"Merkwürdigkeiten" - Regex mit /g verhält sich "komisch"

In diesem Teil von "Merkwürdigkeiten in Perl" geht es um ein Verhalten von Regulären Ausdrücken, über das man durchaus stolpern kann wenn man sich mit Regulären Ausdrücken nicht so auskennt.

Reguläre Ausdrücke sind sehr mächtig und sie sind mit ein Grund dafür, dass Perl in der "Textbearbeitung" so stark ist. Und dank den *Perl Compatible Regular Expressions* (PCRE) ist diese mächtige Syntax auch an vielen anderen Stellen einsetzbar (z.B. Apache, PHP und Postfix). Aber je mächtiger ein Tool ist, umso mehr Verwirrung kann das Tool stiften wenn der User sich nicht so sehr gut damit auskennt.

Die Dokumentation der Regulären Ausdrücke in Perl ist sehr umfangreich und ist auf verschiedene Dokumente aufgeteilt, die unterschiedlich tief in das Thema eintauchen. Diese Dokumente sind:

- perldoc perlre
- perldoc perlrequick
- perldoc perlretut

Und da sich in Perl 5.10 einiges bei den Regulären Ausdrücken getan hat, sind dort noch `perldoc perlreguts` und `perldoc perlreapi` hinzugekommen. Diese Dokumentation behandelt aber nur das Backend.

Doch jetzt zum Thema: Bei perl.de ist eine Frage zu Regulären Ausdrücken und der Auswirkung des `/g`-Modifiers aufgekommen. Zu dem Beispiel

```
$text = "foo foo";

for ($i=1;$i<=10;$i++){
   print "$i " if $text =~ /foo/g;
}
```

gibt es folgendes Ergebnis

```
1 2 4 5 7 8 10
```

und der Fragensteller hat sich gewundert, warum das nicht `1 2 3 4 5 6 7 8 9 10` ergibt.

Das ist keine Stolperstelle, wenn man die Dokumentation (`perldoc perlretut`) gelesen hat. Dort steht deutlich, dass `/g` dafür sorgt, dass Perl sich die Position nach einem Match merkt und dann genau an dieser Stelle weitermacht. Die Positionsangabe wird nur bei einem erfolglosen Versuch oder bei Änderungen am zu durchsuchenden String zurückgesetzt.

Was passiert also bei dem oben gezeigtem Code?

$i = 1 : Regex matcht erstes "foo", merkt sich Position 3
$i = 2: Regex matcht zweites "foo", merkt sich Position 7
$i = 3: Regex kann nichts mehr matchen, weil im String nichts mehr steht -> Keine Ausgabe -> jetzt erst wird die Position zurückgesetzt.
$i = 4: jetzt beginnt das ganze wieder von vorne.

Wie kann man das umgehen?
In dem man den Regulären Ausdruck einfach in Listenkontext verwendet, denn damit wird gleich der gesamte String auf alle Vorkommen von "foo" untersucht und die Treffer werden zurückgeliefert. Da nach dem letzten Treffer ein erfolgloser Match kommt, wird die Position zurückgesetzt. Dadurch beginnt die Regex-Engine beim nächsten Schleifendurchlauf wieder von vorne.

```
$text = "foo foo";

for ($i=1;$i<=10;$i++){
   print "$i " if( () = $text =~ /foo/ig );
}
```

Renée Bäcker