tacocat

Julie Moronuki

Austin Haskell Meetup

% August 18, 2016

Review

- Sum and product types
- Typeclasses and instances

Stack Project

- this isn't mandatory right now
- stack new palindrome simple
- 'simple' asks for different templates than plain 'stack new'
- edit the .cabal file

Starting the project

```haskell
module Main where

pal :: String -> Bool
pal xs = xs == reverse xs

main :: IO ()
main = do
  string <- getLine
  print $ pal string
```

Handling spaces

- what if someone gives us “taco cat”? WHAT THEN?
- one way to handle that:

```
words :: String -> [String]
-- [String] == [[Char]]
concat :: [[a]] -> [a]
(concat . words) :: String -> [Char]
```

What's that dot, though?

```
(.) :: (b -> c) -> (a -> b) -> a -> c
infixr 9

*Main> concat (words "taco cat")
"tacocat"
*Main> (concat . words) "taco cat"
"tacocat"
```

Handling uppercase

```
toLower :: Char -> Char
```

- so it doesn't work on Strings
- but Strings are just lists

I'm the map

```
map :: (a -> b) -> [a] -> [b]

*Main Data.Char> map toLower "JULIE"
"julie"
```

Imports

```
module Main where

import Data.Char
import qualified Data.Char
import qualified Data.Char as DC
import Data.Char (toLower)
```

Using toLower

- add a `map` of `toLower` to `pal2`

Let's talk $

```
*Main> :info ($)
($) :: (a -> b) -> a -> b   -- Defined in ‘GHC.Base’
infixr 0 $
```

More $

- often used instead of parentheses
- with function composition, it means the functions compose before being applied to the argument