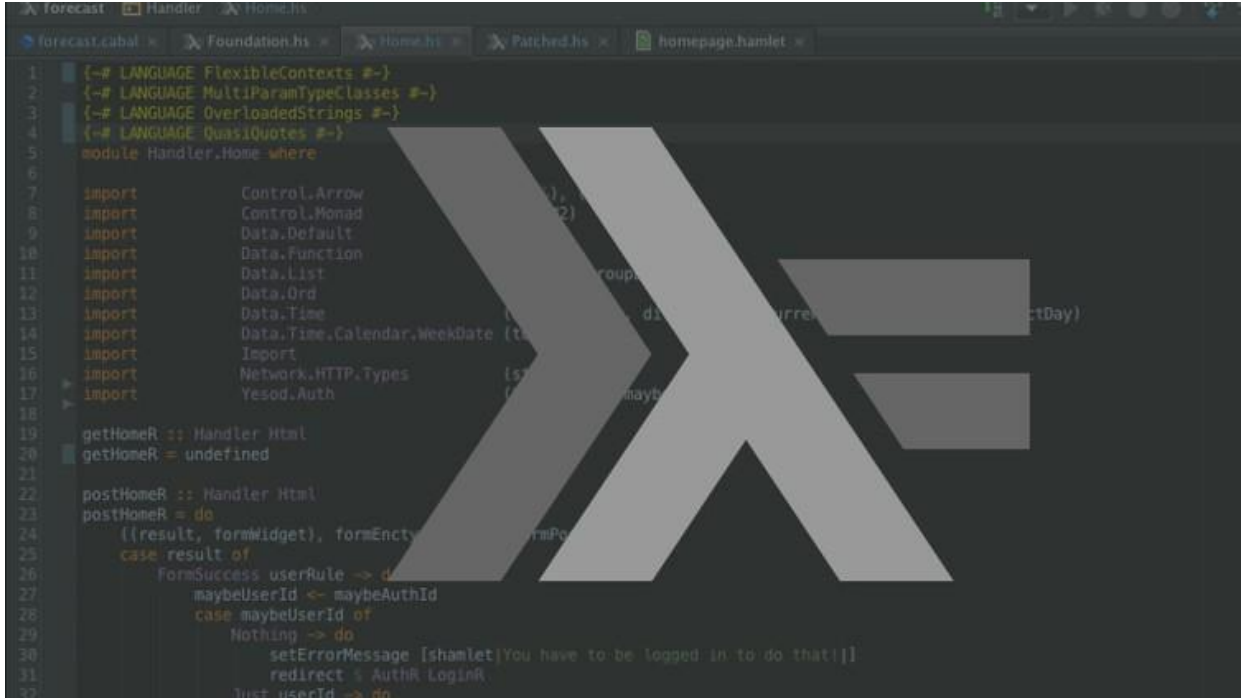
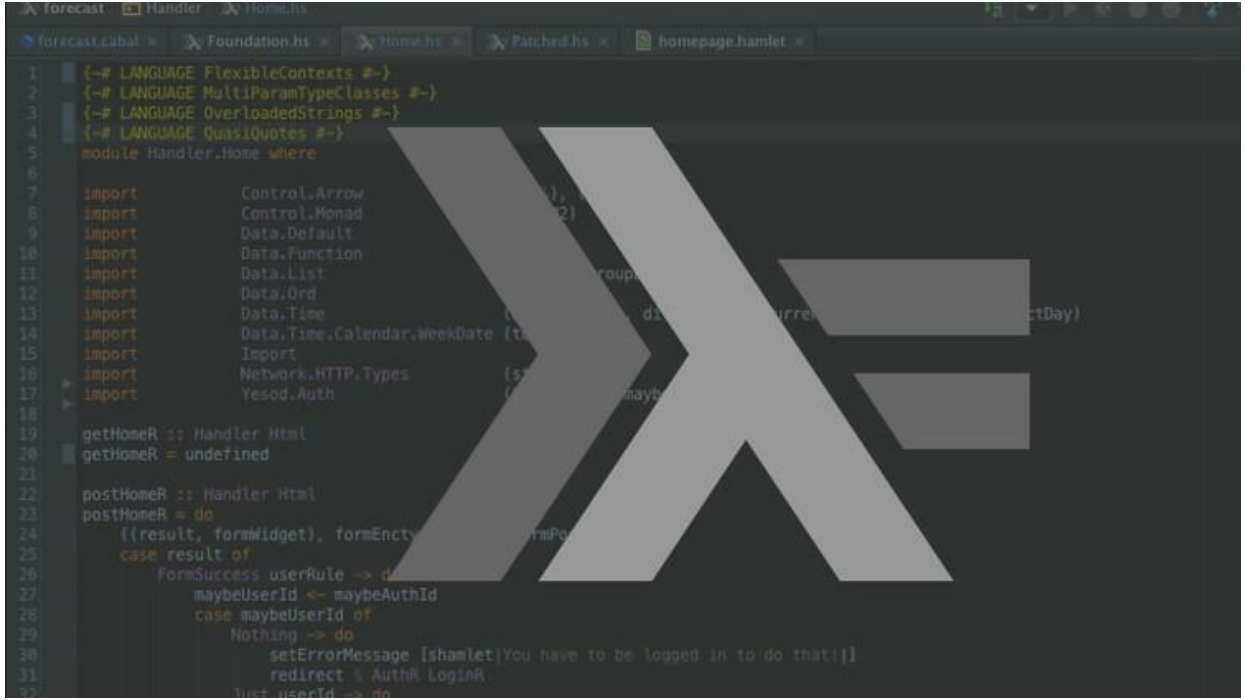
Custom Data Types and Typeclasses



Types and Typeclasses



Types and Typeclasses

- In Haskell, types and typeclasses are both key concepts, but they serve very different purposes.

Types

- A type defines a set of values and how data is structured in a program. It's essentially a way of classifying values into different kinds, such as `Int`, `Bool`, or user-defined.

- Types in Haskell are used to specify what kind of values functions can take as arguments and return as results.

Typeclasses

- A typeclass is more like an interface in other languages. It defines a set of functions that can operate on multiple types, but it doesn't specify the data structure itself.

- Instead, it defines behaviors or capabilities that a type must implement to belong to that class.

- The `Eq` typeclass in Haskell is an excellent example to help understand what a typeclass is and how it works.

Eq

- `Eq` is a typeclass in Haskell that **defines equality for types**.

- Any type that is an instance of the `Eq` typeclass must implement the equality function `==` and its complementary function `/=`.

- In simpler terms, if a type is an instance of `Eq`, it means that values of that type can be compared for equality or inequality.

Types and Typeclasses

- In the following lectures we will define custom data types of our own. Moreover, we will see how we can make a type an instance of the `Eq` typeclass.