ERLANG CHEAT SHEET V1.0

Marouan OMEZZINE (marouan.omezzine[~at~]gmail[~dot~]com) http://www.myownpercept.com/2009/03/erlang-cheat-sheet/

SAMPLE MODULE

```
-module(foo).
-export([sayhi/0]).
% This is a comment.
sayhi() ->
io:format("Happy Erlanging!~n" ).
```

GETTING HELP

erl -man <module_name>

CASE EXPRESSIONS

```
case Expression of
        Pattern1 [when Guard1] -> Expr_seq1;
        Pattern2 [when Guard2] -> Expr_seq2;
        ...
end
```

IF EXPRESSIONS

```
if
        Guard1 ->
                Expr_seq1;
        Guard2 ->
                Expr_seq2;
        ...
end
```

TERM COMPARISONS

X **>** *Y X is greater than Y.*
X **<** *Y X is less than Y.*
X **=<** *Y X is equal to or less than Y.*
X **>=** *Y X is greater than or equal to Y.*
X **==** *Y X is equal to Y.*
X **/=** *Y X is not equal to Y.*
X **=:=** *Y X is identical to Y.*
X **=/=** *Y X is not identical to Y.*

IO

Io:format("I am ~s~n", [String]).
Io:fwrite("I am ~s~n", [String]).
***~n** : new line | **~s** : string | **~f**=> float | **~w** : standard output | **~p** : like ~w but breaks after each line*

LIST

ListA = [1,2,3,4].
Returns [1,2,3,4]
ListB = [1+7,hello,2-2,{cost, apple, 30-20},3].
Returns [8,hello,0,{cost,apple,10},3]
[H|T] = ListA.
Returns the Head and Tail.

Comprehensions

*L = [X*X || X <- [1,2,3]].*
*Or lists:map(fun(X) -> X*X end, [1,2,3])*
This gives the output: [1,4,9]

Simple Comprehensions

*L = [{X,Y} || X <- [1,2,3,4], Y <- [1,2,3,4], X*X == Y].*
This gives the output: [{1,1}, {2,4}]

Permutations

[[X]++[Y] || X<-"HT", Y<-"HT"].
Returns ["HH","HT","TH","TT"]

++

[1,2] **++** *[2,3]. returns [1,2,2,3]*

--

[1,2,3] --[2]. returns [1,3]

TUPLE

T= {1.9, 22, 3.99}
***element**(2, T) returns 22 (2 is the index)*
{_, Val, _} pattern matching to retrieve a value

RECORDS

Define a record
-record(person, {name, surname}).

Create an instance of a record
M=#person{name="Marouan",surname="O"}.

Access a single field in a record
M#person.name.

Extracting multiple fields
# person { name=A, surname=B } = M.

STRING BASICS

Str = "The \\n escape sequence escapes a line"
***lists:sort**("CBDbaazerty").returns "BCDaabertyz".*
***lists:subtract**("abcd", "ab"). returns "cd".*
***lists:suffix**(".mp3", "music.mp3"). returns true.*
***lists:nth**(1, "ABC"). returns 65 (1 is the index).*
***length**("ABC").returns 3.*
lists:duplicate**(5, $*). returns"**"*
string:chars**($*, 5).returns "**"*
***lists:append**(["Happy ", "Erlanging", "!"]).returns "Happy Erlanging!"*

ESCAPE SEQUENCE

\b Backspace
\d Delete
\e Escape
\f Form feed
\n New line
\r Carriage return
\s Space
\t Tab
\v Vertical tab
\NNN \NN \N Octal characters (N is 0..7)
\^a..\^z or \^A..\^Z Ctrl+A to Ctrl+Z
\' Single quote
\" Double quote
\\ Backslash
\C The ASCII code for C (C is a character) (An integer)

ERLANG SHELL:

***init:stop().** shutdown cleanly.*
***erlang:system_info(version).** the erts[i] version*
***init:script_id().**the major Release version.*
***b().** display all variable bindings.*
***e(N).** repeat the expression in query <N>.*
***f().** forget all variable bindings.*
***f(X).** forget the binding of variable X.*
***h().**history.*
***history(N).** set how many previous commands to keep.*
***results(N).** set how many previous command results to keep.*
***v(N).** use the value of query <N>.*
***rd(R,D) .** define a record.*
***rf().** remove all record information.*
***rf(R).** remove record information about R.*
***rl().** display all record information.*
***rl(R).** display record information about R.*
***rp(Term).** display Term using the shell's record information.*
***rr(File).** read record information from File (wildcards allowed).*
***rr(F,R).** read selected record information from file(s).*
***rr(F,R,O).** read selected record information with options.*

***** commands in module c *****

***bt(Pid).** stack backtrace for a process.*
***c(File).** compile and load code in <File>.*
***cd(Dir).** change working directory.*
***flush().** flush any messages sent to the shell.*
***help() .** help info.*
***i().**information about the system.*
***ni().**information about the networked system.*
***i(X,Y,Z).** information about pid <X,Y,Z>.*
***l(Module).** load or reload module.*
***lc([File]).** compile a list of Erlang modules.*
***ls().** list files in the current directory.*
***ls(Dir).** list files in directory <Dir>.*
***m().**which modules are loaded.*
***m(Mod).** information about module <Mod>.*
***memory().** memory allocation information.*
***memory(T).** memory allocation information of type <T>.*
***nc(File).** compile and load code in <File> on all nodes*
***nl(Module).** load module on all nodes*
***pid(X,Y,Z).** convert X,Y,Z to a Pid.*
***pwd().**print working directory.*
***q().** quit - shorthand for init:stop().*
***regs().** information about registered processes.*
***nregs().** information about all registered processes.*
***xm(M).** cross reference check a module.*
***y(File).** generate a Yecc parser.*

***** commands in module i *****

***ih().** print help for the i module.*

FUNCTIONS

Anonymous:
Z = fun(X) -> 2*X end.
Named functions:
cube(X) -> X*X .

BIF: BUILT-IN FUNCTIONS

BIF list:
http://www.erlang.org/doc/man/erlang.html

DATES AND TIME

{Date={Year,Month,Day},Time={Hour, Minutes,Seconds}} = erlang:localtime().
{{2006,9,25},{4,34,29}}
b().
Date = {2006,9,25}
Day = 25
Hour = 4
Minutes = 34
Seconds = 29
Time = {4,35,50}
Year = 2006

CRASH DUMP ANALYZER

Summary : Analyzing the erl_crash.dump after a crash.
webtool:start().

RUN_ERL

Summary : Redirect Erlang input and output streams.
run_erl [-daemon] pipe_dir/ log_dir "exec command [command_arguments]"

FILE TYPES

module ***.erl***
include file ***.hrl***
release resource file ***.rel***
application resource file ***.app***
boot script ***.script***
binary boot script ***.boot***
configuration file ***.config***
application upgrade file ***.appup***
release upgrade file ***relup***

[i] Erlang Run-Time System