Mohammed Rayyan S

STANCH SOLUTIONS PRIVATE LIMITED

JAYANAGAR 4TH T BLOCK,

BENGALURU.

React Front End Test

1. What is React?

React is a JavaScript library for building user interfaces, especially single-page applications. It allows developers to create reusable UI components that manage their state. React is maintained by Facebook and uses a component-based architecture.

2. What is JSX?

JSX (JavaScript XML) is a syntax extension for JavaScript that resembles HTML. It allows you to write HTML-like code within JavaScript. Under the hood, JSX is compiled into JavaScript React.createElement calls.

3. What are props in React?

Props are read-only attributes that allow passing data from a parent component to a child component. They enable data flow in React and help make components dynamic.

4. How does the Virtual DOM work?

The Virtual DOM is a lightweight representation of the actual DOM. When the state of a component changes, React updates the Virtual DOM first, compares it with the previous version via diffing, and then efficiently updates only the changed parts in the real DOM ,a process called reconciliation.

5. What are stateless components?

Stateless components are React components that do not manage or have their own state. They are typically functional components, focusing on presenting data passed through props.

6. What are stateful components?

Stateful components maintain their own internal state and are usually class components or functional components that use hooks like useState. They can track data over time and respond to user interactions.

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

In React, component names must start with a capital letter to distinguish them from native HTML tags. React treats lowercase names as built-in DOM tags.

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

Props are immutable in React. This ensures one-way data flow and predictability in components. If you need to change data, you should use state within the component.

9. Is it possible to use React without JSX?

Yes, React can be used without JSX. However, without JSX, you would need to write JavaScript code using React.createElement, which can be much more harder to read.

10. What is JSON and its common operations?

JSON (JavaScript Object Notation) is a lightweight data-interchange format. Common operations include:

- Parsing a JSON string: JSON.parse()
- Stringifying an object: JSON.stringify()

11. What is the difference between slice and splice?

- slice(): Returns a shallow copy of an array from start to end (without modifying the original array).
- splice(): Modifies the array in place by adding/removing elements.

12. How do you compare Object and Map?

- Object: Keys are strings or symbols. Properties are unordered.
- Map: Keys can be of any data type (including functions, objects, etc.), and keys maintain insertion order.

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

- = = performs type coercion automatic conversion of a value from one data type to another,before comparing (loose equality).
- = = = does not perform type coercion (strict equality), ensuring both value and type are the same.

14. What are lambda expressions or arrow functions?

Arrow functions (()=>) are a more concise way of writing functions in JavaScript. They do not bind their own this and cannot be used as constructors.

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

Currying transforms a function with multiple arguments into a sequence of functions, each with a single argument.

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

console.log(add(2)(3)); // 5
```

16. What is the difference between let and var?

- var is function-scoped, allows redeclaration, and hoists with undefined initialization.
- let is block-scoped, does not allow redeclaration, and is not initialized during hoisting (TDZ).

17. What is the Temporal Dead Zone?

The Temporal Dead Zone (TDZ) is the period between entering a scope and the variable being declared with let or const, during which accessing the variable will result in a ReferenceError.

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

An IIFE is a function that is executed immediately after it is defined.

```
(function() {
        console.log("IIFE executed!");
})();
```

19. What is Hoisting?

Hoisting refers to JavaScript's behavior of moving variable and function declarations to the top of their scope during the compile phase.

20. What are closures? Give an example.

A **closure** is a function that retains access to its outer scope's variables even after the outer function has returned.

```
function outer() {
   let count = 0;
   return function() {
      count++;
      return count;
   };
```

```
}

const counter = outer();

console.log(counter()); // 1
```

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

- Cookies: Can store small amounts of data, have expiration times, and are sent with every HTTP request.
- Local storage: Persistent storage on the client that remains until manually cleared.
- Session storage: Similar to local storage, but data is cleared when the session (or tab) is closed.

22. What is the purpose of double exclamation?

Double exclamation (!!) is used to convert a value to its boolean equivalent.

23. What is the difference between null and undefined?

- null: Represents a deliberately set empty value.
- undefined: Indicates a lack of value or an uninitialized variable.

24. What is eval? Give an example.

eval() executes a string as JavaScript code.

Example: eval("console.log(2 + 2)"); // 4

25. What is the difference between window and document?

- window: Represents the global object that contains everything, including document.
- document: Represents the DOM (HTML document) loaded in the window.

26. What is NaN property?

NaN (Not-a-Number) is a property indicating a value that is not a legal number.

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

```
function multiply(a) {

  return function(b) {

    return a * b; // Return the product of a and b
```

```
  };

}

const result = multiply(5)(6); // Calling the function

console.log(result); // Output: 30
```

28.What will the following code output? 0.1 + 0.2 === 0.3

The expression 0.1 + 0.2 === 0.3 will give the output false in JavaScript.

This is due to the way floating-point arithmetic works in most programming languages, including JavaScript. The numbers 0.1 and 0.2 cannot be precisely represented in binary, leading to small rounding errors when performing arithmetic operations.

When you calculate 0.1 + 0.2, the result is not exactly 0.3 but rather a number very close to 0.3, specifically 0.30000000000000004

29. Define ways to empty an array in JavaScript.

- array.length = 0;
- array = [ ];

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

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

31. What will be the output of the below code: const array = [10, , 30, 40]; const result = array map((num) => num / 2).filter((num) => num >= 15); console.log(result);

[15, 20] Final output
Explanation:

- The array is declared as const array = [10, , 30, 40];.
- Here, the second element is an **empty slot**, which means it has a value of undefined.
- The map method is called on the array: array.map((num) => num / 2).
- This will divide each element by 2:
    - 10 / 2 results in 5.
    - The second element is undefined, so undefined / 2 also results in NaN (Not a Number).
    - 30 / 2 results in 15.
    - 40 / 2 results in 20.

Therefore, after the map operation, the new array will be:
[5, NaN, 15, 20]

- The filter method is called: .filter((num) => num >= 15).

- This will keep only the elements that are greater than or equal to 15:
    - 5 is less than 15 and will be filtered out.
    - NaN is not a number, so it will also be filtered out.
    - 15 is equal to 15, so it will be kept.
    - 20 is greater than 15, so it will also be kept.

Therefore, after the filter operation, the resulting array will be: [15, 20]

32. Find the issue with the below code snippet: setTimeout(function) { console.log("This will be executed after 3 seconds"); }, 3000); clearTimeout();

Summary of the Issues:

1. Incorrect function syntax for setTimeout.
2. Improper use of clearTimeout without a timer ID. It must be provided with the identifier returned from setTimeout to clear a specific timeout.

Corrected Code:

```
const timerId = setTimeout(function() {
  console.log("This will be executed after 3 seconds");
}, 3000);
// If you want to clear the timeout, use:
clearTimeout(timerId); // This will cancel the timeout
```

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

- If the variable is declared using var, it will be hoisted to the top and initialized with undefined, so accessing it before declaration will return undefined.
- If the variable is declared using let or const, it will be in the Temporal Dead Zone (TDZ) and accessing it before declaration will throw a ReferenceError.

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

The **spread operator** (...) allows you to **spread** the elements of an iterable (like arrays or objects) into individual elements.

In arrays: Spreading elements into a new array.

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

In objects: Spreading properties into a new object.

```
const obj1 = { a: 1, b: 2 };
const obj2 = { ...obj1, c: 3 }; // { a: 1, b: 2, c: 3 }
```

35. What is the output of the following code? const x = 5; const y = x++; console.log(y);

The final output of the code will be: 5

Explanation:

Here, x is declared and initialized with the value 5.

The x++ uses the postfix increment operator. This means that the current value of x (which is 5) will be assigned to y before x is incremented.

After this line executes, y will be 5, and then x will be incremented to 6.

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

useMemo is a React Hook that memorizes the result of a function and only recalculates it when its dependencies change. It helps optimize performance by avoiding unnecessary recalculations in functional components.
Example:

```
const memoizedValue = useMemo(() => {
  return expensiveComputation(someVar);
}, [someVar]);
```

37. What will be the output of the following React component? function App() { return {null} / ; }

The output of the component App is an empty element.

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

The useEffect hook in React allows you to perform side effects in functional components, such as:

- Fetching data
- Directly manipulating the DOM
- Setting up timers or intervals

39. What is the output of the following code? const arr = [1, 2, 3 ; .length = 0; console.log(arr);

The output of the corrected code will be an empty array: [].
Explanation:

- The App function is a functional component that returns a JSX element.
- Inside the , we have {null}.
- When React encounters null (or undefined) inside JSX, it simply ignores it and does not render anything for that expression.
- As a result, the will be rendered, but it will be empty because the {null} evaluates to nothing.

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

Functional Components:

- These are plain JavaScript functions that return JSX. They can accept props as an argument.
- They are also known as stateless components because they do not maintain their own state (prior to React 16.8).

Class Components:

- These are ES6 classes that extend from React.Component. They must contain a render() method that returns JSX.
- Class components can hold and manage their own state and lifecycle methods.