PROGRAMMING IN HASKELL

Chapter 8.2 Function Application

$ as function application

$ is the function application operator

```
($) :: (a -> b) -> a -> b
```

```
f $ x = f x
```

$ as function application

It’s function application but:

- normal function application has high precedence, $ has low precedence
- normal function application is left associate, e.g., f a b c === ((f a) b) c
- $ is right associative

$ as function application


```
*Main> (^2) 4 + 3
19

*Main> (^2) $ 4 + 3
49
```

Improved syntax with $

Most often it's a convenience that lets us write fewer parentheses.

Example:

```
sum (map sqrt [1..130])
```

Is better written as:

```
sum $ map sqrt [1..130]
```

when $ is encountered, expression on right is used as parameter to function on left

More examples


```
sqrt (3+4+9)
```

```
sqrt $ 3+4+9
```

```
*Main> sum (filter (> 10) (map (*2) [2..10]))
80
*Main> sum  $ filter (> 10) (map (*2) [2..10])
80
*Main> sum  $ filter (> 10) $  map (*2) [2..10]
80
```

Another example


```
*Main> (10*) $ 3
30
*Main> ($ 3) (10*)
30
```

```
map ($ 3) [(4+), (10*), (^2), sqrt]
```

[7.0,30.0,9.0,1.7320508075688772]

How does this work?

- expression on right is used as parameter to function on left


ANY
QUESTIONS?