Haskell
Purely Functional Programming

http://www.haskell.org/haskellwiki/Haskell_logos
Influences

Lisp
Control structures and recursion control flow, not the sequence of statements, lambda expressions

Scheme
Static scoping, first-class functions

ML
Syntax, static scoping, strongly typed, type inference

Miranda
Polymorphic typing, lazy evaluation and higher-order functions
Timeline

1987 FPCA meeting in Portland, Oregon
   Named for Haskell Curry
1990 Haskell 1.0 Specification Released
1997 Haskell 98 Specification Released
Haskell Objectives

• Suitable for teaching, research, and applications, including building large systems
• Freely available
• Based on ideas that enjoy a wide consensus
• Reduce unnecessary diversity in functional programming languages
Haskell as a Calculator

Infix notation, operator precedence
Variables and Functions

GHC Interactive, version 6.6.1, for Haskell 98.
http://www.haskell.org/ghc/
Type :? for help.

Loading package base ... linking ... done.
Prelude> let x = 4
Prelude> let y = 5
Prelude> let z = x * y
Prelude> z
20
Prelude> let mult a b = a * b
Prelude> mult 6 7
42
Prelude>
Lazy Evaluation

Imperitive Strict Evaluation Psuedo-Code

```
List makeList() {
    List current = new List();
    current.value = 1;
    current.next = makeList();
    return current;
}
```

^ Called by value

Functional Lazy Evaluation Example (Haskell)

```
makeList = 1 : makeList
```

^ Called by name / Called by need
Pairs and Tuples

Heterogeneous entries
Nested entries
Fixed size
Lists

Arbitrary number of elements
Homogeneous entries
Strings

Strings are lists of characters

C-style quoting for character / string
Data Types

Prelude> 5 :: Int
5
Prelude> 5 :: Double
5.0

Prelude> :t 5.0
5.0 :: (Fractional t) => t
Prelude> let q = sqrt 4
Prelude> :t q
q :: Double
Prelude> q
2.0
Prelude> :t "moooooooco"
"moooooooco" :: [Char]
Prelude> :t 'm'
'm' :: Char
Input / Output

Built-In IO Class
getChar
putChar
getLine
putStrLn
...

Input / Output are actions
Actions with specific sequences must be held in lists
Purity and Side Effects

A side effect is an effect that persists after an evaluation is completed.

Pure expressions have no side effects.

Haskell

  Functions do not have side effects.

  Actions have side effects.

Input and output are side effects.
List Comprehensions

Syntactic constructs that denote the results of some operation on elements of lists

Set builder notation from set theory.

\[ \{ (i, j) : i \in \{ 1, 2 \} \land j \in \{ 1, 2, 3, 4 \} \} \]

Hugs> \[ (i,j) | i <- [1,2], j <- [1..4] \] 
\[ [(1,1),(1,2),(1,3),(1,4),(2,1),(2,2),(2,3),(2,4)] \]
Higher Order Functions

map

```
{ i * 2 : i ∈ { 1, 2, 3 } }
Prelude> let times_two x = x * 2
Prelude> map times_two [ 1, 2, 3 ]
[2,4,6]
```

foldl / foldr

```
Prelude> foldl (+) 0 [ 1 .. 10 ]
55
Prelude> 0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10
55
```

filter

```
Prelude> let is_positive x = if x > 0 then True else False
Prelude> filter is_positive [ -1, -2, 2, 5, -13, 6 ]
[2,5,6]
```
Monads

Sequenced operations, such as I/O
Must bind and return
    Return must preserve its argument
    Binding two functions in succession is the same as binding a determinate function
Control

if then else

```haskell
Prelude> let f n = if n <= 1 then 1 else n * f (n-1)
Prelude> f 5
120
```

case

```haskell
let s x =
  case x of
    0 → 1
    1 → 5
    2 → 2
    _ → -1
```

do blocks for sequential operations instead of monads
Haskell Interpreters and Compilers

Hugs
  Interpreted, interactive
GHC / GHCI
  Interpreted, interactive / Compiled
Helium, NHC, Yhc, Gofer, HBC, Jhc, nhc98
Compiling Haskell

```
$ cat Foo.hs
module Main
    where
msg = "Hello world!"
main = putStrLn msg
$ ghc --make Foo.hs -o foo
$ ./foo
Hello world!
$ 
```
Haskell in the Wild

Pugs - perl6 implementation
Frag - first person shooter
DoCon - computer algebra system
AmuZed – graphical UML editor
Learn More

http://haskell.org/

http://learnhaskell.blogspot.com/

Haskell 98 Language and Libraries - The Revised Report available at Rasmusen Library or online at http://www.haskell.org/onlinereport/

Many Haskell libraries are documented at http://www.haskell.org/haskellwiki/Applications_and_libraries
Resources

http://www.haskell.org/haskellwiki/Functional_programming
http://haskell.org/haskellwiki/Introduction_to_Haskell_IO/Actions
http://www.haskell.org/%7Epairwise/intro/section1.html
http://www.cs.nott.ac.uk/~gmh/faq.html
http://homepages.inf.ed.ac.uk/wadler/papers/monads/monads.dvi
http://www.websters-online-dictionary.org/li/list_comprehension.html
http://www.haskell.org/onlinereport/
http://haskell.org/haskellwiki/Haskell_in_practice

Concepts of Programming Languages ~ Robert W. Sebesta