Assignment - 9 : Solution

Ques 1: When and why do we need lazy()?

Solu 1:

- When we want to load the component dynamically then we need lazy().
- React.lazy() allows developers to easily create components with dynamic imports and render them as normal components.
- When the component is rendered, it will automatically load the bundle that contains the rendered component.
- You need to provide a single input parameter to call React.lazy(). It accepts a function as an input parameter, and that function should return a promise after loading the component using import(). Finally, the returned promise from React.lazy() will give you a module with a default export containing the React component.

```
// Without React.lazy()

import AboutComponent from './AboutComponent ';

// With React.lazy()

const AboutComponent = React.lazy(() => import('./AboutComponent '));

const HomeComponent = () => (

  <div><AboutComponent /></div>

)
```

- Why
  i. Reduces initial loading time by reducing the bundle size.
  ii. Reduces browser workload.
  iii. Improves application performance in low bandwidth situations.
  iv. Improves user experience at initial loading.
  v. Optimizes resource usage.

Ques 2: What is suspense?

Solu 2 :

- When we use lazy loading, components are rendered as we navigate. So, we need to have a placeholder for those components until they are loaded.
- As a solution, React.Suspense was introduced, and it acts as a wrapper for the lazy components.
- You can pass the waiting or loading message as the fallback prop, and it will be displayed until the wrapped lazy component is rendered.

```
import React, { Suspense } from "react";

const AboutComponent = React.lazy(() => import('./AboutComponent'));


const HomeComponent = () => (

    <div><Suspense fallback = { <div> Please Wait... </div> } >

            <AboutComponent /></Suspense></div>

);
```

Ques 3: Why we got this error : A component suspended while responding to synchronous input. This will cause the UI to be replaced with loading indicator. To fix, updates that suspend should be wrapped with startTransition? How does suspense fix this error?

Solu 3:

- This error is thrown as Exception by React when the promise to dynamically import the lazy component is not yet resolved and the Component is expected to render in the meantime.
- If only the dynamic import is done and there is no component then this error is shown. React expects a Suspense boundary to be in place for showing a fallback prop until the promise is getting resolved.
- If showing the shimmer (loading indicator) is not desirable in some situations, then startTransistion API can used to show the old UI while new UI is being prepared. React do this without having to delete or remove the Suspense component or its props from

your code.

Ques 4: Advantages and disadvantages of using this code splitting pattern?

Solu 4: Advantage of Lazy loading :

- Reduces initial loading time by reducing the bundle size.
- Reduces browser workload.
- Improves application performance in low bandwidth situations.
- Improves user experience at initial loading.
- Optimizes resource usage.

Disadvantage of Lazy loading :

- Not suitable for small-scale applications.
- Requires additional communication with the server to fetch resources.
- Needs extra piece of code for showing load time like shimmer

Ques 5: When do we and why do we need suspense?

Solu 5:

- When we use lazy loading, components are rendered as we navigate. So, we need to have a placeholder for those components until they are loaded. As a solution, React.Suspense was introduced, and it acts as a wrapper for the lazy components.
- You can wrap a single lazy component, multiple lazy components, or multiple lazy components with different hierarchy levels with React.Suspense. In addition, it accepts a prop named fallback as the placeholder, and you can pass a component or an element for that.
- For example, you can pass the waiting or loading message as the fallback prop, and it will be displayed until the wrapped lazy component is rendered.

```
import React, { Suspense } from "react";
```

```
const AboutComponent = React.lazy(() => import('./AboutComponent'));


const HomeComponent = () => (
        <div><Suspense fallback = { <div> Please Wait... </div> } >
                        <AboutComponent /></Suspense></div>
);
```