elm Amsterdam
26 July 2016

Introduction to **elm**

Matthias Benkort **@KtorZ**

DIMEBOX

github.com/KtorZ/elm-amsterdam-chat

Functional (Reactive) Programming

In the Browser (JS compiled)

Reliable Packages

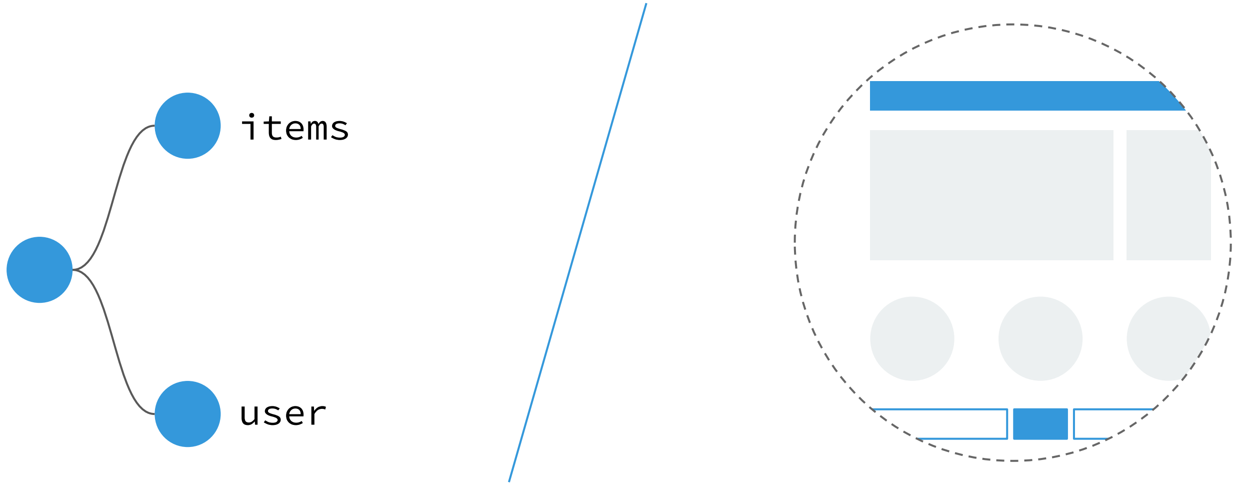
Expressive Compiler

The Elm Architecture

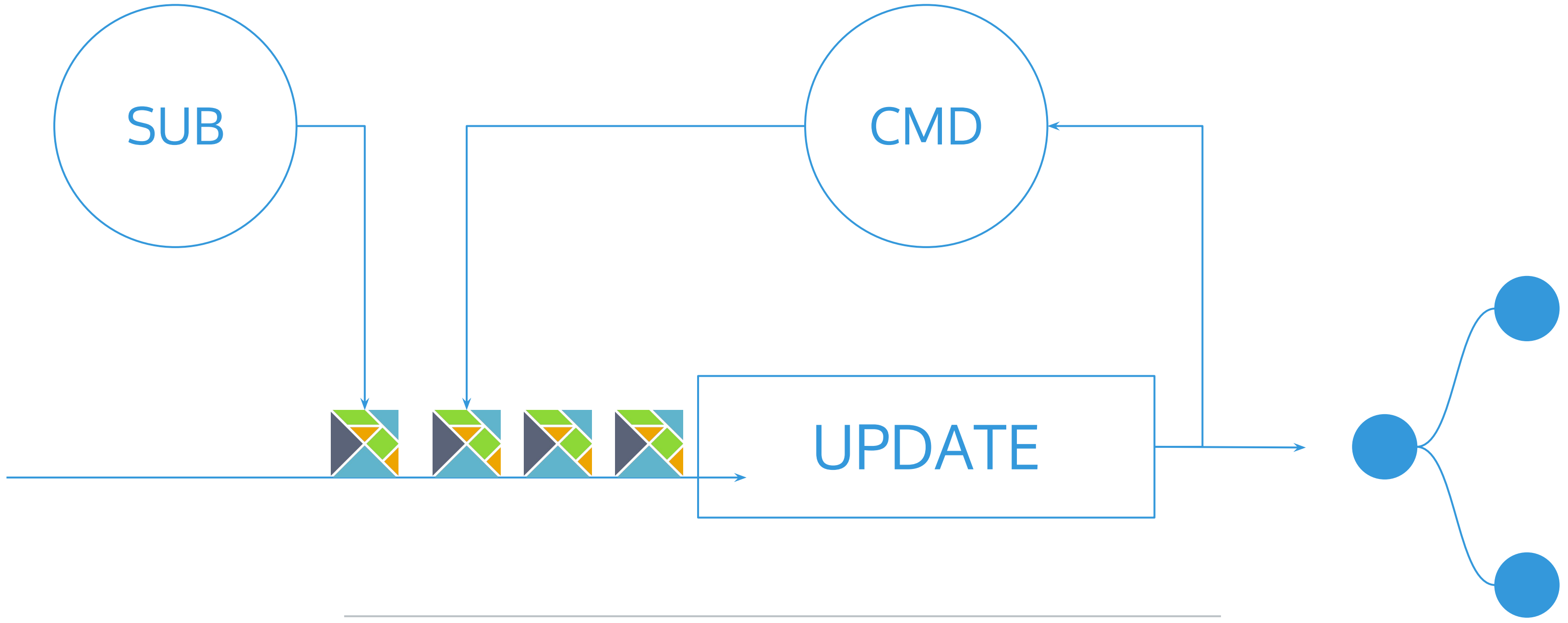
Growing Community

AWESOMENESS

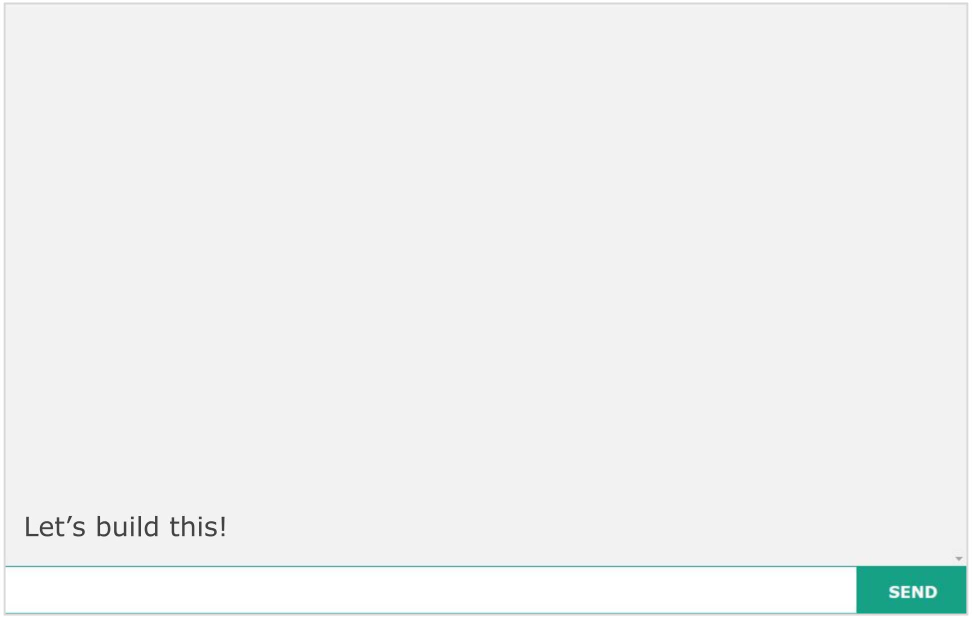
An application **model** represents possible **states**.
One **state** corresponds to a **view** representation.



Commands and **subscriptions** generate **messages**. They **update** the application **model** accordingly.



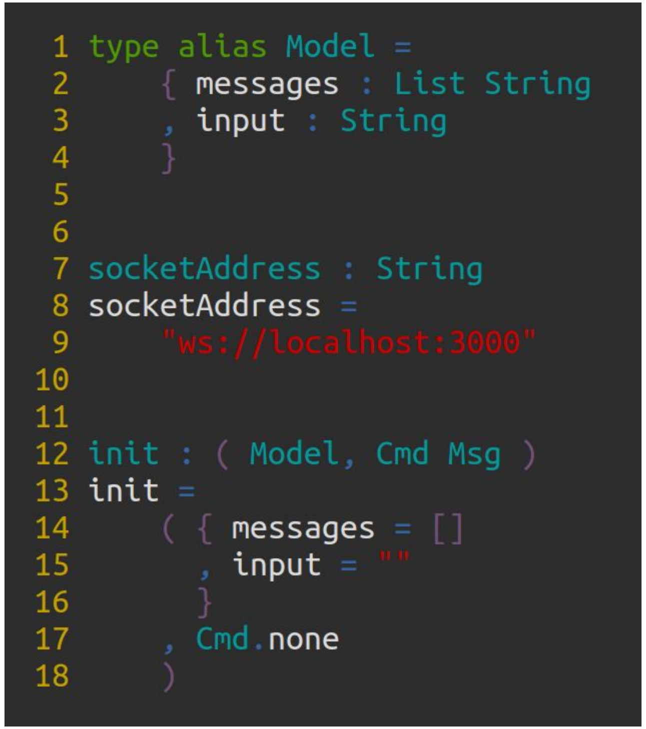
Example: a Chat Client


Let's build this!

http://package.elm-lang.org/packages/elm-lang/html/1.1.0/Html-App

```
program :
    { init : (model, Cmd msg)
    , update : msg -> model -> (model, Cmd msg)
    , subscriptions : model -> Sub msg
    , view : model -> Html msg
    } -> Program Never
```

init : (model, Cmd msg)



```elm
type alias Model =
    { messages : List String
    , input : String
    }


socketAddress : String
socketAddress =
    "ws://localhost:3000"


init : ( Model, Cmd Msg )
init =
    ( { messages = []
      , input = ""
      }
    , Cmd.none
    )
```

There is no “object” in elm, but **aliases** can be defined for particular **records**

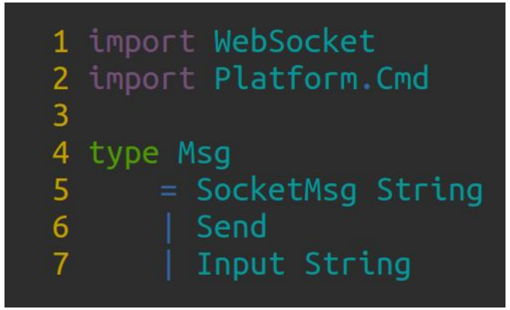
This is a declaration, won’t change… ever

Notice that **Model** is used in the type signature

No command are initially intended

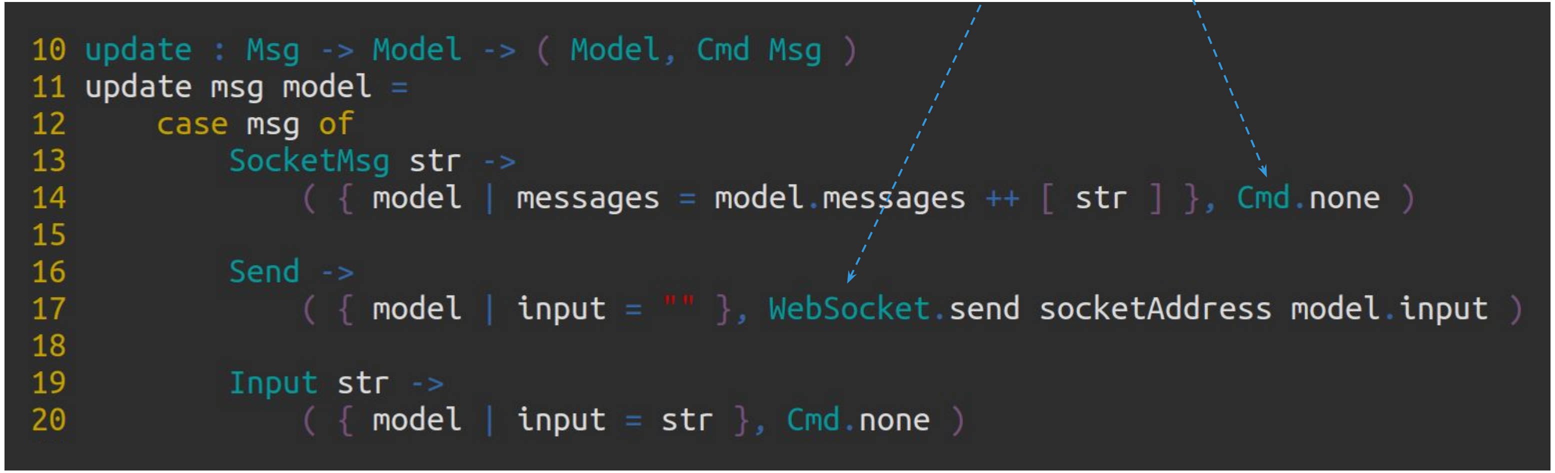
update : msg -> model -> (model, Cmd msg)

Package imports with **explicit** prefixes



```
import WebSocket
import Platform.Cmd

type Msg
    = SocketMsg String
    | Send
    | Input String
```

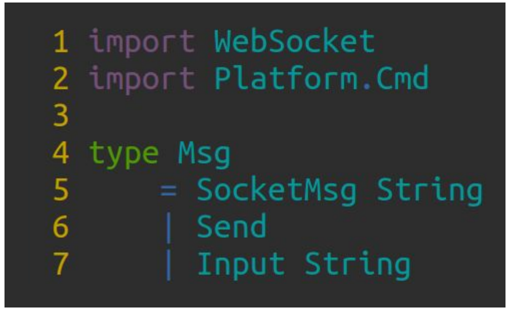


```
update : Msg -> Model -> ( Model, Cmd Msg )
update msg model =
    case msg of
        SocketMsg str ->
            ( { model | messages = model.messages ++ [ str ] }, Cmd.none )

        Send ->
            ( { model | input = "" }, WebSocket.send socketAddress model.input )

        Input str ->
            ( { model | input = str }, Cmd.none )
```

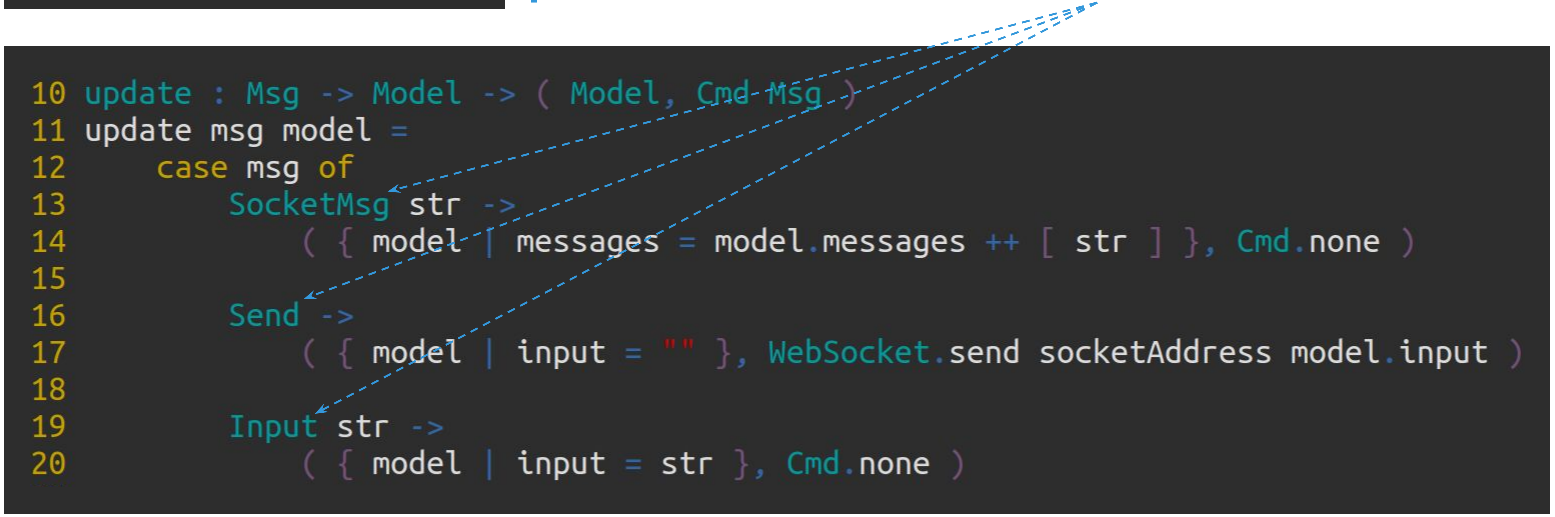
update : msg -> model -> (model, Cmd msg)



```elm
import WebSocket
import Platform.Cmd

type Msg
      = SocketMsg String
      | Send
      | Input String
```

A **union type** defines **messages** which can trigger **transitions** in the application



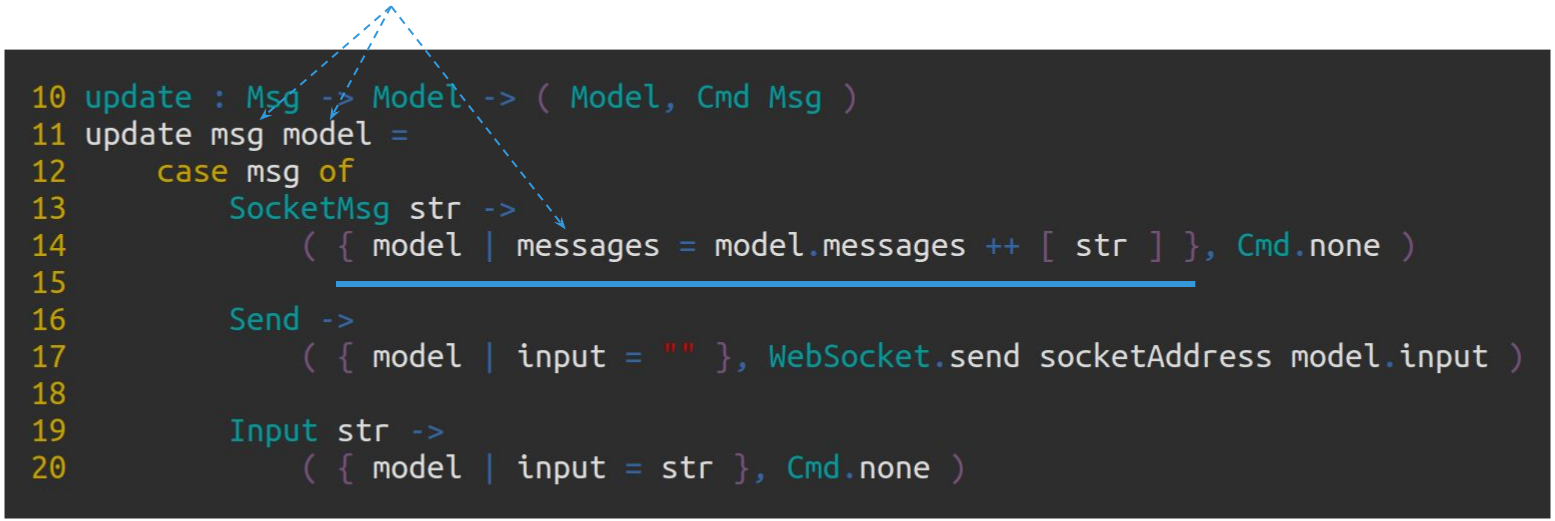
```elm
update : Msg -> Model -> ( Model, Cmd Msg )
update msg model =
    case msg of
        SocketMsg str ->
            ( { model | messages = model.messages ++ [ str ] }, Cmd.none )

        Send ->
            ( { model | input = "" }, WebSocket.send socketAddress model.input )

        Input str ->
            ( { model | input = str }, Cmd.none )
```

`update : msg -> model -> (model, Cmd msg)`

The **model** is **updated** based on the current one and the given **message**



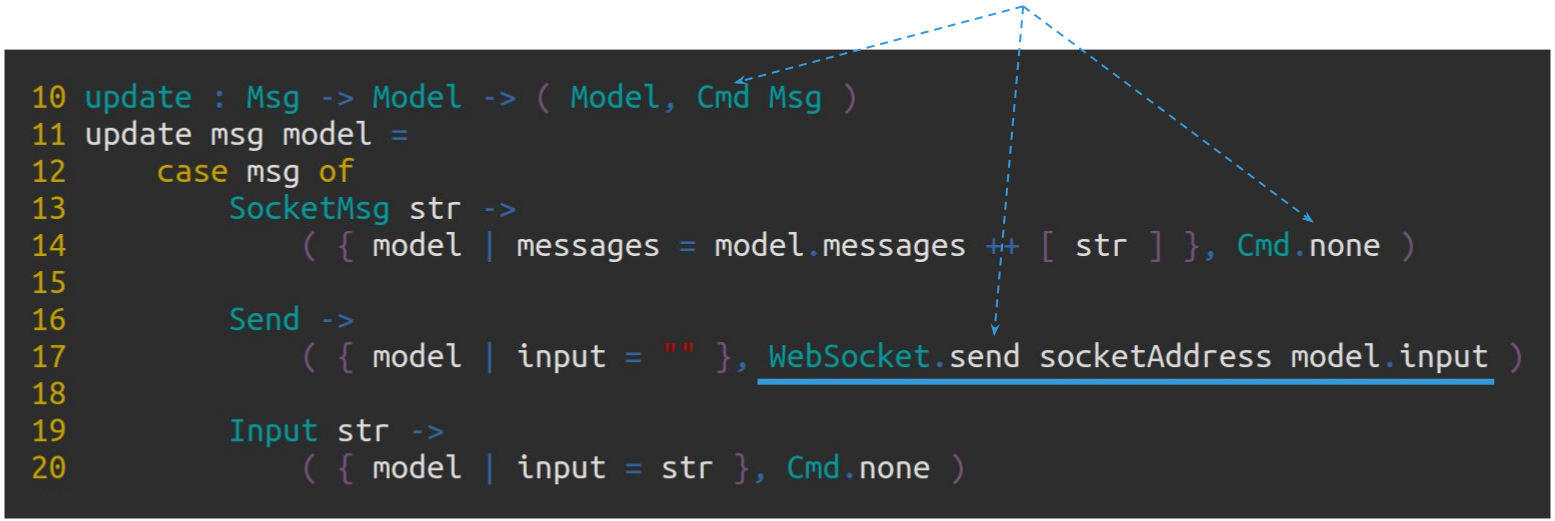
```elm
update : Msg -> Model -> ( Model, Cmd Msg )
update msg model =
    case msg of
        SocketMsg str ->
            ( { model | messages = model.messages ++ [ str ] }, Cmd.none )

        Send ->
            ( { model | input = "" }, WebSocket.send socketAddress model.input )

        Input str ->
            ( { model | input = str }, Cmd.none )
```

update : msg -> model -> (model, Cmd msg)

Commands describe asynchronous actions that may have **side-effects**

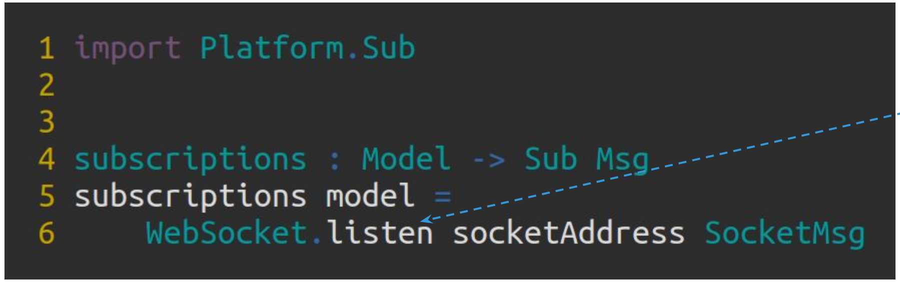


```
update : Msg -> Model -> ( Model, Cmd Msg )
update msg model =
    case msg of
        SocketMsg str ->
            ( { model | messages = model.messages ++ [ str ] }, Cmd.none )

        Send ->
            ( { model | input = "" }, WebSocket.send socketAddress model.input )

        Input str ->
            ( { model | input = str }, Cmd.none )
```

subscriptions : model -> Sub msg

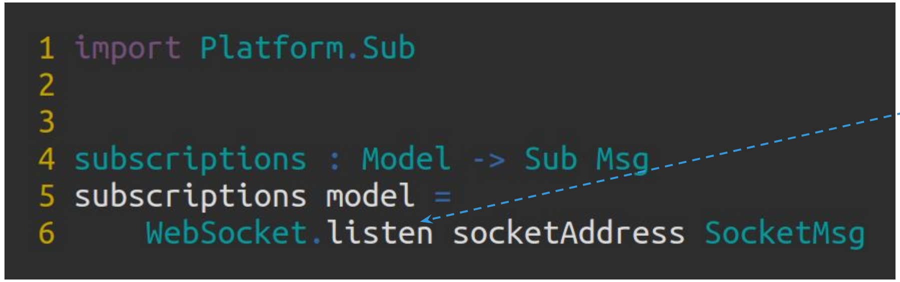


```
import Platform.Sub


subscriptions : Model -> Sub Msg
subscriptions model =
    WebSocket.listen socketAddress SocketMsg
```

A **Subscription** listens to incoming socket messages

subscriptions : model -> Sub msg



```
import Platform.Sub


subscriptions : Model -> Sub Msg
subscriptions model =
    WebSocket.listen socketAddress SocketMsg
```

A **Subscription** listens to incoming socket messages

Commands

Created by us to perform asynchronous operations

Subscriptions

Used to listen to messages emitted sent by other modules

view : model -> Html msg

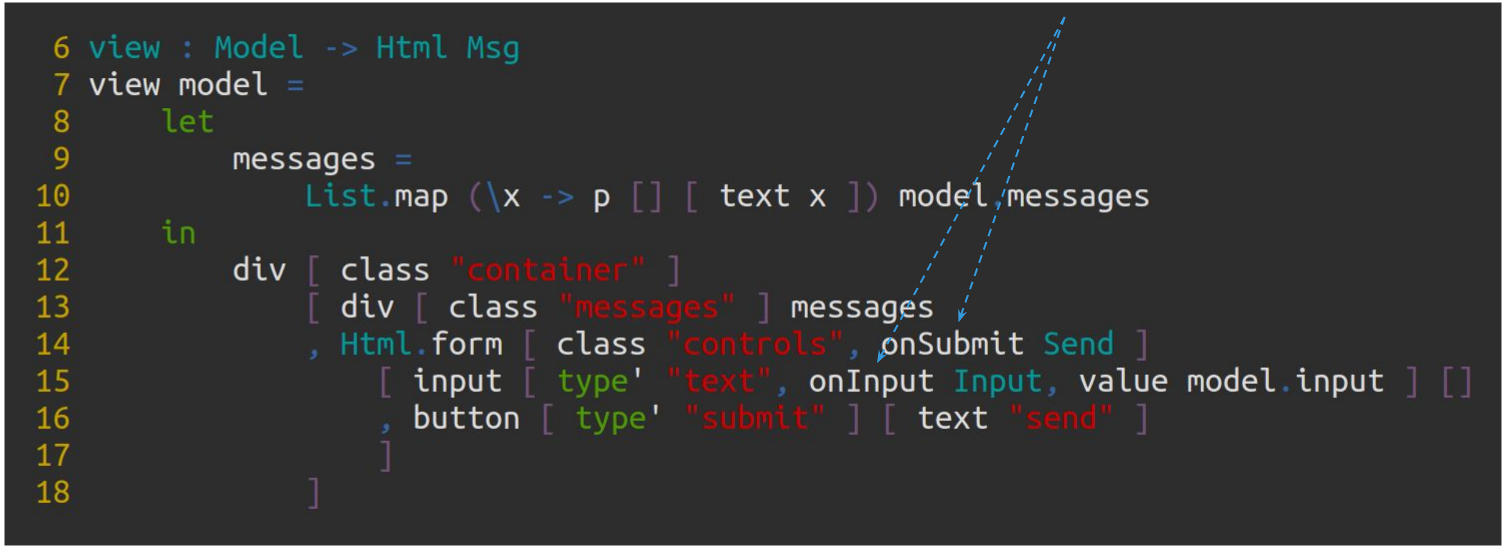
```elm
import Html exposing (..)
import Html.Attributes exposing (..)
import Html.Events exposing (..)
```

Packages can also be imported **implicitly** (optional prefix)

```elm
view : Model -> Html Msg
view model =
    let
        messages =
            List.map (\x -> p [] [ text x ]) model.messages
    in
        div [ class "container" ]
            [ div [ class "messages" ] messages
            , Html.form [ class "controls", onSubmit Send ]
                [ input [ type' "text", onInput Input, value model.input ] []
                , button [ type' "submit" ] [ text "send" ]
                ]
            ]
```

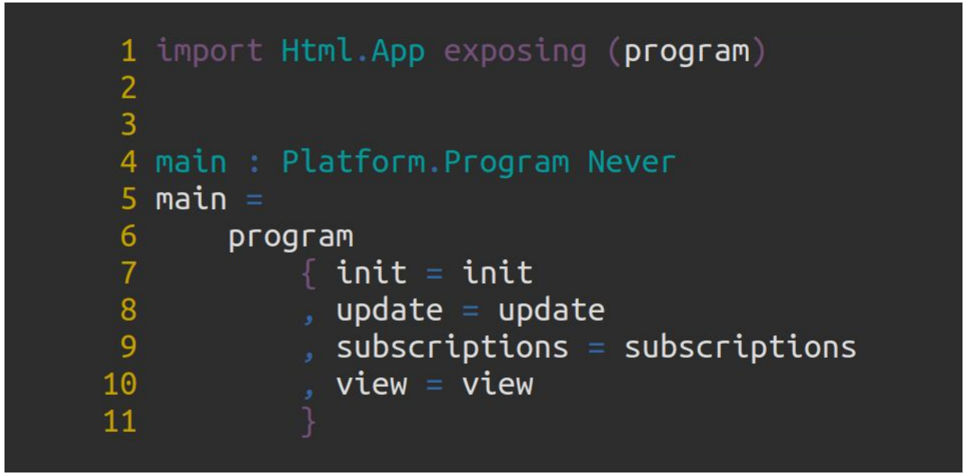
view : model -> Html msg

View interactions generate messages



```
view : Model -> Html Msg
view model =
    let
        messages =
            List.map (\x -> p [] [ text x ]) model.messages
    in
        div [ class "container" ]
            [ div [ class "messages" ] messages
            , Html.form [ class "controls", onSubmit Send ]
                [ input [ type' "text", onInput Input, value model.input ] []
                , button [ type' "submit" ] [ text "send" ]
                ]
            ]
```

```
program :
    { init : (model, Cmd msg)
    , update : msg -> model -> (model, Cmd msg)
    , subscriptions : model -> Sub msg
    , view : model -> Html msg
    } -> Program Never
```



```
import Html.App exposing (program)


main : Platform.Program Never
main =
    program
        { init = init
        , update = update
        , subscriptions = subscriptions
        , view = view
        }
```

76 sloc

67K
(minified)

No
Runtime
Error

Resources

elm-tutorial.org   builtwithelm.co   drboolean.gitbooks.io/mostly-adequate-guide/content

dailydrip.com/topics/elm   elmseeds.thaterikperson.com   elmweekly.nl   frontendnewsletter.com

isRuslan/awesome-elm

@elmlang, @czaplic, @rtfeldman

Dankjewel.