# Regular Expressions

reg(ular expressions?|ex(p|es)?)

# Inhaltsübersicht

- Überlich
- Regular Expression Engine
- Zeichenklassen
- Quantifier
- Modifier
- ... alle weiten Funktionen

# Regular Expressions

- Was?
  - Beschreibung von Mustern in Zeichenketten
- Warum?
  - Suche Textstellen in Dokumenten
  - Validierung von Eingaben
  - Textstellen aus Dokumenten extrahieren
  - Textstellen einfügen oder ersetzen

# Regex Engine

- Eingabe
  - Subjekt: Ein Text (String) auf den ein Muster (Regular Expression) angewandt wird
  - Muster: Zeichenkette welche eine Musterbeschreibung enthält
- Ausgabe
  - Match: ja oder nein
  - Capturings, aus dem Muster

**Abcd e aeaeafa fef a faefaefafich theth lakefeafealfjlaefflaf afa**

# Regex Engine

- Das Muster wird positionsweise auf das Subjekt angewandt.
- Stimmt eine Position des Musters mit einer Position der Subjekts überein, wird die nächste Position des Musters geprüft.
- Stimmt eine Position des Musters nicht mit der des Subjekts überein, kommt es zum Backtracking. Alle Permutationen werden geprüft.
- Sind alle Position des Musters erfolgreich geprüft gilt dies als Match.

# Regex Engine Beispiel

Subjekt: abc dde abcd aabb bb aa bb ab9d ab?d abdd abkkkd

Muster: ab.d

# ASCII – Das ursprüngliche Regex Alphabet

| Dec | Hx | Oct | Char | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 000 | NUL (null) | 32 | 20 | 040 |   | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 1 | 1 | 001 | SOH (start of heading) | 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 2 | 2 | 002 | STX (start of text) | 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 3 | 3 | 003 | ETX (end of text) | 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 4 | 4 | 004 | EOT (end of transmission) | 36 | 24 | 044 | $ | $ | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 5 | 5 | 005 | ENQ (enquiry) | 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |
| 6 | 6 | 006 | ACK (acknowledge) | 38 | 26 | 046 | & | & | 70 | 46 | 106 | F | F | 102 | 66 | 146 | f | f |
| 7 | 7 | 007 | BEL (bell) | 39 | 27 | 047 | ' | ' | 71 | 47 | 107 | G | G | 103 | 67 | 147 | g | g |
| 8 | 8 | 010 | BS (backspace) | 40 | 28 | 050 | ( | ( | 72 | 48 | 110 | H | H | 104 | 68 | 150 | h | h |
| 9 | 9 | 011 | TAB (horizontal tab) | 41 | 29 | 051 | ) | ) | 73 | 49 | 111 | I | I | 105 | 69 | 151 | i | i |
| 10 | A | 012 | LF (NL line feed, new line) | 42 | 2A | 052 | * | * | 74 | 4A | 112 | J | J | 106 | 6A | 152 | j | j |
| 11 | B | 013 | VT (vertical tab) | 43 | 2B | 053 | + | + | 75 | 4B | 113 | K | K | 107 | 6B | 153 | k | k |
| 12 | C | 014 | FF (NP form feed, new page) | 44 | 2C | 054 | , | , | 76 | 4C | 114 | L | L | 108 | 6C | 154 | l | l |
| 13 | D | 015 | CR (carriage return) | 45 | 2D | 055 | - | - | 77 | 4D | 115 | M | M | 109 | 6D | 155 | m | m |
| 14 | E | 016 | SO (shift out) | 46 | 2E | 056 | . | . | 78 | 4E | 116 | N | N | 110 | 6E | 156 | n | n |
| 15 | F | 017 | SI (shift in) | 47 | 2F | 057 | / | / | 79 | 4F | 117 | O | O | 111 | 6F | 157 | o | o |
| 16 | 10 | 020 | DLE (data link escape) | 48 | 30 | 060 | 0 | 0 | 80 | 50 | 120 | P | P | 112 | 70 | 160 | p | p |
| 17 | 11 | 021 | DC1 (device control 1) | 49 | 31 | 061 | 1 | 1 | 81 | 51 | 121 | Q | Q | 113 | 71 | 161 | q | q |
| 18 | 12 | 022 | DC2 (device control 2) | 50 | 32 | 062 | 2 | 2 | 82 | 52 | 122 | R | R | 114 | 72 | 162 | r | r |
| 19 | 13 | 023 | DC3 (device control 3) | 51 | 33 | 063 | 3 | 3 | 83 | 53 | 123 | S | S | 115 | 73 | 163 | s | s |
| 20 | 14 | 024 | DC4 (device control 4) | 52 | 34 | 064 | 4 | 4 | 84 | 54 | 124 | T | T | 116 | 74 | 164 | t | t |
| 21 | 15 | 025 | NAK (negative acknowledge) | 53 | 35 | 065 | 5 | 5 | 85 | 55 | 125 | U | U | 117 | 75 | 165 | u | u |
| 22 | 16 | 026 | SYN (synchronous idle) | 54 | 36 | 066 | 6 | 6 | 86 | 56 | 126 | V | V | 118 | 76 | 166 | v | v |
| 23 | 17 | 027 | ETB (end of trans. block) | 55 | 37 | 067 | 7 | 7 | 87 | 57 | 127 | W | W | 119 | 77 | 167 | w | w |
| 24 | 18 | 030 | CAN (cancel) | 56 | 38 | 070 | 8 | 8 | 88 | 58 | 130 | X | X | 120 | 78 | 170 | x | x |
| 25 | 19 | 031 | EM (end of medium) | 57 | 39 | 071 | 9 | 9 | 89 | 59 | 131 | Y | Y | 121 | 79 | 171 | y | y |
| 26 | 1A | 032 | SUB (substitute) | 58 | 3A | 072 | : | : | 90 | 5A | 132 | Z | Z | 122 | 7A | 172 | z | z |
| 27 | 1B | 033 | ESC (escape) | 59 | 3B | 073 | ; | ; | 91 | 5B | 133 | [ | [ | 123 | 7B | 173 | { | { |
| 28 | 1C | 034 | FS (file separator) | 60 | 3C | 074 | < | < | 92 | 5C | 134 | \ | \ | 124 | 7C | 174 | | | \| |
| 29 | 1D | 035 | GS (group separator) | 61 | 3D | 075 | = | = | 93 | 5D | 135 | ] | ] | 125 | 7D | 175 | } | } |
| 30 | 1E | 036 | RS (record separator) | 62 | 3E | 076 | > | > | 94 | 5E | 136 | ^ | ^ | 126 | 7E | 176 | ~ | ~ |
| 31 | 1F | 037 | US (unit separator) | 63 | 3F | 077 | ? | ? | 95 | 5F | 137 | _ | _ | 127 | 7F | 177 | &#127; | DEL |

Source: www.LookupTables.com

# Zeichenklassen

| Bezeichnung | Bedeutung | Beispiele |
|---|---|---|
| Positive Zeichenklasse | Definition einer Klasse von Zeichen | [a-z]: Alle Kleinbuchstaben<br>[!-~]: Alle ASCII Zeichen |
| Negative Zeichenklasse | Alles außer dem Spezifizierten Zeichen | [^a-z]: Alles außer Kleinbuchstaben |

Innerhalb der []-Klammern müssen nur [, ], -, \ als Metazeichen escaped werden.

# Spezielle Zeichenklassen

| Zeichen | Bedeutung | Alternative |
|---|---|---|
| \d | Eine Zahl | [0-9] |
| \s | Einen “Weißraum” | [ \f\n\r\t\v] und alle Unicodevarianten zB \u00a0 |
| \w | Ein alphanumerisches Zeichen inklusive _ | [A-Za-z0-9_] |
| \D | Alles außer Zahlen | [^0-9], [^\d] |
| \S | Alles außer “Weißraum” | [^\s] |
| \W | Alles außer alphanumerische Zeichen | [^A-Za-z0-9_] |
| \b | Eine Wortgrenze | (^\w\|\w$\|\W\w\|\w\W) |
| . | Alle Zeichen außer Zeilenumbrüche | [^\n] |

# Anker

| Zeichen | Bedeutung |
| --- | --- |
| ^ | Anfang der Zeichenkette (kann durch Modifier verändert werden) |
| $ | Ende der Zeichenkette (kann durch Modifier verändert werden) |
| \A | Anfang der Zeichenkette |
| \z | Ende der Zeichenkette |
| \Z | Ende der Zeichenkette plus ein oder kein Zeilenumbruch |

# Quantifier

| Regex | Bedeutung | Beispiele |
|---|---|---|
| {x} | Zeichenkette der Länge x | [a-z]{2} |
| {x,y} | Zeichenkette der Länge x bis y | [a-z]{2,4} |
| {x,} | Zeichenkette von mindestens Länge x | [a-z]{2,} |
| + | {1,} | [a-z]+ |
| * | {0,} | [a-z]* |
| ? | {0,1} | [a-z]? |

1. Quantifier sind generell “greedy”. Durch das Anhängen eines ? werden sie “lazy”.
2. Der possessiv (besitznehmende) Quantifier + erlaubt “strenges” Gruppieren

# Lookarounds

| Zeichen | Bedeutung | Beispiele |
|---|---|---|
| (?=...) | Positive Lookahead | ab(?=c) |
| (?!...) | Negative Lookahead | ab(?!z) |
| (?<=...) | Positive Lookbehind | (?<=a)bc |
| (?<!...) | Negative Lookbehind | (?<!x)bc |

Die meisten Implementierungen erlauben keine Quantifier in Lookbehind Ausdrücken.

# Capturing Groups

| Bezeichnung | Bedeutung | Beispiel |
|---|---|---|
| Capturing Group | Klammern werden verwendet für Capturing Groups, mittels \1 können die Gruppen referenziert werden (Backreference) | (abc) (def) \1 \2 |
| Non-Capturing Group | ?: wird verwendet damit der Match nicht referenziert werden kann | (?:abc) |
| Named Capturing Group | Anstatt einer numerierten Backreference kann auch eine Bezeichnung verwendet werden und mit \k'name' referenziert werden | (?'name'abc) \k'name' |

# Modifier

| Zeichen | Bedeutung |
|---|---|
| i | Case Insensitive |
| s | . Zeichen inkludiert auch Zeilenumbrüche |
| m | Multiline |
| x | Free Spacing mode, # sind Kommentare |
| g | Globaler Match |

# Unicode

- Unicode kennt einen Unterschied zwischen Zeichen und Zeichencode, deshalb würde ein ü aus zwei Unicode Zeichen bestehen können, der .-Operation kann deshalb zu Problemen führen
- \X für alle Zeichen inklusive Zeilenumbrüche in Unicode
- Unicode ist in 156 Blöcke aufgeteilt. zB Kyrillisch (U+0400…U+04FF)
- Mit \p{Cyrillic} kann ein Zeichen des kyrillischen Alphabet referenziert werden

# Weitere Funktionen

| | | |
|---|---|---|
| IF/ELSE Konstrukt | Falls die referenzierte Capture Group exisitiert wird der if-Teil gematcht ansonsten der else-Teil | (regex)?(?(1)foo\|bar) |
| Rekursion | Rekursion (~Kopie) des gesamten Musters | L(?R)R |
| Atomic Grouping | Hauptsächlich aus Performanzgründen. Backtracking wird verhindert | a(?>bc\|b)c |

# Check beim Erstellen von Regular Expression

- A (Anker): Können Anker oder Wortgrenzen gesetzt werden?
- G (Greedy): Greedy vs. Lazy Quantifier
- R (Repeat): Gibt es Teile des Musters die sich wiederholen und präziser beschrieben werden können?
- A (Atomic): Atomare oder possesive Quantifier verwenden?

# Erstellung von Regular Expressions

- Verwenden eines Emulators zum testen
  - https://regex101.com/
- So spezifisch wie möglich vs. so Allgemein wie möglich
- Wartbarkeit/Lesbarkeit vs Präzision
- Quellcodeformatierung (ja/nein)
- Nach Ideen/Vorschlägen bei Google suchen

# Wichtige Frage

- Wer ist für die Wartung der Regular Expressions zuständig?
- Wann kann eine Regular Expression aufgeteilt werden?
- Wann sind die Grenzen von Regular Expressions erreicht?
- Gibt es Alternativen zu Regular Expressions?