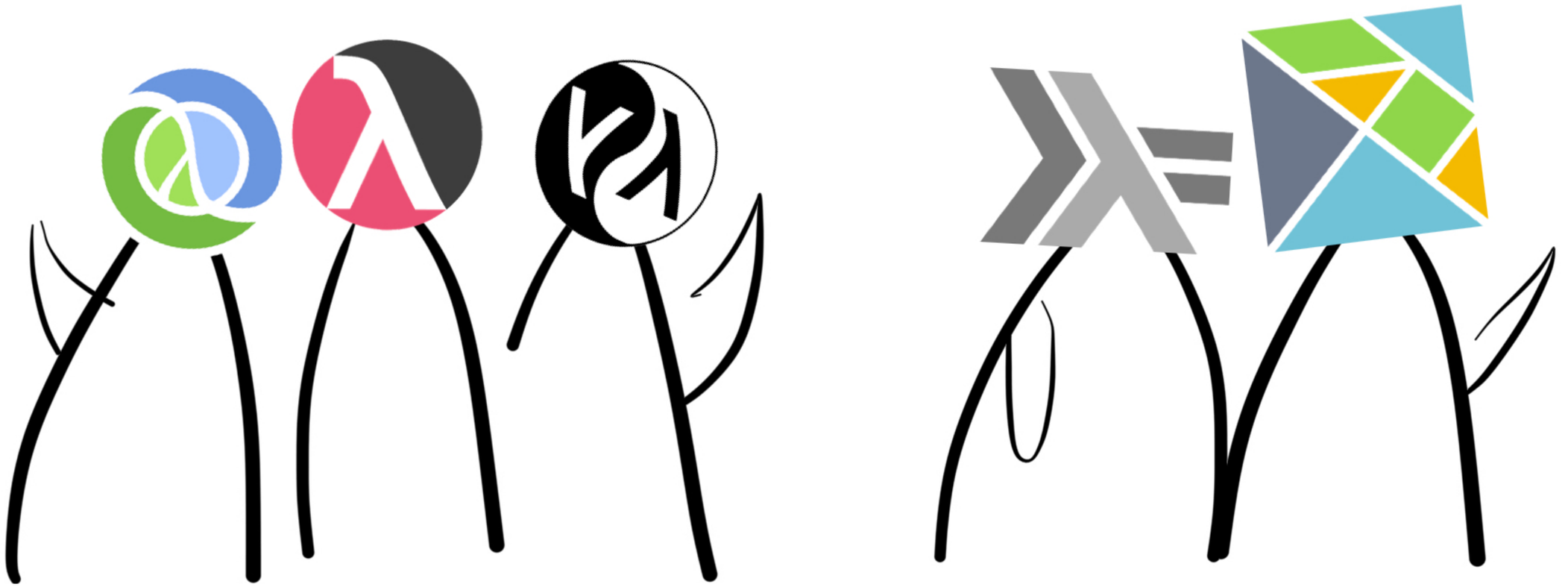
Aksel Wester og Johanne Håøy Horn, Bekk Consulting AS

PLAN

- Funksjonell programmering og webapplikasjoner
- Syntaks og språkkonsepter
- Liten app

FUNKSJONELL PROGRAMMERING I ARBEIDSLIVET

- God kode
- Brukervennlig, testbar, lesbar kode
- Vi bruker kanskje ikke Scheme, men prinsippene er universelle

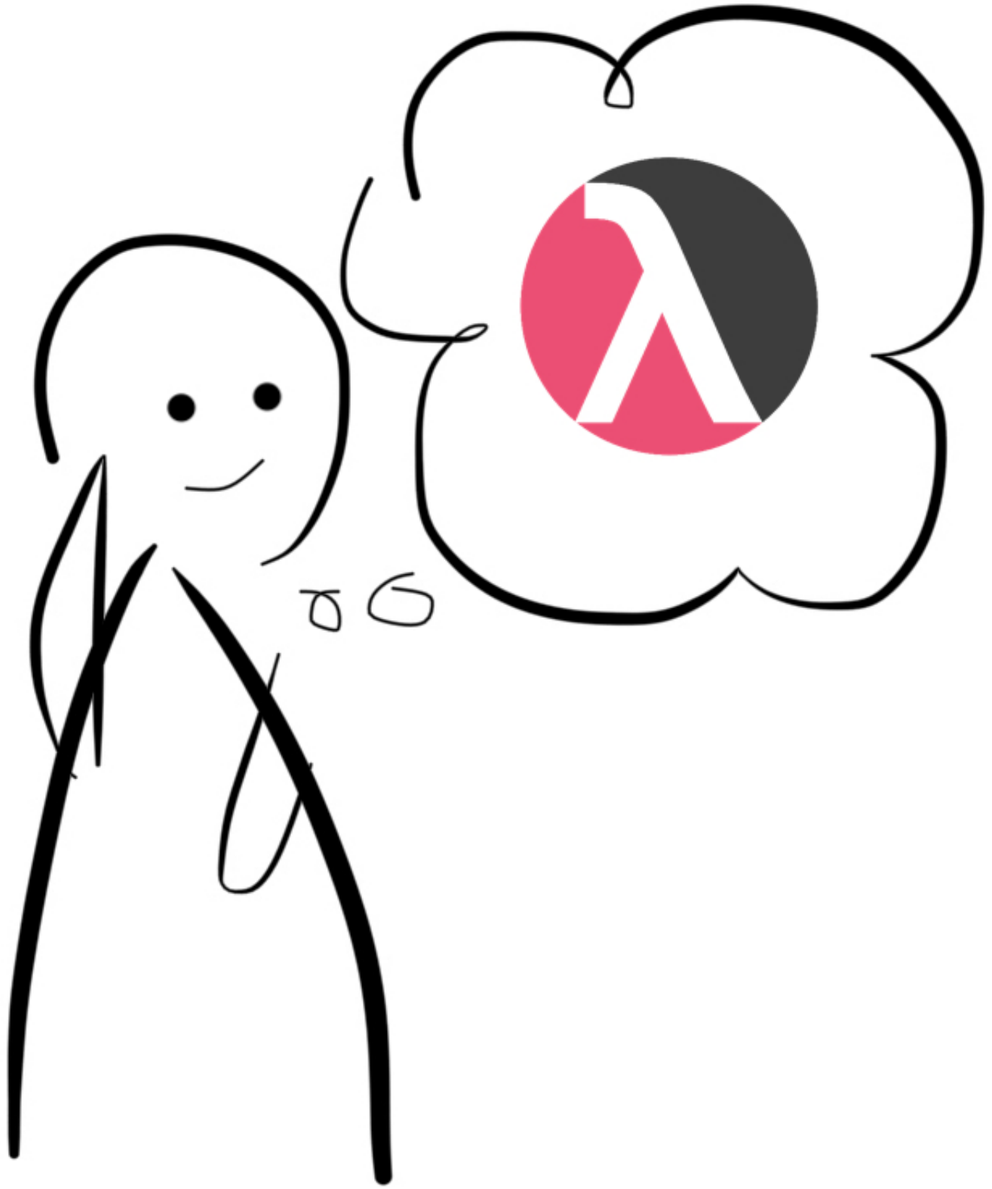


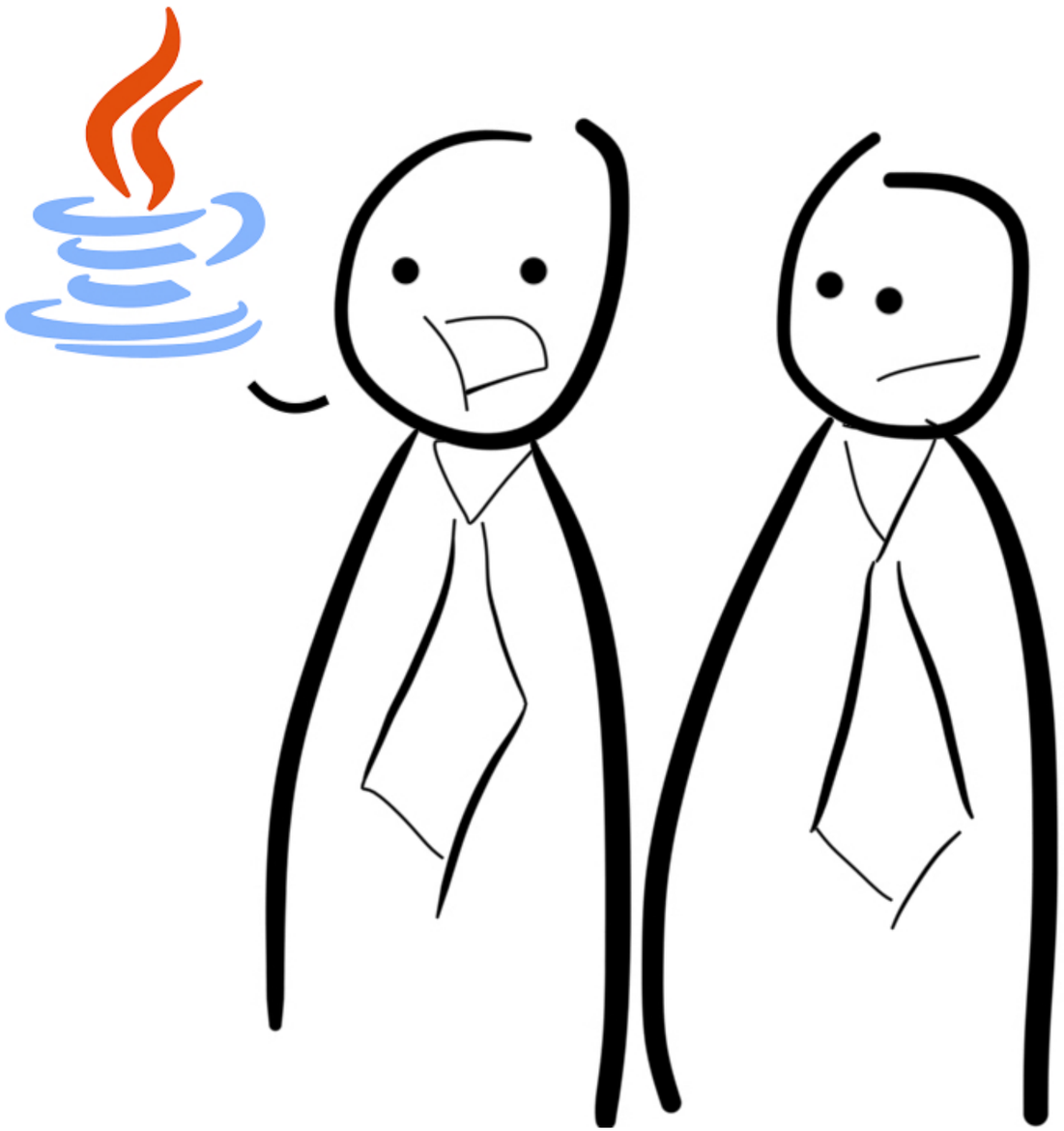

09


HTML

CSS


JS





JAVA != JAVASCRIPT

JAVASCRIPT

- Funksjonelt og objektorientert
- Ikke kompilert
- Dynamisk typet
- Svært fleksibelt

Ryan Bates
@rbates

Følg

I'm starting to wonder if there are more client-side JavaScript frameworks than there are apps that use them.

Oversett tweet

08:38 - 25. sep. 2012

TINGENES TILSTAND PÅ FRONTEND

- JS-stacken funker, men er kompleks
- JS har beveget seg i funksjonell retning: React, Redux
- Statisk typing har blitt mainstream

VANLIGE FEIL I JS

Script error.

Cannot read property 'getFullYear' of undefined

undefined is not a function

ELM: LAGET FOR WEBAPPLIKASJONER

ELM: KOMPILERER NED TIL JAVASCRIPT

ELM: STATISK TYPET FUNKSJONELT SPRÅK

ELM: LOVER “INGEN KJØRETIDSFEIL”

ELM: GODE KOMPILATORFEILMELDINGER

ELM: FOKUSERER PÅ BRUKERVENNLIGHET

ELM: TYDELIG, GJENNOMGÅENDE FILOSOFI

ELM: OFFISIELT ANBEFALTE BIBLIOTEKER FOR DET MAN TRENGER

ELM

FUNKSJONER & TYPEINFERENS

```
increment x =
    x + 1

five = increment 4
```

TYPESIGNATURER

```
increment : Int -> Int
increment x =
    x + 1

five : Int
five = increment 4
```

RECORDS

```
kunde : { navn : String, alder : Int }
kunde =
    { navn = "Ingar"
    , alder = 24
    }
```

TYPE ALIAS

```
type alias Kunde =
    { navn: String
    , alder: Int
    }

ingar : Kunde
ingar =
    { navn = "Ingar"
    , alder = 24
    }
```

Lar oss definere nye typer

TYPE ALIAS

```
type alias Koordinater = (Int, Int)

spillerposisjon : Koordinater
spillerposisjon = (0, 0)
```

TYPE ALIAS

```
type alias Kunde =
    { navn: String
    , alder: Int,
    , avtale: String
    }

ingar : Kunde
ingar =
    { navn = "Ingar"
    , alder = 24
    , avtale = "Student"
    }
```

TYPE ALIAS

```
type alias Kunde =
    { navn: String
    , alder: Int,
    , avtale: String
    , studentRabatt: Int
    }

ingar : Kunde
ingar =
    { navn = "Ingar"
    , alder = 24
    , avtale = "Student",
    , studentRabatt = 50
    }
```

TYPE ALIAS

```
type alias Kunde =
    { navn: String
    , alder: Int,
    , avtale: String
    , studentRabatt: Int
    , bedriftsnavn: String
    }

ingar : Kunde
ingar =
    { navn = "Ingar"
    , alder = 24
    , avtale = "Bedrift",
    , studentRabatt = 0
    , bedriftsnavn = "Bekk Consulting"
    }
```

TRE PROBLEMER:

1. Vi får tomme felter med **dummy**-verdier
2. Enkelt å skrive feil i `type`-feltet
3. Ikke noe hjelp fra kompilatoren

UNION TYPES

```
type Kundeavtale
    = Student
    | Bedrift
    | Privat
```

Som enums på stereoider

UNION TYPES

```
type Kundeavtale
    = Student Int
    | Bedrift String
    | Privat
```

UNION TYPES

```
type alias Rabatt = Int
type alias Bedriftsnavn = String

type Kundeavtale
    = Student Rabatt
    | Bedrift Bedriftsnavn
    | Privat
```

```
type alias Kunde =
    { navn: String
    , alder: Int,
    , avtale: Kundeavtale
    }

ingar : Kunde
ingar =
    { navn = "Ingar"
    , alder = 24
    , avtale = Bedrift "Bekk Consulting"
    }
```

PATTERN MATCHING

```
type Kundeavtale = Student Rabatt | Bedrift Bedriftsnavn | Privat

getRabatt : Kundeavtale -> Rabatt
getRabatt avtale =
    case avtale of
        Student rabatt ->
            rabatt
        Bedrift navn ->
            0
        Private ->
            0
```

Glemt en branch? kompilatoren sier fra!

MAYBE

```
type Maybe a = Just a | Nothing
```

MAYBE

```
type Maybe a = Just a | Nothing

type alias Game =
    { highscore: Maybe Int
    }
```

MAYBE

```
type Maybe a = Just a | Nothing

type alias Game =
    { highscore: Maybe Int
    }

getHighscore : Game -> String
getHighscore game =
    case game.highscore of
        Just score ->
            toString score
        Nothing ->
            "No highscore"
```

HTML

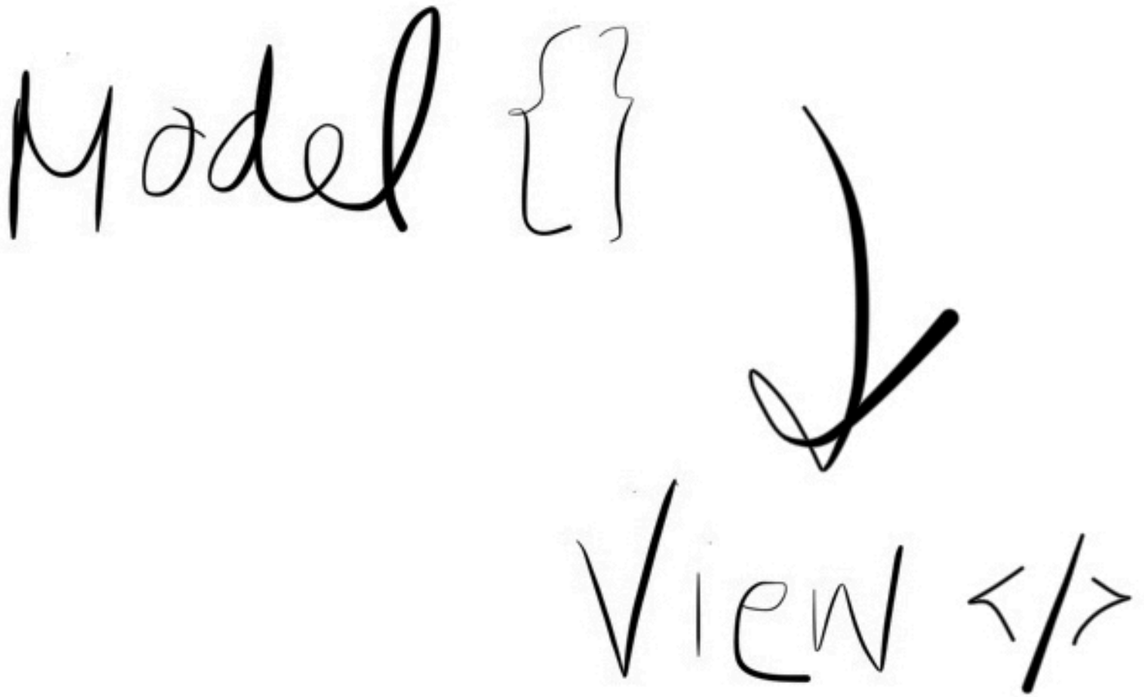
```
<div>
    <img src="/image.png" />
    <h1>Min elm-app!</h1>
</div>
```

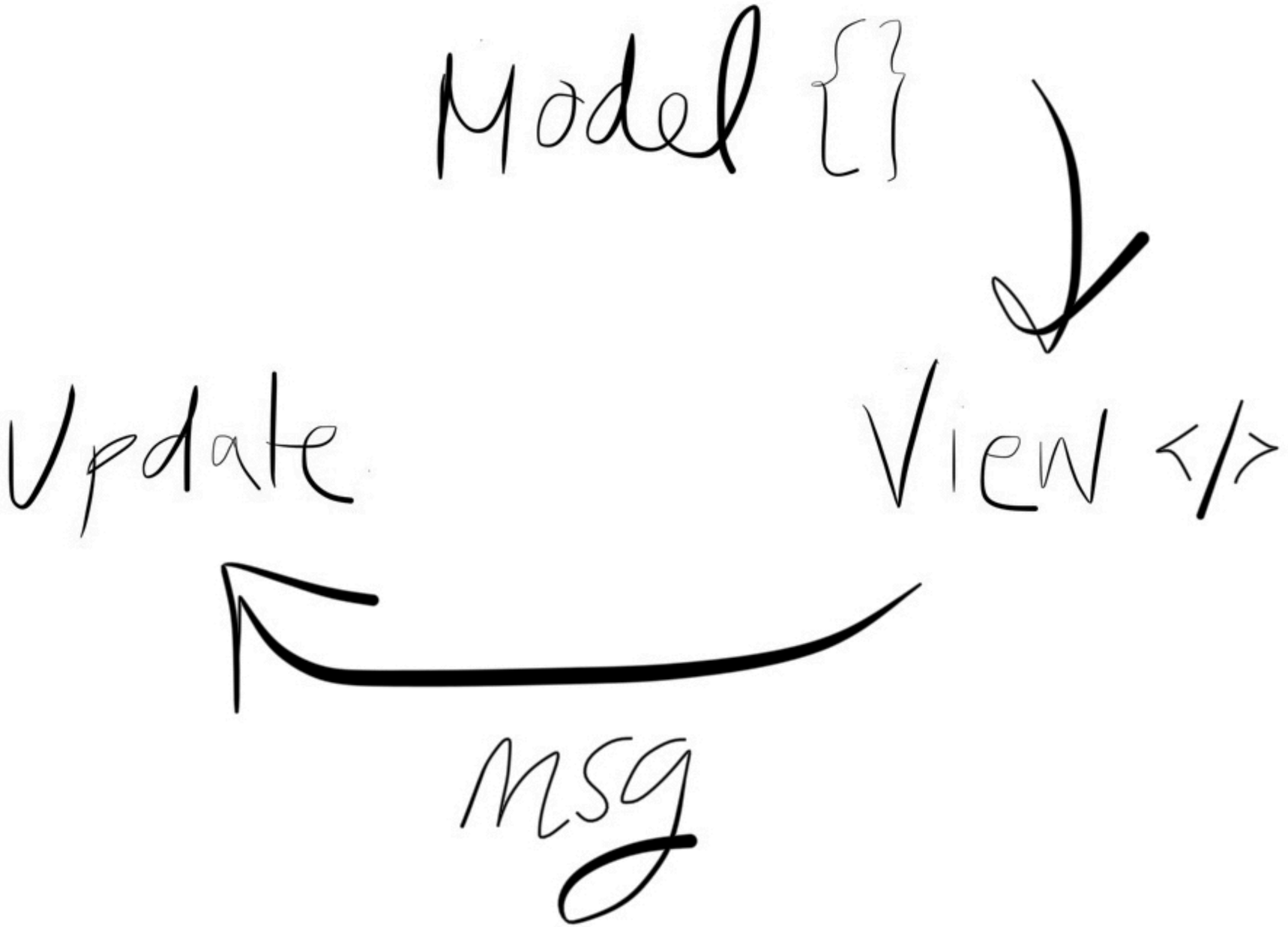
HTML

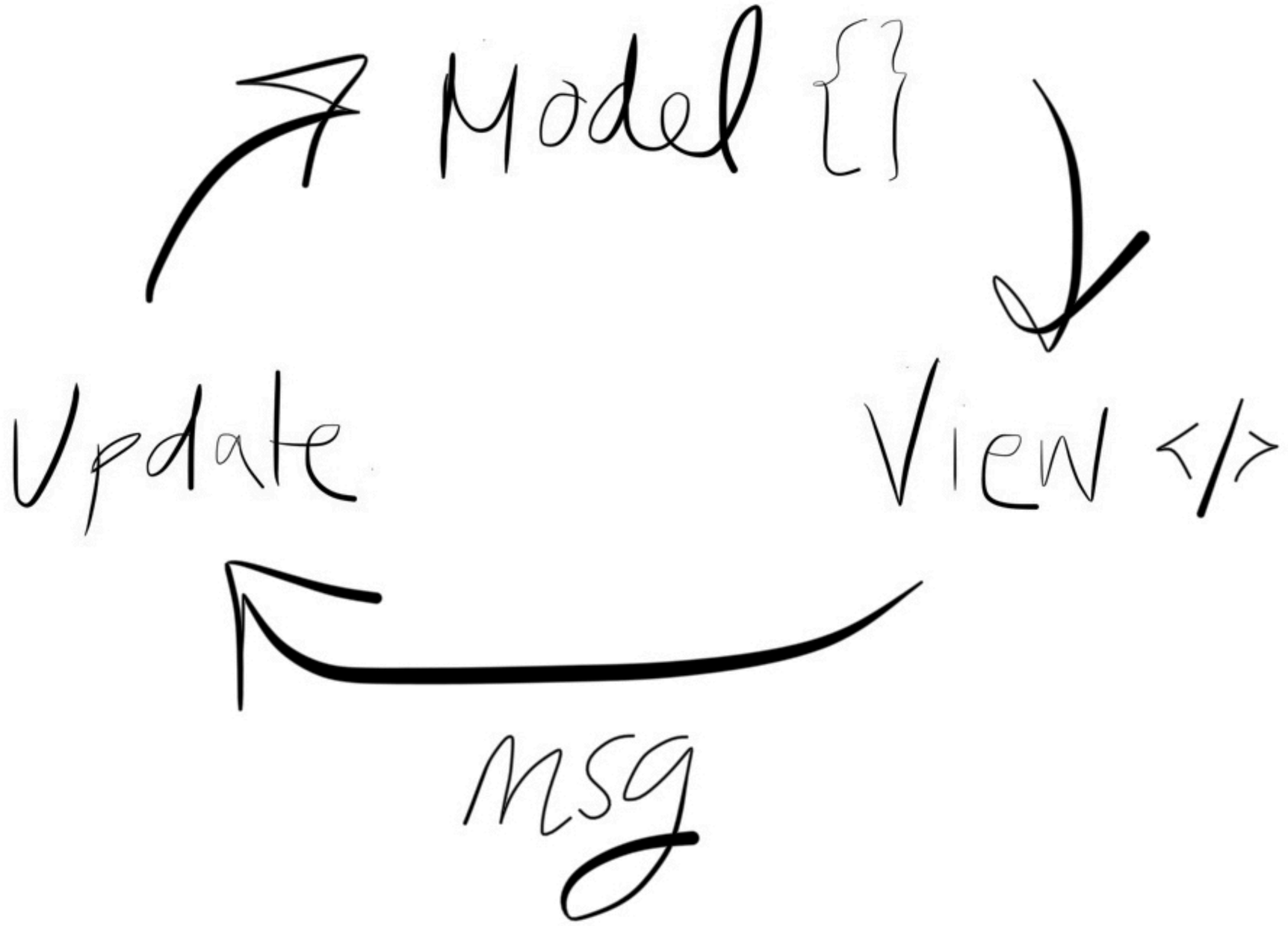
```
div []
    [ img [src "/image.png"] []
    , h1 [] [ text "Min elm-app!"]
    ]
```

THE ELM ARCHITECTURE

Model {}


Model {}
View </>


Model {}
View </>
Update
Msg


Model {}
View </>
Update
Msg

THE ELM ARCHITECTURE

```
type alias Model = ...
type Msg = Forskjellige | Beskjeder

view : Model -> Html Msg

update : Msg -> Model -> Model
```

LIVE-KODING

HVORFOR ELM

- Mindre kompleksitet
- Færre bugs
- Enklere refaktorering og vedlikehold
- Hjelpsomt community
- Folk som liker Elm liker det veldig godt!

VIL DU PRØVE SELV?

```
$ npm install -g create-elm-app
$ create-elm-app min-forste-app
$ elm-app start
```

ANDRE RESSURSER

Introduksjon til Elm: guide.elm-lang.org

Elm-miljøet har en slack: elmlang.herokuapp.com

Elm-workshop: ewendel.github.io/elm-workshop


TALK