Chapter 09 - Optimizing our App

Theory -

- When and why do we need lazy()?

  We use lazy() to **load components only when needed**, not during the initial load.
  This helps:
  - Split code into smaller chunks (code splitting)
  - Reduce initial load time
  - Improve app performance, especially for large apps or slow networks

- What is suspense?

  Suspense lets you **show a fallback UI (like a loader)** while React loads a component or fetches data asynchronously.
  It works with lazy() to improve user experience and reduce code complexity.

- 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?

  This error happens when a **component tries to fetch data or load async code during a sync action** like a click or input.

- **How to fix:**

  Wrap the update with `startTransition()` so React knows it's a **non-urgent update**. Also, make sure Suspense is used properly with a fallback to avoid UI jank.

- Advantages and disadvantages of using this code splitting pattern?

  Advantages:
  - Faster first load
  - Better performance
  - Only load what's needed
  - Great for slow devices or networks
  - Easier maintenance of smaller chunks

  Disadvantages:
  - Slight delay when loading on-demand components
  - Needs extra setup
  - Testing becomes harder
  - Async logic can complicate things

- When do we and why do we need suspense?

  Use Suspense when you:

  - Want to **load components or data async**
  - Need to **show a loader during fetch**
  - Want to keep code clean and user-friendly

Coding -

- Create your custom hooks
- Try out lazy and suspense
- Make your code clean.

References:

- https://reactjs.org/docs/hooks-custom.html
- https://beta.reactjs.org/apis/react/lazy#suspense-for-code-splitting