pubCrawl

for when one just won’t do
The Team

- Matthew Dean
- Sireesh Gururaja
- Kevin Mangan
- Alden Quimby
The Overview

● Easy sharing of computation load across any number of machines
● Focus on list manipulation
● Type inferred for elegance and ease of use
● Essentially a distributed scripting language
The Origin

“a tour taking in several pubs or bars, with one or more drinks at each.”
The Hello World

print(“hello world!”);
The (cooler) Hello World

numsToLetters = a -> {
    myLetters = "abcdefghijklmnopqrstuvwxyz ";
    return myLetters[a];
};
mynums = [7, 4, 11, 11, 14, 26, 22, 14, 17, 11, 3];
print(List.map(mynums, numsToLetters) ^ "!");
The (distributed) Hello World

umsToLetters = a -> {
    myLetters = "abcdefghijklmnopqrstuvwxyz ";
    return myLetters[a];
};

mynums = [7, 4, 11, 11, 14, 26, 22, 14, 17, 11, 3];
result = distribute(mynums, numsToLetters);
print(List.map(result, x -> { return x[0]; })) ^ "!";
The Types

- Numbers
- Booleans
- Characters
- Collections
- Functions
- Objects
The Type Inference

myFunc = (f,a) -> { return f(a); };

aNumber = myFunc(x -> { return x+1; }, 5);
aString = myFunc(x -> { return x ^ " second"; }, "first");

print(aNumber); // 6
print(aString); // “first second”
The Type Inference

unwrap = (o) -> { return List.map("hey", o.a.b[0]); };
Algorithm 1 Type Inference

1: procedure INFERTYPES(ast)
2: \hspace{1cm} sast ← ANNOTATEAst(ast)
3: \hspace{1cm} constraints ← COLLECTCONSTRAINTS(sast)
4: \hspace{1cm} substitutions ← UNIFY(constraints)
5: \hspace{1cm} sast ← APPLYSUBSTITUTIONS(sast, substitutions)
6: \hspace{1cm} return sast
7: end procedure
The Structure
The Implementation in Java

● Everything is an “X”
● PCObjects and PCLists
● IPCFunctions
● Java RMI
Java RMI (Remote Method Invocation)
The Lessons Learned

- Respect the work
- Stay in the loop
- Communicate
- Distribute
The Demos