RegEx

What is?

A regular expression is a sequence of characters that specifies a search pattern in text.

Starting point

Every regular expression starts with a start_tag and an end_tag, like Html, in the case of RegEx it uses the slash.

Example: // , /S/ or /pizza/

Simple patterns

Examples such as "/pizza/" or "/s/" are the simplest regex patterns, consisting only of the start_tag, the content (a character, word or string), the end_tag and the flag (optional).

this type of pattern only matches the exact content of the string

Example: We want to search a product inside a list.

case 1
pattern: /Be/

- Pizzas
- Coffe
- hamburguers
- Beers

case 2
pattern: /h/

- Pizzas
- Coffe
- hamburguers
- Beers

Flags

There are some flags used in RegEx to delimit the range or type of search. These flags are added to the pattern after the end_tag.

- /g : Global
- /i : Case Insensitive
- /m : Multiline
- /s : Single Line (Dotall)
- /u : Unicode
- /y : Sticky

Example:
pattern: /walter/gi

data:
name: Walter White
email: wwalter@gmail.com
num: 9997777888

Basic matches

- . - Matches any characther except line breaks.
- \d - Matches any digit character (0-9).
- \D - Matches any characther that is not a digit
- \w - Matches any word character.
- \W - Matches any character except word characters.
- \s - Matches any whitespace character.
- \S - Matches any character that is not a whitespace.

Example: get a date with format dd-mm-yyyy

pattern: /\d\d\-\d\d\-\d\d\d\d/g

data:
name: Jesse Pinkman
birth date : 20-12-2000
age: 22

Limits

- \b - Word limit
- \B - Not a Word Limit
- ^ - Beginning of a text string
- $ - End of a text string

Example:

Case 1
pattern: /^pizza/gim
data:
- I like the pepperoni pizza
- Pizza is Italian food

Case 2
pattern: /pizza$/gim
data:
- I like the pepperoni pizza
- Pizza is Italian food

Quantifiers

- * - Match 0 or more of the preceding token
- + - Match 1 or more of the preceding token
- ? - Match between 0 or 1 of the preceding token
- {n} - Exact number of characters where n is a number ({2})
- {n,m} - Min and Max number of characters ({2,4})

Example: Get names
pattern: /\w+\s\w+\s?$/gim
data:
name: Saul Goodman
age: 22

name: Jimmy Mcgill
age: 20

Character sets

- [] - Match any character inside the brackets
- [^] - Characters not in square brackets

Example: Get phone numbers

pattern: /\d{3}[\s-]\d{3}[\s-]\d{2}[\s-]/gim
data:
Spyridon Mihalopoulos
21 years old
2001-01-10
666-555-22
333 777 99

Groups

- () - Group multiple tokens together
- | - Acts like a boolean OR

Example: Get dates of 2001 and 2006

pattern: /(2001|2006)\-\d{2}\-\d{2}/g
data:
date_1: 2022-08-13
date_2: 2001-06-25
date_3: 2010-05-16
date_4: 2006-12-21

List of useful patterns

Validate all:
Patter: /./g

Validate the first capital letter:
Patter: /[A-Z]\w+/g

Validate Names (Firstname Lastname):
Patter: /[A-Z]\w+\s[A-Z]\w+/

Validate text without especial characters:
Patter: /[0-9a-zA-Z]+/

Validate text wit especial characters:
Patter: /[0-9a-zA-Z\W]+/

Validate a secure password:
Patter: /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$/

Validate Email:
Patter: /[0-9a-zA-z.$%*?&-]+\@\w+\.\w+/

Validate date format:
Patter: /\d{4}[-/\s]\d{2}[-/\s]\d{2}/
applies to:
- yyyy-mm-dd
- yyyy/mm/dd
- yyyy mm dd

Validate http url:
Patter: /http[s]?:[/]{2}[\d\w_-]+\.?[\d\w_-]+\.\w+[/]?.*/

Resources

- RegExr - Learn and practice RegEx