pra que te quero?

RESPIRE

JS

JS

TIPOS

Preço total: NaN

Bem-vindo, null!

Undefined is not a function

REFATORAR

Depende de testes unitários

Testes manuais, caçando runtime errors

EFEITOS COLATERAIS

MUTABILIDADE

Difícil evitar problemas

ImmutableJS não é compatível com tudo


elm

TIPOS

```
1                  Int

"a"                String

[1, 2, 3]          List Int

{ name = "Hugo" }  { name : String }
```

TIPOS

```
List.length "abc"
```

A função `List.length` esperava uma `List a`, não uma `String`

TIPOS

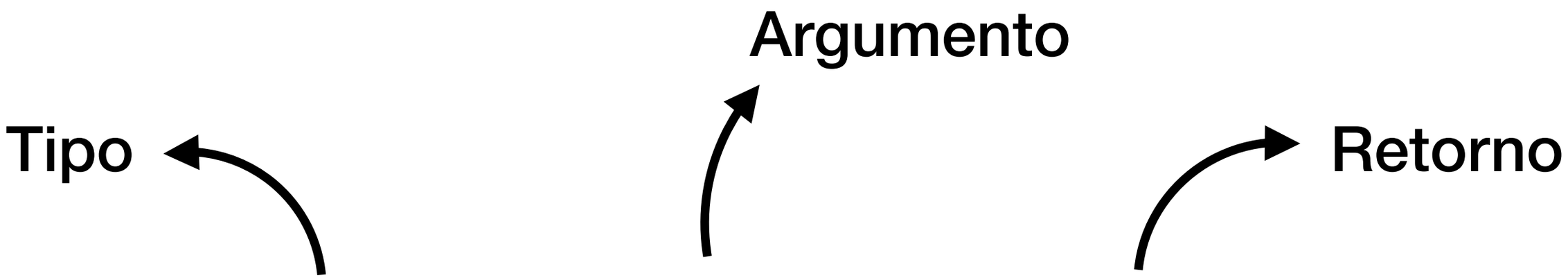
```
String.joni "," ["a", "b"]
```

Você não quis dizer `String.join`?

FUNÇÕES

```
greet user =
    "Hey, " ++ user.name
```

FUNÇÕES




```
greet : User -> String
greet user =
    "Hey, " ++ user.name
```

TIPOS E FUNÇÕES

```
User -> String
User -> Int -> Int
Int -> Float
```

O *último* é sempre
o tipo do *retorno*!

UNDEFINED?

```
Maybe String
= Just String
| Nothing
```

Existe

Não definido

MAYBE!

```
Maybe User
  = Just User
  | Nothing
```

Carregou

Não carregou

MAYBE!

```
case user of
    Just user -> user.name
    Nothing -> "Guest"
```

Não vou te deixar esquecer do `Nothing`!

MAYBE!

```
head : List Int -> Maybe Int
```

```
head [ ] → Nothing

head [ 1, 2 ] → Just 1
```

IMUTABILIDADE

```
list = [ 1, 2, 3 ]
```

Esse valor mudará em algum momento?

IMUTABILIDADE

```
newList = List.map toString list
```

Novo valor

Nunca, te prometo!

PURO

Todas as funções são puras

Efeitos colaterais são declarados

Click

Canvas

Input

WebSockets

Scroll

HTTP Request

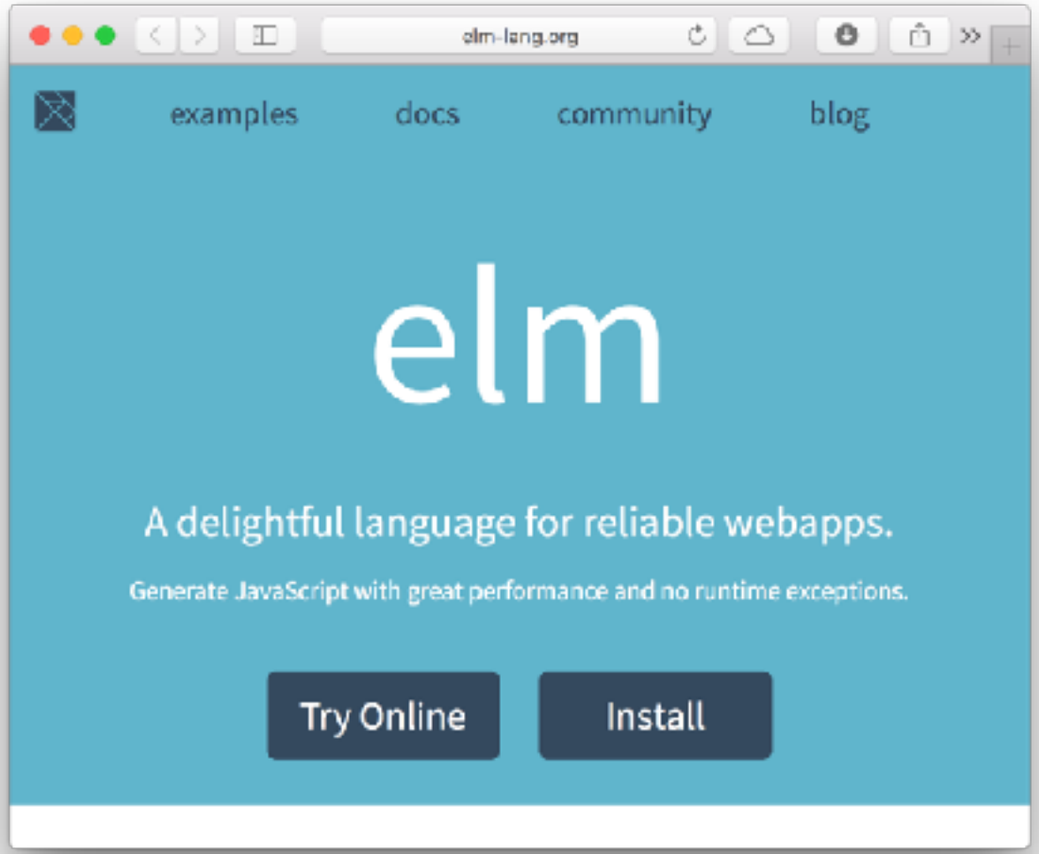
WebGL

DOM

Input Click WebSockets Canvas

HTTP Request WebGL Scroll DOM

ELM RUNTIME



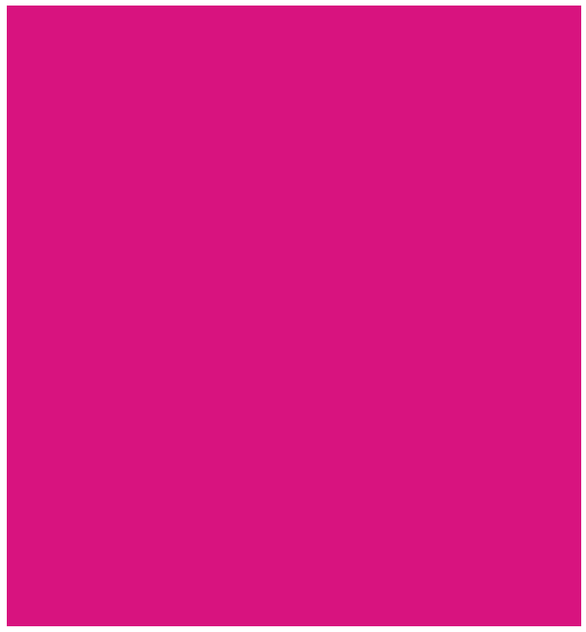
ELM ARCHITECTURE

Arquitetura padrão

Simples forma de dividir seu webapp

ELM ARCHITECTURE

Model



MODEL

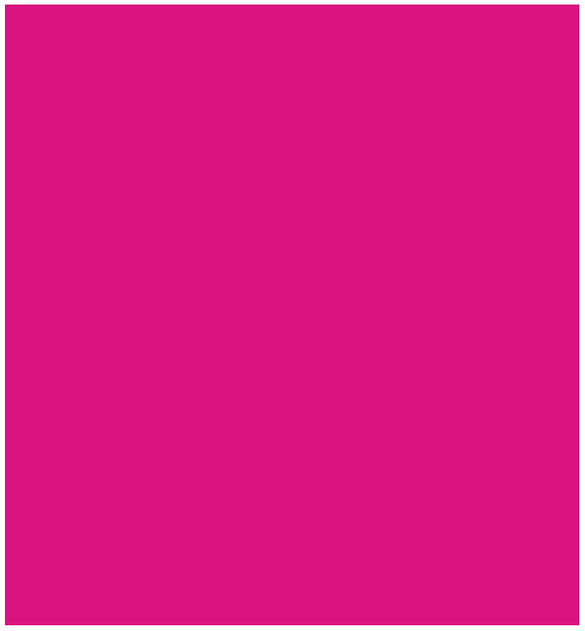
```
type alias Model =
    { counter : Int }
```

MODEL

```
model = { counter = 1 }
```

ELM ARCHITECTURE

Model



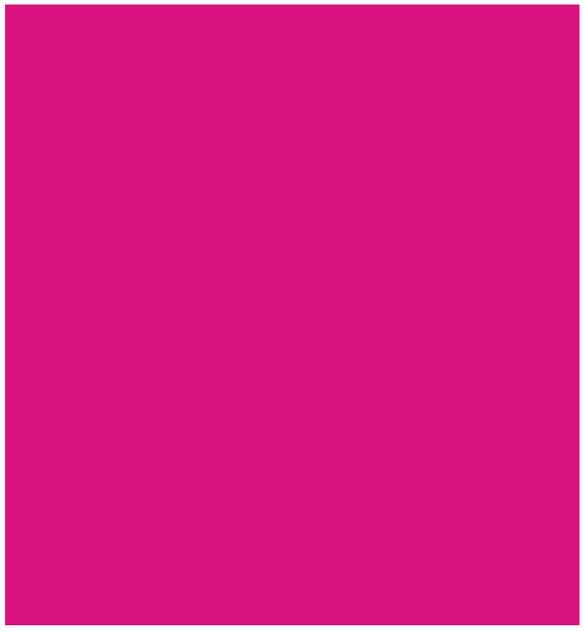
View

VIEW

```
view : Model -> Html Command
view model =
   button [ ]
      [ text (toString model.counter) ]
```

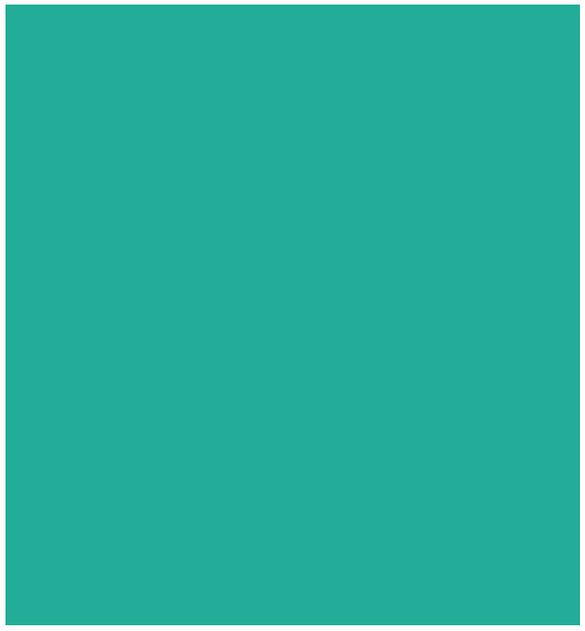
ELM ARCHITECTURE

Model



View

Update



UPDATE

```
type Message = Increment | Decrement
```

UPDATE

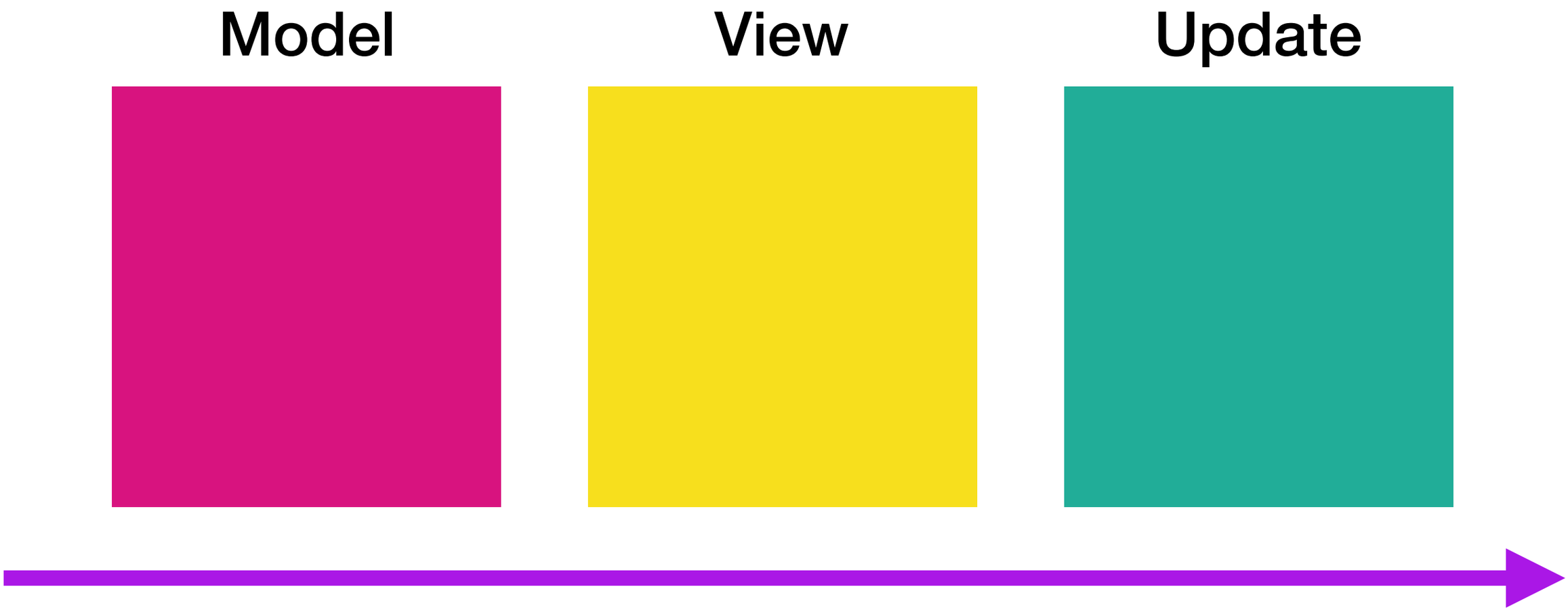
```
update : Message -> Model -> Model
update message model =
  case message of
    Increment ->
      { model | counter = model.counter + 1 }

    Decrement ->
      { model | counter = model.counter - 1 }
```

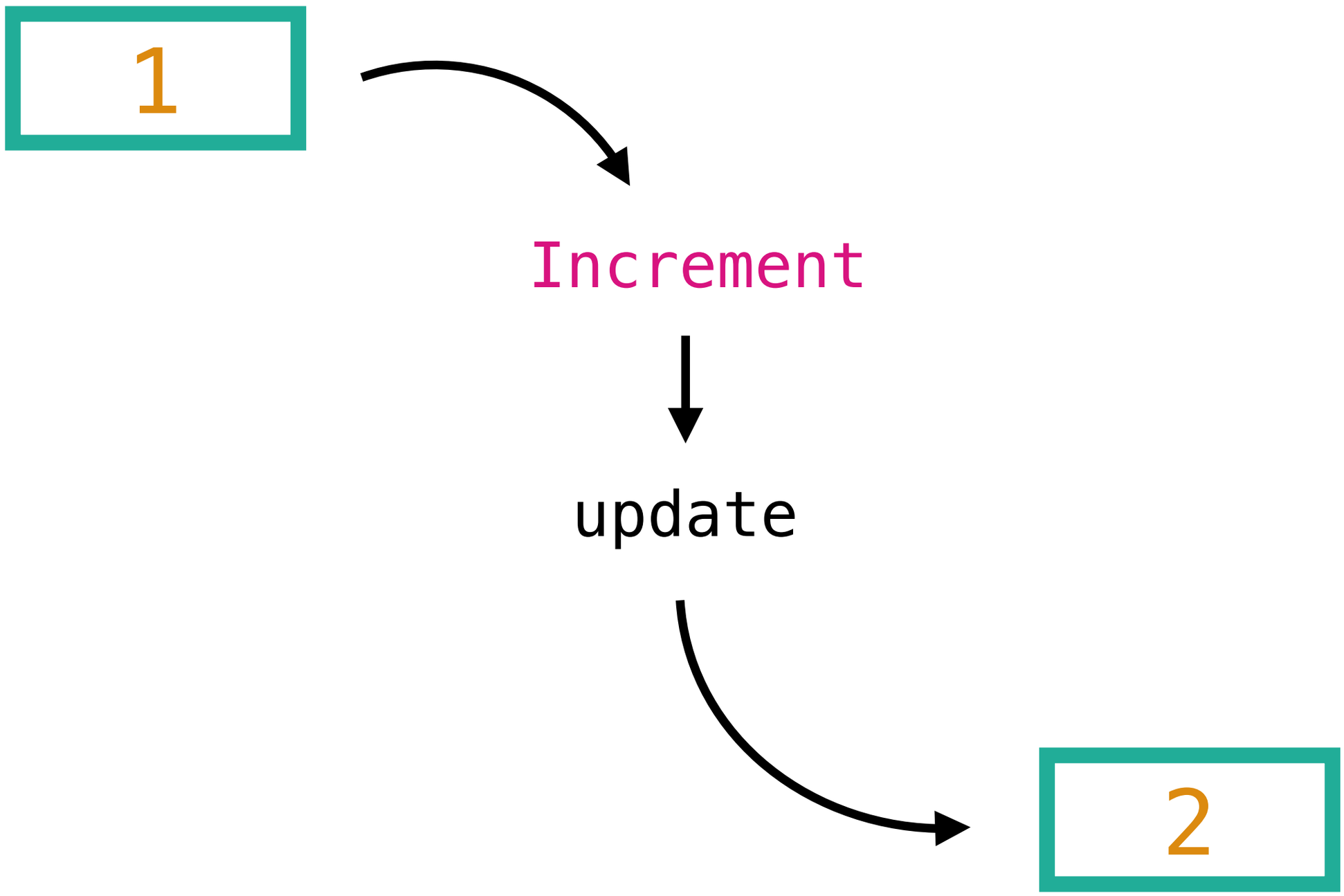
VIEW

```
view : Model -> Html Message
view model =
  button [ onClick Increment ]
         [ text (toString model.counter) ]
```

ELM ARCHITECTURE



ELM ARCHITECTURE



MAIN

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

Tudo conectado!

COMPILAR

```
elm-make App.elm
```

Só abrir index.html
e está tudo rodando!

SIMPLES

CONFIÁVEL

MANUTENÍVEL

VANTAGENS

Compilador inteligente

Arquitetura única e simples

Estruturas de dados imutáveis

Linguagem é simplificada

Pacotes versionados

DESAFIOS

Declarar efeitos colaterais

Interoperabilidade declarativa

Comunidade em construção

Feito para o browser

Estado local

EXPLORE

elm-lang.org

github.com/isRuslan/awesome-elm

guide.elm-lang.org

elmlang.slack.com

hugobessa.com.br

HUGO BESSA

@hugobessaa

hugobessa.com.br