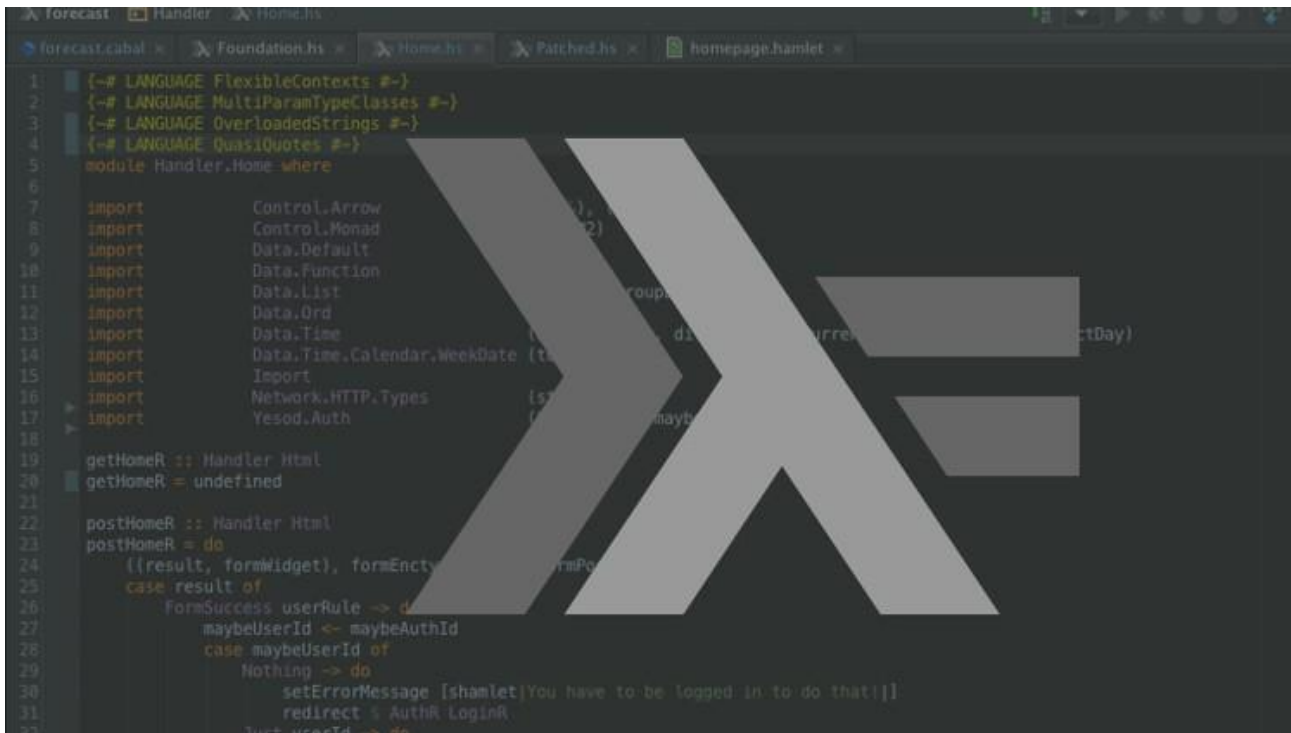


# Introduction to Haskell



```
1  {-# LANGUAGE FlexibleContexts #-}
2  {-# LANGUAGE MultiParamTypeClasses #-}
3  {-# LANGUAGE OverloadedStrings #-}
4  {-# LANGUAGE QuasiQuotes #-}
5  module Handler.Home where
6
7  import Control.Arrow
8  import Control.Monad
9  import Data.Default
10 import Data.Function
11 import Data.List
12 import Data.Ord
13 import Data.Time
14 import Data.Time.Calendar.WeekDate (toDayOfWeek)
15 import Import
16 import Network.HTTP.Types
17 import Yesod.Auth
18
19 getHomeR :: Handler Html
20 getHomeR = undefined
21
22 postHomeR :: Handler Html
23 postHomeR = do
24   ((result, formWidget), formEncType) <- runFormPost
25   case result of
26     FormSuccess userRule -> do
27       maybeUserId <- maybeAuthId
28       case maybeUserId of
29         Nothing -> do
30           setErrorMessage [shamlet|You have to be logged in to do that|]
31           redirect < AuthR LoginR
32         Just userId -> do
```

# Introduction to Haskell

Haskell is a pure functional programming language.

There are not:

- assignments,
- loops,
- side effects,
- explicit memory management.

There are:

- *lazy* evaluation,
- functions as first-order objects,
- static type system,
- automatic type inference.

Haskell is elegant, concise and makes you think differently!