Lazy Loading EXPLAINED

A complete guide with a cheat sheet
and a concrete example.

BRIEF INTRODUCTION

Lazy loading is a highly effective performance management technique used in web programming, especially with modern libraries and frameworks like React. This method involves loading components or resources only when necessary, typically in response to a user action or when these elements are about to be displayed on the screen.

This can reduce the initial load time of the application, decrease resource consumption, and improve the user experience on devices with limited performance or slow internet connections.

HOW LAZY LOADING WORKS WITH REACT

React implements lazy loading primarily through the React.lazy() function, which allows for dynamically loading a component. This is often combined with React.Suspense, which handles the loading state of the component.

Here are the key steps to understand how to implement lazy loading in a React application:

1 - Dynamic Import with React.lazy()

React.lazy() allows for asynchronously importing a component. It uses JavaScript's dynamic import (import()) to load a component only when it is required. The result is a promise that resolves to a module containing a default React component.

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

In this example, LazyComponent is loaded only when React attempts to render it for the first time.

2 - Wrapping with React.Suspense

React.Suspense wraps lazily loaded components. It allows displaying a loading indicator or other fallback content while the component is loading.

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

function App() {
  return (
    <div>
      <Suspense fallback={<div>Loading...</div>}>
        <LazyComponent />
      </Suspense>
    </div>
  );
}
```

Here, <Suspense> waits for LazyComponent to load before displaying it. If the loading takes time, the user will see "Loading...".

Error Handling

React.Suspense also allows for handling errors during component loading. If the loading fails for some reason (e.g., a network outage), you can display an error message or a fallback component.

3 -Integration with Routing

Lazy loading is particularly useful in the context of routing, where different screens or components are loaded based on the URL. With routers like React Router, you can use React.lazy to dynamically load components for each route.

```
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import React, { Suspense } from 'react';

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

function App() {
  return (
    <Router>
      <Suspense fallback={<div>Loading...</div>}>
        <Switch>
          <Route path="/about" component={About} />
          <Route path="/" component={Home} />
        </Switch>
      </Suspense>
    </Router>
  );
}
```