Haskell
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
Haskell is statically-typed

- Everything has a type
- Everything must make sense at compile time
  - Unlike JavaScript where f(x) with f=undefined will not complain until you actually evaluate f(x)
- Is JavaScript typed?
  - A: yes, B: no
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  yes, but at runtime (dynamically)
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 make things fast!
Haskell is **functional**

- Does functional imply no "side-effects"?
  - A: yes, B: no
Haskell is **functional**

- Does functional imply 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

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

➤ Is JavaScript pure? A: yes, B: no
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!
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!
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 a scope where \( x_1, \ldots, x_n \) are defined, all occurrences of \( e \) (where \( \text{FV}(e) = \{x_1, \ldots, x_n\} \)) 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, λ 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
  - Can you cache results in JavaScript? A: yes, B: no
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

➢ Can you cache results in JavaScript? A: yes, B: no
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 and 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]

Successful research languages
Why is this interesting? [SPJ]
Why is this interesting? [SPJ]
goto intro.hs