Frontend Engineer Interview

Starting with HTML:

1. Semantic Tags: Understand and use `<article>`, `<section>`, `<header>`, `<footer>`, `<nav>`.
2. Forms: Implement validation, handle `<input>`, `<textarea>`, `<select>`, `<button>`.
3. Tables: Create accessible tables with `<table>`, `<thead>`, `<tbody>`, `<tfoot>`.
4. Metadata: Use `<meta>` tags for charset, viewport, and SEO.
5. Links and Anchors: Understand `<a>` tags, `href`, `target`, and `download` attributes.
6. Media Elements: Use `<img>`, `<video>`, `<audio>` correctly with attributes like `src`, `alt`, `controls`.
7. Lists: Create ordered `<ol>` and unordered `<ul>` lists, including nested lists.
8. Headings: Use proper heading hierarchy `<h1>` to `<h6>`.
9. Embedding Content: Use `<iframe>`, `<embed>`, and `<object>` for embedding external content.
10. HTML5 APIs: Familiarity with Geolocation, Web Storage, and Fetch API.

Now, CSS:

11. Box Model: Understand margin, padding, border, and how they affect layout.
12. Flexbox: Master alignment, wrapping, and ordering with Flexbox properties.
13. Grid Layout: Create complex layouts using CSS Grid.
14. Responsive Design: Use media queries, viewport meta tag, and responsive images.
15. CSS Preprocessors: Knowledge of Sass, Less, or Stylus syntax and features.
16. CSS-in-JS: Understand frameworks like styled-components or emotion.
17. Animation and Transitions: Implement smooth transitions and keyframe animations.
18. Styling Forms: Customize form elements and improve their appearance.
19. CSS Reset and Normalize: Know when and why to use them.
20. CSS Grid vs Flexbox: Understand the differences and choose the right tool for the job.

JavaScript:

21. ES6+ Features: Use arrow functions, destructuring, spread/rest operators, and template literals.
22. DOM Manipulation: Select elements, modify the DOM, and handle events.

23. Asynchronous JavaScript: Understand Promises, async/await, and fetch API.
24. Event Loop: Explain how the event loop works in JavaScript.
25. Closures: Understand and use closures effectively.
26. Prototypal Inheritance: Explain how prototypal inheritance works in JavaScript.
27. Modules: Use ES6 modules with `import` and `export`.
28. Error Handling: Use try/catch blocks and understand unhandled promise rejections.
29. JavaScript Performance: Optimize code for better performance.
30. Browser Console: Use browser developer tools for debugging.

Frameworks:

31. React.js: Understand components, JSX, state, props, and hooks.
32. Vue.js: Understand Vue instance, directives, components, and reactivity.
33. Angular: Understand components, services, dependency injection, and routing.
34. State Management: Use Redux, Vuex, or Context API for state management.
35. Routing: Implement client-side routing with React Router, Vue Router, etc.
36. Component-Based Architecture: Understand and implement reusable components.
37. Lifecycle Methods: Know React lifecycle methods or Vue hooks.
38. UI Libraries: Use libraries like Bootstrap, Tailwind, or Material-UI.
39. Testing Frameworks: Write tests with Jest, Jasmine, or Cypress.
40. Build Tools: Use Webpack, Babel, or Parcel for building projects.

Tools and Version Control:

41. Git: Use git for version control, including branching, merging, and rebasing.
42. npm/yarn: Manage project dependencies and scripts.
43. Package.json: Understand scripts, dependencies, and devDependencies.
44. Task Runners: Use Gulp or Grunt for automating tasks.
45. Linting: Use ESLint or Prettier for code quality.
46. Browsersync: Use for live reloading during development.
47. Figma/Adobe XD: Understand design handoff and collaborate with designers.

48. API Integration: Fetch data from RESTful or GraphQL APIs.
49. Environment Variables: Manage environment-specific configurations.
50. Continuous Integration: Set up CI/CD pipelines with GitHub Actions or Jenkins.

Performance Optimization:

51. Code Splitting: Implement code splitting with Webpack or dynamic imports.
52. Lazy Loading: Lazy load images, components, and scripts.
53. Minification: Minify CSS, JavaScript, and HTML files.
54. Caching Strategies: Use HTTP caching headers and service workers.
55. Image Optimization: Compress and optimize images for web use.
56. Critical CSS: Inline critical CSS for faster page loads.
57. Web Performance Metrics: Understand Lighthouse, GTmetrix, and PageSpeed Insights.
58. Font Loading: Optimize font loading with WebFont Loader or self-hosting.
59. Avoiding Render-Blocking Resources: Ensure scripts and styles don’t block rendering.
60. Performance Budgets: Set and adhere to performance budgets.

Accessibility:

61. ARIA Roles: Use ARIA roles, states, and properties for better accessibility.
62. Semantic HTML: Choose semantic elements to improve accessibility.
63. Alt Text for Images: Provide meaningful alt text for images.
64. Keyboard Navigation: Ensure the site is navigable with keyboard only.
65. Color Contrast: Use tools to check and improve color contrast.
66. Screen Reader Testing: Test with screen readers like NVDA or VoiceOver.
67. Focus Management: Ensure proper focus management on interactive elements.
68. Accessibility Guidelines: Follow WCAG 2.1 guidelines.
69. Form Accessibility: Use labels, placeholders, and validation correctly.
70. EPub and AODA Compliance: Understand basic compliance standards.

Best Practices:

71. Code Organization: Maintain clean and modular code structures.

72. Documentation: Write clear documentation for components and APIs.
73. Cross-Browser Testing: Test on multiple browsers and devices.
74. Progressive Enhancement: Build sites that work for all users, regardless of browser support.
75. Security: Prevent XSS attacks, use Content Security Policy, and secure APIs.
76. SEO Best Practices: Optimize for search engines with meta tags, headings, and alt text.
77. Versioning: Use semantic versioning for libraries and dependencies.
78. Collaboration Tools: Use GitHub, GitLab, or Bitbucket for team collaboration.
79. Code Reviews: Participate in code reviews and provide constructive feedback.
80. Learning Resources: Stay updated with MDN, blogs, and online courses.

Advanced Topics:

81. WebSockets: Implement real-time communication with WebSockets.
82. PWA (Progressive Web Apps): Understand service workers, offline support, and push notifications.
83. Canvas and SVG: Create graphics with Canvas and SVG elements.
84. CSS Grid and Flexbox Layouts: Implement complex layouts with CSS Grid and Flexbox.
85. Custom Elements: Create custom HTML elements with Web Components.
86. Shadow DOM: Understand and use Shadow DOM for encapsulation.
87. CSS Variables: Use custom properties for theming and dynamic styles.
88. JavaScript Design Patterns: Implement design patterns like Singleton, Observer, and Factory.
89. Internationalization (i18n): Implement language support and localization.
90. Performance Profiling: Use tools like Chrome DevTools for profiling JavaScript and DOM performance.

Cross-Disciplinary Skills:

91. User Experience (UX): Understand UX principles and collaborate with UX designers.
92. User Interface (UI): Create visually appealing and user-friendly interfaces.
93. Project Management: Use Agile methodologies, Scrum, or Kanban for project management.
94. Communication Skills: Effectively communicate with team members and stakeholders.
95. Problem-Solving: Approach problems methodically and find optimal solutions.
96. Adaptability: Quickly learn and adapt to new technologies and tools.

97. Team Collaboration: Work well in a team, share knowledge, and mentor others.
98. Time Management: Prioritize tasks and manage time effectively.
99. Creativity: Bring creative solutions to design and coding challenges.
100. Passion for Learning: Stay curious and continuously improve your skills.