PW SKILLS

Accenture- Frontend Developer

Interview Process

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

Interview Questions

1. What is the arrow function in javascript?
2. Hoisting, promise, Pure Function in JS
3. How to check if an object is empty or not?
4. React Lifecycle
5. Redux flow
6. Difference between Promise and Observables?
7. Call, Apply and bind
8. Uses of lifecycle in React hook.
9. Data-binding, Lazy Loading
10. spread operators
11. Box Model, Grid, FLexbox
12. Spread and rest operators
13. New features in HTML5
14. HTML Web Storage
15. Render JSON from parent to child component.
16. How do you ensure accessibility in your web projects?
17. Describe your approach to optimizing a website for search engines (SEO).
18. Have you worked with any task runners or build tools like Webpack or Gulp? If so, describe your experience.
19. How do you keep up with the latest trends and advancements in front-end development?
20. Describe a challenging technical problem you encountered in a project and how you resolved it.

Solutions

Q1: What is the arrow function in JavaScript?
A1: Arrow functions are a concise syntax for writing function expressions in JavaScript. They have a shorter syntax and lexically bind the `this` value. For example: `const add = (a, b) => a + b;`. This syntax is particularly useful for simple functions and can make code more readable.

Q2: Explain hoisting, promises, and pure functions in JS.
A2: Hoisting is JavaScript's default behavior of moving declarations to the top of their scope. Promises represent the eventual completion or failure of an asynchronous operation, providing better handling of callbacks. Pure functions always return the same output for the same input and have no side effects, enhancing code predictability and testability.

Q3: How do you check if an object is empty or not?
A3: Two common methods to check if an object is empty are:

1. `Object.keys(obj).length === 0`
2. `JSON.stringify(obj) === '{}'`

The first checks for enumerable properties, while the second converts the object to a JSON string for comparison.

Q4: Describe the React lifecycle.
A4: React components have several lifecycle phases:

1. Mounting: constructor, render, componentDidMount
2. Updating: render, componentDidUpdate
3. Unmounting: componentWillUnmount Understanding these phases helps manage side effects and optimize performance in React applications.

Q5: Explain the Redux flow.
A5: Redux follows a unidirectional data flow:

1. An action is dispatched
2. The reducer processes the action
3. The store is updated
4. The view re-renders to reflect the new state This predictable state management enhances understanding of data changes in applications.

Q6: What's the difference between Promises and Observables?
A6: Promises handle single asynchronous values, while Observables can handle multiple values over time. Promises are eager (start immediately when created), while Observables are lazy (start when subscribed to). Observables offer additional features like cancellation and retry operations.

Q7: Explain call, apply, and bind in JavaScript.
A7: These methods allow explicit setting of the `this` value in a function:

- `call()`: Invokes a function with a given `this` and comma-separated arguments
- `apply()`: Similar to `call()`, but takes arguments as an array
- `bind()`: Returns a new function with a fixed `this` value, useful for callbacks

Q8: How are life cycles used in React hooks?
A8: In React hooks, `useEffect()` is primarily used to handle lifecycle events. It combines the functionality of componentDidMount, componentDidUpdate, and componentWillUnmount. The dependency array in `useEffect()` controls when the effect runs, providing flexibility for managing side effects and subscriptions.

Q9: What are data-binding and lazy loading?
A9: Data-binding synchronizes data between the model and view. In React, it's typically one-way, from parent to child via props. Lazy loading is a technique to defer loading of non-critical resources, improving initial load time. React's `React.lazy()` and `Suspense` enable component-level code splitting for lazy loading.

Q10: Explain spread operators in JavaScript.
A10: The spread operator (...) expands an iterable into individual elements. It's useful for:

- Creating shallow copies of arrays or objects
- Combining arrays
- Spreading an array as function arguments Example: `const newArray = [...oldArray, newElement];`

Q11: Describe the CSS box model, grid, and flexbox.
A11: The CSS box model describes element rendering with content, padding, border, and margin. CSS Grid is a two-dimensional layout system for complex layouts. Flexbox is a one-dimensional layout model for distributing space among items in a container, even with unknown sizes.

Q12: What's the difference between spread and rest operators?
A12: While using the same syntax (...):

- Spread expands an iterable (e.g., copying arrays/objects)
- Rest collects multiple elements into an array (often in function parameters)

Q13: What are some new features in HTML5?
A13: HTML5 introduced:

- Semantic elements (header, nav, footer)
- Native audio and video support
- Canvas element for graphics
- New form input types These features reduce plugin dependency and improve web experiences.

Q14: Explain HTML Web Storage.
A14: HTML5 Web Storage includes:

- localStorage: Persistent storage surviving browser restarts
- sessionStorage: Cleared when the session ends Both offer key-value storage on the client-side, providing an alternative to cookies.

Q15: How do you render JSON from a parent to a child component?
A15: In React, JSON data is typically passed from parent to child as a prop: Parent: `<ChildComponent data={jsonData} />` Child: Access via `props.data` This approach promotes component reusability and separation of concerns.

Q16: How do you ensure accessibility in web projects?
A16: Key accessibility practices include:

- Using semantic HTML elements
- Providing alt text for images
- Ensuring keyboard navigation
- Using ARIA attributes when necessary
- Testing with screen readers and accessibility tools like Lighthouse

Q17: Describe an approach to optimizing a website for search engines (SEO).
A17: SEO optimization strategies include:

- Using appropriate meta tags (title, description)
- Creating high-quality, relevant content
- Proper use of header tags (H1, H2, etc.)
- Optimizing images and improving page load speed
- Ensuring mobile-friendliness
- Implementing structured data markup
- Regular monitoring and adjusting based on analytics

Q18: Describe experience with task runners or build tools.
A18: Common build tools and their uses:

- Webpack and Vite: Module bundling, code splitting, asset management
- Gulp: Task automation, file processing These tools improve development workflows and optimize build performance.

Q19: How to keep up with the latest trends in front-end development?
A19: Strategies to stay updated include:

- Following tech blogs (e.g., CSS-Tricks, Smashing Magazine)
- Attending web development conferences and webinars
- Participating in online communities (Stack Overflow, GitHub)
- Experimenting with new technologies in side projects

Q20: How to approach resolving challenging technical problems?
A20: A systematic approach to problem-solving includes:

1. Clearly defining the problem
2. Breaking it down into smaller tasks
3. Researching potential solutions
4. Implementing and testing iteratively
5. Profiling and optimizing as necessary
6. Documenting the process and solution for future reference