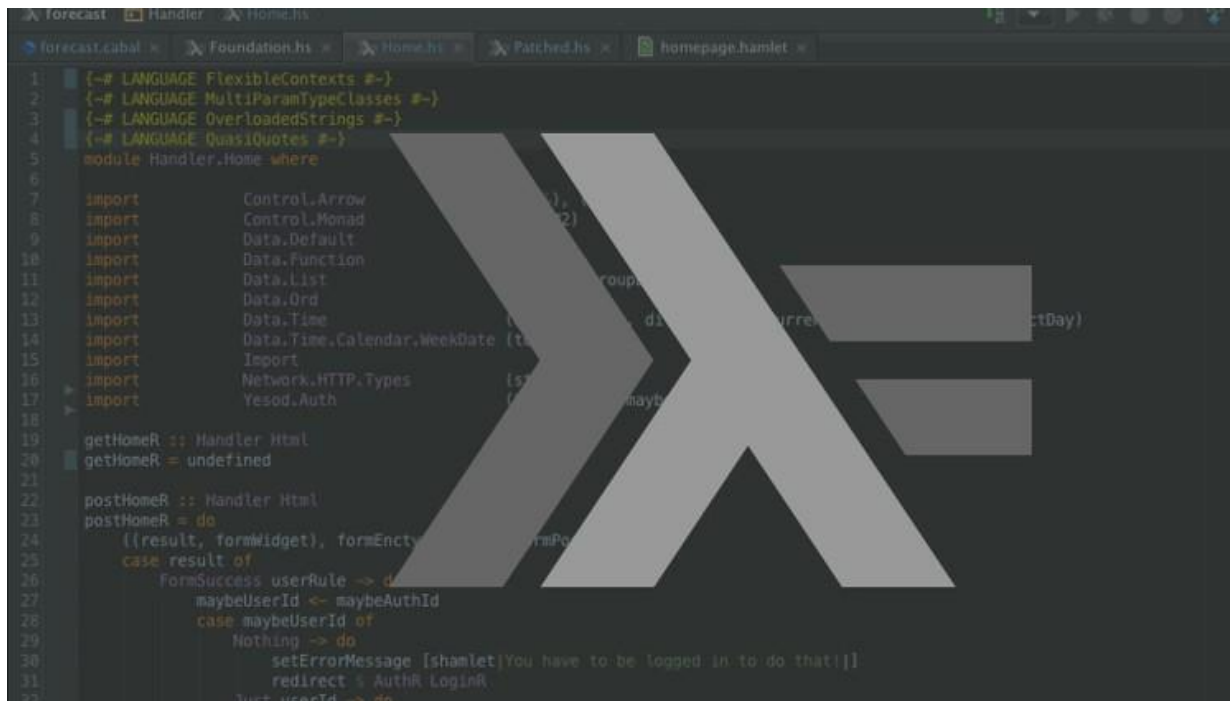


Syntax in Patterns



The screenshot shows a Haskell code editor with a dark theme. The code is as follows:

```
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
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
```

Syntax in Patterns

Pattern decomposition can also be used in the `case`, `where` and `let`.

```
sum list =  
  case list of  
    []      -> 0  
    x:xs    -> x + sum xs
```

```
divMod n m  
  | n < m      = (0, n)  
  | otherwise  = (q + 1, r)  
  where (q, r) = divMod (n - m) m
```

```
firstAndsecond list =  
  let first:second:rest = list  
  in (first, second)
```

Instructor Youtube Channel: Lucas Science

