ELM IS COMING

@KRISAJENKINS

WHAT'S HARD?

FRONTEND PROGRAMMING

- Mirror the backend
- *plus* Error-handling
- *plus* Users
- *plus* Marketing
- *plus* Everyone else
- *plus* Constantly in flux
- *plus* Demands are on the rise

AND…

Our tools suck.

ELM IS…

- Functional
- A variant of Haskell
- (Written in Haskell)
- Compiles to JavaScript
- Easy to learn
- Structurally simple

ELM HAS…

- Friendly static typing
- Pure rendering
- One-way data-flow
- Immutable data
- Pure functions
- Control over *side-effects*

ELM ALSO HAS..

- Fast build tool
- Package manager
- Semver enforcement
- And cool stuff

OVERVIEW OF AN ELM APP

TWO DATATYPES, TWO FUNCTIONS

DATATYPE 1: MODEL

```
type Model = Model
  {username : String
  ,password : String
  ,serverError : Maybe Http.Error}
```

DATATYPE 2: ACTION

```
type Action
  = ChangeUsername String
  | ChangePassword String
  | Submit
  | LoginResponse (Result Error AuthToken)
```

FUNCTION 1: UPDATE

```
update : Action -> Model -> Model
```

EXAMPLE

```
update : Action -> Model -> Model
update action model =
  case action of
    ...
    ChangeUsername s -> {model | username <- s}
    ...
```

BUT…

Sometimes we need to schedule future actions.

FUNCTION 1: UPDATE (V2)

```
update : Action -> Model -> Model
```

...becomes:

```
update : Action -> Model -> (Model, Effects Action)
```

EXAMPLE

```
update : Action -> Model -> (Model, Effects Action)
update action model =
  case action of
    ...
    Submit -> (model
              ,postForm "/login"
                        model.username
                        model.password)
    ...


postForm : ... -> Effects Action
postForm ... = LoginResponse ( ... )
```

ENHANCED EXAMPLE

```
update : Action -> Model -> (Model, Effects Action)
update action model =
  case action of
    ...
    Submit -> ({model | loading <- True}
              ,postForm "/login"
                        model.username
                        model.password)
    ...
```

FUNCTION 2: RENDERING

```
view : Model -> Html
```

EXAMPLE

```
loginForm : Model -> Html
loginForm model =
  form []
       [input [type' "text"
              ,class "form-control"
              ,autofocus True]
              []
       [input [type' "password"
              ,class "form-control"]
              []
       ,button [class "btn btn-primary"
               ,type' "button"
               ,disabled (model.username == "" ||
                          model.password == "")]
               [text "Log In"]]
```

BUT…

An HTML UI is an event source.

FUNCTION 2: RENDERING (V2)

```
view : Model -> Html
```

…becomes:

```
view : Address Action -> Model -> Html
```

EXAMPLE

```
loginForm : Address Action -> Model -> Html
loginForm address model =
  form []
       [input [type' "text"
              ,class "form-control"
              ,onChange address Username
              ,autofocus True]
              []
       [input [type' "password"
              ,class "form-control"
              ,onChange address Password]
              []
       ,button [class "btn btn-primary"
               ,type' "button"
               ,disabled (model.username == "" ||
                          model.password == "")
               ,onClick address Submit]
               [text "Log In"]]
```

ELM ARCHITECTURE

```
type Model

type Action

update : Action -> Model -> (Model, Effects Action)

view : Address Action -> Model -> Html
```

SIMPLE DEMO

COMPARE TO MVC

Elm	**MVC**
Model	Model
Action	-
View	View
Update	Controller

HUGE STRUCTURAL DIFFERENCE

WHEN IT'S SIMPLE

```
M --- C --- V
```

MVC AS IT GROWS

```
   M --- C --- V
              / \
             /   \
            /     \
           /       \
          /         \
         /           \
M - C - V     M - C - V
                     / \
                    /   \
                   /     \
                  /       \
                 /         \
                /           \
       M - C - V     M - C - V
```

HERE'S THE PROBLEM

Simple:

```
view : Address Action -> Model -> Html
```

Grows to

```
view : Address Action -> Model -> Everything
```

HERE'S THE SOLUTION

```
     M
    / \
   /   o
  m   / \
     m   m

     C
    / \
   /   o
  c   / \
     c   c

     V
    / \
   /   \
  v     v
         \
          \
           v

M --- C --- V
```

DEMOS

PARSING

HERE'S SOME JSON

```
{
    spatialReference: {
        wkid: 4326,
        latestWkid: 4326
    },
    candidates: [
        {
            address: "Royal Festival Hall",
            location: {
                x: -0.11599726799954624,
                y: 51.50532882800047
            },
            score: 100,
            attributes: { },
            extent: {
                xmin: -0.120998,
                ymin: 51.500329,
                xmax: -0.110998,
                ymax: 51.510329
            }
        }
    ]
}
```

TO ENTYPIFY THE JSON

DEFINE A PLACE

```elm
type alias Place =
  {address: String
  ,latitude: Float
  ,longitude: Float}
```

DECODE THE LIST OF PLACES

```
decodePlaces : Decoder (List Candidate)
decodePlaces = "candidates" := (list decodePlace)
```

DECODE ONE PLACE

```elm
decodePlace : Decoder Place
decodePlace =
  Place `map`    ("address" := string)
        `apply` (at ["location", "x"] float)
        `apply` (at ["location", "y"] float)
```

DONE

EVENT-TRACKING ANALYTICS

DEFINE AN ANALYTICS EVENT

```
type alias AnalyticsEvent =
  {category : String
  ,action : String}
```

GENERATE ACTIONS

```
toAnalyticsEvent : Action -> Maybe AnalyticsEvent
toAnalyticsEvent action =
  case action of
    BuyProduct id           -> Just {category = "Buy",   action = "P
roduct"}
    ShareProduct Twitter id -> Just {category = "Share", action = "T
witter"}
    ...
    _                       -> Nothing
```

GENERATE AN EFFECT

```
toAnalyticsEffect : Action -> Effects Action
toAnalyticsEffect action =
  case toAnalyticsEvent action of
    Nothing -> none
    Just event -> sendEvent AnalyticsSent event
```

AUGMENT OUR UPDATE FUNCTION

```
updateWithAnalytics : Action -> Model -> (Model, Effects Action)
updateWithAnalytics action model =
  let (newModel,newFx) = update action model
  in (newModel, batch [newFx, toAnalyticsEffect action])
```

DONE

LINKS

Beeline

http://krisajenkins.github.io/beeline-demo/

Blog

http://blog.jenkster.com/

Sewing Browser

http://www.getstitching.com/

Lunar Lander Game

http://krisajenkins.github.io/lunarlander

Learn!

http://www.meetup.com/West-London-Hack-Night/