Mengatasi Masalah Performa dengan `useMemo` dalam React

Pengertian

Pada aplikasi React, saat aplikasi mulai berkembang dengan banyak state dan props, kita mungkin mengalami masalah performa ketika komponen harus merender ulang secara berlebihan. Salah satu cara untuk mengatasi masalah ini adalah dengan menggunakan **`useMemo`**. Hook ini membantu untuk mengoptimalkan aplikasi dengan menghindari perhitungan ulang atau render ulang komponen yang tidak diperlukan. Dalam kode berikut, kita menggunakan `useMemo` untuk mengoptimalkan perhitungan daftar workout berdasarkan waktu yang ditampilkan di aplikasi.

Cara Berpikir React

React merender ulang komponen setiap kali state atau props berubah. Namun, jika ada operasi atau perhitungan yang memakan waktu lama atau tidak perlu diulang, kita bisa mengoptimalkan komponen dengan menggunakan `useMemo`.

Analogi Sederhana

Bayangkan Anda bekerja di dapur, dan setiap kali Anda mendapatkan pesanan, Anda harus menghitung ulang semua bahan yang diperlukan. Namun, bahan-bahan tertentu tidak berubah setiap kali pesanan datang. Dengan menggunakan **`useMemo`**, Anda hanya menghitung bahan yang berubah berdasarkan pesanan (misalnya, sesuai dengan waktu dalam contoh kita), dan bahan yang tidak berubah, Anda biarkan tetap tersedia tanpa perlu dihitung ulang. Ini menghemat waktu dan usaha yang tidak perlu.

Penjelasan Kode

Mari kita uraikan bagian kode yang berfokus pada optimasi performa menggunakan `useMemo`:

1. **State dan Pengaturan Waktu**

```
const [time, setTime] = useState(formatTime(new Date()));
```

- **`time`**: Variabel state yang menyimpan waktu yang diformat.
- **`setTime`**: Fungsi untuk memperbarui waktu.
- **`formatTime`**: Fungsi untuk memformat waktu saat ini dengan format yang mudah dibaca (termasuk bulan, tahun, jam, menit, detik).

2. **Menggunakan `useMemo` untuk Daftar Workout**

```
const workouts = useMemo(() => {
  return [
    {
```

```
        name: "Full-body workout",
        numExercises: partOfDay === "AM" ? 9 : 8,
      },
      {
        name: "Arms + Legs",
        numExercises: 6,
      },
      {
        name: "Arms only",
        numExercises: 3,
      },
      {
        name: "Legs only",
        numExercises: 4,
      },
      {
        name: "Core only",
        numExercises: partOfDay === "AM" ? 5 : 4,
      },
    ];
  }, [partOfDay]);
```

- **`useMemo`**: Hook ini digunakan untuk mengingat hasil perhitungan `workouts` dan hanya menghitung ulang daftar workout jika **`partOfDay`** berubah (dari AM ke PM atau sebaliknya).
- **`partOfDay`**: Berdasarkan waktu yang sedang ditampilkan (AM atau PM), daftar workout yang ditampilkan akan bervariasi. Misalnya, jumlah latihan untuk "Full-body workout" dan "Core only" berbeda tergantung waktu (AM atau PM).
- **`useMemo`** memastikan bahwa perhitungan daftar workout tidak akan terjadi setiap kali state lain berubah (misalnya, setiap detik karena waktu diperbarui), yang dapat memperlambat aplikasi jika daftar workout dihitung ulang tanpa perlu.

3. **Menampilkan Waktu dan Mengatur Interval**

```
useEffect(function () {
  const id = setInterval(function () {
    setTime(formatTime(new Date()));
  }, 1000);

  return () => clearInterval(id);
}, []);
```

- **`useEffect`** digunakan untuk mengupdate waktu setiap detik dengan menggunakan **`setInterval`**. Setiap detik, waktu diformat ulang dan diperbarui.
- Ini memberi pengguna waktu yang terus diperbarui di UI, tetapi kita perlu berhati-hati agar pembaruan ini tidak menyebabkan perhitungan ulang yang tidak perlu pada komponen lain (seperti daftar workout). Dengan menggunakan `useMemo`, kita dapat menghindari perhitungan ulang `workouts` setiap kali waktu diperbarui.

4. **Komponen yang Dikirimkan ke UI**

```
<ToggleSounds allowSound={allowSound} setAllowSound={setAllowSound} />
<Calculator workouts={workouts} allowSound={allowSound} />
```

- **`ToggleSounds`** dan **`Calculator`** adalah komponen yang menerima `workouts` dan `allowSound` sebagai props.
- **`useMemo`** memastikan bahwa komponen `Calculator` tidak menerima prop `workouts` yang dihitung ulang setiap kali waktu diperbarui, sehingga mencegah render yang tidak perlu.

Mengapa `useMemo` Diperlukan untuk Performa?

1. **Menghindari Render Ulang yang Tidak Perlu**

- Setiap kali state atau props berubah, React akan merender ulang komponen. Jika kita tidak menggunakan `useMemo`, daftar workout akan dihitung ulang setiap kali waktu diperbarui setiap detik, yang menyebabkan **render yang tidak perlu**.
- Dengan menggunakan `useMemo`, kita memastikan bahwa daftar workout hanya dihitung ulang jika ada perubahan yang benar-benar relevan (yaitu, ketika `partOfDay` berubah), yang membantu mengurangi jumlah render.

2. **Mengurangi Beban Komputasi**

- Tanpa `useMemo`, perhitungan daftar workout akan dilakukan setiap kali komponen dirender ulang, yang mengarah pada **beberapa perhitungan yang tidak efisien**.
- `useMemo` hanya akan melakukan perhitungan ulang jika dependensi (dalam hal ini `partOfDay`) berubah, yang berarti hanya saat itu perhitungan baru yang dilakukan.

3. **Menjaga Responsivitas Aplikasi**

- Dengan mengoptimalkan render dengan `useMemo`, aplikasi tetap responsif meskipun ada pembaruan waktu setiap detik. Jika kita tidak menggunakan `useMemo`, aplikasi bisa menjadi lebih lambat, terutama ketika jumlah komponen yang harus dirender bertambah banyak.

Kesimpulan

Dalam aplikasi React ini, kita menggunakan **`useMemo`** untuk mengoptimalkan perhitungan daftar workout yang bergantung pada waktu (AM/PM). Tanpa **`useMemo`**, setiap pembaruan waktu akan menyebabkan perhitungan ulang daftar workout yang tidak perlu, memperburuk performa aplikasi.

Dengan menggunakan **`useMemo`**, kita memastikan bahwa perhitungan hanya dilakukan saat benar-benar diperlukan, yaitu ketika **`partOfDay`** berubah. Teknik ini sangat berguna untuk mengurangi render ulang yang tidak perlu, menjaga aplikasi tetap responsif dan efisien.

Penggunaan **`useMemo`** adalah solusi yang sangat tepat dalam kasus ini untuk mengatasi masalah performa yang timbul akibat render ulang komponen yang tidak diperlukan.