

/^(Regular Expre(s)\2ions)\$/
(Regular Expressions)

Alexander Lais
6.12.2007

alexander.lais@gmx.de
unfug.org

“What he says?”

Regular Expressions, was'n das?

Grundlagen

Look-ahead und Look-behind

Ausprägungen

Praktische Beispiele

Lesenswertes

Regular Expressions, was'n das?

- Flexible Textmuster
- Einsatz
 - ➔ Validierung
 - ➔ Textextraktion
 - ➔ Suchen & Ersetzen
- Endlicher Zustandsautomat

Beispiel

Anfang des Strings

Ende des Strings

oder

`/^(Hello (World|Unfug))!?$$/i`

Delimiter

Delimiter

Modifier

Gruppe 1

Gruppe 2

Beispiel

`/^(Hello (World|Unfug))!?$ /i`

- ✓ Hello World!
- ✓ hello unfug
- ✓ HeLLo UnFuG!
- ⊙ Hallo Welt
- ⊙ Hello World!!!!
- ⊙ Baum

Grundlagen

- Literale
- Platzhalter
- Quantifizierer
- Backreferences
- (Non-)Matching Groups
- Greedy & Lazy
- Modifier
- Lookaround

Literale

- Einfacher Text

/Ein Vortrag beim unfug/

- Meta-Zeichen müssen maskiert werden

/http:\\/\\/www\\.unfug\\.org/

Platzhalter (1/2)

^ Anfang des Strings

\$ Ende des Strings

. Beliebiges Zeichen

**** Escape-Zeichen

[a-z] Zeichenklasse, hier z.B. a-z

[^a-z] Zeichenausschluss, z.B. alles außer a-z

**** Backslash

Platzhalter (2/2)

\w	Wortbestandteil (Buchstaben)
\s	Whitespace
\S	Alles außer Whitespace
\d	Zahl
\b	Wortgrenze
\t	Tabulator
\n	Newline

Quantifizierer

- * Beliebig viele Wiederholungen
- + Mindestens eine Wiederholung
- ? Höchstens einmal
- {n,m} n bis m Wiederholungen.

Backreferences

- Verweis auf zuvor erfasstes

$/\langle ([b i u]) \rangle (. * ?) \langle \backslash / \backslash \underline{1} \rangle /i$

1 2

✓ fetter Text

✓ <i>kursiver Text</i>

✓ <u><i>unterstrichen kursiv</i></u>

● <i>falsch

● <a>ein Link

Capturing Groups

```
/^(Hello (World|Unfug))!?$$/i
```

- **Beispieltext:**
Hello World!
- **Match 1:** „Hello World“
- **Match 2:** „World“

Non-Capturing Groups

```
/^(Hello (?:World|Unfug))!?!$/i
```

- Beispieltext:
Hello World!
- Match 1: „Hello World“

Modifizier

- g** Global, matching so oft wie möglich
- m** Multiline Mode, Beeinflusst **^** und **\$**
- U** Ungreedy (PHP-Besonderheit)
- i** Case-insensitive
- s** . erfasst auch **\n**
- x** Free spacing Mode (Kommentare, etc.)
- e** Evaluate

Modifizier - Beispiele - g

/([aeiou])/

Das ist das Haus vom Nikolaus
1

/([aeiou])/g

Das ist das Haus vom Nikolaus
1 2 3 4 5 6 7 8 9 10

Modifizier - Beispiele - m

s/^(.*)\$/“\$1”/mg

vorher

Das ist das ←
Haus vom Nikolaus

nachher

“Das ist das ” ←
“Haus vom Nikolaus”

Modifiers - Beispiele - **s**

/ist (.*) vom/s

Das ist **das** ←
Haus vom Nikolaus

Modifizier - Beispiele - e

Perl

```
$text = "4, 9, 36, 49";  
$text =~ s/(\d+)/sqrt($1)/eg;  
print $text;
```

Ausgabe

2, 3, 6, 7

Greedy & Lazy

- Quantifizierer nehmen alles was geht

```
/* ein Integer... */  
int i = 0;  
/* Noch mehr Kommentar */
```

`/\/*(.*)*/sg`

```
/* ein Integer... */  
int i = 0;  
/* Noch mehr Kommentar */
```

- Greediness lässt sich steuern

Greedy & Lazy

Das selbe nochmal mit Lazy

```
/* ein Integer... */  
int i = 0;  
/* Noch mehr Kommentar */
```

`/\/*(.*)*/sg`

```
/* ein Integer... */  
int i = 0;  
/* Noch mehr Kommentar */
```

- Lazy nimmt so wenig wie möglich

Greedy & Lazy

- Der Lazy ? Operator wirkt auf:
?, +, *, {n,m}
- Lazy ?, also ??, hä?
 - Beeinflusst Ausführung
 - Zuerst wird es ohne probiert
 - Einsatz wenn ein Match unwahrscheinlich ist

Lookaround

- Look-ahead, Text danach

- positiv `/foo(?=bar)/`

- negativ `/foo(?!bar)/`

- Look-behind, Text davor

- positiv `/(?<=foo)bar/`

- negativ `/(?<!foo)bar/`

Look-behind

Einschränkungen (PCRE)

- Fixe Länge des Patterns

✓ `/(?<=foo|food)bar/`

⊙ `/(?<=food?)bar/`

➔ Keine Backreferences

Ausprägungen

- „Geschmacksrichtung“ ;)
- Funktionsweise der „Engine“
Wie wird die Expression angewendet

„Geschmacksrichtungen“

- Einmal Erdbeer und zwei Kirsch?!

(Nein...)

- Englisch: „Flavor“

- Genaue Syntax der Implementierung

- Einschränkungen und Erweiterungen

- Alle Beispiele für PCRE

(Perl Compatible Regular Expressions)

Arten von Engines

- **Deterministic Finite Automaton (DFA)**
 - Jedes Zeichen einmal untersuchen
 - Alle Möglichkeiten gleichzeitig
 - egrep, awk, MySQL, procmail

- **Nondeterministic Finite Automaton (NFA)**
 - Jede Möglichkeit nacheinander
 - Backtracking bei Fehlschlag
 - Reihenfolge der Möglichkeiten wichtig!
 - Perl, Ruby, Python, Java, .NET, ...

DFA - Beispiel*

Regular Expression

`/to(nite|knight|night)/`

Text

`tonight`

* Beispiele aus „Mastering Regular Expressions“, J. E. F. Friedl

DFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

Text

tonight

DFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

Text

tonight

DFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

The diagram shows the regular expression `/to(nite|knight|night)/` in a dark blue rounded rectangle. The opening and closing slashes are green. The parentheses and the vertical bars are blue. The words 'nite', 'knight', and 'night' are white. Two white arrows point upwards to the first 'n' of 'nite' and the first 'n' of 'night'.

Text

tonight

The diagram shows the text 'tonight' in a dark blue rounded rectangle. A white arrow points upwards to the 'n' in 'tonight'.

DFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

Text

tonight

DFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

Text

tonight

DFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

Text

tonight

DFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

Text

tonight

✓ Match

NFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

A diagram of the regular expression `/to(nite|knight|night)/`. The opening and closing slashes are green. The `to` part is white. The parentheses are blue. The three alternatives `nite`, `knight`, and `night` are white, separated by vertical yellow bars.

Text

Backtracking-Position

↓
tonight
↑

A diagram showing the text `tonight`. A yellow arrow points down to the `o` character, and a white arrow points up to the `t` character.

NFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

A diagram of the regular expression `/to(nite|knight|night)/`. The opening and closing slashes are green. The `to` part is white with a white arrow pointing up to the `t`. The parentheses are blue. The three alternatives `nite`, `knight`, and `night` are white, separated by vertical yellow bars.

Text

Backtracking-Position

tonight

A diagram of the text `tonight`. A white arrow points up to the `t` at the beginning. A yellow arrow points down to the `n` at the end of the `ton` prefix.

NFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

The regular expression `/to(nite|knight|night)/` is shown with annotations. The opening and closing slashes are green. The opening parenthesis is blue, and the closing parenthesis is also blue. The vertical bars separating the alternatives are yellow. A white arrow points to the 'n' in 'nite'.

Text

Backtracking-Position

tonight

The word 'tonight' is shown with a yellow arrow pointing down to the 'n' and a white arrow pointing up to the 'n'.

NFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

The regular expression `/to(nite|knight|night)/` is displayed. The opening and closing slashes are green. The opening parenthesis is blue, and the closing parenthesis is also blue. The three alternatives 'nite', 'knight', and 'night' are separated by vertical bars. The first bar is yellow, and the second bar is also yellow. A white arrow points upwards to the 'n' in 'nite'.

Text

Backtracking-Position

tonight

The word 'tonight' is shown. A yellow arrow points downwards to the 'n' in 'tonight'. A white arrow points upwards to the 'i' in 'tonight'.

NFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

Text

Backtracking-Position

tonight

NFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`



Text

Backtracking-Position

tonight



NFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

Text

Backtracking-Position

tonight

NFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

Text

Backtracking-Position

tonight

NFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

Text

Backtracking-Position

tonight

NFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

Text

Backtracking-Position

tonight

NFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

Text

Backtracking-Position

tonight

NFA - Beispiel

Regular Expression

`/to(nite|knight|night)/`

Text

Backtracking-Position

tonight

✓ Match

Praktische Beispiele

- Suchen und Ersetzen (Editor, ...)
- E-Mail-Adressen erkennen
- bbCode-Formatierung
- Ein Wort Wort zuviel
- Validierung von Datum und Zeit

Suchen und Ersetzen

SQL

```
SELECT id, name, mail FROM users;
```

Ziel

```
this.id = db.getField("id");  
this.name = db.getField("name");  
this.mail = db.getField("mail");
```

Suchen und Ersetzen

SQL

```
SELECT id, name, mail FROM users;
```

```
s/SELECT ((([a-z]+)(?:, )?)+) FROM .*/$1/
```

```
id, name, mail
```

```
s/([a-z]+)(?:, )?/this\.$1 = db\.$1 = db\.$1; \n/g
```

```
this.id = db.getField("id");  
this.name = db.getField("name");  
this.mail = db.getField("mail");
```

E-Mail-Erkennung

Text

Schreiben Sie eine Mail an alexander.lais@gmx.de.

Regular Expression

```
/\b([-._\w\d]+\@ # Username @  
# Hostname  
[-a-z0-9]+([-a-z0-9]+)*\.(com|org|[a-z]{2})  
)\b/x
```

Achtung! Keine vollständige Lösung nach RFC.

1. Es fehlen TLDs wie info, museum, etc.
2. alternative Schreibweisen wie IP als Host, etc.

bbCode selbstgemacht

Text

Das `[b]`kann`[/b]` doch `[i]`nicht`[/i]` sein!!!11einselF

Regular Expression

```
s/\([uib]\)\([\^]+\)\[\/\1\]/<$1>$2<\/\1>/ig
```

Ergebnis

Das ``kann`` doch `<i>`nicht`</i>` sein!!!11einselF

Ein Wort Wort zuviel

Text

Wir danken für für Ihr Schreiben vom vom 11.12.2006. Es es ist bisher im Papierkorb gewesen und wurde der derben Chefin vorgelegt. Bitte haben Sie Geduld.

Regular Expression

```
/\b([a-zAÖÜß]+) # Das erste Wort
((?:\s|<[^\>]+>)+) # Whitespace oder <Tags>
(\1\b) # Das zweite Wort
/igx
```

Validierung von Daten

Datum

06.12.2007

Regular Expression

```
/(0?[1-9]|[12][0-9]|3[01])\. #Tag  
(0?[1-9]|1[012])\. #Monat  
([1-9][0-9]{3}) # Jahr /x
```

Achtung! Unsinn wie 31.02.2007 möglich.

Validieren von Uhrzeiten

Uhrzeit

13:37

Regular Expression 1

`/([01]?[0-9]|2[0-3]):([0-5][0-9])/`

0	1	2	3	4	5	6	7	8	9
00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23						

Validieren von Uhrzeiten

Uhrzeit

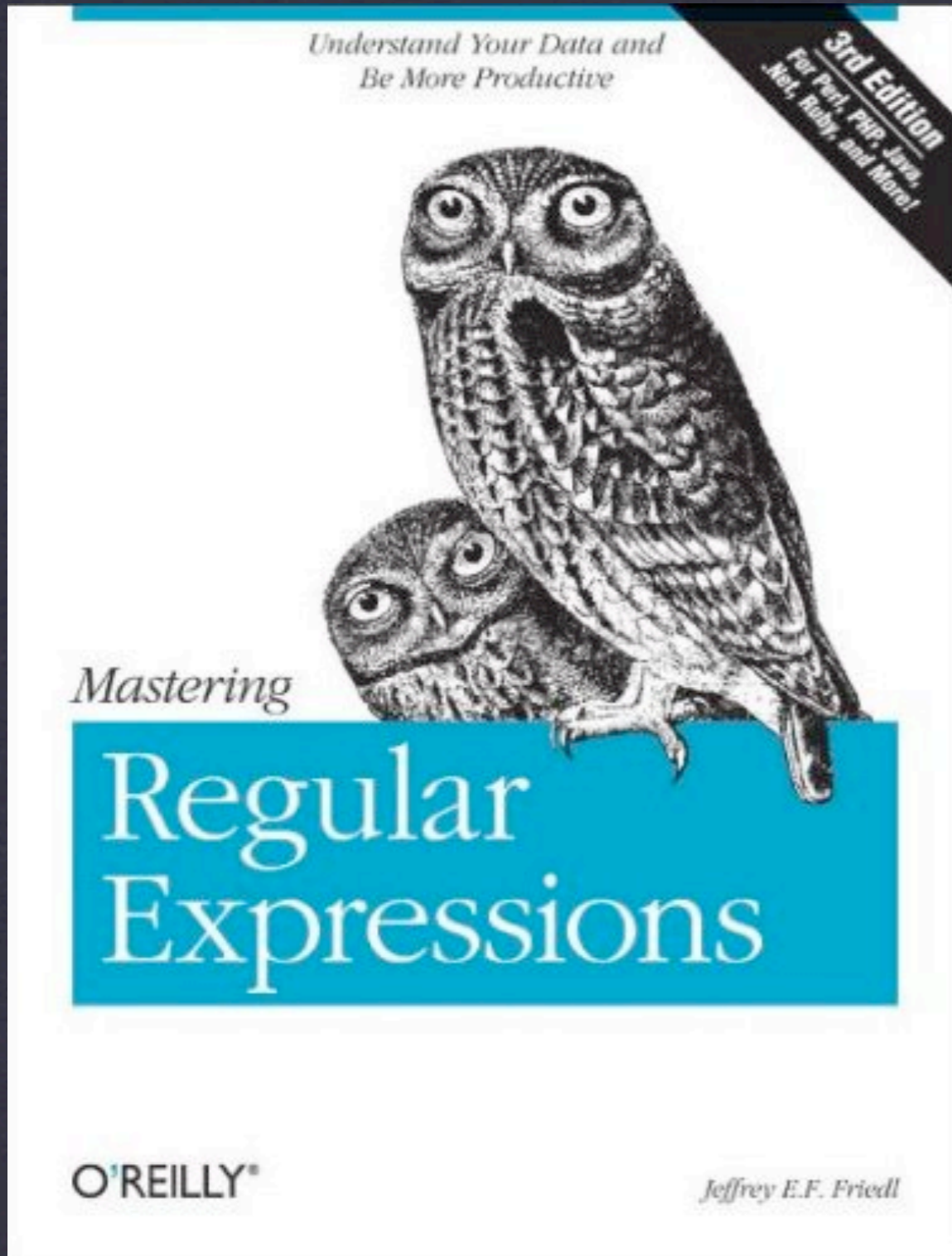
13:37

Regular Expression 2

`/([01]?[4-9]|[012]?[0-3]):([0-5][0-9])/`

0	1	2	3	4	5	6	7	8	9
00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23						

Lesenswertes



Mastering Regular Expressions

Jeffrey E. F. Friedl
O'Reilly, 3rd Edition

Die Referenz

Lesenswertes

The screenshot shows the homepage of [Regular-Expressions.info](http://www.regular-expressions.info). The site has a blue and orange color scheme. At the top, the title "Regular-Expressions.info" is displayed in a large, bold, blue font. Below the title, a navigation bar contains links for "Tutorial", "Tools & Languages", "Examples", "Books", and "Reference". The main content area is divided into several sections:

- Welcome**: A sidebar on the left contains a "Welcome" header and a list of links: "Quick Start", "Tutorial", "Tools and Languages", "Examples", "Books", "Reference", "Print PDF", and "About This Site".
- Easy to create and understand regular expressions today**: A featured article with a small image of a tiger. The text describes RegxBuddy, a tool for creating and analyzing regex patterns, and provides a link to "Get your own copy of RegxBuddy now".
- Welcome to Regular-Expressions.info**: A section with the subtitle "The Premier website about Regular Expressions". It contains a paragraph explaining what a regular expression is and another paragraph discussing the use of regular expressions in text editors and specialized tools like PowerGREP.
- PowerGREP 3**: A section with a small icon of a magnifying glass. It describes PowerGREP as a powerful regex-based text processing tool and provides instructions on how to use regular expressions to search through files.

regular-expressions.info

Grundlegende und weiterführende Themen

Testprogramme für verschiedene Flavors

`/(\?+|\??|\???) /g`