Ashish A ashishaj336@gmail.com +91 7624901467

---

1. What is React?

- React is a popular JavaScript library created by Facebook for building user interfaces, especially for single-page applications. It lets you create reusable UI components, which makes your code cleaner and easier to manage.

2. What is JSX?

- JSX stands for JavaScript XML. It allows you to write HTML-like syntax in your JavaScript code, which looks pretty neat and is easier to understand. Essentially, it’s a way to describe what the UI should look like.

3. What are props in React?

- Props, short for properties, are how data gets passed from a parent component to a child component in React. They act like function arguments, allowing you to customize components without changing their internal logic.

4. How does the Virtual DOM work?

- The Virtual DOM is like a lightweight copy of the actual DOM. When something changes in a component, React updates the Virtual DOM first, then checks what has changed compared to the previous version. It only updates the parts of the real DOM that need changing, which makes the app faster.

5. What are stateless components?

- Stateless components, also called functional components, don’t manage their own state. They simply receive props and render the UI based on those props. If they don’t need to handle any internal data, you’d typically use a stateless component.

6. What are stateful components?

- Stateful components are class components that manage their own state. They can keep track of data that may change over time, like user input or fetched data.

7. Why should component names start with a capital letter?

- In React, starting component names with a capital letter help distinguish them from regular HTML tags. This makes it clear which are built-in elements and which are user-defined components.

8. Why can't you update props in React?

- Props are meant to be read-only. A child component shouldn't change the props it receives from its parent because that can lead to confusing behavior and makes it hard to track data. Instead, the parent component can change the props it sends down.

9. Is it possible to use React without JSX?

- Yes, you can use React without JSX. You can create elements using React.createElement(), but most people prefer JSX because it's more intuitive and visually appealing.

10. What is JSON and its common operations?

- JSON (JavaScript Object Notation) is a lightweight data format that's easy for humans to read and write. You often use JSON.stringify() to convert JavaScript objects to JSON strings and JSON.parse() to turn JSON strings back into JavaScript objects.

11. What is the difference between slice and splice?

- slice() creates a copy of a portion of an array without changing the original array. On the other hand, splice() modifies the original array by adding or removing elements at a specific index.

12. How do you compare Object and Map?

- Objects are collections of key-value pairs where keys are usually strings. Maps can have keys of any data type and remember the order of insertion. Maps also provide more useful methods than Objects for managing data.

13. What is the difference between == and === operators?

- The == operator checks for equality but allows type conversion, while === checks for strict equality, meaning both the value and the type must match. It's generally best to use === to avoid unexpected results from type coercion.

14. What are lambda expressions or arrow functions?

- Arrow functions are a shorter way to write function expressions in JavaScript. They don't have their own this, which makes them handy when you want to preserve the context from the surrounding code.

Code:

```
const add = (a, b) => a + b;
```

15. What is the currying function? Give an example.

- Currying is a technique where a function that takes multiple arguments is transformed into a series of functions that each take a single argument. It allows you to break down the process of providing arguments.

Code:

```
function add(a) {

    return function(b) {

        return a + b;

    };

   }
   const add5 = add(5);

   console.log(add5(6));
```

Outputs: 11

16. What is the difference between let and var?

- let is scoped to the block in which it is defined and cannot be redeclared within that block, whereas var is function-scoped and can be redeclared. Generally, it's recommended to use let for better control over variable scope.

17. What is the Temporal Dead Zone?

- The Temporal Dead Zone (TDZ) refers to the period from the start of a block until a variable declared with let or const is initialized. Accessing it during this time will give you a ReferenceError.

18. What is an IIFE (Immediately Invoked Function Expression)? Give an example.

- An IIFE is a function that runs as soon as it's defined. It creates a new scope to prevent polluting the global namespace.

```
(function() {

    console.log("I am an IIFE!");

   })();
```

19. What is Hoisting?

- Hoisting is a JavaScript feature where variable and function declarations are moved to the top of their enclosing scope during the compile phase. This means you can use functions before you declare them.

20. What are closures? Give an example.

- Closures allow a function to remember its lexical scope, even when the function is executed outside that scope. This is useful for creating private variables.

```
function outer() {
    let count = 0;
    return function inner() {
        count++;
        return count;
    };
   }
   const increment = outer();
   console.log(increment()); // Outputs: 1
   console.log(increment()); // Outputs: 2
```

21. What are the differences between cookie, local storage, and session storage?

- Cookies are small pieces of data sent to the server with every HTTP request. Local Storage allows you to store data with no expiration, while Session Storage stores data for the duration of a page session and is cleared when the tab is closed.

22. What is the purpose of double exclamation?

- The double exclamation mark (!!) converts a value to a boolean. The first ! negates the value, and the second ! turns it back to a boolean.

23. What is the difference between null and undefined?

- null is a value that means “no value” or “empty,” while undefined means a variable has been declared but hasn’t been assigned a value yet.

24. What is eval? Give an example.

- eval is a JavaScript function that evaluates a string as code. It’s best avoided because it can lead to security issues.

```
const result = eval("2 + 2");

console.log(result); // Outputs: 4
```

25. What is the difference between window and document?

- window is the global object representing the browser window, while document is a property of the window that represents the HTML document currently loaded.

26. What is NaN property?

- NaN means "Not-a-Number" and indicates a value that isn’t a legal number, usually as a result of invalid arithmetic operations.

27. Write a function that would allow you to do this: multiply (5)(6)

- We can achieve this using currying:

```
const multiply = a => b => a * b;

console.log(multiply(5)(6)); // Outputs: 30
```

28. What will the following code output?

- For the comparison 0.1 + 0.2 === 0.3, it evaluates to false because of how floating-point arithmetic works in JavaScript.

29. Define ways to empty an array in JavaScript.

- Few ways to empty an array:

1. Set the length to 0:

```
arr.length = 0;
```

2. Use the splice method:

```
arr.splice(0, arr.length);
```

3. Reassign an empty array:

```
arr = [];
```

30. Write a one-liner function in JavaScript to remove duplicate elements from an array.

```
const removeDuplicates = arr => [...new Set(arr)];
```

31. What will be the output of the below code?

```
const array = [10, 20, 30, 40];

const result = array.map((num) => num / 2).filter((num) => num >= 15);

console.log(result);
```

Output: [15, 20]

32. Find the issue with the below code snippet:

```
setTimeout(function) { console.log("This will be executed after 3 seconds");},
3000;

clearTimeout();
```

-The correct usage of setTimeout should be:

```
setTimeout(function() {

    console.log("This will be executed after 3 seconds");

}, 3000);
```

Remember that clearTimeout needs a timer ID to cancel a timeout.

33. What will happen if you try to access a variable before it is declared?

- You'll get a ReferenceError if you try to access a variable declared with let or const before it's declared. For a var, you'll just get undefined because of hoisting.

34. Can you explain how the spread operator works in JavaScript?

- The spread operator (...) lets you expand an iterable (like an array) into individual elements. It's handy for combining arrays or passing multiple arguments to functions.

```
  const arr1 = [1, 2, 3];

  const arr2 = [...arr1, 4, 5]; // arr2 becomes [1, 2, 3, 4, 5]
```

35. What is the output of the following code?

```
const x = 5;

const y = x++;

console.log(y);
```

Output: 5

36. How does the useMemo hook optimize performance in React? Give an example.

- The useMemo hook helps memoize expensive calculations, so they're only recalculated if their dependencies change. This prevents unnecessary calculations and speeds up your app.

Example:

```
import React, { useMemo } from 'react';

   const Component = ({ items }) => {

    const total = useMemo(() => {

        return items.reduce((sum, item) => sum + item, 0);

    }, [items]);

    return <div>Total: {total}</div>;

   };
```

37. What will be the output of the following React component?

```
   function App() {

      return <div>{null}</div>;

  }
```

Output: An empty <div>

38. How would you describe the useEffect hook in React?

- The useEffect hook lets you perform side effects in functional components—like fetching data, directly manipulating the DOM, or setting up subscriptions. It runs after the component renders and can be controlled by specifying dependencies.

39. What is the output of the following code?

```
const arr = [1, 2, 3];

arr.length = 0;

console.log(arr);
```

Output: []

40. What is the difference between functional and class components in React?

- Functional components are simpler and typically use hooks to manage state and lifecycle methods. Class components are a bit more complex and use class syntax with this to manage state and lifecycle events.