# JavaFX

Dr. Stefan Schneider
Chief Technologist ISV-Engineering
stefan.schneider@sun.com, twitter:holodoctor
Sun Microsystems GmbH


Brought to you by
Sun Startup Essentials Program
http://blogs.sun.com/startups

# Sun Startup Essentials:
# Das Programm für junge Unternehmen
# Einschreiben & mitmachen
# http://www.sun.de/startups
# twitter: sun4startups

# Market Trends

## Consumer

Rich Media is Core

Familiarity Across Screens

Expressive

Contextual Content

## Content Author

Growth of Designer Community

Quicker Content Authoring and Deployment

Participative Authoring and Consumers

2008 Sun Microsystems, Incorporated. All Rights Reserved.

# JavaFX Vision

JavaFX is *the* Platform for Creating and Delivering
**Rich Internet Applications Across Multiple Screens**


JavaFX is **Powered by Java**

2008 Sun Microsystems, Incorporated. All Rights Reserved.

# Growth of Rich Internet Applications

| | |
|---|---|
| Video and Audio | Animation |
| Graphics | Transformations and Filters |
| Web Services | Rich Text |

Simple Scripting Language

Graphical Design Tools

2008 Sun Microsystems, Incorporated. All Rights Reserved.

# What Can You Build with JavaFX

- Cross-Browser Video playback
- Interactive and immersive business applications
- Mash-ups with REST based web services
- Applications that run across the browser, desktop, mobile and more!


2008 Sun Microsystems, Incorporated. All Rights Reserved.

# JavaFX Developer Tool Chain

## Media Assets Created By

Adobe Illustrator

Adobe Photoshop


On2 Flix Encoders
(JavaFX File Format, VP6 and MP3)

Adobe CS3
(Flash and Flash Video, JavaFX 1.5)

## Assets Transformed By

JavaFX Production Suite

3rd Party RAD Tool

## Integrated Into IDEs

NetBeans.org

JavaFX Plug-in for IDEs

JavaFX Compiler

eclipse

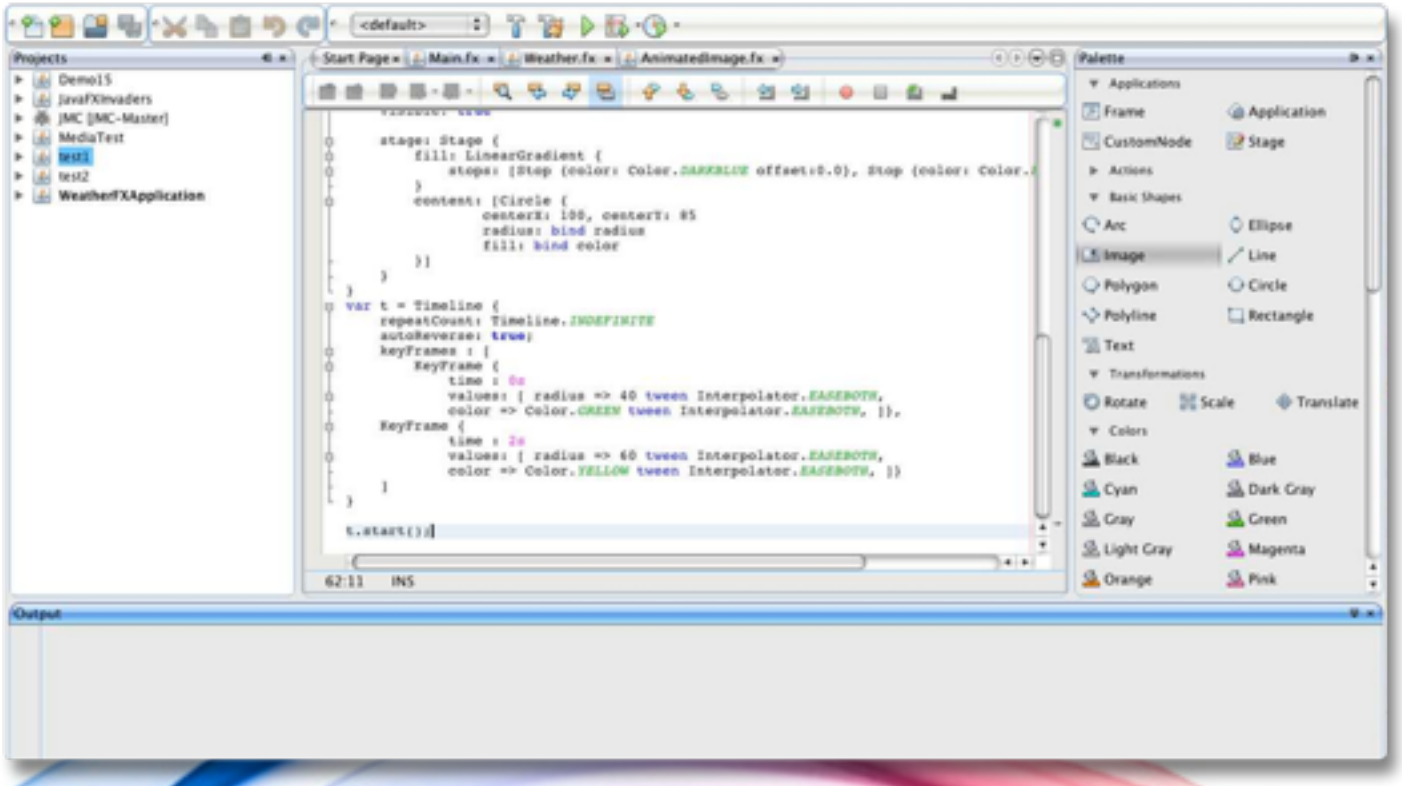
(Open Source)

## Emulated By (If Required)

JavaFX Mobile Emulator

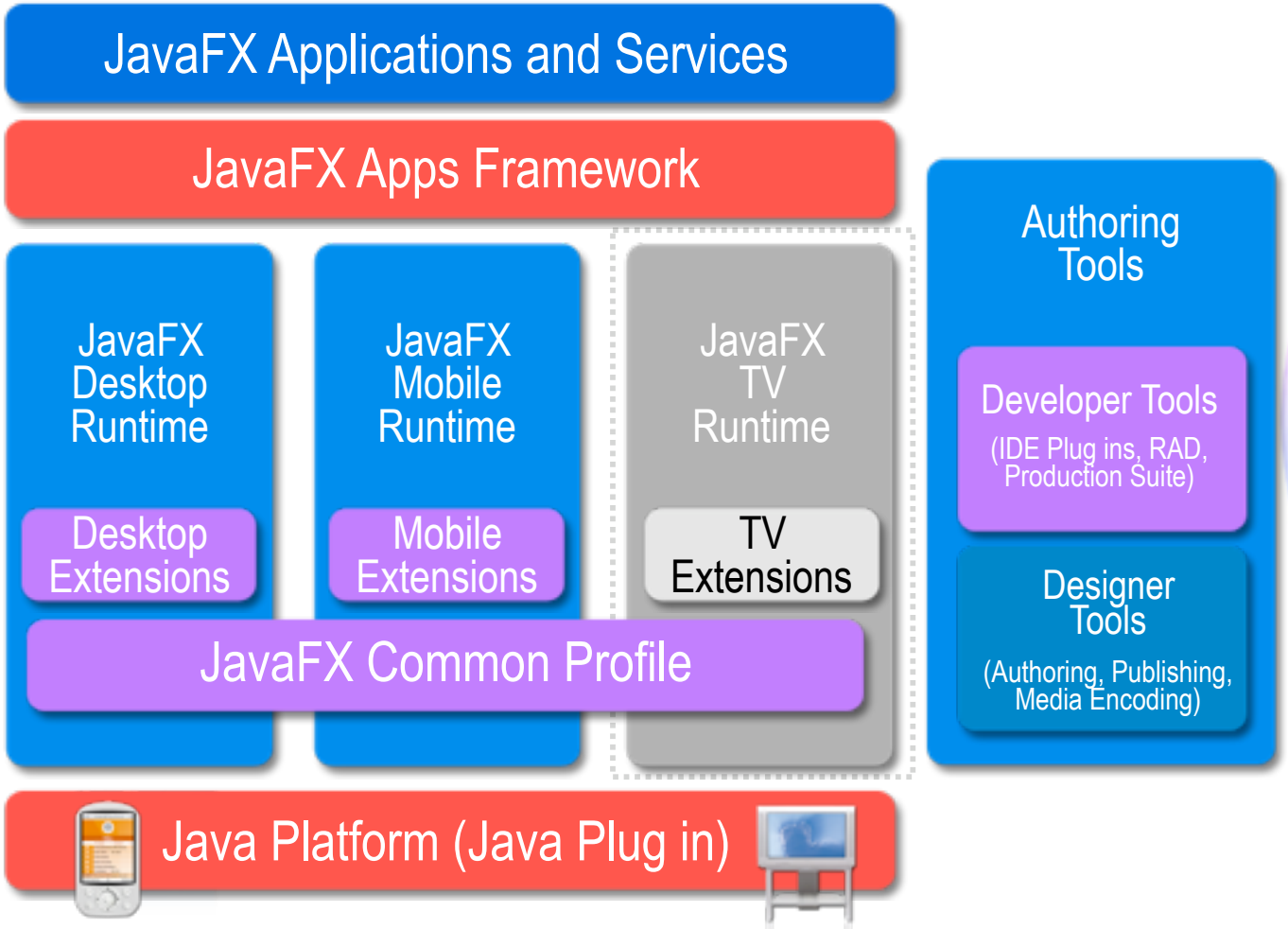
2008 Sun Microsystems, Incorporated. All Rights Reserved.

# JavaFX Plugin for NetBeans




2008 Sun Microsystems, Incorporated. All Rights Reserved.

# JavaFX + Java Marketecture




2008 Sun Microsystems, Incorporated. All Rights Reserved.

# JavaFX Common Profile Features

| Components | Features | |
|---|---|---|
| Compiler and Languages | >SE 5 and CLDC Target | |
| Graphics | >Geometric shapes, lines, curves, arc<br>>Transparency<br>>Gradient, color fill, texture<br>>Stroke styles | >Clip with arbitrary geometric shapes<br>>Image masks<br>>Fullscreen support<br>>transforms (rotate, scale skew) |
| Text | >True Type font rendering<br>>Transforms (rotate, scale, skew)<br>>Content embedded font | |

2008 Sun Microsystems, Incorporated. All Rights Reserved.

# JavaFX Common Profile Features

| Components | Features | |
|---|---|---|
| Animation | >Key frame animation with tweening<br>>Path-based animation | >Standard animations (rotate, zoom, slide) |
| Media | >Cross platform audio (mp3) and video (On2)<br>>Codec native media framework support (DirectShow and Core Video), play, pause, seek, volume, balance, speed controls | >Volume and audio balance control<br>>Http streaming with buffering fxm file format (FLV subset) |
| Other | >Web services (JSON/XML parser, RESTful APIs)<br>>Common text input control (CSS skinning)<br>>Input handling (keyboard, mouse, touch | |

2008 Sun Microsystems, Incorporated. All Rights Reserved.

# JavaFX Desktop Extensions

Filter Effects

Swing Elements

Runtime Features

2008 Sun Microsystems, Incorporated. All Rights Reserved.

# JavaFX Desktop Deployment Model

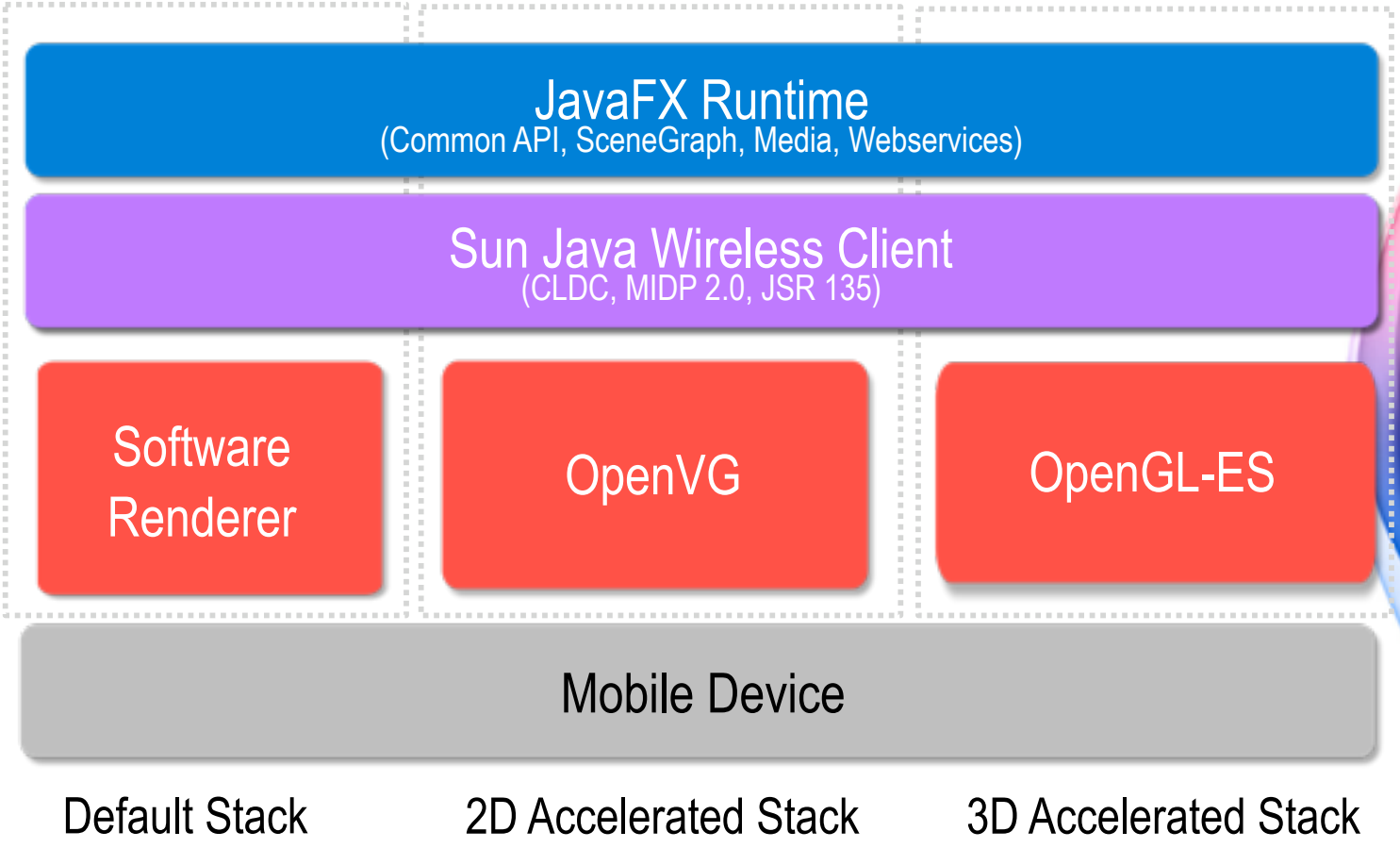
## Runtime Deployment

## Desktop Application Deployment

## System Requirements

2008 Sun Microsystems, Incorporated. All Rights Reserved.

# JavaFX Mobile Architecture




2008 Sun Microsystems, Incorporated. All Rights Reserved.

# New Features in JavaFX 1.2

- Significantly faster Desktop & browser start-up time
- Improved Mobile runtime performance
- Smaller compiled code size
- Streaming media benefits from faster start through RTSP
- Support for localization (Latin character sets)
- Support for reading RSS & ATOM feeds
- Early acces run time & tools for OpenSolaris & Linux
- Improved class libraries (Heads up! A few incompatible changes)

2008 Sun Microsystems, Incorporated. All Rights Reserved.

# Some Selected Language Features

- constants, variables
- objects and object literals
- sequences
- expressions
- binding, bound functions
- inheritance, mixin classes
- replace triggers

# Variables and Constants

```
def numOne = 100;
def numTwo = 2;
var result;

add();
subtract();
multiply();
divide();

function add() {
    result = numOne + numTwo;
    println("{numOne} + {numTwo} = {result}");
}
function subtract() {
    result = numOne - numTwo;
    println("{numOne} - {numTwo} = {result}");
}
```

key word: def

key word: var
a variable without assignment.
type interference!

# Objects and Object Literals

an object (instance)

```
Address {
    street: "Altrottstr. 31";
    city: "Walldorf";
    state: "BW";
    zip: "69190";
}
```

opening/closing braces required

instant assigment to a variable

```
def myAddress Address {
    street: "Altrottstr. 31";
    city: "Walldorf";
    state: "BW";
    zip: "69190";
}
```

instance variables

```
def customer = Customer {
    firstName: "Stefan";
    lastName: "Schneider";
    phoneNum: "(06227) 58235";
    address: Address {
        street: Altrottstr- 31";
        state: "BW";
        zip: "69190";
    }
}
```

nested objects

# Sequences

A sequence

```
def weekDays = ["Mon","Tue","Wed","Thu","Fri"];
def days = [weekDays, ["Sat","Sun"]];
def nums = [1..100];
```

A sequence declared within another

A shorthand notation

a predicate

```
def nums = [1,2,3,4,5];
def numsGreaterThanTwo = nums[n | n > 2];
def days = ["Mon","Tue","Wed","Thu","Fri","Sat","Sun"];

println(days[0]);
println(days[1]);
println(days[2]);
println(days[3]);
println(days[4]);
println(days[5]);
println(days[6]);
```

index starts with 0

numerical index in square brackets

# Expressions

```
var nums = [5, 7, 3, 9];
var total = {
     var sum = 0;
     for (a in nums) { sum += a };
     sum;
}
println("Total is {total}.");
```

```
Console output:
Total is 24.
```

# Binding

```
var myStreet = "1 Main Street";
var myCity = "Santa Clara";
var myState = "CA";
var myZip = "95050";

def address = bind Address {
     street: myStreet;
     city: myCity;
     state: myState;
     zip: myZip;
};

println("address.street == {address.street}");
myStreet = "100 Maple Street";
println("address.street == {address.street}");
```

use keyword „bind“here

address gets updated automatically

# Bound Functions

```
var scale = 1.0;
bound function makePoint(xPos : Number, yPos : Number) : Point {
     Point {
          x: xPos * scale
          y: yPos * scale
     }
}
class Point {
     var x : Number;
     var y : Number;}

var myX = 3.0;
var myY = 3.0;
def pt = bind makePoint(myX, myY);
println(pt.x);

myX = 10.0;
println(pt.x);

scale = 2.0;
println(pt.x);
```

use keyword „bound“here

new semantic takes effect here

```
Console output for code on
the left:
3.0
10.0
20.0
```

```
Console output without
„bound“ key word
3.0
10.0
10.0
```

# Inheritance

```
class CheckingAccount extends Account {

     var hasOverDraftProtection: Boolean;

     override function withdraw(amount: Number) : Void {
          if(balance-amount<0 and hasOverDraftProtection){
              // code to borrow money from an
             //overdraft account would go here

          } else { balance -= amount; }
    }
}
```

# Mixin Classes

```
def myContact = MyContact{};
myContact.printName();
myContact.printAddress();

mixin class MyNameMixin {
    var firstName = "John";
    var lastName = "Doe";
    function printName(){
        println("My name is: {firstName} {lastName}");
    }
}
mixin class MyAddressMixin {
    var address = "1 Main Street, Anytown USA";
    function printAddress(){
        println("My address is: {address}");
    }
}

class MyContact extends MyNameMixin, MyAddressMixin { }
```

use keyword „mixin“here

# Replace Triggers

```
var password = "foo" on replace oldValue {
     println("\nALERT! Password has changed!");
     println("Old Value: {oldValue}");
     println("New Value: {password}");
};
password = "bar";
```

```
Console output:
ALERT! Password has changed!
Old Value:
New Value: foo
ALERT! Password has changed!
Old Value: foo
New Value: bar
```

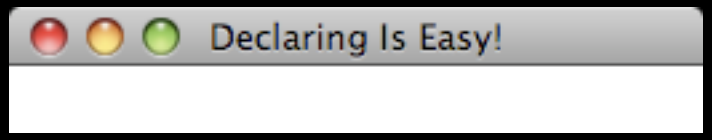
# Make it work: Building „Hello World“ Applications

- stages, nodes
- basic objects and transformations
- animation through binding and timelines
- user interaction

# Creating a Stage (A Window)

object literal

```
Stage {
title: "Declaring Is Easy!"
}
```




scene will hold nodes

```
Stage {
...
scene: Scene {
    width: 100
    height: 50
    content: [ ]
    }
}
```

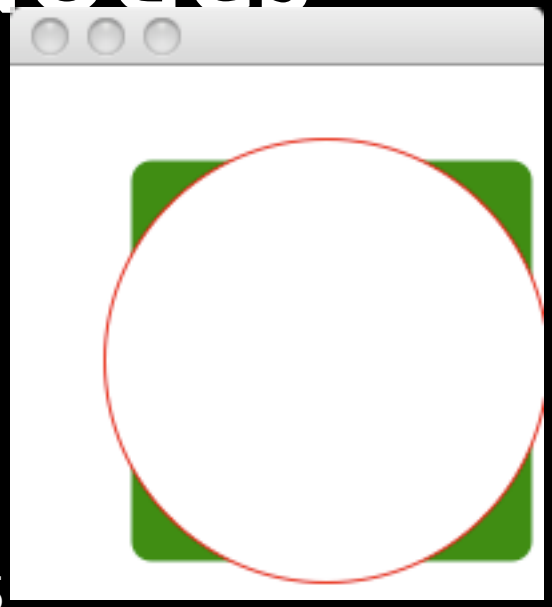
Stage Variables:
containsFocus, extensions, fullScreen, height, width, icons, onClose, opacity, resizable, scene, style, title, visible, x, y

Stage Functions:
close(), toBack(), to Front()

# A Scene with two Nodes

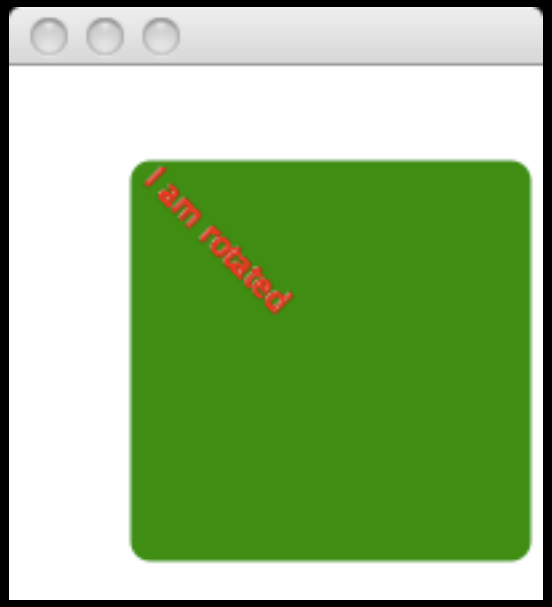
```
Stage {
...
scene: Scene {
    width: 200 height:200
    content: [ Rectangle {
                    x: 45 y: 35
                    width: 150, height: 150
                    arcWidth: 15, arcHeight: 15
                    fill: Color.GREEN
                    },
               Circle {
                    centerX: 118, centerY: 110 radius: 83
                    fill: Color.WHITE
                    stroke: Color.RED}
            ] }
}
```

comma is optional!



# Applying Transformations

```
Stage {
...
scene: Scene {
    width: 200 height:200
    content: [ Rectangle {
                    x: 45 y: 35
                    width: 150, height: 150
                    arcWidth: 15, arcHeight: 15
                    fill: Color.GREEN
                    },
               Text {
                    x: 45 y: 45
                    transforms: Transform.rotate(45, 50, 50)
                    content: "I am rotated"
                    fill: Color.WHITE
                    stroke: Color.RED}
            ] }
}
```



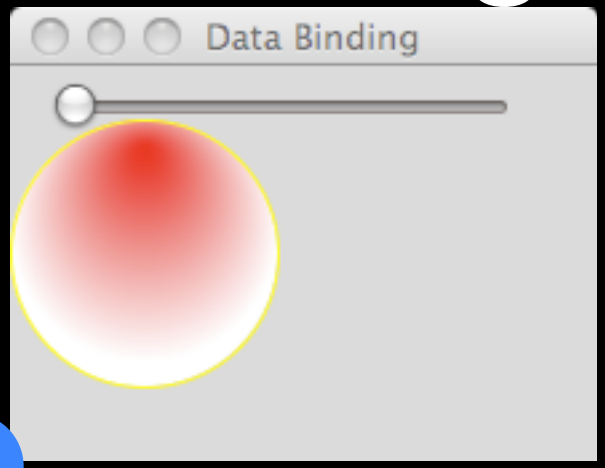

degrees, pivotA, pivotB

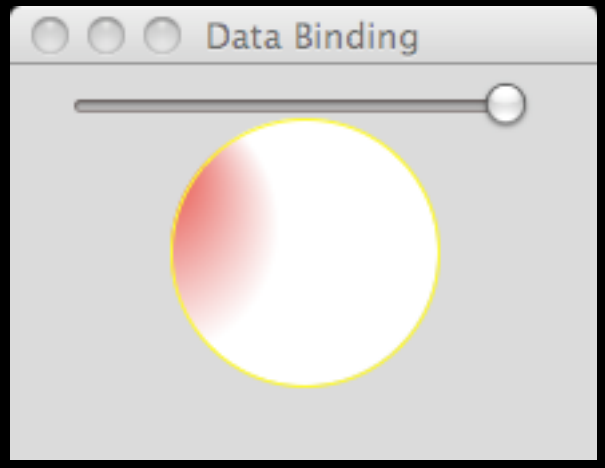
# Animation through Binding

```
var slider = SwingSlider{minimum: 0 maximum: 60
        value: 0 translateX: 10 translateY: 1}
Stage {
    title: "Data Bindi
    width: 220 height: 170
    scene: Scene {
        fill: Color.LIGHTGRAY;
        content: [
            slider,
            Circle {
                centerX: bind slider.value+50 centerY: 70 radius: 50 stroke: Color.YELLOW
                fill: RadialGradient {
                    centerX: 50 centerY: 60 radius: 50 focusX: 50 focusY: 30
                    proportional: false
                    stops: [
                    Stop {offset: 0 color: Color.RED},
                    Stop {offset: 1 color: Color.WHITE},
            }//Circle
        ]
        }//Scene
        }//Stage
```

Slider implementation

permanently update centerX



# Animated Objects The Timeline

```
var opacity = 1.0;
Timeline {
    repeatCount: Timeline.INDEFINITE
    keyFrames : [
        KeyFrame {
            time : 5s
            canSkip : true
                    values : [
                opacity => 0.2 tween Interpolator.LINEAR
            ]
        }
        KeyFrame {
            time : 10s
            canSkip : true
                    values : [
                opacity => 1.0 tween Interpolator.LINEAR
        ]
        }
    ]
}.play();
```

change variable opacity over time

Timeline first half:
change opacity within 5 seconds
from 1.0 to 0.2

Timeline second half:
change opacity within 10 seconds
back to 1.0

start execution right away

# Animated Objects: The Binding

```
Stage {
    title: "My First JavaFX Sphere" width: 300 height: 300
    scene: Scene {
        content: [
                Circle {
                    centerX: 100, centerY: 100 radius: 90
                    opacity: bind opacity
                    fill: RadialGradient {
                        centerX: 75 centerY: 75 radius: 90 proportional: false
                        stops: [
                            Stop { offset: 0.0 color: Color.web("#3B8DED") },
                            Stop { offset: 1.0 color: Color.web("#044EA4") }
                ] //stops
                }
        }
         Text {
            font : Font { size : 22 }
            x: 20, y: 90 textAlignment: TextAlignment.CENTER
            content: "Welcome tp \nJavaFX World" fill: Color.WHITE
        }       ]    }}
```

bind opacity variable


# User Interaction I

```
var opacity = 1.0;
var x = 100;
var y = 100;

Timeline {
...
}.play();

Stage {
    title: "My First JavaFX Sphere"
    scene: Scene {
        content: Group {
                content: [
                 Circle {
                     translateX: bind x translateY: bind y
                    radius: 90 opacity: bind opacity
                    fill: RadialGradient {
                            ... }
                        ...
                } //Circle
```

move the sphere to the position stored in these variables

bind x,y to node

inherited from javafx.scene.node
translate[x,y] coordinates of translation to be added to tranformed node

# User Interaction II

listen to mouse dragged

```
        Text {
            ....
                } //Text
        ] //content
                onMouseDragged: function( e: MouseEvent ):Void {
            x = e.x;
            y = e.y;
        }

    } //Group
} //Scene

}
```

pick coordinates from event and assign them to the variables x,y



See the live example

# STOP the Code Inspections!

- Most material has been taken from:
  - Examples: http://javafx.com/samples
  - Examples: MyFirstJavaFXSphere
    - http://javafx.com/docs/gettingstarted/javafx/create-first-javafx-app.jsp
  - Language tutorial: http://java.sun.com/javafx/1/tutorials/core/
  - GUI tutorial: http://java.sun.com/javafx/1/tutorials/ui/

# Time for Demos!