GENERAL

JavaFx

OVERVIEW
Agenda

- HISTORY
- SCENE GRAPH
- JAVA API
- PROPERTIES
- BINDINGS
- CONTROLS
- CSS
- WEBVIEW
- JFXPANEL
- CHARTS
Some
HISTORY
Roadmap

7u6
- JRE on Mac complete
- JavaFX 2.2 integration
- Linux ARM V6/V7
- JavaFX on Mac and Linux

Major Serviceability improvements
- Java Flight Recorder in JDK
- Native memory tracking
- Java Discovery Protocol
- App Stores Packaging tools
- Last Public Release of JDK 6

JDK 8
- Lambda
- Complete JVM Convergence
- JavaScript Interop
- JavaFX 8
  - Public UI Control API
  - Java SE Embedded support
  - Enhanced HTML5 support

JDK 9
- Jigsaw
- Interoperability
- Optimizations
- Cloud
- Ease of Use
- JavaFX JSR

2012
- NetBeans IDE 7.2
  - Support for JDK 7 on Mac
  - Support for JavaFX on Mac and Linux
- Scene Builder 1.0
  - Windows and Mac

2013
- NetBeans IDE 7.3
  - ARM/Linux support
  - Scene Builder 1.1 support
- Scene Builder 1.1
  - Linux support

2014
- NetBeans IDE 8
  - JDK 8 support
  - Scene Builder 2.0 support
- Scene Builder 2.0
  - JavaFX 8 support
  - Enhanced Java IDE support

2015
- NetBeans IDE 9
  - JDK 9 support
  - Scene Builder support
- Scene Builder 3.0
  - JavaFX support
What JavaFX really is...
It is the successor to Java Swing
and it’s still not finished
Available for

✴ WINDOWS
✴ MAC OS X
✴ LINUX
✴ ARM ✴
Available for

* APPLE IOS*

* ANDROID*
Versions

* JAVAFX 2.2 BUNDLED WITH JDK
  > JAVA 7U6
  > STANDALONE FOR JAVA6*

* WINDOWS ONLY
The architecture

JavaFX Public API’s and Scene Graph

Quantum Toolkit

Prism

Class Windowing Toolkit

Open CL

Media Engine

D3D

Web Engine

Java Virtual Machine
The architecture

Prism processes the rendering jobs.
The architecture

JavaFX Public API’s and Scene Graph

Quantum Toolkit

Prism

OpenGL

Java2D

D3D

Class Windowing Toolkit

Media Engine

Web Engine

Java Virtual Machine

OpenGL on Mac, Linux, Embedded
The architecture

JavaFX Public API’s and Scene Graph

Quantum Toolkit

Prism

Java2D

Open CL

D3D

Class Windowing Toolkit

Media Engine

Web Engine

Java Virtual Machine

DirectX 9 on Windows XP, Vista

DirectX 11 on Windows 7
Java2D when hardware acceleration is not possible
The architecture

JavaFX Public API’s and Scene Graph

Quantum Toolkit

Prism

Java2D

Open CL

D3D

Web Engine

Media Engine

Class Windowing Toolkit

Provides low level native operating system services
The architecture

Ties Prism and Glass Windowing Toolkit together and makes them available to the JavaFX layer above
The architecture
Open Source

JAVAFX SOURCE CODE IS PART OF THE OPEN JFX PROJECT

HTTP://OPENJDK.JAVA.NET/PROJECTS/OPENJFX/

nearly complete open source around 02/2013
Again a new PLUGIN
Browser Plugin

✴ Faster loading of JavaFX web apps based on Prism

✴ Pre-loader for improved user experience
The SCENE GRAPH
Collection of NODES
Scene Graph

- Handles the UI
- Tree structure
- Has one root node
- Branch + leaf nodes
Scene Graph

ROOT

BRANCH

LEAF
The only node without a parent node

THE ONLY NODE WITHOUT A PARENT NODE
Branch Nodes

- ARE DERIVED FROM javafx.scene.Parent
- CAN CONTAIN OTHER NODES
Leaf Nodes

- SHAPES
- IMAGES
- TEXT
- WEBVIEW
- MEDIA
- CONTROLS
- CHARTS
Leaf Nodes

* Have no
  getChildren()
The Nodes

Node (abstract)

Parent (abstract)

Group
  - non-resizable
Region
  - resizable + CSS stylish
  - Pane
Control (abstract)
  - resizable, skinnable + CSS stylish
    - TabPane
    - TitledPane
    - SplitPane
    - Accordian
    - ToolBar

StackPane

HBox

VBox

TilePane

FlowPane

AnchorPane

BorderPane

GridPane
In JavaFX

Stage

Branch Node

Leaf Node

Click me
Speed limit

60 FPS
Scene Graph

- Root node is a parent
- Stage is container for root
- Alive...no dead bitmaps
public class SceneGraphStructure extends Application {

    @Override public void start(Stage stage) {
        stage.setTitle("Hello World");
        Button button = new Button("Say 'Hello World'");
        button.setOnAction(new EventHandler<ActionEvent>() {
            @Override public void handle(ActionEvent evt) {
                System.out.println("Hello World");
            }
        });
        StackPane root = new StackPane();
        root.getChildren().add(button);
        stage.setScene(new Scene(root, 300, 250));
        stage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
JavaFx Script is NOT DEAD
It lives on as VISAGE
Now we have

PURE JAVA
Some examples
// Java FX 1.x
public def timer = Timeline {
  repeatCount: Timeline.INDEFINITE
  keyframes: KeyFrame {
    time: 1s
    action: function() {...}
  }
}

// Java FX 2.x
private Timeline timer =
  TimelineBuilder.create()
    .cycleCount(Timeline.INDEFINITE)
    .keyFrames(
      new KeyFrame(Duration.seconds(1),
               new EventHandler() {...}
    ))
    .build();
// Java FX 1.x
view = ImageView {
    image:image
    translateX:bind x + (view.scaleX - 1)
    translateY:bind y + (view.scaleY - 1)
};

// Java FX 2.x
view = new ImageView(image);
view.translateXProperty().bind(
    x.add(view.getScaleX() - 1));
view.translateYProperty().bind(
    y.add(view.getScaleY() - 1));
Properties and Bindings
// Property string
private static final String VALUE_PROPERTY = "value";

// A double property
private double value = 0;

// The getter method
public double getValue() {
    return value;
}

// The setter method
public void setValue(double newValue) {
    double oldValue = value;
    value = newValue;
    firePropertyChange(VALUE_PROPERTY, oldValue, value);
}
// A double property
private DoubleProperty value = new SimpleDoubleProperty(0);

// The getter method
public double getValue() {
    return value.get();
}

// The setter method
public void setValue(double newValue) {
    value.set(newValue);
}

// The property method
public DoubleProperty valueProperty() {
    return value;
}
// A double property
DoubleProperty value;

// The getter method
public double getValue() {
    return value.get();
}

// The setter method
public void setValue(double newValue) {
    value.set(newValue);
}

// The property method
public DoubleProperty valueProperty() {
    return value;
}
Bindings

- High-level binding
- Fluent API
- Bindings class
- Low-level binding
Bindings

* UNIDIRECTIONAL BINDING
  bind();

* BIDIRECTIONAL BINDING
  bindBidirectional();
IntegerProperty number1 = new SimpleIntegerProperty(1);
IntegerProperty number2 = new SimpleIntegerProperty(2);
DoubleProperty number3 = new SimpleDoubleProperty(0.5);

// High-Level Binding (Fluent API)
NumberBinding sum1 = number1.add(number2);
NumberBinding result1 = number1.add(number2).multiply(number3);

// High-Level Binding (Binding class)
NumberBinding sum2 = Bindings.add(number1, number2);
NumberBinding result2 = Bindings.add(number1, multiply(number2, number3));
High-Level

- Fluent API is self-explaining
- More readable code
- Might be a bit slower
IntegerProperty number1 = new SimpleIntegerProperty(1);
IntegerProperty number2 = new SimpleIntegerProperty(2);
DoubleProperty number3 = new SimpleDoubleProperty(0.5);

// Low-Level Binding
DoubleBinding db = new DoubleBinding() {
    {
        super.bind(number1, number2, number3);
    }

    @Override protected double computeValue() {
        return (number1.get() + number2.get() * number3.get());
    }
};
Low-Level

- Overrides a binding class
- Is more flexible
- Could be faster
Some examples
A simple label with a graphic on the left.

Hyperlink with Image

Left Button  Center Button  Right Button

Make a choice...
Option 1
Option 2
Option 3
Option 4
Option 5
Option 6
Longer ComboBox item
Option 7

Simple checkbox
Three state checkbox
Disabled
Control structure

* CONTROL
* SKIN
* BEHAVIOR
* CSS
CSS \rightarrow Control \rightarrow Skin \rightarrow Behavior
Styling with CSS
Remember

LOOK + FEELS

in Swing?
Forget them...
Using CSS

- **ONE DEFAULT CSS CASPIAN.CSS FOR ROOT AND CONTROLS**

- **JAVAFX CSS IS BASED ON W3C CSS 2.1 + SOME ADDITIONS**
Using CSS

* Either override the defaults to style your app
* Or apply your own CSS file
.root {
    -fx-base                  : #d0d0d0;
    -fx-background            : #f4f4f4;
    -fx-color                 : -fx-base;
    -fx-hover-base            : ladder(-fx-base, 
                                derive(-fx-base, 20%) 20%,
                                derive(-fx-base, 30%) 35%,
                                derive(-fx-base, 40%) 50%);
    -fx-pressed-base          : derive(-fx-base, -20%);
    -fx-focused-base          : -fx-base;
    -fx-body-color            : linear-gradient(to bottom,  
                                derive(-fx-color, 34%) 0%,
                                derive(-fx-color, -18%) 100%);
    ...
}
.button {
  -fx-skin: "com.sun.javafx.scene.control.skin.ButtonSkin";
  -fx-background-color: -fx-shadow-highlight-color, -fx-outer-border,
                      -fx-inner-border, -fx-body-color;
  -fx-background-insets: 0 0 -1 0, 0, 1, 2;
  -fx-background-radius: 5, 5, 4, 3;
  -fx-padding: 0.166667em 0.833333em 0.25em 0.833333em;
  -fx-text-fill: -fx-text-base-color;
  -fx-alignment: CENTER;
  -fx-content-display: LEFT;
}
The custom CSS

```css
.root {
    -fx-base: #252525; /* scene.getRoot().setStyle("-fx-base: #252525"); */
}

.button {
    -fx-font-family : "Verdana";
    -fx-font-size : 16px;
    -fx-background-radius: 9, 9, 8, 7;
    -fx-padding : 9px 16px 9px 16px;
}
```
A simple app
Caspian Styler
Scene scene = new Scene(pane, Color.rgb(75, 75, 75));
scene.getStylesheets().add("file:///customStylesheet.css");
A simple app
WebView and WEBENGINE
SCENE

WEBVIEW

WEWEBENGINE

WebKit
WebView

- Extension of Node
- Encapsulates WebEngine
- Incorporates HTML into the Scene
WebEngine

* PROVIDES WEBPAGE FUNCTION
* SUPPORTS USER INTERACTION
* ENABLES DOM ACCESS AND JS
stage.setTitle("WebView");

WebView browser = new WebView();
WebEngine engine = browser.getEngine();
engine.load("http://harmonic-code.org");

StackPane pane = new StackPane();
pane.getChildren().add(browser);
stage.setScene(new Scene(pane, 980, 720));
stage.show();
SteelSeries 3.9.30 released and moved to github

This is just a short info on the SteelSeries Java Swing library:

I moved the SteelSeries Swing library from project Kenai to github to have all projects in one place. Because I was working on it anyway I also created another major release which mainly contains some bugfixes (nothing special). In addition I have added the possibility to switch off the Lcd background and the possibility to blink the Lcd text (both features have been requested by users).

So if you would like to get the latest source code you should from now on take the code from the github repo and also issues should be filed on github instead of project Kenai.

Cheers and keep coding...
What about INTERACTION
How JavaFx INTERACTS with Html5
<head>
  <title>MyPage</title>
  <script type="text/javascript">
    var gauge;
    ...
  </script>
</head>

...
Interaction

WebView browser = new WebView();
WebEngine engine = browser.getEngine();
engine.load("http://mypage.html");

// JavaFX interact with WebView
engine.executeScript("gauge.setValue(5)");
How Html5 interacts with JavaFx
Listen to DOM EVENTS
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>MyPage</title>
</head>
<body>
  <button id="buttonId">Click me</button>
</body>
</html>
webView interact with JavaFX (Part I: Document Events)

```java
engine.getLoadWorker().stateProperty().addListener(new ChangeListener<State>() {
    @Override
    public void changed(ObservableValue<? extends State> ov, State old, State now) {
        if (newState == State.SUCCEEDED) {
            Document doc = (Document) engine.executeScript("document");
            EventTarget button = (EventTarget) doc.getElementById("buttonId");
            button.addEventListener("click", new DocEventListener(), true);
        }
    }
});

private static class DocEventListener implements EventListener {
    @Override
    public void handleEvent(Event event) {
        System.out.println("Event received: " + event.getType());
    }
}
```
Listen to STATUS
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>MyPage</title>
<script type="text/javascript">
    function setStatus(id) {
        window.status = id;
    }
</script>
</head>
<body>
<button id="buttonId">Click me</button>
</body>
</html>
// WebView interact with JavaFX (Part II: via window.status)
engine.getLoadWorker().stateProperty().addListener(new ChangeListener<State>() {
    @Override
    public void changed(ObservableValue<? extends State> ov, State old, State now) {
        if (newState == State.SUCCEEDED) {
            engine.setOnStatusChanged(new EventHandler<WebEvent<String>>() {
                @Override
                public void handle(WebEvent<String> event) {
                    // Get the window.status value
                    System.out.println("Status value: " + event.getData());
                }
            });
        }
    }
});
Inject a

JSOBJECT
class Bridge {
    public void exit() {
        Platform.exit();
    }
}

// Inject the JSObject with the name "java" into the html page
JSObject jsobj = (JSObject) webEngine.executeScript("window");
jsobj.setMember("java", new Bridge());

// Remove the JSObject again
JSObject.end();
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Close JavaFX from HTML</title>
  <script type="text/javascript">
    function closeJavaFXProgram() {
      java.exit();
    }
  </script>
</head>
<body>
  <button onclick="closeJavaFXProgram()">Close Java</button>
</body>
</html>
Migrating with JFXPanel
What is it?
* Behaves like JPanel
* Hosts a JavaFX Scene
* Forwards events
* Should be accessed from the Edit
How does it work?
// Add a JFXPanel to a Swing JFrame
JFrame frame = new JFrame("JFXPanel");
JFXPanel fxPanel = new JFXPanel();
frame.add(fxPanel);

Platform.runLater(new Runnable() {
    @Override public void run() {
        initFX(fxPanel);
    }
});
// Initialize the JFXPanel
void initFX(JFXPanel fxPanel) {
    // Code to create a JavaFX scene
    ...

    fxPanel.setScene(scene);
}
So you could use **JAVA FX** in Swing...
...means also HTML 5 in Swing
But keep in mind
You have 2 UI-THREADS
It's up to you to synchronize them manually.
JavaFx

CHARTS
JavaFX Charts

* CAN BE ANIMATED
* CAN BEStyled USING CSS
* CAN BE CUSTOMIZED
@Override public void start(Stage stage) {
    Scene scene = new Scene(new Group(), 500, 500);
    stage.setTitle("Imported fruits");

    ObservableList<PieChart.Data> pieChartData =
        FXCollections.observableArrayList(
            new PieChart.Data("Grapefruit", 13),
            new PieChart.Data("Oranges", 25),
            new PieChart.Data("Plums", 10),
            new PieChart.Data("Pears", 22),
            new PieChart.Data("Apples", 30));
    final PieChart chart = new PieChart(pieChartData);
    chart.setTitle("Imported Fruits");

    ((Group) scene.getRoot()).getChildren().add(chart);
    stage.setScene(scene);
    stage.show();
}
Need more CONTROLS?
here you go

JFXTRAS
Some
EXAMPLES
YOU WANNA BE PART OF
THE PARTY?
WE WANT YOU AT JFXTRAS
What’s new in
JAVA FX 8
JavaFx 8

- Support for Embedded
- 3D Support
- Swing-Node (Hopefully)
- Public API for Controls
- Performance++
- No focus on plugin anymore
Keep coding...