Front End Technologies - Subjective Interview Questions

Download PDF

Edunet Foundation

Interview Questions – Front End Technologies

Question 1: What is the difference between HTML elements, tags, and attributes?

Answer: HTML elements are the individual components that make up a web page, such as headings, paragraphs, and images. Tags are used to mark the beginning and end of HTML elements. Attributes provide additional information or modify the behavior of HTML elements.

Question 2: What are the main differences between HTML4 and HTML5?

Answer: HTML5 introduced significant improvements over HTML4, including new semantic elements like <header>, <footer>, and <article>, native support for multimedia elements such as <video> and <audio>, as well as the <canvas> element for dynamic graphics. Additionally, HTML5 introduced APIs for offline storage, geolocation, drag-and-drop, and improved form handling.

Question 3: What is the basic structure of an HTML document?

Answer: An HTML document typically starts with a <!DOCTYPE> declaration, followed by the <html> element enclosing the entire document. Inside <html>, there are <head> and <body> sections. The <head> contains metadata like title, links to stylesheets, and scripts, while the <body> contains the visible content of the page.

Question 4: Describe the difference between block-level and inline elements in HTML.

Answer: Block-level elements start on a new line and occupy the full width available, such as and <p>. Inline elements do not start on a new line and only occupy as much width as necessary, like and <a>. Block-level elements can contain other block-level and inline elements, while inline elements cannot contain block-level elements.

Question 5: What is the purpose of the colspan and rowspan attributes in the <td> and <th> tags?

Answer: The colspan attribute specifies the number of columns a cell should span, and the rowspan attribute specifies the number of rows a cell should span.

Question 6: How do you create a hyperlink in HTML?

Answer: Hyperlinks are created using the <a> (anchor) tag, with the "href" attribute specifying the URL of the destination. Optionally, you can include link text between the opening and closing <a> tags to provide descriptive text for the link.

Question 7: What is the purpose of the "alt" attribute in an HTML image tag?

Answer: The "alt" attribute provides alternative text for an image, which is displayed if the image fails to load or for accessibility purposes. Screen readers use this text to describe the image to visually impaired users, improving accessibility and understanding of the content.

Question 8: What is the significance of the "DOCTYPE" declaration in HTML?

Answer: The <!DOCTYPE> declaration specifies the HTML version and helps the browser render the web page correctly by triggering standards mode. It ensures that the browser interprets the HTML code correctly and renders the page according to the specified version of HTML.

Question 9: How can you embed multimedia content, such as videos and audio, in an HTML document?

Answer: Multimedia content can be embedded using the <video> and <audio> tags in HTML5. These tags allow you to specify the source of the multimedia content using the "src" attribute and provide fallback content for browsers that do not support the specified format or codec.

Question 10: Explain the difference between the and elements in HTML.

Question 11: What are semantic HTML elements, and why are they important?

Answer: Semantic HTML elements convey meaning about the content they contain, making the structure of the web page clearer to both developers and assistive technologies. By using semantic elements like <header>, <footer>, <nav>, and <article>, developers improve accessibility and search engine optimization, as search engines and screen readers can better understand the content's context and hierarchy.

Question 12: How can you create a form in HTML, and what are the different form input types?

Answer: Forms are created using the <form> tag, which contains input fields and buttons for user interaction. Various input types like text, password, checkbox, radio, select, and textarea can be used to collect user input. Each input type has specific attributes and behaviors for capturing different types of data.

Question 13: Describe the difference between the GET and POST methods in HTML forms.

Answer: The GET method sends form data as part of the URL, visible to users and limited in the amount of data it can send. It is suitable for retrieving data from the server but not recommended for sensitive or large amounts of data. The POST method sends form data in the HTTP request body, suitable for sensitive or large amounts of data, and does not have restrictions on data length.

Question 14: What is the purpose of the <meta> tag in HTML?

Answer: The <meta> tag provides metadata about the HTML document, such as character encoding, viewport settings, and keywords for search engines. It is used to specify information that is not displayed on the web page but is important for search engines, browsers, and other web services.

Question 15: How can you add comments in an HTML document, and why are they useful?

Answer: Comments in HTML are added using <!-- comment --> syntax and are not displayed in the web page. They are useful for documenting code, providing explanations, and temporarily disabling code without removing it. Comments help developers understand the purpose and structure of the code, making it easier to maintain and collaborate on projects.

Question 16: What is the "box-sizing" property in CSS, and how does it affect layout?

Answer: The "box-sizing" property in CSS allows you to define how the total width and height of an element are calculated, including padding and borders. The default value is "content-box," where padding and borders are added to the specified width and height. By setting it to "border-box," padding and borders are included within the specified width and height, making it easier to manage layout and avoid unexpected resizing.

Question 17: How do CSS preprocessors like Sass and Less enhance the development process?

Answer: CSS preprocessors like Sass and Less introduce features such as variables, mixins, nested rules, and functions, which streamline CSS authoring and make stylesheets more maintainable and reusable. They allow for modularization, easier theming, and faster development by enabling developers to write more concise and organized CSS code.

Question 18: What is the purpose of the "z-index" property in CSS?

Answer: The "z-index" property in CSS controls the stacking order of positioned elements along the z-axis (depth) of the document. Elements with a higher "z-index" value appear in front of elements with a lower value. It is commonly used to control the stacking order of overlapping elements, such as positioned or floated elements, and is particularly useful for creating layered layouts and managing the visibility of elements.

Question 19: Explain the difference between "em" and "rem" units in CSS.

Answer: Both "em" and "rem" units are relative units used for sizing elements in CSS. However, "em" units are relative to the font size of the parent element, while "rem" units are relative to the font size of the root (html) element. This makes "rem" units particularly useful for creating consistent and scalable designs, as they are not affected by nested font sizes.

Question 20: What is the purpose of CSS preprocessors like Autoprefixer?

Answer: CSS preprocessors like Autoprefixer automatically add vendor prefixes to CSS properties, ensuring cross-browser compatibility and saving developers time and effort. They analyze CSS code and add the necessary prefixes for properties that require vendor-specific prefixes to work correctly in different browsers, helping to maintain consistent styling across various platforms and devices.

Question 21: How does the "display" property in CSS affect the layout of elements?

Question 22: Explain the concept of CSS specificity and how it affects style resolution.

Answer: CSS specificity determines which styles are applied to an element when multiple conflicting styles are defined. It is calculated based on the combination of selectors used to target an element. Specificity is represented numerically, and the style with the highest specificity value takes precedence. Understanding specificity helps developers predict which styles will be applied and resolve styling conflicts efficiently.

Question 23: What is the purpose of CSS preprocessors like PostCSS?

Answer: CSS preprocessors like PostCSS provide a framework for transforming CSS with JavaScript plugins. Unlike traditional preprocessors, PostCSS is highly flexible and extensible, allowing developers to use plugins to automate tasks, optimize code, add future CSS features, and more. PostCSS enables developers to create custom CSS processing pipelines tailored to their specific needs and project requirements.

Question 24: How do you create a responsive design in CSS?

Answer: Responsive design in CSS involves using media queries, flexible grids, and fluid layouts to adapt the layout of a website to different screen sizes and devices. Media queries allow developers to apply specific styles based on device characteristics such as screen width, height, orientation, and resolution. By designing with responsiveness in mind, websites can provide an optimal viewing experience across a range of devices, from desktops to smartphones.

Question 25: Explain the purpose of the "position" property in CSS and its different values.

Answer: The "position" property in CSS specifies the positioning method used for an element. Its values include "static," "relative," "absolute," "fixed," and "sticky." "Static" is the default positioning, where elements flow naturally in the document. "Relative" positions an element relative to its normal position. "Absolute" positions an element relative to its nearest positioned ancestor. "Fixed" positions an element relative to the viewport. "Sticky" is a hybrid of "relative" and "fixed," where an element is relative until it crosses a specified threshold, then it becomes fixed.

Question 26: How can you create CSS transitions and animations?

Answer: CSS transitions and animations allow developers to add dynamic effects to elements, such as fading, scaling, rotating, and moving. Transitions are applied to property changes over a specified duration, while animations define keyframes of property changes and their timing. Both transitions and animations can be triggered by user interactions, such as hover or click events, and provide smooth visual feedback to enhance the user experience.

Question 27: What are the benefits of using CSS frameworks like Bootstrap or Foundation?

Answer: CSS frameworks like Bootstrap and Foundation provide pre-designed components, grid systems, and styling patterns that accelerate the development of responsive and visually appealing websites. They offer a consistent design language, reduce the need for custom CSS coding, and ensure cross-browser compatibility. CSS frameworks also streamline collaboration among developers and designers by providing common design patterns and best practices.

Question 28: How do you vertically align text within a container in CSS?

Answer: Vertically aligning text within a container in CSS can be achieved using various techniques, such as setting the container's "display" property to "flex" and using the "align-items" property, applying the "line-height" property equal to the container's height, or using CSS grid properties like "align-self" or "align-items" on the text element within the container.

Question 29: Explain the concept of CSS specificity and how it affects style resolution.

Answer: CSS specificity determines which styles are applied to an element when multiple conflicting styles are defined. It is calculated based on the combination of selectors used to target an element. Specificity is represented numerically, and the style with the highest specificity value takes precedence. Understanding specificity helps developers predict which styles will be applied and resolve styling conflicts efficiently.

Question 30: What are CSS variables, and how do they differ from traditional CSS properties?

Answer: CSS variables, also known as custom properties, allow developers to define reusable values that can be used throughout a stylesheet. They are declared using the "--" prefix and can be assigned values dynamically using the "var()" function. Unlike traditional CSS properties, which are static and need to be manually updated, CSS variables enable more flexible and maintainable styling by centralizing values and promoting consistency across stylesheets.

Question 31: What is the purpose of the "box-shadow" property in CSS, and how can you use it to create shadow effects?

Question 32: Describe the difference between "inline" and "inline-block" display values in CSS.

Answer: Both "inline" and "inline-block" display values allow elements to flow horizontally within their containing block, but they have different behaviors regarding layout and interaction. "Inline" elements do not respect height, width, padding, or margin properties, while "inline-block" elements do. This makes "inline-block" elements behave more like block-level elements while still allowing them to flow inline with other elements.

Question 33: What is the "clearfix" technique in CSS, and when is it used?

Answer: The "clearfix" technique in CSS is used to contain floated elements within a container. When elements are floated, their container may collapse, causing layout issues. The "clearfix" technique typically involves adding a pseudo-element with clear:both; property to the container, forcing it to expand to contain its floated children properly.

Question 34: Explain the difference between "max-width" and "width" properties in CSS.

Answer: The "width" property in CSS sets the width of an element explicitly, while the "max-width" property sets the maximum width an element can have. If the content within an element exceeds the specified width, the "max-width" property ensures that the element does not exceed the specified maximum width, allowing it to remain responsive and adapt to different screen sizes.

Question 35: How can you center an element horizontally and vertically in CSS?

Answer: Centering an element horizontally can be achieved by setting its left and right margins to "auto" and giving it a defined width. Vertically centering an element within its container is more complex and often requires techniques such as using flexbox or CSS grid. One common method is to use the "flex" property with a combination of "align-items" and "justify-content" set to "center" on the container, along with specifying a height for the container.

Question 36: What is Bootstrap, and what is its primary purpose?

Answer: Bootstrap is a popular front-end framework developed by Twitter. Its primary purpose is to provide developers with a set of tools and components for building responsive and mobile-first websites and web applications efficiently.

Question 37: Describe the grid system in Bootstrap and its significance in web development.

Answer: The grid system in Bootstrap is a flexible and responsive layout system based on a 12-column grid. It allows developers to create responsive designs by dividing the page into rows and columns, where content can be placed. The grid system is significant as it enables developers to create consistent layouts across different screen sizes and devices, ensuring a seamless user experience.

Question 38: How does Bootstrap ensure responsiveness in web design?

Answer: Bootstrap ensures responsiveness in web design through its grid system, which adapts to various screen sizes using media queries. Additionally, Bootstrap provides responsive utility classes and components that allow developers to hide, show, or adjust elements based on screen size, ensuring that websites and web applications look and function well on desktops, tablets, and mobile devices.

Question 39: What are the key components of Bootstrap, and how are they used in web development?

Answer: Some key components of Bootstrap include the grid system, navigation bars, buttons, forms, modals, carousels, and more. These components provide pre-styled and customizable building blocks for creating responsive and visually appealing interfaces in web development projects.

Question 40: How does Bootstrap handle browser compatibility issues?

Answer: Bootstrap is designed to be compatible with modern web browsers, including Chrome, Firefox, Safari, Edge, and others. It uses CSS prefixes and JavaScript polyfills to ensure compatibility with older browsers. Additionally, Bootstrap's responsive design approach helps mitigate browser-specific issues by adapting layouts and styles based on the capabilities of the browser.

Question 41: Explain the difference between Bootstrap's containers: .container, .container-fluid, and .container-xl.

Answer:

.container: This class creates a fixed-width container that is responsive and adjusts its width based on the screen size. It has predefined widths for different breakpoints.

.container-xl: Introduced in Bootstrap 5, this class creates a fixed-width container for extra-large screens, providing more control over the layout on larger devices.

Question 42: How can you customize Bootstrap components and styles to match your project's design requirements?

Answer: Bootstrap allows for extensive customization through Sass variables and mixins. Developers can modify variables such as colors, fonts, spacing, and breakpoints to match the project's design requirements. Additionally, Bootstrap provides utilities for overriding default styles and creating custom components to ensure consistency with the project's design language.

Question 43: What role do Bootstrap utilities play in web development, and provide some examples of commonly used utilities?

Answer: Bootstrap utilities are CSS classes that provide quick and easy ways to apply common styles and functionality to elements. Some commonly used utilities include spacing utilities (e.g., mt-3 for margin-top), text utilities (e.g., text-center for centering text), and display utilities (e.g., d-flex for creating flexible layouts with flexbox).

Question 44: How does Bootstrap support accessibility in web design?

Answer: Bootstrap follows best practices for accessibility, including semantic HTML markup, proper use of ARIA attributes, and keyboard navigation support. It also provides built-in accessibility features for components such as modals, tooltips, and dropdowns, ensuring that websites and web applications built with Bootstrap are accessible to all users, including those with disabilities.

Question 45: Describe the process of integrating Bootstrap into a web project and optimizing its performance.

Answer: Integrating Bootstrap into a web project involves linking the Bootstrap CSS and JavaScript files in the HTML document, either by downloading the files or using a CDN. Developers can then use Bootstrap's classes and components to build the project's UI. To optimize performance, developers should only include the necessary Bootstrap components and styles, minimize HTTP requests, and leverage browser caching techniques. Additionally, using Bootstrap's built-in customization options and optimizing images and assets can help improve page load times.

Question 46: What are the differences between let, const, and var in JavaScript?

Answer: var declarations are function-scoped, and they can be re-declared and updated.

let declarations are block-scoped, and they cannot be re-declared but can be updated.

const declarations are also block-scoped, but they cannot be re-declared or updated (except for objects and arrays, where their properties or elements can be modified).

Question 47: Explain the difference between "null" and "undefined" in JavaScript.

Answer: In JavaScript, "null" represents the intentional absence of any value and is explicitly assigned by developers. "Undefined," on the other hand, indicates that a variable has been declared but has not been assigned a value. It is also the default value for uninitialized variables.

Question 48: What are the different data types in JavaScript?

Answer: JavaScript has several data types, including primitive types such as numbers, strings, booleans, null, undefined, and symbol (added in ES6), as well as non-primitive type, which is object.

Question 49: Explain the concept of closures in JavaScript and provide an example.

Answer: Closures are functions that retain access to variables from their containing (enclosing) lexical scope even after the parent function has finished executing. They allow functions to "remember" and access variables from their surrounding scope.

Question 50: What is the difference between "==" and "===" in JavaScript?

Answer: The "==" operator in JavaScript performs type coercion, meaning it attempts to convert the operands to the same type before comparison. On the other hand, the "===" operator (strict equality) compares both the value and the type of the operands without coercion. For example, 5 == '5' would return true, while 5 === '5' would return false.

Question 51: Explain the concept of prototypal inheritance in JavaScript.

Answer: Prototypal inheritance is a mechanism in JavaScript where objects can inherit properties and methods from other objects through their prototype chain. Each object in JavaScript has a prototype object, and when a property or method is accessed on an object, JavaScript will first look for it on the object itself. If not found, it will traverse up the prototype chain until it finds the property or method or reaches the end of the chain (Object.prototype).

compile phase. This means that regardless of where variables and functions are declared in the code, they are accessible from any point within their containing scope.

Question 53: What are the different ways to create objects in JavaScript?

Answer: There are several ways to create objects in JavaScript, including object literals, constructor functions, the "new" keyword, Object.create(), and class syntax (introduced in ES6).

Question 54: Explain the event bubbling and event capturing phases in JavaScript.

Answer: Event bubbling and event capturing are two phases of event propagation in the DOM. During event capturing, the event travels from the top of the DOM hierarchy down to the target element, while during event bubbling, the event travels from the target element back up to the top of the hierarchy. Event handlers can be registered to listen for events during either phase.

Question 55: What is the "this" keyword in JavaScript, and how does it work?

Answer: The "this" keyword in JavaScript refers to the context in which a function is executed and provides access to the current object. Its value is determined by how a function is called. In a function, "this" typically refers to the object that invokes the function or the context in which the function is called.

Question 56: Explain the concept of event delegation in JavaScript.

Answer: Event delegation is a technique in JavaScript where a single event listener is attached to a parent element to listen for events that occur on its child elements. This allows for efficient event handling, especially for dynamically created elements, as it reduces the number of event listeners needed and improves performance.

Question 57: What are promises in JavaScript, and how do they work?

Answer: Promises in JavaScript are objects representing the eventual completion or failure of an asynchronous operation and its resulting value. They provide a cleaner alternative to traditional callback-based asynchronous code and allow developers to handle asynchronous operations more easily using methods like ".then()" and ".catch()".

Question 58: How does JavaScript handle asynchronous programming, and what are some techniques for managing asynchronous code?

Answer: JavaScript uses asynchronous programming to handle tasks that take time to complete, such as network requests or file operations, without blocking the execution of other code. Some techniques for managing asynchronous code include callbacks, promises, async/await, and event listeners.

Question 59: What are arrow functions in JavaScript, and how do they differ from traditional function expressions?

Answer: Arrow functions are a more concise syntax for writing function expressions in JavaScript, introduced in ES6. They have a shorter syntax, implicit return for single expressions, and lexical scoping of the "this" keyword. Unlike traditional function expressions, arrow functions do not have their own "this" or "arguments" bindings.

Question 60: Explain the concept of scope in JavaScript.

Answer: Scope refers to the visibility and accessibility of variables and functions in different parts of a JavaScript program. JavaScript has two main types of scope: global scope, where variables and functions are accessible throughout the entire program, and local scope, where variables and functions are only accessible within their containing block or function.

Question 61: How would you optimize the performance of a web page, particularly focusing on front-end performance?

Answer: Front-end performance optimization involves techniques like file minification, browser caching, lazy loading, image optimization, reducing HTTP requests, utilizing CDNs, and prioritizing critical content loading.

Question 62: Can you describe the difference between progressive enhancement and graceful degradation, and provide examples of when you might use each approach?

Answer: Progressive enhancement starts with a basic version and adds enhancements for modern browsers, ensuring all users get core functionality. Graceful degradation starts with a fully-featured version and simplifies for older browsers, ensuring basic functionality remains intact. For example, progressive enhancement might involve starting with plain HTML and enhancing with CSS and JavaScript, while graceful degradation might involve building advanced CSS3 animations and simplifying for browsers that don't support CSS3.

Answer: Address compatibility by utilizing progressive enhancement, feature detection, polyfills, vendor prefixes, thorough testing across browsers, and providing fallbacks for unsupported features, ensuring a consistent user experience across different platforms.

Question 64: What are some common techniques for achieving responsive design, and how do they differ from one another?

Answer: Common techniques include using media queries, fluid grids, and flexible images. Media queries allow CSS to adapt to different viewport sizes, fluid grids use percentages instead of fixed units for layout, and flexible images adjust their size based on the size of the viewport.

Question 65: Can you discuss the benefits and limitations of using front-end frameworks like React or Angular in web development?

Answer: Front-end frameworks like React and Angular offer benefits such as faster development, component reusability, and better state management. However, they may also introduce a learning curve, increased complexity, and potential performance overhead compared to vanilla JavaScript development.

Question 66: What is React JS?

Answer: React JS is a JavaScript library for building user interfaces. It's component-based, meaning you break down your UI into reusable components.

Question 67: What is JSX?

Answer: JSX is a syntax extension for JavaScript that allows you to write HTML-like structures within JavaScript code. It makes writing UI components more intuitive and readable.

Question 68: Explain Virtual DOM and how it works.

Answer: Virtual DOM is a lightweight in-memory representation of the actual DOM. When state changes, React updates the virtual DOM, calculates the minimum number of changes required to update the actual DOM, and efficiently updates only the necessary parts.

Question 69: What are the key differences between function components and class components?

Answer: Function Components: Simpler, often used for stateless components. Can use hooks like useState and useEffect for state and side effects.

Class Components: More complex, requiring class syntax. Use state and lifecycle methods for managing state and side effects.

Question 70: What are hooks?

Answer: Hooks are functions that let you "hook into" React state and lifecycle features from function components. Common hooks include useState, useEffect, useContext, useReducer, and more.

Question 71: Explain the concept of props.

Answer: Props are used to pass data from parent components to child components. They are immutable, meaning they cannot be modified by child components.

Question 72: What is state in React?

Answer: State is a mechanism for managing data that can change over time within a component. It allows you to create dynamic and interactive UIs.

Question 73: What are the different lifecycle methods in class components?

Answer: componentDidMount, componentDidUpdate, componentWillUnmount, render

Question 74: How can you fetch data in a React component?

Answer: Use the useEffect hook to fetch data from an API and update the component's state.

Question 75: What is the difference between useEffect and useLayoutEffect?

Answer: useEffect runs after DOM mutations, while useLayoutEffect runs synchronously after DOM mutations. Use useLayoutEffect for case where the component's layout needs to be adjusted immediately after rendering.

render specific components based on the URL.

Question 77: Explain the BrowserRouter and Router components.

Answer: BrowserRouter is used to wrap your application and handle routing.

Router is used to define routes with Route and Switch components.

Question 78: What are some techniques for optimizing React app performance?

Answer: Memoization: Use React.memo to memoize components and avoid unnecessary re-renders.

Code Splitting: Break down your application into smaller bundles to reduce initial load time.

Lazy Loading: Load components only when they are needed.

Virtualization: Render only the visible parts of a large list or table.

Profiling: Use browser developer tools to identify performance bottlenecks.

Question 79: What is Context API?

Answer: The Context API allows you to pass data through the component tree without having to pass props down manually at every level.

Question 80: What is Higher-Order Components (HOCs)?

Answer: HOCs are a pattern for reusing component logic by wrapping components with additional functionality.

Question 81: What is Render Props?

Answer: Render Props is a technique for sharing child rendering logic between components.

Question 82: What is React Suspense?

Answer: React Suspense allows you to declaratively suspend rendering while data is being fetched.

Question 83: What are the key differences between React Native and React JS?

Answer: React Native is used for building mobile applications, while React JS is used for building web applications. However, they share many core concepts and use similar syntax.

Question 84: What is Server-Side Rendering (SSR) in React?

Answer: SSR renders React components on the server, improving initial page load time and SEO.

Question 85: What are the main features of React?

Answer: The main features include JSX, Virtual DOM, components, one-way data binding, and performance optimization.

Question 86: What are components in React?

Answer: Components are the building blocks of a React application. They can be functional or class-based and encapsulate the UI and behavior.

Question 87: What is the difference between state and props?

Answer: State is a local data storage that is mutable and managed within the component. Props are read-only data passed from parent to child components.

Question 88: What are hooks in React?

Answer: Hooks are functions that let you use state and other React features in functional components. Examples include useState, useEffect, and useContext.

Question 89: What is the purpose of useState hook?

Answer: The useEffect hook lets you perform side effects in functional components, such as fetching data or directly manipulating the DOM.

Question 91: What happens if you omit the dependency array in useEffect?

Answer: If you omit the dependency array, the useEffect callback will run after every render of the component. This can lead to performance issues if the effect involves heavy computations or repeated API calls.

Question 92: What is the difference between controlled and uncontrolled components?

Answer: Controlled components have their state managed by React, while uncontrolled components manage their own state using the DOM.

Question 93: What is PropTypes?

Answer: PropTypes is a library for type-checking props in React components. It helps catch bugs by ensuring components receive props of the correct type.

Question 94: What is the difference between React.createRef and useRef?

Answer: React.createRef is used in class components to create a reference to a DOM element, while useRef is a hook used in functional components for the same purpose.

Question 95: What is the purpose of useMemo hook?

Answer: The useMemo hook memoizes a value, recomputing it only when its dependencies change, to optimize performance.

Question 96: What is the purpose of useCallback hook?

Answer: The useCallback hook returns a memoized callback function, which helps prevent unnecessary re-renders of child components.

Question 97: How does React handle events differently from HTML?

Answer: In React, events are handled using camelCase syntax (e.g., onClick instead of onclick). Additionally, React uses synthetic events, which are wrappers around the browser's native events, providing consistent behavior across different browsers.

Question 98: What are React Fragments and why are they used?

Answer: React Fragments allow you to group multiple elements without adding extra nodes to the DOM. They are useful for returning multiple elements from a component's render method without wrapping them in a div.

Question 99: How do you handle error boundaries in React?

Answer: Error boundaries are React components that catch JavaScript errors anywhere in their child component tree, log those errors, and display a fallback UI. They are implemented using componentDidCatch and getDerivedStateFromError lifecycle methods.

Question 100: How does React optimize rendering performance?

Answer: React optimizes rendering performance using techniques like virtual DOM for efficient updates, React.memo to prevent unnecessary re-renders, and hooks like useMemo and useCallback to memoize expensive calculations or function references. Additionally, React uses key props for efficient reconciliation.