Q: What is the difference between block-level and inline elements in HTML?

A: Block-level elements occupy the full width available, starting on a new line, while inline elements take up as much width as necessary and don't start on a new line.

Q: What are semantic HTML tags, and why are they important?

A: Semantic HTML tags clearly describe their meaning in a human- and machine-readable way. They improve accessibility, SEO, and code readability.

Q: Explain the box model in CSS.

A: The box model consists of content, padding, border, and margin, determining the size and space an element occupies on the page.

Q: What is the difference between display: none and visibility: hidden?

A: display: none removes the element from the document, while visibility: hidden makes the element invisible but still occupies space.

Q: How would you implement a responsive design? What frameworks or methods do you prefer?

A: Responsive design can be implemented using media queries, flexbox, grid, or frameworks like Bootstrap and Tailwind CSS.

Q: What are media queries, and how do you use them?

A: Media queries are used to apply CSS based on device characteristics like screen size, typically for responsive design.

Q: Explain the difference between relative, absolute, fixed, and sticky positioning in CSS.

A: Relative is positioned relative to its normal position, absolute relative to its nearest ancestor, fixed relative to the viewport, and sticky toggles between relative and fixed.

Q: What is Flexbox, and when would you use it?

A: Flexbox is a layout model for one-dimensional layouts like rows or columns and is useful for

responsive designs.

Q: What is CSS Grid, and how does it differ from Flexbox?

A: CSS Grid is a two-dimensional layout system, useful for both rows and columns, while Flexbox is one-dimensional.

Q: How do you optimize CSS for performance?

A: To optimize CSS performance: minify CSS, remove unused styles, reduce file size, avoid deep selectors, and use modern layouts like Flexbox or Grid.

Q: What are closures in JavaScript?

A: A closure is a function that can access variables from its parent scope even after the parent function has closed.

Q: Explain the difference between let, const, and var.

A: let and const are block-scoped, var is function-scoped. let can be reassigned, const cannot.

Q: What is event delegation in JavaScript?

A: Event delegation attaches a single event listener to a parent element to handle events from child elements.

Q: What is the difference between synchronous and asynchronous JavaScript?

A: Synchronous operations block code execution until the current task is finished, while asynchronous allows other tasks to run in the background.

Q: Explain this keyword in JavaScript.

A: this refers to the object that is currently executing the code, which can vary depending on how a function is called.

Q: What are promises, and how do they differ from callbacks?

A: Promises represent an eventual completion or failure of an async task, and provide a cleaner way

to handle async logic compared to callbacks.

Q: What are arrow functions, and how do they differ from regular functions?

A: Arrow functions have a shorter syntax and do not have their own this context.

Q: Explain the difference between null, undefined, and NaN.

A: null is an intentional absence of value, undefined is an unassigned variable, NaN is a result of invalid mathematical operations.

Q: What is the difference between == and ===?

A: == compares values after type coercion, === compares both value and type.

Q: How does JavaScript handle event bubbling and event capturing?

A: Event bubbling propagates events from the target element up the DOM tree, while event capturing propagates from the outermost element down.

Q: What is the virtual DOM, and how does React use it?

A: Virtual DOM is a lightweight representation of the real DOM. React updates the virtual DOM and selectively updates the real DOM.

Q: Explain the lifecycle methods of a React component.

A: React lifecycle methods: Mounting (componentDidMount), Updating (componentDidUpdate), Unmounting (componentWillUnmount).

Q: What are hooks in React, and how do they differ from class components?

A: Hooks allow state and other features in functional components, eliminating the need for class components.

Q: What is the useEffect hook used for?

A: useEffect is used to handle side effects, such as fetching data or manually changing the DOM.

Q: How does useState work in functional components?

A: useState allows local state to be added to functional components, providing a state variable and a function to update it.

Q: What is prop drilling, and how can you avoid it?

A: Prop drilling is passing props through many layers of components. It can be avoided with Context API or Redux.

Q: What are controlled and uncontrolled components in React?

A: Controlled components have form data managed by React state, while uncontrolled components rely on the DOM to manage form data.

Q: What are higher-order components (HOCs) in React?

A: Higher-order components are functions that take a component and return a new component with additional behavior.

Q: Explain the concept of state and props in React.

A: State is mutable and managed within the component, while props are read-only and passed from parent to child.

Q: How do you handle form submissions in React?

A: Use controlled components, capture the form's input, and handle it using an event handler like onSubmit.

Q: What are the advantages of using a build tool like Webpack or Parcel?

A: Build tools bundle files, optimize code (minification, tree-shaking), and enable module hot-reloading.

Q: How do you optimize the performance of a web application?

A: Optimize web apps by lazy loading, code splitting, minimizing CSS/JS, compressing images,

caching, and using CDNs.

Q: What is lazy loading, and how do you implement it?

A: Lazy loading defers loading of resources until they are needed, implemented with dynamic imports or image attributes.

Q: What is tree shaking in JavaScript bundlers?

A: Tree shaking eliminates unused code (dead code) in the final bundle, resulting in smaller output.

Q: How would you handle browser compatibility issues?

A: Browser compatibility can be handled using feature detection, CSS prefixes, polyfills, and transpiling using tools like Babel.

Q: Explain how you would set up continuous integration and deployment for a front-end project.

A: Use CI/CD tools (GitHub Actions, Jenkins, etc.) to automate testing, building, and deploying your front-end project.

Q: What are some ways to reduce the load time of a website?

A: Reduce load times by minimizing HTTP requests, compressing assets, enabling caching, and using a CDN.

Q: What are RESTful APIs, and how do they work?

A: RESTful APIs use HTTP methods to interact with resources (GET, POST, PUT, DELETE) and exchange data, often using JSON.

Q: What is CORS, and how do you handle it in a front-end application?

A: CORS prevents cross-origin requests unless permitted by the server. Handle it by adding appropriate headers on the server.

Q: What is a Single Page Application (SPA), and how does it differ from a traditional web app?

A: SPAs load a single HTML page and dynamically update content using JavaScript, whereas

traditional web apps reload the entire page.

Q: How does client-side routing work in a SPA?

A: Client-side routing changes the browser URL and renders different views without refreshing the page, typically using libraries like React Router.

Q: Explain the differences between localStorage, sessionStorage, and cookies.

A: localStorage persists data even after the browser is closed, sessionStorage clears on browser close, cookies are small and sent with HTTP requests.