● When and why do we need lazy()?

To split/chunking the code whenever the page has been loaded it will load separate JS file

● What is suspense?

Suspense is a powerful tool for managing asynchronous operations in React applications and can help improve the perceived performance and user experience. Keep in mind that the usage of suspense and concurrent mode is evolving, and it's recommended to check the React documentation for the latest updates and best practices.

● Why we got this error : A component suspended while responding to synchronous input. This will cause the UI to be replaced with a loading indicator. To fix, updates that suspend should be wrapped with startTransition? How does suspense fix this error?

The error message you provided indicates that a React component suspended while handling synchronous input, which could lead to the UI being replaced with a loading indicator. To address this issue, it suggests wrapping the updates that suspend with startTransition.

Let's break down the error and the solution:

Error Explanation:

A component suspended during the rendering process.

The suspension occurred while responding to synchronous input (likely user interactions or other synchronous events).

Without proper handling, this could result in the UI being replaced with a loading indicator, causing a poor user experience.

Solution:

The error suggests using startTransition to wrap the updates that might suspend.

startTransition is a part of the concurrent mode feature in React, which helps manage asynchronous rendering and improves the overall user experience.

By using startTransition, you can indicate to React that the updates within that transition should not block the user interface immediately. Instead, React can prioritize rendering the visible changes first and then work on the suspended updates in the background.

● Advantages and disadvantages of using this code splitting pattern?

Advantages:

Reduced Initial Loading Time:

Code splitting helps reduce the initial loading time of the application by loading only the essential code needed for the initial view.

Improved Page Responsiveness:

By loading code chunks asynchronously, the main thread is not blocked, resulting in improved page responsiveness, especially on slower network connections.

Optimized Bandwidth Usage:

Users only download the code they need when they navigate to a specific part of the application, which optimizes bandwidth usage and reduces unnecessary data transfer.

Better User Experience:

Users experience faster page loads and smoother interactions, leading to an overall better user experience.

Granular Caching:

Smaller code chunks are more cacheable, and when updates occur, only the changed chunks need to be reloaded, reducing the amount of data transferred.

Parallel Loading:

Multiple code chunks can be loaded in parallel, taking advantage of the client's ability to make concurrent network requests.

Dynamic Imports:

React.lazy allows for dynamic imports, making it easier to conditionally load components based on user interactions or application state.

Disadvantages:

Complexity in Development:

Implementing code splitting with React.lazy and Suspense introduces some complexity to the development process, as developers need to consider where to split the code and manage the state during loading.

Learning Curve:

Developers new to code splitting and React.lazy may need time to understand the concepts and integrate them effectively into their applications.

Potential for Overhead:

For small applications or applications with relatively simple navigation flows, the overhead introduced by code splitting might outweigh the benefits.

Route-Based Splitting Challenges:

Implementing code splitting based on routes might require careful consideration to ensure that the split points align with logical sections of the application.

Not Suitable for All Scenarios:

Code splitting may not be suitable for all types of applications. In some cases, the additional complexity and overhead may not provide significant performance gains.

Compatibility Concerns:

Some older browsers or environments might not fully support dynamic imports and code splitting, potentially leading to compatibility issues.

● When do we and why do we need suspense?

Suspense in React is a feature that allows components to "suspend" rendering while waiting for some asynchronous operation to complete, such as data fetching or lazy-loading components. It was introduced to improve the user experience by providing a better way to handle asynchronous operations in a more declarative and efficient manner.