Elixir Functional language on BEAM

CONCEPTS

> Everything is an expression and data is immutable: functions return new values, and = matches rather than assigns.

Source .ex / .exs
↓ elixirc / mix
Bytecode .beam files
↓ loads into
BEAM VM lightweight processes
↓ message passing
Supervision restart on crash

RUN & COMPILE

`iex`
Start the interactive shell

`iex -S mix`
Shell with the project compiled and loaded

`elixir` *`<file.exs>`*
Run a script file

`iex --sname` *`<name>`* **`--remsh`** *`<node>`*
Attach to a running node

IEX HELPERS

`h` *`<Module.fun/arity>`*
Show docs for a function

`i` *`<term>`*
Inspect type and details of a value

`recompile`
Recompile the Mix project

`Ctrl+C Ctrl+C`
Quit the shell

`dbg(`*`<expr>`***`)`**
Print expression with pipeline steps

DATA & PATTERN MATCHING

`{:ok, v} =` *`<expr>`*
Match a tuple, bind v

DATA & PATTERN MATCHING (cont.)

`[h | t] =` *`<list>`*
Split head and tail

`%{key: v} =` *`<map>`*
Pull a value out of a map

CONTROL FLOW

`x |> f() |> g(a)`
Pipe result as first argument

`case` *`<expr>`* **`do pattern -> ... end`**
Branch on patterns

`with {:ok, a} <- f(), {:ok, b} <- g(a) do ... end`
Chain matches, bail on first miss

MODULES & FUNCTIONS

`defmodule` *`<Name>`* **`do ... end`**
Define a module

`def` *`<name>`***`(`***`<args>`***`) do ... end`**
Public function

STANDARD LIBRARY

`Enum.map(`*`<en>`***`, fn x -> ... end)`**
Transform each element

`Enum.filter(`*`<en>`***`, &pred/1)`**
Keep matching elements

`Enum.reduce(`*`<en>`***`, `***`<acc>`***`, fn x, acc -> ... end)`**
Fold into one value

`IO.inspect(`*`<v>`***`, label: "`***`<x>`***`")`**
Print any value, returns it

TESTING & DOCS

`mix test`
Run all tests

`test "`*`<name>`***`" do ... end`**
Define a test

`assert` *`<expr>`*
Fail unless truthy