

JavaFX Script

Weiqi Gao
St. Louis Java Users Group
July 12, 2007

About Me

- Principal Software Engineer at OCI
 - Java programmer since 1998 (1.1.4)
 - Member of St. Louis JUG steering committee since 2003
 - Like to explore programming languages
 - Write a weblog at <http://www.weiqigao.com/blog/>
-
-

JavaFX Script, What Is It?

- Announced at JavaOne 2007, JavaFX is a new Sun product family
 - JavaFX Script is a member of the family
 - created by Christopher Oliver of Sun
 - will leverage the ubiquity of Java
 - will deliver high-impact rich media and content
 - to desktop, mobile devices, set-top boxes
 - JavaFX Mobile is another member of the family
 - Java on a Linux based open source phone
-
-

JavaFX Script: Characteristics

- declarative
 - statically typed
 - first-class functions
 - list comprehensions
 - incremental dependency-based evaluation
 - calling to Java APIs
 - packages, classes, inheritance
 - separate compilation and deployment units
-
-

Why?

- It is fun
- It feels easy
- It feels robust
- It feels powerful
- It feels productive
- It promises wide delivery options
- Ignoring “why didn't you do this in JRuby?”—
Priceless

Getting Started

- Official Website (tar ball, svn, mailing lists, forums, language spec, tutorials)
<http://openjfx.dev.java.net>
 - Christopher Oliver's weblog (back ground articles, demos, news) <http://blogs.sun.com/chrisoliver/>
 - Community wiki (world editable)
http://jfx.wikia.com/wiki/Main_Page
 - Currently under an evaluation license, will be released as Open Source
-
-

Getting JavaFX Script

- From java.net OpenJFX project page
 - via svn
 - via .tgz or .zip
- Tar balls are made from svn
 - every two to three weeks
 - three so far
- New features are added in each drop



A Look Inside

```
$ tar zxvf OpenJFX-200706181411.tgz
[...]
```

```
$ ls
branches  LICENSE  README  tags  trunk
```

```
$ ls trunk
bin  demos  doc  lib  src  www
```

```
$ ls trunk/bin
javafx.bat  javafx.sh  javafxstart.bat
  javafxstart.sh
```

```
$ ls trunk/demos
demo  javafxpad  projects  README  studiomoto
  tesla  tutorial
```

```
$ ls trunk/lib
Filters.jar  javafxrt.jar  swing-layout.jar
```

Hello World

```
$ cat hello.fx
```

```
import java.lang.System;  
System.out.println("Hello, world");
```

```
$ javafx.sh hello.fx
```

```
compile thread: Thread[AWT-  
  EventQueue-0,6,main]
```

```
compile 0.01
```

```
Hello, world
```

```
init: 0.036
```

JavaFX: The Language

- A very rough language reference for JavaFX is available from the OpenJFX project on java.net
 - Many clarifications are sought and given on the mailing list
 - Some questions on the mailing list resulted in tightening up of the language
 - I found it useful to peruse through the JavaFX library source files
-
-

Packages and Imports

```
package com.weiqigao.jugdemo;  
import java.lang.System as Sys;
```

- The package statement has the same meaning as in Java.
- The import statement has additional functionalities
 - aliasing class name
 - can appear anywhere
 - importing a JavaFX file searches and runs the file

Classes

- JavaFX has four basic types
 - String
 - Boolean
 - Number
 - Integer
- User can define classes

```
public class Foo {  
}
```

Attributes

- Classes can have attributes, member functions and member operations

```
public class Foo {  
    attribute str: String;  
    attribute boo: Boolean?;  
    attribute num: Number*;  
    attribute int: Integer+;  
}
```

Functions and Operations

- Classes can have member functions and member operations

```
public class Foo {  
    public function fun(input:  
        String): String;  
    public operation opr(input:  
        Number): Number;  
}
```



Initializers

- Attribute, member function and member operation initializers are defined outside the class definition (like in C++)

```
attribute Foo.str = "init";
function Foo.fun(input) {
    return "{input}{input}";
}
operation Foo.opr(input) {
    return 2 * input;
}
```

Objects

- Objects are instantiated with JavaFX object literal notation

```
Foo {  
    str: "Hi"  
    boo: true  
    num: [3.14, 6.28]  
    int: [1024]  
}
```

```
Foo {str: "Hi", int: [9216]}
```

Variables, Named Instances

- A variable is introduced by the `var` keyword
- Variable types are inferred from initializer

```
var str = "Hi"; // String  
var nums = [1..10]; // Number*
```

- Named instances are global

```
INS:Foo = { str: "Hi" };
```

Double Meaning of var

- Inside an object literal, var: names the current instance (odd syntax)

```
class Foo { attribute bar: Bar; }  
class Bar { attribute foo: Foo; }
```

```
var foo = Foo {  
  var: me  
  bar: Bar { foo: me }  
}
```

Functions

- Functions may contain several variable declarations followed by one return statement
- Functions are first-class citizens

```
function addN(n) {  
    return function (x) = x + n;  
}
```

```
println(addN(5)(10));
```

Operations

- Operations may contain more than one statements
- Operations have side effects

```
operation foo(i) {  
    println(i);  
    println(i*i);  
}
```

```
foo(10);
```

Expressions

- Relational operators
 - ==, <>, <, >, <=, >=
 - Boolean operators
 - and, or, not
 - Arithmetic operators
 - +, -, *, /, %, +=, -=, *=, /=, %=
 - Other operators
 - sizeof, indexof, if/then/else, select, foreach, new, opr(), instanceof, this, bind, :, [], format as, <<>>, {}, (expr), reverse, [1,3..99]
-
-

Statements

- Statement may appear at the top level or inside operations
 - if/else if/else, while, try/catch/finally, throw, break, continue and return are like those in Java, but braces are required, and anything can be thrown or caught
 - do statements farm work off the EDT (everything else is on EDT)
 - do later is like invokeLater
-
-

Statements

- for statement is more powerful and supports list-comprehension style syntax

```
for (i in [1..3], j in [1..i] where
    (i + j) % 2 == 0) {
    println("i: {i}, j: {j}");
}
```

Sequences (Arrays)

- Arrays (to be renamed Sequences) is a key data structure of JavaFX

```
var a = [1, 2, 3, 4, 5, 6, 7];  
var i = a[3]; // by index  
var j = a[. % 2 == 0]; // [2, 4, 6]  
var k = a[indexof . % 2 == 0];  
var aa = [a, a]; // flattened  
var b = ([] == null); // true  
var s = sizeof a; // 7
```

List Comprehension

- List comprehension can be done in two ways: foreach and select

```
var ns = [1, 2, 3, 4, 5];  
var ss = ["3", "4", "5", "6", "7"];  
var c = select n from n in ns, s in  
    ss where s == "{n}"; // [3, 4, 5]  
var d = foreach (n in ns, s in ss  
    where s == "{n}") n; // [3, 4, 5]  
var e = select n*n from n in  
    [1..10]; // [1, 4, 9, ... 100]
```

Sequence Manipulation

```
var a = [1];  
insert 2 into a; // [1,2]  
insert 0 as first into a; //  
  [0,1,2]  
delete a[1]; // [0,2]  
insert 3 before a[1]; // [0,3,2]  
delete a[. == 0]; // [3,2]  
insert 4 after a[. > 2]; // [3,4,2]  
delete a; // []
```

Classes Revisited

- Bidirectional relation

```
class Foo {  
    attribute bar: Bar inverse foo;  
}  
class Bar {  
    attribute foo: Foo inverse bar;  
}  
var f = Foo {}; // f.bar is null  
var b = Bar {foo: f}; // f.bar is b
```

Classes Revisited

- Triggers are called when their triggering events happen

```
class Foo { attribute ns: Number* }
trigger on (new Foo) { /* ... */ }
trigger on (Foo.ns[old]=new) { }
trigger on insert n into Foo.ns { }
trigger on delete n from Foo.ns { }
```

Classes Revisited

- Multiple inheritance
- Implementing Java interfaces
- Abstract class
 - Classes with undefined member functions or operations can still be instantiated
 - To make a class abstract, use ...

```
class Foo { ... }
```

```
var foo = Foo {}; // Error
```

Reflection

- As in Java, the `.class` operator is the key to the reflective world

```
var x = Foo { str: "Hello" };  
var c = x.class;  
var attrs = c.Attributes;  
var ops = c.Operations;  
var v = x[attrs[Name=='str']];  
var w = ops[Name=='opr'](x,  
    "Hi");
```

The Bind Operator

- The bind operator provide support for incremental evaluation and lazy evaluation
- All attributes of JavaFX classes are observable

```
var x1=Foo{a:1, b:2, c:3};  
var x2=Foo{  
    a:x1.a  
    b:bind x1.b,  
    c:bind lazy x1.c  
};
```

JavaFX Script: Widget Set

- LayoutManagers
 - GridPanel, GridBagPanel, FlowPanel, BorderPanel, Box, StackPanel, CardPanel, GroupPanel
- Borders
 - EmptyBorder, LineBorder, BevelBorder, SoftBevelBorder, MatteBorder, TitledBorder
- Menus
 - MenuBar, Menu, MenuItem, RadioButtonMenuItem

JavaFX Script: Widget Set

- Widgets
 - Label, SimpleLabel, Button, TabbedPane, ListBox, SplitPane, RadioButton, ToggleButton, ButtonGroup, ComboBox, Tree, Table, Spinner
- Text components
 - TextField, PasswordField, TextArea, EditorPane, TextPane

JavaFX Script: 2D Primitives

- Canvas
- Shapes
 - Rect, Circle, Ellipse, Line, Polyline, Polygon, Arc, CubicCurve, QuadCurve, Star, Text, Path (MoveTo, LineTo, Hline, Vline, CurveTo, QuadTo, ClosePath)
- Painting
 - Stroke, Fill, Gradient, Pattern
- Transformations
 - translate, rotate, scale, skew



JavaFX Script: 2D Primitives

- Group
- Swing components
 - View
- Images
 - ImageView
- Transparency
 - opacity
- Filters
 - Shadow, Blur, Noise, ShapeBurst



JavaFX Script: 2D Primitives

- MouseEvents
 - onMouseEntered, etc.
 - Area operations
 - Add, Subtract, Intersect, XOR
 - Clipping
 - User defined graphics objects
 - CompositeNode
 - Animation
 - The dur (duration) operator
 - Shape Morphing
-
-

Animation: The dur operator

- The documentation for the dur operator is sparse and the syntax is still being worked out

```
var x = [0, 0.1 .. 1.0]
      dur 1000 linear
      while shouldStop
      continue if shouldRepeat;
```

Deployment

- As JavaFX Script applications
 - Install JavaFX Script on target machine and run “javafx.sh MyApp.fx” from the command line
 - As Java applications
 - Bundle JavaFX Script runtime, MyApp.fx, and a bootstrap Java class in a jar
 - Install the JRE on target machines and run “java -jar MyApp.jar”
 - As Java Web Start downloads
 - This is how Chris Oliver delivers his apps
-
-

Deployment

- As Java applets
 - It's not easy, but can be done
 - The promised improved JRE will make applets competitive again
 - The bootstrap Java class for JavaFX
 - JSR 223 engine for JavaFX Script
 - Bundled within the tarball
 - putting a Java object into a Binding require a “name:Type” key
 - To run a JavaFX Script file simply eval the “import MyApp.fx” string
-
-

Future

- Sun is working on a JavaFX Scripts compiler
 - Sun is working on tools to support JavaFX development. Plug-ins are available for NetBeans 5.5 and 6.0, and Eclipse 3.2
 - Sun is working on making JavaFX Script runnable on Java based mobile devices and set-top boxes
 - Sun is working on a faster-downloading, faster-starting-up JRE
 - A JavaFX Script painter is available from ReportMill, <http://www.reportmill.com/jfx>
-
-