
Taller de Optimización y Memoización en React

Objetivo

Poner en práctica las técnicas de optimización y memoización aprendidas en esta guía. Los estudiantes trabajarán con diferentes escenarios y aplicarán React.memo, useMemo, y useCallback para mejorar el rendimiento de una aplicación de React.

Instrucciones

1. **Crear una Aplicación React Simple**
    - Crea una nueva aplicación de React con create-react-app.
    - Crea una estructura de componentes que incluya un contador, una lista de elementos, y un botón para cambiar el tema (oscuro/claro).
2. **Parte 1: Optimización con React.memo**
    - Crea un componente de lista que reciba una lista de elementos como prop.
    - Memoriza el componente de lista usando **React.memo**.
    - Haz que el componente solo se vuelva a renderizar cuando la lista cambie.

Preguntas:

- ¿Cómo cambia el comportamiento del componente cuando se usa React.memo?
- ¿Qué situaciones pueden causar que el componente se vuelva a renderizar?

3. **Parte 2: Uso de useMemo para Cálculos Costosos**
    - Agrega una función costosa que se calcule cuando el estado de la lista cambie.
    - Usa **useMemo** para memorizar el valor calculado.
    - Verifica que la función solo se ejecute cuando sus dependencias cambien.

Preguntas:

- ¿Cuándo debería usarse useMemo?
- ¿Qué sucede si no se usa useMemo en un cálculo costoso?

4. **Parte 3: Uso de useCallback para Funciones Memorables**

- Crea un componente de botón que reciba una función handleClick como prop.
- Usa **useCallback** para memorizar la función y evitar que el botón se renderice innecesariamente.
- Verifica el comportamiento del componente antes y después de usar useCallback.

Preguntas:

- ¿En qué escenarios es útil useCallback?
- ¿Cómo se determina cuándo una función debe memorizarse?

5. **Parte 4: Lazy Loading y Suspense**
   - Implementa **Lazy Loading** para cargar un componente solo cuando sea necesario.
   - Usa React.lazy y Suspense para dividir el código y mejorar la carga inicial de la aplicación.

Preguntas:

- ¿Cómo ayuda el Lazy Loading a mejorar el rendimiento de la aplicación?
- ¿Qué consideraciones debes tener al usar Suspense?

Desafíos Adicionales

1. **Crear un Perfil de Rendimiento**: Usa las herramientas de desarrollo de tu navegador para crear un perfil de rendimiento de tu aplicación antes y después de aplicar estas optimizaciones. Analiza la diferencia en tiempo de renderizado.
2. **Implementar un Tema Global**: Usa el contexto de React (React.Context) para manejar el estado del tema (oscuro/claro) y verifica cómo afectan las técnicas de memoización al rendimiento.