How to Haskell

igorgue.com

Me

Person | Programmer | Politics Junky | Hipster Music

Programming Languages
NERD

Bash | Boo | C | C# | C++ | Clojure | CoffeeScript |
Erlang | Gambas | Go | **Haskell** | Java | JavaScript |
Lisp | Objective-C | PHP | Perl | Python | R | Ruby |
SQL | Scala | SuperCollider | Vala | Visual Basic

What is Haskell?

A standardized, general-purpose **purely functional** programming language, with **non-strict semantics** and **strong static typing**.

Hello, World!

```
main = print "Hello, World!"
```

A Formal Hello

```haskell
-- hello.hs: Saying "Hello, World!"
module Main where

main :: IO ()
main = putStrLn "Hello, World!"
```

What’s Functional Programming?

Wait... What's a Function?

A **function**, in **mathematics**, associates one quantity, the *argument* of the function, also known as the ***input***, with another quantity, the *value* of the function, also known as the ***output***.

Pure Functional Programming follows the same definition as in mathematics. A function will ***output*** the same value if the same ***input*** is passed to it.

And a functional programming language is?

A programming language where ***functions*** are first-class citizens

Cool Haskell Stuff

The only one pure functional programming language

http://www.youtube.com/watch?v=UuamC0T3hv8

No Side-Effects! YAY!

A Side Effect

```
def function(y):
    return y + x

x = 1
print function(1) # => 2
x += 1
print function(1) # => 3
```

Haskell Turn!

```
function y = y + x

x = 1

function 1 -- 2 (This wont be executed by the compiler)

x = x + 1  -- SYNTAX ERROR!

function 1 -- 3 (This will never happen)
```

Lazy Evaluation

Python vs Haskell

```
print range(1, 10000000)[:3] # Python
```

```
print (take 3 [1..10000000]) -- Haskell
```

python range_of_10_mm.py **0.59s** user 0.38s system 97% cpu 0.996 total

runhaskell range_of_10_mm.hs **0.21s** user 0.04s system 90% cpu 0.270 total

```
print (take 3 [1..100000000000000000000000000000000]) -- Haskell
```

runhaskell range_of_many_mm.hs **0.21s** user 0.04s system 92% cpu 0.271 total

Haskell is Dynamic Too!

```haskell
string = "foobar"
integer = 10
float   = 5.34

main = do
       putStrLn ("String: " ++ string)
       putStrLn ("Integer: " ++ show integer)
       putStrLn ("Float: " ++ show float)
```

But, It’s Still Strongly Typed!

```haskell
string :: String
string = "foobar"

integer :: Int
integer = 10

float :: Float
float  = 5.34

main :: IO ()
main = do
       putStrLn ("String: " ++ string)
       putStrLn ("Integer: " ++ show integer)
       putStrLn ("Float: " ++ show float)
```

My Recent Problem Implementation of in_groups_of

I solved it with Haskell... kinda

Problem

Given a **list of n** return a **list of lists of x size**. e.g.:

in_groups_of 2 [1, 2, 3, 4, 5, 6]
[[1, 2], [3, 4], [5, 6]]

Rails Implementation

```ruby
# File activesupport/lib/active_support/core_ext/array/grouping.rb, line
22
def in_groups_of(number, fill_with = nil)
  if fill_with == false
    collection = self
  else
    # size % number gives how many extra we have;
    # subtracting from number gives how many to add;
    # modulo number ensures we don't add group of just fill.
    padding = (number - size % number) % number
    collection = dup.concat([fill_with] * padding)
  end

  if block_given?
    collection.each_slice(number) { |slice| yield(slice) }
  else
    returning [] do |groups|
      collection.each_slice(number) { |group| groups << group }
    end
  end
end
```

Haskell Turn!

```haskell
inGroupsOf :: [a] -> Int -> [[a]]
inGroupsOf [] _ = []
inGroupsOf xs n =
    let (ys, zs) = splitAt n xs
    in ys : inGroupsOf zs n
```

Write Web Apps with it!

http://snapframework.com/

```haskell
igorPage :: Application ()
igorPage = do
    message <- decodedParam "greeting"
    heistLocal (bindString "message" (T.decodeUtf8 message))
$ render "echo"
  where
    decodedParam p = fromMaybe "" <$> getParam p

site :: Application ()
site = route [ ("/",             index)
             , ("/echo/:stuff", echo)
             , ("/igor/:greeting", igorPage)
             ]
       <|> serveDirectory "resources/static"
```

Why Haskell?

BECAUSE HASKELL IS FUN!!!

Thanks!

http://igorgue.com/presentations/howtohaskell.pdf