e-ISSN: 2549-7952
p-ISSN: 2580-3336

Analisa Performa Aplikasi Web Berbasis Manipulasi DOM dan Virtual DOM

Yanuangga Galahartlambang[1], Titik Khotiah [2], Jumain [3]
[1,2, 3]Teknologi Informasi, Fakultas Teknik dan Bisnis,
Institut Teknologi dan Bisnis Ahmad Dahlan Lamongan
E-mail: *[1]*yanuangga.id@gmail.com*, [2]titikaye@gmail.com, [3]jumain.dj@gmail.com

Abstrak – *Aplikasi web yang modern menghabiskan banyak data. Model Objek Dokumen yang disebut DOM, adalah bagian penting dari membuat situs web interaktif. Ini merupakan antarmuka yang memungkinkan bahasa pemrograman untuk memanipulasi konten, struktur, dan gaya situs web. Dengan kata lain DOM merupakan standar cara mendapatkan, mengubah, menambah, atau menghapus elemen HTML halaman web. Merender elemen DOM dalam jumlah besar juga dapat mengakibatkan waktu muat yang lambat, kelambatan halaman, dan masalah kinerja lainnya. Virtual DOM adalah abstraksi dari HTML DOM. Lebih ringan dan terlepas dari detail implementasi khusus browser. Karena DOM itu sendiri sudah merupakan abstraksi, virtual DOM sebenarnya adalah abstraksi dari sebuah abstraksi. Virtual DOM adalah memori dan dapat dimanipulasi berkali-kali sebelum membuat perubahan pada DOM itu sendiri, inilah alasan mengapa bekerja dengan virtual DOM lebih efisien. Untuk melakukan analisis kinerja, sebuah eksperimen dilakukan di mana aplikasi web uji dibuat menggunakan kerangka kerja yang dipilih, membandingkan metrik terkait kinerja menggunakan google chrome dan firefox*.

Kata Kunci *— Javascript, DOM, Virtual DOM, Performa DOM*

1. PENDAHULUAN

Dalam beberapa tahun terakhir, banyak yang telah berubah dalam hal menjelajahi web serta cara menggunakannya. Sebelumnya, halaman web terdiri dari halaman web statis, di mana seluruh halaman dirender ulang ketika ada perubahan. Tapi hari ini hanya bagian halaman yang relevan yang ditampilkan. Aplikasi web semakin besar, dengan antarmuka pengguna yang lebih kompleks dan data yang terus berubah.

Bahasa pemrograman JavaScript sebagian besar digunakan dalam pengembangan aplikasi web. DOM merupakan singkatan dari Document Object Model dan merupakan abstraksi dari teks terstruktur. Untuk pengembang web, teks ini adalah kode HTML, dan DOMnya disebut HTML DOM. Elemen HTML menjadi node di DOM. HTML DOM menyediakan antarmuka (API) untuk melintasi dan memodifikasi node. DOM berisi metode seperti getElementById atau removeChild. Pada umumnya bahasa pemrograman JavaScript digunakan untuk berinteraksi dengan DOM. DOM HTML berbentuk seperti pohon terstruktur, yang dtuliskan dalam bentuk struktur dokumen HTML. Hal ini dilakukan agar pengembang aplikasi website dapat melakukan penetrasi pada struktur dokumen dengan cukup mudah. Namun, mudah tidak berarti cepat di sini. Struktur dokumen DOM sangat besar hari ini. Karena hari ini, pengembang aplikasi web semakin didorong ke arah aplikasi web dinamis (Aplikasi Halaman Tunggal - SPA), dimana perlu banyak memodifikasi pohon DOM. Hal ini membutuhkan waktu pengembangan aplikasi website lebih panjang.

1.1 JavaScript

JavaScript adalah bahasa pemrograman yang memungkinkan pengembang aplikasi berinteraksi dengan fungsionalitas yang disediakan oleh web browser. Lebih khusus lagi, JavaScript adalah bahasa skrip, yang berarti (a) secara tradisional, kode sumber JavaScript diinterpretasikan pada saat runtime dan tidak dikompilasi sebelumnya menjadi kode byte dan (b) secara praktis, tujuan utamanya adalah untuk mengubah perilaku aplikasi lain yang biasanya ditulis dalam bahasa pemrograman yang berbeda, di mana itu ditafsirkan dan dijalankan secara real time. Oleh karena itu, bahasa pemrograman yang banyak digunakan pengembang saat ini untuk membangun dan berinteraksi dengan aplikasi web dikenal sebagai JavaScript[1].

1.2 ECMAScript

Asosiasi Produsen Komputer Eropa, sekarang dikenal sebagai Ecma International [2], mengambil alih tata kelola dan standarisasi JavaScript pada tahun 1996 dan terus mempertahankan spesifikasi bahasa tersebut hingga saat penulisan artikel ini. Spesifikasi JavaScript secara resmi bernama ECMAScript, yang didefinisikan oleh standar ECMA-262 yang diterbitkan oleh Ecma International[3]. Jadi secara paradoks, JavaScript mengarah pada pengembangan standar ECMAScript yang sekarang mengatur pengembangan JavaScript. JavaScript juga berisi

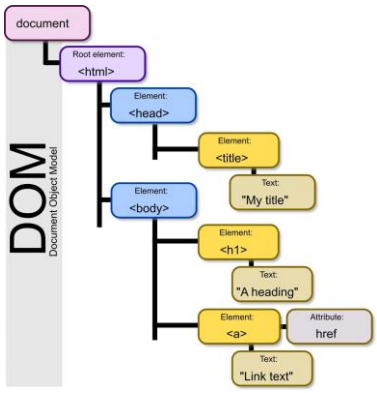
e-ISSN: 2549-7952
p-ISSN: 2580-3336

fungsionalitas untuk mengakses standar teknologi yang tidak diatur oleh Ecma International, seperti HTML5 <canvas>[4] dan Web Graphics Library (WebGL)[5] untuk grafik 2D dan 3D di halaman web.

JavaScript bukan satu-satunya implementasi ECMAScript. Karena banyaknya pengembang yang memprogram dalam JavaScript, ECMAScript telah digunakan sebagai kerangka kerja untuk mengembangkan teknologi berbasis non-browser lainnya. Node.js[6] mengkompilasi ECMAScript untuk sistem server. ActionScript[7] adalah implementasi Adobe dari ECMAScript yang menyediakan fungsionalitas skrip ke platform Adobe Flash Player yang sekarang sudah tidak digunakan lagi. Rhino[8] dan mesin penggantinya Nashorn [9], menyediakan lingkungan skrip ECMAScript berbasis Java dalam aplikasi Java. ECMAScript tentu memiliki berbagai utilitas dan implementasi di berbagai platform, artikel ini berfokus pada implementasi JavaScript dari ECMA-Script karena berlaku untuk HTML5 sisi klien.

1.3 DOM

Document Object Model (DOM) adalah antarmuka pemrograman untuk dokumen HTML dan XML. Ini mewakili halaman sehingga program dapat mengubah struktur, gaya, dan konten dokumen. DOM mewakili dokumen sebagai node dan objek. Dengan begitu, bahasa pemrograman dapat terhubung ke halaman[10].

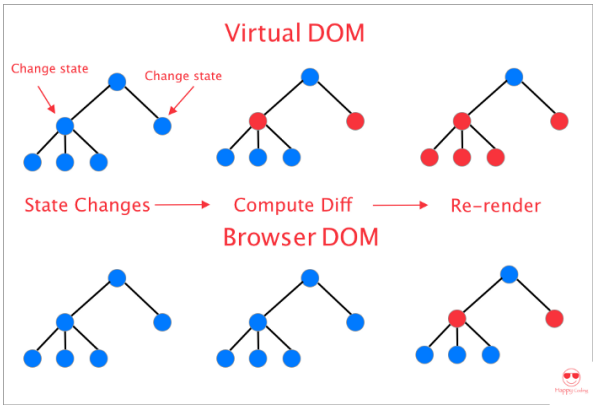


Gambar 1. Struktur Document Object Model

Halaman web adalah dokumen. Dokumen ini dapat ditampilkan di jendela browser atau sebagai sumber HTML. Document Object Model (DOM) mewakili dokumen yang sama sehingga dapat dimanipulasi. DOM adalah representasi berorientasi objek dari halaman web, yang dapat dimodifikasi dengan bahasa skrip seperti JavaScript. DOM mewakili dokumen dengan pohon logis. Setiap cabang pohon berakhir pada sebuah node, dan setiap node berisi sebuah objek. Dengan DOM memungkinkan manipulasi ke dalam struktur dokumen; pengembang aplikasi dapat mengubah struktur, gaya atau isi dari dokumen. Tiap node dapat memiliki event handler yang dapat didefinisikan. Setelah event dipicu, event handler akan dieksekusi.

1.4 Virtual DOM

Pendekatan DOM virtual memungkinkan pengembang menjelaskan bagaimana tampilan aplikasi pada waktu tertentu, dan DOM virtual menangani pembaruan UI untuk menampilkan tampilan yang benar. Saat data dasar dalam aplikasi berubah, DOM virtual memperbarui UI dengan hanya memperbarui bagian yang telah diubah[11]. Konsep DOM virtual menyangkut pembuatan representasi DOM dangkal tanpa membangun representasi DOM yang sebenarnya. Representasi DOM virtual dibuat menggunakan JavaScript dan jauh lebih efisien daripada membuat representasi DOM nyata. Representasi DOM virtual baru dibuat saat perubahan. Dengan menghitung perbedaan antara representasi DOM virtual baru dan sebelumnya, perubahan minimal yang perlu dijalankan pada DOM ditemukan[11].



Gambar 2. Virtual DOM dan Struktur DOM

2. METODE PENELITIAN

Penelitian ini membandingkan javascript vanilla yang akan menampilkan HTML DOM dan library javascript React.js versi 16.8 dan Vue versi 3.0.6 yang akan menampilkan Virtual DOM, library ini dipilih berdasarkan popularitasnya dari github untuk kategori library javascript. Setiap pustaka akan dibagi menjadi 3 kelompok yang ditunjukkan pada tabel 1.

Tabel 1. Komponen grup matrik yang akan diuji kinerjanya.

Grup	Komponen	Keterangan
Duration(1)	Create Rows	Membuat 1000 baris
Duration	Replace All Rows	Update semua 1000 baris

e-ISSN: 2549-7952
p-ISSN: 2580-3336

Duration	Partial Update	Update tiap 10 baris
Duration	Select Row	Tandai baris terpilih
Duration	Swap Rows	Ganti 2 baris pada tabel
Duration	Remove Row	Hapus satu baris
Duration	Create Many Rows	Membuat 10.000 baris
Duration	Append Rows to Large Table	Menambah 1000 baris
Duration	Clear Rows	Hapus 1000 baris
Startup(2)	Consistently Interactive	CPU dan Network dalam kondisi idle
Startup	Script Bootup Time	Total dalam milisecond yang dibutuhkan untuk parse/kompile/evaluasi semua page
Startup	Total Kilobyte Weight	Waktu transfer jaringan dari semua sumber daya yang dimuat ke halaman
Memory(3)	Ready Memory	Memory digunakan setelah halaman dimuat
Memory	Run Memory	Memory digunakan setelah menambah 1000 baris
Memory	Update each 10th row for 1K rows	Memory digunakan setelah memilih update tiap 10 baris sebanyak 5x
Memory	Replace 1000 rows	Memory digunakan setelah membuat 1000 baris 5x
Memory	Creating/Clearing 1000 rows	Memory digunakan setelah membuat dan menghapus 1000 baris 5x

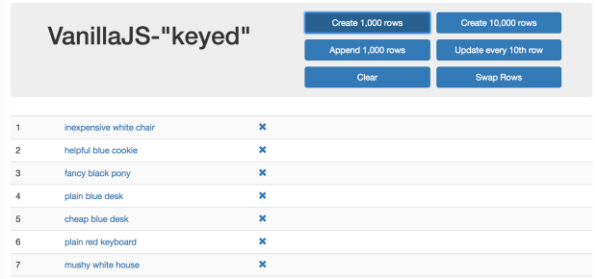
(1) Miliseconds (2) Miliseconds (3) Megabytes

Percobaan dilakukan pada komputer Desktop dengan spesifikasi :

- processor: Intel Core i7-6700K (8 cores, 4.0 GHz),
- memory: 16 GB,
- hard drive: 256 GB SSD M.2,
- graphics card: Intel HD Graphics 530,
- operating system: Windows 10

3. HASIL DAN PEMBAHASAN

Untuk menguji perbandingan kinerja DOM dan Virtual Dom, penulis membuat alat untuk menampilkan antarmuka berupa website untuk melakukan beberapa variabel uji yang telah ditentukan. Antarmuka ini tidak dijalankan secara manual, dijalankan secara otomatis menggunakan perpustakaan dari driver web chrome yang dijalankan menggunakan node js. Hasil tampilan antar muka dapat dilihat pada gambar 3.



Gambar 3. Antarmuka pengujian performa library javascript

Hasil pengujian beberapa variabel pada tabel 1, diterapkan pada library terpilih yaitu vanilla js, react dan vue js. Output dari proses pengujian disimpan dalam file Json (gambar 4). Untuk memudahkan proses penghitungan hasil kinerja nantinya.


```
{
  "framework": "vue-v3.0.6-keyed",
  "keyed": true,
  "benchmark": "08_create1k-after1k_x2",
  "type": "cpu",
  "min": 289.481,
  "max": 336.421,
  "mean": 318.42420000000004,
  "median": 319.4145,
  "geometricMean": 318.15257684539046,
  "standardDeviation": 13.728810426738852,
  "values": [
    289.481,
    308.418,
    311.23,
    316.191,
    319.018,
    319.811,
    321.012,
    327.101,
    335.559,
    336.421
  ]
}
```


Gambar 4. Output JSON proses pengujian setiap variabel

File output JSON dari hasil uji performa di masing-masing library dirangkum dalam bentuk tampilan tabel data. Dari tabel 2 terlihat bahwa kolom pertama adalah variabel yang diuji, kolom kedua, ketiga dan keempat adalah library yang dibandingkan kinerja DOM dan virtual DOM masing-masing. Perbedaan warna hijau, kuning, merah adalah kinerja terbaik ditandai dengan warna hijau, sedangkan kinerja cukup baik ditandai dengan warna kuning, dan kinerja buruk ditandai dengan warna merah.

e-ISSN: 2549-7952
p-ISSN: 2580-3336

Tabel 2. Hasil pengujian kinerja library javascript

Startup metrics (lighthouse with mobile simulation)

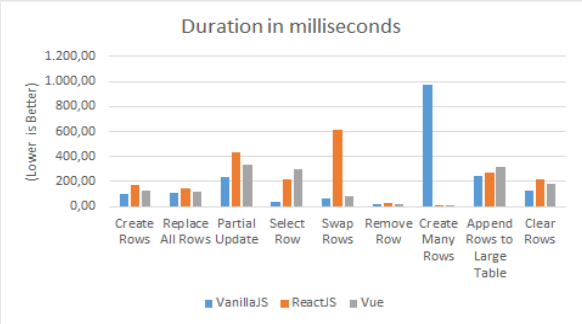
Name	react-v17.0.1	vanillajs	vue-v3.0.6
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	2,580.5 ± 0.9 (1.32)	1,955.6 ± 0.1 (1.00)	2,105.9 ± 0.3 (1.08)
script bootup time the total ms required to parse/compile /evaluate all the page's scripts	16.0 ±0.0 (1.00)	16.0 ±0.0 (1.00)	16.0 ±0.0 (1.00)
total kilobyte weight network transfer cost (post-compression) of all the resources loaded into the page.	272.6 ±0.0 (1.81)	150.3 ±0.0 (1.00)	197.9 ±0.0 (1.32)
geometric mean of all factors in the table	1.34	1.00	1.12

Memory allocation in MBs ± 95% confidence interval

Name	react-v17.0.1	vanillajs	vue-v3.0.6
ready memory Memory usage after page load.	1.3 (1.24)	1.1 (1.00)	1.2 (1.16)
run memory Memory usage after adding 1000 rows.	4.4 (2.76)	1.6 (1.00)	3.5 (2.22)
update eatch 10th row for 1k			

Dari rangkaian pengujian variabel untuk kategori performance, durasi waktu eksekusi, dapat dilihat pada tabel 3. Kolom kedua dari tabel 3 merupakan variabel yang diuji, misalnya pada baris pertama kolom kedua yaitu "create rows" adalah tes untuk membuat 1000 dan 10.000 baris tabel. Kolom ketiga, keempat dan kelima dari tabel 3 adalah lama waktu eksekusi untuk setiap library. Semakin kecil nilainya, perpustakaan memiliki kinerja durasi yang baik dan sebaliknya.

Group	Metric	VanillaJS	ReactJS	Vu
Duration(1)	Create Rows	99,70	168,80	126,
Duration	Replace All Rows	106,20	145,90	119,
Duration	Partial Update	236,50	434,40	332,
Duration	Select Row	41,40	215,40	297,
Duration	Swap Rows	61,40	610,30	79,
Duration	Remove Row	21,60	26,60	23,
Duration	Create Many Rows	974,00	1,66	1,
Duration	Append Rows to Large Table	246,40	275,00	319,
Duration	Clear Rows	123,90	213,30	179,

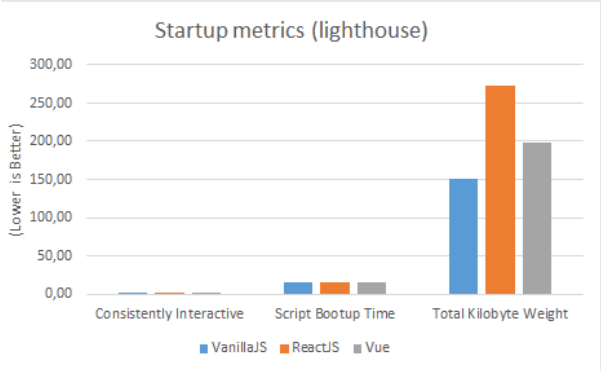


Gambar 5. Performa Durasi (dalam Milidetik)

Hasil Analisis Performa Durasi (dalam Milidetik) dapat dilihat pada gambar 5.

Hasil Analisis Performa StartUp dapat dilihat pada gambar 6

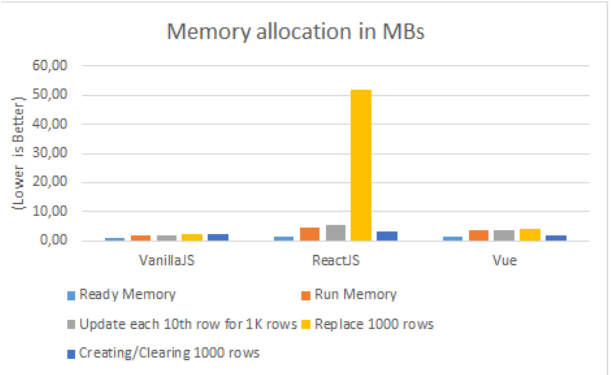
Group	Metric	VanillaJS	ReactJS	Vue
Startup(2)	Consistently Interactive	1,95	2,58	2,11
Startup	Script Bootup Time	16,00	16,00	16,00
Startup	Total Kilobyte Weight	150,30	272,60	197,90



Gambar 6. Performa StartUp (dalam Milidetik)

Hasil Analisis Performa Penggunaan Memori (dalam MegaBytes) dapat dilihat pada gambar 7

Group	Metric	VanillaJS	ReactJS	Vue
Memory(3)	Ready Memory	1,10	1,30	1,20
Memory	Run Memory	1,60	4,40	3,50
Memory	Update each 10th row for 1K rows	1,90	5,20	3,70
Memory	Replace 1000 rows	2,20	52,00	4,10
Memory	Creating/Clearing 1000 rows	2,30	3,00	1,61



Gambar 7. Performa Penggunaan Memori (MegaBytes)

4. SIMPULAN

Dari hasil dan pembahasan dapat disimpulkan sebagai berikut :

1. Terkait dengan metodologi manipulasi DOM, ada perbedaan antara Vanilla JavaScript(bukan library) dan kerangka kerja yang dipilih. Dalam Vanilla JavaScript, manipulasi DOM ditangani dengan interaksi langsung dengan antarmuka DOM, sedangkan saat menggunakan kerangka kerja yang dipilih, interaksi dengan antarmuka DOM ditangani oleh kerangka kerja. Sementara

e-ISSN: 2549-7952
p-ISSN: 2580-3336

React dan Vue.js keduanya telah mengimplementasikan Virtual DOM untuk mengoptimalkan interaksi DOM.

2. Vanilla JavaScript secara signifikan lebih kecil daripada library javascript yang membutuhkan biaya performa yang harus ditanggung karena kompleksnya library tersebut.

5. SARAN

Untuk perbandingan performa lebih lanjut dapat dipilih lebih banyak lagi library javascript serta diujikan pada beberapa perangkat atau device mengingat tantangan dari penggunaan aplikasi web saat ini banyak digunakan di perangkat multi device.

DAFTAR PUSTAKA

[1] Kevin J. Theisen, Programming languages in chemistry: a review of HTML5/JavaScript, Journal of Cheminformatics, pp. 1-19. DOI: https://doi.org/10.1186/s13321-019-0331-1
[2] Welcome to Ecma International. https ://www.ecma-international.org. Accessed 27/02/2021.
[3] ECMAScript 2018 Language Specification. https://262.ecma-international.org/9.0/. Diakses 27/02/2021
[4] HTML Canvas 2D Context. https://www.w3.org/TR/2dcontext/. Diakses 27/02/2021
[5] WebGL Specifications. https://www.khronos.org/registry/webgl/specs/latest/. Diakses 27/02/2021
[6] Node.js. https://nodejs.org/. Accessed 27/02/2021
[7] ActionScript Technology Center. https://www.adobe.com/devnet/actionscript.html. Diakses 27/02/2021
[8] Rhino M | MDN. https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Rhino. Diakses 27/02/2021
[9] Oracle Nashorn: a next-generation javascript engine for the JVM. https://www.oracle.com/technetwork/articles/java/jf14-nashorn-2126515.html. Diakses 27/02/2021
[10] MDN Web Docs. (2021). Introduction to the DOM. https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction. Diakses 24/02/2021
[11] Facebook. Why React?. facebook.github.io/react/docs/why-react.html. Diakses 24/0/2021
[12] Ved Antani, et Al. Object-Oriented JavaScript 3rd edition. Packt Publisher. 2017
[13] Dr. V. Sakhivel, et al. ,Comparative Analysis of Reactjs and Vuejs. in: International Journal of Future Generation Communication and Networking. 2020. pp. 3871-3880.
[14] Sanchit Aggarwal, Modern Web-Development using ReactJS, in: International Journal of Recent Research Aspect, 2018, pp. 133-137
[15] Mattias Levlin, 2020, DOM benchmark comparison of the front-end JavaScript frameworks React, Angular, Vue, and Svelte, Master Thesis in Computer Science. Abo Akademi University.