c) let is block-scoped, const is function-scoped
d) const cannot be used with objects
Answer: b) Both are block-scoped

80. What will be logged?

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

a) true false
b) false true
c) true true
d) false false
Answer: a) true false

81. Which method can be used to merge two objects in ES6?
a) Object.assign or spread syntax {...obj1, ...obj2}
b) concat
c) merge()
d) combine()
Answer: a) Object.assign or spread syntax {...obj1, ...obj2}

82. What will typeof NaN return?
a) "number"
b) "NaN"
c) "undefined"
d) "object"
Answer: a) "number"

83. Which operator allows default values in destructuring?
a) =
b) ||
c) ??
d) &&
Answer: a) =

84. What will be logged?

```
let x = [1, 2, 3];
let y = x;
y.push(4);
console.log(x);
```

a) [1, 2, 3]
b) [1, 2, 3, 4]
c) [4]
d) Error
Answer: b) [1, 2, 3, 4]

85. Which ES6 feature allows functions to have variable numbers of arguments?
a) Spread syntax
b) Rest parameters
c) Default parameters
d) Arrow functions
Answer: b) Rest parameters

86. What will console.log([...'hello']) output?
a) ['hello']
b) ['h', 'e', 'l', 'l', 'o']
c) "hello"
d) Error
Answer: b) ['h', 'e', 'l', 'l', 'o']

87. Which array method returns a new array without mutating the original?
a) push
b) splice
c) slice
d) sort
Answer: c) slice

88. What is the output?

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

a) "" "[object Object]" "[object Object]"
b) [] {} {}
c) undefined undefined undefined
d) Error
Answer: a) "" "[object Object]" "[object Object]"

89. Which keyword is used to create a class in ES6?
a) class
b) function
c) constructor
d) prototype
Answer: a) class

90. What is the output?

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

a) 10 10
b) 11 10
c) 11 11
d) 10 11
Answer: b) 11 10

91. What will console.log('5' - 3) output?
a) "53"
b) 2
c) "2"
d) Error
Answer: b) 2

92. Which of the following is NOT a JavaScript primitive type?
a) boolean
b) string
c) object
d) symbol
Answer: c) object

93. What does Object.freeze() do?
a) Prevents object properties from being changed or added
b) Deletes all object properties
c) Makes the object immutable recursively
d) Converts object to JSON
Answer: a) Prevents object properties from being changed or added

94. Which method checks if all elements in an array pass a test?
a) every()
b) some()
c) filter()

d) map()
Answer: a) every()

95. What is the result of null == undefined?
a) true
b) false
c) Error
d) undefined
Answer: a) true

96. Which statement about arrow functions is true?
a) They have their own this binding
b) They inherit this from their enclosing scope
c) They must always return a value
d) They can be used as constructors
Answer: b) They inherit this from their enclosing scope

97. What is the output?

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

a) "object"
b) "function"
c) "method"
d) "callable"
Answer: b) "function"

98. What is the correct syntax for optional chaining in JavaScript?
a) obj?.prop
b) obj.?prop
c) obj::prop
d) obj?.[prop]
Answer: Both a) and d)

99. Which operator is used for nullish coalescing in JavaScript?
a) ??
b) ||
c) &&
d) ?:
Answer: a) ??

100. What is the output?

```
console.log(1 < 2 < 3);
```

```
console.log(3 > 2 > 1);
```

a) true true
b) true false
c) false false
d) false true
Answer: b) true false