Lab 05 - Intro To Elm

CS 1XA3

Feb 5th, 2018

Elm - What the hell is it?

Let's start with a small overview of Web Development

Front End

- Run by a browser
- **Scripting:** JavaSript
- **MarkUp:** HTML, CSS

Back End

- Run by a server
- **Scripting:** PHP, Python, etc
- **Databases:** SQL, MongoDB, etc

Elm - What the hell is it?

Elm replaces traditional front-end development:

- compiles to Javascript, HTML and CSS. Providing a unified language for devlopment
- competes with projects like React, Angular as a better way to javascript
- functional, built to resemble haskell, be simple and get great performance (`http://elm-lang.org/blog/blazing-fast-html`)

Note: we won't concern ourselves with back-end development in this course

Elm Tools

You should have installed the following tools with the Elm-Platform (`https://github.com/elm-lang/elm-platform`)

- elm repl: an interpreter similar to ghci
- elm reactor: an interactive development tool that simulates your elm code running on a local server
- elm make: build tool, use it to generate corresponding JavaScript / HTML
- elm package: the elm package manager, use it to install new packages (`http://package.elm-lang.org/`)

Note: make sure you have version **0.18.0** of elm installed with the command elm –version

Elm Package

- Installs packages locally (i.e only in the project directory you use it)
- Puts package information in elm-package.json
- Use the following command to install elm-lang/html at version 2.0.0

```
elm package install elm-lang/html 2.0.0
```

Use http://package.elm-lang.org/ to find packages to install

Elm REPL

Read-Eval-Print-Loop: similar to the python or haskell interpreters your familiar to.

```
module Sample exposing (..)

fac : Int -> Int
fac n = if n > 1
        then n * fac (n-1)
        else 1
```

Save the above code as Sample.elm, open the elm-repl and enter import Sample.elm

Elm Reactor

Example Usage:

- Save the following code as Hello.elm

```elm
import Html exposing (text)

main =
  text "Hello, World!"
```

- Execute elm reactor in the same directory
- Open your browser and go to http://localhost:8000
- Click on Hello.elm
- Use **Ctrl-c** to stop the reactor

Elm Make

- Used to *compile* your elm code to html or javascript
- Use the following to compile Hello.elm to Hello.html

```
elm-make Hello.elm --output Hello.html
```

- Open Hello.html to view in your browser locally

Other Useful Elm Tools

You may want to make use of the following online tools (or build their equivalents locally)

- Html to Elm: `https://mbylstra.github.io/html-to-elm/`
- Try Elm Online Editor: `http://elm-lang.org/try`
- Online Elm *Time Travel* Debugger: `http://debug.elm-lang.org/`

Elm Modules

- Defining Modules

```
module ModName exposing (..)
-- expose everything
module ModName exposing (fun1,fun2)
-- expose only fun1, fun2
```

- Importing Modules

```
import ModName exposing (..)
-- don't do this
import ModName exposing (fun1,fun2)
-- if you just want a few things
import ModName as MN
-- if you want alot of stuff
```

Using Qualified Imports

Most of the time, your going to want to use the as keyword when importing. This is good conventional in Haskell, but particularly important in Elm due to the absence of typeclasses

Example

```
import List as LS

sum xs = LS.foldr (+) 0 xs
```

Differences In Elm and Haskell

Swapped usage of **:** and **::**

```
-- Haskell
someFun :: SomeType1 -> SomeType2

-- Elm - Uses single :
someFun : SomeType1 -> SomeType2

-- Haskell List
xs :: [Int]
xs = 1 : (2 : 3 : [])

-- Elm List
xs : List Int
xs = 1 :: (2 :: 3 :: [])
```

Differences In Elm and Haskell

- No type classes
- No where clauses, only let-in
- The data keyword is the type keyword
- The type keyword is type alias
- No gaurds or pattern matching (outside of case)

Learn By Example

Try the examples on the elm website (`http://elm-lang.org/examples`)

Try running them in various ways on your own machines, i.e elm reactor,elm make, elm repl