Elm

Voyage en Terre Inconnue - Episode 2

zendesk

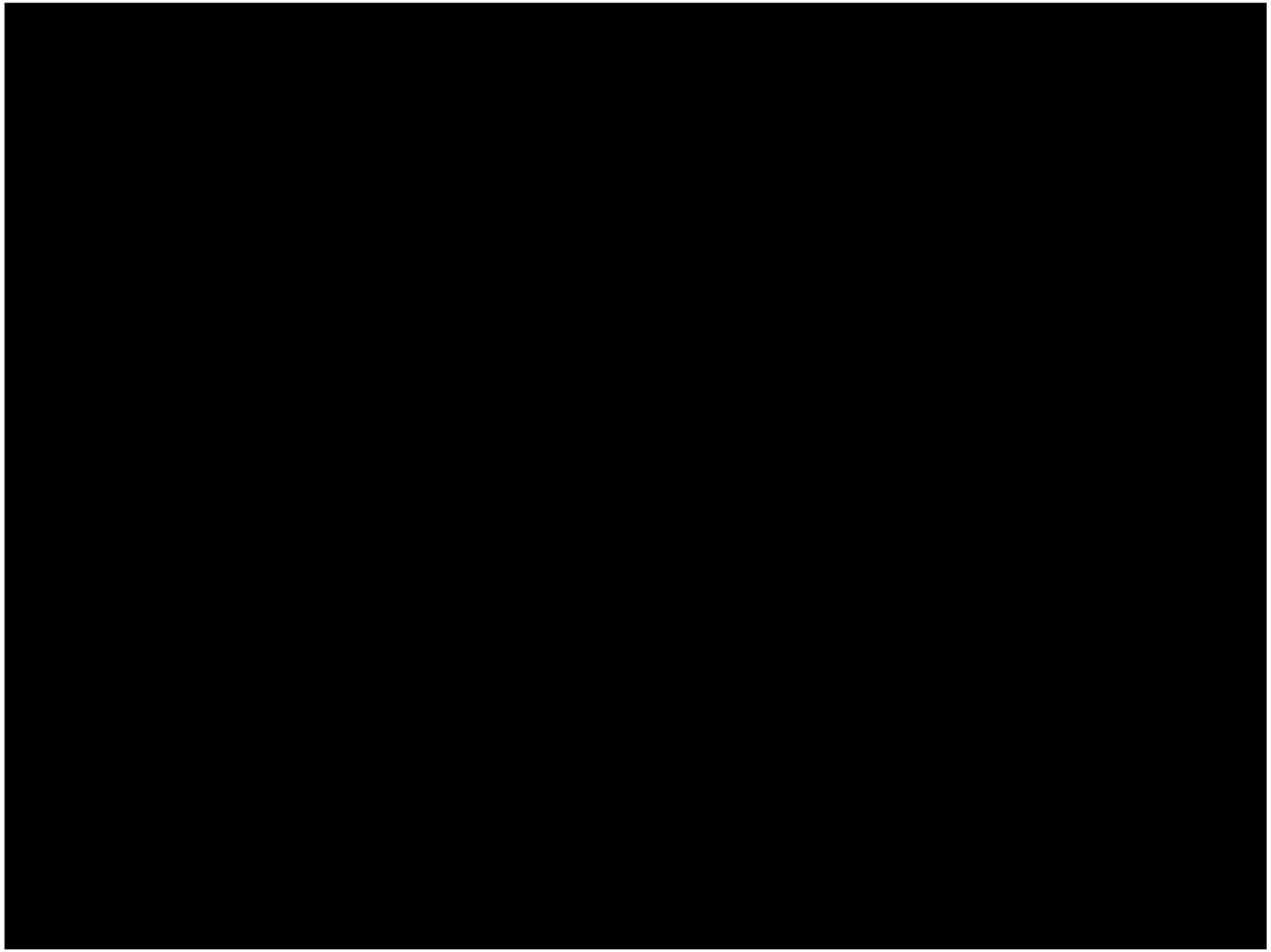
Functional Meetup Montpellier #2
23 Octobre 2017 - Tabmo

Romain Endelin

*A long time ago, in a **village** far, far away...*

Conway's Game of life

Disclaimer: Le pire code Elm que vous verrez jamais



Mais alors, Elm c'est quoi?

- Créé en *2012*
- BDFL: *Evan Czaplicki*
- Version actuelle: *0.18*

- Utilisation: *Front-End Web*
  - Transpile en JavaScript

- Paradigmes:
  - *Fonctionnel*
  - *Reactive*

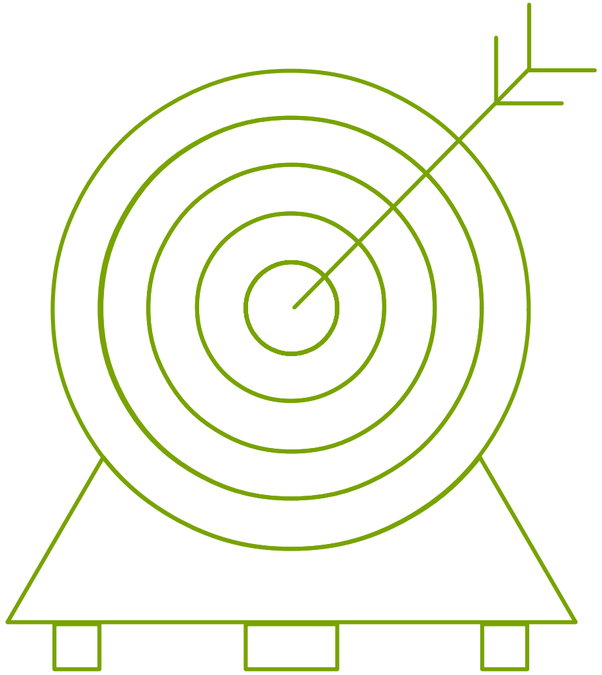
Et pourquoi c'est si bien?

Elm, je t'aime!

Si ça compile, ça marche

Si ça compile, ça marche

- *"You can definitely mess with Elm if you want to, but the languages makes it very hard for you to do so"*
- Pas de *Null Pointer Exceptions (NPE)*
- Typage fort
- On ne touche pas au DOM, compris?
- Pas de problèmes de concurrences
- Tout est immutable, TOUT!
- Pas de récursion infinie



Un exemple

```
type Operation -- Union Type
    = ToggleDone
    | SetName String -- Type with parameter


type alias Item = -- Type alias, storing records
    { id : Int
    , name : String
    , done : Bool
    }


modifyItem : Operation → Item → Item -- arg1 → arg2 → returnTypes
modifyItem operation item =
    case operation of -- Will not compile unless we handle all possible cases
        ToggleDone → -- Updating a record
            { item | done = (not item.done) }

        SetName name →
            { item | name = name }


create : Item
create =
    { id = 1, name = "my item", done = False }
        ▷ modifyItem ToggleDone
        ▷ modifyItem (SetName "my new name")
-- Returns { id =1, name = "my new name", done = True}
```

Si ça ne compile pas…

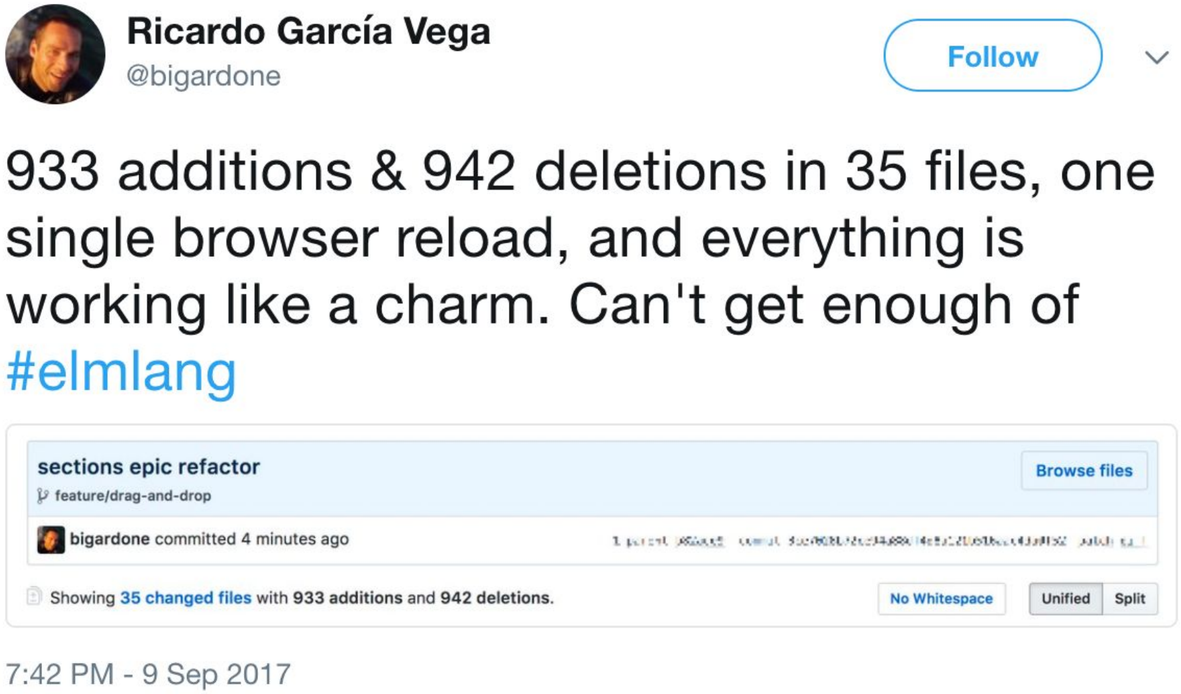
Le compilateur est votre meilleur ami

Si ça ne compile pas...

- *"Le compilateur de Elm, c'est comme avoir un programmeur expert derrière votre épaule, qui vous prévient gentiment chaque vous que vous perdez le cap"*
- Le compilateur est formidable 🤩

```
Your dependencies form a cycle:


   ┌─────┐
   │    View
   │     ↓
   │    Main
   └─────┘


You may need to move some values to a new module to get rid of the cycle.
```

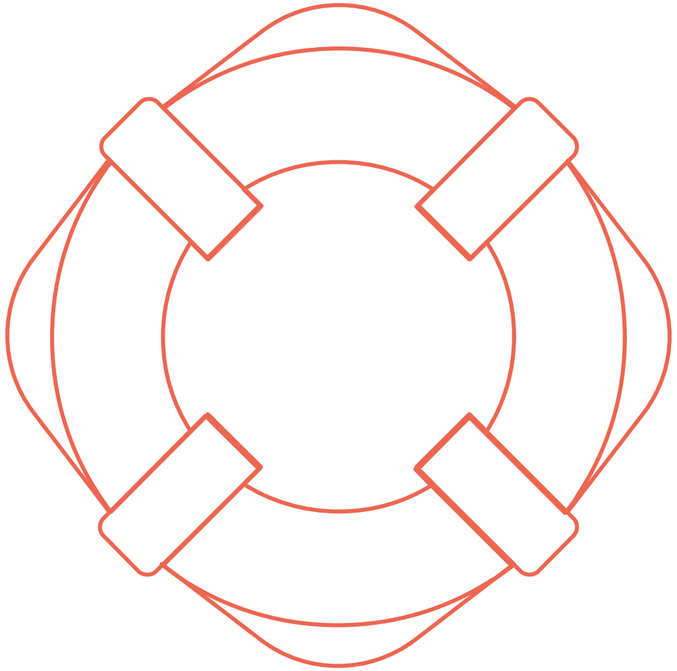
Refactorer du code Elm



Simple et efficace

Ouvert aux débutants

- On ne parle pas de monades!
- Beaucoup plus facile d'accès que Haskell
- Bonne documentation
- Le compilateur, *encore et toujours*
- Facile de comprendre ses erreurs, grâce aux types



Pas de tooling complexe

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

- create-elm-app (https://github.com/halfzebra/create-elm-app) :
  $ create-elm-app my_app
  $ elm-app start
  $ elm-app build
  $ elm-app eject

Auto-formatter

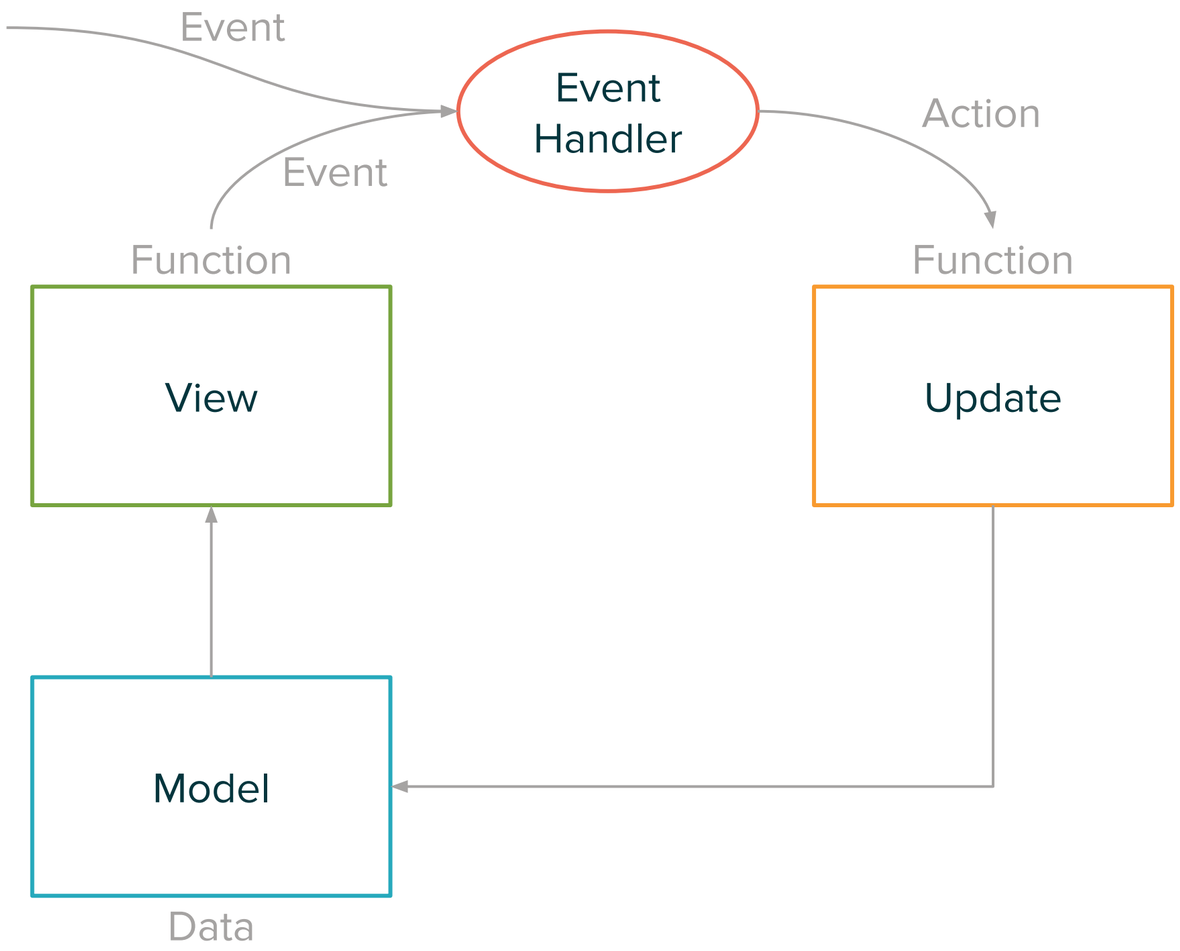
```
main : Program Never Model Msg

main =
    Html.program
        { init = init , view = view
        , update = update
        , subscriptions = subscriptions
        }



grid_width:Int
grid_width =
    100
grid_height : Int
grid_height = 60
frequency : Float
frequency =100 * millisecond


cell_size :String
cell_size =
    "5px"
```

```
main : Program Never Model Msg
main =
    Html.program
        { init = init
        , view = view
        , update = update
        , subscriptions = subscriptions
        }



grid_width : Int
grid_width =
    100


grid_height : Int
grid_height =
    60


frequency : Float
frequency =
    100 * millisecond


cell_size : String
cell_size =
    "5px"
```

Bonus: Upgrade automatique de Elm (v0.17 -> v0.18)

L'architecture de Elm

L'architecture de Elm



L'architecture - Côté code

```elm
main : Program Never Model Msg
main =
    Html.program
        { view = view
        , init = init
        , update = update
        , subscriptions = always Sub.none
        }
```

```elm
view : Model → Html Msg
view model =
    div [] (List.map model.items)


viewItem : Item → Html Msg
viewItem item =
    div
        [ onMouseEnter (Select item)
        , onClick (SetDone (not item.done) item)
        ]
        [ text item.name ]
```

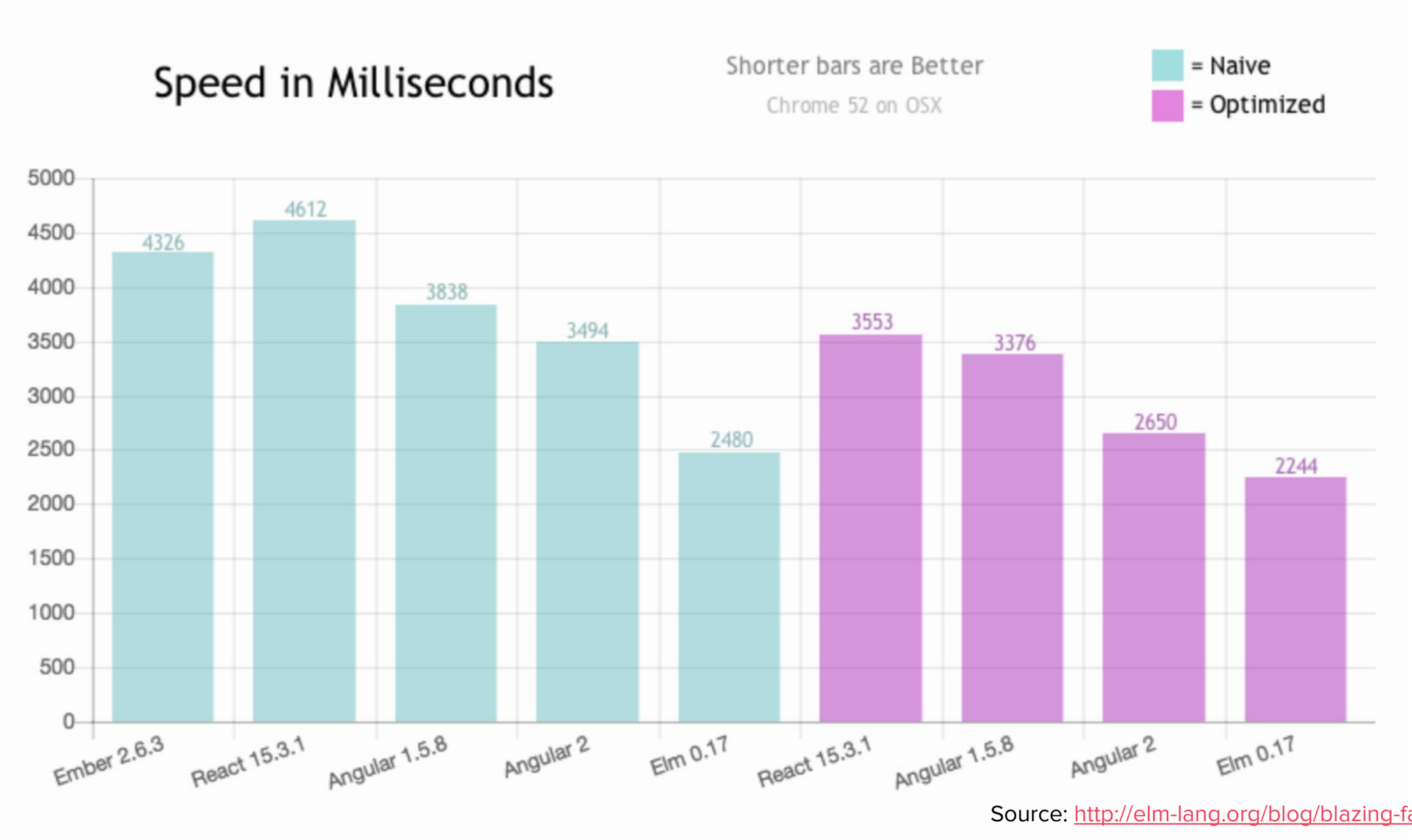
```elm
type Msg
    | SetDone Bool Item


update : Msg → Model → ( Model, Cmd Msg )
update msg model =
    case msg of
        SetDone isDone item →
            ( { model
              | items =
                  (replaceIf ((==) item) { item | done = isDone } model.items)
              }
            , Cmd.none
            )
```

```elm
type alias Model =
    { items : List Item
    }


init : ( Model, Cmd Msg )
init =
    ( { items =
            [ { id = 1, name = "Item1", done = False }
            , { id = 2, name = "Item2", done = False }
            ]
      }
    , Cmd.none
    )
```

Les performances

Les performances

Ça va vite!



Source: http://elm-lang.org/blog/blazing-fast-html-round-two

La communauté

La communauté

- Elm News (http://elm-news.com/)
- Reddit (https://www.reddit.com/r/elm/)
- Slack (https://elmlang.herokuapp.com/ - https://elmlang.slack.com)

L'écosystème

Les restes du monde

Elm Package Manager

- Environ 750 packages
- Killer feature: Semantic Versioning

```
------ Changes to module Basics - MINOR ------

    Added:
        type Never


------ Changes to module Date - MINOR ------

    Added:
        now : Task.Task x Date.Date


------ Changes to module Debug - MAJOR ------

    Removed:
        trace : String -> Graphics.Collage.Form -> Graphics.Collage.Form
        watch : String -> a -> a
        watchSummary : String -> (a -> b) -> a -> a


------ Changes to module Dict - MINOR ------

    Added:
        merge : (comparable -> a -> result -> result) -> (comparable -> a -> b -> result -> result) -> (comparable -> b -> result -> result) -> Dict.Dict comparable a -> Dict.Dict
 comparable b -> result -> result


------ Changes to module Random - MAJOR ------

    Added:
        step : Random.Generator a -> Random.Seed -> (a, Random.Seed)

    Changed:
      - generate : Random.Generator a -> Random.Seed -> (a, Random.Seed)
      + generate : (a -> msg) -> Random.Generator a -> Platform.Cmd.Cmd msg



------ Changes to module Task - MAJOR ------

    Added:
        type alias Task err ok = Platform.Task err ok
        perform : (x -> msg) -> (a -> msg) -> Task.Task x a -> Platform.Cmd.Cmd msg

    Removed:
        type Task x a
        type ThreadID
        sleep : Task.Time -> Task.Task x ()
        spawn : Task.Task x a -> Task.Task y Task.ThreadID
```

Interopérabilité avec Javascript

On place Javascript en quarantaine


```
port module Spelling exposing (..)

...

-- port for sending strings out to JavaScript
port check : String -> Cmd msg

-- port for listening for suggestions from JavaScript
port suggestions : (List String -> msg) -> Sub msg

...
```

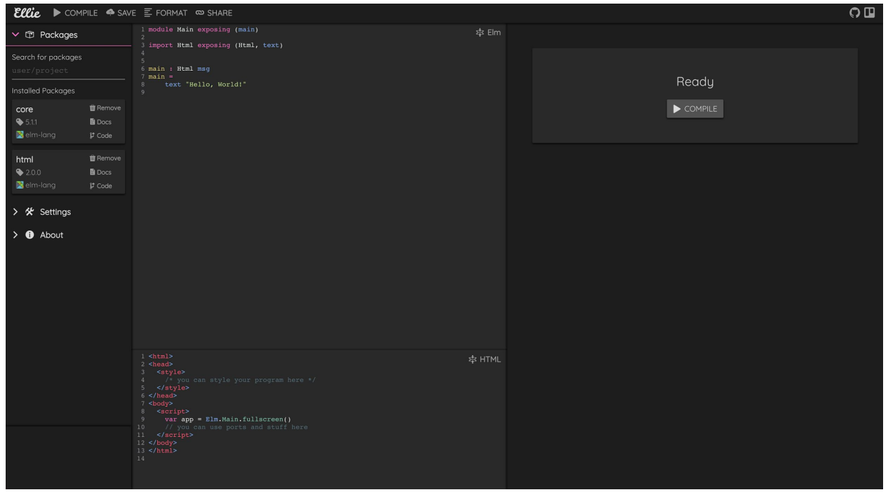
Pourquoi l'utiliser ?

Et quand ne pas l'utiliser ?

Pourquoi l'utiliser ?

- Pour le fun!
- Pour découvrir la programmation fonctionnelle
- Pour faire du Front-End sans se salir les mains sur Javascript
- Pour rejoindre une très bonne communauté
- Pour en finir avec les Null-Pointer Exceptions

Essayez par vous-mêmes!



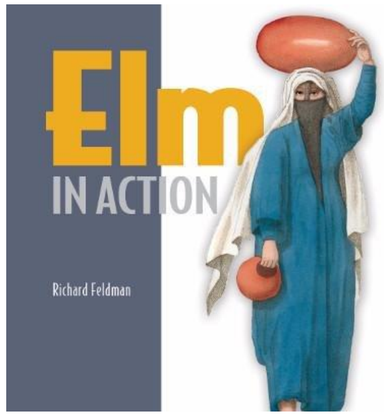
Ellie App, pour essayer sans rien installer

En entreprise:

- Commencez par écrire un composant modeste en Elm.
- Pas de réécriture complète pour commencer!

Quelques ressources:

- La documentation officielle
- Elm SPA Example, par Richard Feldman (https://github.com/rtfeldman/elm-spa-example)
- Elm in Action, par Richard Feldman (early access)

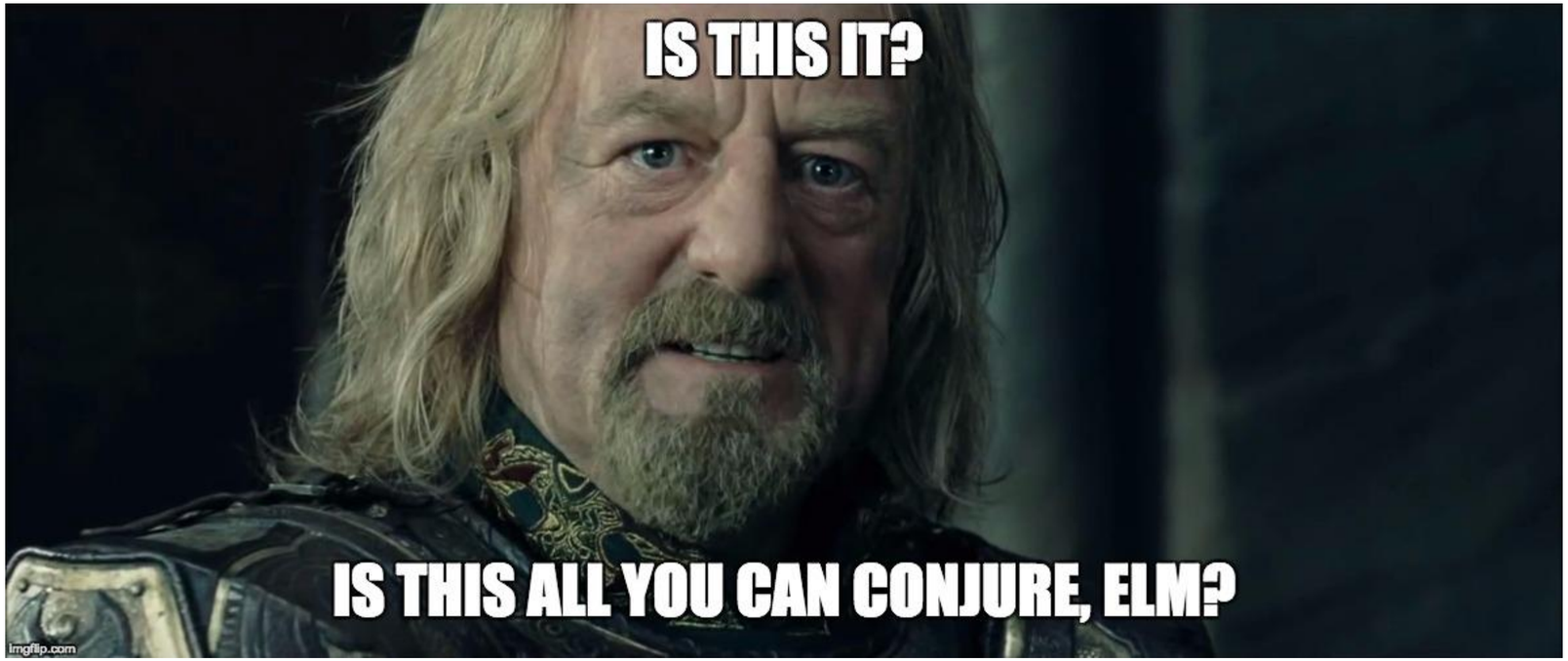


Quand ne pas l'utiliser?

- Si vous utilisez beaucoup de librairies Javascript
- Si vous souhaitez un langage stable
- Si vous êtes déjà un expert en Haskell
- Si vous souhaitez faire du développement mobile
- Si vous souhaitez utiliser Elm en Back-End

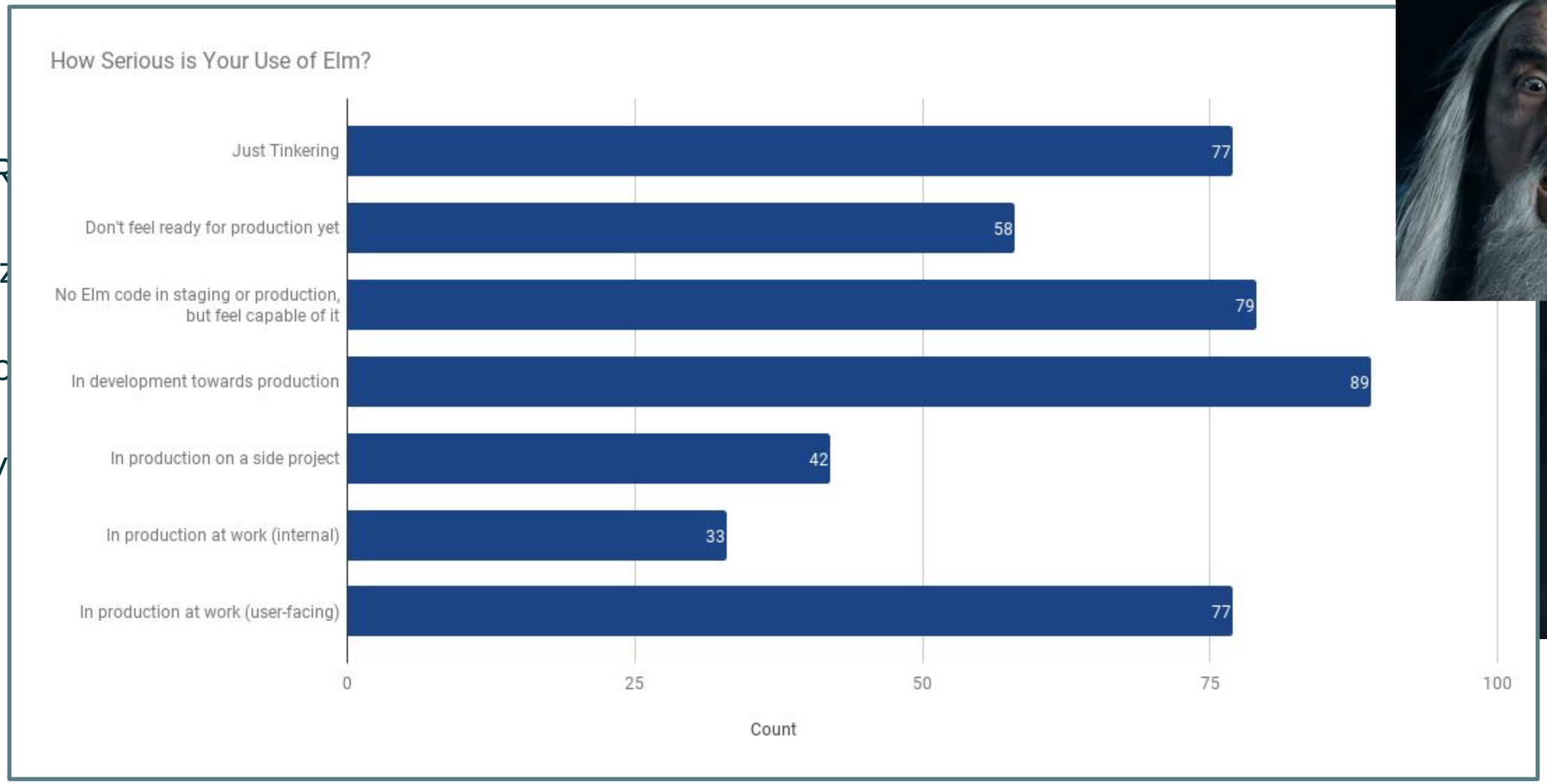
Quelques entreprises qui développent en Elm

- NoRedInk
- Prezi
- Pivotal Tracker
- Day One



Quelques entreprises qui développent en Elm

- NoR
- Prez
- Pivo
- Day



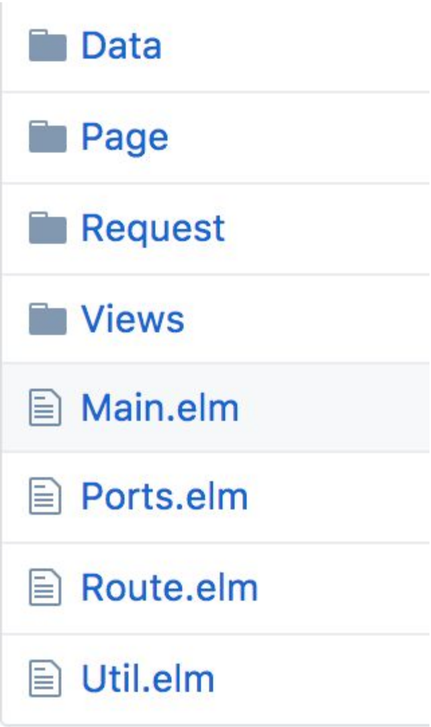
Source: https://www.brianthicks.com/post/2017/07/27/state-of-elm-2017-results/

L'avenir

L'avenir - Elm 0.19

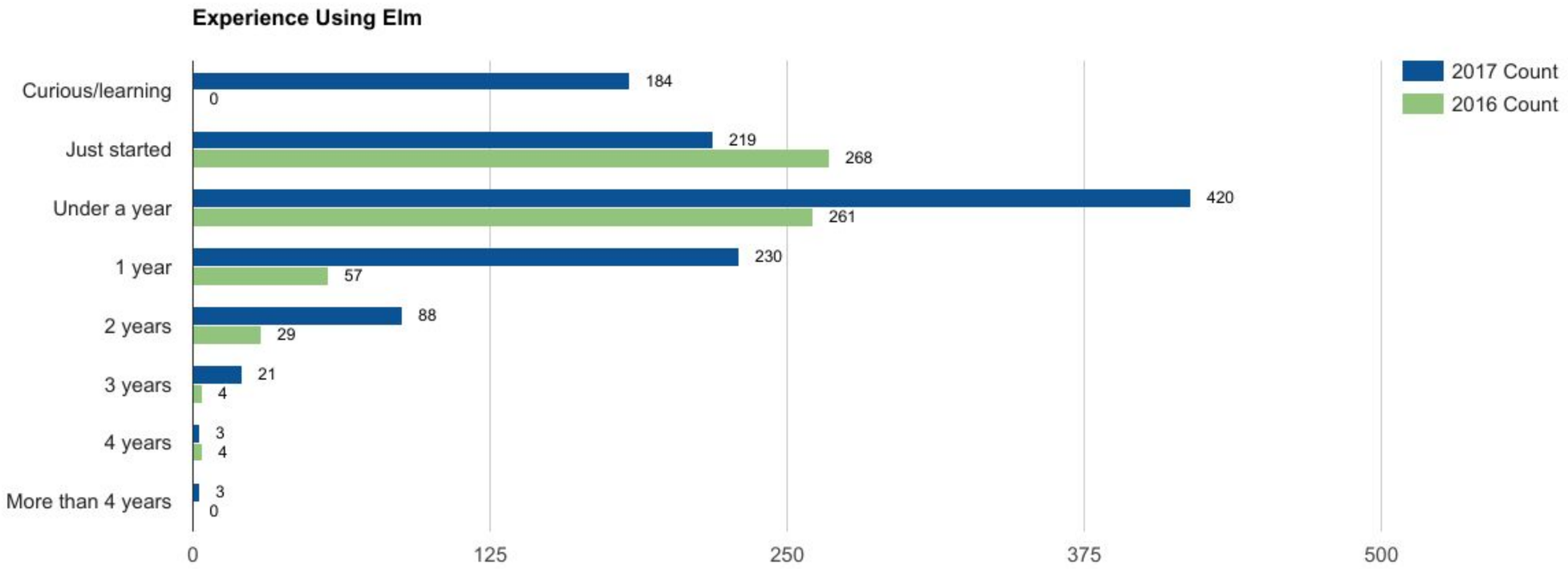
Elm 0.19: Asset Management

- Séparation en plusieurs modules
- Navigation par URL / Routing



- Dead-Code Elimination
- Lazy Loading
- Server-Side Rendering
- Routing

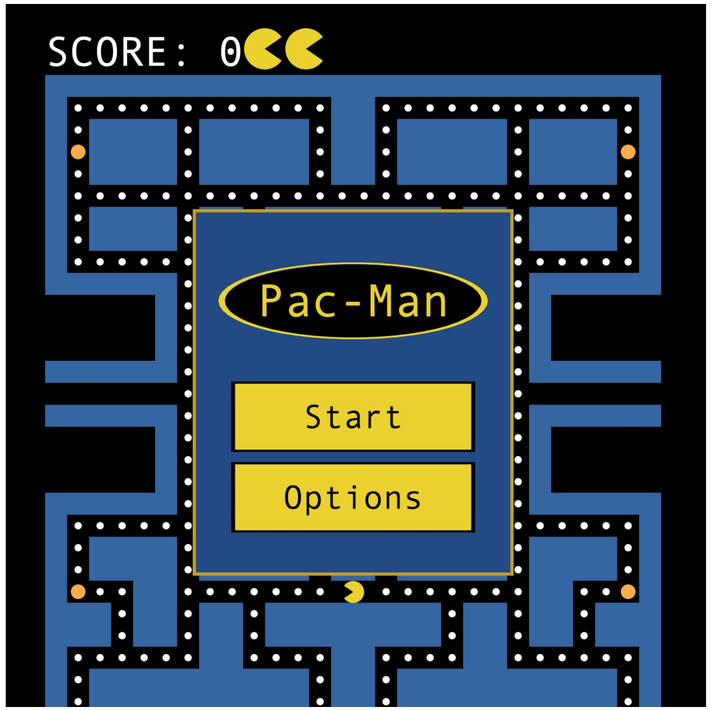
La communauté

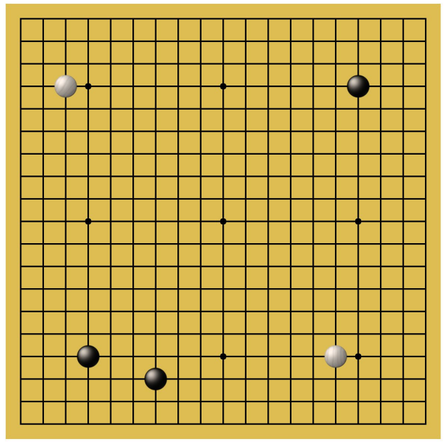


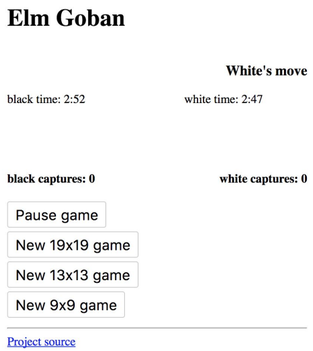
Source: https://www.brianthicks.com/post/2017/07/27/state-of-elm-2017-results/

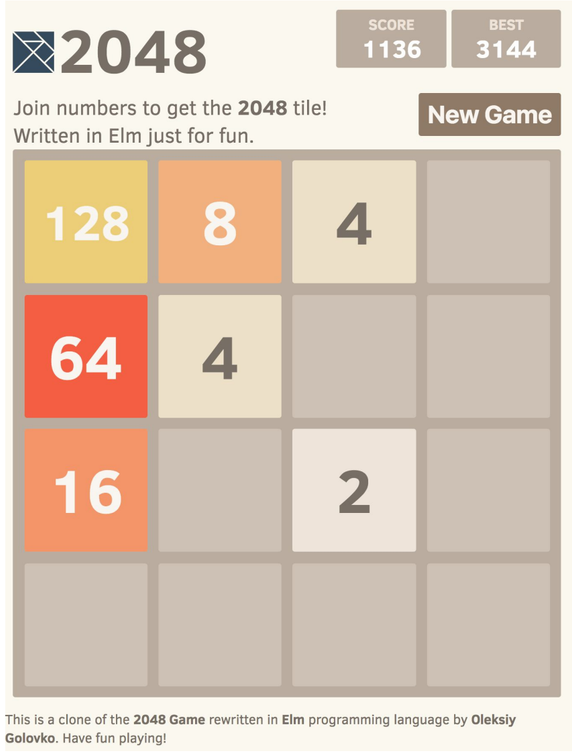
Pour finir... Quelques exemples!

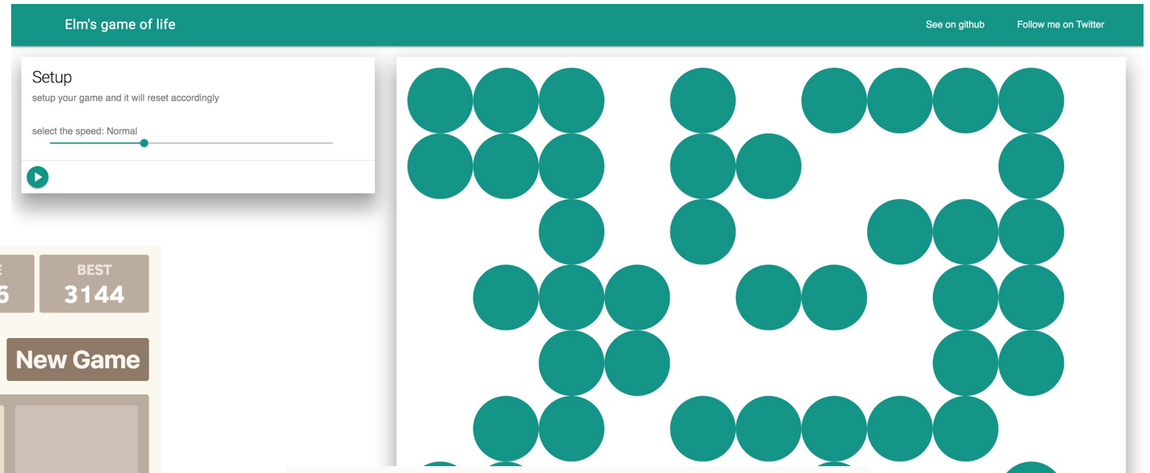
Jeux

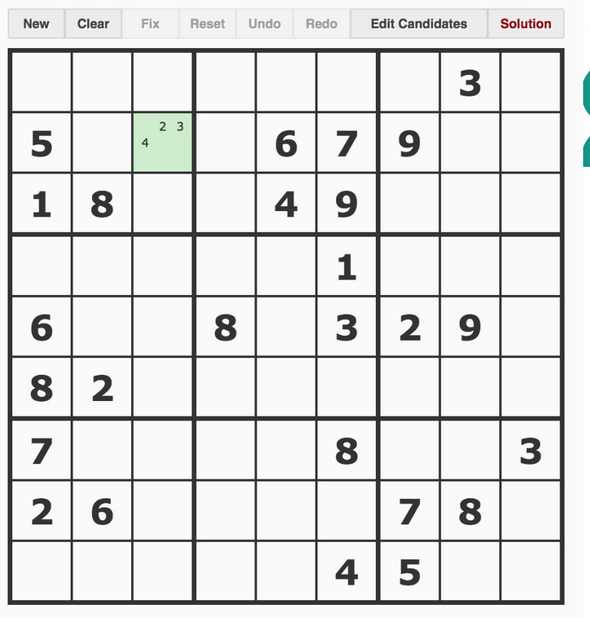

SCORE: 0
Pac-Man
Start
Options



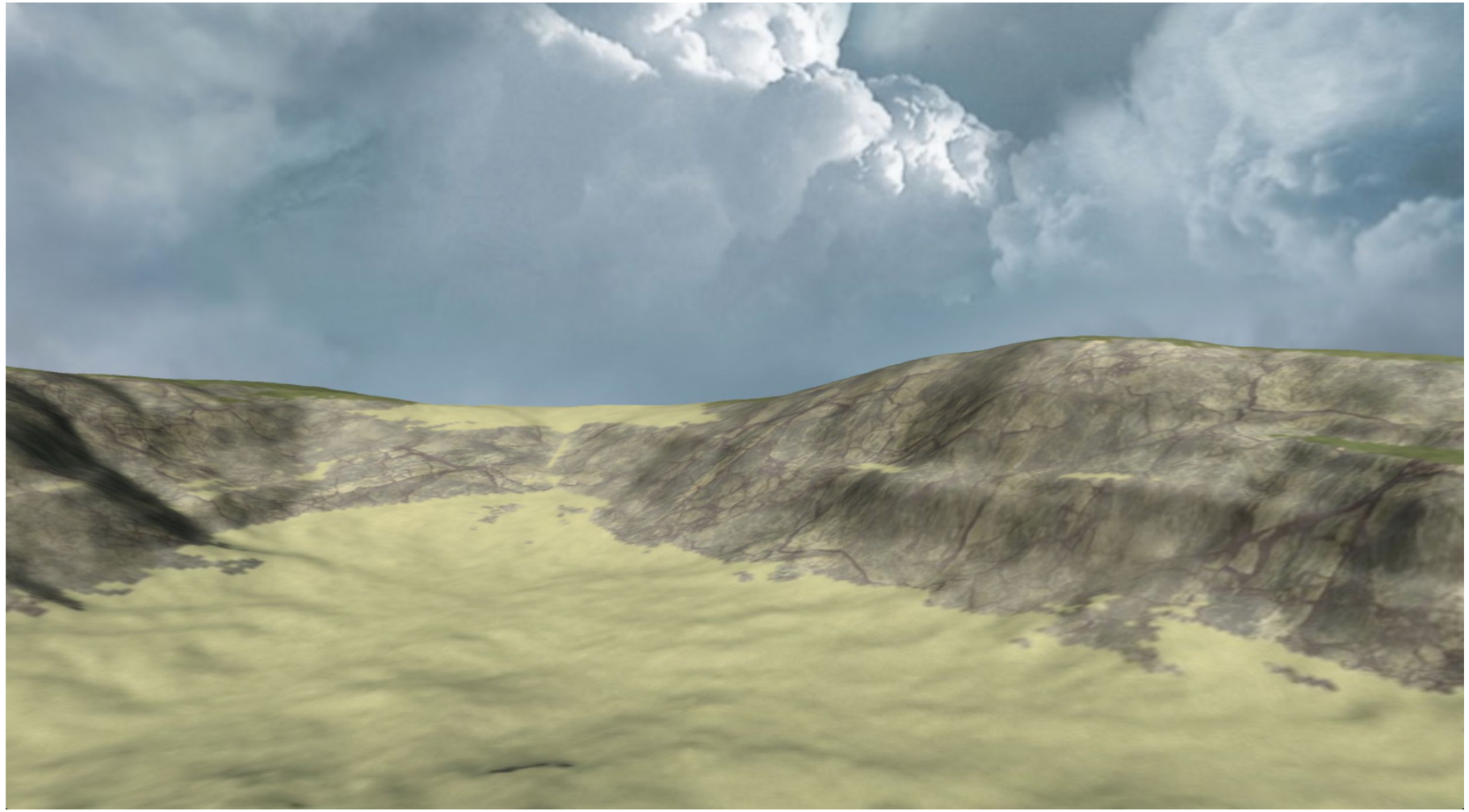

Elm Goban
White's move
black time: 2:52
white time: 2:47
black captures: 0
white captures: 0
Pause game
New 19x19 game
New 13x13 game
New 9x9 game
Project source


2048
SCORE
1136
BEST
3144
Join numbers to get the 2048 tile!
Written in Elm just for fun.
New Game
128
8
4
64
4
16
2
This is a clone of the 2048 Game rewritten in Elm programming language by Oleksiy Golovko. Have fun playing!

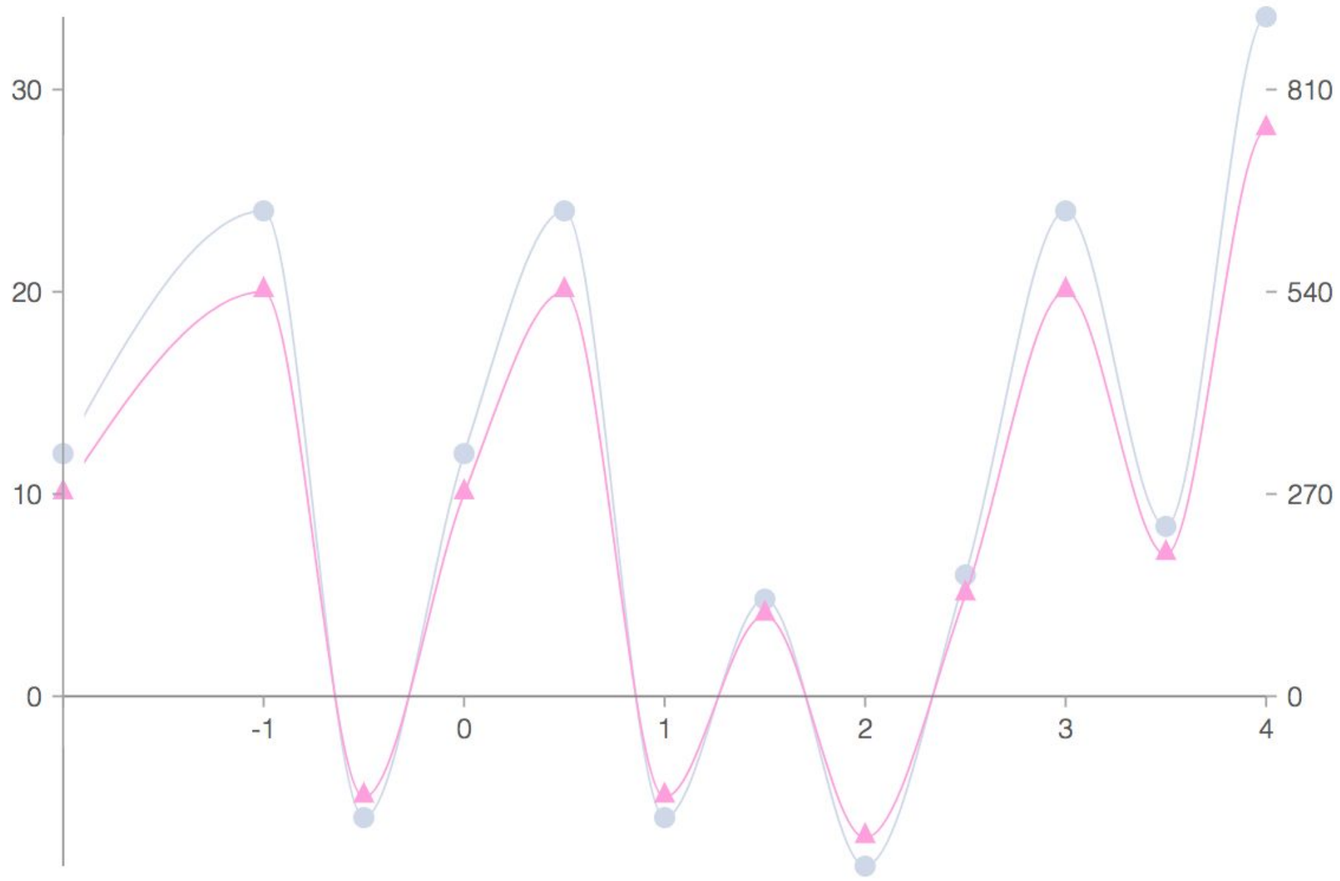

Elm's game of life
See on github
Follow me on Twitter
Setup
setup your game and it will reset accordingly
select the speed: Normal


New
Clear
Fix
Reset
Undo
Redo
Edit Candidates
Solution

WebGL - Génération de Terrain

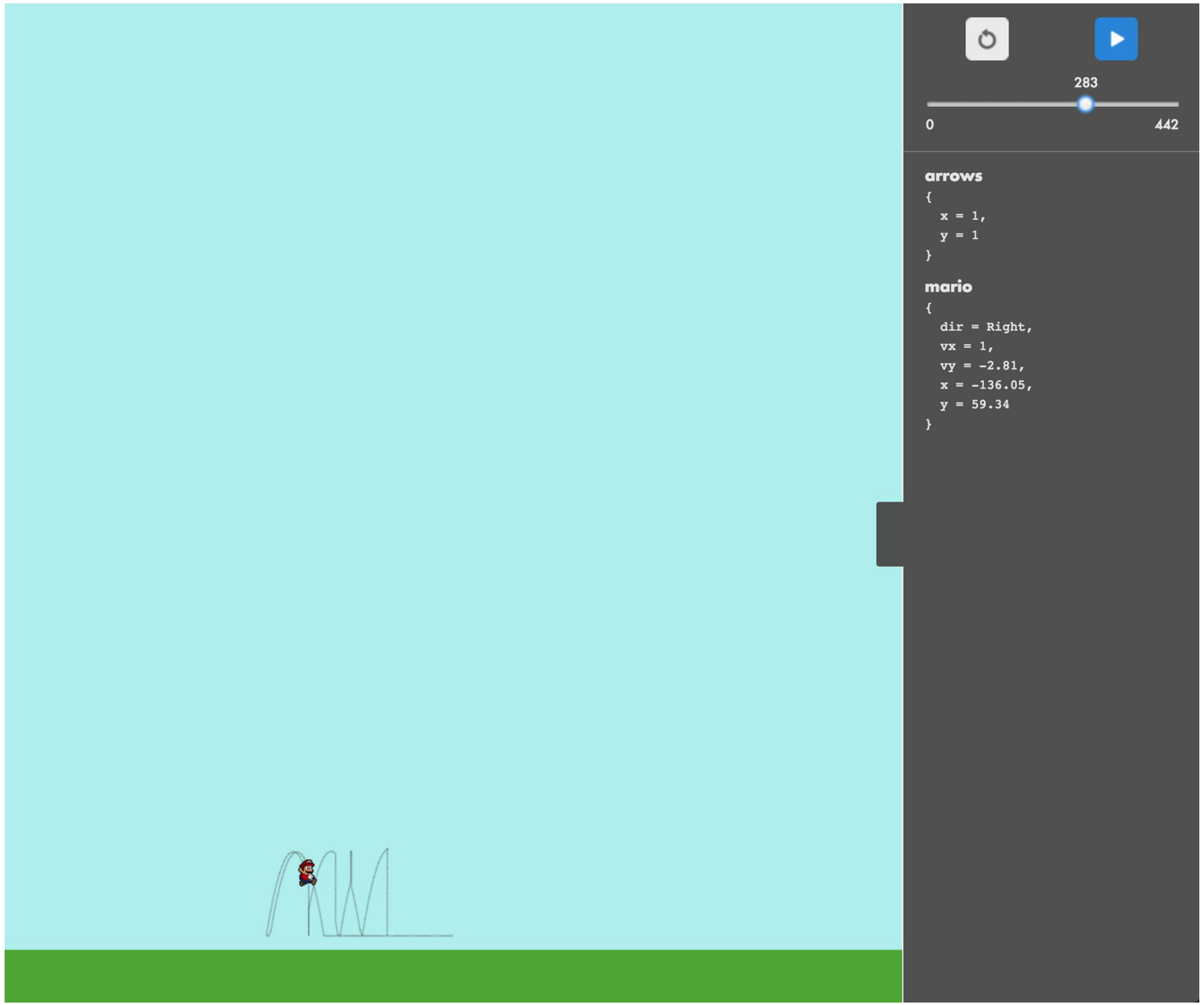


Dataviz - Elm Plot

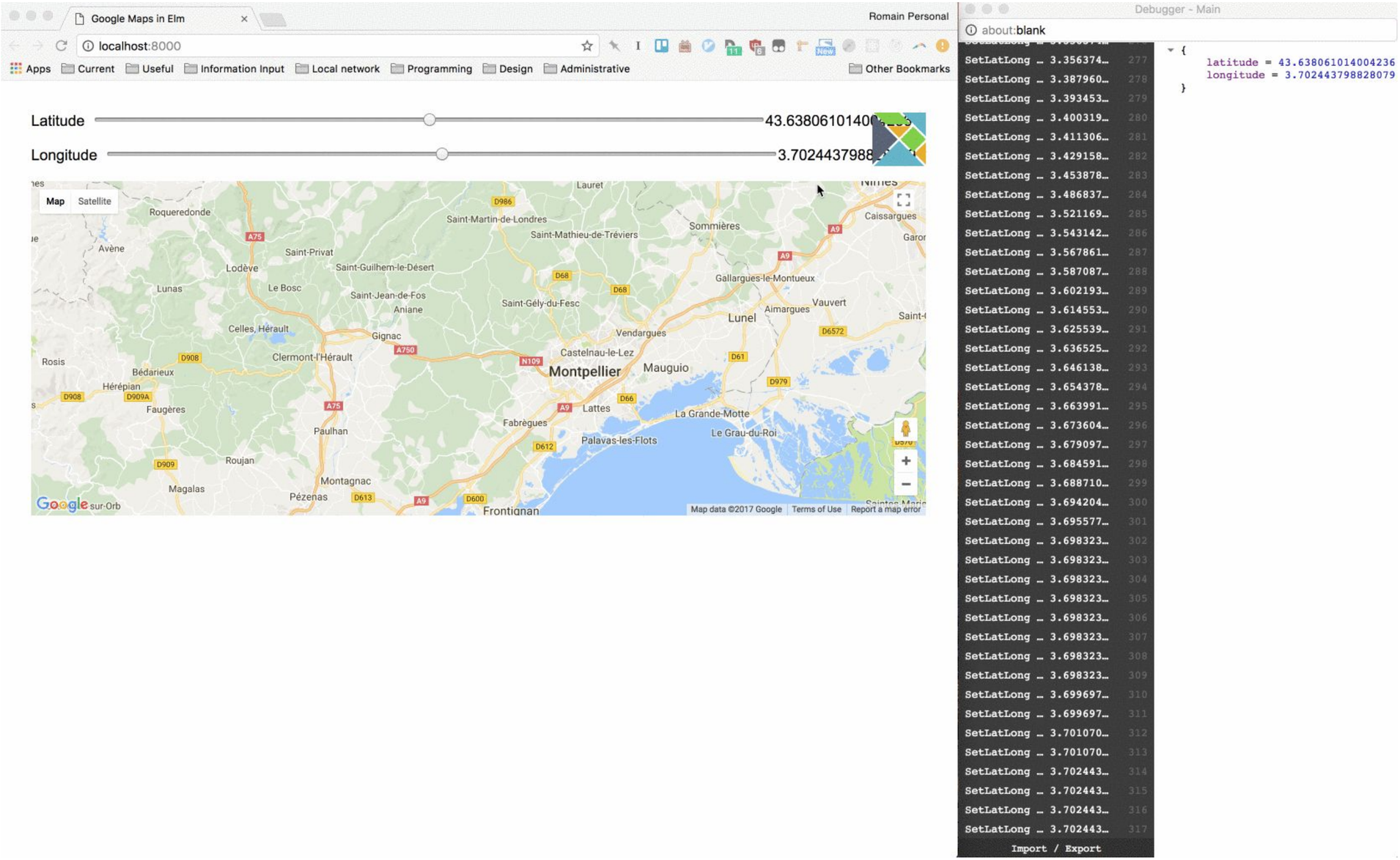



```
customArea : Series (List ( Float, Float )) msg
customArea =
  { axis = rightAxis
  , interpolation = Monotone Nothing [ stroke pinkStroke ]
  , toDataPoints = List.map (\( x, y ) -> triangle x y)
  }


customLine : Series (List ( Float, Float )) msg
customLine =
  { axis = axisAtMin
  , interpolation = Monotone Nothing [ stroke blueStroke ]
  , toDataPoints = List.map blueCircle
  }


blueCircle : ( Float, Float ) -> DataPoint msg
blueCircle ( x, y ) =
  dot (viewCircle 5 blueStroke) x (y * 1.2)


rightAxis : Axis
rightAxis =
  customAxis <| \summary ->
    { position = Basics.max
    , axisLine = Nothing
```

Mario - Time-Travel Debugger



Google Map - Time-Travel debugger



Conclusion

Elm, c'est bien!

Questions?