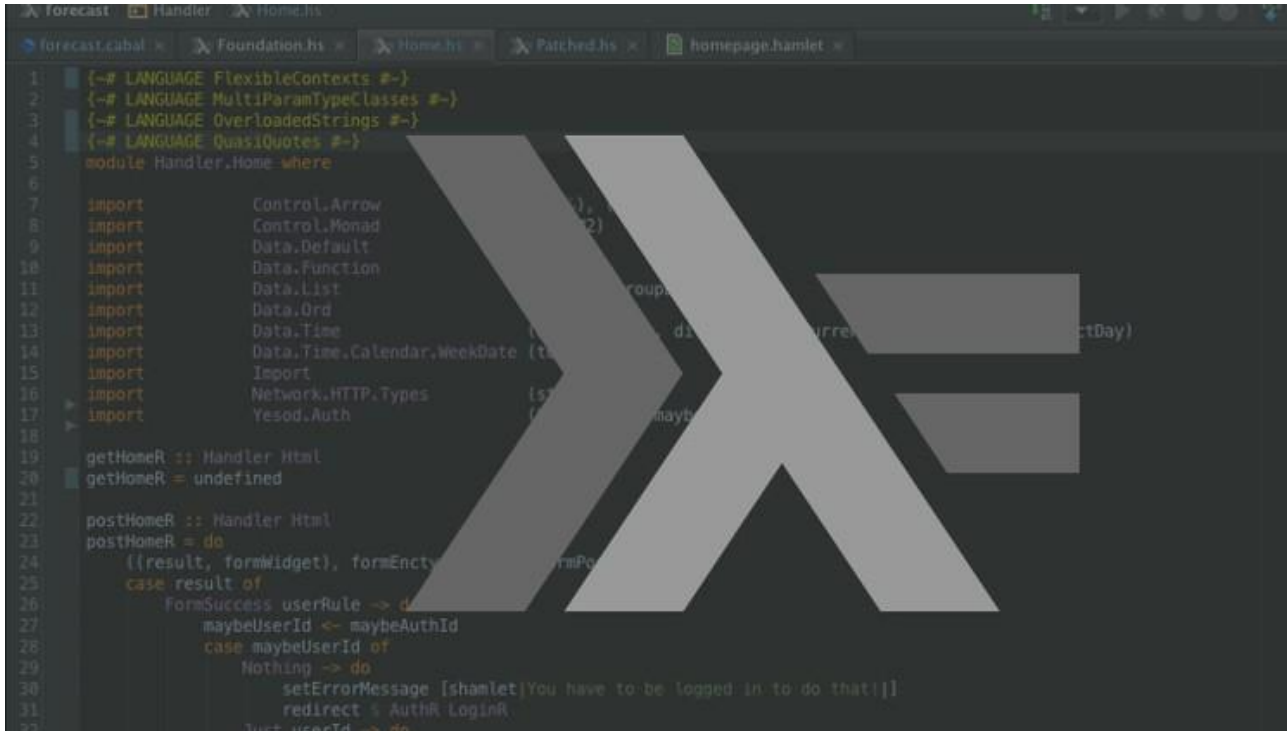


Basic Fundamentals



```
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.Map
13 import Data.Maybe
14 import Data.Time
15 import Data.Time.Calendar.WeekDate
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, formPost) <- 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
```

Expressions

```
λ> 3 + 2 * 2  
↵ 7
```

```
λ> (3 + 2) * 2  
↵ 10
```

```
λ> even 62  
↵ True
```

```
λ> even(62)  
↵ True
```

```
λ> even "Albert"
```

```
λ> div 14 4  
↵ 3
```

Parentheses **not** necessary.

Type error

Type

```
λ> :type 'D'  
↳ 'D' :: Char
```

```
λ> :type "Emma"  
↳ "Emma" :: [Char]
```

```
λ> :type not  
↳ not :: Bool -> Bool
```

```
λ> :type length  
↳ length :: [a] -> Int
```

Factorial

```
factorial :: Integer -> Integer  
  
factorial 0 = 1  
factorial n = n * factorial (n - 1)
```

```
λ> factorial 5  
↵ 120
```

```
λ> map factorial [0..5]  
↵ [1, 1, 2, 6, 24, 120]
```

Summary

- Haskell is pure functional programming language.
- We have made a first taste of its characteristics.
- There are many more things!