haskell

cons

In haskell **consing** is done via the infix operator (:).

For example:

(cons 1 (cons 2 (cons 3 null)))

is the same as

1:2:3:[]

in haskell

types

- Haskell has types! Love the typechecker
- Some of these types you are familiar with, for example:
- Integer
- Double
- Char
- Bool
- and their associated list types like [Bool]

More types

- Haskell also supports **polymorphism**
- For example the identity function looks like

$$\text{id} :: a \rightarrow a$$

$$\text{id } x = x$$

- Polymorphic types have type variables, in the above example 'a' is the type variable.
- When writing down your own types you can use whatever character sequence you want as type variables

Arrow types

- Haskell also has types you are potentially unfamiliar with
- For example:

$$\text{Integer} \rightarrow \text{String} \rightarrow \text{Bool}$$

  is a type

associativity

- Racket has no syntax
- We needed to write our programs as an **abstract syntax tree**
- This was good because there was no ambiguity about how things **associate**
- It was bad because we had to get used to reading

parentheses-encrusted code))]))))]))

What is associativity?

- Associativity is the order in which things execute in the absence of parentheses
- Parentheses make it clear how things associate
- **Left associativity** is when we execute statements left-to-right, for example if ~ is a binary operator, then

$$1 \sim 2 \sim 3 \sim 4 \sim 5$$

  Is computed in the order

$$((((1 \sim 2) \sim 3) \sim 4) \sim 5)$$

What is associativity?

- **Right associativity** the same as left associativity except from right-to-left. So the previous example would be computed as

$$(1 \sim (2 \sim (3 \sim (4 \sim 5))))$$

Arrow types associativity

- It is **important** to remember that in haskell arrow types are **right associative**
- The arrow type

$$t1 \rightarrow t2 \rightarrow t3 \rightarrow t4 \rightarrow t5$$

  is **implicitly** parenthesized like

$$(t1 \rightarrow (t2 \rightarrow (t3 \rightarrow (t4 \rightarrow t5))))$$

- However the type checker will remove parentheses when they are not required to enforce correct precedence