Tech Mahindra- Frontend Developer

Interview Process

Round 1: Telephonic Round
Round 2: Assignment
Round 3: Coding Round
Round 4: HR Round

Interview Questions

1. Let, var, const in JavaScript
2. Error boundary and how to handle
3. Event.bind and event.property
4. Use of never in typescript
5. What hooks have you used?
6. What is the difference between use reducer and use state?
7. Why do we use state and props?
8. How will you pass data from parent to child?
9. Write your own custom hook?
10. Explain map filter and reduce functions
11. How can you clone an object?
12. What are the three stages of event propagation and how to handle it?
13. How to handle errors in your react application?
14. Write code to call fake api and display the title where category is “some category “
15. How do you prioritize tasks when faced with multiple deadlines?
16. Describe a situation where you had to work as part of a team to achieve a goal.
17. How do you handle receiving feedback on your work, and what steps do you take to improve based on feedback?
18. How do you implement responsive design principles in your projects?
19. Have you worked with CSS frameworks like Bootstrap or Foundation? If so, can you provide an example of how you've used them to create a responsive layout?
20. What is responsive web design, and why is it important?

Solutions

Q1: Let, var, const in JavaScript
A1: `let`, `var`, and `const` are all used to declare variables in JavaScript. `var` is function-scoped and can be redeclared, leading to potential issues in larger codebases. `let` is block-scoped, providing better control over variable scope and cannot be redeclared within the same scope. `const` is also block-scoped but is used to declare constants, meaning the variable cannot be reassigned after its initial assignment.

Q2: Error boundary and how to handle
A2: Error boundaries are React components that catch JavaScript errors anywhere in their child component tree, log those errors, and display a fallback UI instead of crashing the entire application. They are implemented using `componentDidCatch` and `getDerivedStateFromError` lifecycle methods. Error boundaries are useful for handling runtime errors and improving the resilience of React applications.

Q3: Event.bind and event.property
A3: `event.bind` refers to binding a function to a specific `this` context in event handling, ensuring that `this` refers to the correct object when the event is triggered. `event.property` usually refers to properties of the event object that provide details about the event, such as `event.target` or `event.type`. Binding ensures the correct context, while event properties give you the necessary details to handle the event appropriately.

Q4: Use of never in TypeScript
A4: The `never` type in TypeScript represents values that never occur. It is used for functions that never return (e.g., functions that throw exceptions or run indefinitely) or when you need to assert that a certain case should be impossible. It is useful in exhaustive type checks, ensuring that all code paths are correctly handled.

Q5: What hooks have you used?
A5: Common hooks I have used include `useState` for managing state, `useEffect` for side effects, `useContext` for accessing context values, `useReducer` for complex state logic, `useMemo` for memoization, and `useCallback` to memoize functions. Custom hooks are also useful for extracting reusable logic from components.

Q6: What is the difference between useReducer and useState?
A6: `useState` is a basic hook used to manage simple state in a component, while `useReducer` is more suited for complex state logic that involves multiple sub-values or state transitions based on actions. `useReducer` provides a way to manage state in a more predictable and centralized manner, similar to how Redux works.

Q7: Why do we use state and props?
A7: In React, `state` is used to manage the internal, dynamic data of a component that may change over time, triggering re-renders. `props` are used to pass data and behavior from parent components to child components, allowing for communication and data sharing between components in a unidirectional flow.

Q8: How will you pass data from parent to child?
A8: Data is passed from a parent component to a child component via `props`. The parent component includes the child component in its JSX and provides data as attributes, which the child component can then access via its `props` object.

Q9: Write your own custom hook?
A9: Here's a simple custom hook that tracks the window width:

```
import { useState, useEffect } from 'react';

function useWindowWidth() {
    const [width, setWidth] = useState(window.innerWidth);

    useEffect(() => {
        const handleResize = () => setWidth(window.innerWidth);
        window.addEventListener('resize', handleResize);
        return () => window.removeEventListener('resize',
handleResize);
    }, []);

    return width;
}
```

This hook can be used in any component to get the current window width and update it when the window is resized.

Q10: Explain map, filter, and reduce functions
A10: `map`, `filter`, and `reduce` are array methods in JavaScript. `map` creates a new array by applying a function to each element of the original array. `filter` creates a new array containing only the elements that pass a specified test. `reduce` accumulates array elements into a single value based on a reducer function, which processes each element in turn.

Q11: How can you clone an object?
A11: An object can be cloned in JavaScript using methods like `Object.assign({}, obj)` or the spread operator `{...obj}` for shallow cloning. For deep cloning, where nested objects are also cloned, methods like `JSON.parse(JSON.stringify(obj))` or libraries like Lodash (`_.cloneDeep(obj)`) are used.

Q12: What are the three stages of event propagation and how to handle it?
A12: The three stages of event propagation are **Capturing** (events are captured from the root down to the target element), **Target** (the event reaches the target element), and **Bubbling** (events bubble back up from the target element to the root). You can handle these stages using event listeners with options like `{ capture: true }` for capturing or allowing bubbling by default.

Q13: How to handle errors in your React application?
A13: Errors in React applications can be handled using Error Boundaries for catching runtime errors in the UI. Additionally, try-catch blocks can be used for handling asynchronous code errors, and logging services like Sentry can be integrated for monitoring and reporting errors.

Q14: Write code to call a fake API and display the title where category is "some category"
A14:

```
import React, { useState, useEffect } from 'react';

function App() {
    const [title, setTitle] = useState('');

    useEffect(() => {
        fetch('https://jsonplaceholder.typicode.com/posts')
            .then(response => response.json())
            .then(data => {
                const post = data.find(item => item.category === 'some
category');
                if (post) setTitle(post.title);
            })
            .catch(error => console.error('Error:', error));
    }, []);

    return <h1>{title}</h1>;
}

export default App;
```

Q15: How do you prioritize tasks when faced with multiple deadlines?
A15: When faced with multiple deadlines, I prioritize tasks based on their urgency and importance, breaking them down into smaller, manageable steps. I focus on high-impact tasks first, set clear deadlines for each task, and adjust priorities as needed. Time management techniques, such as the Eisenhower Matrix or task batching, help me stay organized and meet deadlines effectively.

Q16: Describe a situation where you had to work as part of a team to achieve a goal.
A16: You could answer this in this format → In a recent project, our team collaborated to build a complex web application under tight deadlines. We divided tasks based on each member's strengths, communicated regularly to ensure alignment, and supported each other through code reviews and pair programming. This collaborative effort led to the successful and timely delivery of the project.

Q17: How do you handle receiving feedback on your work, and what steps do you take to improve based on feedback?
A17: I view feedback as an opportunity for growth. When I receive feedback, I listen carefully, seek clarification if needed, and reflect on how it can be applied to improve my work. I then make a plan to implement the suggested changes and monitor the results, continuously iterating to enhance my skills and output.

Q18: How do you implement responsive design principles in your projects?
A18: I implement responsive design by using fluid grids, flexible images, and media queries in CSS. I ensure that the layout adapts to different screen sizes by designing with a mobile-first approach and testing across multiple devices. Additionally, I utilize CSS frameworks like Bootstrap or Tailwind for consistent and scalable responsive design.

Q19: Have you worked with CSS frameworks like Bootstrap or Foundation? If so, can you provide an example of how you've used them to create a responsive layout?
A19: Yes, I’ve worked with Bootstrap extensively. For example, I used Bootstrap’s grid system to create a responsive e-commerce website. The grid system allowed me to define column structures that adapt to various screen sizes, ensuring that product listings, navigation bars, and other components displayed appropriately on mobile, tablet, and desktop devices.

Q20: What is responsive web design, and why is it important?
A20: Responsive web design is an approach that ensures a website’s layout and content adjust seamlessly to different screen sizes and devices. It’s important because it provides a consistent and user-friendly experience across all devices, from mobile phones to desktops, which is crucial in today’s multi-device world. Responsive design also improves accessibility and SEO, enhancing overall site performance.