1. `<link href="style.css" />` **vs** `@import("something.css")`

Both methods serve the same purpose: linking CSS files to an HTML document. However, `<link>` is preferred as it is faster and more efficient. It allows the browser to load CSS files concurrently with HTML. On the other hand, `@import` is slower as it delays CSS loading and blocks rendering of the page. It's generally better to use `<link>` in modern development.

2. **Spread and Rest**

Both spread and rest operators in JavaScript are denoted by `...`. The spread operator is used to unpack elements from an array or object, typically to pass them as arguments. The rest operator collects multiple elements into a single array. They look identical but serve opposite purposes in handling data.

3. **ES6 vs ES5 Features**

ES6, or ECMAScript 2015, introduced many modern features such as arrow functions, promises, template literals, and `let`/`const` for variable declaration. It also added modules, classes, and destructuring. ES5, the previous version, lacked these features and used `var` for variable declarations, along with callback-based handling for asynchronous code.

4. **How React Components React to Each Other**

In React, components can communicate via props and state. Parent components pass data to child components through props, and child components can send data back using callback functions. React's virtual DOM ensures efficient updates and renders only the parts of the UI that change.

5. **Communication Between Two Browser Tabs**

Tabs can communicate via the `localStorage` or `sessionStorage` events. One tab can store data in these storage mechanisms, and the other tab can listen for changes using the `storage` event. This allows two tabs to exchange information in real-time.

6. **rem vs em**

Both `rem` and `em` are relative units used for font sizes and layout styling in CSS. `rem` stands for "root em" and is based on the font size of the root element (usually `<html>`), making it consistent throughout the page. `em` is relative to the font size of its parent element, so it can change depending on its context.

7. **Closure**

A closure is a function that retains access to its lexical scope, even after the outer function has finished executing. This allows inner functions to "remember" the environment in which they were created, making them useful for creating private variables or functions.

8. **React Lifecycle Hooks in Functional Components**

React's lifecycle methods (such as `componentDidMount`, `componentWillUnmount`, etc.) are typically used in class components. In functional components, React hooks like `useEffect` serve the same purpose. The `useEffect` hook runs side effects like data fetching or subscriptions when a component mounts or updates.

9. **useState vs Props**

`useState` is a hook used to declare state in functional components. It allows you to create a state variable and update it. Props, on the other hand, are immutable and passed from parent to child components. They allow parent components to provide data to child components but cannot be modified by the child component itself.

useMemo
`useMemo` is a React hook used to memoize the result of a computation to avoid recalculating it unnecessarily. It optimizes performance by recomputing only when its dependencies change. Useful for expensive calculations.

useCallback
`useCallback` is a React hook used to memoize a function so that it doesn't get recreated on every render. It is beneficial when passing callbacks to child components to prevent unnecessary re-renders.

memo
`memo` is a higher-order component that wraps a React component to memoize its rendering. It ensures the component only re-renders when its props change, improving performance for components with static or rarely changing props.

useRef
`useRef` is a React hook that creates a mutable reference object. It is often used to persist values across renders without causing re-renders, or to directly access and manipulate DOM elements.

What is Redux:
Redux is a state management library for JavaScript applications, commonly used with React. It provides a predictable state container to manage the application's state globally.

Why:
Redux helps manage complex application states, especially in large-scale apps. It centralizes state, making debugging, testing, and sharing data between components easier.

How:

1. **Dispatch**: Use `dispatch` to send actions to the store to update the state.
2. **Select**: Use `useSelector` or `mapStateToProps` to retrieve specific parts of the state from the store.

Simple Tips to Optimize Frontend Performance

- Split code into smaller parts and load them when needed.
- Load images and components only when they are visible (lazy loading).
- Compress files like CSS, JavaScript, HTML, and images.
- Use a Content Delivery Network (CDN) for faster asset delivery.
- Reduce the number of server requests by combining files.
- Cache static assets to avoid loading them repeatedly.
- Use modern JavaScript features and remove unused code.
- Minimize animations and complex DOM operations.
- Analyze performance using tools like Lighthouse or DevTools.