BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Sebelumnya

Peneliti melakukan studi literatur pada jurnal terkait perbandingan *Framework* dalam penelitian sebelumnya. Berikut adalah penjelasan mengenai studi literatur tersebut.

Penelitian pertama melakukan penelitian komparasi performa *Framework* front-end Javascript menggunakan lighthouse tool. Penelitian ini memiliki tujuan untuk memahami perbedaan performa dari *Framework* React, Angular, Vue, Svelte, dan Solid. Pengujian ini memiliki metode pengujian *Framework* dengan 5 kriteria, yaitu *First Contentful Paint*, *Largest Contentful Paint*, *Time to Interactive*, *Speed Index*, *Total Blocking Time*, dan *Cumulative Layout Shift* menggunakan Lightouse tool. Berdasarkan hasil pengujian menunjukan bahwa Vue memberikan performa terbaik saat mengimplementasikan Weather App. Lainnya tidak secepat Vue, tetapi masih dapat memberikan kinerja yang baik saat menerapkan *Weather App* [7].

Penelitian kedua melakukan penelitian komparasi 3 *Framework* Javascript terpopuler tahun 2019 yaitu Angular, React, dan Vue. Kriteria yang akan dipakai sebagai bahan komparasi adalah popularitas, seberapa sulit dalam belajar *Framework* tersebut, dan hasil pengujian performa *Framework*. Pengujian *Framework* menggunakan metode studi literatur serta pengujian performa *Clientside.* Hasil dari penelitian ini menunjukan bahwa React lebih populer dibanding Angular dan vue, Vue adalah *Framework* yang paling mudah dipelajari untuk para pengembang web muda dan pendatang baru yang ingin belajar *Framework*, dan Vue merupakan *Framework* tercepat dari semua test yang dilakukan di dalam penelitian [5].

Penelitian ketiga melakukan komparasi tolak ukur performa *Framework* React, Angular, Vue, dan Svelte dalam memproses DOM(Document Object Model*).* Tujuan dari penelitian ketiga ini adalah untuk mencari tahu *Framework* yang cocok untuk pengembangan web dan apa saja kekurangan dan kelebihan setiap *Framework*. Pengujian dilakukan dengan pembuatan aplikasi sederhana yang dapat

melakukan penambahan data sebanyak 10000, mengedit salah satu data, edit data 10000, menghapus salah satu data, menghapus 10000 data, dan kompilasi kode *Framework*. Dari penelitian ini menghasilkan sebuah fakta bahwa React memiliki performa yang konsisten sedangkan Angular memiliki performa paling buruk [9].

Penelitian keempat melakukan analisis *Framework* sesuai dengan kebutuhan pengembangan SPA dan MPA. Pemilihan *Framework* berdasarkan dari 3 *Framework* terpopuler yaitu, Angular, Vue, dan React. Analisis dilakukan dengan metode kualitatif deskriptif dimana memilih *Framework* dengan kebutuhan dan kegunaan dari SPA dan MPA. Penelitian ini menghasilkan bahwa Angular adalah *Framework* yang paling cocok digunakan untuk pengembangan SPA berdasarkan dari popularitas dan stabilitas [10].

Penelitian kelima melakukan analisis performa dari aplikasi yang dibangun dari *Framework* Svelte dan Angular. Tujuan utama dari penelitian ini adalah untuk memeriksa apakah bagian klien dari website yang dibangun dari Svelter lebih efektif daripada pendekatan Angular pada umumnya. Penelitian ini menyajikan perbadingan waktu rendering komponen halaman berdasarkan dua aplikasi yang disiapkan dikedua *Framework*. Metode penelitian menggunakan scenario yang disiapkan dimana waktu penambahan dan penghapusan sejumlah komponen halaman yang berbeda diperiksa. Pengujian aplikasi yang dibangun dilakukan menggunakan Selenium Webdriver. Hasil yang didapatkan dari penelitian ini adalah aplikasi yang menggunakan *Framework* Svelte 4 kali lebih cepat dalam hal merender 100, 1000, dan 10000 komponen daripada aplikasi yang menggunakan *Framework* Angular . Sangat penting untuk disadari bahwa browser dan mode yang digunakan sangat mempengaruhi dalam kecepatan proses menampilkan komponen [11].

Tabel 2.1 Penelitian Terdahulu

No	Peneliti	Judul	Metode	Hasil	Perbedaan dengan Penelitian Terdahulu
1	Siahaan, Mangapul dan Vianto, Vincent Octarian	Comparative Analysis Study of Front-End JavaScript *Framework*s Performance Using Lighthouse Tool	Metode yang digunakan adalah dengan membuat aplikasi web menggunakan metode SDLC Waterfall dengan *Framework* React, Angular, Vue, Svelte, dan Solid. Selanjutnya dilakukan testing 6 point kriteria yaitu *First Contentful Paint*, *Largest Contentful Pain*, *Time to Interactive*, *Speed Index*, *Total Blocking Time*, dan *Cumulative Layout Shift*	Berdasarkan hasil analisis dan pengujian yang dilakukan dalam penelitian ini dengan menggunakan Google Lighthouse, Vue memberikan performa terbaik saat mengimplementasikan Weather App. Lainnya tidak secepat Vue, tetapi masih dapat memberikan kinerja yang baik saat menerapkan *Weather App*.	Perbedaan berada di versi *Framework* yang tidak disebutkan pada penelitian ini, sedangkan penelitian yang peneliti lakukan menyebutkan batasan versi *Framework* yang digunakan. Perbedaan selanjutnya terletak diaplikasi web yang dibangun dan alat Lighthouse dengan webpagetest yang digunakan untuk melakukan pengujian aplikasi.

2	Saks, Elar	Javascript *Framework*s: Angular vs React vs Vue	Pengujian dilakukan dengan membandingkan Popularitas, kemudahan dalam mempelajari *Framework*, dan performa dari *Framework* tersebut.	Hasil dari penelitian ini menunjukan bahwa React lebih populer disbanding Angular dan vue, Vue adalah *Framework* yang paling mudah dipelajari untuk para pengembang web muda dan pendatang baru yang ingin belajar *Framework*, dan Vue merupakan *Framework* tercepat dari semua test yang dilakukan di dalam penelitian.	Pada penelitian ini berfokus pada 3 parameter besar yaitu performa, popularitas, kemudahan pembelajaran.
3	Levlin, Mattias	DOM Benchmark comparison of the front-end Javascript *Framework*s React, Angular, Vue, and Svelte	Pengujian dilakukan dengan pembuatan aplikasi sederhana yang dapat melakukan penambahan data sebanyak 10000, mengedit salah satu data, edit data 10000, menghapus salah satu data, menghapus 10000 data, dan kompilasi kode *Framework*	Dari penelitian ini menghasilkan sebuah fakta bahwa React memiliki performa yang baik dan konsisten sedangkan Angular memiliki performa yang paling buruk.	Hanya melakukan pengujian kecepatan proses penampilan DOM sebagai tolak ukur perbandingan antara keempat *Framework*.

4	Sireteanu, NapoleonAlexandru dan Homocianu, Daniel	Front-end *Framework*s For Development of SPA and MPA Web Applications	Dalam proses Komparasi menggunakan metode deskriptif dimana kelebihan dan kekurangan Framework dikaitkan dengan kegunaan dan fungsi dari Single Page Application	React lebih cepat daripada Angular dalam proses penampilan item, namun sangat lambat dalam pengeditan item. React sangat direkomendasikan untuk pembuatan aplikasi yang memproses banyak data.	tidak melakukan pengujian eksperimen sebagai data pendukung dalam penelitian.
5	Bialecki, Gabriel dan Panczyk, Beata	Performance Analysis of Svelte and Angular Applications	Uji efisiensi performa dilakukan berdasarkan 2 aplikasi uji sederhana yang identik secara fungsional yang diterapkan pada Svelte versi 3.0 dan Angular versi 10.2. fungsi yang dibuat untuk melakukan uji efisiensi adalah menampilkan 100, 1000, 10000 data dan menghapus 100, 1000, 10000 data.	Aplikasi yang menggunakan Svelte 4 kali lebih cepat dalam hal merender 100, 1000, dan 10000 komponen daripada Angular.	Hanya melakukan pengujian kecepatan penampilan DOM

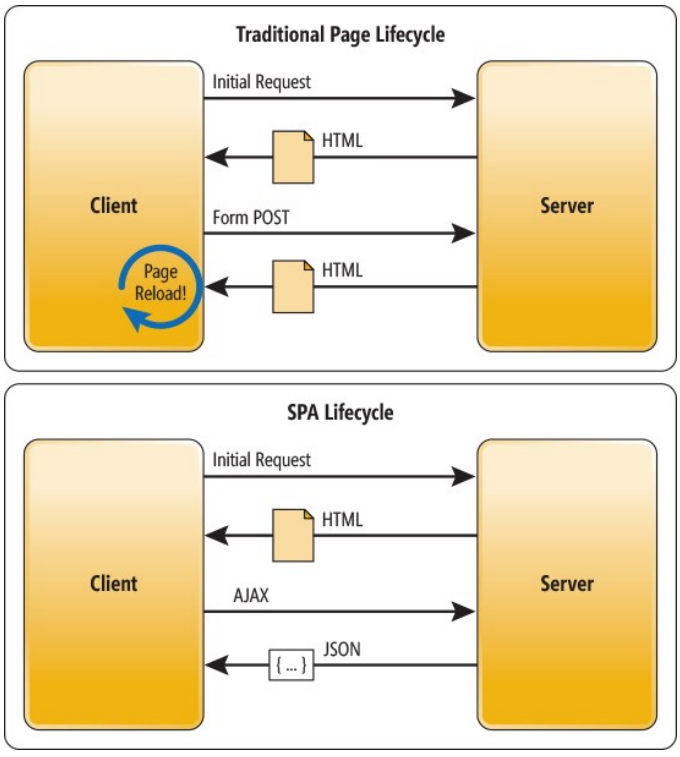
5	Bialecki, Gabriel dan Panczyk, Beata	Performance Analysis of Svelte and Angular Applications	Uji efisiensi performa dilakukan berdasarkan 2 aplikasi uji sederhana yang identik secara fungsional yang diterapkan pada Svelte versi 3.0 dan Angular versi 10.2. fungsi yang dibuat untuk melakukan uji efisiensi adalah menampilkan 100, 1000, 10000 data dan menghapus 100, 1000, 10000 data.	Aplikasi yang menggunakan Svelte 4 kali lebih cepat dalam hal merender 100, 1000, dan 10000 komponen daripada Angular.	Hanya melakukan pengujian kecepatan penampilan DOM

2.2 Dasar Teori

Berikut adalah kajian mengenai beberapa teori yang digunakan pada penelitian ini:

2.2.1 Single Page Application (SPA)

Dalam aplikasi web tradisional, setiap kali aplikasi memanggil server maka server akan menampilkan halaman baru yang menyebabkan refresh halaman di browser. Dalam SPA, interaksi dengan server dilakukan melalui AJAX yang mengembalikan data dalam format JSON dan digunakan untuk memperbarui halaman tanpa *refresh*. gambar 2.1 Menunjukan perbedaan antara kedua pendekatan ini.



Gambar 2. 1 Perbandingan Proses Tradisional dengan Proses SPA

SPA adalah arsitektur aplikasi web yang menampilkan satu halaman HTML dan secara dinamis akan berubah sesuai apa yang dilakukan klien saat melakukan interaksi dengan aplikasi web. SPA memiliki manfaat seperti lebih fleksibel dan responsif tanpa *the jarring effect* dari pemuatan ulang halaman. Selain itu, dengan mengirimkan data aplikasi sebagai JSON, membuat pemisahan antara tampilan

(markup HTML) dan logika dari aplikasi (AJAX dan respon JSON) yang memudahkan dalam desain dan perkembangan setiap komponen. SPA pada umumnya melaukan pemrosesan interaksi tampilan secara langsung di sisi klien melalui Javascript dan CSS, sementara server berperan sebagai komponen layanan yang mengatur permintaan dan pengiriman data. Dengan menggunakan SPA, pemrosesan pada sisi klien dan dikembangkan tanpa harus mengubah semua struktur yang sudah berjalan [12].

2.2.2 Javascript

JavaScript dikenal sejak tahun 1995 sebagai cara untuk menambahkan program pada halaman web pada browser Netscape Navigator. Saat ini, bahasa ini digunakan oleh semua browser web grafis utama. Bahasa ini memungkinkan aplikasi web modern tanpa perlu melakukan reload halaman untuk setiap aksi. Namun, perlu dicatat bahwa JavaScript tidak memiliki hubungan dengan bahasa pemrograman Java, nama yang sama hanya diambil untuk memanfaatkan popularitas bahasa Java saat itu. Setelah diterima oleh software lain, bahasa ini dibuat standar oleh organisasi Ecma International dan dikenal dengan nama ECMAScript dan JavaScript [13].

2.2.3 Typescript

Menurut sumber resmi dari TypeScript, bahasa ini menyediakan semua fitur dari JavaScript dan juga menambahkan sistem untuk mendefinisikan tipe data tertentu untuk variabel atau objek. Dalam *Framework* Angular, kelas layanan digunakan untuk berbagi data antar komponen yang tidak terkait tetapi memiliki tampilan tertentu. Kelas layanan ini diberi tahu oleh dekorator @Injectable() yang menyediakan metadata untuk cara penggunaannya. Misalnya, kelas layanan dapat digunakan untuk mengambil data dari server atau berbagi data antar komponen.

Keuntungan utama dari penggunaan TypeScript adalah kemampuannya untuk menekankan perilaku yang tidak terduga dalam kode dan mengurangi kemungkinan terjadinya bug. File TypeScript menggunakan akhiran .ts daripada .js yang digunakan untuk file JavaScript. Meskipun TypeScript adalah versi yang diperluas dari JavaScript, kode dalam file .js sudah bisa dianggap sebagai kode

TypeScript. Ini mempermudah proses migrasi kode dari JavaScript ke TypeScript tanpa perlu menulis ulang kode secara keseluruhan.

Selain itu, TypeScript diintegrasikan dengan router di Angular untuk menentukan jalur navigasi antara negara bagian yang berbeda. Ketika kode TypeScript dikompilasi, tsc (TypeScript compiler) digunakan untuk mengonversi kode tersebut ke JavaScript agar bisa dijalankan oleh browser. Dalam proyek Angular, konfigurasi untuk TypeScript disimpan dalam file tsconfig.json yang berisi opsi dasar untuk kompilasi. Pengaturan seperti noImplictAny digunakan untuk mengontrol perilaku kompiler dalam menentukan tipe data variabel. TypeScript juga dapat digunakan dalam *Framework* lain seperti React dan Vue [14].

2.2.4 Node.js

Node muncul ketika pengembang asli menggunakan Javascript, sesuatu yang biasanya hanya dapat developer jalankan di dalam browser, dan Node membiarkannya berjalan di mesin sebagai proses yang berdiri sendiri. Ini berarti pengembang dapat membuat aplikasi menggunakan Javascript di luar konteks browser.

Sekarang, JavaScript sebelumnya memiliki rangkaian fitur terbatas. Ketika menggunakannya di browser, pengembang dapat melakukan hal-hal seperti memperbarui URL dan menghapus logo Node, menambahkan acara klik atau apa pun, tetapi pengembang tidak dapat melakukan lebih banyak lagi. Dengan Node, pengembang sekarang memiliki kumpulan fitur yang terlihat jauh lebih mirip dengan bahasa lain, seperti Java, Python, atau PHP.

Sekarang, Node dan JavaScript yang dijalankan di dalam browser pengembang, keduanya berjalan di mesin yang sama persis. Ini disebut mesin runtime JavaScript V8. Ini adalah mesin open source yang mengambil kode JavaScript dan mengkompilasinya menjadi kode mesin yang jauh lebih cepat. Dan itulah bagian besar yang membuat Node.js begitu cepat [15].

2.2.5 React

Framework React.js, yang asalnya berasal dari sebuah proyek internal Facebook dan dibuka sumbernya pada Mei 2013, adalah sebuah perpustakaan JavaScript yang digunakan untuk membangun antarmuka pengguna. Ini dapat menggabungkan beberapa fragmen kode yang independen menjadi sebuah antarmuka UI yang kompleks. Memiliki performa tinggi, logika kode yang sederhana, dan dapat memecahkan masalah kompatibilitas cross-browser dengan mudah. React.js memungkinkan antarmuka pengguna untuk langsung digabungkan dengan komponen seperti tombol dan dialog. Dengan menggabungkan komponen-komponen tersebut, para pengembang dapat memiliki halaman web interaktif yang kaya. Pada saat yang sama, penggunaan sintaks JSX yang relevan pada antarmuka pengguna membuat proses pereusan komponen menjadi lebih mudah dan memastikan struktur internal komponen jelas. Di atas komponen-komponen tersebut, React Js juga dapat membedakan kode dari target yang sebenarnya dengan cepat dan memanfaatkan kemampuan rendering DOM di browser untuk mengembangkan halaman web, sehingga memfasilitasi pengembangan aplikasi mobile native [16].

2.2.6 NestJs

NestJS atau Nest adalah sebuah kerangka kerja untuk membuat aplikasi server di platform Node.js. Kerangka kerja ini ditulis menggunakan TypeScript dan bahasa yang direkomendasikan untuk pengembangan aplikasi dengan Nest adalah TypeScript, namun juga mendukung pengembangan dengan JavaScript. NestJS sebenarnya merupakan lapisan abstraksi di sekitar kerangka kerja yang sudah ada untuk mengembangkan aplikasi server di Node.js. Express.js merupakan kerangka kerja yang direkomendasikan dan diasumsikan sebagai latar belakang Nest, tetapi Nest juga dapat dikonfigurasi untuk berjalan di kerangka kerja lain. Pengembangan aplikasi dengan Nest tidak tergantung pada kerangka kerja di latar belakang karena antarmuka yang disediakan oleh Nest bersifat seragam.

Nest adalah sebuah kerangka kerja yang memiliki pendekatan yang konsisten, yang berarti para pencipta NestJS menetapkan cara-cara tertentu untuk

melakukan sesuatu. Hal ini sebenarnya positif, karena pada akhirnya semua aplikasi yang dibangun dengan Nest akan memiliki struktur yang serupa, dan bagi pengembang yang sudah berpengalaman dengan Nest, akan lebih mudah untuk beradaptasi jika terlibat dalam pengembangan aplikasi lain. Selain itu, dengan pertimbangan bahwa Nest dikembangkan menggunakan TypeScript, aplikasi yang dibuat dengan kerangka kerja ini menjadi modular, mudah untuk dipelihara, dan dapat diskalakan [17].

2.2.7 Svelte

Svelte adalah *Framework* komponen seperti React atau Vue yang memiliki perbedaan penting. Berbeda dari kerangka kerja tradisional yang membutuhkan browser untuk mengubah kode deklaratif menjadi operasi DOM, Svelte memproses kode saat build menjadi kode yang efisien dan memperbaharui DOM secara tepat. Versi pertama Svelte fokus pada pengujian hipotesis, versi kedua adalah perbaikan kecil, dan versi 3 adalah perombakan besar dengan fokus memberikan pengalaman pengembang yang luar biasa.

Anda bisa menulis komponen dengan boilerplate yang lebih sedikit dibandingkan dengan kerangka kerja lain [18].

2.2.8 Webpagetest Tools

Webpage test adalah salah satu alat populer dan gratis untuk mengukur kinerja halaman web. Alat ini mendukung uji kinerja web pada situs pengguna dari berbagai lokasi di seluruh dunia menggunakan berbagai browser.

Keuntungan utama dari WebPageTest adalah tes bisa dilakukan dari lokasi di seluruh dunia, menggunakan browser yang sebenarnya dengan kecepatan koneksi yang sebenarnya. Ini memungkinkan untuk menguji kinerja di mana pengguna sebenarnya berada, dan melihat waktu muat yang sebenarnya daripada skor acak dari 0 hingga 100. Dengan alat ini bisa melakukan berbagai tes yang berbeda, termasuk melakukan tes sederhana, tes multi-langkah lanjutan, merekam video, memblokir konten, membandingkan visual situs multi, dan menguji *traceroute* [19]. Penggunaan Webpagetest secara umum dapat diakses melalui https://www.webpagetest.org/.

2.2.9 Matriks Performansi

Matriks Performansi adalah sebuah matriks yang berisi beberapa variabel yang akan digunakan sebagai acuan komparasi. Variabel tersebut yaitu:

2.2.9.1 First Contentful Paint(FCP)

FCP mengukur berapa lama waktu yang dibutuhkan untuk menampilkan konten pertama pada halaman. Konten yang dipertimbangkan dalam pengukuran FCP adalah gambar, elemen kanvas yang tidak putih, dan SVG [20]. Berikut adalah katergori FCP pada tabel 2.1:

Tabel 2. 1 Kategori FCP

Waktu FCP (detik)	Kategori
0-1.8	Cepat (baik)
1.8-3	Sedang (sedang)
>3	Lambat (buruk)

2.2.9.2 Largest Contentful Paint(LCP)

LCP adalah konten terbesar yang berhubungan dengan warna berkaitan dengan pembacaan (pemuatan) dan waktu pengukuran ketika konten yang paling efektif halaman kemungkinan ditampilkan. Kepuasan rentang dimana konten halaman terbesar harus ditampilkan adalah dari 0 hingga 2,5 detik dari awal mulai memuat halaman [21]. Berikut adalah katergori LCP pada tabel 2.2:

Tabel 2. 2 Kategori LCP

Waktu LCP (detik)	Kategori
0-2.5	Cepat (baik)
2.5-4.0	Sedang (sedang)

>4.0	Lambat (buruk)

Untuk memberikan pengalaman yang baik kepada pengguna, halaman harus memiliki LCP dalam waktu 2.5 detik atau kurang. Untuk mencapai tujuan ini dan memberikan pengalaman yang baik bagi sebagian besar pengguna, standar yang baik untuk diukur adalah waktu pemuatan halaman pada persentil ke-75.

2.2.9.3 Speed Index(SI)

SI menunjukkan waktu rata-rata bagi konten halaman untuk terisi secara terlihat. Semakin rendah SI, semakin baik kinerjanya. SI dirumuskan dengan persamaan berikut [22]:

$$\int_1^{end} 1 - \frac{VC}{100} \quad (2.1)$$

$$end = selesai\ render(dalam\ milisekon)$$

$$VC = Presentase\ visual\ yang\ selesai\ dirender$$

Berikut adalah Kategori SI pada tabel 2.3:

Tabel 2. 3 Kategori SI

Waktu SI (dalam detik)	Kategori
0-3.4	Cepat (baik)
3.4-5.8	Sedang (sedang)
>5.8	Lambat (buruk)

2.2.9.4 Total Blocking Time(TBT)

Total Blocking Time (TBT) mengukur berapa lama waktu yang terblokir selama pemuatan halaman. Jumlah dihitung dengan menambahkan bagian pemblokiran dari semua *long task* antara FCP dan TTI. Setiap tugas yang dieksekusi

lebih dari 50 ms adalah *long task*. Jumlah waktu setelah 50 ms adalah bagian pemblokiran [21]–[23]. Berikut adalah Kategori TBT pada tabel 2.4:

Tabel 2. 4 Kategori TBT

Waktu TBT (dalam milisekon)	Kategori
0-200	Cepat (baik)
200-600	Sedang (sedang)
>600	Lambat (buruk)

2.2.9.5 Cumulative Layout Shift(CLS)

CLS adalah ukuran ledakan terbesar dari pergeseran tata letak tak terduga yang terjadi saat halaman berlangsung. CLS diukur dari skor pergeseran tata letak setiap pergeseran tata letak tak terduga selama masa hidup halaman. Pergeseran tata letak terjadi saat elemen yang terlihat berubah posisi dari satu frame ke frame berikutnya. Ledakan pergeseran tata letak disebut jendela sesi, saat satu atau lebih pergeseran tata letak berlangsung berturut-turut dalam waktu kurang dari 1 detik dan maksimal 5 detik selama jendela sesi. Ledakan terbesar adalah jendela sesi dengan skor kumulatif maksimal dari semua pergeseran tata letak di dalam jendela tersebut [24], [25]

Berikut kategori CLS pada tabel 2.5:

tabel 2. 5 Kategori CLS

Waktu CLS (dalam milisekon)	Kategori
0-100	Cepat (baik)
100-250	Sedang (sedang)
>250	Lambat (buruk)

2.2.9.6 Page Weight

Page weight adalah ukuran *byte* dari sebuah halaman web. Sebuah halaman web jarang sekali hanya terdiri dari HTML dari URL yang dilihat di bilah alamat browser. Sebaliknya, sebuah halaman web seperti yang dilihat dan direnderi di browser menggunakan elemen-elemen dan aset tertentu. *Page weight* mencakup semua aset yang digunakan untuk membuat halaman[26], yaitu:

a. HTML yang menyusun kerangka halaman
b. Gambar dan media lainnya(video, audio, dan lain lain) yang disematkan dalam halaman
c. CSS atau *styles* yang digunakan untuk merancang tampilan halaman
d. Javascript untuk memberikan interaktivitas
e. Sumber daya pihak ketiga yang berisi satu atau lebih dari yang disebutkan diatas.

Page weight adalah faktor penting dalam mengukur performa halaman web, dikarenakan semakin besar *page weight*, semakin lama waktu muat halaman dan semakin tinggi konsumsi *bandwidth* .