Haskell

Deian Stefan

(adopted from my & Edward Yang’s CSE242 slides)
Why Haskell?
The great ideas [Haskell]

Expressive power (say more with less)

- First-class functions
- Type inference
- Monads
- Pattern matching
- Exception handling
- Continuations

Reliability and reuse

- Type polymorphism
- Modules
- Type classes
- Objects & inheritance

Cross-cutting concerns

- Memory management
- Concurrency
What is Haskell?

a typed, lazy, purely functional language
Haskell is statically-typed
Haskell is *statically-typed*

- Everything has a type
- Everything must make sense at compile time
  - Unlike JavaScript where \( f(x) \) with \( f=\text{undefined} \) will not complain until you actually evaluate \( f(x) \)
- Is JavaScript typed?
  - A: yes, B: no
Why is this cool?
Why is this cool?

• Removes whole classes of bugs

• Address bugs early vs. after they have been triggered
  ➤ Prevent weird errors from creeping up on you
  ➤ Important for safety, security, and compositionally

• Easier to optimize and write faster code
  ➤ You can remove your typeof checks; compiler can do fast things. V8 relies on types to makes things fast!
Haskell is functional

• This means no “side-effects”?  
  ➤ A: yes, B: no
Haskell is functional
Haskell is **functional**

- Support for high-order, first-class functions
- Meaning of programs centered around:
  - evaluating expressions
  - not executing instructions
Haskell is pure
Haskell is pure

• Expressions (e.g., functions) don’t have “side effects”

➤ Is JavaScript pure? A: yes, B: no

• Everything is immutable: mutation is a side-effect!

• What does it mean for an expression to not have side-effects?

➤ In scope where $x_1, \ldots, x_n$ are defined all occurrences of $e$ (where $FV(e) = \{x_1, \ldots, x_n\}$) have the same value
Why is this cool?

Don’t take it from me, take it from Backus
Why is this cool?

• Algebraic laws: equational reasoning & optimizations
  ➤ Can always replace things that are equal, $\lambda$ calculus!

• Easier to think about
  ➤ e.g., don’t need to worry if $x$ changed after calling $f$

• Parallelism
  ➤ Can evaluate expressions in parallel!
Haskell is lazy
Haskell is lazy

• You don’t evaluate an expression until its result is absolutely necessary: in contrast to JavaScript
  ➤ Remember: call-by-name

• Haskell’s evaluation strategy is called call-by-need
  ➤ Because of the other properties: you actually only evaluate an expression once and cache the result
Why is this cool?
Why is this cool?

• Can define your own control structures using functions
  ➤ E.g., defining if-then-else is much easier in Haskell can be done naturally; less so in JavaScript; why?

• Can define infinite data structures
  ➤ E.g., infinite lists, trees, etc.
  ➤ Can solve general problem and then project solution
Haskell is a committee language
Why is this interesting? [SPJ]
Why is this interesting? [SPJ]
Why is this interesting? [SPJ]
Why is this interesting? [SPJ]
Why is this interesting? [SPJ]
intro.hs