Front-End Interview Question.

- **HTML Question Answers.**

1. **What is HTML.**

- HTML : Hyper Text Markup Language.

2. **What is Hyper Text and Markup Language.**

- **Hyper Text :** Hyper Text is text which contains links to other texts.
- **Markup Language :** it is a way to given instructions to a computer about how content should be organized and displayed.

3. **What is the History of HTML.**

- **Hyper Text :** Hyper Text was developed by Ted Nelson in 1960.
- 1980 Tim Berners Lee is a British computer scientist introduced the world wide web and developed in first HTML page.

4. **What is the Rule of write a code in HTML.**

- The child element must start after adding 4 spaces from where its parent element started. These spaces are called leading spaces.

5. **Why we use HTML.**

- We use HTML to structure and display content on the web, defining elements like text, images, and links.

6. **What is a Basic HTML Document Structure.**

- These tags are essential for creating a valid HTML document.
- <!DOCTYPE html>: Defines the document type and version of HTML (HTML5 in this case).
- <html>: Root element of the HTML document.
- <head>: Contains metadata, links to stylesheets, scripts, and other head elements.
- <body>: Contains the content of the webpage visible to the user.
- <title>: Specifies the title of the webpage (appears in the browser tab).
- <meta>: Defines metadata such as character encoding or viewport settings.
- <link>: Defines relationships to external resources, such as linking to a stylesheet.
- <style>: Contains internal CSS styles.
- <script>: Defines JavaScript code or links to external JavaScript files.

7. **What is Text Content tags.**

- These tags are used to display text content on the webpage.
- <h1> to <h6>: Define headings, <h1> being the highest level and <h6> the lowest.
- <p>: Defines a paragraph of text.
- <br>: Creates a line break (newline).
- <hr>: Represents a thematic break (horizontal rule), typically a line separating content.
- <span>: A generic inline container for styling and grouping text.

8. **What is Grouping Content Tags.**

- These tags are used to group sections of content together.
- <div>: A generic block-level container used for grouping and styling content.
- <section>: Defines a section of content within a document, typically with its own heading.
- <header>: Represents the header of a page or section, often containing navigation or introductory content.
- <footer>: Represents the footer of a page or section, often containing copyright, contact info, or related links.
- <nav>: Defines navigation links or menus.

9. **What is Linking and Navigation Tags.**

- These tags are used for creating hyperlinks and navigation.
- <a>: Defines a hyperlink, allowing users to click to navigate to another page or resource.
- <link>: Links external resources like stylesheets to the document.
- <nav>: Defines a navigation block containing links to different parts of the website or app.

10. **What is Table Tags.**

- These tags are used to create and manage tables.
- <table>: Represents a table.
- <tr>: Defines a table row.
- <td>: Defines a table cell (data).
- <th>: Defines a table header cell (usually bold and centered).
- <thead>: Groups the header content of a table.
- <tbody>: Groups the body content of a table.
- <tfoot>: Groups the footer content of a table.
- <caption>: Defines a caption for a table.
- <col>: Specifies column properties for the table.

- <colgroup>: Groups columns in a table for styling or formatting.

11. What is Forms and Input Tags.

- These tags are used to create interactive forms where users can input data.
- <form>: Defines a form element.
- <input>: Defines an input field where the user can enter data.
- <textarea>: Defines a multiline text input field.
- <button>: Represents a clickable button.
- <select>: Defines a dropdown menu.
- <option>: Defines an option within a <select> dropdown.
- <label>: Defines a label for an input element.
- <fieldset>: Groups related form elements together.
- <legend>: Defines a caption for a <fieldset>.
- <datalist>: Specifies a list of predefined options for an input element.
- <optgroup>: Groups options inside a <select> dropdown.
- <input type="radio">: Defines a radio button, used for selecting one option from a group.
- <input type="checkbox">: Defines a checkbox, used for binary choices (checked or unchecked).

12. What is Media Tags.

- These tags are used to embed images, audio, video, and other media.
- <img>: Embeds an image.
- <audio>: Embeds an audio file.
- <video>: Embeds a video file.
- <source>: Specifies multiple media resources for elements like <audio> and <video>.
- <track>: Defines text tracks for <video> and <audio>, such as subtitles or captions.
- <picture>: Specifies multiple image resources for responsive images (for different screen sizes and resolutions).
- <iframe>: Embeds another HTML page or external resource inside the current page.

13.What is Scripting Tags.

- These tags are used for scripting and embedding code.
- <script>: Embeds or links to JavaScript code.
- <noscript>: Provides content for users who have JavaScript disabled in their browser.
- <object>: Embeds an external resource like a plugin or interactive content.
- <embed>: Embeds external content, such as multimedia, applications, or files.
- <canvas>: Defines an area for drawing graphics on the fly via JavaScript.
- <svg>: Defines scalable vector graphics (SVG), often used for creating interactive graphics.
- <math>: Embeds mathematical equations, used with MathML for rendering mathematical content.

14.What is Miscellaneous Tags.

- Other useful HTML tags for various purposes.
- <b>: Bold text (use <strong> for stronger emphasis).
- <i>: Italic text (use <em> for emphasized text).
- <u>: Underlined text.
- <del>: Represents deleted text, shown with a strikethrough.
- <ins>: Represents inserted text, shown with an underline.
- <cite>: Defines a citation, such as the source of a quote.
- <dfn>: Represents the defining instance of a term.
- <kbd>: Represents keyboard input.
- <samp>: Represents sample output from a program or computer.
- <var>: Represents a variable in a mathematical expression or programming context.

15.What is HTML5 Tags.

- Some HTML5-specific tags that help with modern web design.
- <main>: Represents the main content of a document.
- <section>: Groups related content together in a section.
- <header>: Represents the introductory content or navigation of a page.
- <footer>: Represents the footer of a page or section.
- <figure>: Represents content like images or diagrams, often with captions.

16.What are Semantic Elements.

- A semantic element clearly describes its meaning to both the browser and the developer.
- **<header> :** Defines the header of a webpage or section.
- **<article> :** Represents a self-contained piece of content, like a blog post.
- **<nav> :** Indicates navigation links.
- **<section> :** Groups related content within a page.
- **<footer> :** Defines the footer, typically containing copyright or contact info.
- **<aside> :** Represents content tangentially related to the main content (like a sidebar).
- **<main> :** Represents the primary content of the document.

17.What is meta data tag.

- A meta data tag, or meta tag, is a piece of HTML code that provides information about a web page to search engines and other clients.

18.What is HTML SEO.

- HTML SEO is the use of Hypertext Markup Language (HTML) to improve a website's visibility in search engines.

- **CSS Interview Question.**

1. **What is CSS.**

- **CSS :** Cascading Style Sheets.
- Cascading Style Sheets is a style sheets language used to describe the look and presentation of a document written in HTML and XML.
- It defines how elements are displayed on a web page.
- CSS makes websites visually appealing and user-friendly.

2. **What is XML.**

- **XML :** Extensible Markup Language**.**
- XML is a markup language used to store, transport and exchange data. HTML is a markup language used to create web pages. XML focuses on storing data. HTML displays data.

3. **What are different ways to apply styles to a Web page?**

- There are three ways to style a web page.
- Inline styling/ style attribute.
- Internal style sheet/style tag.
- External style sheet (CSS).
- **Inline styling.**
- An inline style may be used to apply a unique style for a single element.
- **Internal style sheet/style tag.**
- An internal style sheet may be used if one single HTML page has a unique style.
- Style tag <style> in a head.
- **External style sheet (CSS).**
- An external style sheet is a separate CSS file that can be accessed by creating a link within the head section of the webpage.

4. **What is the difference between class and id selectors in CSS?**

- **Class selector**: Can be used on multiple elements in a page (e.g., .button {}).
- **ID selector**: Should be unique for each element on the page and used for a single element (e.g., #header {}).

5. **What is the CSS box model?**

- The box model defines the rectangular boxes that elements are rendered as. It includes:
- Content: The actual content (e.g., text, images).
- Padding: Space between the content and the border.
- Border: The boundary surrounding the padding (optional).
- Margin: Space outside the border separating the element from others.

6. **What is the difference between padding and margin?**

- **Padding** is the space between the content and the border, while.
- **margin** is the space between the border of an element and other surrounding elements.

7. **How can you center a block element horizontally?**

- Use margin: 0 auto; along with a defined width for the element.

8. **What is the difference between position: relative and position: absolute?**

- **relative**: Positions the element relative to its normal position without affecting the layout of other elements.
- **absolute**: Positions the element relative to the nearest positioned ancestor, or the initial containing block if no such ancestor exists.

9. **What are pseudo-classes and pseudo-elements?**

- **Pseudo-classes**: They target elements based on their state or position (e.g., :hover, :focus, :nth-child()).
- **Pseudo-elements**: They target specific parts of an element (e.g., ::before, ::after, ::first-letter).

10. **What is a CSS selector?**

- A CSS selector is used to select an HTML element and apply styles to it. Examples include element selectors (e.g., h1), class selectors (e.g., .button), and ID selectors (e.g., #header).

11. **What is the difference between display: none and visibility: hidden?**

- **display: none**: Completely removes the element from the document layout.
- **visibility: hidden**: Hides the element but still takes up space in the layout.

12. **What is Flexbox in CSS and when would you use it?**

- Flexbox is a layout model in CSS that allows items within a container to be aligned and distributed flexibly, making it easier to build complex layouts. Use it when you need flexible, responsive designs.

13. **How does flex-grow, flex-shrink, and flex-basis work in Flexbox?**

- **flex-grow**: Defines the ability of a flex item to grow relative to the other items.
- **flex-shrink**: Defines the ability of a flex item to shrink if necessary.
- **flex-basis**: Defines the initial size of a flex item before any remaining space is distributed.

14. **Explain the z-index property.**

- z-index determines the stacking order of elements along the z-axis (vertical). Higher values appear on top of lower values. It only works on elements that have a position other than static.

15. **What is the difference between inline and block elements?**

- **inline**: Elements don't start on a new line and only take up as much width as needed (e.g., <span>, <a>).
- **block**: Elements start on a new line and take up the full width available (e.g., <div>, <p>).
- **What are media queries in CSS?**
- Media queries are used to apply styles based on the characteristics of the device (such as screen width, height, resolution, etc.), enabling responsive design.

16. **What is opacity in CSS, and how does it work?**

- The opacity property controls the transparency of an element, where 0 is fully transparent and 1 is fully opaque. Values between 0 and 1 are used for semi-transparency.

17. **What is a CSS preprocessor?**

- A CSS preprocessor like **Sass**, **LESS**, or **Stylus** extends CSS by adding features such as variables, nested rules, mixins, and functions. It makes writing CSS more efficient and maintainable.

18. **What is float in CSS and how does it work?**

- The float property allows an element to be positioned to the left or right of its container, allowing text or inline elements to wrap around it. It is commonly used for creating layouts with columns.

19. **What is the difference between border-box and content-box in the box-sizing property?**

- **content-box**: Default value; the width and height apply to the content only, and padding and border are added outside.
- **border-box**: The width and height include padding and borders, which makes the box-sizing calculation more intuitive when styling elements.

20. What is Grid Layout in CSS, and how does it differ from Flexbox?

- CSS Grid is a two-dimensional layout system, allowing both rows and columns to be defined. Flexbox is one-dimensional, suitable for one-direction layouts (either row or column). Use Grid for complex layouts with both rows and columns.

21. What is the CSS will-change property used for?

- will-change is used to hint to the browser which properties are likely to change, so the browser can optimize rendering. It helps improve performance in animations and transitions.

22. What is the difference between transform and transition in CSS?

- **transform**: Used to move, scale, rotate, or skew an element without affecting its layout. Example: transform: translateX(10px);.
- **transition**: Used to smoothly change property values from one state to another over a specified duration. Example: transition: all 0.5s ease;.

23. Explain the concept of a "CSS animation" and how it differs from a transition.

- CSS animations allow for more complex animations than transitions. They can run continuously or a set number of times, and keyframes define multiple stages of the animation.

24. What is the clamp() function in CSS?

- The clamp() function allows you to set a property to a value that scales between a defined minimum and maximum. It's useful for responsive typography and layout adjustments. Example: font-size: clamp(1rem, 5vw, 3rem);

25. Explain the concept of contain in CSS.

- The contain property in CSS provides a way to limit the scope of styles or layout changes to an element. It can optimize rendering by telling the browser that an element's styling won't affect other elements outside it.

26. What is the backface-visibility property used for?

- backface-visibility controls whether the back face of an element is visible when it is rotated. For example, when an element is flipped using transform: rotateY(180deg), setting backface-visibility: hidden; hides the backside.

27. What is the object-fit property in CSS?

- The object-fit property controls how an image or video should be resized to fit its container. Values include fill, contain, cover, none, and scale-down.

28. What is the difference between @import and link in CSS?

- @import is used to import stylesheets at the top of a CSS file, while <link> is used to link external stylesheets within the HTML document. @import can cause slower page load times because it's synchronous.

29. What is the use of :root selector in CSS?

- :root is a pseudo-class that matches the highest-level element (typically <html> in HTML). It's commonly used to define global variables (e.g., --primary-color) that can be accessed throughout the CSS.

- **JavaScript Interview Question.**
- **Basic JavaScript Interview Questions**

1. **What is JavaScript?**

- A programming language used to create dynamic content on web pages, it runs on the client side (in the browser), and can also run on the server-side using environments like Node.js.

2. **What are variables in JavaScript?**

- Variables are used to store data values. In JavaScript, you can declare variables using var, let, or const.

3. **What's the difference between** var**,** let**, and** const**?**

- var: Function-scoped, can be re-declared and updated.
- let: Block-scoped, can be updated but not re-declared in the same block.
- const: Block-scoped, cannot be re-declared or updated after initialization.

4. **What is a JavaScript function?**

- A function is a block of code designed to perform a particular task. Functions are defined using the function keyword or ES6 arrow functions.

5. **What is the difference between** == **and** ===**?**

- == performs type coercion (compares values after converting them to the same type), while === checks both the value and type, ensuring they are exactly the same.

6. **What are JavaScript data types?**

- JavaScript has 6 primitive data
  types: undefined, null, boolean, number, string, symbol, and 1 non-primitive type: object.

7. **What is a closure in JavaScript?**

- A closure is a function that retains access to its lexical scope, even when the function is executed outside that scope.

8. **What is the difference between** null **and** undefined**?**

- undefined means a variable has been declared but not assigned a value, while null is an intentional assignment to indicate the absence of a value

- **Intermediate JavaScript Interview Questions**

9. **What is hoisting in JavaScript?**

- Hoisting is JavaScript's behavior of moving variable and function declarations to the top of their containing scope during compile time. Only the declarations are hoisted, not the assignments.

10. **What are callback functions in JavaScript?**

- A callback is a function passed as an argument to another function, and it's executed after the completion of the main function.

11. **What are promises in JavaScript?**

- A promise is an object representing the eventual completion (or failure) of an asynchronous operation. It has three states: pending, fulfilled, and rejected.

12. **What is asynchronous programming in JavaScript?**

- Asynchronous programming allows JavaScript to perform non-blocking operations (e.g., fetching data from an API) without blocking the execution of other code.

13. **What is the event loop in JavaScript?**

- The event loop is responsible for executing code, handling events, and executing queued sub-tasks in the JavaScript runtime.

14. **What are higher-order functions?**

- Higher-order functions are functions that take other functions as arguments, return functions as values, or both.

15. **What is the 'this' keyword in JavaScript?**

- this refers to the context in which a function is executed. Its value depends on how the function is called.

16. **What are JavaScript modules?**

- Modules are a way to separate code into reusable pieces. ES6 introduced import and export for handling modules.
- **Advanced JavaScript Interview Questions**

17. **What is the difference between** call()**,** apply()**, and** bind()**?**

- call(): Invokes a function with a specified this context and arguments passed individually.
- apply(): Similar to call(), but arguments are passed as an array.
- bind(): Returns a new function that, when called, has its this set to the provided value, with the given arguments.

18. **What is the prototype chain in JavaScript?**

- Every JavaScript object has a prototype. The prototype chain is used to look up properties and methods in an object's prototype when they are not found in the object itself.

19. **What is the** new **keyword in JavaScript?**

- The new keyword creates a new instance of a user-defined object type or built-in object type. It sets the this context to the new object and returns the new object by default.

20. **What are JavaScript design patterns?**

- Design patterns in JavaScript are reusable solutions to common problems. Examples include Singleton, Factory, Module, Observer, and MVC patterns.

21. **What is functional programming in JavaScript?**

- Functional programming treats computation as the evaluation of mathematical functions and avoids changing state and mutable data. JavaScript supports first-class functions, higher-order functions, closures, and immutability.

22. **Explain the concept of "currying" in JavaScript.**

- Currying is the technique of transforming a function that takes multiple arguments into a sequence of functions, each taking a single argument.

23. **What is the difference between** Object.create() **and a constructor function?**

- Object.create() is used to create a new object with a specific prototype, while constructor functions are used to create objects and initialize properties using the new keyword.

24. **What is the difference between** var **and** let **in terms of scope?**

- var is function-scoped or globally scoped if declared outside a function, while let is block-scoped.

25. **Explain the concept of a "pure function".**

- A pure function always produces the same output for the same input and does not cause any side effects (does not modify any external state).

26. **What is the "spread" operator in JavaScript and how is it different from the "rest" operator?**

- The **spread** operator (...) is used to expand elements of an array or object into individual elements, while the **rest** operator (...) is used to collect remaining elements into an array.

27. **What are Web Workers in JavaScript?**
- Web Workers allow for multi-threading in JavaScript by enabling background threads to run scripts concurrently with the main thread.

28. **What are "setImmediate" and "setTimeout"?**
- setImmediate() is used to execute code immediately after the current event loop cycle, while setTimeout() is used to delay code execution by a specified amount of time.

29. **Explain what** Symbol **is in JavaScript.**
- Symbol is a primitive data type introduced in ES6, which represents a unique and immutable value that is often used as property keys to avoid name conflicts.

30. **What are Observables in JavaScript?**
- Observables are a part of reactive programming (often used with libraries like RxJS) that provide a way to handle asynchronous data streams, enabling the composition of asynchronous events.

31. **Explain the difference between synchronous and asynchronous execution.**
- **Synchronous** execution means the code is executed in a sequential order, blocking the next operation until the current one finishes. **Asynchronous** execution allows operations to run in parallel, enabling non-blocking behavior.

- **Expert-Level JavaScript Interview Questions**

32. **What are WeakMaps and WeakSets?**
- A **WeakMap** is a collection of key-value pairs where the keys must be objects and are weakly referenced, meaning they don't prevent garbage collection. A **WeakSet** is similar, but only stores objects and also weakly references them.

33. **What is memoization in JavaScript?**
- Memoization is an optimization technique used to speed up functions by caching their results for previously encountered inputs.

34. **What is the event delegation pattern in JavaScript?**
- Event delegation is a technique where events are not directly bound to individual elements, but to a parent element. This improves performance and reduces the number of event listeners.

35. **What are generators in JavaScript?**
- A generator is a function that can be paused and resumed, allowing it to return multiple values one at a time. Generators are defined using function* and yield.

36. Explain the difference between Object.seal() **and** Object.freeze()**.**

- Object.seal() prevents new properties from being added and existing properties from being deleted, but allows modification of existing property values. Object.freeze() makes an object immutable, preventing changes to property values and structure.

37. What is the async/await **syntax in JavaScript?**

- async/await provides a cleaner and more readable way to handle asynchronous code. async functions always return a promise, and await pauses execution until the promise resolves.

38. What is the purpose of Object.defineProperty() **in JavaScript?**

- Object.defineProperty() allows you to define or modify properties on an object, including setting accessors (getter and setter), making properties read-only, or making them non-enumerable.

39. How does JavaScript handle memory management?

- JavaScript uses automatic memory management through garbage collection. The garbage collector reclaims memory that is no longer in use (i.e., objects that are no longer referenced).

40. What is the V8 engine, and how does it work?

- The V8 engine is Google's open-source JavaScript engine that powers Chrome and Node.js.

- **React JS Interview Question.**
- **Basic ReactJS Interview Questions.**

1. **What is React?**

- React is an open-source JavaScript library for building user interfaces, primarily for single-page applications where you can create reusable UI components.

2. **What are components in React?**

- Components are the building blocks of a React application. They are reusable and can either be class components or functional components. A component typically returns a JSX (JavaScript XML) code that defines the UI.

3. **What is JSX?**

- JSX is a syntax extension for JavaScript that looks similar to XML or HTML. It is used in React to describe the UI structure, allowing you to write HTML elements in JavaScript.

4. **What is the difference between a class component and a functional component?**

- **Class component:** Requires extending the `React.Component` class, and can have state and lifecycle methods.
- **Functional component:** A simpler form of a component that is written as a JavaScript function. With the introduction of React hooks, functional components can now also manage state and side effects.

5. **What are props in React?**

- Props (short for "properties") are used to pass data from parent to child components. Props are read-only and cannot be modified within the child component.

6. **What is state in React?**

- State is a data structure that holds information about the component's behaviour or appearance. Unlike props, state is managed within the component and can change over time.

7. **What are controlled components in React?**

- Controlled components are input elements (like `<input>`, `<textarea>`) where React manages their value. The component's state is the single source of truth for the input field.

8. **What is the Virtual DOM?**

- The Virtual DOM is a lightweight in-memory representation of the actual DOM elements. React uses it to optimize rendering by making minimal updates to the real DOM.

9. **What is the difference between state and props in React?**

- **State:** Managed within a component, and can be changed using `set State()` (class components) or `useState()` (functional components).
- **Props:** Passed from a parent component to a child component and cannot be changed by the child component.

10. **What are React Hooks?**

- React Hooks are functions that let you use state and other React features in functional components. Common hooks include `useState`, `useEffect`, and `useContext`.
- **Intermediate ReactJS Interview Questions.**

11. **What is the purpose of the useState hook?**

- useState allows you to add state to functional components. It returns an array with the current state value and a function to update it.

12. **What is the useEffect hook and how is it used?**

- useEffect is used to perform side effects in functional components, such as fetching data, subscribing to a service, or manually modifying the DOM. It takes a function and an optional dependency array to control when the effect runs.

13. **What is the purpose of the useContext hook?**

- useContext allows you to access the value of a React context directly without having to pass props through every level of the component tree.

14. **What is Context API?**

- The Context API is a feature in React that allows you to share values across the entire component tree without explicitly passing props to every level. It is often used for managing global state (e.g., user authentication, themes).

15. **What are higher-order components (HOCs)?**

- HOCs are functions that take a component and return a new component with additional props or behavior. They are used to reuse component logic.

16. **What is the difference between useEffect and componentDidMount?**

- **useEffect:** Can be used in functional components and runs after the render. It can also clean up after itself when a component unmounts or updates.
- **componentDidMount:** A lifecycle method used in class components that is called once, after the component mounts to the DOM.

17. **What are React Fragments?**

- React Fragments are used to group multiple elements without adding extra nodes to the DOM. It can be written as <React.Fragment> </React.Fragment> or using the shorthand <> </>.

18. **What is React Router?**

- React Router is a library that enables navigation between different components in a React application. It helps create single-page applications by managing routing inside the app.

19. **What is lazy loading in React?**

- Lazy loading is the technique of deferring the loading of a component until it is needed, improving performance by reducing the initial bundle size. In React, you can use React.lazy() and Suspense to implement lazy loading.

20. **What are error boundaries in React?**

- Error boundaries are components that catch JavaScript errors in their child components and display a fallback UI instead of crashing the entire app. Error boundaries implement the componentDidCatch lifecycle method.

21. **Advanced ReactJS Interview Questions.**

- **What is reconciliation in React?**
- Reconciliation is the process by which React updates the DOM to reflect changes in the component tree. React compares the Virtual DOM with the real DOM to determine the most efficient way to update the UI.

22. **What are keys in React and why are they important?**

- Keys are used to uniquely identify elements in a list of siblings in React. They help React optimize the rendering process by quickly identifying which elements need to be updated, added, or removed.

23. What is the role of shouldComponentUpdate in React?

- shouldComponentUpdate is a lifecycle method in class components that allows you to optimize rendering. It can be used to prevent unnecessary re-renders by returning false when the component doesn't need to update.

24. Explain the concept of React's "unidirectional data flow."

- React follows unidirectional data flow, meaning data flows from parent components to child components through props. Child components cannot modify the props they receive, but they can trigger events to notify the parent to update the state.

25. What is the use of React.memo?

- React.memo is a higher-order component that memoizes a functional component. It prevents unnecessary re-renders of the component by checking if the props have changed.

26. What is the purpose of useCallback and useMemo hooks?

- useCallback: Memoizes a function so that it is only recreated when its dependencies change.
- useMemo: Memoizes a value (such as the result of a computation) to prevent recalculating it unless its dependencies change.

27. How do you optimize performance in React applications?

- Use React.memo for functional components.
- Use shouldComponentUpdate for class components.
- Use the useMemo and useCallback hooks.
- Lazy load components.
- Code-splitting.
- Optimize state management to avoid unnecessary re-renders.

28. What is Server-Side Rendering (SSR) and how does it work with React?

- SSR is the process of rendering React components on the server rather than in the browser. The server sends a fully rendered HTML page to the client, which improves SEO and load times. Tools like Next.js are often used to enable SSR in React applications.

29. What is the difference between React.createElement and JSX?

- React.createElement is the underlying function that JSX compiles to. JSX is a syntactic sugar for React.createElement. JSX looks like HTML, but React.createElement creates JavaScript objects representing React elements.

30. What is the useLayoutEffect hook?

- useLayoutEffect is similar to useEffect but runs synchronously after all DOM mutations. It is useful for reading layout from the DOM and synchronously re-rendering the component.

31. What are the different ways of managing state in a React application?

- Local component state (useState).
- Global state (using Context API, useContext).
- State management libraries (Redux, MobX, Zustand).
- Server-side state (React Query, SWR).
- Persistent state (localStorage, sessionStorage).

32. How do you handle side effects in React?

- Side effects like data fetching, subscriptions, or manually interacting with the DOM can be managed using the useEffect hook in functional components. For class components, you can use lifecycle methods like componentDidMount, componentDidUpdate, and componentWillUnmount.