



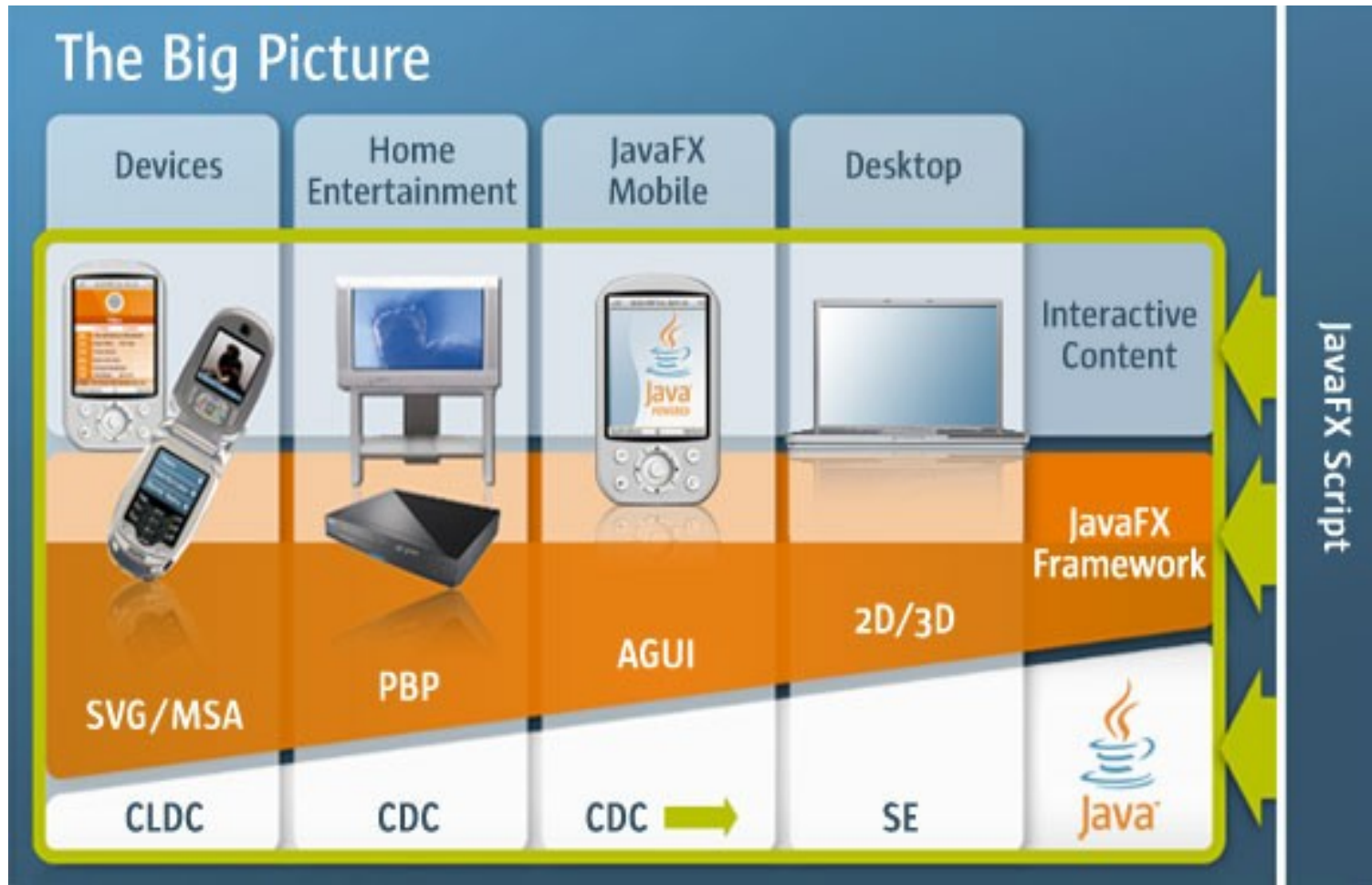
# JavaFX

**Simon Ritter**

Technology Evangelist

Sun Microsystems

# JavaFX



# JavaFX: Design Questions

- Why does it take a long time to write GUI programs?
- How can we avoid the “Ugly Java GUI” stereotype?
- Why do Flash programs look different than Java programs?
- Why does it seem easier to write web-apps than Swing programs?
- And how can I avoid having an enormous, writhing mass of listener patterns?

# JavaFX Basics

- Programming Language for the Java Platform
- Object-oriented
- Declarative Syntax
- Statically-typed with type-inference
- Automatic data binding
- Extensive Widget library encompassing Swing and Java 2D™ API
- Development tools including NetBeans™ and Eclipse IDE plugins

# The “Ugly Java GUI” Stereotype

- AWT/Swing Container/Component Hierarchy
  - > A tree of rectangular (mostly grey) boxes
  - > If all you do is compose Swing components together, the result is typically “the Ugly Java GUI”
  - > Same problem exists with other toolkits, e.g., GTK, VB
- UI Designers and Swing programmers are using different building blocks
  - > UI Designers compose designs in tools like Photoshop and Illustrator
  - > The building blocks they use have direct analogs in Java 2D, but not always directly in Swing

# A Basic Java GUI: Not Very Pretty



# Java 2D API

- To match the designs of UI designers requires using Java 2D API
- Java 2D API doesn't have compositional behavior
  - > Makes it too complex for many programmers to use efficiently
- In addition to Swing Components, JavaFX includes SVG-like interfaces to Java 2D API as first-class elements which can be composed together into higher-level components
- FX allows declarative expression of this composition

# JavaFX Script

# Variables

- JavaFX has four basic types:
  - > String            `java.lang.String`
  - > Boolean           `java.lang.Boolean`
  - > Number           `java.lang.Number`
  - > Integer           `byte, short, int, long, BigInteger`

# Variable Declaration

- `var name : type [ ?, +, * ] = initializer;`
  - > ? = optional
  - > + = one or more
  - > \* = zero or more
- `var digits : Number* = [1, 2, 3];`
- `var digits = [1, 2, 3];`
- `var name = 'foo'; // or = "foo";`
- `{ }` for variable name in strings
- `<< >>` to use reserved word for variable name
- Newlines can be placed directly in strings

# Functions

- Pure functional subset of Java language
- Only contains var declarations and return statement

```
function percent(a, b) {  
    var i = a * 100;  
    return i / b;  
}
```

# Operations (Procedures)

```
operation substring(s:String, n:Number) :  
  String {  
    try {  
      return s.substring(n);  
    catch (e : StringOutOfBoundsException) {  
      return "Index out of bounds";  
    }  
  }  
}
```

# Array Definitions

- Enclosed in [ ]
- Separated by commas
- Do not nest
- Use .. to indicate arithmetic range
  - > `var oneToTen = [1 .. 10];`
- Can contain expressions
  - > `var greaterThanFive = oneToTen[. > 5];`
- indexof function
  - > `list[indexof . > 0];`  
// all but first element

# Inserting Into Arrays

- insert expr [ as first | as last ] into expr2;
- insert expr before expr2;
- insert expr after expr2;

```
var x = [1,2];
```

```
insert 12 into x; // [1,2,12]
```

```
insert 10 as first into x; // [10,1,2,12]
```

```
insert 11 after x[. == 2]; //  
[10,1,2,11,12]
```

# Deleting From Arrays

- delete var;
- delete expr.attribute;
- delete variable[predicate];
- delete expr.variable[predicate];

```
var x = [1, 2, 3, 4, 5];  
delete x[. == 2]; // [1, 3, 4, 5]  
delete x[. > 3]; // [1, 3]  
delete x; // []
```

# Querying Arrays (List Comprehension)

```
var titleTracks =  
    select indexof track + 1 from  
        album in albums,  
        track in album.tracks  
    where track == album.title;  
  
var squares = select n*n from n in [1..10];
```

# Formatting

- expr format as << directive >>
- directive can be:
  - > java.text.DecimalFormat
  - > java.text.SimpleData
  - > java.util.Formatter (always starts with %)

**100.896 format as <<%f>> // 100.896000**

**31.intValue() format as <<%02X>> // 1F**

# Expressions

- if, while, try – Same syntax as Java

```
for (i in [0..10]) ...
```

```
for (i in [0..10] where i%2 == 0) ...
```

```
for (i in [0..10], j in [0..10]) ...
```

# Avoiding the Event Dispatch Thread

```
do {  
    // block of code executes in  
    // separate thread  
}
```

```
do later {  
    // block of code using  
    // java.awt.EventQueue.invokeLater  
}
```

# Classes

```
class Person {  
    attribute name: String;  
    attribute parent: Person inverse  
        Person.children;  
    attribute children: Person* inverse  
        Person.parent;  
    function getNumberOfChildren(): Number;  
}  
function Person.getNumberOfChildren() {  
    return sizeof this.children;  
}
```

# Attributes

```
class Point {  
    attribute x: Number;  
    attribute y: Number;  
    attribute z: Number;  
}
```

```
attribute Point.x = 10;  
attribute Point.y = bind x + 10;  
attribute Point.z = bind lazy y + 10;
```

```
var p = new Point(); // x=10, y=20, z=0;  
p.x = 5 // x=5, y=15, z=0  
System.out.println("z="+p.z); // "z=25" 21
```

# Triggers

```
class X {
    attribute nums: Number*;
}

trigger on new X { // Creation trigger
    insert [1,2] into this.nums;
}

trigger on insert num into X.nums {
    System.out.println("{num} added to X");
}

trigger on delete num from X.nums {
    System.out.println("{num} deleted from X");
}

trigger on X.nums[oVal] = nVal {
    System.out.println("{nVal} replaced {oVal} in X");
}
```

# 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");
```

# JavaFX Widget Set

- Maps to Swing and Java 2D components
- Can be used easily in scripts
  - > Define attributes quickly and easily
  - > Code functionality around this

# JavaFX and Swing Components

- Borders and Layout Managers
- Menus
- Labels
- Group Panel, Simple Label, and TextField
- Buttons
- ListBoxes
- SplitPanels
- RadioButton, RadioButtonMenuItem, ToggleButton, and ButtonGroup
- ComboBoxes
- Trees
- Tables
- TextComponents
- Spinners and Sliders

# Example Widget Usage

```
Button {  
    row: row  
    column: column2  
    opaque: false  
    mnemonic: W  
    text: "Width"  
    action: operation() {  
        width = [0..200] dur 1000;  
    }  
}
```

# 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;
```

# Getting Clever: Database Access

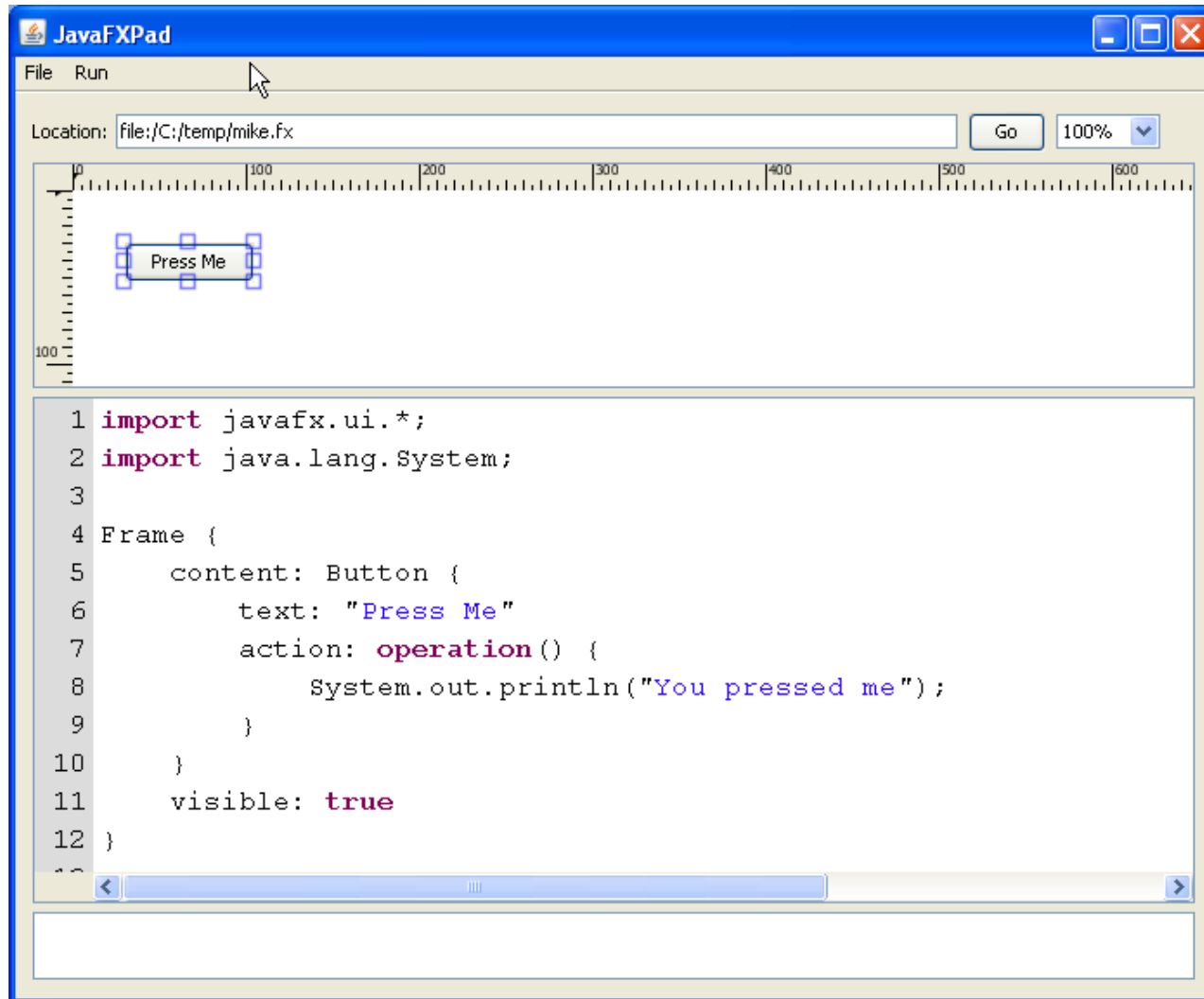
```
// Load driver class using context class loader
var thread = Thread.currentThread();
var classLoader = thread.getContextClassLoader();
var driverClass =
    classLoader.loadClass(driverClassName);

// Instantiate and register JDBC driver
driver = (Driver) driverClass.instantiate();
DriverManager.registerDriver(driver);
```

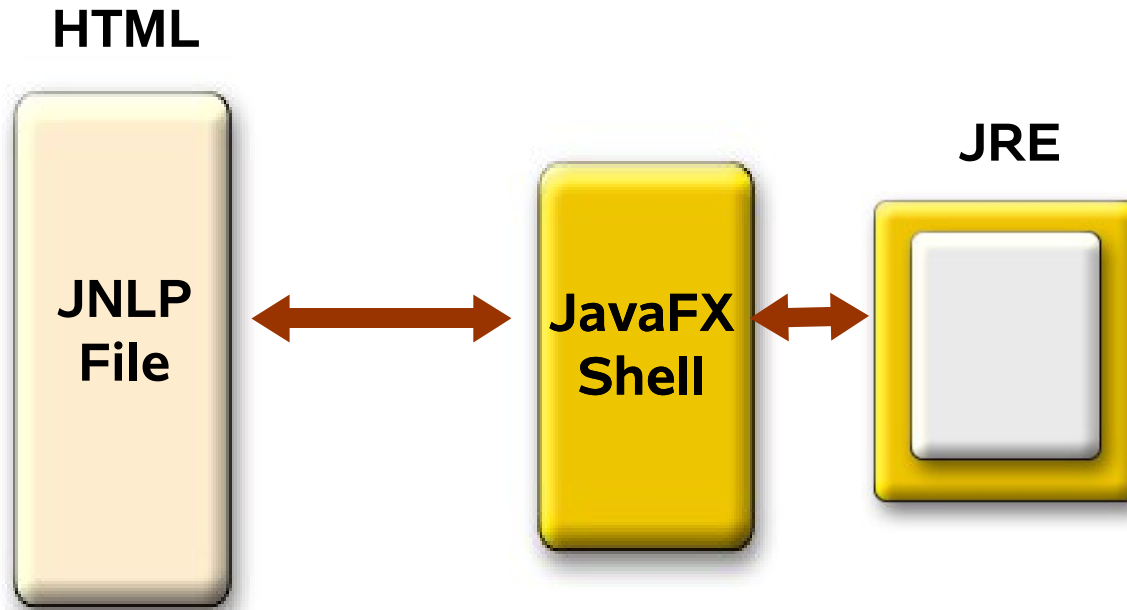
# Tooling: NetBeans Plugin

- Understands the Edit-Compile-Debug lifecycle for JavaFX programs on NetBeans
- Run JavaFX programs in NetBeans easily with the JavaFX shell

# Tooling: JavaFXPad



# JavaFX Script Deployment using Java Web Start



# JavaFX Script Deployment

- JavaFX Script Runtime
  - > 1.5 MB jars (700 kb with pack200)
- JavaFX Script Deployment the same as Java
  - > JavaFX Script files are archived in Jar files and loaded via the Java class loader
  - > Standalone Java Application
  - > Java Web Start
  - > Applet

# Invoking JavaFX From Java Apps

- Use FXShell
  - > Use simple wrapper to call the main() method of the FXShell class
  - > FXShell is part of the JavaFX jars
  - > Not officially supported
- Use JSR-223
  - > Invoke JavaFX scripting engine through standard APIs
  - > Requires JDK 6 or JSR-223 support
  - > More flexible

# Future

- JavaFX Script compiler
- Tools to support JavaFX development. Plug-ins are available for NetBeans 5.5 and 6.0, and Eclipse 3.2
- Making JavaFX Script runnable on Java based mobile devices and set-top boxes
- Integrate with consumer JRE
- A JavaFX Script painter is available from ReportMill, <http://www.reportmill.com/jfx>

# Consumer JRE

- Quickstarter
- Java Kernel
- Deployment toolkit
- Installer improvements
- Graphics performance
  - > Windows
- Nimbus
  - > New cross-platform look-and-feel

# Summary

- JavaFX is a family of products and technologies aimed at content creators
- JavaFX script simplifies GUI programming
  - > Let the graphic artists do the hard work
- More coming, watch this space
  - > Better tools (more drag'n'drop)
  - > Ease of deployment
  - > Consumer, modular JRE

# Further Information

<http://www.sun.com/javafx>

<http://openjfx.org>

<http://blogs.sun.com/chrisoliver>

<http://evc-cit.info/jfx/makeapi/api/index.html>

# Sun TechDays Coming To Germany

- Two days of detailed Java information
- Hands on labs
- FREE!
- Frankfurt am Main, Congress Center
- December 3-5, 2007
- Register at

[de.sun.com/techdays](http://de.sun.com/techdays)

# Demos

# JavaFX

**Simon Ritter**

[simon.ritter@sun.com](mailto:simon.ritter@sun.com)

<http://blogs.sun.com/simonri>

Sun Microsystems