Refcard #056

JavaFX

Making it Easier to Build Better RIAs

by Stephen Chin

Gets you started with JavaFX, which makes it easier to build better RIAs with graphics, animation, and media.

SECTION 1

About JavaFX

JavaFX is an exciting new platform for building Rich Internet Applications with graphics, animation, and media. It is built on Java technology, so it is interoperable with existing Java libraries, and is designed to be portable across different embedded devices including mobile phones and set-top boxes. This Refcard will help you get started programming with JavaFX Script and also serve as a convenient reference once you have mastered the language.

To get started, you will have to download the latest JavaFX SDK from the JavaFX website here: http://javafx.com/.

The instructions in the following tutorial assume that you are using an IDE, such as NetBeans. However, it is possible to do everything from the command line as well.

SECTION 2

JFXPoetry, a Simple Example

To illustrate how easy it is to build an application that melds graphics, text, animation, and media, we will start with a simple tutorial. The goal will be to write an application that:

- Loads and displays an image from the internet
- Displays and animates a verse of poetry
- Declaratively mixes in graphic effects
- Plays media asynchronously

For the JFXPoetry theme, we will use “Pippa’s Song,” a wellknown excerpt from Robert Browning’s Pippa Passes.

Loading an Image on the Stage

Stage and Scene are the building blocks of almost every JavaFX program. A Stage can either be represented as a Frame for desktop applications, a rectangle for applets, or the entire screen for mobile devices. The visual content of a Stage is called a Scene, which contains a sequence of content Nodes that will be displayed in stacked order. The following program creates a basic Stage and Scene which is used to display an image:

```
1
2  var scene:Scene;
3  Stage {
4    title: “Pippa’s Song by Robert Browning”
5    scene: scene = Scene {
6      content: [
7        ImageView {
8          image: bind Image {
```

```
8       image: bind image {
9         height: scene.height
10        preserveRatio: true
11        url: “http://farm1.static.flickr.com/39/
12          121693644_75491b23b0.jpg”
13      }
14    }
15  ]
16 }
17 }
```

Notice that that JavaFX syntax makes it simple to express nested UI structures. The curly braces “{}” are used for object instantiation, and allow inline initialization of variables where the value follows the colon “:”. This is used to instantiate an ImageView with an Image inside that loads its content from the given URL. To ensure the image resizes with the window, we set preserveRatio to true and bind the Image. Binding is a very powerful facility in JavaFX that makes it easy to update values without heavyweight event handlers. Compiling and running this application will display a picture of a misty morning in Burns Lake, BC, Canada taken by Duane Conlon as shown in Figure 1.1 2

Figure 1: A JavaFX Stage containing an image loaded from the network

Displaying Text with Effects

Displaying text in JavaFX is as simple as instantiating a Text Node and setting the content to a String. There are many variables available on Text, but for this example we will set the font, fill color, and also add a Drop Shadow effect to make the text stand out on the background.

---

1 Creative Commons Attribution 2.0 License: http://creativecommons.org/licenses/by/2.0/

2 Duane Conlon’s Photostream: http://www.flickr.com/photos/duaneconlon/

```
1
2 var text:Text;
3 Stage {
4         ...
5         ImageView {
6       ...
7         },
```

```
        text = Text {
            effect: DropShadow {}
            font: bind Font.font("Serif", FontWeight.BOLD,
                                 scene.height / 12.5)
          fill: Color.GOLDENROD
          x: 10
          y: bind scene.height / 6
          content: "The year's at the spring,\n"
                   "And day's at the morn;\n"
                   "Morning's at seven;\n"
                   "The hill-side's dew-pearled;\n"
                   "The lark's on the wing;\n"
                   "The snail's on the thorn;\n"
                   "God's in His heaven--\n"
                   "All's right with the world!"
      }
```

Notice that rather than specifying the whole poem text on one line we have split it across several lines, which will automatically get concatenated. Also, we have used the bind operator to set both the font size and y offset, which will update their values automatically when the scene height changes. Figure 2 shows the updated example with text overlaid on the Image.

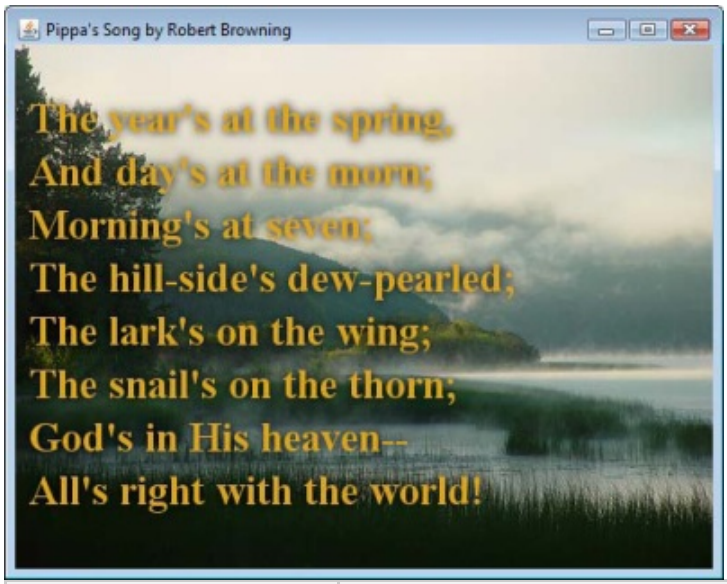

Figure 2: Updated example with a Text overlay

JavaFX offers a large set of graphics effects that you can easily apply to Nodes to create rich visual effects. Table 1 lists all the available effects you can choose from.

Table 1. Graphics effects available in JavaFX

Effect	Description
Blend	Blends two inputs together using a pre-defined BlendMode
Bloom	Makes brighter portions of the Node appear to glow
BoxBlur	Fast blur with a configurable quality threshold
ColorAdjust	Per-pixel adjustments of hue, saturation, brightness, and contrast
DisplacementMap	Shifts each pixel by the amount specified in a DisplacementMap
DropShadow	Displays an offset shadow underneath the node
Flood	Fills a rectangular region with the given Color
GaussianBlur	Blurs the Node with a configurable radius
Glow	Makes the Node appear to glow with a given intensity level
Identity	Passes an image through to a chained effect
InnerShadow	Draws a shadow on the inner edges of the Node
InvertMask	Returns a mask that is the inverse of the input

Lighting	Simulates a light source to give Nodes a 3D effect
MotionBlur	Blurs the image at a given angle to create a motion effect
PerspectiveTransform	Maps a Node to an arbitrary quadrilateral for a perspective effect
Reflection	Displays an inverted view of the Node to create a reflected effect
SepiaTone	Creates a sepia tone effect to mimic aged photographs
Shadow	Similar to a DropShadow, but without the overlaid image

Animated Transitions

Animations in JavaFX can be accomplished either by setting up a Timeline from scratch, or using one of the pre-fabricated Transitions. To animate the Text rising onto the screen, we will use a TranslateTransition, which adjusts the position of a Node in a straight line for the specified duration:

```
1
2 var animation = TranslateTransition {
3   duration: 24s
4   node: text
5   fromY: scene.height
6   toY: 0
7   interpolator: Interpolator.EASEOUT
8 }
9 animation.play();
```

By setting an interpolator of EASEOUT, the text will start at full speed and gradually deaccelerate as it approaches its destination. Animations and Transitions can also be configured to repeat, run at a specific rate, or reverse. To run the transition, all you need to do is call the play() function, which will animate the text as shown in Figure 3.

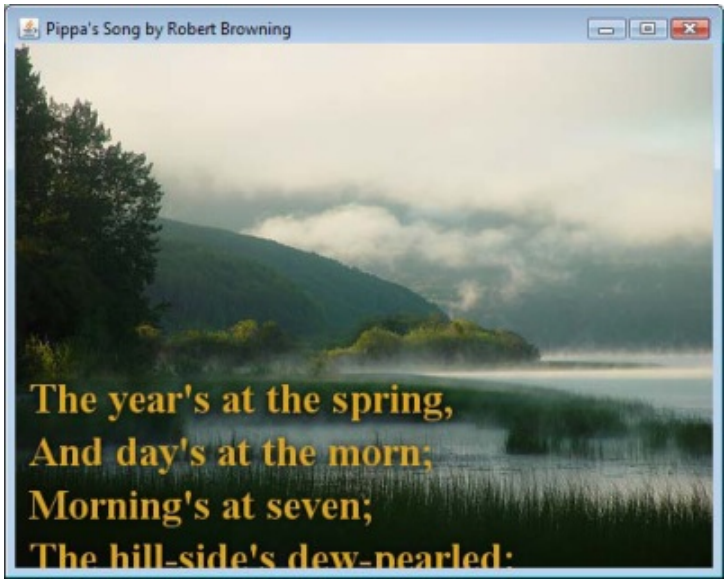

Figure 3: Animated Text Scrolling Into View

Table 2 lists all of the available transitions that are part of the JavaFX API. To get a feel for how the different transitions work, try adding a FadeTransition that will gradually fade the background in over a 5 second duration.

Table 2. Transitions Supported by JavaFX

Transition	Description
FadeTransition	Changes the opacity of a node over time
ParallelTransition	Plays a sequence of transitions in parallel
PathTransition	Animates nodes along a Shape or Path
PauseTransition	Executes an action after the specified delay

PauseTransition	Executes an action after the specified delay
RotateTransition	Changes the rotation of a node over time
ScaleTransition	Changes the size of a node over time
SequentialTransition	Plays a sequence of transitions in series
TranslateTransition	Changes the position of a node over time

Interacting with Controls

The JavaFX 1.2 release features a new library of skinnable controls written in pure JavaFX. Table 3 lists some of the new controls and what they can be used for.

Table 3. Controls Available in JavaFX 1.2

Control	Description
Button	Button that can contain graphics and text
CheckBox	Selectable box that can be checked, unchecked, or undefined
Hyperlink	HTML-like clickable text link
Label	Text that can be associated with anther control
ListView	Scrollable list that can contain text or Nodes
ProgressBar	Progress bar that can show percentage complete or be indeterminate
RadioButton	Selectable button that can belong to a group
ScrollBar	Scroll control typically used for paging
Slider	Draggable selector of a number or percent
TextBox	Text input control

The simplest control to use is a Button, which can easily be scripted to play the animation sequence again from the beginning.

```
1
2  var button:Button;
3  Stage {
4  ...
5  text = Text {
6  ...
7     },
8     button = Button {
9        translateX: bind (scene.width - button.width) / 2
10       translateY: bind (scene.height - button.height) / 2
11       text: “Play Again”
12     visible: bind not animation.running
13     action: function() {
14       animation.playFromStart();
15     }
16    }
17  ]
```

Ths bind operator is used to both hide the button while the animation is playing and also center the button in the window. Initially the button is invisible, but we added a new SequentialTransition that plays a FadeTransition to show the button after the translation is complete. Clicking the button shown in Figure 4 will hide it and play the animation from the beginning.

animation from the beginning.

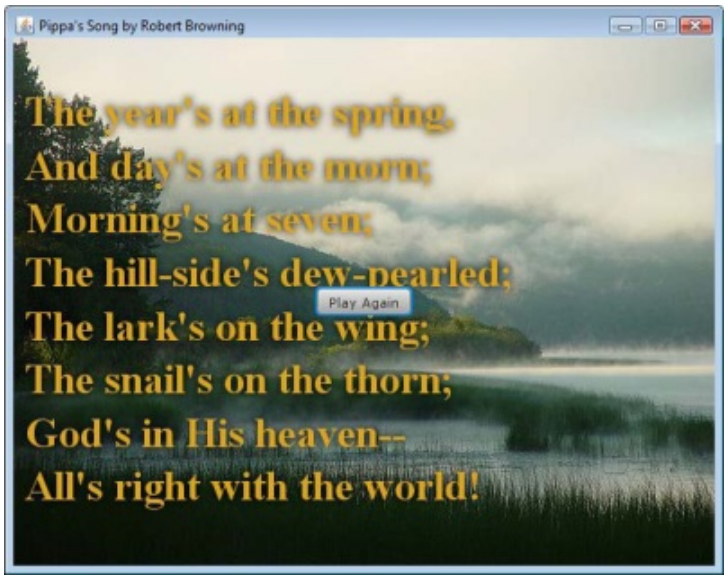

Figure 4: Button Control to Play the Animation Again

Panning with Layouts

JavaFX 1.2 comes with several new layouts that make it easy to design complex UIs. One of these is the ClipView, which we will use to support dragging of the poem text. ClipView takes a single Node as the input and allows the content to be panned using the mouse:

```
1
2  content: [
3 ...
4     ClipView {
5         width: bind scene.width
6         height: bind scene.height
7         override var maxClipX = 0
8         node: text = Text {
9 ...
10    }
11   }
```

To ensue the ClipView takes the full window, its width and height are bound to the scene. Also, we have overridden the maxClipX variable with a value of 0 to restrict panning to the vertical direction. The text can now be dragged using the mouse as shown in Figure 5.

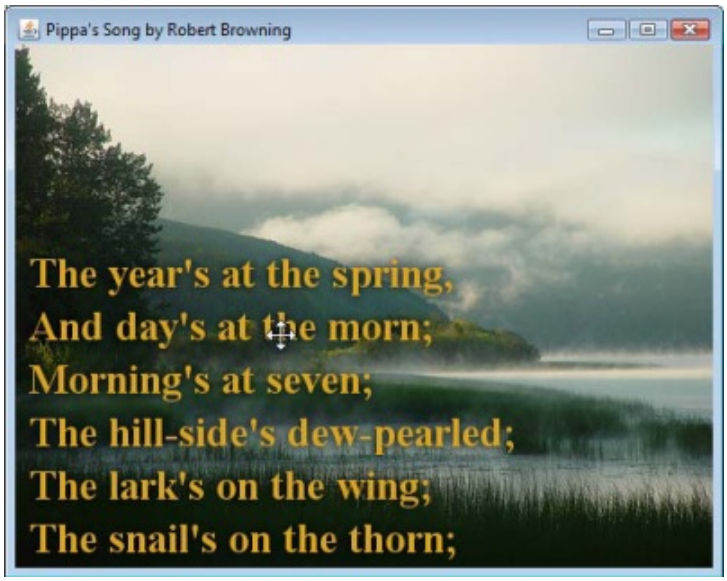

Figure 5: Panning the Text using a ClipView

Table 4 lists all of the available layouts that come JavaFX comes with. HBox and VBox have been around since the 1.0 release, but all the other layouts are new in JavaFX 1.2.

Table 4. Layouts Available in JavaFX 1.2

Layout	Description
HBox	Lays out its contents in a single, horizontal row
VBox	Lays out its contents in a single, vertical column
ClipView	Clips its content Node to the bounds, optionally allowing panning
Flow	Lays out its contents either vertically or horizontally with wrapping
Stack	Layers its contents on top of each other from back to front
Tile	Arranges its contents in a grid of evenly sized tiles

Finishing with Media

JavaFX has built-in media classes that make it very simple to play audio or video either from the local files or streaming off the network. To complete the example we will add in a public domain clip of Indigo Bunting birds chirping in the background. Adding in the audio is as simple as appending a MediaPlayer with autoPlay set to true that contains a Media object pointing to the URL.

```
1
2 MediaPlayer {
3   autoPlay: true
4   media: Media {
5     source: “http://video.fws.gov/sounds/35indigobunting.mp3”
6   }
7 }
```

In this example we are using an mp3 file, which is supported across platforms by JavaFX. Table 5 lists some of the common media formats supported by JavaFX, including all the crossplatform formats.

Table 5. Common Media Formats Supported by JavaFX

Type	Platform	Format	File Extension
Audio	Cross-platform	MPEG-1 Audio Layer 3	mp3
Audio	Cross-platform	Waveform Audio Format	wav
Audio	Macintosh	Advanced Audio Coding	m4a, aac
Audio	Macintosh	Audio Interchange File Format	aif, aiff
Video	Platform	Format	File Extension
Video	Cross-platform	Flash Video	flv, f4v
Video	Cross-platform	JavaFX Multimedia	fxm
Video	Windows	Windows Media Video	wmv, avi
Video	Macintosh	QuickTime	mov
Video	Macintosh	MPEG-4	mp4

To try the completed example complete with animation and audio, you can click on the following url:

http://jfxtras.org/samples/jfxpoetry/JFXPoetry.jnlp

The full source code for this application is available on the JFXtras Samples

The full source code for this application is available on the JFXtras Samples website: http://jfxtras.org/portal/samples

Running on Mobile

To run the sample in the Mobile Emulator all you have to do is pass in the MOBILE profile to the javafxpackager program or switch the run mode in your IDE project properties. JavaFX Mobile applications are restricted to the Common Profile, which does not include all the features of desktop applications. The full list of restrictions is shown in Table 5.

Table 5. Functionality Not Available in the Common Profile

Class(es)	Affected Variables and Methods
javafx.ext.swing.*	All
javafx.reflect.*	All
javafx.scene.Node	effect, style
javafx.scene.Scene	stylesheets
javafx.scene.effect.*	All
javafx.scene.effect.light.*	All
javafx.scene.shape.ShapeIntersect	All
javafx.scene.shape.ShapeSubtract	All
javafx.scene.text.Font	autoKern, embolden, letterSpacing, ligatures, oblique, position
javafx.stage.AppletStageExtension	All
javafx.util.FXEvaluator	All
javafx.util.StringLocalizer	All

Over 80% of the JavaFX API is represented in the Common Profile, so it is not hard to build applications that are portable. In this example we used a DropShadow on the text that, once removed, will let us run the example in the Mobile Emulator as shown in Figure 6.

Figure 6: JFXPoetry application running in the Mobile Emulator

Running as a Desktop Widget

You can deploy your application as a desktop widget using the WidgetFX open-source framework. Any JavaFX application can be converted to a widget by including the WidgetFX-API.jar and making some small updates to the code.

The Following code fragment highlights the code changes required:

```
1
2 var widget:Widget = Widget {
```

```
var widget:Widget = Widget {
  resizable: false
  width: 500
  height: 375
  content: [
...
      height: widget.height
...
    font: bind Font.font("Serif", FontWeight.BOLD,
            widget.height / 12.5)
...
    y: bind widget.height / 6
...
  ]
}
```

```

...
    fromY: widget.height
...
widget;
```

The updates to the code include the following three changes:

- Wrap your application in a Widget class. The Widget class extends javafx.scene.layout.Panel, which makes it easy to extend.
- Set the initial widget width/height and modify references from scene to widget.
- Return the widget at the end of the script.

To run the widget, simply change your project properties to run the application using Web Start Excecution. This will automatically create a JNLP file compatible with WidgetFX and launch the Widget Runner, which allows you to test your widget as shown in the Figure 7.

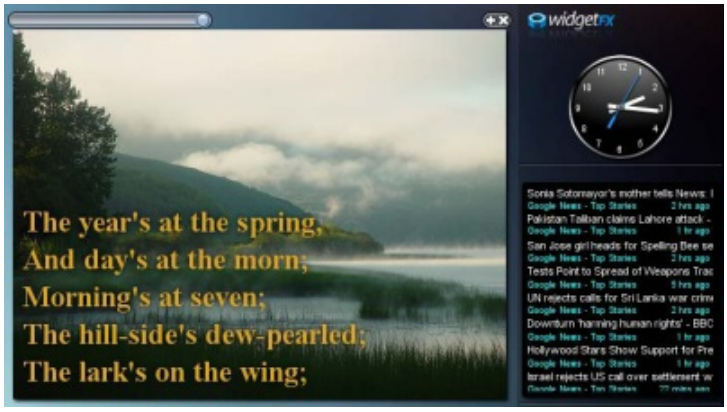

Figure 7: JFXPoetry running as a desktop widget

For more information about WidgetFX, including SDK download, documentation, and additional tutorials, check out the project website:http://widgetfx.org/

SECTION 3

JavaFX Reference

Language Reference

JavaFX supports all the Java datatypes plus a new Duration type that simplifies writing animationed UIs.

writing animations etc.

Data Types:

DataType	Java Equivalent	Range	Examples
Boolean	boolean	true or false	true,false
Integer	int	-2147483648 to 2147483647	2009, 03731, 0x07d9
Number	float	1.40×10 45 and 3.40×1038	3.14, 3e8, 1.380E-23
String	String	N/A	“java’s”, ‘in”side”er’
Duration	<None>	-263 to 263-1 milliseconds	1h, 5m, 30s, 500ms
Character	char	0 to 65535	0,20,32
Byte	byte	-128 to 127	-5, 0,5
Short	short	-32768 to 32767	-300, 0, 521
Long	long	-263 to 263-1	2009, 03731,0x07d9
Float	float	1.40x10 45 and 3.40x1038	3.14, 3e8, 1.380E-23
Double	double	4.94x10 324 and 1.80x10308	3.14, 3e8, 1.380E-123

JavaFX Characters cannot accept literals like ‘a’ or ‘0’, because they are treated as Strings. The primary way of getting Characters will be by calling a Java API that returns a char primitive, although you can create a new character by assigning a numeric constant

Operators:

The following table lists all the mathematical, conditional, and boolean operators along with their precedence (1 being the highest).

Operator	Meaning	Precedence	Examples
++	Pre/post increment	1	++i, i++
--	Pre/post decrement	1	--i, i--
not	Boolean negation	2	not (cond)
*	Multiply	3	2 * 5, 1h * 4
/	Divide	3	9 / 3, 1m / 3
mod	Modulo	3	20 mod 3
+	Add	4	0 + 2, 1m + 20s
-	Subtract (or negate)	4 (2)	-2, 32 - 3, 1h - 5m
==	Equal	5	value1 == value2, 4 == 4
!=	Not equal	5	value1 != value2, 5 != 4
<	Less than	5	value1 < value2, 4 < 5
<=	Less than or equal	5	value1 <= value2, 5 <= 5
<	Greater than	5	value1 > value2, 6 > 5
>=	Greater than or equal	5	value1 >= value2, 6 >= 6
instanceof	Is instance of class	6	node instanceof Text
as	Typecast to class	6	node as Text

and	Boolean and	7	cond1 and cond2
or	Boolean or	8	cond1 or cond2
+=	Add and assign	9	value += 5
-=	Subtract and assign	9	value -= 3
*=	Multiply and assign	9	value *= 2
/=	Divide and assign	9	value /=4
=	Assign	9	value = 7

- Multiplication and division of two durations is allowed, but not meaningful
- Underflows/Overflows will fail silently, producing inaccurate results
- Divide by zero will throw a runtime exception

Sequences:

JavaFX sequences provide a powerful resizable and bindable list capability under a simple array-like syntax. All of the sequence operators (sizeof, reverse, indexof) have a relative precedence of 2.

Operation	Syntax	Examples
Construct	1 2 [x,y,z] 3 [y..z] 4 [y..<z] 5 [y..z step w]	1 2 var nums = [1, 2, 3, 4]; var letters = [“a”, 3 [1..5] = [1, 2, 3, 4, 5] 4 [1..>5] = [1, 2, 3, 4] 5 [1..9 step 2] = [1, 3, 5, 7, 9]
Size	sizeof seq	sizeof nums; // = 4
Index	indexof variable	1 2 for(x in seq) { 3 indexof x; 4 }
Element	seq[i]	letters[2]; // = “c”

Slice	1
	2 eq[x..y]
	3 seq[x..<y]
		1
		2 nums[1..2]; // = [2, 3]
		3 letters[0..<2]; // = [“a”, “b”]

Predicate	seq[x|boolean]	nums[n|n mod 2 == 0]; // = [2, 4]
Reverse	reverse seq	reverse letters; // = [“c”, “b”, “a”]

Insert	1
	2 insert x into seq
	3 insert x before seq[i]
	4 insert x after seq[i]
		1
		2 insert 5 into nums; // = [1, 2, 3, 4, 5]
		3 insert “gamma” before letters[2]; // = [“a”
		4 “gamma”, “c”]
		5 insert “2.3” after nums[1]; // = [1, 2, 2.3,

Delete	1
	2 delete seq[i]
	3 delete seq[x..y]
	4 delete x from seq
	5 delete seq
		1
		2 delete letters[1]; // = [“a”, “c”]
		3 delete nums[1..2]; // = [1, 4]
		4 delete “c” from letters; // = [“a”, “b”]
		5 delete letters; // = []

- The javafx.util.Sequences class provides additional functions, which allow you to manipulate sequences, such as min, max, search, shuffle, and short.
- Nested sequences are automatically flattened, so [[1,2], [3,4]] is equivalent to [1,2,3,4].
- Sequences require commas after all elements except close braces; however it is recommended to always use commas
- You can declare a sequence as a nativearray. This is an optimization so that arrays returned from a Java method don’t need to be converted to a sequence.

Access Modifiers:

The JavaFX access modifiers are based upon Java with the addition of extra variable-only modifiers.

Modifier	Name	Description
<Default>	Script only access	Only accessible within the same script file
package	Package access	Only accessible within the same package
protected	Protected access	Only accessible within the same package or by subclasses
public	Public access	Can be accessed anywhere
publicread	1 2 Read access 3 modifier	Var/def modifier to allow a variable to be read anywhere
public-init	Init access modifier	Var/def modifier to allow a variable to be initialized or read anywhere

- Unlike Java the default permission in JavaFX is script-only rather than package.
- The var/def access modifiers can be stacked with other modifiers, such as public-read protected

Expressions:

JavaFX supports many of the same expressions as Java, but adds in powerful inline functions and for loop extensions.

Expression	Syntax
if	`if (cond) expr1 else expr2` `if (cond) then expr1 else expr2`

for	`for (x in seq) expr`
	`for (x in seq where cond) expr`
	`for (x in seq, y in x) expr`

while	while (bool) expr
try/catch/ finally	try {expr1} catch(exception) {expr2} finally {expr3}
function	function(params):returnType{}

Just like in Java programs:

- continue can be used to skip a for or while loop iteration
- break can be used to exit a for or while loop
- return can be used to exite from a function event if inside a loop

Magic Variables:

JavaFX provides some built-in variables that can be accessed from any code running inside a script.

Name	Description
__DIR__	Directory the current classfile is contained in
__FILE__	Full path to the current classfile
__PROFILE__	The current profile, which can be 'desktop' or 'mobile'

API Reference

In the short span of a few pages you have already seen quite a bit of the JavaFX platform. Some other functionality that JavaFX offers includes:

Package	Description
javafx.animation	Animation and Interpolation
javafx.async	Asynchronous Tasks and Futures
javafx.data.feed	RSS/Atom Feed support
javafx.data.pull	XML and JSON Pull Parsers
javafx.ext.swing	Additional Swing-based Widgets
javafx.fxd	Production Suite (FXD)
javafx.io	Local Data Storage
javafx.reflect	JavaFX Reflection Classes
javafx.chart	Charting and Graphing
javafx.scene.media	Media (Audio and Video) Playback
javafx.scene.shape	Vector Shapes

An easy way to view and navigate the full JavaFX API is using the JFXplorer application. The following URL will launch it in as a web start application that you can use to start exploring the JavaFX API today:

http://jfxtras.org/samples/jfxplorer/JFXplorer.jnlp

Additional Resources

- JavaFX API documentation: http://java.sun.com/javafx/1.2/docs/api/
- JFXStudio, a great place to find sample JavaFX applications: http://jfxstudio.wordpress.com/

- JFXtras, utilities and add-ons for JavaFX: http://jfxtras.org/
- WidgetFX, deploy your JavaFX application as a desktop ,br> widget: http://widgetfx.org/
- Sang Shin and Jim Weaver's Free JavaFX Class: http://www.javapassion.com/javafx

Publications

- Featured
- Latest
- Popular

NaNundefined NaNundefined