JScheme : A Scheme Interpreter Embedded Within Java Source Code

Jeff Sember

CPSC 511 Term Project, Fall 2007

Problem

- mixing programming languages

Problem

- mixing programming languages
- multiple compilers

Problem

- mixing programming languages
- multiple compilers
- multiple file types

Our Approach

- Embed source in Java comments

Our Approach

- Embed source in Java comments
- JScheme compiler

Our Approach

- Embed source in Java comments
- JScheme compiler
- JSRuntime interpreter

JScheme Language

A subset of scheme

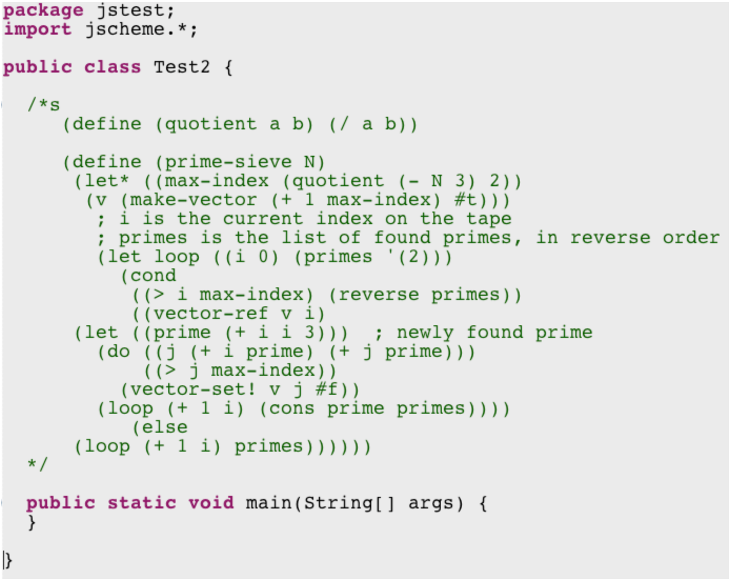
```
<program> ::=  <elem>*
<elem> ::= <exp> | <def> | <import> | ( begin <elem>+ )
<import> ::= #import <file:string>
<def> ::= ( define <id> <exp> )
 | ( define ( <id> <formal:id>* ) <body> )
 | ( define ( <id> . <varformals:id> ) <body> )
 | ( define-datatype <name:id> <predicate:id> <dt-var>* )
<body> ::= <def>* <exp>+
<dt-var> ::= ( <variant:id> <dt-field>* )
<dt-field> ::= ( <field:id> <predicate:expr> )
<lit> ::= <boolean>  | <number> | <character> | <string>
 | <quotation>
<boolean> ::= #t | #f
<character> ::= #\<any character>  | #\space  | #\newline
<quotation> ::= '<datum>  | (quote <datum>)
<datum> ::=  <boolean>  | <number> | <character> | <string>
 | <symbol> | <list> | <vector>
<symbol> ::= <id>  | <keyword>
<list> ::= ( <datum>* )  | ( <datum>+ . <datum> ) | ' <datum>
<vector> ::= #( <datum>* )
```

JScheme Language

A subset of scheme

`*`	`+`	`-`
`/`	`<`	`<=`
`=`	`>`	`>=`
`add1`	`append`	`boolean?`
`cadr`	`car`	`cdr`
`char->integer`	`char?`	`cons`
`currbindings`	`display`	`equal?`
`eqv?`	`expt`	`foldl`
`foldr`	`integer->char`	`length`
`list`	`list->vector`	`list-of`
`list-ref`	`list?`	`make-vector`
`map`	`member`	`newline`
`not`	`null?`	`number->string`
`number?`	`pair?`	`printf`
`procedure?`	`reverse`	`set-car!`
`set-cdr!`	`string->symbol`	`string-append`
`string-length`	`string-ref`	`string?`
`sub1`	`symbol->string`	`symbol?`
`vector`	`vector->list`	`vector-fill!`
`vector-length`	`vector-ref`	`vector-set!`
`vector?`	`write-char`	

Embedding JScheme source



```
package jstest;
import jscheme.*;

public class Test2 {

  /*s
     (define (quotient a b) (/ a b))

     (define (prime-sieve N)
      (let* ((max-index (quotient (- N 3) 2))
       (v (make-vector (+ 1 max-index) #t)))
        ; i is the current index on the tape
        ; primes is the list of found primes, in reverse order
        (let loop ((i 0) (primes '(2)))
          (cond
           ((> i max-index) (reverse primes))
           ((vector-ref v i)
      (let ((prime (+ i i 3)))  ; newly found prime
        (do ((j (+ i prime) (+ j prime)))
            ((> j max-index))
          (vector-set! v j #f))
        (loop (+ 1 i) (cons prime primes))))
           (else
      (loop (+ 1 i) primes))))))
  */

  public static void main(String[] args) {
  }

}
```

Embedding JScheme source

Storing annotations in comments

- `javadoc`

Embedding JScheme source

Storing annotations in comments

- `javadoc`
- ESC Java

Embedding JScheme source

Storing annotations in comments

- `javadoc`
- ESC Java
- ...many others?

Embedding JScheme source

- run `jscomp`, the JScheme compiler

Embedding JScheme source

- run `jscomp`, the JScheme compiler
- scans Java source

Embedding JScheme source

- run `jscomp`, the JScheme compiler
- scans Java source
- compiles JScheme code

Embedding JScheme source

- run `jscomp`, the JScheme compiler
- scans Java source
- compiles JScheme code
- inserts JSRuntime field

Embedding JScheme source

- run `jscomp`, the JScheme compiler
- scans Java source
- compiles JScheme code
- inserts JSRuntime field
- ✔ `-d` option

Embedding JScheme source

- run `jscomp`, the JScheme compiler
- scans Java source
- compiles JScheme code
- inserts JSRuntime field
- ✓ `-d` option
- ✓ backups

Embedding JScheme source

```
package jstest;
import jscheme.*;

public class Test2 {

  /*s
     (define (quotient a b) (/ a b))

     (define (prime-sieve N)
      (let* ((max-index (quotient (- N 3) 2))
       (v (make-vector (+ 1 max-index) #t)))
        ; i is the current index on the tape
        ; primes is the list of found primes, in reverse order
        (let loop ((i 0) (primes '(2)))
          (cond
           ((> i max-index) (reverse primes))
           ((vector-ref v i)
      (let ((prime (+ i i 3)))  ; newly found prime
        (do ((j (+ i prime) (+ j prime)))
            ((> j max-index))
          (vector-set! v j #f))
        (loop (+ 1 i) (cons prime primes))))
           (else
      (loop (+ 1 i) primes))))))
  */

  public static void main(String[] args) {
  }

  //[500
  static JSRuntime rt = new JSRuntime(
          "WonX3/R02U0HhAFiwAsAl6fb5dY8XbaXAW5AO4xvpd3kMh7QAtgB7fC63LKXx/S3HLACsN9vOCANgMvT"
        + "7fIc4APIy/ayfF4G2Mtj+lvekpfNALg83S4D1AB7eUx/y1vzMj0EGL/dcwBgexuAV3Vels9favi83BaX"
        + "5a/7W76ew8Pj8ks9p5fn9NdcPn6p5/TynP6ilud0mUsdtoe31IZRACMEsXFXtgnB2IggbEQgAGBLbBJF"
        + "MEHZChtxbYJEMJuYbUIANinbhOBsRFA2YhgAAAAANmMQAAAAANBW2JhsE4K0YdkmBGojhgAAAICNCAyw"
        + "WUJga2zItQnRJbBNGwAAAADahhmbgDQIAAAAbIWNuTYB2gobVGWbELiNCNhGBAbYrGwTgrdBQQOArbBB"
        + "VbYJAdysAG1EwICtsBmRIG5YtgmB2oiAbUTANmIYAAAAALbChlybAAbIzck2IVAbEbCNCCJwc7JNCNRG"
        + "BHIjgghshU3KNiFwGxHIjQgMsFEXsBU25so2IZgbFqCNCORGCABgK2xItgnB2pRsEwK1EUMAAADARgQM"
        + "2IxsEwK6KUHcmKABAAAAt8RGWNkmBGtTsk0I1EYMAQAAABsRMGAzggYAAAAAAAAA");
  //]500
}
```

Java ⟶ JScheme

`JSRuntime` object methods:

- `SNode eval(String src)`:
  compiles string, evaluates result

```
public static void main(String[] args) {
  rt.eval("(display (prime-sieve 30))");
}
```

Java $\longrightarrow$ JScheme

`JSRuntime` object methods:

- `SNode eval(String src)`:
  compiles string, evaluates result

```java
public static void main(String[] args) {
  rt.eval("(display (prime-sieve 30))");
}
```

- `SId define(SNode value)`:
  stores JScheme object in environment, returns id

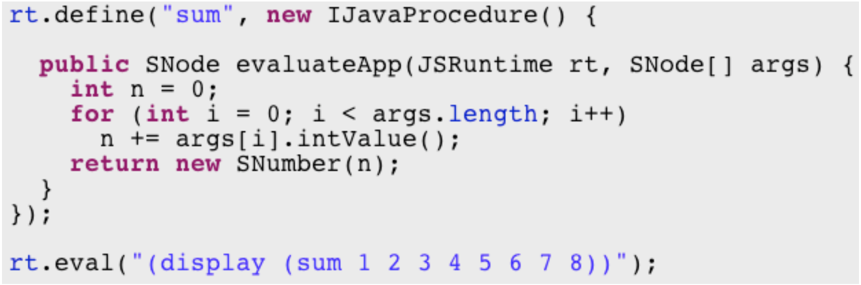
```java
SNode id = rt.define(rt.eval("(prime-sieve 50)"));

rt.eval("(printf \"The first 50 primes are ~s~n\" "+id+")");
```

JScheme $\longrightarrow$ Java

`JSRuntime` object methods:

- `void define(String name, IJavaProcedure proc)`: binds name to JScheme procedure implemented in Java



```java
rt.define("sum", new IJavaProcedure() {

  public SNode evaluateApp(JSRuntime rt, SNode[] args) {
    int n = 0;
    for (int i = 0; i < args.length; i++)
      n += args[i].intValue();
    return new SNumber(n);
  }
});

rt.eval("(display (sum 1 2 3 4 5 6 7 8))");
```

Results

- ✔ Simple: run `jscomp`, use `rt` object

Results

- ✔ Simple: run `jscomp`, use `rt` object
- ✔ Obfuscation

Results

- ✔ Simple: run `jscomp`, use `rt` object
- ✔ Obfuscation
- ✔ Java $\Longleftrightarrow$ JScheme

Results

- ✔ Simple: run `jscomp`, use `rt` object
- ✔ Obfuscation
- ✔ Java $\Longleftrightarrow$ JScheme
- ✘ Slow(er)

Results

- ✔ Simple: run `jscomp`, use `rt` object
- ✔ Obfuscation
- ✔ Java $\Longleftrightarrow$ JScheme
- ✘ Slow(er)
- ✔✘ IDE support

Future Enhancements

- continuations

Future Enhancements

- continuations
- tail recursion

Future Enhancements

- continuations
- tail recursion
- more numeric types

Future Enhancements

- continuations
- tail recursion
- more numeric types
- more efficient closures

Questions