Regular Expressions in Perl

Simple Matching

```
m/abc/;       # find 'abc'
m#abc#;       # ...
ma\abca;      # ...
/abc/;        # ...
/abc def/;    # find 'abc def'

/^abc/;       # abc at beginning
/abc$/;       # abc at the end
/^$/;         # empty line
```

Substitution

```
s/a/b/;       # first a->b
s/a/b/g;      # all a->b

s/Hi!/Ho!/g;  # 'Hi' -> 'Ho'

# remove control chars
s/[[:ctrl:]]//g;
```

Translation

```
tr/a/b/;       # all a->b
y/a/b/;        # all a->b

tr/abc/x/;     # a->x,b->x,c->x
tr/xxx/abc/;   # only x->a

tr/[a-z]/[A-Z]/;   # upper case

tr/A-Za-z/N-ZA-Mn-za-m/; # ROT13
```

Special Characters

\d	Digit
\D	Non-Digit
\w	Word Character
\W	Non-Word Character
\s	Whitespace
\S	Non-Whitespace

Quantities

```
/^\s?\S/;    # 0..1 spaces
/^\s*\S/;    # 0..n spaces
/^\s+\S/;    # 1..n spaces

/a{3}/;      # 3 times 'a'
/ab{3}/;     # 3 times 'b'
/(ab){3}/;   # 3 times 'ab'

/a{3,4}/;    # 3..4 times 'a'
/a{3,}/;     # 3..n times 'a'

/a.+b/;      # maximal match
/a.+?b/;     # non-greedy match
```

Character Classes

:alpha:	alphabetic
:alnum:	alpha numeric
:upper:	upper case
:lower:	lower case
:digit:	\d
:xdigit:	hex number
:print:	printable
:space:	\s
:blank:	space, enter
:punct:	punctuation
:graph:	alnum and punct
:word:	\w
:ascii:	ASCII chars
:control:	control chars

Grouping and Alternatives

```
/(abc)def/;        # $1='abc'
/(a)b(cd)/;        # $1='a',$2='cd'

/(a)(?:b)(c)/;     # $1='a',$2='c'

/(start|begin)/;   # either 'start'
                   # or 'begin'
```

Perl Language Overview

Blocks and Statements

```
1;          {            {
$a;           s1;          s1;
$a=1;         s2;          s2
            }            }
```

Functions

```
sub add {
   my ($a, $b) = @_;
   return $a + $b;
}
```

Data Types

```
$s='Hallo';    # scalar string
$s="Hallo";
$nr=5;         # scalar decimal
$nr=1.456;     # scalar float point

@arr=(1,2,3);    # array of numbers
$#arr;           # (length-1) -> 2
$arr[0];         # gives '1'
$arr[1];         # gives '2'
$arr[2];         # gives '3'
$arr[-1];        # gives '3';

%h=(1,'joe',2,'lisa');
$h{1};           # gives 'joe'
$h{3}='dog';     # adding {3 => dog}

$ref=\$nr;     # scalar ref
$$ref;         # resolve scalar ref
$ref=\%h;      # hash ref
%$ref;         # resolve hash ref
```

Working with Arrays

```
@arr = split /,/, $text;
$text = join ',', @a;

@arr = sort @arr;
@arr = reverse @arr;

push @arr, $i; $i = pop @arr;
unshift @arr, $i; $i = shift @arr;

foreach my $i (@arr) {
   print $i;
}
```

Working with Hashes

```
foreach my $s (keys %h) {
   print $s."=>".$h{$s}."\n";
}

@arr = values(%h);
```

Variable Scopes

```
my $i;         # local private var
local $s;      # dynamic variable
local FILE;    # local file handle
```

Using Modules

```
use XML::Parser;
my $p = new XML::Parser();

require Time::Local;
print localtime();
```

Control Flow

```
if($a == 1) {      unless($i >= 55) {
   doThis();          doThat();
} else {           } else {
   doThat();          doThis();
}                  }

exit if($o ne 'false');
exit unless($o eq 'false');

while($i > 5 && $i < 25)
   doThis();
}
for($i=0;$i<100;$i++) {
   print $i**;
}
```

Working with Files

```
# reading a file in one step
open FILE, '/etc/hosts';
@hosts = <FILE>;
close FILE

# reading and writing per line
open(FILE1, '<$filename1');
open(FILE2, '>$filename2');
print FILE2, $_ while(<FILE1>);
close(FILE1);
close(FILE2);
```