Inside Fdb.Script

A quick overview of a pure functional programming language

Why creating a new language

- Narrow domain
- Runtime optimisations
- Increased readability
- Learning aspect

Design Assumptions

- Fully functional (no variables or state)
- Declarative parallel execution
- Pipeline data processign
- Portability

Example.fdbs run.sh

```
1 module Example
2
3  2 + 2 * 2
4
```

```
public java.lang.Object invoke(com.jpbnetsoftware.fdbscript.runtime.InvokeContext);
  Code:
     0: ldc2_w        #15                 // double 2.0d
     3: invokestatic  #22                 // Method com/jpbnetsoftware/fdbscript/runtime/RuntimeMethods.createNumber:
     6: ldc2_w        #15                 // double 2.0d
     9: invokestatic  #22                 // Method com/jpbnetsoftware/fdbscript/runtime/RuntimeMethods.createNumber:
    12: ldc2_w        #15                 // double 2.0d
    15: invokestatic  #22                 // Method com/jpbnetsoftware/fdbscript/runtime/RuntimeMethods.createNumber:
    18: invokestatic  #26                 // Method com/jpbnetsoftware/fdbscript/runtime/RuntimeMethods.mul:(Ljava/la
    21: invokestatic  #29                 // Method com/jpbnetsoftware/fdbscript/runtime/RuntimeMethods.add:(Ljava/la
    24: areturn
```

Hello World

```
1 module Example
2
3 fac = f(n):
4     (n == 1) -> 1
5     _ -> self(n - 1) * n
6
7
8 fac(4)
```

```
ublic java.lang.Object invoke(com.jpbnetsoftware.fdbscript.runtime.InvokeContext);
 Code:
    0: aload_1
    1: checkcast     #18                 // class com/jpbnetsoftware/fdbscript/runtime/InvokeContext
    4: ldc           #10                 // String n
    6: checkcast     #8                  // class java/lang/String
    9: invokestatic  #24                 // Method com/jpbnetsoftware/fdbscript/runtime/RuntimeMethods.getVal
g/Object;
   12: dconst_1
   13: invokestatic  #28                 // Method com/jpbnetsoftware/fdbscript/runtime/RuntimeMethods.create
   16: invokestatic  #32                 // Method com/jpbnetsoftware/fdbscript/runtime/RuntimeMethods.isEqua
   19: checkcast     #34                 // class java/lang/Boolean
   22: invokevirtual #38                 // Method java/lang/Boolean.booleanValue:()Z
   25: ifeq          37
   28: nop
   29: dconst_1
   30: invokestatic  #28                 // Method com/jpbnetsoftware/fdbscript/runtime/RuntimeMethods.create
   33: goto          106
   36: nop
   37: nop
   38: aload_1
   39: checkcast     #18                 // class com/jpbnetsoftware/fdbscript/runtime/InvokeContext
   42: aload_1
   43: checkcast     #18                 // class com/jpbnetsoftware/fdbscript/runtime/InvokeContext
   46: ldc           #40                 // String self
   48: checkcast     #8                  // class java/lang/String
   51: invokestatic  #24                 // Method com/jpbnetsoftware/fdbscript/runtime/RuntimeMethods.getVal
g/Object;
   54: checkcast     #42                 // class com/jpbnetsoftware/fdbscript/runtime/IInvokable
   57: iconst_1
```

Factorial

```
1 module Example
2
3 fac2 = f(n): reduce(
4         range(2, n + 1),
5         1,
6         f(): $acc * $e)
7
8 fac2(4)
```

Factorial v2

Features

- Lists and objects - [1, 2, 3], { prop: “value” }
- List concatenation - [1, 2, 3] @ [4, 5] yields [1, 2, 3, 4, 5]
- Function parameter injection - f(): $acc * $e
- Infinite lists - range(1, 99999999)

Future

- parallel execution - ||f(24) + ||f(88)
- object extension - {prop: “v”} @ {prop2: “v”}
- Runtime optimisations
- bytecode optimisations

Thank you!

- https://github.com/burzyk/FdbScript.Compiler