RegEx Reference (todo)

Based on modern regex-directed engines with features like lazy quatifiers and backreferences.

Useful

```
^(?=.*?\bbubble\b)(?=.*?\bgum\b).*
```

```
^(?!.*bubble).*
```

```
^(?!(?:foo|bar)$)\w+$
```

Matching

[cols="^1,4,3"] |=== |RegEx |Matching|Example

|[abc]|A single character: a, b or c| |[^abc]|Any single character but a, b, or c| |[a-z]|Any single character in the range a-z| |[A-Z] | Uppercase letters from A to Z| |[a-zA-Z]|Any single character in the range a-z or A-Z| |[0-9] | Digits from 0 to 9 | |^|Start of line| |$|End of line| |\A|Start of string| |\z|End of string| |\b | Word boundary| |\B | Not word boundary| |< | Start of word| |> | End of word| |. | anything | |\s|Any whitespace character| |\S|Any non-whitespace character| |\d|Any digit| |\D|Any non-digit| |\w|Any word character (letter, number, underscore)| |\W|Any non-word character| |a | a character | |ab | ab string | |a|b | a or b | |a?|Zero or one of a| |a*|Zero or more of a| |a+|One or more of a| |a{3}|Exactly 3 of a| |a{3,}|3 or more of a| |a{3,6}|Between 3 and 6 of a| |i | ignore case | |\d | one digit 0-9 | `ie: foo_\d\d = file_55` |\D | non-digit | |\w | one word,letters, digits or _, | `ie: \wfoo- = foo-, foo-bar` |\W | non-word| |\s | whitespace | `ie: foo\sbarr = foo bar` |\S | non-whitespace | |\x | Hexadecimal digit| |\O | Octal digit| |===

Flags

Regex may have flags that affect the search. For example there are 6 of them in JavaScript:

[cols="^1,8"] |=== |Flag | Explanation

|i | With this flag the search is case-insensitive: no difference between A and a.

|g |With this flag the search looks for all matches, without it - only the first match is returned.

|m |Multiline mode.

|s |Enables "dotall" mode, that allows a dot . to match newline character \n.

|u |Enables full Unicode support. The flag enables correct processing of surrogate pairs.

|y |“Sticky” mode: searching at the exact position in the text. |===

Character

[cols="^1,4,3"] |=== |\t | tab| |\r | return| |\n | new line| |\v | Vertical tab| |\f | Form feed| |\xxx | Octal character xxx| |\xhh | Hex character hh| |\ | escapes regex special character, |`ie: \.\*\+\?\$\^\/\\ = .*\|+? $^/\` |===

Quantifer

[cols="^1,2,4"] |=== |+ | once or more, |`ie: \w-\d+ = img-1, img-23, img-1001` |? | once or none, |`ie: \w-\d? = img-1, img-2, img-` |* | zero or more, |`ie: \w*-\d = imgimg-1, imgimg-2` |{3} | three times, |`ie: \w{3}-\d = imgimgimg-1` |{2,} | two or more times, |`ie: \w-\d{2,} = img-23, img-1001` |[123] | matches class, |`ie: [1-3] = 1,2,3` |q[^x] | negates matches, |`ie: q[^x] = question but not iraq` |===

Anchor

[cols="^1,2,4"] |=== |^ | beginning of line, not in [] |`ie: ^app = appMain.js, appAuth.\|js` |$ | end of line, |`ie: .*?Auth.js$ = appAuth.js` |\A | beginning of string| |\Z | end of string| |===

Logic

[cols="^1,2,4"] |=== |() | capturing group, |`ie: A(nt\|pple) = pple in Apple` |(?) | non-capuring group| |===

Modifier

[cols="^1,4,4"] |=== |(?i) | case-insensitive | `ie: (?i)Monday = Monday, monday, monDAY` |(?m) | multiple lines so ^ and $ match in multiple places, | `ie: (?m)1\r\n\^2$\r\n\^3$ = 1\n2\n3` |(?x) | whitespace mode| |===

Lookaround

[cols="^1,2,4"] |=== |(?= | positive lookahead | `ie: (?=\d{10})\d{5} = 01234 in 0123456789` |(?<= | positive lookbehind | `ie: (?<=\d)cat = cat in 1cat` |(?! | negative lookahead | `ie: (?!theatre)the\w+ = theme` |(?<! | negative lookbehind | `ie: \w{3}(?<!mon)ster = Munster` |===

Replacement

[cols="^1,2,4"] |=== $foo - inserts foo $& - inserts entire match $` - inserts preceding string $' - inserts following string $Y - insert Y'th captured group |===

REGULAR EXPRESSIONS YOU SHOULD KNOW:

[cols="2,6"] |=== |Maching | RegEx

|Username| /^[a-z0-9_-]{3,16}$/ |Password| /^[a-z0-9_-]{6,18}$/ |Hex Value| /^#?([a-f0-9]{6}|[a-f0-9]{3})$/ |Slug| /^[a-z0-9-]+$/ |Email| /^([a-z0-9_.-]+)@([\da-z.-]+).([a-z.]{2,6})$/ |URL| /^(https?://)?([\da-z.-]+).([a-z.]{2,6})([/\w .-]*)/?$/ |IP Address| /^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?).){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$/ |HTML Tag| /^<([a-z]+)([^<]+)*(?:>(.*)</\1>|\s+/>)$/ |===

Example & Breakdown

.find html hexidecimal colors like #fff, #aa0099, #CCCCCC [source,bash]

^#(?:[0-9a-fA-F]{3}){1,2}$

[cols="1,4"] |=== |^|anchor for start of string |#|the literal # |(|start of group ^|?:|indicate a non-capturing group that doesn't generate backreferences ^| [0-9a-fA-F]|hexadecimal digit ^| {3}|three times |)|end of group |{1,2}|repeat either once or twice |$|anchor for end of string |===

Examples

[cols="4,3"] |=== |sep[ae]r[ae]te | find `separate` even if it's misspelled seperate |<div\b[^>]>(.?) | match open and closing html `div` tag |\b[A-Z0-9._%+-]+@[A-Z0-9.-]+.[A-Z]{2,4}\b. | match valid email |(?(?=condition)(then1|then2|then3)|(else1|else2|else3)). | if/then/else conditionals |===

.example [source,bash]

[a-f0-9]{3}

- Matches any string consisting of lowercase characters between `a-f` and digits between `0-9`
- At 3 total characters `{3}`.

.example [source,bash]

#?

- Another Quantifier ?,
- matches between 0 and 1 characters of the preceeding expression boundaries #

.example [source,bash]

[a-f0-9]{6}|[a-f0-9]{3}

- Matches any string consisting of lowercase characters between `a-f` and digits between `0-9` At 6 total characters `{6}`
-

OR +

- Matching any string consisting of lowercase characters between `a-f` and digits between `0-9` At 3 total characters `{3}`.

.example [source,bash]

([a-f0-9]{6}|[a-f0-9]{3})

This specifies within the brackets that we are looking for and extracting strings consisting of `a-z` and `0-9` criteria and plating them as Capture Groups.

.example [source,bash]

(/^([a-z0-9_.-]+)@([\da-z.-]+).([a-z.]{2,6})$/)

- In the above pattern, `/^([a-z0-9_\.-]+)` says that email must begin with alpha-numeric characters with only lower case, it may also include ., _, -.
- `@` meaning there must be @ sign after first chars.
- The pattern `[\da-z\.-]+` says that after @ char there must be lowercase alpahabet and may also contain ., -.
- `\.` indicates that there should be a period to separate domain and sub-domain names in the email.
- The pattern `[a-z\.]{2,6})$/` says that the email should end with 2 to 6 lowercase chars. Here 2 indicates the minimum number and 6 indicates the maximum number of characters.

TODO https://gist.github.com/bashworthj/95eee63656342321c248463e1d052970

References:

http://www.regular-expressions.info/quickstart.html

http://www.rexegg.com/regex-quickstart.html

http://en.wikibooks.org/wiki/Regular_Expressions/syntax/posix_basic_regular_expression

http://blog.codinghorror.com/excluding-matches-with-regular-expressions/

Revision #2
Created 2023-05-01 01:54:08 UTC by Dušan Fefer
Updated 2023-05-04 21:27:32 UTC by Dušan Fefer