ADAM KASSAM, BHAVIN PATEL

BACKGROUND

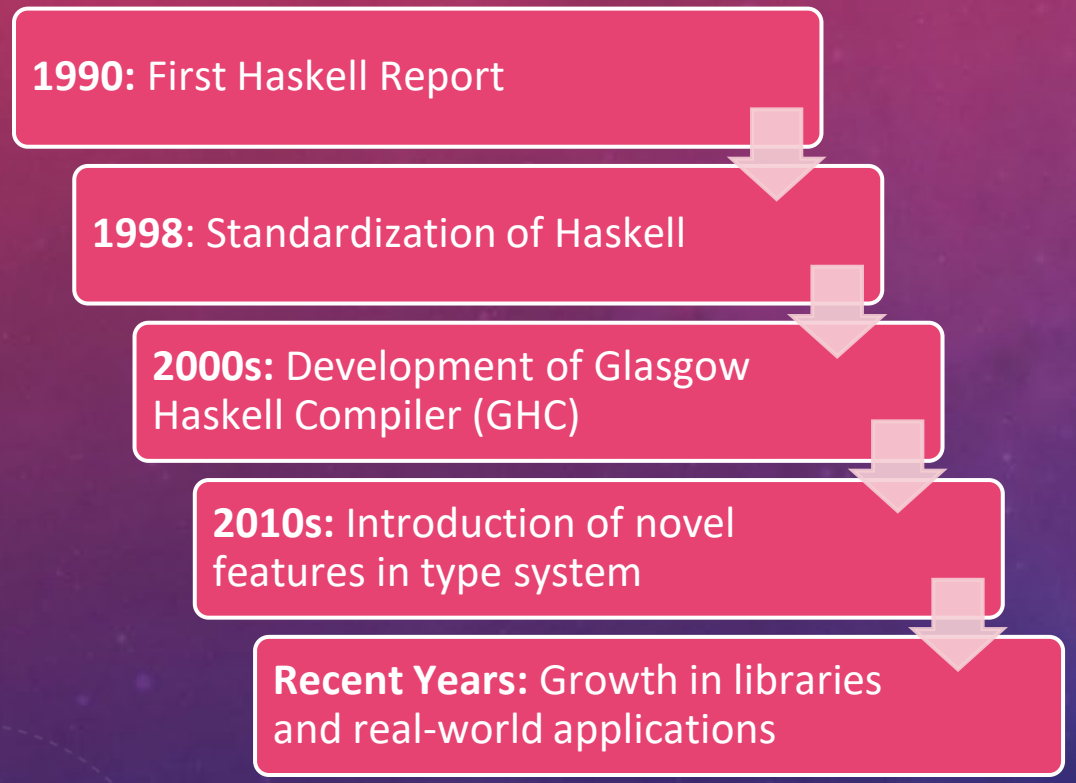
Type: Purely functional, lazy programming language

First Report: 1990

Standardization: Adopted in 1998

Key Features: Advanced type system, lazy evaluation

EVOLUTION AND FEATURES OF HASKELL



- **Type System Evolution:** Continuous improvements, especially in the type system.
- **GHC:** Importance of Glasgow Haskell Compiler in Haskell's development.
- **Community Growth:** Increasing user base and supportive community.
- **Real-World Applications:** Adoption in various domains.

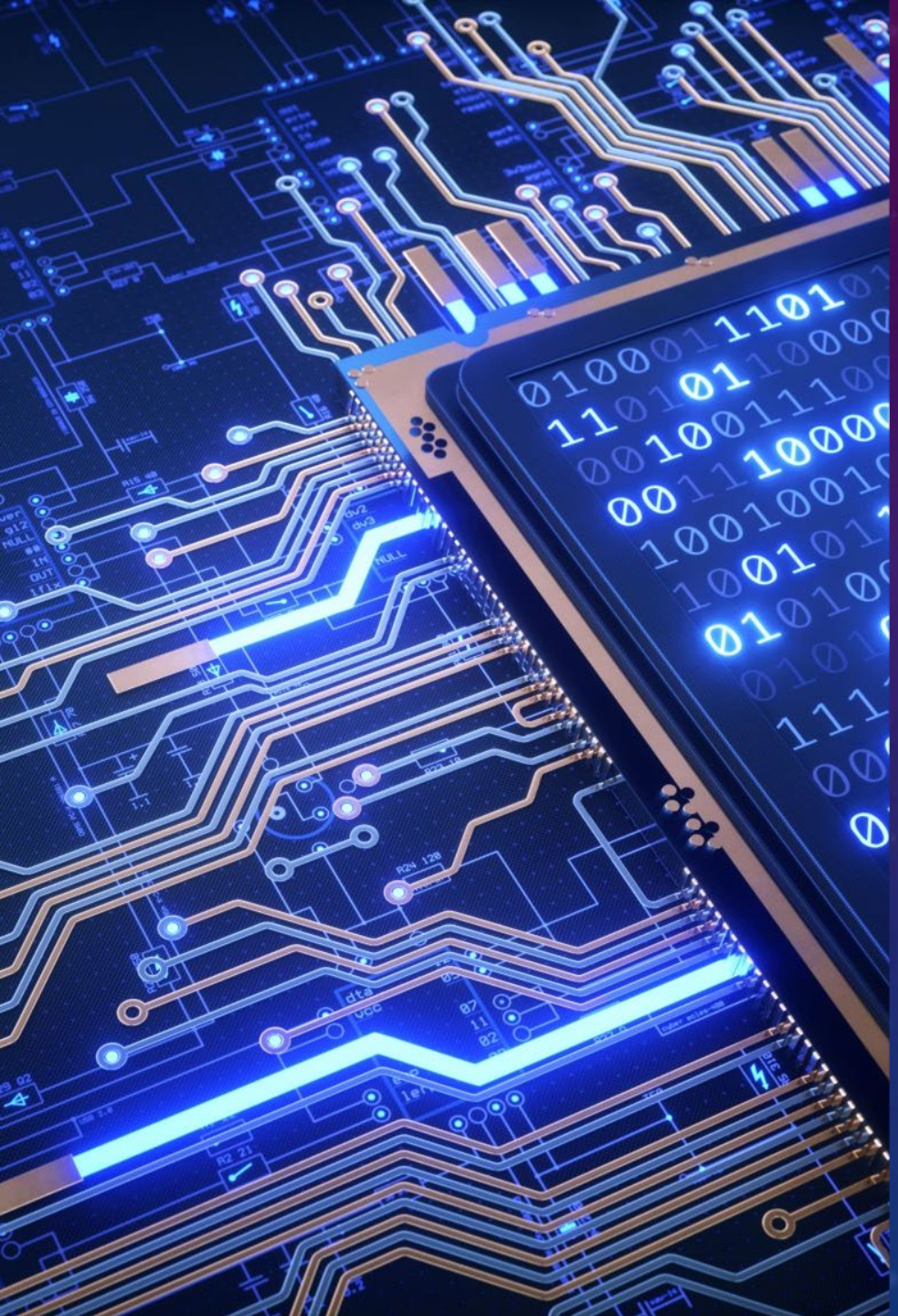
CODE SAMPLE

AverageCalculator.hs > ...

```haskell
-- AverageCalculator.hs
-- Program to calculate the average of a list of numbers

-- Define the 'average' function
average :: [Double] -> Double -- Function type declaration
average xs = sum xs / fromIntegral (length xs)
-- ^ Calculate the average: sum the list and divide by its length

-- Main function: the entry point of the program
main :: IO () -- Indicates an IO action returning nothing (Unit type)
main = do
    putStrLn "Enter a list of numbers separated by spaces:" -- Prompt the user
    input <- getLine -- Read a line from the console
    putStrLn "" -- Print an empty line for spacing
    let numbers = map read . words $ input -- Convert the input string to a list of numbers
    putStrLn $ "Average: " ++ show (average numbers) -- Print "Average: " followed by the calculated average
```



CLASSIFICATION (PARADIGMS)

- **Overview of Major Programming Paradigms:**

1.Imperative Programming: Changing program state through commands.

2.Functional Programming: Computation as evaluation of mathematical functions.

3.Object-Oriented Programming: Data encapsulation in objects and interaction through methods.

4.Logic Programming: Problem-solving using formal logic.

- **Haskell's Classification:**
- Primarily **Functional Programming**:
    - Purely functional nature.
    - Emphasis on immutability and function purity.
- **Influences from Other Paradigms:**
    - **Imperative Programming**: 'do' notation for sequencing actions.
    - **Logic Programming**: Use of pattern matching and list comprehensions.

CLASSIFICATION (DESIGN PRINCIPLES)

Abstraction: Haskell's functions and higher-order functions.

Automation: Type inference in Haskell.

Information Hiding: Haskell's module system.

Orthogonality: Independent functions in Haskell.

Portability: Haskell's platform independence.

Regularity: Consistent syntax and semantics in Haskell.

Simplicity: Haskell's minimalistic yet expressive syntax.

Structure: Correspondence between static program structure and dynamic computation.

Evaluation

- Readability
- Writability
- Reliability
- Cost

CONCLUSION

- Haskell's uniqueness as a purely functional, lazy language.
- Its evolution and continuous improvement in features like the type system.
- Demonstrated practicality in software development, particularly in complex problem-solving.
- Alignment with key programming principles and high marks in readability, writability, reliability, and cost.
- Supportive community, rich ecosystem, and its increasing relevance in diverse applications.

THANK YOU!

SOURCES

- https://www.haskell.org/
- https://wiki.haskell.org/Introduction
- https://haskell.foundation/
- https://www.geeksforgeeks.org/what-is-haskell-programming-language/
- https://stackoverflow.com/questions/1604790/what-is-haskell-used-for-in-the-real-world
- https://serokell.io/blog/companies-that-use-haskell
- https://typeable.io/posts/2019-03-15-do-you-know-where-haskell-is-used.html