Elena Schmitt

WHAT IS HASKELL?

Purely functional language

- No side-effects

Statically typed

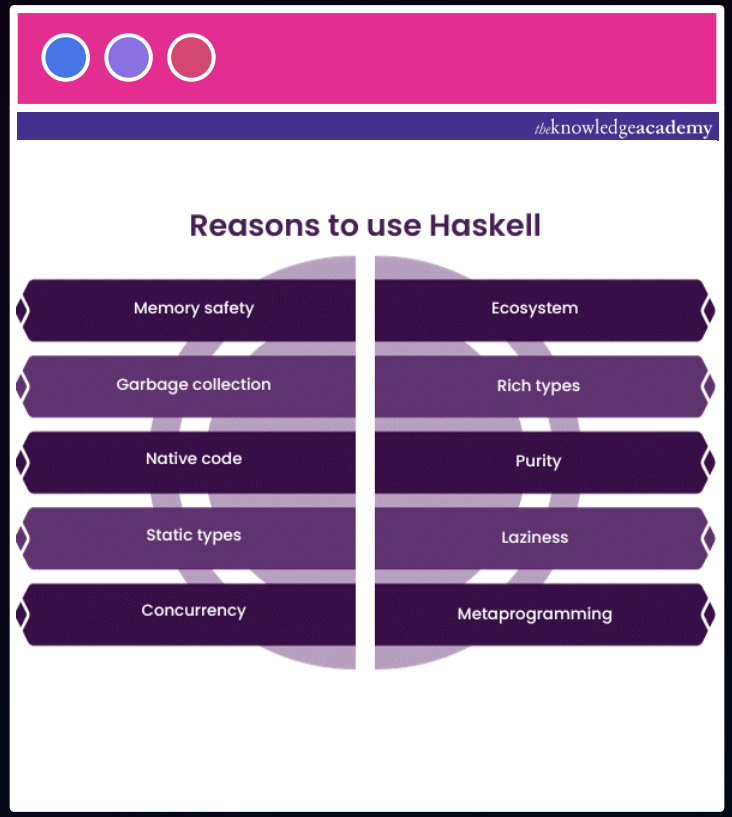
- Types of expressions are checked before the program runs and do not change during execution

Higher-order Functions

- Can take in other functions as parameters and return functions

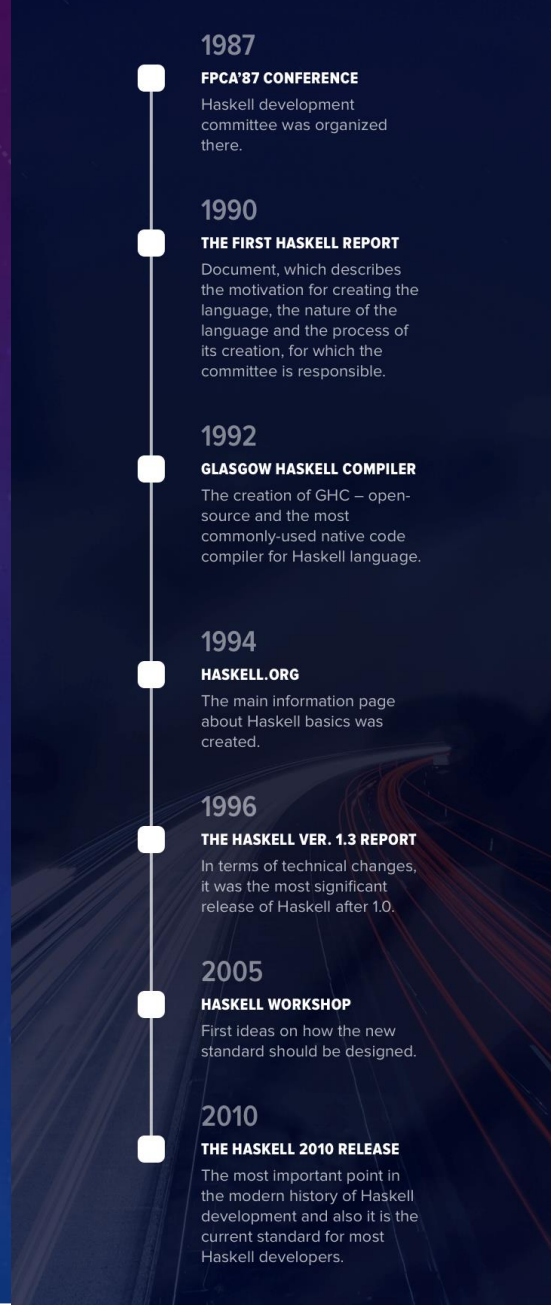
Lazy Evaluation

- Expressions not evaluated until needed



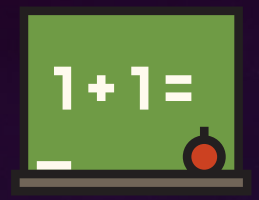
History of Haskell

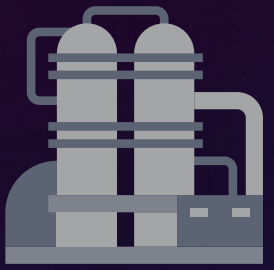
- Named after mathematician Haskell Brooks Curry
- In 1987 a committee was formed to create a common functional programming language
- Haskell 98 released and a standard library was published
- In 2006 work on a successor to Haskell 98 began
- Haskell 2010 is released
- GHC Haskell is the implementation widely used today



START

WHERE IT'S USED





Pure Functional

Strong Type System

Parallel Programming

Reasoning

Efficiency

- **TEACHING**
  - Good for learning programming concepts.
- **RESEARCH**
  - Strength in reasoning, useful for statistics.
  - Ability to handle large datasets.
- **INDUSTRY**
  - Used in areas like aerospace, finance, and healthcare.
  - Good at handling real-time data streams.

Monads

RETURN

```
RETURN :: A → M A
```

BIND

```
(>>=) :: M A → (A → M B) → M B
```

THEN

```
(>>) :: M A → M B → M
```

```
class Monad m where
      (>>=)  :: m a → ( a → m b) → m b
      (>>)   :: m a → m b → m b
      return :: a → m a
```

- m monadic type constructor (Maybe, IO, Reader, etc.).
- a, b type variables- a is the type input to the monad, b is the type input you get out of the monad

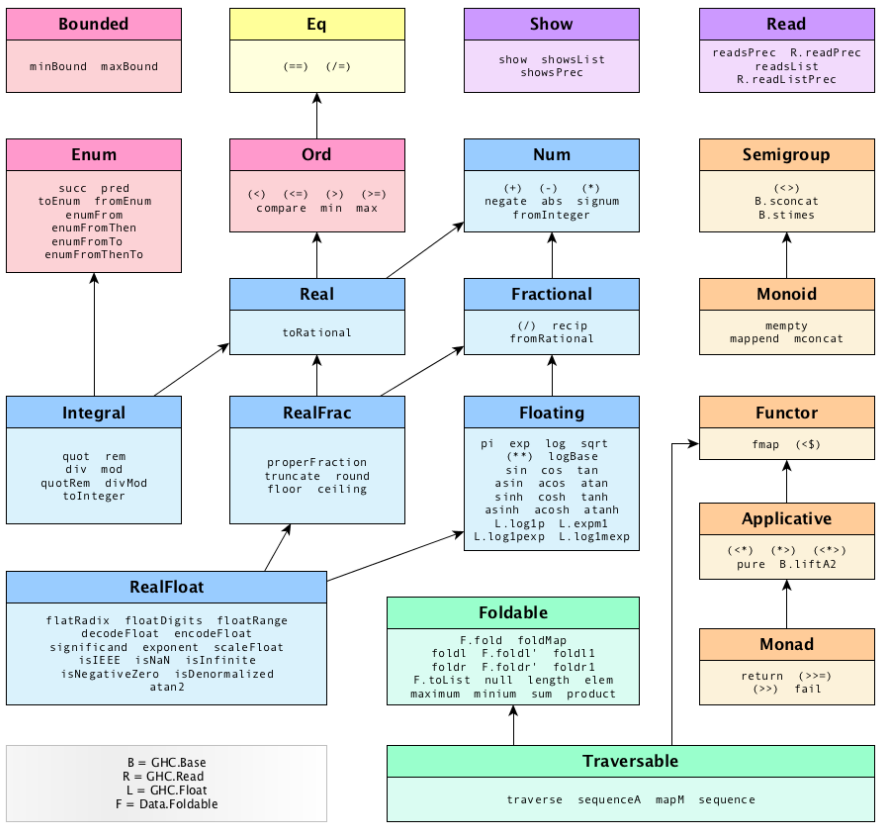
Return: Wraps a value in a monad

Bind: Allows for chaining together monadic operations

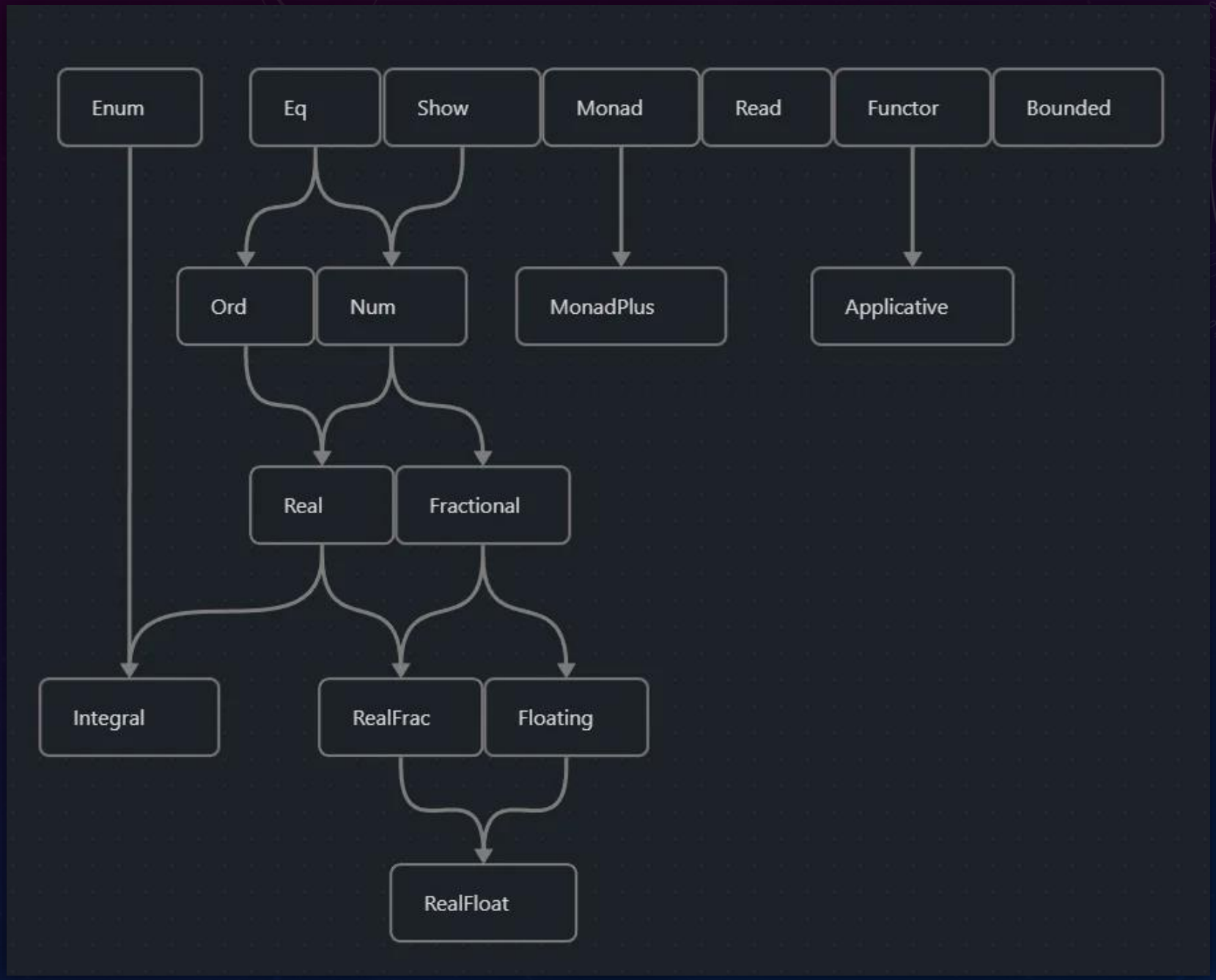
Then: Similar to bind, but ignores the result of the first monadic operation

TYPECLASSES

- Define a set of methods that are shared between multiple types
  - Only able to accept certain types for a function
- Compile-time type restraints
  - Types are checked before running
- Have set operations so that they behave consistently
- Inheritance allows for creating more specific types
- Polymorphism



START


Enum
Eq
Show
Monad
Read
Functor
Bounded
Ord
Num
MonadPlus
Applicative
Real
Fractional
Integral
RealFrac
Floating
RealFloat

Demonstration

START

SOURCES

https://cs.lmu.edu/~ray/notes/introhaskell/

https://scrapingrobot.com/blog/haskell-programing-language/#programming-in-haskell

https://serokell.io/blog/haskell-history

https://medium.com/sourcescribes/what-is-haskell-853b6949a800

https://wiki.haskell.org/Monad