Elm

A delightful language

for reliable webapps

Plan for dagen

- Intro/presentasjon til Elm
- Lunsj
- Lage spill workshop

Hvorfor Elm?

- Hva kan du gjøre?
- Hva kan du *ikke* gjøre?

Du kan ikke bruke `null`

Du kan ikke bruke exceptions

Du kan ikke endre/mutere ting

Du kan ikke gjøre nettverkskall hvor og når du vil

Mye lettere å forstå hva koden faktisk gjør

Typesikkerhet og rubusthet

Stabilt språk

Enkelt å refaktorere, enkelt å vedlikeholde

Enkelt å debugge

Innebygget time traveling debugger

Syntax

Verdier

```
-- Elm
volum = 11
```

```
// JavaScript:
const volum = 11;
```

Flere verdier

```
volum = 11

pi = 3.14

kultProsjekt = "Statens Vegvesen"

detErHelg = False
```

Funksjoner

```elm
-- Elm:
increment x =
    x + 1

four = increment 3
```

```javascript
// JavaScript:
function increment(x) {
    return x + 1;
};

const four = increment(3);
```

Funksjoner & typeinferens

```elm
-- Elm:
increment x =
    x + 1

five = increment (increment 3)
```

```javascript
// JavaScript:
function increment(x) {
    return x + 1;
};

const five = increment(increment(3));
```

```
-- TOO MANY ARGS ---------------------------------------------------- elm

The `increment` function expects 1 argument, but it got 2 instead.

5|    increment increment 3
      ^^^^^^^^^^

Are there any missing commas? Or missing parentheses?
```

Funksjoner & typeinferens

```elm
-- Elm:
increment x =
    x + 1

five = increment (increment 3)
```

```javascript
// JavaScript:
function increment(x) {
    return x + 1;
};

const five = increment(increment(3));
```

```
> increment "1"
-- TYPE MISMATCH ------------------------------------------------------ elm

The 1st argument to `increment` is not what I expect:

5|   increment "1"
                ^^^
This argument is a string of type:

    String

But `increment` needs the 1st argument to be:

    number

Hint: Try using String.toInt to convert it to an integer?
```

Funksjoner & typeinferens

```elm
-- Elm:
increment x =
    x + 1

five = increment (increment 3)
```

```javascript
// JavaScript:
function increment(x) {
    return x + 1;
};

const five = increment(increment(3));
```

Lambda

```elm
-- Elm
increment x =
    x + 1

increment = \x -> x + 1
```

```javascript
// JavaScript:
function increment(x) {
    return x + 1;
}

const increment = (x) => x + 1;
```

Typer

```
volum : Int
volum = 11

pi : Float
pi = 3.14

kultProsjekt : String
kultProsjekt = "Statens Vegvesen"

detErHelg : Bool
detErHelg = False
```

Typesignaturer

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

five : Int
five = increment (increment 3)
```

Lister

```
favorittMat : List String
favorittMat = [ "Pizza", "Lasagne", "Enchiladas" ]

tidligereTemperatur : List Float
tidligereTemperatur = [ 20.2, 21, 5, 19.5, 19.9 ]

oppdatertTemperatur : List Float
oppdatertTemperatur = 21.1 :: tidligereTemperatur
```

Tupler

```
unit : ()
unit = ()

svaret : Int
svaret = 42

posisjon : ( Int, Int )
posisjon = ( 5, 10 )

person : ( String, Int, Bool )
person = ( "Robin", 30, False )
```

Records

```elm
-- Elm:
kunde =
    { navn = "Aksel"
    , alder = 31
    }
```

```javascript
// JavaScript:
const kunde = {
    navn: "Aksel",
    alder: 31
};
```

Records

```elm
-- Elm:
kunde : { navn : String, alder : Int }
kunde =
    { navn = "Aksel"
    , alder = 31
    }
```

```javascript
// JavaScript:
const kunde = {
    navn: "Aksel",
    alder: 31
};
```

Records

```elm
-- Elm:
kunde : { navn : String, alder : Int }
kunde =
    { navn = "Aksel"
    , alder = 31
    }
```

```javascript
// JavaScript:
const kunde = {
    navn: "Aksel",
    alder: 31
};
```

Type alias

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

kunde : Kunde
kunde =
    { navn = "Aksel"
    , alder = 31
    }
```

Lar oss definere nye typer

Type alias

```
type alias Person = ( Name, Age, Cool )

type alias Name = String

type alias Age = Int

type alias Cool = Bool
```

Type alias

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

kunde : Kunde
kunde =
    { navn = "Aksel"
    , alder = 31
    , avtale = "Student"
    }
```

Type alias

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

kunde : Kunde
kunde =
    { navn = "Aksel"
    , alder = 31
    , avtale = "Student"
    , studentRabatt = 50
    }
```

Type alias

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

kunde : Kunde
kunde =
    { navn = "Aksel"
    , alder = 31
    , avtale = "Bedrift"
    , studentRabatt = 0
    , bedriftsnavn = "Statens Vegvesen"
    }
```

Tre problemer:

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

```
{ navn = "Aksel"
, alder = 31
, avtale = "Bedrift"
, studentRabatt = 0
, bedriftsnavn = "Statens Vegvesen"
}
```

Custom Types

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

Making Impossible States Impossible! 👏

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

kunde : Kunde
kunde =
    { navn = "Aksel"
    , alder = 31
    , avtale = Bedrift "Statens Vegvesen"
    }
```

Pattern Matching

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

getRabatt : Kundeavtale -> Int
getRabatt avtale =
    case avtale of
        Student rabatt ->
            rabatt
        Bedrift navn ->
            0
        Privat ->
            0
```

Glemt en branch? Kompilatoren sier fra!

Pattern Matching

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

getRabatt : Kundeavtale -> Int
getRabatt avtale =
    case avtale of
        Student rabatt ->
            rabatt
        _ ->
            0
```

Glemt en branch? Kompilatoren sier ikke fra! 😒

HTML

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

HTML

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

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

HTML typen

```
view : Html a
view =
  div []
    [ img [ src "/image.png" ] []
    , h1 [] [ text "Min elm-app!" ]
    ]
```

HTML typen

```
type Msg = VisBilde

view : Html Msg
view =
  div []
    [ img [ src "/image.png" ] []
    , h1 [] [ text "Min elm-app!" ]
    ]
```

Din tur

```
https://mbolstad.github.io/elm-workshop-memory
```

Elm Workshop

Funksjonell programmering

- Funksjoner er førsteklasses borgere
- Høyereordens funksjoner
- Immutable datastrukturer
- Rene funksjoner (ingen side-effekter)

Currying

```
concat : String -> String -> String
concat one two =
  one ++ two
```

Currying

```
concat : String -> String -> String
concat one two =
  one ++ two

greeting =
  concat "Hello "
```

Currying

```
concat : String -> String -> String
concat one two =
  one ++ two

greeting : String -> String
greeting =
  concat "Hello "
```

Currying

```
concat : String -> String -> String
concat one two =
  one ++ two

greeting : String -> String
greeting name =
  concat "Hello " name
```

Currying

```
concat : String -> String -> String
concat one two =
  one ++ two

greeting : String -> String
greeting =
  concat "Hello "

greeting "World" == "Hello World"
```

Currying (i JS)

```
function concat(a) {
    return function (b) {
        return a + b;
    }
}


concat("Hello ")("World") == "Hello World";

const greeting = greeting("Hello ");

greeting("World") == "Hello World";
greeting("Kitty") == "Hello Kitty";
```

Partial application

```
List.map greeting [ "Gaute", "Even", "Aksel" ] == [ "Hello Gaute", "Hello Even", "Hello Aksel" ]


List.map (concat "Hello ") [ "Gaute", "Even", "Aksel" ] == [ "Hello Gaute", "Hello Even", "Hello Aksel" ]


greetings : List String -> List String
greetings =
  List.map greeting


greetings [ "Gaute", "Even", "Aksel" ] == [ "Hello Gaute", "Hello Even", "Hello Aksel" ]
```

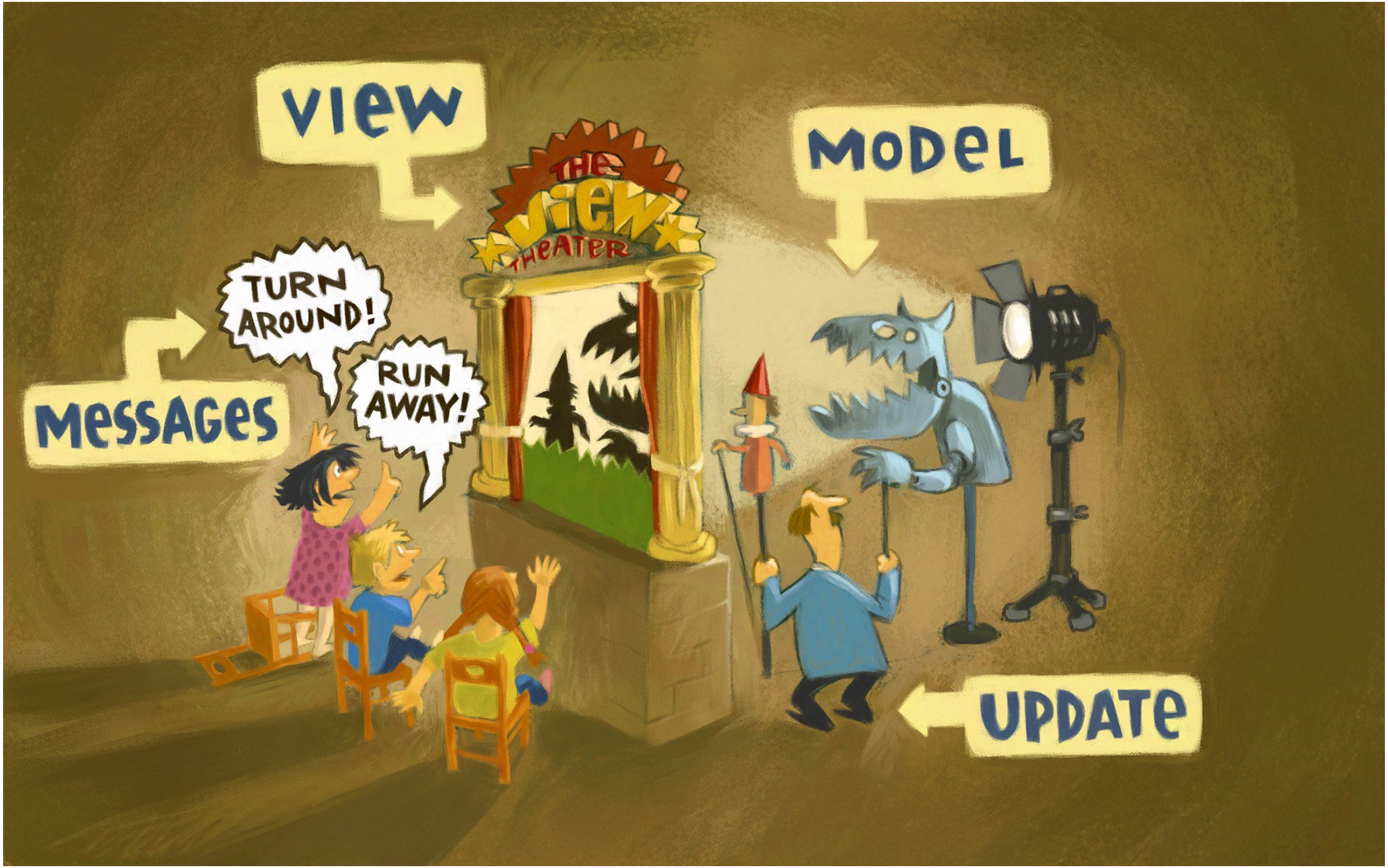
Pipes

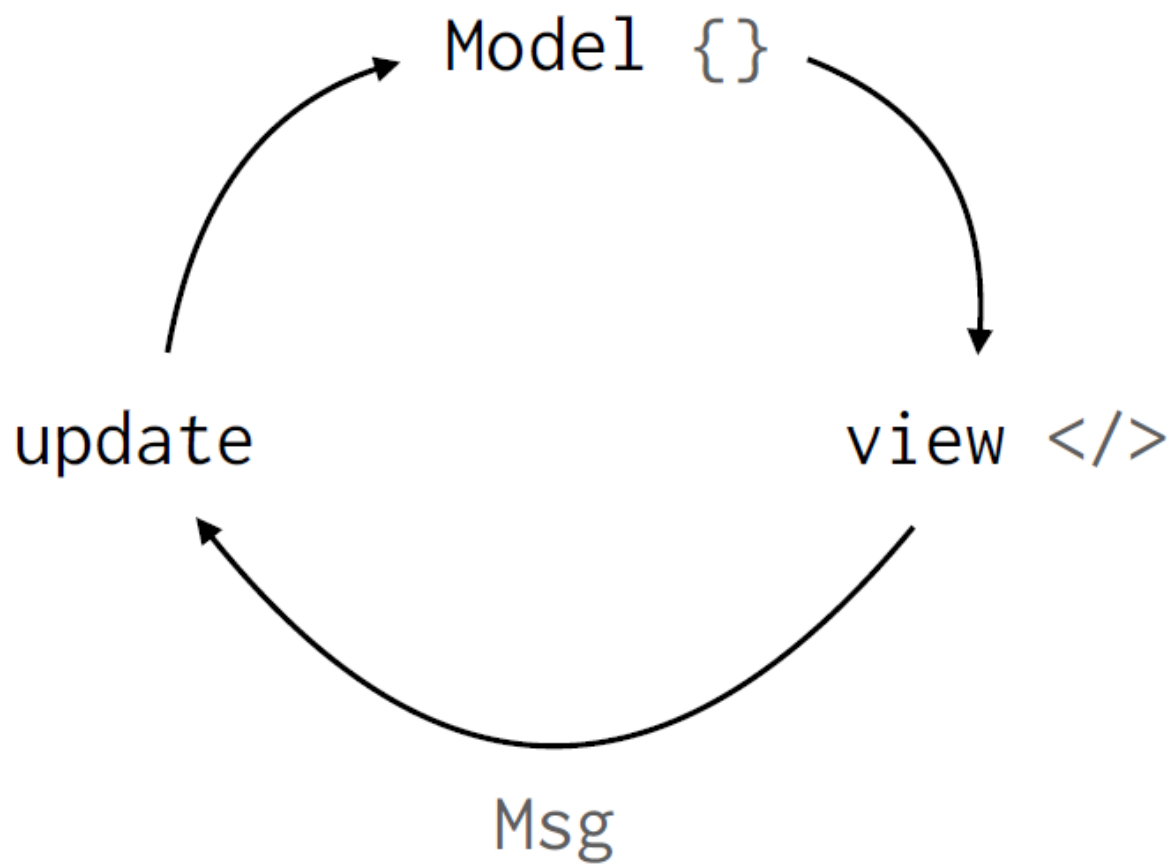
```
myString =
    String.toUpper (String.repeat 2 (String.reverse "olleh"))


"olleh"
    |> String.reverse
    |> String.repeat 2
    |> String.toUpper

--> "HELLOHELLO"
```

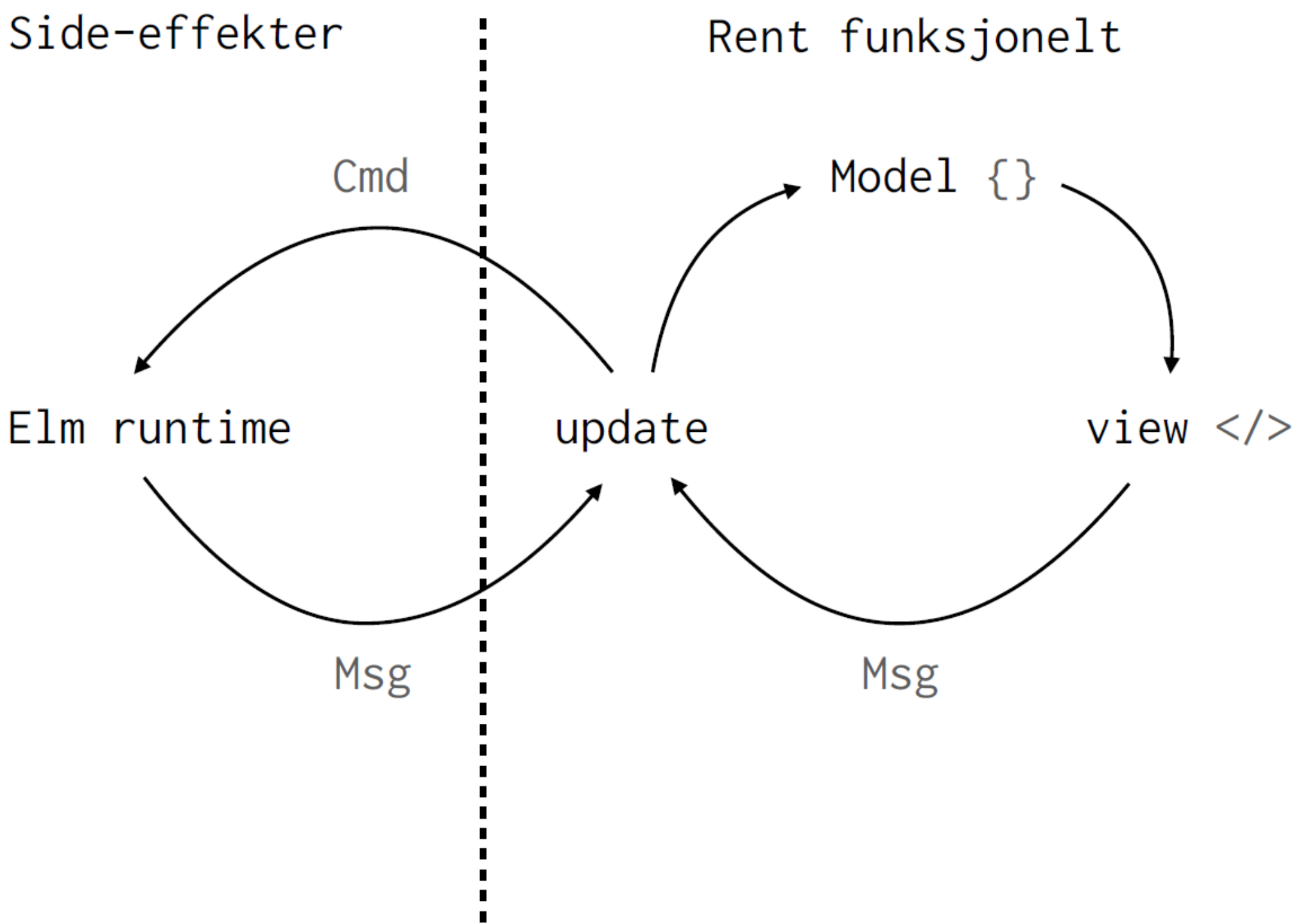
Let

```
sirkelAreal r =
    let
        pi = 3.14
        r2 = r * r
    in
        pi * r2
```

Elm Architecture


view
THE
View
THEATER
model
TURN AROUND!
RUN AWAY!
messages
update


Model {}
update
view </>
Msg


Side-effekter
Rent funksjonelt
Cmd
Model {}
Elm runtime
update
view </>
Msg
Msg

The Elm Architecture

```
type alias Model = {...}

type Msg = BrukerTrykketPaaEnKnapp | NoeAnnetSkjedde

view : Model -> Html Msg

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

-

4

+

Browser.sandbox

```
main =
  Browser.sandbox
    { init = init
    , view = view
    , update = update
    }
```

- Browser.element: Tillater sideeffekter (HTTP, JS-interop, hente dato og tid)
- Browser.document: Som element, men gir kontroll over <title> og <body>
- Browser.application: Lager en applikasjon som håndterer URL-endringer (routing)

Init

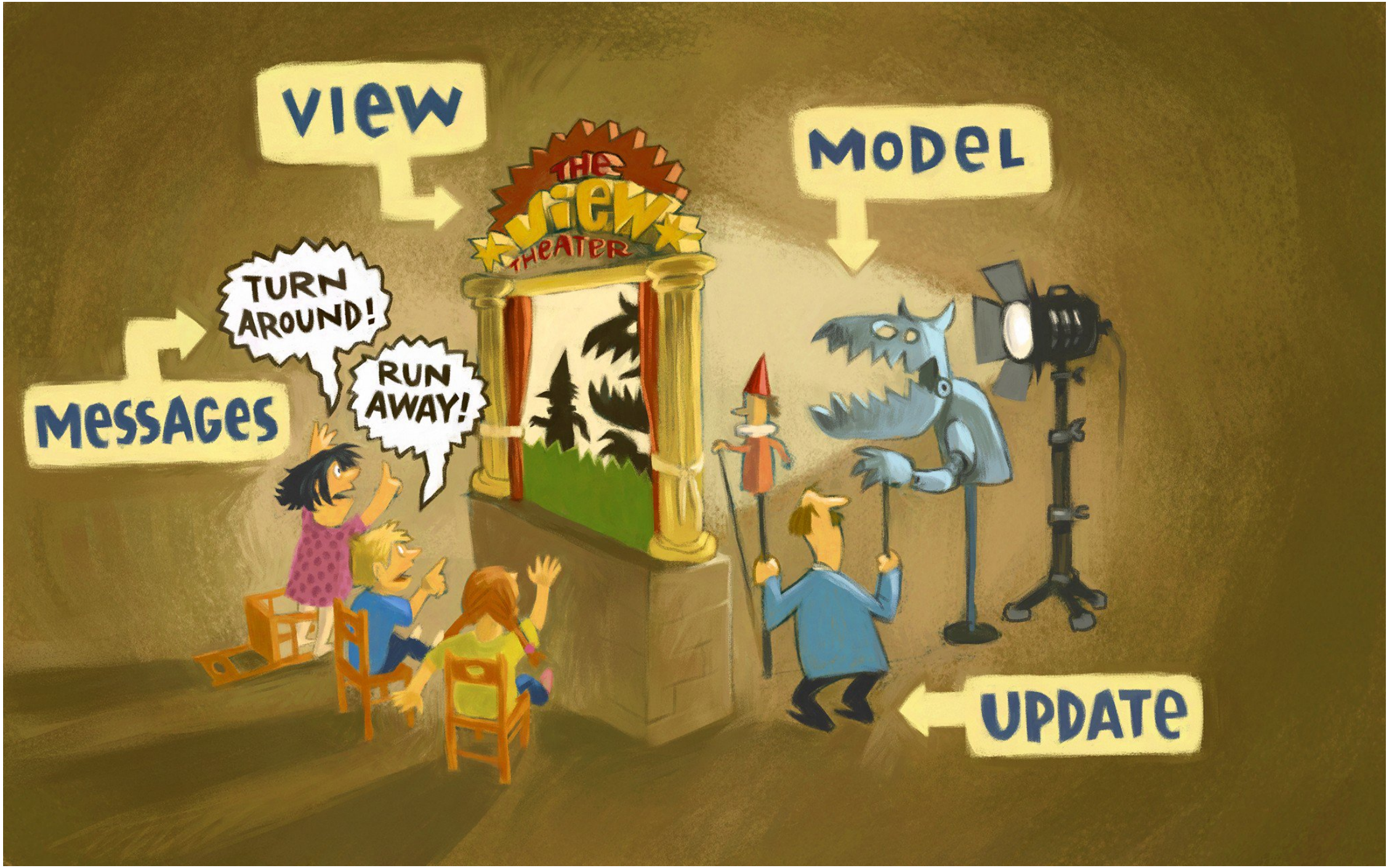
```
type alias Model = Int

init : Model
init = 0
```

View

```
type Msg
  = PlussKlikket
  | MinusKlikket


view : Model -> Html Msg
view model =
  div []
    [ button [ onClick MinusKlikket ]
        [ text "-" ]
    , text (String.fromInt model)
    , button [ onClick PlussKlikket ]
        [ text "+" ]
    ]
```

Update

```
update : Msg -> Model -> Model
update msg model =
  case msg of
    PlussKlikket ->
      model + 1

    MinusKlikket ->
      model - 1
```


view
THE
view
THEATER
model
TURN
AROUND!
RUN
AWAY!
messages
update

Maybe

```
type Maybe a
  = Just a
  | Nothing


type alias Spill =
  { tittel : String
  , personligRekord : Maybe Int
  }
```

Feilhåndtering - Maybe

```
visPersonligRekord : Spill -> String
visPersonligRekord spill =
  case spill.personligRekord of
    Just pers ->
      String.fromInt pers

    Nothing ->
      "Ingen personlig rekord"
```

Result

```
type Result error value
  = Err error
  | Ok value
```

```
isReasonableAge : Int -> Result String Int
isReasonableAge age =
  if age < 0 then
    Err "Please try again after you are born."

  else if age > 135 then
    Err "Are you some kind of turtle?"

  else
    Ok age
```

Result

```
type Result error value
  = Err error
  | Ok value

type AgeError
  = TooYoung
  | TooOld
```

```
toReasonableAge : Int -> Result AgeError Int
toReasonableAge age =
  if age < 0 then
    Err TooYoung

  else if age > 135 then
    Err TooOld

  else
    Ok age
```

Feilhåndtering - Result

```
viewAge : Result AgeError Int -> String
viewAge ageResult =
  case ageResult of
        Ok age ->
          "Age: " ++ String.fromInt age

        Err TooOld ->
          "Are you some kind of turtle?"

        Err TooYoung ->
          "Please try again after you are born."
```

Mapping-funksjoner

```
Maybe.map : (a -> b) -> Maybe a -> Maybe b

visPersonligRekord : Spill -> String
visPersonligRekord spill =
  spill.personligRekord
    |> Maybe.map String.fromInt
    |> Maybe.withDefault "Ingen personlig rekord"
```

Mapping-funksjoner

```
Maybe.map : (a -> b) -> Maybe a -> Maybe b

Result.map : (a -> b) -> Result x a -> Result x b

List.map : (a -> b) -> List a -> List b
```

Json-dekoding

- Typesikkerhet
- Feilhåndtering

Json-dekoding

```
import Json.Decode exposing
        ( Decoder
        , field
        , int
        , string
    )

ageDecoder : Decoder Int
ageDecoder =
  field "age" int

-- int : Decoder Int
-- field : String -> Decoder a -> Decoder a

nameDecoder : Decoder String
nameDecoder =
  field "name" string

-- string : Decoder String
```

```
{ "name": "Tom", "age": 42 }
```

Json-dekoding

```
Decode.map : (a -> b) -> Decoder a -> Decoder b


ageDecoder : Decoder Int
ageDecoder =
  field "age" int


chineseAgeDecoder : Decoder Int
chineseAgeDecoder =
  Decode.map (\age -> age + 1) ageDecoder
```

Json-dekoding

```
Maybe.map2 : (a -> b -> value) -> Maybe a -> Maybe b -> Maybe value

Maybe.map2 (+) (Just 1) (Just 2)
--> Just 3

List.map2 : (a -> b -> value) -> List a -> List b -> List value

List.map2 (++) [ "Hei", "Hello" ] [ "Verden", "World" ]
--> [ "HeiVerden", "HelloWorld" ]

Decode.map2 : (a -> b -> value) -> Decoder a -> Decoder b -> Decoder value
```

Json-dekoding

```
-- Person : String -> Int -> Person
type alias Person =
  { name : String
  , age : Int
  }

map2 : (a -> b -> value) -> Decoder a -> Decoder b -> Decoder value

personDecoder : Decoder Person
personDecoder =
  Decode.map2 Person
      nameDecoder
      ageDecoder
```

Json-dekoding

```
-- Person : String -> Int -> Person
type alias Person =
  { name : String
  , age : Int
  }

map2 : (a -> b -> value) -> Decoder a -> Decoder b -> Decoder value

personDecoder : Decoder Person
personDecoder =
  Decode.map2 Person
      (field "name" string)
      (field "age" int)
```

Json-dekoding

NoRedInk/elm-json-decode-pipeline

Json-dekoding

```
type alias Person =
  { name : String
  , age : Int
  , phone : String
  }

personDecoder : Decoder Person
personDecoder =
    Decode.succeed Person
        |> required "name" string
        |> required "age" int
        |> optional "name" string ""
```

Json-dekoding

```
type alias Person =
  { name : String
  , age : Int
  , phone : Maybe String
  }

personDecoder : Decoder Person
personDecoder =
    Decode.succeed Person
        |> required "name" string
        |> required "age" int
        |> optional "name" (maybe string) Nothing
```

Json-dekoding

```
{
  "version": 1,
  "name": "Tom",
  "phone": 99112233
}

{
  "version": 2,
  "name": "Tom",
  "phone": "+47 99112233"
}
```

Json-dekoding

```
andThen : (a -> Decoder b) -> Decoder a -> Decoder b
```

```
type alias Person =
  { name : String
  , phone : String
  }

versionedPersonDecoder : Decoder Person
versionedPersonDecoder =
  field "version" int
    |> Decode.andThen personDecoder
```

```
personDecoder : Int -> Decoder Person
personDecoder version =
  case version of
    2 ->
      Decode.succeed Person
        |> required "name" string
        |> required "phone" string

    1 ->
      Decode.succeed Person
        |> required "name" string
        |> ( required "phone" int
               |> Decode.map String.fromInt
```

Json-dekoding

```
type Msg
  = GotPerson (Result Http.Error Person)


getPerson : Cmd Msg
getPerson =
  Http.get
    { url = "/person/123"
    , expect = Http.expectJson versionedPersonDecoder GotPerson
    }
```