1. **When and why do we need lazy()?**

* The React.lazy function lets you render a dynamic import as a regular component.
* React.lazy takes a function that must call a dynamic import(). This must return a Promise which resolves to a module with a default export containing a React component.

1. **What is suspense?**

* Suspense lets your components “wait” for something before they can render.
* Suspense is not a data fetching library. It’s a **mechanism for data fetching libraries** to communicate to React that *the data a component is reading is not ready yet*.
* React can then wait for it to be ready and update the UI.

1. **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?**

* Any component may suspend as a result of rendering, even components that were already shown to the user.
* In order for screen content to always be consistent, if an already shown component suspends, React has to hide its tree up to the closest <Suspense> boundary.
* However, from the user’s perspective, this can be disorienting.
  + import React, { Suspense } from 'react';
  + import Tabs from './Tabs';
  + import Glimmer from './Glimmer';
  + const Comments = React.lazy(() => import('./Comments'));
  + const Photos = React.lazy(() => import('./Photos'));
  + function MyComponent() {
  + const [tab, setTab] = React.useState('photos');
  + function handleTabSelect(tab) {
  + setTab(tab);
  + };
  + return (
  + <div>
  + <Tabs onTabSelect={handleTabSelect} />
  + <Suspense fallback={<Glimmer />}>
  + {tab === 'photos' ? <Photos /> : <Comments />}
  + </Suspense>
  + </div>
  + );
  + }
* In this example, if tab gets changed from 'photos' to 'comments', but Comments suspends, the user will see a glimmer.
* This makes sense because the user no longer wants to see Photos, the Comments component is not ready to render anything, and React needs to keep the user experience consistent, so it has no choice but to show the Glimmer above.
* However, sometimes this user experience is not desirable. In particular, it is sometimes better to show the “old” UI while the new UI is being prepared. You can use the new [startTransition](https://reactjs.org/docs/react-api.html#starttransition) API

1. **Advantages and disadvantages of using this code splitting pattern?**

* Advantages -
  + Improves the performance of the app.
  + Improves the impact on memory.
  + Improves the downloaded Kilobytes (or Megabytes) size.
* Disadvantages –
  + The only drawback about code splitting is that you can only use it in client side rendering.
  + Both the above techniques won’t work during server side rendering when it comes to React.

1. **When and why do we need suspense?**

* The lazy component should then be rendered inside a Suspense component, which allows us to show some fallback content (such as a loading indicator) while we’re waiting for the lazy component to load.
* The fallback prop accepts any React elements that you want to render while waiting for the component to load.
* You can place the Suspense component anywhere above the lazy component. You can even wrap multiple lazy components with a single Suspense component.