Elm Workshop

Day 2

Ce facem azi?

1. Elm Test
2. JSON
3. Prânz
4. Architecture II

1. Elm Test

Setup

```
> npm install -g elm-test
> mkdir elm-workshop
> cd elm-workshop
> elm init
> elm-test init
> elm-test
```

Anatomy of a Test

```
describe : String → List Test → Test
test : String → (() → Expectation) →  Test
equal : a → a → Expectation
```

```
describe "suite description"
 [ test "test 1" (\_ → Expect.equal 2 (1 + 1))
 , test "test 2" (\_ → Expect.equal 0 (0 * 5))
 ]
```

Expectations

```
equal : a → a → Expectation
notEqual : a → a → Expectation
lessThan : comparable → comparable → Expectation
greaterThan : comparable → comparable → Expectation
true : String → Bool → Expectation
false : String → Bool → Expectation
equalLists : List a → List a → Expectation
pass : Expectation
fail : String → Expectation
```

Fuzz Test

```
fuzz : Fuzzer a → String → (a → Expectation) → Test
bool : Fuzzer Bool

fuzz bool "not" (\b → Expect.notEqual b (not b))
```

Property Testing for Addition

```
numberPair : Fuzzer (Int, Int)
numberPair = map2 Tuple.pair int int

describe "addition"
[ fuzz numberPair "commutativity"
  (\(x,y) → Expect.equal (x+y) (y+x))

, fuzz int "neutral element" (\x → Expect.equal (x+0) x)
]
```

2. JSON

Setup

```
> elm install elm/json
```

JSON Encoding

```
encode : Int → Value → String
string : String → Value
int : Int → Value
float : Float → Value
bool : Bool → Value
list : (a → Value) → List a → Value
object : List (String, Value) → Value
```

JSON Encoding

```elm
import Json.Encode as Encode exposing (Value)

type alias User = { name : String, age : Int }

encode : User → Value
encode user = Encode.object
  [ ("name", Encode.string user.name)
  , ("age", Encode.int user.age)
  ]
```

JSON Decoding

```
decodeString : Decoder a → String → Result Error a
decodeValue : Decoder a → Value → Result Error a
errorToString : Error → String

string : Decoder String
bool : Decoder Bool
int : Decoder Int
nullable : Decoder a → Decoder (Maybe a)
list : Decoder a → Decoder (List a)
field : String → Decoder a → Decoder a
map : (a → value) → Decoder a → Decoder value
map2 : (a → b → value) → Decoder a → Decoder b → Decoder value
```

JSON Decoding

```elm
import Json.Decode as Decode exposing (Decoder)

type alias User = { name : String, age : Int }

decode : Decoder User
decode =
  Decode.map2
    User
    (Decode.field "name" Decode.string)
    (Decode.field "age" Decode.int)
```

3. Prânz?

4. Architecture II

Thank you for attending!

Useful links

Elm packages
https://package.elm-lang.org/

Elm Search by Type
https://klaftertief.github.io/elm-search/

The Official Elm Guide
https://guide.elm-lang.org/