Explore Regular Expressions!

Regular Expression atau disingkat Regex adalah suatu pattern / deretan karakter spesial yang mendefinisikan sebuah pola untuk pencarian text , text matching

Objectives

- [ ] Mengetahui Kegunaan Regular Expressions
- [ ] Memahami simbol-simbol yang digunakan pada Regular Expressions

Learnings

Mengetahui Kegunaan Regular Expressions

Dengan menggunakan regex, kita dapat menyederhanakan pencarian text string atau interger dalam sebuah variabel. Mungkin kalian berpikir untuk pencarian text kan kita bisa pake fungsi-fungsi String dan Array seperti substr(), indexOf(), slice(), dll.

Tetapi bagaimana kalo pencarian rumit ? dan ada pattern yang harus ditentukan ?. Hal ini jauh lebih mudah kalau kita gunakan Regex. contoh sederhana nya coba temukan / hitung kata **'far'** di dalam paragraph dibawah ini :

```
var paragraph =
far far away, behind the word mountains, far from the countries Vokalia and Consonantia,
there live the blind texts!. Separated they! live in far away from Bookmarksgrove right at the coast of the
a large language ocean. A small . When she reached the first hills! of the Italic Mountains,she had a last
and the subline of her own road!, the Line Lane. Pityful a rethoric question ran over her cheek!
```

Dengan menggunakan regex .match cukup seperti ini :

```
console.log(paragraph.match(/far/g));
```

hasil nya ada 4 kata **'far'** dan sudah langsung masuk kedalam sebuah array.

```
["far", "far", "far", "far"]
```

Kuncinya untuk memahami Regular Expressions adalah mampu menghapal simbol-simbol dibawah ini, beserta kegunaannya. Untuk memudahkannu, kamu bisa mencatat, membuat semacam cheatsheet, atau membuat jembatan keledai dengan caramu sendiri untuk memudahkanmu.

```
. // - Mencocokan karakter apapun, kecuali line breaks(jeda baris/enter).
* // - Mencocokan 0 atau lebih dari karakter terdahulu.
+ // - Mencocokan 1 atau lebih dari karakter terdahulu.
? // - Karakter terdahulu menjadi opsional. Mencocokan 0 atau 1.
\d // - Mencocokan digit apapun
\w // - Mencocokan karakter pada sebuah kata (alphanumeric dan underscore/garis bawah).
$ // - Mencocokan ujung dari sebuah string.
^ // - Mencocokan awal dari sebuah string.
[^a-z] // - Ketika didalam sebuah class karakter, tanda ^ artinya NOT; dalam kasus ini, regex akan mencocok
```

Penggunaan Regular Expressions dalam JavaScript

Menulis Regex

Menulis Regex dalam javascript bisa dalam 2 bentuk, yaitu antara dengan membuat Regex object dengan menggunakan `new RegExp()`, atau menggunakan nilai literal yang diapit oleh karakter flash (*/*).

```
var regexContohSatu = new RegExp("abc");
var regexContohDua = /abc/;
```

Test

Fungsi `test()` akan mengembalikan nilai `true` atau `false`, sesuai dengan pattern regex yang dibuat.

Menggunakan Literal Value

```
var message = 'Regex itu Mudah!';
console.log(/[A-Z]/.test(message));
// mengembalikan nilai true karena minimal satu karakter memenuhi pattern A-Z. Dan true, karena regex itu m

var messageAllLowerCase = 'regex itu susah?';
console.log(/[A-Z]/.test(messageAllLowerCase));
// mengembalikan nilai false karena tidak ada satupun karakter yang memenuhi pattern A-Z. statement tersebu
```

Menggunakan RegExp object

```
var regexPattern = new RegExp('[A-Z]');

var message = 'Regex itu Mudah!';
console.log(regexPattern.test(message));
// mengembalikan nilai true karena minimal satu karakter memenuhi pattern A-Z. Dan true, karena regex itu m

var messageAllLowerCase = 'regex itu susah?';
console.log(regexPattern.test(messageAllLowerCase));
// mengembalikan nilai false karena tidak ada satupun karakter yang memenuhi pattern A-Z. statement tersebu
```

Split

Fungsi `split()` akan memecah setiap pattern yang ditemui ke dalam bentuk array. Fungsi `split()` ini pernah kita gunakan sebelumnya dalam memecah string menjadi array.

```
var str = 'belajar regex itu menyenangkan';
console.log(str.split(/\s/));
// mengembalikan nilai "belajar, regex, itu, menyenangkan" karena \s adalah sebuah pattern untuk satu spasi
```

Replace

Fungsi `replace()` akan mengganti seluruh sebuah blok tertentu dalam sebuah teks, dan menggantinya dengan karakter atau teks lain.

```
var stringAwal = 'Regex itu sangat susah!';
stringHasil = stringAwal.replace(/susah/, 'mudah');
console.log(stringHasil); // mengembalikan nilai "Regex itu sangat mudah!"
```

Match

Fungsi `match()` akan mengembalikan dalam bentuk array setiap kali sebuah kecocokan dengan pattern ditemukan di dalam teks.

Contoh 1 Penggunaan Match - Mencocokan Karakter

```
var message = 'Regex seru DEH!';
console.log(message.match(/e/));
// menampilkan "e", namun hanya sekali

console.log(message.match(/e/g));
// menampilkan "e" untuk setiap "e" yang terdapat di dalam teks. `g` menandakan pencarian secara global, ti

console.log(message.match(/e/gi));
// menampilkan "e" untuk setiap "e" yang terdapat di dalam teks. `i` menandakan pencarian karakter dengan i
```

Contoh 2 Penggunaan Match - Mencocokan Karakter dan Mengecualikan Punctuation atau Simbol

```
var string = 'Walaupun regex banyak mengandung simbol, tapi tidak serumit seperti !@#%^%#$*( , ^%&*!!^& dan
console.log(string.match(/[a-z]+/gi));
//menampilkan ["Walaupun", "regex", "banyak", "mengandung", "simbol", "tapi", "tidak", "serumit", "seperti"
```

Simbol-simbol diatas, sering disebut sebagai Punctiation. Seringkali dalam beberapa kasus, kita mau menghapus semua simbol-simbol diatas.

Jika kamu teliti, kamu pasti menemukan simbol `+` dibelakang `[a-z]`. Simbol `+` disini berarti match akan menyatukan seluruh karakter yang cocok dengan pattern a-z hingga menemukan pattern lain diluar pattern tersebut. Dalam kasus contoh di atas, setiap kali menemukan spasi, contohya pada `Walaupun regex` match akan memisahkan `Walaupun` dan `regex` karena ditemukannya spasi tersebut. Apabila kamu penasaran, cobalah hapus simbol `+` dari code diatas, dan jalankanlah kembali. Hasilnya akan berbeda!

Bagaimana jika pattern regex tidak ditemukan ?

Jika pattern regex tidak dapat ditemukan dalam string, maka fungsi match( ) akan return bukan array kosong, tapi null. Bedanya array kosong dan null: array kosong memiliki length = 0, null tidak memiliki length

Sebagai analogi: array kosong adalah gelas yang tidak terisi air dan null adalah tidak ada gelas sama sekali!

Karena itu, kita perlu berhati-hati saat menggunakan properti length dari hasil fungsi match( ) karena null tidak memiliki length! Null.length akan menyebabkan error. Untuk mengecek apabila suatu fungsi match membalikkan array atau null, kita bisa menggunakan kode if(newArray) , seperti berikut:

```
if (newArray) {
        console.log('newArray bukan null!');
} else {
        console.log('Tidak ada newArray, ini null!');
}
```

Mengetest Kemampuan Regular Expressions

Ada sebuah platform yang cocok untuk melatih kamu dalam menggunakan regex, yaitu regexr.com

References

- RegexOne - Belajar Regex dengan Interaktif
- Regex Codecademy
- [Video Tutorial] (https://www.youtube.com/watch?v=EkluES9Rvak)