Reguläre Ausdrücke *(8.10.2014 - Konstantin Kobs)*

Alle Materialien sind unter `http://github.com/konstantinkobs/RegEx/` abrufbar.

Cheatsheet

Aufbau

```
/Pattern/Flags
```

- Die `/` vor und nach dem `Pattern` heißen **Delimiter/Begrenzer**
- `Pattern` beschreibt den Aufbau der zu suchenden Zeichenkette
- `Flags` sind optional und beeinflussen die Art und Weise, wie im Text nach dem Pattern gesucht wird

Pattern

Bis auf einige Zeichen mit besonderer Bedeutung (**Metacharacters**) stehen alle Zeichen im Pattern für sich selbst. Um *Metacharacters* für sich selbst stehen zu lassen, müssen sie mit einem Backslash (`\`) davor escaped werden.

- `|` ist der `Oder`-Operator, d.h. was links oder rechts steht muss zutreffen
- `( )` definieren Gruppen
- `{min,max}` hinter einem Buchstaben oder einer Gruppe geben die minimale und maximale Wiederholungsanzahl an; `{min,}` matcht alles **ab** `min` Vorkommnissen; `{anzahl}` matcht die genaue Anzahl `anzahl`
- *Quantoren:* `+ = {1,}`, `* = {0,}` und `? = {0,1}`
- `[ ]` wählt eines der aus in der Menge stehenden Zeichen
- `[a-z]` ist ein *Range* von `a` bis `z`; Ranges können beliebige Spannen haben
- `.` matcht jedes Zeichen *(bis auf neue Zeilen)*

- `[^a]` bedeutet: Jedes Zeichen außer `a`
- `^` und `$` bezeichnen den Anfang und das Ende des zu durchsuchenden Textes
- `\d = [0-9]`; `\D = [^\d]`
- `\w = [a-zA-Z0-9]`; `\W = [^\w]`
- `\s` sind Leerräume und neue Zeilen; `\S` Gegenteil
- `\b` stellt Wortanfänge und -enden dar

Flags

Buchstaben, die am Ende des Regulären Ausdruckes stehen. Sie haben keine zu beachtende Reihenfolge und sind optional.

- `g` *(__g__lobal)*: Sucht alle Vorkommnisse im Text
- `i` *(__i__gnore case)*: Nicht mehr auf Groß- und Kleinschreibung achten
- `m` *(__m__ultiline)*: `^` und `$` beziehen sich nicht auf den Anfang und das Ende des *Strings*, sondern jeder *Zeile*.

Greedy und Lazy *(Gierig und Genügsam)*

Die Quantoren `+` und `*` sind von Haus aus *greedy*, das heißt, sie matchen eine möglichst große Zeichenkette. Manchmal ist dies nicht das gewünschte Verhalten. Sie lassen sich mit einem nachgestellten `?` *lazy* machen, sprich, sie matchen so kurze Zeichenketten wie möglich.

Backreference

Um nur auf Teile der gefundenen Zeichenkette zurückgreifen zu können, umschließen wir den Teil des Regulären Ausdrucks mit Klammern. Diese Abschnitte nennt man **Capturing Group**.

Im Pattern kann man dann mit `\1` auf die erste Gruppe zurückreferenzieren, mit `\2` auf die zweite usw.

Beim Ersetzen von Text mit Hilfe von Regulären Ausdrücken wird mit `$1`, `$2`, usw. auf die zuvor entdeckten Gruppen zugegriffen.