Erlang/OTP BEAM runtime and shell

CONCEPTS

Every Erlang shell expression ends with a dot. Variables start upper case and bind once; use f(X). or f(). to forget them.

Node one running VM (erl)
- **Application** started as a unit
  - **Supervisor** restarts its children
    - **Process** lightweight, own mailbox

Nodes only talk if they share a cookie and the same name style: -sname and -name nodes cannot connect to each other.

START THE VM

`erl`
Start an interactive shell

`erl -noshell`
Run without a shell

`erl -noinput`
Never read stdin (implies -noshell)

`erl -eval "<expr>"`
Evaluate an expression at start

`erl -pa <dir>`
Prepend a directory to the code path

`erl -config <file>`
Load application config (sys.config)

COMPILE

`erlc <file>.erl`
Compile to a .beam file

`erlc -o <dir> <file>.erl`
Put output in a directory

IN THE SHELL

`c(<mod>).`
Compile and load a module

`l(<mod>).`
Load or reload a module

IN THE SHELL (cont.)

`q().`
Quit the node

`b().`
Show variable bindings

`f().`
Forget all bindings

`v(-1).`
Reuse the last result

`Ctrl+G`
Job control menu

`Ctrl+C`
Break menu (a to abort)

DISTRIBUTION

`erl -sname <name>`
Node with a short host name

`erl -name <name>@<host>`
Node with a full host name

`erl -setcookie <cookie>`
Set the shared cookie

`erl -remsh <node>`
Shell connected to a remote node

`net_adm:ping(<node>).`
Connect; pong means reachable

`rpc:call(<node>, <mod>, <fun>, [<args>]).`
Call a function on another node

ESCRIPT

`escript <file>`
Run a script

`main(Args) →`
Entry point, Args is a list of strings

REBAR3

`rebar3 compile`
Fetch deps and compile

`rebar3 shell`
Shell with project on the path