FUNCTIONAL PROGRAMMING (WITH ELIXIR)

http://krishnanm.com

Presented on 3rd September 2021

FIRSTLY ...

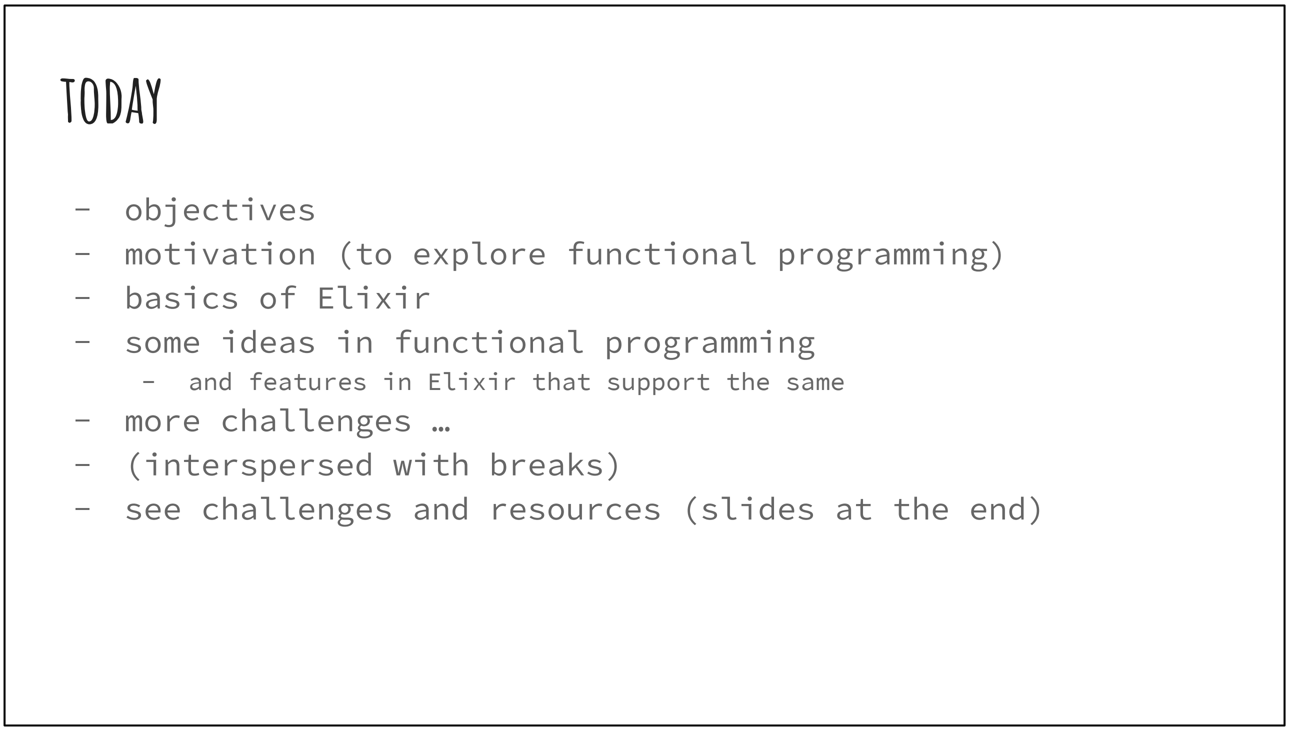
Fork the repository at:

https://github.com/krishnan-mani/elixir_starter

then, follow instructions in the README.md to launch your own workspace on Gitpod.io

ADMINISTRIVIA

- For those in the room
    - Please stay COVID-safe!
    - submit your lunch orders by 9:30
- For everyone,
    - **make it interactive**: share your ideas, opinions, experiences, questions, answers, challenges as they come to you
    - don’t hesitate to ask for assistance, there is no judgement
    - stay connected (before and after breaks)
    - volunteers needed: time-keeping, zoom attendants
- For those remote
    - mute/unmute with care



TODAY

- objectives
- motivation (to explore functional programming)
- basics of Elixir
- some ideas in functional programming
  - and features in Elixir that support the same
- more challenges …
- (interspersed with breaks)
- see challenges and resources (slides at the end)

1. break after each top-level item

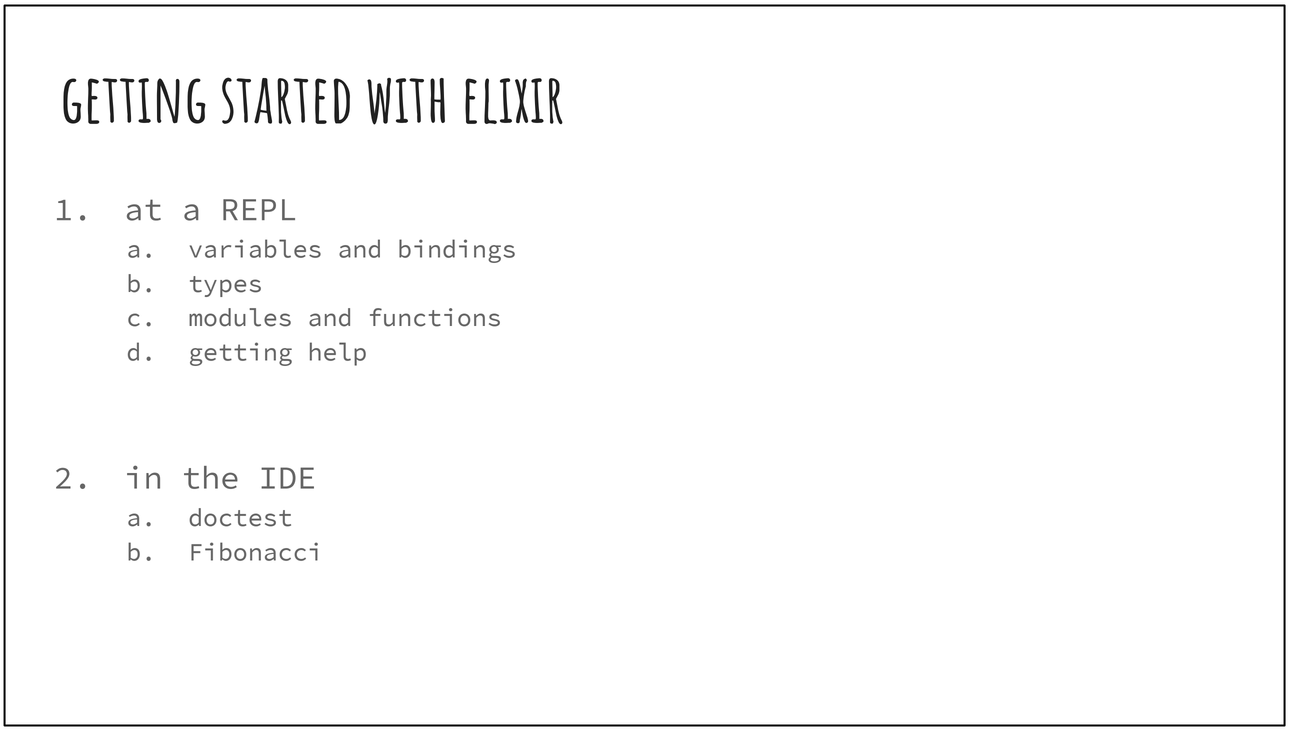
OBJECTIVES

- get a taste of an alternative approach to solving problems (writing software)
- learn about and write some code, in an idiomatic fashion using Elixir
- focus on problem-solving

MOTIVATION

- focus on data and transformation, stay close to data
- avoid mutation and state: easier to reason about, easier to test, ...
- prefer a declarative approach: the “what”, not “how”

using Elixir



GETTING STARTED WITH ELIXIR

1. at a REPL
   a. variables and bindings
   b. types
   c. modules and functions
   d. getting help

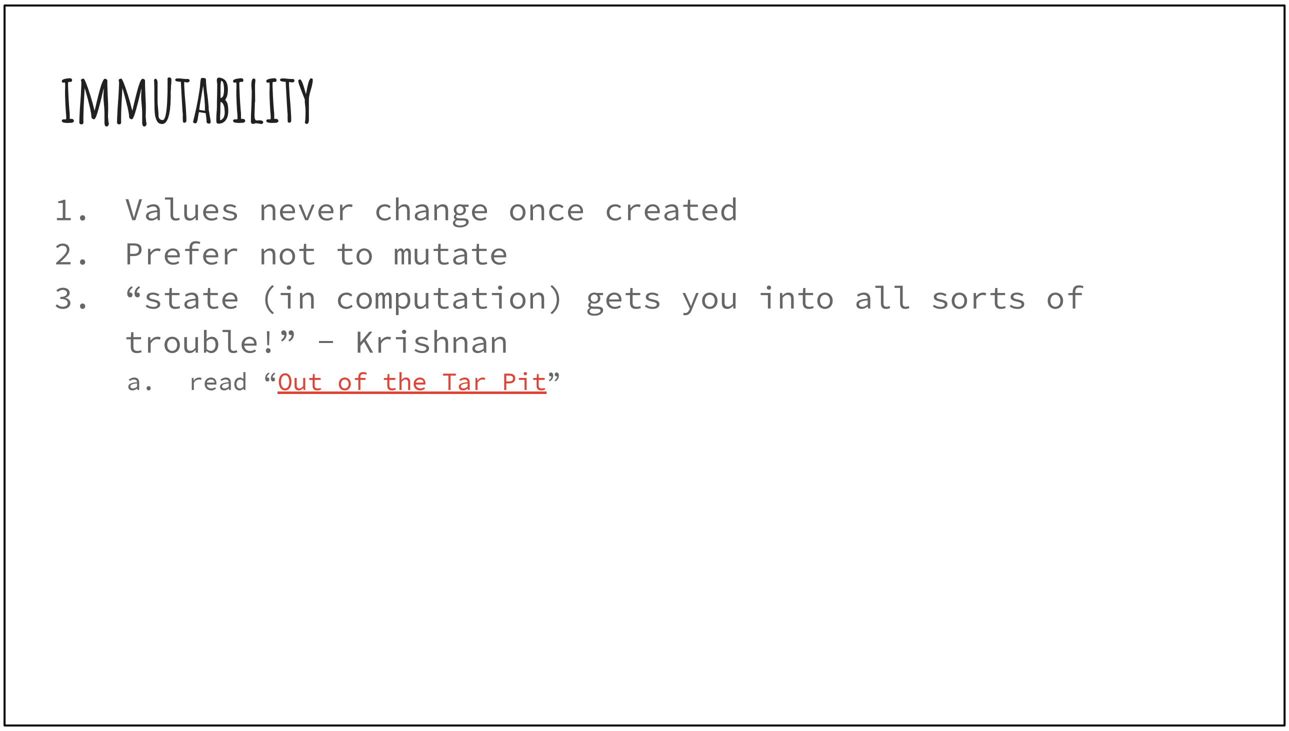
2. in the IDE
   a. doctest
   b. Fibonacci

1. basic data types
2. data structures: tuples, lists, maps
3. destructuring
4. modules, functions, arity
5. anonymous functions

Getting started with functional programming

It is better to have 100 functions operate on one data structure than 10 functions on 10 data structures

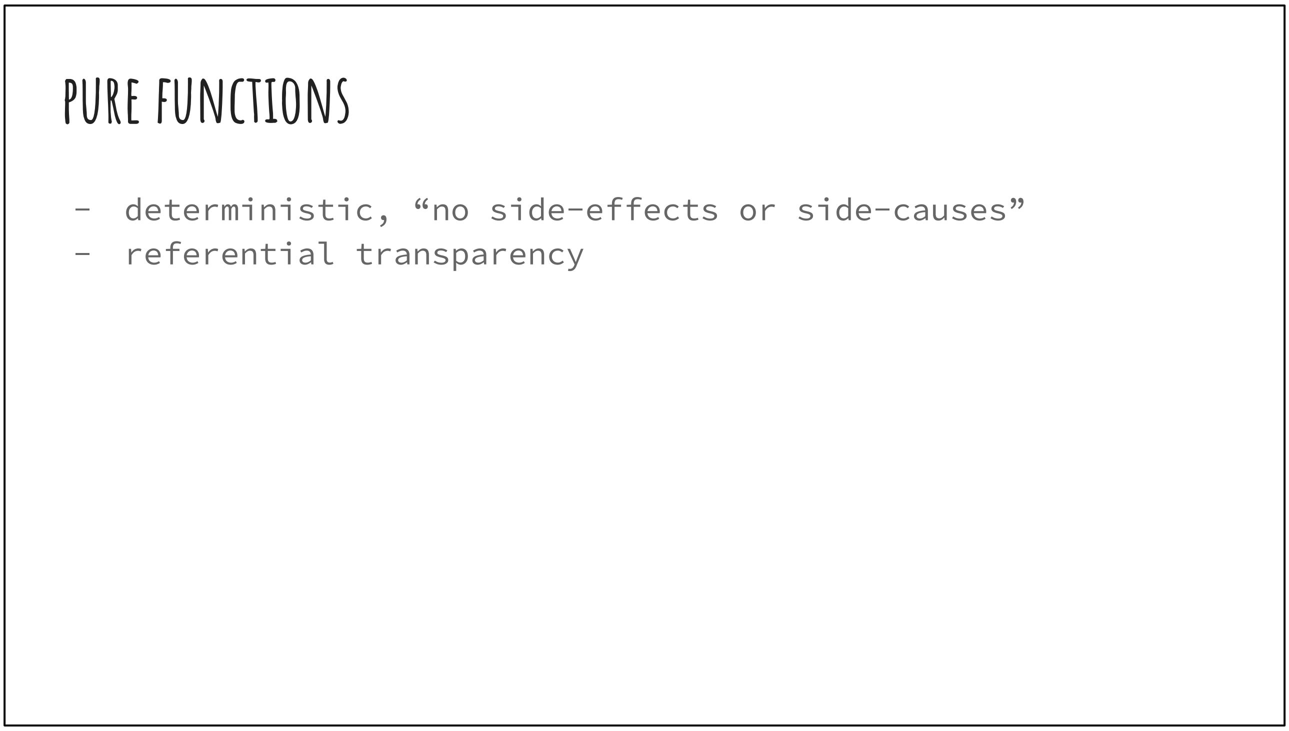
- Alan Perlis



IMMUTABILITY

1. Values never change once created
2. Prefer not to mutate
3. “state (in computation) gets you into all sorts of trouble!” - Krishnan
   a. read “Out of the Tar Pit”

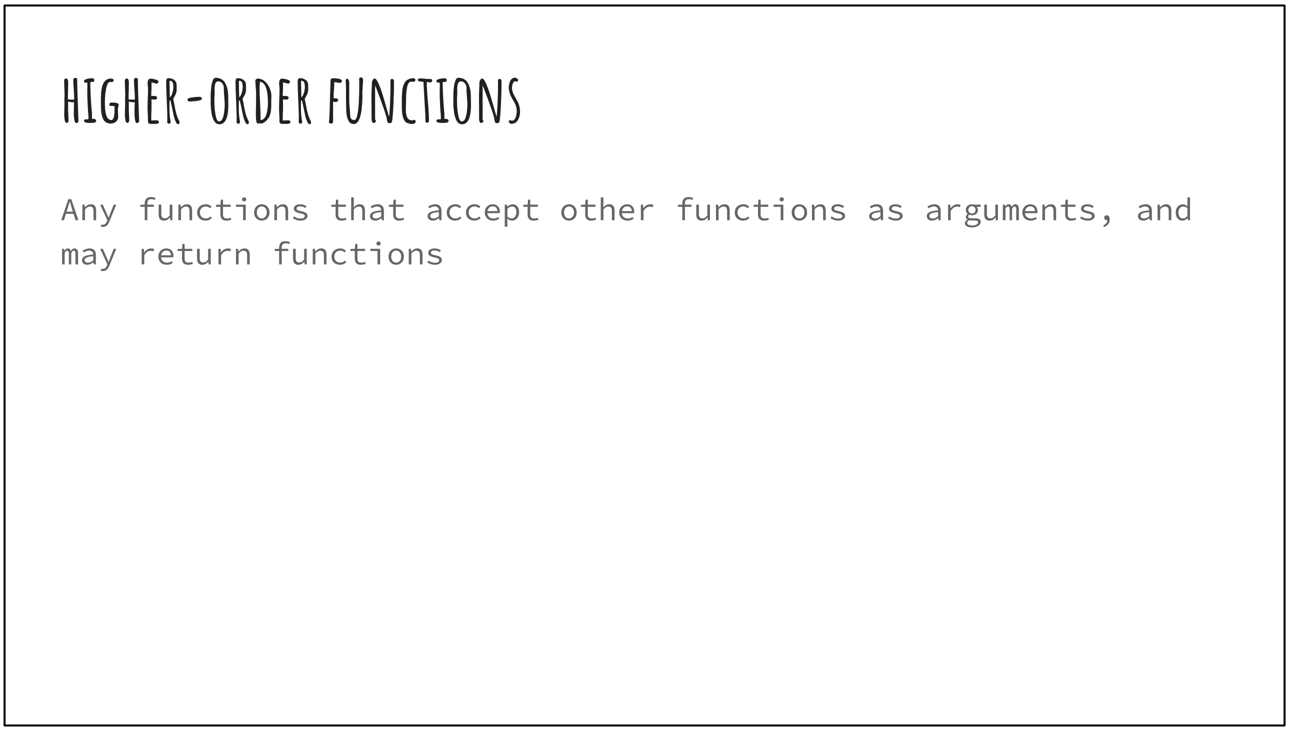
- persistent data structures



PURE FUNCTIONS

- deterministic, “no side-effects or side-causes”
- referential transparency

- examples
- closures
- composing functions

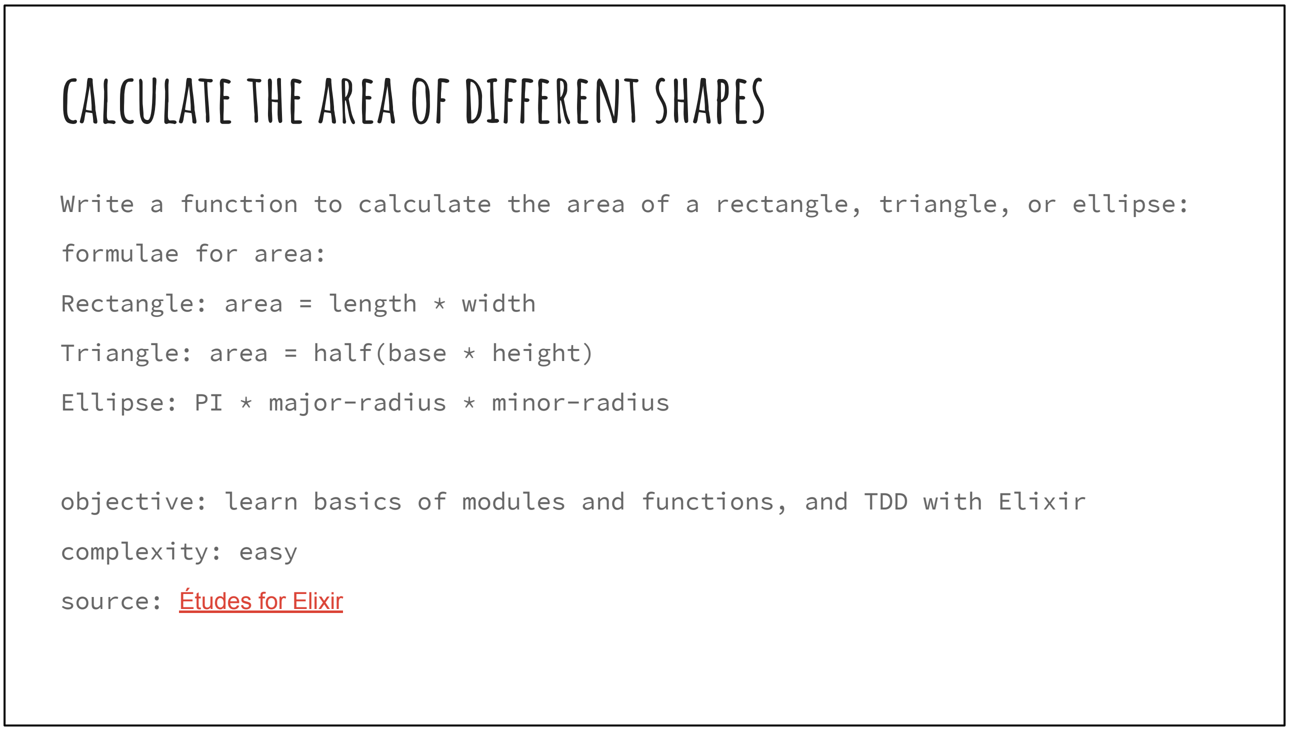


HIGHER-ORDER FUNCTIONS

Any functions that accept other functions as arguments, and may return functions

- map: apply a transformation to each element of a collection (more accurately, an “Enumerable” in elixir) to produce a new collection
- reduce: compute a single value, given a collection. Each computation on elements in the collection acts upon an “accumulator” (or “residue) available to it from the computation on the previous element, to produce the next value of the “accumulator”, and when every element has been evaluated, the final value of the “accumulator” is the outcome of the reduce operation.
- filter and predicates: select elements of the collection based upon the “predicate” to form elements of a new collection. A “predicate” is a function that evaluates to a true or false result.

PROBLEM SOLVING



CALCULATE THE AREA OF DIFFERENT SHAPES

Write a function to calculate the area of a rectangle, triangle, or ellipse:

formulae for area:

Rectangle: area = length * width

Triangle: area = half(base * height)

Ellipse: PI * major-radius * minor-radius

objective: learn basics of modules and functions, and TDD with Elixir

complexity: easy

source: Études for Elixir

- Use “doctest”

Binary boarding

source: Advent of Code, 2020, Day 5

“... Instead of zones or groups, this airline uses binary space partitioning to seat people. A seat might be specified like FBFBBFFRLR, where F means "front", B means "back", L means "left", and R means "right". ...” (see full problem description at the link above)

objective: use higher-order functions

complexity: moderate