IN.5022 — Concurrent and Distributed Computing

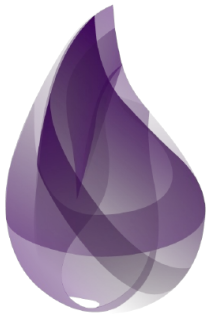
Elixir

Prof. P. Felber

pascal.felber@unine.ch

Agenda

- Introduction to Elixir
- Language essentials (vs. Erlang)
- Learning resources



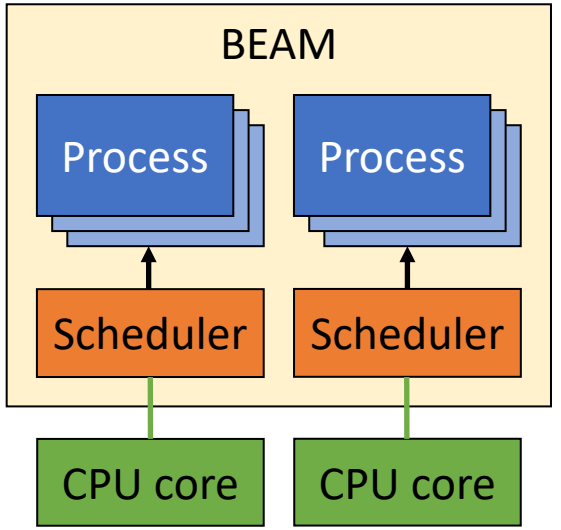
https://pragprog.com/titles/elixir16/
https://www.manning.com/books/elixir-in-action-second-edition

What is Elixir?

- Elixir is a dynamic, functional language designed for building scalable and maintainable applications
  - Created by José Valim
  - Combines features of Ruby, Erlang and Clojure
  - Provides productive tooling and an extensible design
    - Compile-time metaprogramming with macros, polymorphism via protocols, etc.
- Elixir leverages the Erlang VM
  - Inherits from its ability to run low-latency, distributed and fault-tolerant systems
- Used in production by many companies (e.g., Discord, Divvy, Heroku…)

Elixir builds on the BEAM VM

- The Erlang virtual machine, BEAM, is a single OS process
  - It uses its own schedulers to distribute execution of Erlang processes (*unit of concurrent execution*) on CPU cores
  - The scheduler is an OS thread responsible for executing multiple Erlang processes
  - BEAM uses multiple schedulers to parallelize the work over available CPU cores



Elixir shell

- Start with
  `iex`
- Stop with
  `^C ^C`
- Get help with
  `h()`
- Unlike Erlang, expressions do *not* end with dot

```
% iex
Erlang/OTP 23 [erts-11.0.3] …
Interactive Elixir (1.11.1) …
iex(1)> ^C
BREAK: (a)bort (A)bort with dump …
^C
% iex
Erlang/OTP 23 [erts-11.0.3] …
Eshell V11.0.3  (abort with ^G)
iex(1)> h()
Welcome to Interactive Elixir …
iex(2)> h(Enum)
Provides a set of algorithms …
iex(3)> IO.puts("Hello, world!")
Hello, world!
:ok
```

Elixir vs. Erlang: running code

- Erlang shell

```
-module(module_name).
-compile(export_all).

hello() ->
  io:format("~s~n", ["Hello, world!"]).
```

```
1> c("module_name").
{ok,useless}
2> module_name:hello().
Hello, world!
ok
```

- Elixir shell

```
defmodule ModuleName do
  def hello do
    IO.puts "Hello, World!"
  end
end
```

```
iex(1)> c("module_name.ex")
[ModuleName]
iex(1)> ModuleName.hello
Hello, world!
:ok
```

Elixir vs. Erlang: operator names

- Some operators are spelled differently or not supported

Erlang	Elixir	Meaning
`and`	N/A	Logical ‘and’, evaluates both arguments
`andalso`	`and`	Logical ‘and’, short-circuits
`or`	N/A	Logical ‘or’, evaluates both arguments
`orelse`	`or`	Logical ‘or’, short-circuits
`=:=`	`===`	Match
`=/=`	`!==`	Negative match
`/=`	`!=`	‘Not equals’
`=<`	`<=`	‘Less than or equals’

Elixir vs. Erlang: delimiters

- In Erlang, expressions are terminated with a dot (`.`)
- Comma (`,`) is used to evaluate multiple expressions in one context

- In Elixir, expressions are delimited by a line break (⏎) or a semicolon (`;`)

```
X = 2, Y = 3.
X + Y.
```

```
x = 2; y = 3
x + y
```

Elixir vs. Erlang: variable names

- Variables in Erlang can only be assigned once
  - Special command **f()** in shell to erase the binding variables
  - Capitalised variable names (≠ functions, modules, atoms)

```
1> X = 10.
10
2> X = X + 1.
** exception error: no match of right…
3> f(X).
ok
4> X = 10 + 1.
11
```

- Elixir allows assigning to a variable more than once
  - Circumflex (^) is used to match against the value of a previously assigned variable
  - "Snake case" for variable and functions, "camel case" for modules, either for atoms

```
iex(1)> a = 1
1
iex(2)> a = 2
2
iex(3)> ^a = 3
** (MatchError) no match of right…
```

Elixir vs. Erlang: calling functions

- Erlang uses a colon (`:`) to separate function from module names

```
lists:last([1, 2]).
```

- Elixir uses a dot (`.`) to separate function from module names (`:` for atoms)

```
List.last([1, 2])
```

- Erlang modules are represented by atoms and Erlang functions can be invoked in Elixir as follows

```
:lists.sort([3, 2, 1])
```

Elixir: next steps…

1. Install Elixir
2. Read “Erlang/Elixir Syntax: A Crash Course”
3. Watch the 1-hour Elixir tutorial by *Derek Banas*
4. Watch the 10-minute Elixir documentary by *Honeypot*
5. Practice with Elixir!

https://elixir-lang.org/…
…crash-course.html

https://youtu.be/pBNOavRoNL0

https://cult.honeypot.io/originals/elixir-the-documentary

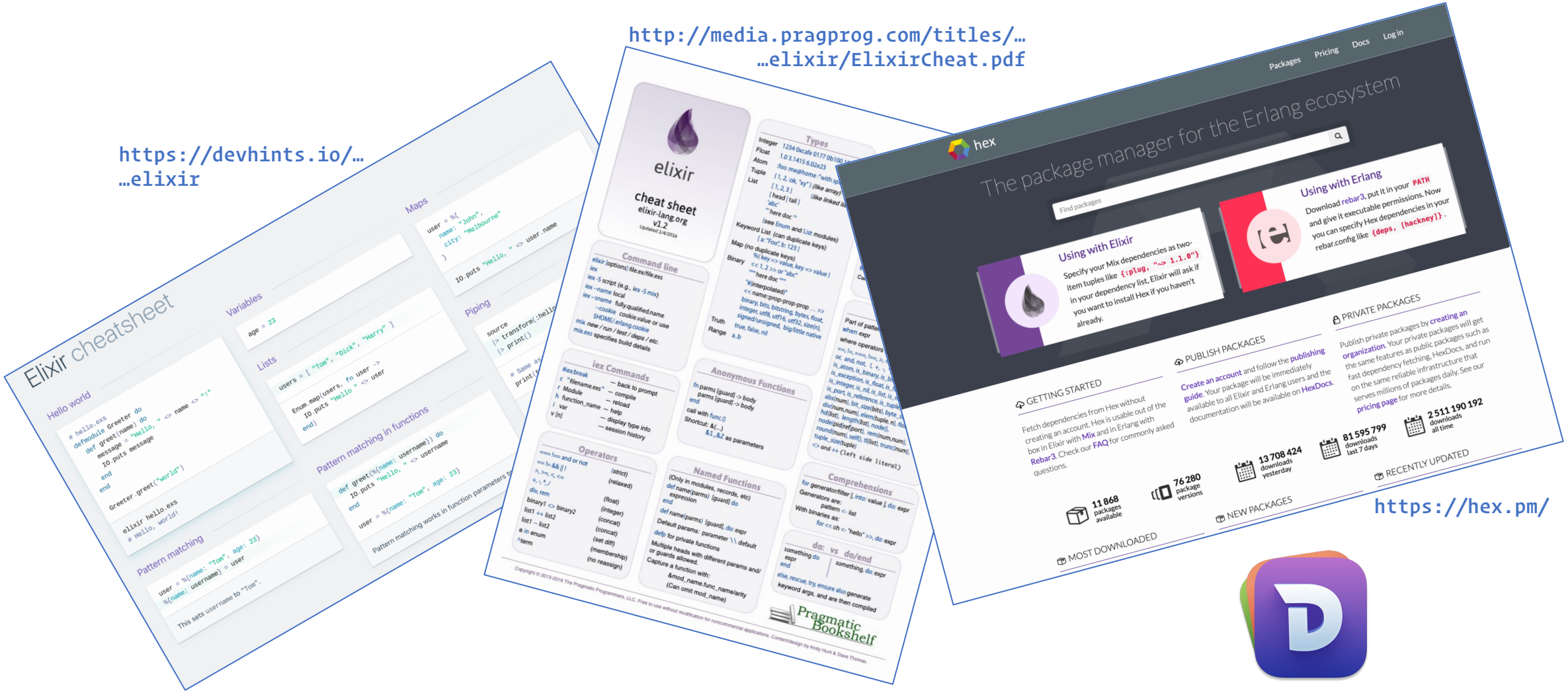
Learning Elixir

- Official documentation: best resource!
  - https://elixir-lang.org/
  - https://elixir-lang.org/getting-started/
- Some good tutorials
  - https://elixirschool.com/
  - https://www.tutorialspoint.com/elixir/
  - https://learnxinyminutes.com/docs/elixir/
    (also in French)
- A list of learning material
  - https://serokell.io/blog/learn-elixir

Useful resources: cheat sheets, packages

https://devhints.io/…
…elixir

http://media.pragprog.com/titles/…
…elixir/ElixirCheat.pdf



https://hex.pm/

https://kapeli.com/dash