Elm

Origem

- Evan Czapplicki
- 2012
- Programção funcional
- Desenvolvimento Web
- Haskell, Standart-ML e OCalm

Classificação

- Funcional pura
- Tipagem forte
- Estrutura de dados persistente

Por que Elm ?

- Melhor experiência de desenvolvimento
- Escalável
- Sem erros em tempo de execução
- “Elm Archtecture”

Compilador inteligente

```elm
import Html exposing (..)


view users =
  div [] (List.nap viewUser users)


viewUser user =
  span [] [text user.name]

```

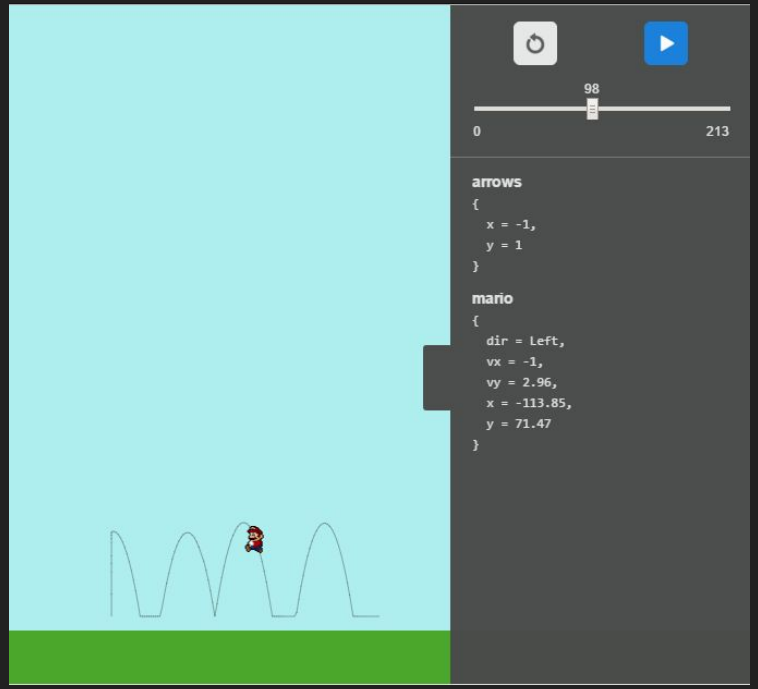
```
-- NAMING ERROR ------------------------------------- naming/unknown-name.elm

Cannot find variable `List.nap`.

6|   div [] (List.nap viewUser users)
           ^^^^^^^^

`List` does not expose `nap`. Maybe you want one of the following?

    List.map
    List.any
    List.map2
    List.map3
```

```elm
import Html exposing (..)
import Html.Attributes exposing (..)


alice =
  img [src "/users/alice/pic"] []


bob =
  img [src "/users/bob/pic"] []


userPics =
  [ alice, bob, "/users/chuck/pic" ]

```

```
-- TYPE MISMATCH ------------------------------------------ types/list.elm

The 3rd element of this list is an unexpected type of value.

15|   [ alice, bob, "/users/chuck/pic" ]
                  ^^^^^^^^^^^^^^^^^^

All elements should be the same type of value so that we can iterate over the
list without running into unexpected values.

As I infer the type of values flowing through your program, I see a conflict
between these two types:

    Html

    String
```

Por que Elm?

- Time travel debugger
- Gerenciador de pacotes
- Editor Online
- REPL
- Interpolação com HTML, CSS e JS



Avaliação Comparativa

```
// Javascript + HTML
<!DOCTYPE html>
<html><head><title>TODO list</title></head>
<body>
    <h3>A fazer</h3>
    <input placeholder="Todo" id="taskBox">
    <h3>Feito</h3>
    <ul id="list">

    </ul>
    <button onclick="addTask()">Add Message</button>
    <script>
        var doc = document;
        function addTask(){
            var text = doc.getElementById("input").value;
            var textNode = doc.createTextNode(text);
            var newElement = doc.createElement("li");
            newElement.append(textNode);
            doc.getElementById("list").appendChild(newElement);
        }
    </script>
</body></html>
```

```
main =
  Html.beginnerProgram { model = model, view = view, update = update }

-- MODEL
type alias Model = {
  to_do_list: List String,
  taskInput: String
}

model : Model
model = Model [] ""

-- UPDATE
type Msg = ListenChange String | AddTask

update : Msg -> Model -> Model
update msg model =
  case msg of
    ListenChange taskBoxContent ->  {model| taskInput = taskBoxContent }
    AddTask -> {model| to_do_list = List.append model.to_do_list [model.taskInput]}

-- VIEW
render_to_do lst=
    List.map (\to_do -> li [] [text to_do]) lst

view : Model -> Html Msg
view model =
  div []
    [
      h3 [] [ text "A fazer"]
    , input [ placeholder "Todo" ,id "taskBox", onInput ListenChange] []
    , h3 [] [ text "Feito"]
    , ul [id "list"] (render_to_do model.to_do_list)
    , button [ onClick AddTask] [text "Add Message"]
    ]
```

Conclusão