**PENGEMBANGAN WEB (TEORI)**

**LAPORAN EKSPERIMEN MENGENAI PROBLEM EVENT HANDLING YANG TERLALU SERING DIPANGGIL DAPAT MENURUNKAN PERFORMA APLIKASI**

*Laporan ini disusun untuk memenuhi tugas 1 mata kuliah Pengembangan Web (Teori)*



Disusun oleh kelompok B4:

**Asri Husnul Rosadi 221524035**

Faris Abulkhoir 221524040

Mahardika Pratama 221524044

Muhamad Fahri Yuwan 221524047

Najib Alimudin Fajri 221524053

Septyana Agustina 221524058

Sarah 221524059

Dosen Pengampu:

Joe Lian Min, M.Eng.

**JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA**

**PROGRAM STUDI D4 TEKNIK INFORMATIKA**

**POLITEKNIK NEGERI BANDUNG**

**2024**

# **DAFTAR ISI**

[**DAFTAR ISI** i](#_Toc176203121)

[**A.** **IDENTIFIKASI PROBLEM** 1](#_Toc176203122)

[**B.** **DESKRIPSI PROBLEM** 1](#_Toc176203123)

[**C.** **METODOLOGI EKSPERIMEN** 1](#_Toc176203124)

[**D.** **PELAKSANAAN EKSPERIMEN** 2](#_Toc176203125)

[**E.** **ANALISIS HASIL EKSPERIMEN** 3](#_Toc176203126)

[**F.** **KESIMPULAN** 3](#_Toc176203127)

Link chatGPT : https://chatgpt.com/share/3782a324-8dda-439c-a71a-c633d27c6a8b

# **IDENTIFIKASI PROBLEM**

Masalah yang diidentifikasi adalah seringnya pemanggilan fungsi secara berlebihan saat menangani event seperti input teks atau scrolling, yang dapat menyebabkan penurunan performa aplikasi. Masalah ini terjadi terutama pada event yang dipicu berulang-ulang dalam waktu singkat, seperti mengetik cepat atau menggeser halaman.

# **DESKRIPSI PROBLEM**

Pemanggilan fungsi tanpa kontrol pada event yang sering terjadi dapat menyebabkan beban berlebih pada sistem, memperlambat performa, dan menyebabkan aplikasi terasa lamban atau tidak responsif. Contohnya, saat mengetik dalam kolom pencarian, setiap huruf yang ditekan memicu fungsi pencarian yang tidak perlu, atau saat scrolling halaman panjang, event scroll dapat memanggil fungsi ratusan kali dalam beberapa detik. Debouncing dan throttling adalah solusi yang memungkinkan kontrol frekuensi pemanggilan fungsi untuk meningkatkan performa.

# **METODOLOGI EKSPERIMEN**

Metodologi eksperimen yang digunakan melibatkan dua pendekatan:

1. **Debouncing:** Fungsi event hanya dipanggil setelah aktivitas berhenti dalam waktu tertentu.
2. **Throttling:** Fungsi event dipanggil pada interval waktu yang tetap, terlepas dari seberapa sering event tersebut terjadi.

Eksperimen ini dilakukan dengan membuat dua skenario:

* Pertama, menangani event input pada field teks dengan dan tanpa debounce.
* Kedua, menangani event scroll dengan dan tanpa throttle.

Hasil eksperimen dianalisis menggunakan Developer Tools pada browser untuk memantau performa aplikasi, terutama dengan melihat jumlah pemanggilan fungsi dan dampaknya terhadap kinerja.

# **PELAKSANAAN EKSPERIMEN**

1. **Persiapan Lingkungan:**

* Dibuat dua skenario eksperimen menggunakan HTML dan JavaScript.
* Skenario pertama untuk event input pada kolom teks tanpa debouncing dan dengan debouncing.
* Skenario kedua untuk event scroll tanpa throttling dan dengan throttling.

1. **Implementasi Skenario:**

* **Tanpa Debounce/Throttle:** Event listener untuk input dan scroll dipasang langsung tanpa pembatasan. Setiap kali event terjadi, fungsi dipanggil dan dicatat ke dalam log.
* **Dengan Debounce/Throttle:** Fungsi debounce dipasang pada event input dan fungsi throttle dipasang pada event scroll. Event listener diperbarui untuk menangani pemanggilan fungsi dengan kontrol frekuensi melalui debounce dan throttle.

1. **Pengumpulan Data:**

* Menggunakan Developer Tools pada browser (Chrome), performa aplikasi dipantau dengan melihat log di Console dan memeriksa rekaman aktivitas pada tab Performance.
* Data dikumpulkan dalam bentuk frekuensi pemanggilan fungsi serta beban yang dihasilkan pada aplikasi.

# **ANALISIS HASIL EKSPERIMEN**

|  |  |
| --- | --- |
| **Tanpa Debounce & Throttling** | **Dengan Debounce & Throttling** |
|  |  |

* **Tanpa Debounce:**

Setiap kali pengguna mengetik, fungsi langsung dipanggil berkali-kali, bahkan saat input belum selesai. Ini menghasilkan banyak log yang tidak perlu dan memperlambat aplikasi saat input diberikan dengan cepat.

* **Dengan Debounce:**

Fungsi hanya dipanggil sekali setelah pengguna berhenti mengetik selama interval waktu tertentu. Hal ini mengurangi jumlah log secara drastis dan mencegah pemborosan sumber daya. Performanya menjadi lebih stabil dan aplikasi lebih responsif.

* **Tanpa Throttle:**

Event scroll memanggil fungsi terus-menerus selama pengguna menggulir halaman. Hal ini menyebabkan terlalu banyak log dan potensi penurunan performa pada perangkat dengan spesifikasi rendah.

* **Dengan Throttle:**

Fungsi scroll hanya dipanggil pada interval waktu yang diatur (misalnya, setiap 200ms). Ini mengurangi jumlah pemanggilan fungsi dan meningkatkan performa, terutama saat pengguna menggulir halaman dengan cepat.

# **KESIMPULAN**

Penggunaan debouncing dan throttling terbukti efektif dalam mengoptimalkan event handling pada aplikasi web. Debouncing berhasil mengurangi pemanggilan fungsi yang tidak perlu selama input teks, sementara throttling mampu membatasi pemanggilan fungsi scroll yang berlebihan. Implementasi kedua teknik ini dapat meningkatkan performa aplikasi secara signifikan, menjadikannya lebih efisien dan responsif, terutama saat menangani event yang sering terjadi secara berurutan dalam waktu singkat. Selalu lakukan pengujian mendalam untuk menilai dampak keseluruhan dari teknik optimasi dalam konteks aplikasi spesifik.

**Debouncing**

**Kelebihan:**

1. **Mengurangi Pemanggilan Fungsi Berlebihan:**
   * Fungsi hanya dipanggil sekali setelah periode waktu tertentu setelah aktivitas berhenti, mengurangi jumlah pemanggilan fungsi yang tidak perlu.
2. **Meningkatkan Performa Aplikasi:**
   * Meminimalkan beban server dan penggunaan sumber daya dengan menghindari eksekusi fungsi yang terlalu sering.
3. **Mengurangi Latensi:**
   * Fungsi dieksekusi hanya setelah pengguna berhenti berinteraksi, memberikan respons yang lebih efisien.

**Kekurangan:**

1. **Waktu Tunggu:**
   * Pengguna mungkin mengalami penundaan sebelum mendapatkan feedback, karena fungsi hanya dijalankan setelah jeda waktu tertentu.
2. **Kompleksitas Implementasi:**
   * Memerlukan manajemen timer yang baik dan penyesuaian waktu untuk memastikan fungsi dipanggil pada waktu yang tepat.

**Throttling**

**Kelebihan:**

1. **Kontrol Frekuensi yang Konsisten:**
   * Membatasi seberapa sering fungsi dipanggil pada interval waktu tetap, memberikan performa yang stabil pada event yang sering terjadi seperti scroll.
2. **Menghindari Lag:**
   * Mencegah aplikasi dari lag atau penurunan performa dengan mengontrol pemanggilan fungsi yang berlebihan.
3. **Pengalaman Pengguna yang Lebih Baik:**
   * Memberikan feedback yang lebih konsisten dan responsif selama interaksi yang sering terjadi.

**Kekurangan:**

1. **Fungsi Tidak Selalu Up-to-Date:**
   * Ada jeda waktu antara saat event terjadi dan saat fungsi dieksekusi, yang bisa menyebabkan informasi atau respons yang ditampilkan tidak selalu terbaru.
2. **Kebutuhan Manajemen Interval:**
   * Memerlukan penentuan interval yang optimal untuk fungsi, yang mungkin memerlukan penyesuaian berdasarkan kebutuhan aplikasi untuk menjaga performa dan responsivitas.

**Kesimpulan:**

* **Debouncing** cocok untuk situasi di mana fungsi hanya perlu dipanggil setelah aksi pengguna berhenti, ideal untuk input teks dan pencarian.
* **Throttling** cocok untuk situasi di mana fungsi perlu dibatasi frekuensinya selama event yang sering terjadi, seperti scrolling atau resizing.

Pemilihan metode tergantung pada jenis event dan kebutuhan aplikasi untuk performa dan responsivitas.