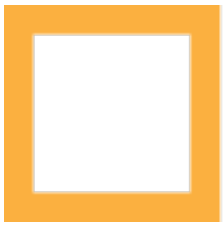
Elm Introduction
Functional Programming

Jens Egholm Pedersen and Anders Kalhauge



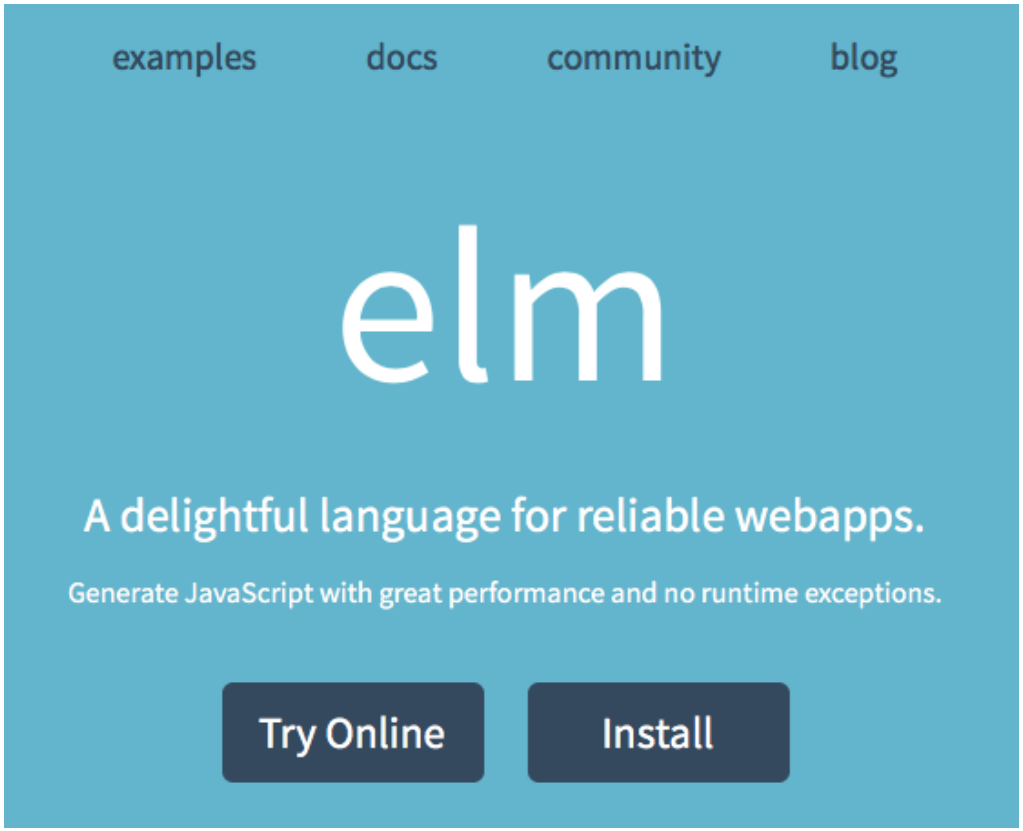
cphbusiness

Spring 2017

Outline

Elm website

Go to `http://elm-lang.org`
Select install



Elm command line tools

cphbusiness
COPENHAGEN BUSINESS ACADEMY

- `elm-repl` — play with Elm expressions
- `elm-reactor` - get a project going quickly
- `elm-make` - compile Elm code directly
- `elm-package` - download packages

Atom packages

cphbusiness
COPENHAGEN BUSINESS ACADEMY

language-elm 1.5.0 — 35,190

Syntax highlighting and autocompletion for the language Elm


edubkendo

Uninstall | Disable

linter-elm-make 0.22.5 — 27,383

Lint Elm code with elm-make


mybuddymichael

Uninstall | Disable

save-commands 0.6.11 — 2,929

Package for Atom. Assign parametrized shell commands to file globs to be automatically run whenever the file is saved


JsonHunt

Uninstall | Enable

Elm css packages

cphbusiness
COPENHAGEN BUSINESS ACADEMY

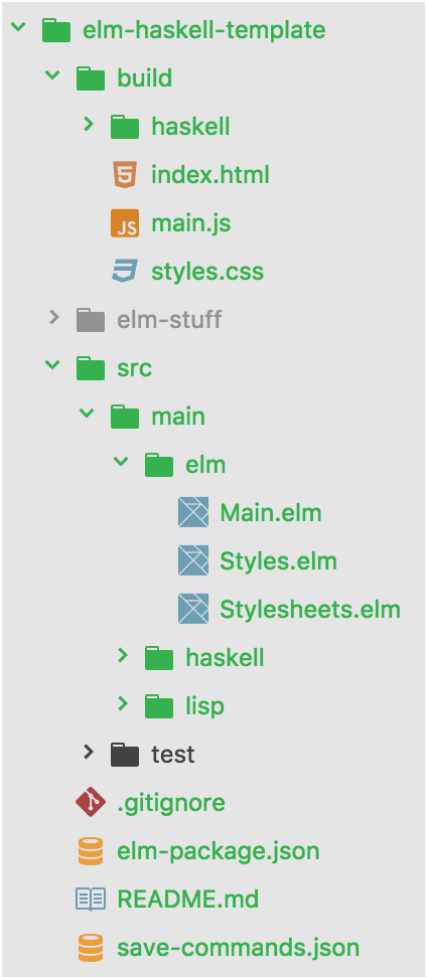
See: https://github.com/rtfeldman/elm-css

```
$ npm install -g elm-css
$ git clone https://github.com/rtfeldman/elm-css.git
$ cd elm-css/examples
$ elm-css src/Stylesheets.elm
$ less homepage.css

elm package install rtfeldman/elm-css-helpers
```

Project structure

Structure of the elm-haskell-template



index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
      <meta charset="UTF-8">
      <title>Title</title>
      <link rel="stylesheet"
        href="styles.css"
        type="text/css">
      <script src="main.js"></script>
  </head>
  <body>
    <script type="text/javascript">
      Elm.Main.fullscreen()
    </script>
  </body>
</html>
```

save-commands Script

- src/main/elm/**/*.elm

```
elm make src/main/elm/Main.elm
-output=build/main.js
```

- src/main/elm/Styles.elm

```
elm-css src/main/elm/Stylesheets.elm -output build
```

- src/main/haskell/**/*.hs

```
ghc -make src/main/haskell/Main.hs -o
build/haskell/main
```

Values

- Strings
  - `"Hello"`
  - `"Hello"++" "++"World!"` is `"Hello World!"`
- Numbers
  - `7`
  - `22.67`
  - `2 + 3 * 4` is `14`
  - `9/2` is `4.5`
  - `9//2` is `4`

Functions

```
> isNegative n = n < 0
<function>

> isNegative 4
False

> isNegative -7
True

> isNegative (-3 * -4)
False
```

Selections


```
> if True then "hello" else "world"
"hello"

> if False then "hello" else "world"
"world"
```

Lists


```
> names = [ "Alice", "Bob", "Chuck" ]
["Alice","Bob","Chuck"]

> List.isEmpty names
False

> List.length names
3

> List.reverse names
["Chuck","Bob","Alice"]
```

Lists

cphbusiness
COPENHAGEN BUSINESS ACADEMY

```
> numbers = [1,4,3,2]
[1,4,3,2]

> List.sort numbers
[1,2,3,4]

> double n = n * 2
<function>

> List.map double numbers
[2,8,6,4]
```

Tuples


```
> import String

> goodName name = \
|    if String.length name <= 20 then \
|      (True, "name␣accepted!") \
|    else \
|      (False, "name␣was␣too␣long")

> goodName "Tom"
(True, "name␣accepted!")
```

Records


```
> point = { x = 3, y = 4 }
{ x = 3, y = 4 }

> point.x
3

> bill = { name = "Gates", age = 57 }
{ age = 57, name = "Gates" }

> bill.name
"Gates"
```

Records


```
> .name bill
"Gates"

> List.map .name [bill,bill,bill]
["Gates","Gates","Gates"]

> { bill | name = "Nye" }
{ age = 57, name = "Nye" }

> { bill | age = 22 }
{ age = 22, name = "Gates" }
```

Records

```
> under70 {age} = age < 70
<function>

> under70 bill
True

> under70 { species = "Triceratops", age = 68000000 }
False
```

Primitives and Lists

cphbusiness
COPENHAGEN BUSINESS ACADEMY

```
> "hello"
"hello" : String

> not True
False : Bool

> round 3.1415
3 : Int
```

```
> [ "Alice", "Bob" ]
[ "Alice", "Bob" ] : List String

> [ 1.0, 8.6, 42.1 ]
[ 1.0, 8.6, 42.1 ] : List Float

> []
[] : List a
```

Functions

```
> import String
> String.length
<function> : String -> Int

> String.length "Supercalifragilisticexpialidocious"
34 : Int

> String.length [1,2,3]
-- error!

> String.length True
-- error!
```

Lambdas - anonymous functions

cphbusiness
COPENHAGEN BUSINESS ACADEMY

```
> \n -> n / 2
<function> : Float -> Float

> (\n -> n / 2) 128
64 : Float

> oneHundredAndTwentyEight = 128.0
128 : Float

> half = \n -> n / 2
<function> : Float -> Float

> half oneHundredAndTwentyEight
64 : Float

> half n = n / 2
<function> : Float -> Float
```

Functions with more arguments

cphbusiness
COPENHAGEN BUSINESS ACADEMY

```
> divide x y = x / y
<function> : Float -> Float -> Float

> divide 3 2
1.5 : Float

> divide x = \y -> x / y
<function> : Float -> Float -> Float

> divide = \x -> (\y -> x / y)
<function> : Float -> Float -> Float
```

Functions - again

```
divide 3 2
```

Functions - again

```
divide 3 2
```

```
(divide 3) 2                    -- 1: Implicit parentheses
```

Functions - again

```
divide 3 2
```

```
(divide 3) 2                    -- 1: Implicit parentheses
```

```
((\x -> (\y -> x / y)) 3) 2 -- 2: Expand 'divide'
```

Functions - again

```
divide 3 2
```

```
(divide 3) 2                        -- 1: Implicit parentheses
```

```
((\x -> (\y -> x / y)) 3) 2 -- 2: Expand 'divide'
```

```
(\y -> 3 / y) 2                     -- 3: Replace x with 3
```

Functions - again

```
divide 3 2
```

```
(divide 3) 2                          -- 1: Implicit parentheses
```

```
((\x -> (\y -> x / y)) 3) 2           -- 2: Expand 'divide'
```

```
(\y -> 3 / y) 2                       -- 3: Replace x with 3
```

```
3 / 2                                 -- 4: Replace y with 2
```

Functions - again

```
divide 3 2
```

```
(divide 3) 2                          -- 1: Implicit parentheses
```

```
((\x -> (\y -> x / y)) 3) 2           -- 2: Expand 'divide'
```

```
(\y -> 3 / y) 2                       -- 3: Replace x with 3
```

```
3 / 2                                 -- 4: Replace y with 2
```

```
1.5                                   -- 5: Do the math
```

Exercise 1 - Fibonacci numbers

$$fib(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ fib(n-1) + fib(n-2) & n > 1 \end{cases}$$

- Define a function recursively that calculates the fibonacci number of n

Exercise 1 - Fibonacci numbers

cphbusiness
COPENHAGEN BUSINESS ACADEMY

$$fib(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ fib(n-1) + fib(n-2) & n > 1 \end{cases}$$

- Define a function recursively that calculates the fibonacci number of n
- Define an **effective** recursive function for the problem

Type annotations

cphbusiness
COPENHAGEN BUSINESS ACADEMY

```
half : Float -> Float
half n =
  n / 2

divide : Float -> Float -> Float
divide x y =
  x / y

askVegeta : Int -> String
askVegeta powerLevel =
  if powerLevel > 9000 then
    "It's over 9000!!!"
  else
    "It is " ++ toString powerLevel ++ "."
```