**PENGEMBANGAN WEB (TEORI)**

**LAPORAN EKSPERIMEN MENGENAI PROBLEM EFEKTIVITAS METODE OPTIMASI JAVASCRIPT: MINIFICATION VS TREE SHAKING**

*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](#_Toc175910438)

[**A.** **IDENTIFIKASI PROBLEM** 1](#_Toc175910439)

[**B.** **DESKRIPSI PROBLEM** 1](#_Toc175910440)

[**C.** **METODOLOGI EKSPERIMEN** 1](#_Toc175910441)

[**D.** **PELAKSANAAN EKSPERIMEN** 2](#_Toc175910442)

[**E.** **ANALISIS HASIL EKSPERIMEN** 3](#_Toc175910443)

[**F.** **KESIMPULAN** 3](#_Toc175910444)

Link chatGPT : https://chatgpt.com/share/fc194dfa-1cb6-4fc9-b108-512a6e568219

# **IDENTIFIKASI PROBLEM**

Dalam pengembangan aplikasi web modern, performa dan efisiensi kode JavaScript sangat penting untuk pengalaman pengguna yang baik. Dua metode optimasi utama, yaitu **minification** dan **tree shaking**, sering digunakan untuk meningkatkan performa aplikasi dengan mengurangi ukuran file JavaScript dan menghapus kode yang tidak digunakan. Namun, penting untuk mengevaluasi efektivitas kedua metode ini dalam konteks ukuran bundel dan metrik performa halaman web, seperti **First Contentful Paint (FCP)** dan **Total Blocking Time (TBT)**.

# **DESKRIPSI PROBLEM**

1. **Minification** adalah proses menghapus whitespace, komentar, dan memperpendek nama variabel dalam kode JavaScript untuk mengurangi ukuran file.
2. **Tree Shaking** adalah teknik yang menghapus kode yang tidak digunakan dari bundel JavaScript, mengurangi ukuran file dengan menghilangkan bagian kode yang tidak terpakai.

Eksperimen ini bertujuan untuk mengukur perbedaan performa antara bundel JavaScript yang tidak dimodifikasi, yang telah dimodifikasi dengan minification, dan yang telah dimodifikasi dengan tree shaking. Fokus utama adalah untuk memeriksa pengaruh masing-masing metode optimasi terhadap ukuran bundel dan metrik performa seperti **FCP** dan **TBT**.

# **METODOLOGI EKSPERIMEN**

Eksperimen ini dilakukan dengan membandingkan tiga metode optimasi JavaScript pada bundel Webpack: tanpa optimasi, minification, dan tree shaking. Tujuan utama adalah untuk mengevaluasi dampak masing-masing metode terhadap ukuran bundel dan metrik performa halaman web, termasuk **First Contentful Paint (FCP)** dan **Total Blocking Time (TBT)**. Proses eksperimen mencakup:

1. **Setup Proyek:**

* Siapkan proyek Webpack dengan kode JavaScript yang sama untuk semua metode optimasi.
* Gunakan file index.html dengan konten yang kompleks untuk evaluasi performa.

1. **Konfigurasi Webpack:**

* **Tanpa Optimasi:** Konfigurasi Webpack dalam mode development.
* **Dengan Minification:** Konfigurasi Webpack dengan plugin minifier seperti TerserPlugin dan mode production.
* **Dengan Tree Shaking:** Konfigurasi Webpack dengan optimasi usedExports dan mode production.

1. **Pengukuran Ukuran File:**

* Bangun proyek menggunakan masing-masing konfigurasi dan ukur ukuran file bundle.js.

1. **Pengukuran Performa:**

* Gunakan alat **Lighthouse** untuk mengukur **FCP** dan **TBT** dari file index.html yang dimuat di browser.

# **PELAKSANAAN EKSPERIMEN**

1. **Konfigurasi Build:**
   1. **Tanpa Optimasi:**
      1. Jalankan build dengan Webpack dalam mode development.
      2. Ukur ukuran file bundle.js dan catat metrik performa menggunakan Lighthouse.
   2. **Dengan Minification:**
      1. Ubah konfigurasi Webpack untuk menggunakan minification dan mode production.
      2. Jalankan build dan ukur ukuran file bundle.js.
      3. Lakukan pengujian performa menggunakan Lighthouse untuk mengevaluasi FCP dan TBT.
   3. **Dengan Tree Shaking:**
      1. Konfigurasikan Webpack untuk menggunakan tree shaking dengan mode production.
      2. Jalankan build dan ukur ukuran file bundle.js.
      3. Uji performa menggunakan Lighthouse untuk mengukur FCP dan TBT.
2. **Pengumpulan Data:**
   1. Catat ukuran file bundle.js untuk setiap metode optimasi.
   2. Lakukan pengujian performa untuk mendapatkan metrik FCP dan TBT dari file index.html.
3. **Analisis Hasil:**
   1. Bandingkan hasil ukuran file dan metrik performa antara ketiga metode optimasi.

|  |  |  |
| --- | --- | --- |
| No Modification | Minification | Tree Shaking |
|  |  |  |

* 1. Evaluasi apakah pengurangan ukuran file berhubungan dengan peningkatan performa halaman atau adanya trade-offs seperti peningkatan TBT.

|  |
| --- |
| No Modification |
|  |
| Minification |
|  |
| Tree Shaking |
|  |

# **ANALISIS HASIL EKSPERIMEN**

1. **Ukuran File:**

* **Tanpa Optimasi:** Ukuran bundel terbesar.
* **Dengan Minification:** Ukuran bundel lebih kecil dibandingkan tanpa optimasi, tetapi mungkin tidak sebesar tree shaking.
* **Dengan Tree Shaking:** Ukuran bundel terkecil, dengan pengurangan lebih signifikan dibandingkan minification.

1. **Performa:**

* **First Contentful Paint (FCP):** Metode optimasi seperti minification dan tree shaking dapat mempercepat FCP, tetapi hasilnya mungkin serupa jika ukuran bundel tidak terlalu besar.
* **Total Blocking Time (TBT):** TBT mungkin menunjukkan peningkatan pada build dengan minification dan tree shaking dibandingkan tanpa optimasi. Hal ini bisa disebabkan oleh overhead tambahan yang diperkenalkan oleh proses optimasi.

1. **Diskusi:**

* Minification umumnya mengurangi ukuran file tetapi dapat memperkenalkan overhead kecil.
* Tree shaking lebih efektif dalam mengurangi ukuran file dengan menghapus kode yang tidak digunakan tetapi mungkin memperkenalkan overhead tertentu dalam bundel yang lebih besar.

# **KESIMPULAN**

Eksperimen ini menunjukkan bahwa baik minification maupun tree shaking dapat mengurangi ukuran bundel JavaScript dan meningkatkan **First Contentful Paint (FCP)**. Namun, meskipun ukuran bundel lebih kecil dengan kedua metode, **Total Blocking Time (TBT)** menunjukkan bahwa ada peningkatan kecil dalam waktu blocking pada build dengan optimasi, kemungkinan disebabkan oleh overhead tambahan dari proses optimasi.

Untuk hasil yang optimal, disarankan untuk melakukan kombinasi dari kedua metode optimasi dengan perhatian pada trade-off antara ukuran file dan metrik performa seperti TBT. Selalu lakukan pengujian mendalam untuk menilai dampak keseluruhan dari teknik optimasi dalam konteks aplikasi spesifik.

**Minification**

**Kelebihan:**

* **Pengurangan Ukuran File:** Menghapus spasi, komentar, dan memperpendek nama variabel untuk mengurangi ukuran file JavaScript.
* **Peningkatan Kecepatan Load:** Ukuran file yang lebih kecil dapat mengurangi waktu muat halaman dan meningkatkan kecepatan pemrosesan di browser.

**Kekurangan:**

* **Tidak Menghapus Kode Tidak Digunakan:** Minification hanya mengurangi ukuran file tetapi tidak menghilangkan kode yang tidak terpakai.
* **Masih Memproses Semua Kode:** Kode yang tidak terpakai tetap ada dalam bundel, meskipun ukurannya sudah diperkecil.

**Kapan Digunakan:**

* **Selalu digunakan** sebagai bagian dari build untuk produksi untuk mengurangi ukuran file dan meningkatkan performa aplikasi secara umum.

**Tree Shaking**

**Kelebihan:**

* **Penghapusan Kode Tidak Digunakan:** Menghapus kode yang tidak pernah dipanggil atau digunakan, yang dapat mengurangi ukuran bundel secara signifikan.
* **Optimisasi Lebih Lanjut:** Meningkatkan efisiensi bundel dengan hanya menyertakan kode yang relevan.

**Kekurangan:**

* **Tergantung pada Struktur Kode:** Tree shaking hanya efektif jika kode diorganisir dengan baik dan menggunakan sistem modul yang mendukung tree shaking (seperti ES6 modules).
* **Mungkin Memerlukan Konfigurasi:** Konfigurasi Webpack perlu diatur dengan benar untuk mendukung tree shaking.

**Kapan Digunakan:**

* **Selalu digunakan** bersama dengan minification dalam build untuk produksi, terutama pada proyek besar dengan banyak kode yang tidak digunakan.

**Kesimpulan**

* **Minification** dan **tree shaking** adalah dua teknik optimasi yang saling melengkapi.
* **Minification** mengurangi ukuran file JavaScript dengan memproses kode yang sudah ada.
* **Tree shaking** mengurangi ukuran bundel dengan menghapus kode yang tidak digunakan sebelum minification diterapkan.

**Praktik terbaik:** Gunakan **tree shaking** dan **minification** secara bersamaan dalam proses build untuk memaksimalkan efisiensi dan performa aplikasi web Anda. Tree shaking akan mengurangi jumlah kode yang harus diminify, sehingga hasil akhirnya akan lebih optimal.