Reading Circle
Why FP Matters

"Well-structured software is easy to write and to debug, and provides a collection of modules that can be reused to reduce future programming costs."

John Hughes

The Plot

Modules are important

They depend on means of "glue"

FP offers two more glues:
higher-order functions and
lazy evaluation

Higher-Order Functions

Examples:
fold (reduce, inject, collect)
sum, product, every, some, append,
reverse, map, ... on lists and trees

-> localize knowledge about the details
of representation

Lazy Evaluation

infinite lazy lists (1D and 2D) and trees

Examples:
approximations, improvements for fixed-point combinators, differentiation, integration, game tree minimax

-> avoid "fusion" confusion

Discussion

Higher-Order Functions

Does it only apply to FP?
(Strategy, Method Object, *Function Pointers)

Have we covered all advantages?
(growth with stabilized core)

Did we cover all constraints?

Lazy Evaluation

Does it only apply to FP?
(opt-in, opt-out, streams, iterators)

Have we covered all advantages?
(extending the solution space)

Did we cover all constraints?

Modules

Have we covered all advantages?
(incremental development, dependency graph, contract)

Functional Programming

Have we covered all or even the most important advantages?
(enforce contract through the type system, strong inference through expressions instead of statements)

Paper Critique

Claims, logical cohesion, strong vs weak points, line of argumentation, conclusion from examples, counter-examples, actual measurements