**REGULAR EXPRESSION**

Table of Contents

[**1.** **Character Class** 2](#_Toc152253644)

[**2.** **Special Character Classes** 2](#_Toc152253645)

[**3.** **test() Method** 2](#_Toc152253646)

[**4.** **Anchors** 3](#_Toc152253647)

[**5.** **Quantifiers** 3](#_Toc152253648)

[**6.** **Flags** 3](#_Toc152253649)

[**7.** **Lookaheads and Lookbehinds** 3](#_Toc152253650)

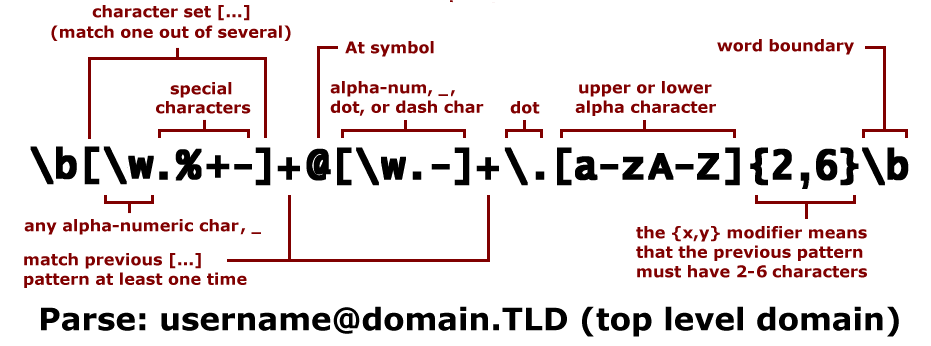
[**8.** **Word boundary (\b \B)** 4](#_Toc152253651)

[**9.** **String character có thể so sánh dc** 5](#_Toc152253652)

[**10.** **ASCII Table** 6](#_Toc152253653)

[**11.** **Dấu ngoặc [ ]** 6](#_Toc152253654)

[**12.** **Lazy quantifiers (x+?, x\*?)** 7](#_Toc152253655)



1. **Character Class**

Dùng dấu [ ] : you can tell the regex engine to match only one out of several characters. The order of the characters inside a character class does not matter. The results are identical/the same

[0-9]: You can use a hyphen inside a character class to specify a range of characters

[0-9a-fA-F]: You can use more than one range. Matches a single hexadecimal digit, case insensitively

Dùng dấu [] để match với nhìu loại character, nên có thể kết hợp với nhau dc. **[^\d:]** nghĩa là match những ký tự ko phải là số và không phải là dấu :

1. **Special Character Classes**

. : is used to find a single character, except newline or other line terminators/ only in 1 line

\w : A word character is a character from a-z, A-Z, 0-9, including the \_ (underscore) character. (e.g [a-zA-Z0-9\_])

\W: not word

\d: find a digit from 0-9. (e.g [0-9])

\D: not digit

\s: whitespace character (

A space,

a tab,

A carriage return character,

a new line/ line-break,

a vertical tab,

a form feed

\S: non-whitespace character

[^abc]: not in brackets. ([^0-9], [^A-Z],..)

1. **test() Method**

const regex = /^0[0-9]{9}$/

let str = 89735968

regex.test(str) return true/false

1. **Anchors**

Chỉ một vị trí trong chuỗi

^ : (not in [ ]) start the string

Khác hoàn toàn với ^ ở trong []

$: end the string

/b: is used to find a match at the beginning or end of a word (/bwer/b)

/B: not beginning or end of a word

1. **Quantifiers**

**n**+ : matches any string that contains at least one **n** or more

n? : matches any string that contains zero or one of n.

n\* : matches any string that contains zero or more of n

n{X} : matches any string that contains a sequence of X n's. (n{2} // nn)

n{X, Y} : matches any string that contains a sequence of X to Y n's. (n{2,4} // nn, nnn, nnnn)

n{X,} : matches any string that contains a sequence of at least X n's. (n{2,} // nn, nnnnn, nnnnnnnnn, nnn...nn)

**{} để check xem có xuất hiện liên tục hay ko!**

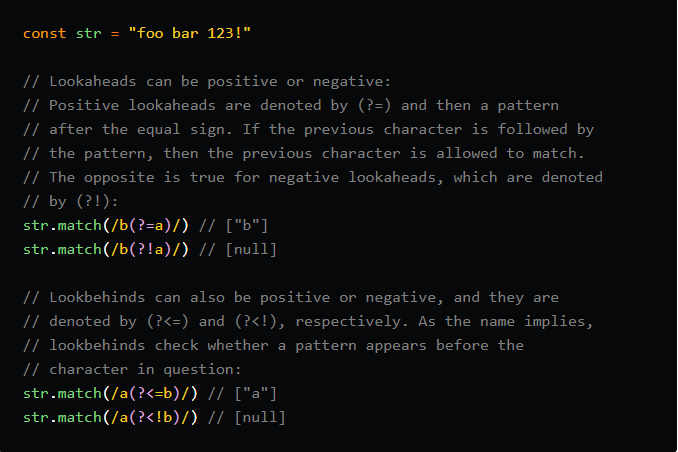
1. **Flags**

/[a-z]/g : a global match (find all matches rather than stopping after the first match). (nếu ko có g thì chỉ match cái đầu tiên)

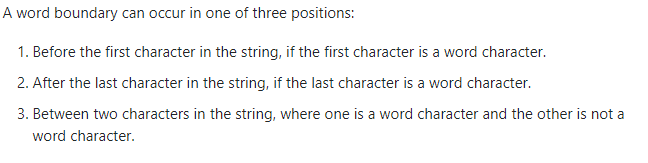
/it/i : used to perform case-insensitive matching. (nếu có thì sẽ tính lun cả HOA và THƯỜNG)

/it/m : beginning (^) and end ($) characters to match the beginning or end of each line of a string(delimited by \n or \r)

1. **Lookaheads and Lookbehinds**



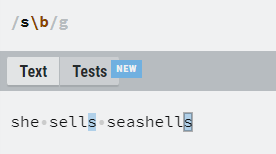
1. **Word boundary (\b \B)**



Nó ko phải là character đặc biệt. Chỉ dùng để check vị trí của ký tự.

Có 3 trường hợp để \b \B này check:

1. ký tự chữ đầu câu
2. ký tự chữ cuối câu
3. 2 ký tự trong 1 chuỗi, 1 ký tự là chữ và 1 ký tự ko phải chữ (ko quan tâm vị trí trước sau của 2 ký tự này)

vd: regex này sẽ check những vị trí có ký tự **s**

và **ký tự ko phải chữ đứng sau s**

và check vị trí **đầu tiên** và vị trí **cuối cùng** của chuỗi

(vị trí **đầu tiên** của chuỗi là s nhưng ko thỏa \b nên ko lấy)

vd: regex này sẽ check word boundary

check ký tự **s**  đứng sau **ký tự ko phải chữ**

và check vị trí **đầu tiên** và vị trí **cuối cùng** của chuỗi

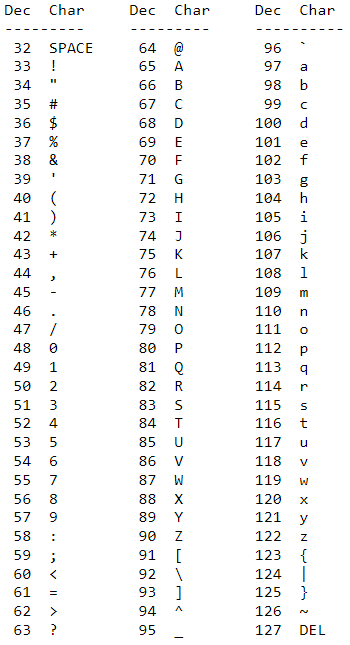
(vị trí **cuối cùng** của chuỗi là s nhưng ko thỏa \b nên ko lấy)

1. **String character có thể so sánh dc**

var solution = (t) => t.split('').map(c => c>'a' ? c.toUpperCase() : c.toLowerCase()).join("")

trong ví dụ c > ‘a’ thì nó sẽ parse sang chartCode j đó để so sánh.

Dựa vào **ASCII Table**

1. **ASCII Table**

ASCII: American Standard Code for Information Interchange

Note:

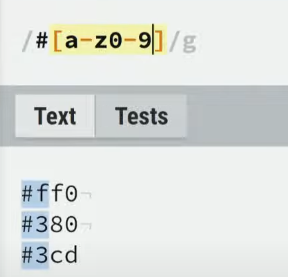
* Ký tự số 48-57
* ký tự in hoa từ 65-90
* Ký tự thường từ 97- 122
* Nên ký tự thường > hơn ký tự hoa
* Nên ký tự số < ký tự chữ

1. **Dấu ngoặc [ ]**

* Gọi là tập hợp hoặc 1 khoảng ký tự



Vd trên là tìm tập hợp các chữ **M** hoặc **t** hoặc **p** hoặc **c** theo sau **op**

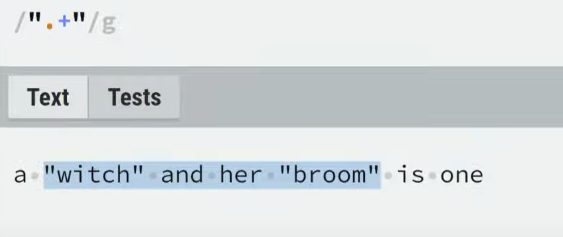
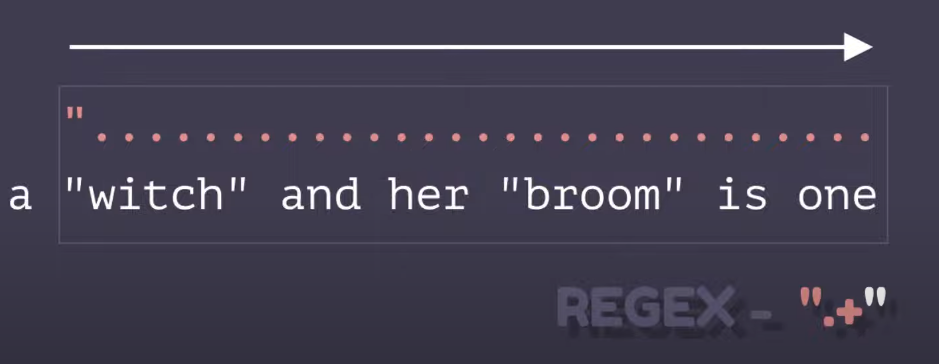
****

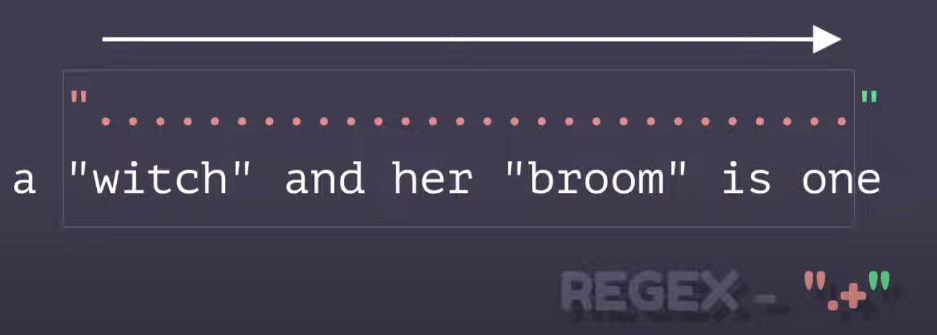
vd trên là tập hợp trong khoảng a đến z **hoặc** 0 đến 9

note: phần [] này là đại diện cho 1 ký tự.

1. **Lazy quantifiers (x+?, x\*?)**

Vấn đề: lấy từ nằm trong dấu “”

 Dùng dấu x+ thì nó lấy dài hết lun  






Cách xử lý: dùng **/”.+?”/g**. Cách này để regex sau khi tìm dc **ký tự bất kỳ** thì sẽ dừng lại tìm **dấu nháy kép** ngay sau từ đó lun.

