Rejuvenasi Bentuk Fungsi dalam JavaScript Menggunakan Regular Expression

Gibran Darmawan
Prodi Teknik Informatika
Sekolah Tinggi Informatika dan Elektro
Bandung, Indonesia
13520061@std.stei.itb.ac.id

Abstract*—Pemrograman komputer adalah proses melakukan perhitungan tertentu (atau lebih umum, mencapai hasil komputasi tertentu), biasanya dengan merancang dan membangun program komputer yang dapat dieksekusi. Pemrograman melibatkan tugas-tugas seperti analisis, menghasilkan algoritma, akurasi algoritma profil dan konsumsi sumber daya, dan implementasi algoritma (biasanya dalam bahasa pemrograman yang dipilih, biasa disebut sebagai coding). Salah satu bahasa pemrograman komputer adalah JavaScript. Javascript mengalami pembaharuan selama masa hidupnya baik dalam penambahan syntax dan fitur baru maupun optimisasi kode ketika dijalankan. Pada update ES6, developer JavaScript menambahkan bentuk fungsi baru, yaitu "const" dan "let". Regular Expression merupakan strategi algoritma yang dipelajari oleh mahasiswa mata kuliah IF2211 Strategi Algoritma. Mengganti sebuah kumpulan fungsi dalam suatu kode secara satu persatu akan memakan waktu yang lama. Paper ini akan memperbaharui bentuk kode yang masih menggunakan fungsi *oldschool* JavaScript menjadi bentuk baru yang disebut sebelumnya dengan menerapkan Regular Expression secara otomatis agar tidak memakan waktu yang lama.

Keywords—Regular Expression, regex, JavaScript, pembaharuan, fungsi

I. PENDAHULUAN

Pemrograman adalah proses menulis, menguji dan memperbaiki, dan memelihara suatu kode yang membangun suatu program komputer. Kode ini dapat ditulis berbagai macam bahasa pemrograman. Tujuan dari pemrograman adalah untuk memuat suatu program yang dapat melakukan suatu perhitungan atau 'pekerjaan' sesuai dengan keinginsan si pemrogram. Untuk melakukan pemrograman, diperlukan ketermapilan dalam algoritma, logika, bahasa pemrograman, dan pada beberapa kasus, pengetahuan dari ilmu matematika. Salah satu bahasa pemrograman adalah JavaScript.

JavaScript selama masa hidupnya mengalami beberapa perubahan, dengan versi paling awal di-*release* pada tahun 1995. Versi paling awal ini hanya digunakan oleh karyawan Netscape hingga pada tahun 1997 Netscape akhirnya menyerahkan JavaScript kepada ECMA International agar semua pembuat browser bisa menggunakan JavaScript. Versi yang diserahkan kepada EcmaInternational disebut dengan ECMAScript 1. Kemudian pada tahun 1998 lahir ECMAScript 2, pada tahun 1999 ECMAScript 3 dan pembuatan ECMAScript 4 dimulai pada tahun 2000 tetapi tidak pernah di-*release*. ECMAScript 5 (ES5) dirilis pada tahun 2009 dan memiliki beberapa update besar, seperti integrasi JavaScript V8 yang dikembangakan Google, Node.js oleh Ryan Dahl. Pada tahun 2015, ECMAScript 2015 (ES6) dirilis oleh Mozilla. Dalam update ini, terdapat penambahan bentuk fungsi baru yaitu "cosnt" dan "let". "const" merupakan bentuk variable yang hanya bisa dipanggil dalam block tempat variable tersebut dibuat. Beberapa kode lama mungkin bisa saja memanfaatkan bentuk variable baru ini. Namun, jika ada banyak bentuk variable yang inign diubah dan seorang developer mencoba untuk mengubahnya satu-satu, akan memakan waktu yang lama.

Waktu merupakan factor yang penting bagi sebuah developer karena tidak ada waktu yang tidak terbatas bagi sebuah developer. Entah developer tersebut harus pindah ke project lain atau memang tidak ada waktu saja. Dengan itu, dalam makalah ini akan membahas implementasi Regular Expression untuk mengubah suatu variable menjadi bentuk *const*.

II. TEORI DASAR

A. *Penentuan Karakter*

Fungsi "const" relative baru dalam bahasa pemrograman, hingga beberapa kode yang dibuat sebelum fungsi ini dirilis tentu saja belum mengimplementasikannya. Mungkin beberapa dari kode lama tersbut bisa memanfaatakan dari kelebihan bentuk variable baru tersebut. Salah satu contoh kode yang bisa memanfaatkannya adalah:

```
1
2  function haloUser(first_name, last_name){
3      console.log("Hello there ", first_name, last_name)
4  }
```

Gambar 2.1 Contoh Kode

Fungsi tersebut dapat memanfaakan bentuk *const* karena fungsi tersebut tidak akan berubah selama kode bekerja. Jika sebuah developer ingin merubah kode tersebut menjadi const, iya harus mengganti "function" menjadi "const" dan menambahkan suatu panah ('=>') sebelum kurung kurawal pembuka kode. Mungkin jika hanya satu fungsi yang diganti tidak akan memakan waktu yang lama, bagaimana jika terdapat 100, 200, 300 dst. kode yang perlu diubah? Untuk membuat sebuah kode yang bisa melakukan apa yang developer butuhkan untuk merubah bentuk tersebut, sang kode harus bisa mendeteksi karater mana saja yang harus diubah dan mengidentifikasi bentuk dari kode tersebut.

Dengan itu, ada salah satu strategi algoritma yang dapat dimplementasikan untuk menjalankan pekerjaan diatas, yaitu penerapan Regular Expression.

B. *JavaScript*

JavaScript (JS) adalah bahasa pemrograman terkompilasi yang ringan, ditafsirkan, atau tepat waktu dengan fungsi kelas satu. Meskipun paling terkenal sebagai bahasa scripting untuk halaman Web, banyak lingkungan non-browser juga menggunakannya, seperti Node.js, Apache CouchDB dan

Adobe Acrobat. JavaScript adalah bahasa berbasis prototipe, multi-paradigma, utas tunggal, dinamis, mendukung gaya berorientasi objek, imperatif, dan deklaratif (misalnya pemrograman fungsional).

Gambar 2.2 Contoh kode JavaScript

JavaScript ditemukan oleh Brendan Eich pada tahun 1995. Bahasa ini dikembangkan untuk Netscape 2, dan menjadi standar ECMA-262 pada tahun 1997. JavaScript dibuat untuk menghilangkan limitasi bahwa web pages tidak bisa keluar dari bentuk statis. Setelah Netscape menyerahkan JavaScript ke ECMA, yayasan Mozilla terus mengembangkan JavaScript untuk browser Firefox. Selama tahun 2000-an, browser Internet Explorer sangatlah popular dikalangan pengguna internet. Namun, dalam browser itu scripting dalam bagian user mulai tidak menghasilkan perkembangan. Mozilla, penerus dari Netscape, mengeluarkan broser terbarunya Firefox serta implementasi JavaScript dalam browser itu. Firefox secara instan menjadi browser terpopuler. Beberapa tahun kemudian pada tahun 2008, Google mengeluarkan browser terbarunya Chrome yang menggunakan mesin JavaScript. Chrome membelah pasar browser yang didominasi oleh Firefox. Ketika dua browser yang terbuat dari mesin JavaScript mendominasi pasar pengguna internet, javascript pun mendominasi bahasa yang digunakaan untuk membuat website yang dibuka oleh browser tersebut.

Pekerjaan ambisius pada bahasa ini telah berlanjut selama beberapa tahun dan memuncak dalam koleksi ekstensif penambahan dan penyempurnaan pada ECMAScript 6 edisi 2015

Pembuatan Node.js pada tahun 2009 oleh Ryan Dahl menyebabkan peningkatan yang signifikan dalam penggunaan JavaScript di luar browser web. Node menggabungkan mesin V8, loop peristiwa dan I/O API untuk menyediakan sistem eksekusi JavaScript mandiri. Pada 2018, Node digunakan oleh jutaan pengembang dan npm memiliki unit terbanyak di dunia dari manajer paket.

Draf skrip ECMAScript saat ini disimpan secara publik di GitHub dan diterbitkan dalam snapshot tahunan reguler. [36] Kemungkinan perubahan bahasa akan diperiksa dalam proses proposal yang komprehensif. [37] [38] Pengembang sekarang memeriksa status properti yang ditampilkan satu per satu, bukan nomor masalah. [36]

Ekosistem JavaScript modern memiliki banyak pustaka dan kerangka kerja, metode pemrograman terbaik, dan menggunakan JavaScript secara ekstensif di luar browser. Selain itu, dengan munculnya aplikasi satu halaman dan halaman web intensif JavaScript lainnya, sejumlah besar responden telah dibuat untuk mendukung proses pengembangan.

Gambar 2.3 Website yang Dibuat dengan Javascript

Lebih dari 80% situs web menggunakan perpustakaan JavaScript pihak ketiga atau kerangka kerja web untuk skrip sisi klien mereka. jQuery sejauh ini merupakan perpustakaan paling populer, digunakan oleh lebih dari 75% situs web. Facebook membuat perpustakaan React untuk situs webnya dan kemudian merilisnya sebagai open source; situs lain, termasuk Twitter, sekarang menggunakannya. Demikian pula, kerangka kerja Angular yang dibuat oleh Google untuk situs webnya, termasuk YouTube dan Gmail, kini menjadi proyek sumber terbuka yang digunakan oleh orang lain Sebaliknya, istilah "Vanilla JS" telah diciptakan untuk situs web yang tidak menggunakan pustaka atau kerangka kerja apa pun, melainkan sepenuhnya mengandalkan fungsionalitas JavaScript standar

C. Regular Expression dalam JavaScript

Regular Expression atau biasa disingkat menjadi regex adalah sebuah *pattern* atau pola kombinasi kararter yang ditemukan dalam string. Konsep regular expression dimulai pada 1950-an, ketika matematikawan Amerika Stephen Cole Kleene meresmikan konsep bahasa reguler. Mereka mulai umum digunakan dengan utilitas pemrosesan teks Unix. Sintaks yang berbeda untuk menulis regular expression telah ada sejak tahun 1980-an, satu menjadi standar POSIX dan lainnya, banyak digunakan, menjadi sintaks Perl.

Regular Expression pantas untuk digunakan jika seseorang ingin memanipulasi sebuah kumpulan kata atau *string*, mencocokan suatu string, validasi sebuah string, dan berbagai contoh lainnya.

Regular Expression digunakan di mesin pencari, dalam mencari dan mengganti dialog pengolah kata dan editor teks, dalam utilitas pemrosesan teks seperti sed dan AWK, dan dalam analisis leksikal. Sebagian besar bahasa pemrograman tujuan umum mendukung kemampuan regex baik secara asli atau melalui perpustakaan, termasuk misalnya Python, C,C++, Java, dan JavaScript.

Dalam JavaScript, Regular Expression dibagi menjadi dua tipe, yaitu:

1. RegExp Object. RegExp Object merupakan objek global yang tersedia tanpa harus menambah atau memerlukan yang lain.

```
6
7   let regExp = new RegExp('a|b');
```

Gambar 2.4 Contoh RegExp Object

2. Literal Notation. *Literal notation* mendefinisikan sebuah regex dengan notasi yang dikelilingi oleh '/'

```
9   let regExp = /a|b/;
```

Gambar 2.5 Contoh Regex Literal Notation

Kedua bentuk tersebut menghasilkan output yang sama. Dalam paper ini, akan digunakan bentuk Literal Notation. Dalam regex, ada aturan yang harus dipenuhi. Beberapa aturan yang terpenting tersebut adalah:

- Character Class

Pattern	Arti
\d	Menyeseuaikan dsatu digit atau karakter dari 0 hingga 9
\s	Menyesuaikan dengan symbol *whitespace '\n'*
\w	Menyesuaikan dengan karakter ASCII [A-Za-z0-0_]
\D	Menyesuaikan semua karakter kecuali sebuah angka.
\W	Menyesuaikan semua karakter kecuali ASCII [A-Za-z0-0_]
\S	Menyesuaikan semua karakter kecuali *whitespace*

Tabel 2.1 Character Class

- Modifier atau Flags

Pattern	Arti
g	Melakukan pencarian secara global.
i	Melakukan pencarian yang sensitif terhadap huruf kapital.
M	Pencarian multi-*line.*
S	Membiarkan '.' Untuk menyesuaikan karakter newline
u	"Unicode"; memperlakukan sebuah pattern sebagai sebuah rangkaian Unicode.
Y	Melanjutkan pencarian dari posisi terakhir sebuah pencarian dilakukan.

Tabel 2.2 Modifier atau Flags

- Quantifiers

Pattern	Arti
.*	Jumlah kejadian lebih besar sama dengan 0
+	Jumlah kejadian lebih besar sama dengan 1
?	Jumlah kejadian 9 kali atau 1 kali
{n}	Jumlah kejadian sebanyak *n* kali
{n,}	Jumlah kejadian paling sedikit *n* kali
{n.m}	Jumlah kejadian beriris dari *n* kali hingga *m* kali.
*?	Jumlah kejadian 0 kali atau 1 kali sesedikit mungkin
+?	Jumlah kejadian lebih besar sama dengan 1 kali sesedikit mungkin

Tabel 2.3 Tabel Quantifiers.

- Grouping

Karakter	Arti
x|y	Mencocokkan *x* atau *y.*
[xyz]	Mencocokkan salah satu antara x, y, atau z.
[A-Z]	Mencocokkan salah satu antara A-Z.
[^A-N]	Mencocokkan karakter apapun kecuali A-N.
[^xyz]	Mencocokkan karakter apapun kecuali antara x, y, atau z.
(xyz)	Mencocokkan xyz keselurhan atau tidak sama sekali kemudian mengingatnya
(?:xyz)	Mencocokkan xyz keseluruhan atau tidak sama sekali tapi tidak mengingatnya

Tabel 2.4 Tabel Grouping

D. Capturing Group

Capturing group merupakan sebuah proses yang mengambil beberapa karakter sebagai satu kelompok (group). Pengimplementasiannya adalah sebuah karakter yang sudah dikelompokkan akan dimasukkan ke dalam sebuah tanda kurung. Misalnya seperti sebuah string yang akan di capture adalah:

http://websitedummy.com/mainpage.html

Seorang developer meminta bagian awal dot com (Host Name) dan dipisahkan dengan path Host Name tersebut. Capture group yang dibuat untuk Hostname adalah

([A-z]\.]*)

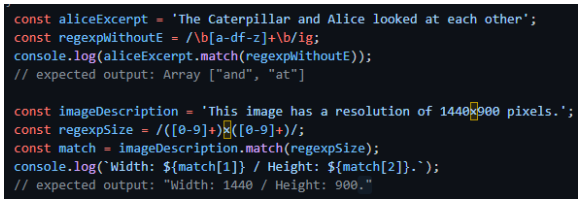
Untuk menangkap path dari Hostname , capture groupnya adalah:

(.*)

Gabungkan kedua capture group tersebut untuk membentuk *capture group* akhir:

([A-z]\.]*)/(.*)

Contoh lain dari bagaimana capturing bekerja adalah:



Gambar 2.6 Contoh Capturing Group

Dalam conoth pertama pada Gambar 2.6, *capturing group* hanya menangkap sebuah kata yang tidak mengandung huruf ‘e’ yaitu pada kata ‘and’ dan ‘at’. *Capturing group* yang di-*state* oleh kode adalah

[a-df-z]

Dari statement tesrsebut, diminta bahwa semua kata yang mengandung huruf dari a—d dan f—z akan masuk ke dalam capturing group.

Pada contoh kedua, capturing group hanya mengangkap angka resolusi dari string imageDescription. Regex untuk capturing group tersebut adalah

[0-9]

Dengan regex tersebut, capturing group hanya akan mengangkap karakter yang merupakan angka 0—9 saja.

III. Pembahasan

Proses awal pengimplementasian adalah menentukan file kode yang developer ingin ganti. Dalam kasus ini, fungsi pada file JavaScript mengandung fungsi yang ingin diganti semua menjadi bentuk *const.*

Pastikan bahwa semua package dependencies JavaScript sudah terinstall semua.

Pertama, program akan meng-*capture* nama fungsi tersebut, parameter-parameternya dan isi dari fungsi tersebut dan baru memulai restrukturasi kode tersebut. Berdasarkan kebutuhan pada kalimat sebelumnya, kita membutuhkan tiga *capturing group*. *Capturing Group* ini berbentuk

(

Berikut merupakan capturing groupnya

```
/*
    function (.+)(\(.+\))(\{.+\})
    (.+) - nama fungsi
    (\(.+\)) - parameter
    (\{.+\}) - isi fungsi
*/
const regExp = /function (.+)(\(.+\))(\{.+\})/gms
```

Gambar 3.1 Implementasi Capturing Group

Untuk membaca kode file, gunakan fungsi ReadFile bawaan dari JavaScript.

```
5   fs.readFile("./test.js", (err, cnt) => {
6
7   })
```

Gambar 3.2 Fungsi ReadFile

Selanjutnya, implementasikan Regular Expression yang mengubah fungsi yang sudah dicapture menjadi bentuk fungsi *const arrow*

```
10    replace(regExp, "const $1 = $2 => $3")
```

Gambar 3.3 Fungsi Regex

Realisasikan kode regex yang sudah dibuat dengan menggabungkan semuanya.

```
const fs = require("fs")

const regExp = /function (.+)(\(.+\))(\{.+\})/gms

fs.readFile("./test2.js", (err, cnt) => {
    console.log(cnt.toString().replace(regExp, "const $1 = $2 => $3"))
})
```

Gambar 3.4 Bentuk Program Akhir

Program akan menuliskan hasil kepada command line yang bisa di-*copy* oleh developer untuk dimasukkan kepada file JavaScript yang ingin diubah.

```
JS test.js > welcomeMessage
1
2   function welcomeMessage(firstname, lastname){
3       console.log("Selamat Datang ", firstname, lastname)
4   }
```

Gambar 3.5 Program dengan Fungsi yang Ingin Diubah

Gambar diatas merupakan salah satu contoh fungsi yang akan diubah dari suatu program. Fungsi tersebut akan mendapatkan manfaat dari perubahan menjadi *arrow const* karena isi fungsi tersebut tidak akan berubah.

```
PS C:\Gibran\ITB\PBO\UAS> node "c:\Gibran\ITB\PBO\UAS\javas.js"

const welcomeMessage = (firstname, lastname) => {
    console.log("Selamat Datang ", firstname, lastname)
}
```

Gambar 3.5 Hasil Akhir dari Program

Programmer dapat mengambil hasil yang ada dalam console dengan mencopy kode tersebut kedalam file yang ingin di update bentuk fungsinya.

IV. Kesimpulan

Bahasa pemrograman selalu berevolusidari tahun ke tahun, entah karena ada bahasa aynag baru ataupun bahasa yang sudah ada mengeluarkan fitur baru untuk mengoptimalkan bahasa tersebut. Sebagai programmer kita harus bisa mengikuti perubahan tersebut dan harus bisa untuk mengikuti perubahan tersebut dengan cepat. Pada salah satu contohnya yaitu fitur baru dari JavaScript ECS 6 yang menambahkan bentuk variable baru ‘const’. Mayoritas dari website yang dibuat menggunakan salah satu bentuk dari JavaScript dan dari semua website tersebut bisa saja membutuhkan variable tipe ‘const’ agar website tersebut lebih optimal. Dengan program ini dan bantuan dari regex, seorang developer dapat merubah sebuah fungsi menjadi fungsi baru tersebut dengan cepat dan otomatis.

V. Ucapan Terima Kasih

Puji syukur kepada Tuhan yang Maha Esa karena berkat dan rahmat-Nya, penulis dapat menyelesaikan makalah Strategi Algrotima yang berjudul “Rejuvenasi Bentuk Fungsi dalam JavaScript Menggunakan Regular Expression” dengan baik. Terima kasih kepada tim pengajar IF2211 Strategi Algoritma Semester II Tahun Ajar 2021/2022 terutama pengajar kelas penulis pada Kelas 01 yaitu Ibu Dr. Masayu

Leylia Khodra, S.T., M.T. yang telah mengajarkan kami materi yang dibutuhkan agar makalah ini selesai. R. Munir.

REFERENCES

[1] Munir, R. (n.d.). Informatika. Retrieved May 22, 2022, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/stima20-21.htm

[2] *Applications of string matching algorithms*. GeeksforGeeks. (2020, September 3). Retrieved May 22, 2022, from https://www.geeksforgeeks.org/applications-of-string-matching-algorithms/

[3] . "Chapter 4. How JavaScript Was Created". speakingjs.com. Archived from the original on 2020-02-27. Retrieved 2017-11-21.

[4] Weber, Tim (May 9, 2005). "The assault on software giant Microsoft". BBC News. Archived from the original on September 25, 2017.

[5] Munir, R. (n.d.). *String matching Dengan regular expression - institut teknologi bandung*. Retrieved May 22, 2022, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf